

**PONTIFICAL CATHOLIC UNIVERSITY OF PARANÁ
POLYTECHNIC SCHOOL
INDUSTRIAL AND SYSTEMS ENGINEERING GRADUATE PROGRAM**

RICARDO LUHM SILVA

**TOWARDS COGNITIVE MACHINE VISION FOR 2D IMAGE BASED
INDUSTRIAL INSPECTIONS: AN IMPLEMENTATION GUIDELINE
MODEL**

CURITIBA

2019

RICARDO LUHM SILVA

**TOWARDS COGNITIVE MACHINE VISION FOR 2D IMAGE BASED
INDUSTRIAL INSPECTIONS: AN IMPLEMENTATION GUIDELINE
MODEL**

Dissertation document presented to the Industrial and Systems Engineering Graduate Program, from Pontifical Catholic University of Paraná, as partial requirement for the Master's degree in Industrial and Systems Engineering.

Advisor: Marcelo Rudek

CURITIBA

2019

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná
Sistema Integrado de Bibliotecas – SIBI/PUCPR

Silva, Ricardo Luhm
S586t Towards cognitive machine vision for 2D image based industrial inspections :
2019 an implementation guideline model / ; orientador, *Marcelo Rudek*. -- 2019
122 f. : il. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Paraná, Curitiba,
2019.
Bibliografia: f. 116-122

1. Engenharia da produção. 2. Inteligência artificial. 3. Indústrias - Inovações
tecnológicas. I. Rudek, Marcelo. II. Pontifícia Universidade Católica do Paraná.
Programa de Pós-Graduação em Engenharia de Produção e Sistemas.
III. Título.

CDD 20. ed. – 670

Biblioteca Central
Edilene de Oliveira dos Santos CRB/9 1636

TERMO DE APROVAÇÃO

Ricardo Luhm Silva

TOWARDS COGNITIVE MACHINE VISION FOR 2D IMAGE BASED INDUSTRIAL INSPECTIONS: AN IMPLEMENTATION GUIDELINE MODEL.

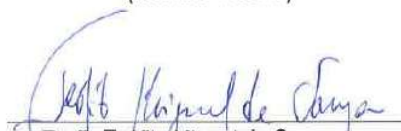
Dissertação aprovada como requisito parcial para obtenção do grau de Mestre no Curso de Mestrado em Engenharia de Produção e Sistemas, Programa de Pós-Graduação em Engenharia de Produção e Sistemas, da Escola Politécnica da Pontifícia Universidade Católica do Paraná, pela seguinte banca examinadora:



Presidente da Banca
Prof. Dr. Marcelo Rudek
(Orientador)



Prof. Dr. Osiris Canciglieri Junior
(Membro Interno)



Prof. Teófilo Miguel de Souza
(Membro Externo / PPGE/UNESP)

Curitiba, 13 de junho de 2019.

ACKNOWLEDGMENT

First of all, I would like to thank my wife Aline Bardini, for the everlasting patience and friendship. She helped in the moments of despair, where I could not believe in myself. I also would like to thank my mom Karin and my dad Nelson for the good conversation time about science, politics and any other different subjects that helped me to refresh my energies when I was depleted from the immersion in the dissertation.

I would like to thank my thesis advisor prof. Marcelo Rudek for his guidance and patience. He steered me in the right the direction whenever I was in the wrong direction. I also would like to thank prof. Osiris Canciglieri for our first talks about the master degree and for the initial opportunity in this research.

I would like to thank all my colleagues, Cassiano Beller, Guinther Kowalski, Muriel Mazzeto, Matheus Maziero, Ricardo Massao, and Weverton Estalk which participated directly or indirectly in my master degree daily activities, through discussions, ideas and small talks.

ABSTRACT

Machine Vision Systems (MVS) is widely used for industrial quality control. Artificial Intelligence (AI) solutions integrated with MVS has the potential to improve the existing solution and create newer solutions which computer vision techniques alone could not achieve. Contributions to AI applied for computer vision have been increasing over the years, but the knowledge and skills required to implement them are not easily available for industries. Industries must know which the main requisites of an AI MVS solution are to avoid an incorrect implementation. This research evaluates the current status of MVS and which can are their current limitations. It can be highlighted that the contribution of this document A systematic review and content analysis potential is performed to identify which are the main contribution and limitations of the current AI technologies to MVS, which are the newer technologies still to be implemented and what are the requirements needed to implement in real cases. An implementation model for AI MVS for industrial inspection is proposed considering the best practices found in the literature and from practical experiments performed during the validation of the model. The proposed implementation model alongside with the extraction of the state of the art techniques of AI MVS are the main contribution of this document. The proposed model can be used by industries as a first step to structure their AI MVS technologies up to the moment, and for researchers to develop newer solution considering some industrial requirements allowing them to evolve into cognitive inspection systems.

Keywords: Industrial Inspection, Artificial Intelligence Machine Vision System, Deep Learning, Implementation Model.

RESUMO

Sistemas de Visão de Máquina (MVS) são amplamente utilizados no controle de qualidade industrial. As soluções de Inteligência Artificial (AI) integradas à MVS têm o potencial de melhorar as soluções existentes e criar novas aplicações onde as técnicas de visão computacional sozinhas não conseguiram. As contribuições acadêmicas para AI aplicada à visão computacional têm aumentado ao longo dos anos, entretanto o conhecimento e as habilidades necessárias para implementá-las não são facilmente disponibilizados para as indústrias. As indústrias precisam saber quais são os principais requisitos de uma solução de AI MVS para evitar uma implementação incorreta. Esta pesquisa avaliou o status atual de MVS e quais são as suas limitações atuais. Foi realizado, por meio de uma revisão sistemática e um de análise de conteúdo, a identificação de quais são as principais contribuições e limitações atuais das tecnologias de IA para MVS, quais são as tecnologias mais recentes ainda a serem validadas e quais são os requisitos necessários para que estas sejam implementadas em casos reais. Um modelo de implementação de AI MVS para inspeção industrial foi proposto considerando as melhores práticas encontradas na literatura, e a partir de experimentos práticos realizados durante a validação do modelo. O modelo de implementação em conjunto com a extração das informações mais relevantes sobre AI MVS são as principais contribuições desse documento. O modelo proposto pode ser usado pelas indústrias como um primeiro passo para estruturar suas tecnologias de IA existentes para MVS até o momento, e também para que pesquisadores possam desenvolver novas soluções, considerando requisitos industriais e permitindo que elas evoluam para sistemas mais completos de inspeção cognitiva.

Palavras-chave: Industrial Inspection, Inteligência Artificial, Sistemas de Visão de Máquina, Deep Learning, Implementation Model.

LIST OF FIGURES

Figure 1 - Simplified workflow for visual quality inspections.....	17
Figure 2 - A block diagram for a typical vision system operation and applicable AI techniques.....	20
Figure 3 - Conventional MVS types and detection modes.....	21
Figure 4 - MVS integrated with a simplified 4.0 Industry diagram.....	24
Figure 5 - Complexity examples of AI models: Simple reflex agent and the utility-based agent.....	25
Figure 6 - Comparison of machine learning and deep learning.....	26
Figure 7 - A simplified neural network with a single hidden layer.....	27
Figure 8 - Example of mathematical operations between the input layer and the hidden layer.....	28
Figure 9 - Example of mathematical operations between the hidden layer and the output layer.....	28
Figure 10 - Research methodology steps performed in this study.....	31
Figure 11 - Proposed field groups correlation.....	32
Figure 12 - The number of publications between 2006 and 2018 distributed into the research group combinations.....	36
Figure 13 - Filtering criteria for title reading.....	36
Figure 14 - Number of publications according to each technique group.....	38
Figure 15 - Review and Survey number of publications divided by M/M-F-A (a) and divided by technology (b).....	38
Figure 16 - The proposed evaluation model for complete title reading divided by subgroups.....	40
Figure 17 - AI contribution (a) and MVS solution type (b) results.....	43
Figure 18 - Supervision Type and data source results.....	43
Figure 19 - Dataset improvement results.....	44
Figure 20 - Classification (a) and Localization (b) metrics.....	44
Figure 21 - Time to train (a) and deployed solution performance (b) metrics results.....	45
Figure 22 - Presence of Workflow (a) and Layer Parameters (b) results.....	45
Figure 23 - Presence of training hardware (a) and embedded hardware (b).....	46
Figure 24 - The proposed AI MVS implementation model divided into the main group and its subcategories.....	53
Figure 25 - IIR group with its main activities and simplified descriptions.....	54
Figure 26 - ASA group with its main activities and simplified descriptions.....	60
Figure 27 - Correlation between the main components for AI programming.....	69
Figure 28 - Confusion Matrix Example.....	71

Figure 29 - Relationship between the model output value and the confusion matrix given a threshold value.	71
Figure 30 - The four types of possibilities of a confusion matrix given threshold value.	72
Figure 31 - Intersect of Union localization metric.....	74
Figure 32 - CNN specific details group with its main activities and simplified descriptions.	76
Figure 33 - Dataset division smaller parts for training and model evaluation.	79
Figure 34 - Example of a generic kernel applied over an image input and its respective feature map.....	80
Figure 35 - Example the mathematical operation of the kernel over the input and the first resulting value.	81
Figure 36 - Example the mathematical operation of the kernel over the input given a stride of one and the next, resulting value.....	81
Figure 37 - Example the complete mathematical operation of the kernel over the input and all the resulting values.	82
Figure 38 - Example of Valid Padding with different stride values.....	83
Figure 39 - Example of Same Padding with stride value equals two.	84
Figure 40 - Example of stacked feature maps gives an image input and a pack of convolution filters.....	85
Figure 41 - Example of max and average pooling.....	88
Figure 42 - Localization of the pooling layer and its relationship with the stacked feature maps. Source: The Author.....	89
Figure 43 - Feature maps being fed into the input layer and their relationship with a single neuron in the hidden layer.....	90
Figure 44 - Relationship of the fully connected layer output and the classifier layer..	91
Figure 45 - Transformation of the label input into a one-hot encoding form.	92
Figure 46 - Relationship of the label input and the classification output and its one hot encoding vector.	93
Figure 47 - Example of stochastic learning on CNN.....	94
Figure 48 - Differences between stochastic learning and batch learning regarding memory consumption.	94
Figure 49 - Example of full batch learning in CNN.....	95
Figure 50 - Example of mini-batch learning on CNN.	95
Figure 51 - Error vs. Epoch curve for full batch and mini-batch behavior comparison.	96
Figure 52 - Error vs. Epoch graph with a model that already achieved convergence.	96
Figure 53 - Example of the effects of different learning rate values.....	97

Figure 54 - Example of different learning rate values in the loss/error vs. epoch curve.	97
Figure 55 - AI-MNGT group with its main activities and simplified descriptions.....	99
Figure 56 - Example of the extraction of relevant data in the proposed transfer learning technique.....	105
Figure 57 - AI MVS Management Platform main workflow and its functionalities. ...	106

LIST OF TABLES

Table 1 - Keywords used as the inclusion filter divided into the research groups.....	32
Table 2 - Inclusion and Exclusion criteria for the first step of the systematic review. .	34
Table 3 - Output datasheet format and relevant columns used in the next steps.	35
Table 4 - AI techniques divided by group.....	37
Table 5 - Selection and Exclusion Criteria	39
Table 6 - Filtering steps between title selection to abstract reading resume.....	39
Table 7 - Detailed criteria for each subgroup in the proposed evaluation model.	40
Table 8 - Detailed fields of how the papers were evaluated.	42
Table 9 - Content Analysis Resume.....	47
Table 10 - Example of commonly activation function for image-based solutions.	86
Table 11- Summary of the implementation steps of the Robotic Inspection MVS....	109
Table 12- Summary of the implementation steps of the Non-AI MVS Prototype.....	111
Table 13 - Summary of the implementation steps of the First AI MVS Prototype.....	113
Table 14 - Summary of the implementation steps of the Final AI MVS Prototype....	115

LIST OF ACRONYMS

1D	One Dimensional
2D	Two Dimensional
3D	Three Dimensional
ACU	Active Convolution Unit
AdaBoost	Adaptive Boosting
AFRS	Automatic Fault Recognition System
AI	Artificial Intelligence
ANN	Artificial Neural Networks
ASIC	Application-specific integrated circuit
BLAS	Basic Linear Algebra Subprograms
CM	Confusion Matrix
CNN	Convolutional Neural Network
CPS	Cyber-Physical Systems
CPU	Computer Processing Units
DBN	Deep Belief Neural Network
DL	Deep Learning
DSL	Deep Super Learner
ELM	Extreme Machine Learning
FN	False Negative
FP	False Positive
FPS	Frames per Second
GPU	Graphic Processing Units
HMI	Human-Machine Interface
HOG	Histogram of Gradients
ILSVRC	ImageNet Large Scale Visual Recognition Competition
IoT	Internet of Things
IQC	Industrial Quality Control
ISSN	International Standard Serial Number
KPI	Key Performance Indicator
LDA	Linear discriminant analysis
ML	Machine Learning
MLP	Multi-Layer Perceptron
MV	Machine Vision

MVS Machine Vision System
NAS Neural Architecture Search
NN Neural Network
ORB Orientated fast and Rotated Brief
PCA Principal Component Analysis
PHOG Pyramid Histogram of Gradients
P-TELU Parametric Tan Hyperbolic Linear Unit
PU Processing Units
RBM Restricted Boltzmann Machine
R-CNN Region Based Convolutional Neural Network
ReLU Rectified Linear Unit
RGB Red, Green and Blue Color channels
RIS Research Information Systems
ROC Receiver operating characteristics
RorS Review or Survey
RPN Region Proposal Network
SAE Sparse Autoencoder
SIFT Scale Invariant Feature Transform
SPP Spatial Pyramid Pooling
SSD Single Shot MultiBox Detector
SURF Speeded Up Robust Features
SVM Support Vector Machine
TN True Negative
TP True Positive
TPU Tensor processing units
UOI Union of Intersects
VAE Variational autoencoders
YOLO You only look once

SUMMARY

1	INTRODUCTION	16
1.1	Problem.....	18
1.2	Objective.....	19
1.2.1	Specific Objectives	19
2	BACKGROUND.....	19
2.1	IQC	19
2.2	Machine Vision System (MVS)	20
2.2.1	MVS core functions	20
2.2.2	1D and 2D MVS industrial applications.....	21
2.2.3	3D MVS industrial applications.....	22
2.2.4	MVS industrial evaluation	23
2.2.5	MVS applications to Industry 4.0	23
2.3	AI techniques.	24
2.3.1	Traditional ML and Deep Neural Networks for MVS	25
2.3.2	Simplified AI learning concepts.....	27
3	RESEARCH METHODOLOGY	30
3.1	Systematic Review.....	31
3.2	Content Analysis.....	39
3.3	Content Analysis Contributions	46
4	PROPOSED MODEL.....	52
4.1	AI MVS Implementation Model.....	52
4.1.1	Industrial inspection requirements (IIR).	53
IIR-1.	Quality Control Assessment.....	54
IIR-2.	Scene Constraints Assessment	55
IIR-3.	Technical Constraints Assessment.....	57
4.1.2	Artificial Intelligence Assessment (ASA).	59
ASA-4.	Solution type selection	61
ASA-5.	AI technique for MVS.....	62
ASA-6.	AI training requisites	64
ASA-7.	AI Hyperparameters.....	66
ASA-8.	Hardware/Software for AI.....	66

ASA-9. Validation Metrics	70
4.1.3 CNN specific details (CNN)	75
CNN-10. Dataset	77
CNN-11. Convolution Operation.....	80
CNN-12. Activation Function and Normalization	86
CNN-13. Pooling Operation	87
CNN-14. Fully Connected Layer	89
CNN-15. Classifier.	91
CNN-16. AI Learning Method	93
4.1.4 AI MVS Management (AI-MNGT).....	98
AI-MNGT 17. User Profiles	100
AI-MNGT 18. AI Frameworks.....	102
AI-MNGT 19. AI Trained Models.....	103
AI-MNGT 20. AI MVS Management Platform.....	106
5 IMPLEMENTATION EXAMPLE OF THE PROPOSED FRAMEWORK	
MODEL 108	
5.1 First Implementation Scenario – Non-AI-MVS Robotic Arm.....	108
5.2 Second Implementation Scenario – Non-AI – MVS.....	110
5.3 Third Implementation Scenario – First AI MVS Prototype.....	112
5.4 Fourth Implementation Scenario – Final AI MVS Prototype.....	114
6 CONCLUSION.....	116
REFERENCES.....	118

1 INTRODUCTION

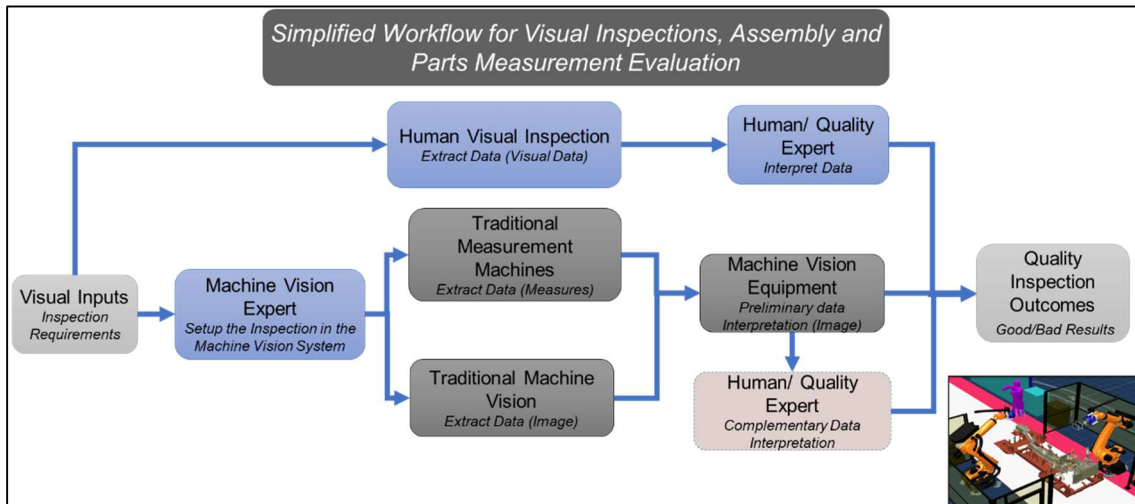
The intense competition in the global market creates an environment in which companies must be able to develop new products quickly with lower production costs and meet customer expectations. Resources must be invested in quality management to achieve customer expectations. Quality investment starts during the product development phase go through the product manufacturing and end with the perceived quality by the client. In a production line, it is possible to monitor quality during the execution of each process or at the end of it. According to Montgomery (2005), a process consists in transforming an input, which can be raw materials, components, or subassemblies into an intermediary or the final product. To perform process changes in the assembly line after the project of the product and the installed machinery are complete, it is usually a costly operation. Any process may be subject to quality problems, so a quality inspector or inspection equipment can be placed right after an operation is finished.

Visual inspection is an essential process in an industry to recognize defective parts, to assure quality conformity of a product and fulfill customer demands (Vergara-Villegas *et al.*, 2014), (Satorres Martínez *et al.*, 2012). Human inspectors can perform product and process inspection in assembly and manufacturing activities, but due to several factors (e.g., fatigue, small parts, small details, hazardous inspection conditions, process complexity), this task may not assure quality output. In this case, a machine vision solution is recommended (Vergara-Villegas *et al.*, 2014), (Szkilnyk, 2012), (Golnabi; Asadpour, 2007).

Automated visual inspections, also known as Machine Vision System (MVS) consist in applying computer vision to industrial solutions. It can operate with high-resolution cameras or sensors that can detect a broader range of light spectrum along with a special lens that amplifies the detection of small parts or defects. Most of automated MVS relies on controlled lighting environment with pre-built algorithms to detect specific types of defects (Pérez *et al.*, 2016). MVS must be adaptable to scenarios that have a wide variety of product features, high production speed assembly lines, and other complex environment variables (Labudzki *et al.*, 2014), (Lerones *et al.*, 2005).

Traditional MVS or trained quality inspectors can perform the quality inspection without Artificial Intelligence (AI). Figure 1 is an example of a Simplified workflow for visual quality inspections considering traditional machine vision systems and human inspector.

Figure 1 - Simplified workflow for visual quality inspections.



Source: The Author

A quality inspector must receive proper training to allow him to extract and interpret the inspection data. If a non-conformity is detected, the specialist can look for additional sources of knowledge to analyze the root cause of the problem. He can also search for related quality indicators to perform a more complex evaluation (Guo *et al.*, 2015). Besides human limitations such as fatigue, it is also required a considerable amount of time to train a specialized staff. The experience acquired by quality inspectors may be lost if any event displaces the employee from its current inspection activities, without a proper knowledge transfer. When a quality inspector does not meet the inspection requirements, an MVS is an alternative.

MVS requires a computer vision expert to identify the inspection requirements and perform its configuration. Existing MVS might not work correctly when manufacturing process parameters are submitted to process changes. System recalibration or to include a new inspection functionality requires well-trained staff, despite improvements of MVS user interface applications. This specialized worker must understand what is happening and perform the most appropriate fine-tuning task. The computer vision expert alone may require the assistance of a qualified expert to validate its reliability. Another drawback of traditional MVS is that it does not learn with

previous inspections data. Artificial Intelligence (AI) can improve MVS and add learning features.

AI can be used to enhance existing quality management tools and methods through state-of-the-art technologies. It may be applied to improve detection, provide data analysis with potential insights problem analysis, and create predictive models to avoid future quality problems. These improvements allow factories to walk toward the Smart Factories and Cyber-Physical System concepts, which belongs to the core concepts of the fourth industrial revolution or industry 4.0 (Foidl; Felderer, 2016). Although AI solutions can provide relevant improvements to product quality, its implementation may be challenging for several companies. They must face an adaptative process which requires technological and financial resources to integrate these new features. Workers must have minimum skills to be able to interact with these technologies to allow the transformation process that comes with it.

Recent advances on computer vision techniques combined with AI techniques, such as Machine Learning (ML) and Deep Learning (DL), shows potential applications to improve existing MVS to be less sensitive to external factors, increasing the capacity to detect image features and allows machines to learn from existing data.

1.1 Problem

AI applied to MVS, and Industrial Quality Control (IQC) shows promising improvements, but there are technical constraints that must be understood in order to allow a correct implementation in an industrial scenario. Current AI MVS applications are commonly tailor-made. Each solution has specific detection needs with variable image input parameters making it difficult to provide a standard Key Performance Indicator (KPI) to evaluate the system performance or its replicability in any other industrial scenario. An increased number of publications considering MVS and Machine Learning (ML) and Deep Learning (DL) solutions were previously observed in the last ten years but selecting ML and DL solution it is still a challenge from the industrialization perspective.

Industrialization and technical requisites must be considered to make this solution achieve a higher degree of product maturity (Silva *et al.*, 2018). It is required a combination of technology availability, classification assertiveness, detection precision, hardware robustness, trained staff in Computer Vision, and Artificial Intelligence algorithms. Companies which has research and development (R&D) as a

core business or a specialized department may have employees with the required skills and infrastructure to develop a customized solution. Companies which does not have R&D may have to wait for off-the-shelf solutions, hire specialized companies, or develop its own solution without the proper conditions. The lack of proper knowledge and know how to apply AI MVS into an existing quality inspection process can be subject to successive implementation failures, increased development cost, or inferior performance when compared to validated traditional solutions.

1.2 Objective

The objective of this research is to create ~~provide~~ an implementation model of AI MVS, that can perform better than traditional MVS and human-based inspections. The model was built based on the main contributions and points of development found in the systematic review and content analysis of state-of-the-art AI techniques. The model also considers observations made by the author during his time while working for automobile factory, composed of several assembly lines with existing MVS. This document also provides a background of 2D MVS main aspects, AI basic concepts and machine and deep learning techniques for 2D image system.

1.2.1 Specific Objectives

- Identify the current limitations of traditional MVS
- Identify technological requirements for AI solutions and its applications to MVS.
- Create an implementation model for AI MVS in industrial inspections
- Validate the using case models.

2 BACKGROUND

2.1 IQC

Industrial Quality control (IQC) is a management tool used to establish acceptable product specifications and characteristics limits and maintain the output production within control limits (Hitomi, 2017). There are several phases during product development and manufacturing, where quality control tools and concepts can be applied. Each one of these tools can be associated with a quality cost, which is one way to bring a qualitative evaluation or an indirect quantitative measure into measurable value.

Materials, components, subassemblies, and products inspections are related to quality appraisal cost, which aims to ensure the output meets its desired specifications. Assertive inspection projects tend to decrease internal failure costs, such as rework, factory downtime, and product downgrading. It also affects the external failure cost, which are related after the product leaves the industry, reducing complaints, warranty charges, and returned product costs. Improving the bottom line is the goal.

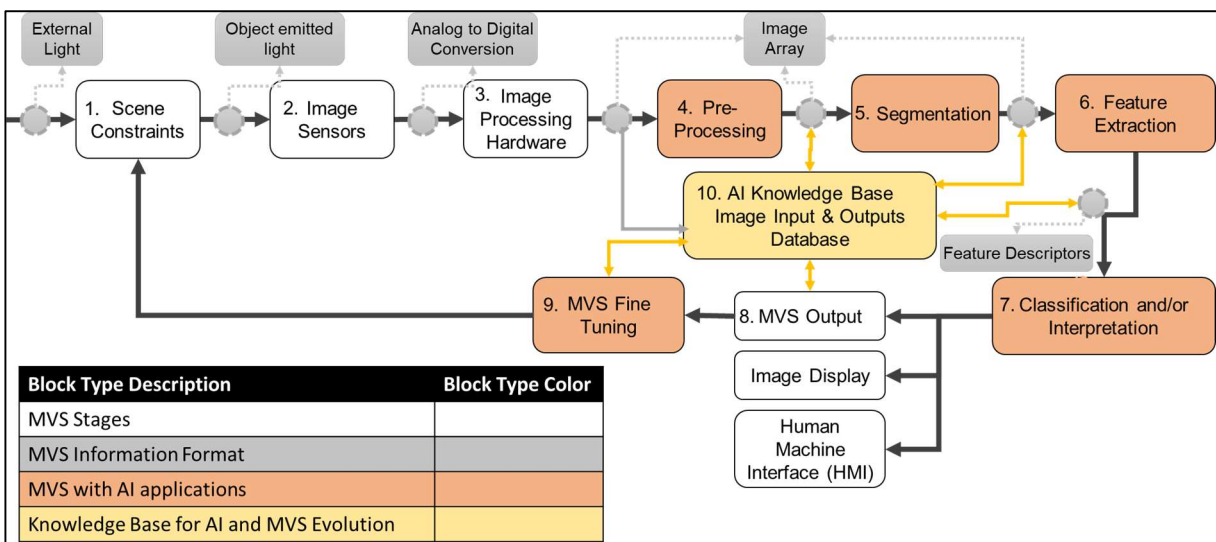
2.2 Machine Vision System (MVS)

A typical MVS or Computer Vision are composed of several tasks, which ranges from the image capturing or acquisition, image pre-processing and enhancement, image segmentation, feature extraction, classification and/or interpretation and actuation.

2.2.1 MVS core functions

Golbani and Asadpour (2007) proposed a block diagram for a typical vision system operation when artificial intelligence techniques were still being developed for MVS. According to Silva’s review, AI techniques in all steps ranging from image pre-processing to the MVS fine tuning. Figure 2 provides a diagram containing a modified diagram with new AI techniques identified in recent literature.

Figure 2 - A block diagram for a typical vision system operation and applicable AI techniques.



Source: Adapted from Wang, J. *et al.* (2018) Golnabi; Asadpour (2007) Vergara-Villegas *et al.* (2014)

In this framework is important to emphasize the need for a knowledge dataset. This dataset contains several inputs from gathered from all the steps within the MVS,

such as object features and quality criteria definitions, that may be used to improve the current learning methods (Wang, J. *et al.*, 2018), (Vergara-Villegas *et al.*, 2014).

2.2.2 1D and 2D MVS industrial applications

One (1D) and two dimensional (2D) MVS have a wide range of applications such as measurement, surface, and depth inspection, thermal inspection, and robot vision. Each kind of application has its own characteristic equipment with different image gathering source, such as photoelectric sensor, lasers, cameras, and so on (Chauhan; Surgenor, 2015). Some MVS source types and applications are shown in Figure 3.

Figure 3 - Conventional MVS types and detection modes.

CVS type	CONTROL TYPES	Detailed Description	EXAMPLE	
CAMERA BASED CVS	Presence/Absence	Check if the part present or absent in the checked spot		
	Orientation	Check if the part is placed in the right orientation		
	Position	Check if the part is in the right place or it is not missing		
	Color	Check the presence/absence of colors in the desired parts		
	Recognition/Content analysis	Check codes, letters, bar codes, RFID or any other kind of information.		
	Geometric Control	Check if the part has the desired dimensions and desired geometric tolerances		
PHOTOELECTRIC OR LASER BASED CVS	Presence/Absence	Check if the part present or absent in the checked spot		
	Position and Geom. Control	Check if the part has the desired dimensions and positioned in the right place		
	Color/Surface	Check the reflected color intensity		

Source: Adapted from Chauhan; Surgenor (2015), Vergara-Villegas *et al.* (2014)

Camera-based MVS for industrial applications is commonly used to verify presence or absence of components, verify if the components are in their correct position and orientation, verify if components have the desired colors, analyze and

recognize image content such as code bars and inspect size and measure of parts and assembly components (Golnabi; Asadpour, 2007).

Photoelectric and laser based MVS inspections can also be used to presence or absence of components, check component positioning, and verify desired colors, mainly to measure parts.

2.2.3 3D MVS industrial applications

The optical non-contact 3D measurement technique has been used to measure an image of an object and extract its geometrical information. It can be divided into passive and active 3D sensing systems, where passive works with natural lighting from the scene without controlling the light that goes to the inspected object. While active sensing systems use external light, such as laser or a known projected light, by measuring the speed of light, laser coherence, or applying triangulation techniques (Su; Zhang, 2010).

Structured light is an active 3D sensing system, which illuminates the object with predefined patterns and analyses how these patterns are deformed by the object when observed from a different angle of the projection. Some systems adopt non-visible structured light to avoid interfering with other computer vision (Van der Jeught; Dirckx, 2015).

Stereo vision profilometry techniques simulate human vision through two camera setups angled with each other, which aims to identify and match common features of object images from multiple allowing it to be reconstructed through triangulation techniques (Van der Jeught; Dirckx, 2015). Stereo vision normally is a passive 3-D sensing system, but there are new camera setups, which also uses a projected structured light in the object that turns them into active stereo vision (Pérez *et al.*, 2016).

Another active vision technique is a time of flight light measurement. This technique uses light pulses with a known camera range so the time for the emitted light to travel from the camera and hits the object and is reflected on the camera is measured, based a fixed and known light speed the distance can be calculated (Pérez *et al.*, 2016).

Light coding imaging is also an active 3D sensing system, but instead of using light pulses, it keeps light source constantly turned on. It also uses an infrared spectrum emitter and receiver, which analyses lens distortion, the emitted light pattern and the

distance between object, emitter, and receiver and the deformation of the light over the inspected object (Pérez *et al.*, 2016).

2.2.4 MVS industrial evaluation

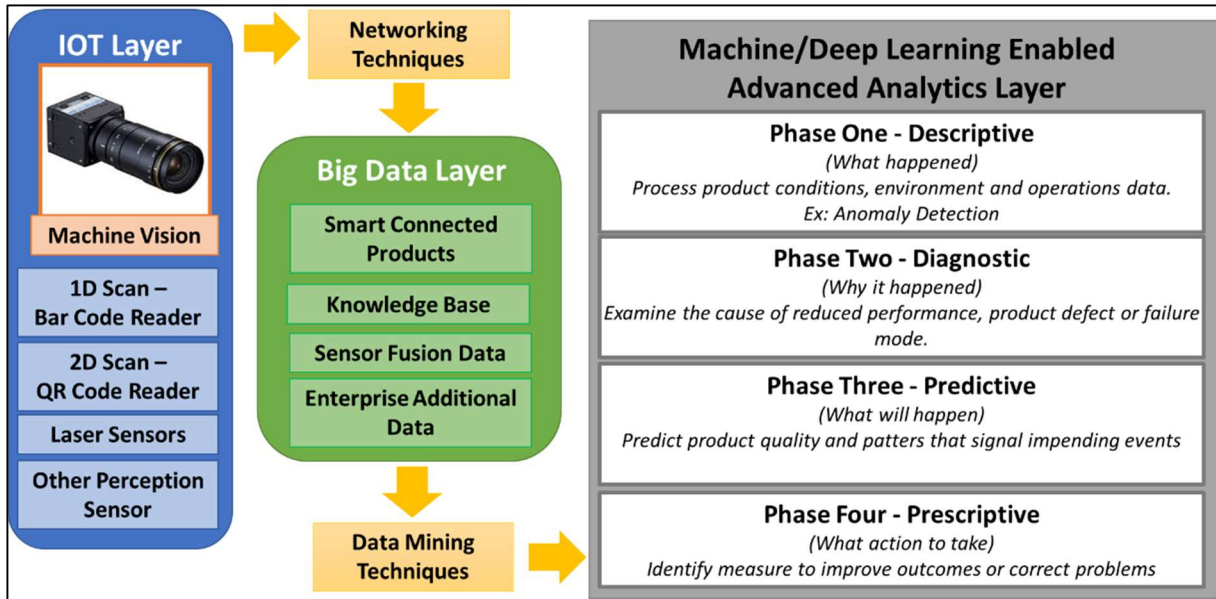
Pérez *et al.* (2016) compared several 3D machine vision techniques applied to industrial environments emphasizing which factors need to be considered in order to select the adequate vision application, considering, system accuracy, working distance, image output, system advantages, and limitations. Environment light influence was one of the major detected limitations to MVS. Only a few systems were not subject to external light negative influence. Some MVS may need the camera and the object to remain static so it can work properly. Detection accuracy and working distances are also important variables that must be taken into account.

2.2.5 MVS applications to Industry 4.0

The fourth industrial revolution or Industry 4.0 aims to develop intelligent factories with upgraded manufacturing technologies through new features such as cyber-physical system (CPS), Internet of Things (IoT), Big Data and Cloud computing. New manufacturing systems propose simultaneous monitoring of physical processes with being controlled by digital technologies, being able to make a smart decision through real-time communication and interaction between humans, machines, or any smart device (Zhong *et al.*, 2017)

Figure 4 contains a simplified 4.0 Industry diagram, adapted from more complex diagrams available in the literature. Machine vision is in the IoT layer. Its function is to provide image data through a connected network to a big data cloud server. This data will be subject to mining and cleaning procedures, removing unnecessary information. This information can be used as an input for machine learning techniques, allowing an integrated system to detect and describe what happened to the product, determining why that happened, predict what may happen and prescribe which actions must be taken (Pal *et al.*, 2016).

Figure 4 - MVS integrated with a simplified 4.0 Industry diagram.



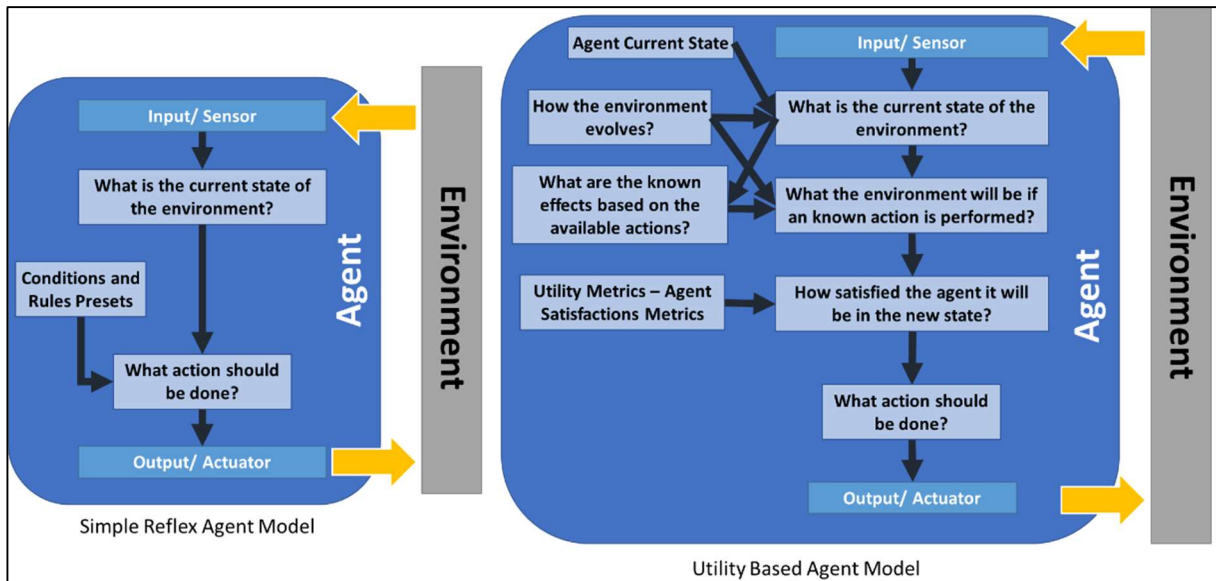
Source: Adapted from Pal *et al.* (2016) Zhong *et al.* (2017)

MVS future solutions, such as knowledge-driven decision-making, real-time control, online advanced analytics, and AI in CPS, are considered a challenging implementation process. It may take from 3 to 10 years for industries to achieve a concrete degree of maturity and obtain a fully operational system with these functionalities (Leitão *et al.*, 2016).

2.3 AI techniques.

AI is composed of several subfields which aim to understand, mimic, and improve how we think, perceive, understand, predict, and act upon an environment. That allows us to build intelligent entities to solve complex problems. Russell; Norvig (2010) defines an AI solution as a sentient agent which interacts with an environment being able to sense what is happening and what should be its output actions based on known functions. AI model with low complexity performs simple reflex interactions types with the environment. Complex models can evaluate which is the best action to be taken based on previous knowledge before performing it. The behavior of the agent can improve through learning mechanism, which allows the model to adapt to new scenarios. Figure 5 - Complexity examples of AI models: Simple reflex agent and the utility-based agent. shows an example of the simple reflex model and the utility-based model.

Figure 5 - Complexity examples of AI models: Simple reflex agent and the utility-based agent.



Source: Adapted from Russell; Norvig (2010)

There are several types of AI solutions which can work alone as an agent itself, or they can be combined to a provided hybrid solution with improved functionalities. Each AI solution has unique aspects and architectures. Machine learning (ML) is one of these possible solutions that may be applied to MVS. It uses the artificial neural network (ANN) architecture concepts, which creates an approximation of how our neurons interact with each other and allow it to extract representative information for decision making.

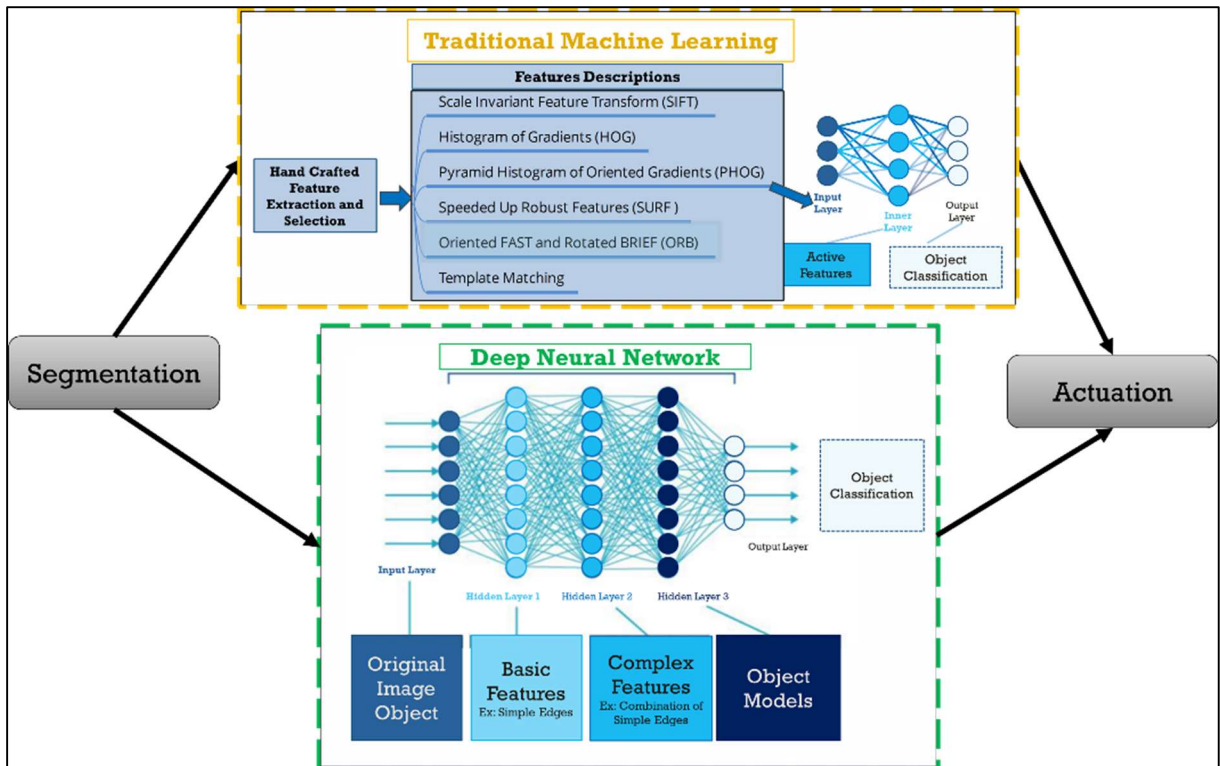
These neural networks (NN) are composed of three types of layers: the input layer, the inner or hidden layers, and the output layer. The input layer is responsible for receiving data and distributing to the first inner layer. Inner layers are responsible for processing and extracting the relevant information from the input layer or from a previous inner layer. The output layer receives the processed data from the inner layers and provides a relevant output.

2.3.1 Traditional ML and Deep Neural Networks for MVS

Machine learning (ML) and deep learning (DL) are data-driven artificial intelligence techniques, which may be applied to MVS. Both techniques use neural network architecture concepts, which transforms raw data into representative information for decision making. Figure 6 - Comparison of machine learning and deep learning.

Source: Adapted from Wang, J. *et al.* (2018) shows a simplified comparison of traditional machine learning and deep learning.

Figure 6 - Comparison of machine learning and deep learning.



Source: Adapted from Wang, J. *et al.* (2018)

Traditional Machine Learning uses different features extraction types to obtain features descriptors, which will be later used in a simple layer neural network. The traditional ML can be considered as a handcrafted process because it requires an image processing expert to select which feature extraction technique is being used as input for the neural network. DL is applied in both feature extraction and classification steps in a unified multi-layer neural network (NN). The neurons will select and extract the most relevant features and will evaluate their influence in the classification layer

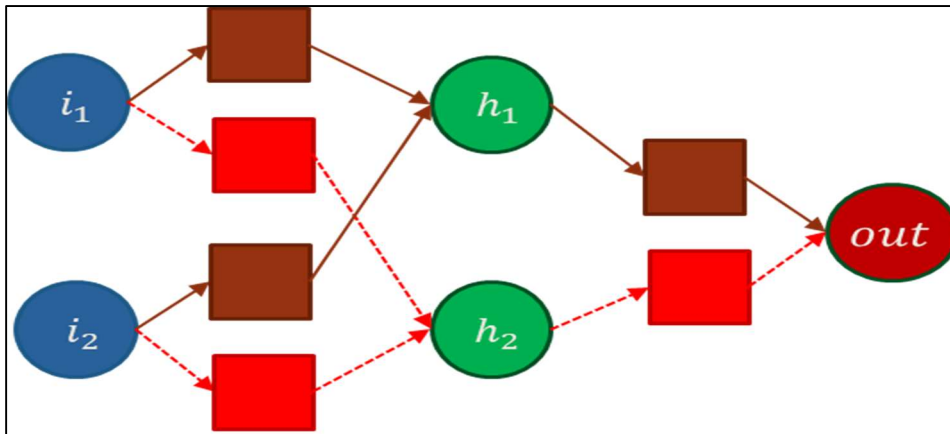
Deep learning techniques are within the ML field of study, and its main characteristic is the multiple inner layers that allow more complex information extraction, but it demands more computational processing power.

ML and DL techniques are currently being applied for two types of applications: classification and regression. Classification is used to analyze data and label into a different type of classes. Regression is used to analyze data and predict future values. MVS with ML or DL can be applied to IQC for inspections based on pattern recognition and classification to improve detection of non-conformities and consequently increases quality assurance. DL solutions initially require an AI programmer with MVS knowledge to train the system.

2.3.2 Simplified AI learning concepts.

AI learning concepts are based on a mathematical model which transform a given input data into a prediction value. Each neuron receives an input value, and it will be multiplied by a weight, and a bias addition is applied to the result. Figure 7 will be used as a simplified network considering two input values connected to a hidden layer with two neurons, and a single output neuron layer.

Figure 7 - A simplified neural network with a single hidden layer.



Source: The Author.

The circles represent the neurons while the squares represent the mathematical operations which happen between neurons. A simple neuron typically has at least two mathematical operations. First, are the weight multiplication and the bias addition. Equation 1 shows the first mathematical operation considering an input node.

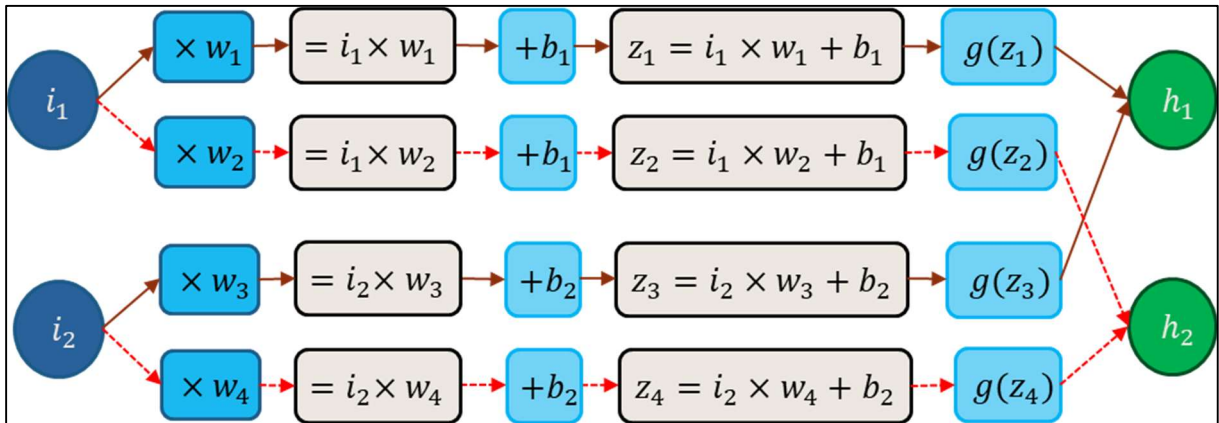
$$z_n = i_n * w_n + b = \text{input} * \text{weight} + \text{bias} \quad (1)$$

The second mathematical operation is the activation function, as shown in the Equation 2, which takes the input z_n and applies a function defined by the letter 'g'.

$$g(z_n) = \text{activation function output} \quad (2)$$

An example of the mathematical operations performed between the input layer and the hidden layer is shown in Figure 8.

Figure 8 - Example of mathematical operations between the input layer and the hidden layer.



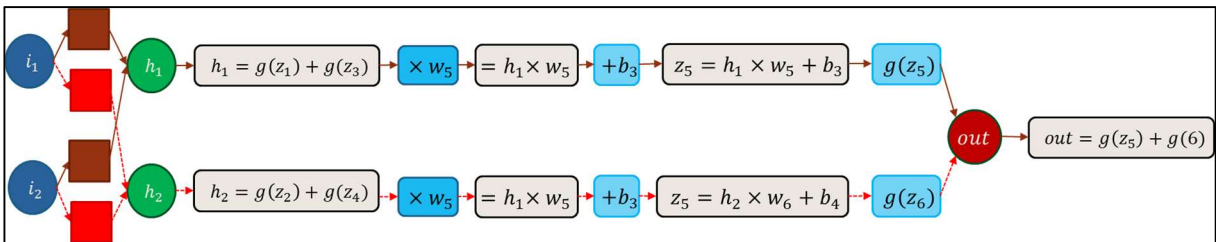
Source: The Author.

Each hidden neuron output is the sum of each input value and its respective weights, bias, and activation function. Equation 3 shows an example of the output of the first hidden neuron shown in the previous figure.

$$h_1 = g(z_1) + g(z_3) \tag{3}$$

The operations between the input and hidden neurons are like the relationship with the hidden and the output neuron. Figure 9 shows how the hidden neuron is connected and its mathematical operations between them.

Figure 9 - Example of mathematical operations between the hidden layer and the output layer.



Source: The Author

The output value is also known as the predicted value. The error of a model is based on the predicted value and the desired output value. The cost function is defined by the squared value of the difference between the predicted and the desired value. This function is also known as loss function or error rate depending on the authors. Equation 3 shows how the cost function is calculated.

$$\text{cost or loss error rate} = (\text{Predicted Value} - \text{Desired Value})^2 \tag{3}$$

The main objective of an AI model is to minimize the cost function, where the optimal solution is the global minimum where error tends to zero. This can be achieved by updating the weights and biases, evaluating how a change in the weight or bias

affects the cost function. Calculating the derivate of the cost function with respect to each weight and bias, it is possible to know how small changes in any of the parameters will affect the cost function. For the following equations, the cost function will be treated as 'C', the predicted value as 'P' and the desired output will be treated as y. Equation 4 shows how the cost function is calculated.

$$\mathbf{C} = (\mathbf{P} - \mathbf{y})^2 \quad (4)$$

Equation 5 shows an example of the derivative of the cost function regarding the weight value w_5 located between the hidden neuron and the output neuron using the chain rule for derivatives.

$$\frac{\partial \mathbf{C}}{\partial w_5} = \frac{\partial z_5}{\partial w_5} * \frac{\partial \mathbf{P}}{\partial z_5} * \frac{\partial \mathbf{C}}{\partial \mathbf{P}} \quad (5)$$

Calculating separately each derivative, the Equation 6 is the derivative of the cost function with respect to the predicted output value:

$$\frac{\partial \mathbf{C}}{\partial \mathbf{P}} = 2 * (\mathbf{P} - \mathbf{y}) \quad (6)$$

The Equation 7 is the derivative of the predicted output value with respect to the sum of the output values of the activation function outcomes:

$$\frac{\partial \mathbf{P}}{\partial z_5} = \mathbf{g}'(z_5) \quad \text{where} \quad \mathbf{P} = \mathbf{g}(z_5) + \mathbf{g}(z_6) \quad (7)$$

The Equation 8 is the derivative of the value z_5 with respect to the weight w_5 :

$$\frac{\partial z_5}{\partial w_5} = \mathbf{h}_1 \quad \text{where} \quad z_5 = \mathbf{h}_1 * w_5 + \mathbf{b}_3 \quad (8)$$

Multiplying each derivate value and replacing in the Equation 9:

$$\frac{\partial \mathbf{C}}{\partial w_5} = 2 * (\mathbf{P} - \mathbf{y}) * \mathbf{g}'(z_5) * \mathbf{h}_1 \quad (9)$$

The cost function will be affected by the difference between the predicted and the desired value, the derivate value of the activation function, and the input value of the hidden neuron. Based on that, it is possible to recalculate the new w_5 value, which depends on the learning rate value. Equation 10 resumes how the weight equation is updated.

$$\mathbf{w}_{5_{\text{new}}} = \mathbf{w}_5 - \alpha_{\text{learning rate}} * \frac{\partial \mathbf{C}}{\partial w_5} \quad (10)$$

Replacing the derivative in the Equation 11 above we have:

$$\mathbf{w}_{5_{\text{new}}} = \mathbf{w}_5 - \alpha_{\text{learning rate}} * 2 * (\mathbf{P} - \mathbf{y}) * \mathbf{g}'(z_5) * \mathbf{h}_1 \quad (11)$$

During the learning process and weight update process, it is possible that the derivative of the activation function, given by $g'(x)$, causes the vanishing gradient. Sigmoid functions are one type of activation functions which can be subject to the vanishing gradient problem since its derivatives tend to zero depending on how the input values vary. The vanishing gradient effects propagation greatly increases as the neurons layer a far from the output value. This can be observed in the following Equations 12,13,14,15 and 16, where the cost function derivative with respect to w_1 is evaluated:

$$\frac{\partial C}{\partial w_1} = \frac{\partial z_1}{\partial w_1} * \frac{\partial h_1}{\partial z_1} * \frac{\partial z_5}{\partial h_1} * \frac{\partial P}{\partial z_5} * \frac{\partial C}{\partial P} \quad (12)$$

Calculating each one of the derivatives:

$$\frac{\partial z_1}{\partial w_1} = i_1 \quad \text{where} \quad z_1 = i_1 * w_1 + b_1 \quad (13)$$

$$\frac{\partial h_1}{\partial z_1} = g'(z_1) \quad \text{where} \quad h_1 = g(z_1) + g(z_3) \quad (14)$$

$$\frac{\partial h_1}{\partial z_1} = w_5 \quad \text{where} \quad z_5 = h_1 * w_5 + b_3 \quad (15)$$

Replacing the values calculated above we have:

$$\frac{\partial C}{\partial w_1} = i_1 * g'(z_1) * w_5 * g'(z_5) * 2 * (P - y) \quad (16)$$

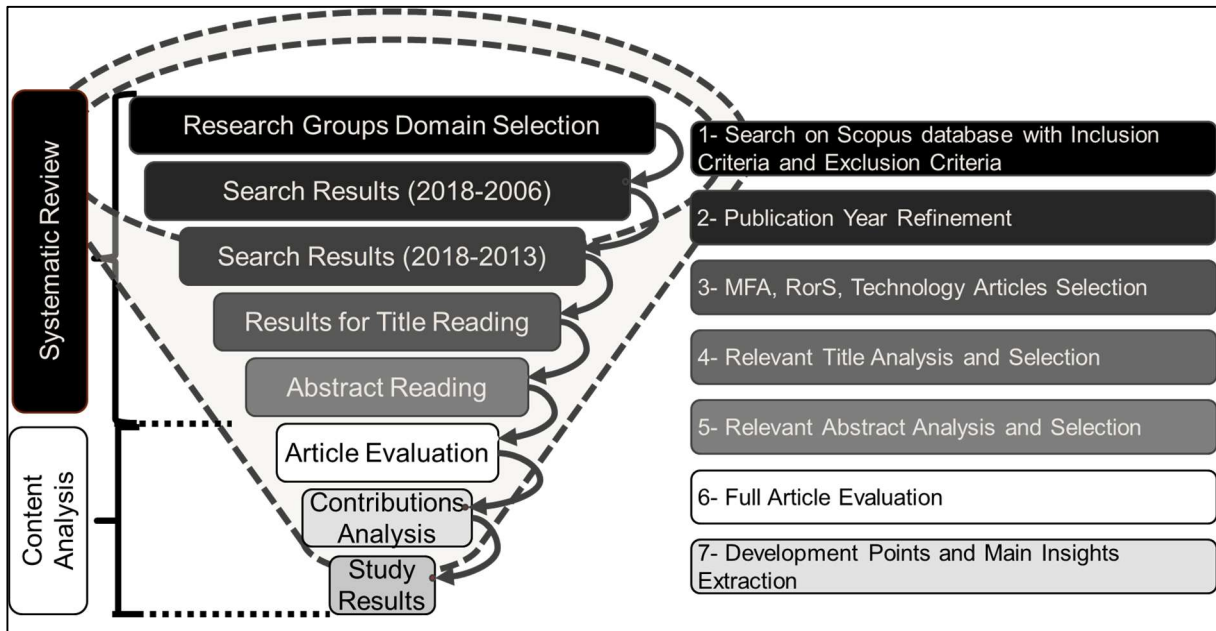
The influence of the w_1 is twice affected by derivate values from the activation function. In this case, the vanishing gradient will be twice as strong as the effect on the weight w_5 . The farthest a neuron is from the output, the greater the effect of the vanishing gradient.

To perform the learning of the model, each parameter w_n and b_n should be subject to update. When all the values are updated, the new prediction value can be calculated. This process is repeated infinitely, and it can be stopped when cost function achieves a minimum value, or a certain number of iterations is performed.

3 RESEARCH METHODOLOGY

The research method used in this article was the systematic review followed by content analysis. The purpose of this review is to identify which are the main requisites for AI MVS and which are the state-of-the-art techniques that can be applied to industrial inspection. Figure 10 provides a detailed explanation of each step performed in this study. Each step will be detailed later in this document.

Figure 10 - Research methodology steps performed in this study.



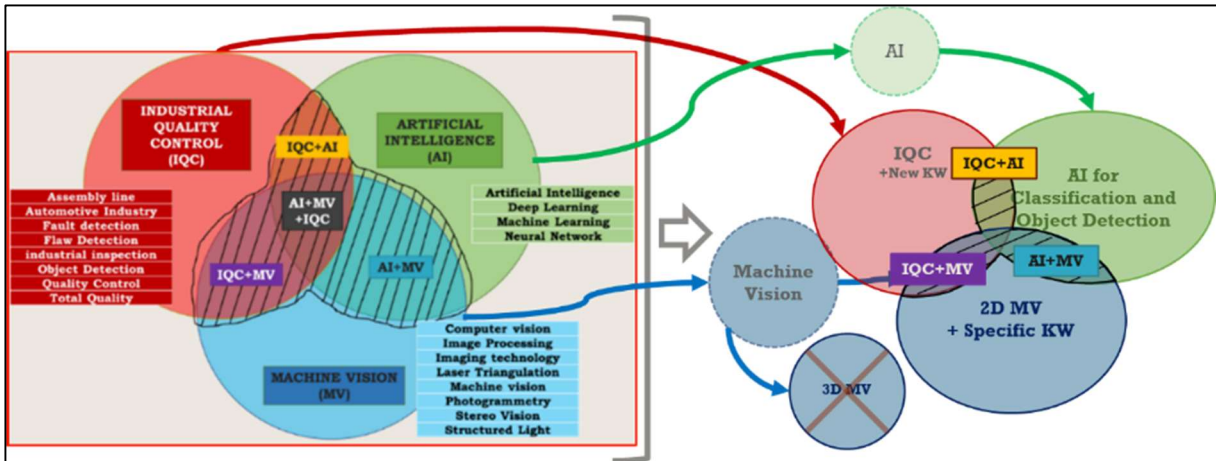
Source: The Author

3.1 Systematic Review

Step 1. The first technique is used to ensure the selection of relevant articles of a chosen study area and its followed by a rigorous filtering process which provides an accurate search result list. This type of review benefits from the available literature datasets related to the desired research field integrating information from different studies which may present converging or diverging results.

The review begins with the definition of the main study groups. The study groups for this paper were based on the previous fields of study as described by Silva *et al.* (2018). The research groups were slightly modified to provide an evaluation aligned with the research questions from this paper. The main research groups were IQC, AI, and MV. This article provides a wider evaluation, considering articles that belonged not only to the intersection between the three main groups but the relationships within each group pair. Figure 11 shows the differences between, and this paper relative to new keywords group names.

Figure 11 - Proposed field groups correlation.



Source: Adapted from Silva *et al.* (2018)

MV group was divided into two types of solutions: 2D MV applications and 3D MV applications. 3D keywords were used as an exclusion filter, while 2D keywords were used in the inclusion filter. AI keywords were added based on Schmidhuber (2015) survey and Druzhkov; Kustikova (2016) review. Specific AI techniques for image classification and object detection, such as ML and DL, were also added in the inclusion keyword group list. New IQC keywords were found using similar industrial applications keywords based on findings from Silva *et al.* (2018) and keywords related to quality control and performance evaluation. Table 1 shows the keywords used as the inclusion filter for each group.

Table 1 - Keywords used as the inclusion filter divided into the research groups.

AI Keywords	IQC Keywords	MV Keywords
Automatic Encoder*	Assembly line	Computer Vision
Convolutional Neural Network or CNN	Automotive Industry	Feature Extraction
Deep Belief Network or DBN	Confusion Matrix	Image Classification or Image Classifier
Deep Learning	Factory or Factories	Image Processing
Extreme Machine Learning or ELM	Fault detection or Anomaly Detection	Imaging Technology
Multi Layer Perceptron or MLP	Flaw Detection or Defect Detection	Machine Vision
Machine Learning	Industrial Application	Multi Target Tracking or MTT
Neural Network	Industrial Inspection	Object Detection
Restricted Boltzmann Machine or RBM	KPI or Key Performance Indicator	Object Recognition or Part Detection
R-CNN	Manufact*	ORB
Recursive Neural Network	Quality Control or Automated Quality Control	Pattern Recognition or Shape Recognition
	Product Analysis or Product Quality	SIFT
	Quality Expert	SURF
RNTN	Quality Prediction	Template Matching
	Total Quality or Visual Quality	

Source: The Author.

The search was performed using advanced document search features, such as operators and combine queries options available through Scopus dataset. Search combinations were performed only between different keyword groups. Only papers in the English language that were classified by Scopus as articles, reviews, conference proceedings, and conference reviews were considered in this search.

Papers covering a period between 2006 to the first half of 2018, along with the additional premises and criteria are shown in Table 2.

Table 2 - Inclusion and Exclusion criteria for the first step of the systematic review.

Exclusion Criteria
Keywords related to:
<p>- medicine: x-ray ; Alzheimer ; analgesic ; auditory ; biopsy ; blastocyst ; blood ; auditory ; biopsy ; blastocyst ; blood ; bone ; bone ; brain tumor ; brainstem ; breast ; cancer ; cardiac ; cardiology ; cardiovascular ; cartilage ; catheter ; dermoscopy ; diabetic; disease ; DNA ; Echocardiography ; EEG ; electrocardiogram; electroencephalogram ; electrophysiological ; endoscopy ; fetal ; fluoroscopy ; genome; gemologic* ; health ; hippocampus ; histopathology ; laparoscopic ; liver ; lung ; mammographic ; mammography ; medic ; mitotic ; MRI ; neuroimaging ; medic ; mitotic ; MRI ; neuroimaging ; olfactory ; olfactory ; pancreatic ; pathological ; patient ; pediatric ; physiotherapy ; pneumonia ; protein ; proteomics ; psychology ; proteomics ; psychology ; radiation ; radiology ; radiotherapy ; rehabilitation ; resonance ; retinal ; saliva ; skeleton ; spinal cord ; stroke ; subcutaneous ; surgery ; syndrome ; thyroid ; tissue ; tissue ; tomography ; toxicity ; tractography ; tumor ; white matter ; cytometry</p>
<p>- human-related applications: balancing ; behavioral ; body postures ; celebrity ; crowd analysis ; crowd behaviour ; driver models ; ear recognition ; emotional ; face ; facial ; finger detection ; gesture ; hand gesture ; handwrit ; Handwriting ; human actions ; human activity ; human movement ; human part ; human pose ; multi-touch ; online purchase ; pedestrian ; sign language ; speech ; speech recognition ; surveillance ; traffic ; urban area ; urban environmental ; urban scenes ; urban stree* ; privacy ; guidance ; fingerprint</p>
<p>- non-industrial applications and non-MVS industrial applications: HEVC ; spores ; image restoration ; Hyperspectral ; accelerometer ; aerial images ; aerodynamic ; agriculture ; air quality ; airborne ; animal ; animation ; archaeology ; Astronomy ; audio ; Autonomous vehicles ; barley ; BIM ; biological ; broccoli ; bunn ; canvas ; canvases ; cartographic ; cellular ; chemical sensor ; cinema ; clothing ; collision ; cultivated ; customer ; dentistry ; dried figs ; drying ; egg ; energy ; engine test ; forecast ; fruit ; Galactic ; Galaxy ; Geographic ; geospatial ; gyroscope ; harmonic ; horticulture ; hydraulic ; hydrological ; hydrophobicity ; impedance ; insect ; landmarks ; landscape ; landslid* ; language identification ; leaves ; line balancing ; magneto-optical ; meteorolog* ; microscope ; microscopic ; microscopy ; music ; Navigation; PID control; potato ; power transformer ; RFID ; risk ; road-vectorization ; rocks ; satellite ; schedule ; scheduling ; seismic ; smart grid ; soccer ; sonar ; sound ; substation ; supplying ; Text detection ; Text recognition ; transformer faults ; transmission lines ; ultrasonic; vibration ; voltag* ; Waldo ; wastewater ; water distribution ; water quality ; web ; wood ; yarn</p>
<p>- Measurement MVS: LiDAR ; radar ; superresolution; Laser ; measurement</p>
Inclusion Criteria
Keyword selection according to related articles of previous review and surveys
<p>Search combinations only between different groups: - "Keywords Group IQC" AND "Keywords Group MV" - "Keywords Group IQC" AND "Keywords Group AI" - "Keywords Group AI" AND "Keywords Group MV"</p>
Keyword matching only in Abstract, Title or Main Keywords of papers available through Scopus dataset, covering the period between 2006 to 2018
<p>Document types searched: - Article - " ar "; - Conference Proceedings - " cp "; - Conference Review - " cr "; - Review - " re "</p>
<p>Example of search string : TITLE-ABS-KEY (("AI") AND ("MV")) AND PUBYEAR > 2005 AND PUBYEAR < 2019 AND (LIMIT-TO (DOCTYPE , "cp") OR LIMIT-TO (DOCTYPE , "ar") OR LIMIT-TO (DOCTYPE , "cr") OR LIMIT-TO (DOCTYPE , "re")) AND (LIMIT-TO (LANGUAGE , "English")) AND NOT ("Exclusion Keywords List")</p>
Document language: English
Search Results Dataset Information
<p>Search output format in the Research Information Systems (RIS) format containing the following information's: Author, Title, Publication Year, Document Type, Serial Identifiers, and Abstract.</p>

Source: The Author.

This publication period was chosen to provide an overview of how each pair of keyword group behaves over the years. The keyword exclusion criteria were based on keywords extracted from Silva *et al.* (2018) paper that already detected a significant number of articles non-relevant to the study objectives. Keywords related to medicine, human features, non-industrial, or other applications were considered in the exclusion filter. Table 2 resumes the inclusion and exclusion criteria applied in this paper.

The output of each search result was exported in the Research Information Systems (RIS), containing the following information about the articles: Authors, Title, Publication Year, Document Type, Journal or Conference Name, Serial Identifiers, main Keywords, and Abstract. These output files were converted into an organized datasheet. All the search results were unified into a single datasheet adding the information of the pair of keyword group search string. Table 3 shows an example of the simplified datasheet output, but the main file output file contains twelve columns for authors and more than fifty columns for keywords for each tile.

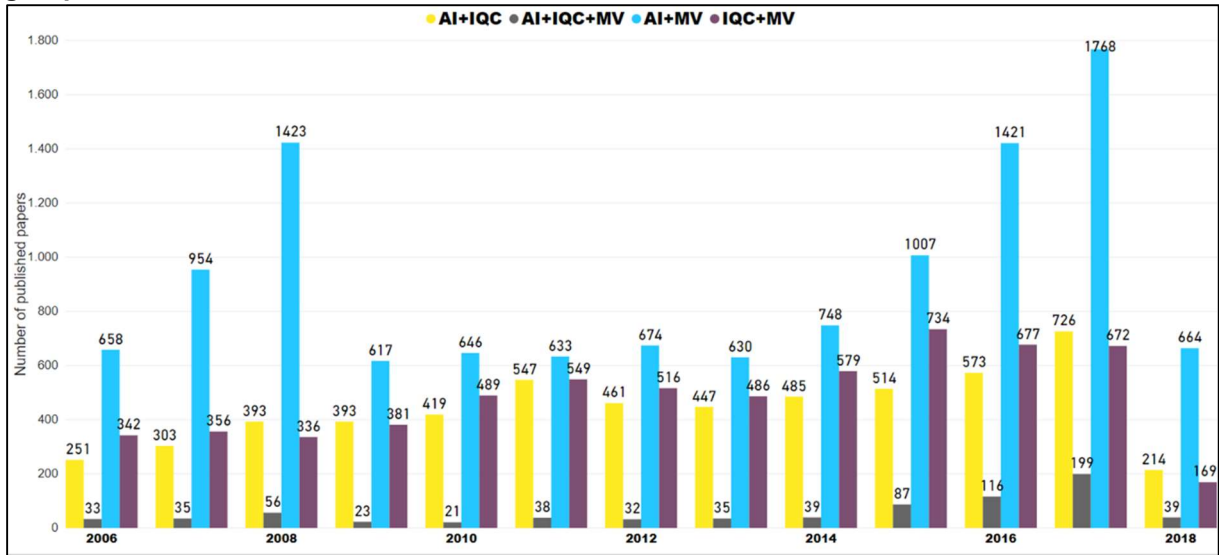
Table 3 - Output datasheet format and relevant columns used in the next steps.

SEARCH GROUP PAIR	AUTHOR 1	AUTHOR 2	TITLE	PUB YEAR	ABSTRACT	DOC TYPE	KEYWORD 1	KEYWORD 2	ISSN/ ISBN	JOURNAL/ CONF NAME
AI+IQC	a1	a2	Title Example	2017	Full Abstract Example	CONF	k1	k2	ISSN/ ISBN	Proceedings name

Source: The Author.

The combined search results totalized 24.608 articles after removing duplicate values. Duplicate results that were found in more than one different search group were labeled as 'AI+MV+IQC'. Figure 12 shows the number of papers published over the years after the removal of the duplicate value. Between 2006 and 2009 there was a significant increase in articles related to the 'AI+MV' group and another increase after 2014. The 'AI+IQC' and 'IQC +MV' does not present any major increase in publications, but the themes remain steady over the years. The 'AI+MV+IQC' theme has a less significant number of published articles, but its interest increased over the years 2016 and 2017.

Figure 12 - The number of publications between 2006 and 2018 distributed into the research group combinations.

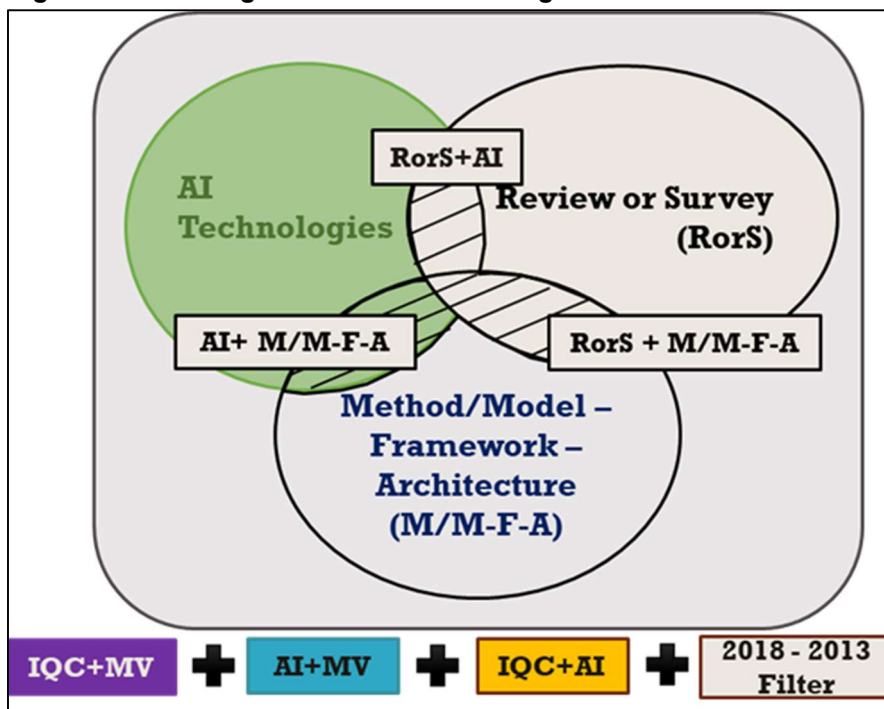


Source: The Author.

Step 2. To provide a better insight into what is currently being developed in each area, it was assumed that the recent papers in the last 5 years contain the most relevant and updated information. A filter was applied considering only publications between the year 2013 and 2018 resulting 13029 remaining papers.

Step 3. The next step of data extraction is based on a new keyword group. These groups are review or survey (RorS) keywords, AI techniques, and the Method/Model-Framework-Architecture (M/M-F-A) as shown in Figure 13.

Figure 13 - Filtering criteria for title reading.



Source: The Author.

RorS keywords analysis can give a very accurate filtering step for what is currently being developed since paper objectives are to provide an overview of each subject. The presence of M/M-F-A keywords provides insights about which is the focus solution, which may be related to a specific or general solution. The concept of method, framework, model, and architecture is based on Van Bon *et al.* (2007) definitions. A method is normally related to an approach that achieves a specific result, while a framework is an organized combination of methods. Different frameworks can be built based on the same model. Models provide a conceptual or schematic representation of a real system. Architecture can be considered a physical process and product or a combination of multiple frameworks.

AI techniques were divided into 3 groups to find more specific information about AI MVS. The first group is the Artificial Neural Network (ANN) Group, which has a common set of techniques applied to computer vision. The second group is composed of the common computer vision feature extraction techniques. The second group alone does not have AI characteristics, but it can be combined with them. The last group is composed of all other AI techniques. Table 4 shows the list of the groups and its respective keyword set.

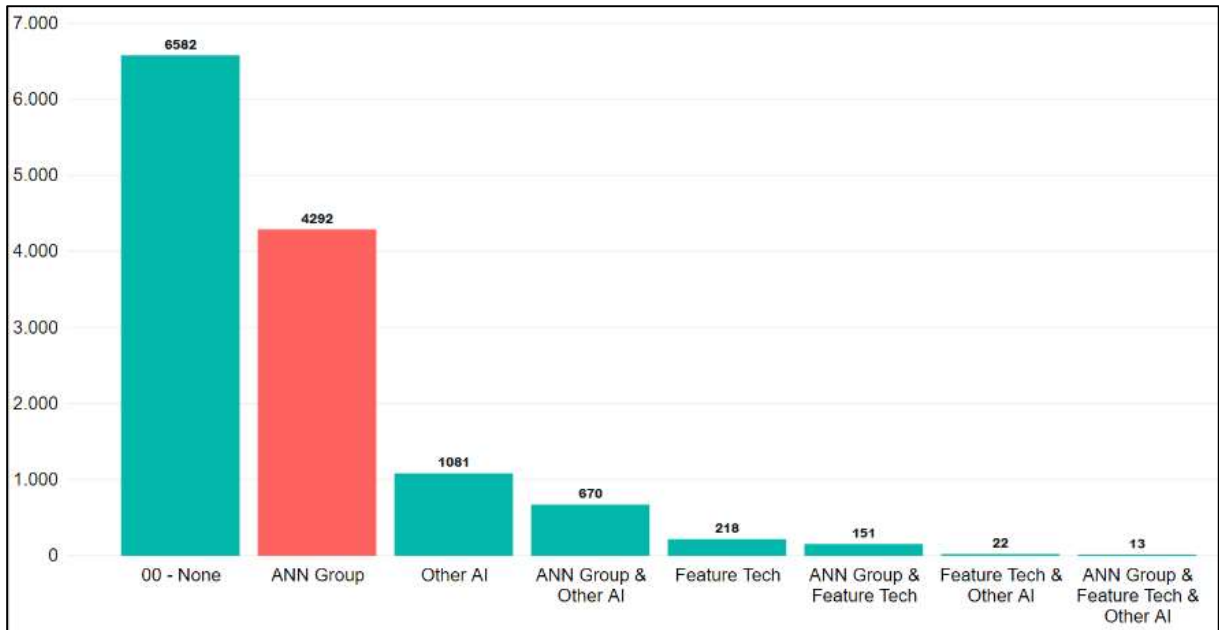
Table 4 - AI techniques divided by group.

ANN Group	Feature Extraction Techniques
ANN ; Artificial Neural Network ; CNN ; Convolution ; Convolutional Neural Network ; R-CNN ; Deep Learning ; Machine Learning	ORB ; SIFT ; SURF ; Template Matching
Other AI Techniques	
Automatic Encoder; Automatic Encoders ; Bayesian ; DBN ; Deep Belief ; Deep Belief Network ; EML ; Encoder ; Extreme Machine Learning ; Fuzzy ; Genetic Algorithm ; MLP ; Multi Layer Perceptron ; MultiLayer Perceptron ; Recursive Neural Network ; Recursive Neural Tensor Network ; Reinforcement Learning ; Restricted Boltzmann machine ; RNTN ; Support Vector Machine ; SVM ; Swarm Intelligence ; Swarm Optimization ; RBM	

Source: The Author.

An analysis of the AI techniques group alone shows that 6582 articles do not have any keywords of the groups above in their title or abstract. The second most relevant value is the ANN group, which confirms the predominance of these techniques applied to MVS. Other AI techniques and its combination with the ANN group has a fewer article published, but its relevance should not be dismissed in the following filter steps. Figure 14 shows the number of publications according to each technique group.

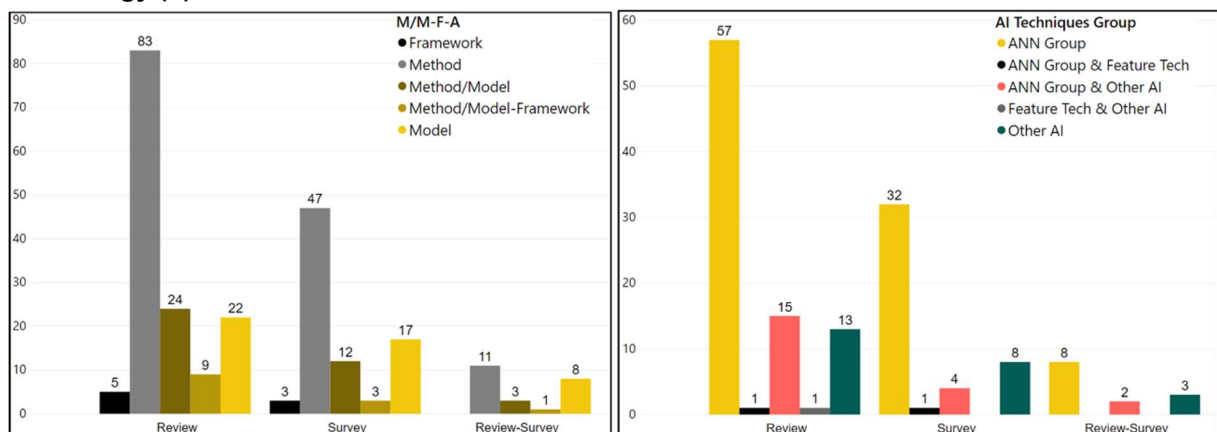
Figure 14 - Number of publications according to each technique group.



Source: The Author.

RorS keyword group combined with M/M-F-A analysis is shown in the left graphic in figure 8. Most of the reviews and surveys performed are related to methods, followed by a combination of models and methods and models alone. Based on these insights, more specific solutions are more likely to be found rather than schematic representations of the solution. It can also be observed that RorS results are strongly related to the ANN group, based on the graphic in the right of Figure 15.

Figure 15 - Review and Survey number of publications divided by M/M-F-A (a) and divided by technology (b)



Source: The Author.

The title reading and abstract reading selection and exclusion criteria were based on the information extracted from the previous analysis. Table 5 resumes the selection criteria and the exclusion criteria before title reading step.

Table 5 - Selection and Exclusion Criteria

Selection Criteria
All the papers within the RorS group; Papers with specific contributions to the ANN group applied to MVS/computer vision; Papers with other AI solutions applied to MVS/computer vision; Papers with ANN group and other AI solutions applied to MVS/computer vision; Solutions containing Model and Frameworks keywords for AI and MVS; Papers with insights on how AI methods and models are being validated; Papers with insights into how-to setup AI model parameters.
Exclusion Criteria
Papers which did not belong to any of the keyword groups: RorS, M/M-F-A or AI techniques. AI applied to quality control that is not related to MVS. Papers

Source: The Author.

Step 4 and 5. While applying the first exclusion criteria, 3420 papers were removed, remaining only 9609. All these papers were subject to the selection and exclusion criteria, and 8631 articles were removed during title reading phase, and 650 were removed during the abstract reading step. The remaining 328 papers were selected for the complete paper reading phase and content analysis. Table 6 resumes the filtering steps between title selection to abstract reading.

Table 6 - Filtering steps between title selection to abstract reading resume.

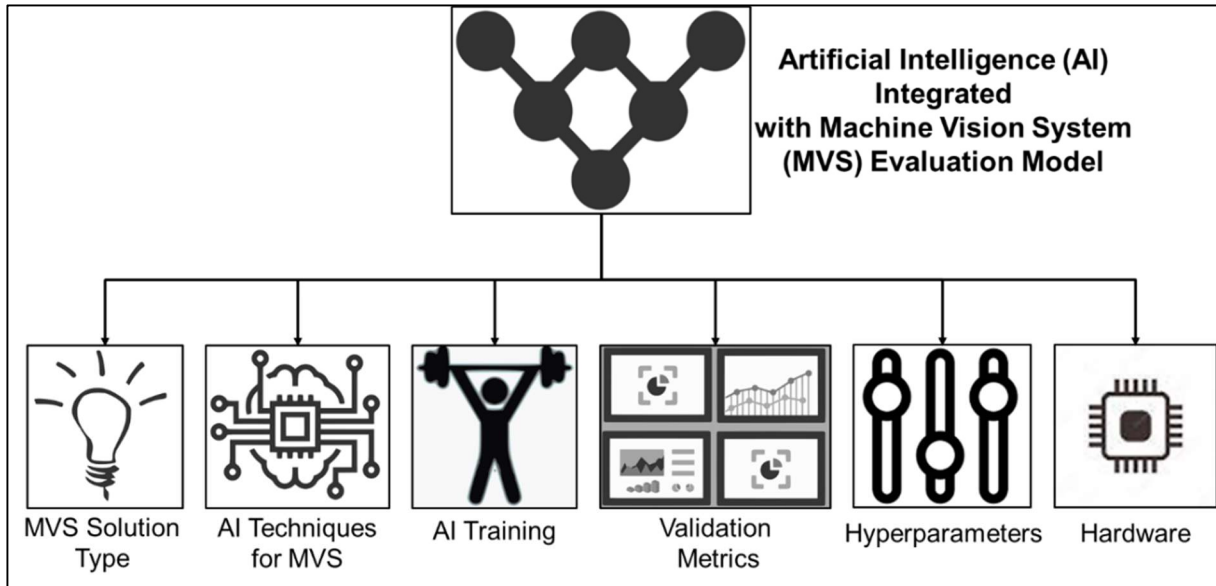
Status	Title Count
Not selected for Title Reading	3420
Removed during Title Reading step	8631
Removed during Abstract Reading	650
Selected for complete paper reading	328
Total	13029

Source: The Author.

3.2 Content Analysis

Step 6. While reading the 328 selected papers, a common structure was detected. This pattern can be used and divided into main knowledge groups, which are commonly found in almost every paper. An evaluation model of the AI integrated MVS solution is proposed by the authors, and it is built based on 6 knowledge groups found: MVS solution type, AI techniques for MVS, AI Training, Validation Metrics, Hyperparameters, and Hardware. Figure 16 shows the proposed evaluation model with the knowledge groups mentioned above.

Figure 16 - The proposed evaluation model for complete title reading divided by subgroups.



Source: The Author.

All selected papers were evaluated under the following criteria that were applied during the full article reading step. Table 7 details the criteria that refine content analysis.

Table 7 - Detailed criteria for each subgroup in the proposed evaluation model.

Group	Criteria Description
MVS Solution Type	Papers which has the main application in Object Recognition, Image Classification, Image Labelling, Object Classification, and Semantic Segmentation;
AI technique for MVS	Papers which has detailed information about AI technique applied for the solution: Example ML, DL, ANN, CNN, R-CNN, SVM
AI Training	Papers with clear and detailed information about the training type, dataset source or dataset improvement;
Hyperparameters	Papers with a clear and detailed explanation of the framework, dataset or layers parameters;
Validation Metrics	Papers with a clear and detailed explanation of the metrics used to validate the developed solution;
Hardware	Papers with clear and detailed information about the hardware used to train or the hardware used to deploy the trained model;
Specific Criteria	<ul style="list-style-type: none"> - Review and Survey – Relevant papers containing review and surveys applied to image classification, object detection or improvement of ML, DL, ANN or CNN techniques were kept; - Articles containing ML and DL architectures or framework with potential solutions to improve existing classification and detection techniques were also kept.
Exclusion Criteria	<ul style="list-style-type: none"> - Papers that contain only theoretical solutions without details of implementation - Papers with specific techniques or very exclusive solutions - Papers with a mathematical approach without solution validation.

Source: The Author.

The result of this step resulted in 98 potential articles to be fully evaluated by the proposed model. The remaining 230 papers were removed because they were included in the previous exclusion criteria, which removes specific solutions, papers with theoretical or mathematical approach without a detailed validation or application.

The complete evaluation has the objective to identify what is the contribution type regarding the AI perspective and MVS perspective. It also assesses what types of training, which kind of dataset and if there are any type of improvement with the training material. It also verifies if the has a workflow or diagram detailing how the proposed solutions are applied, verifying whether the hyperparameters are used or not. Validations metrics are also included in the assessment to verify if the paper informs if the solution was validated and which metrics were used. A solution can be validated to see how it classifies an image and whether it can detect the location of an object in a crowded environment of different objects. Metrics regarding how much time it is necessary to train an AI model and what it is performed after the model is deployed for inference. Model training and inference are directly related to the hardware used. The evaluation model verifies if the papers inform which hardware is used for both types of hardware. Some papers like reviews and surveys may not be subject to the non-applicable option. Table 8 provides detailed evaluation fields.

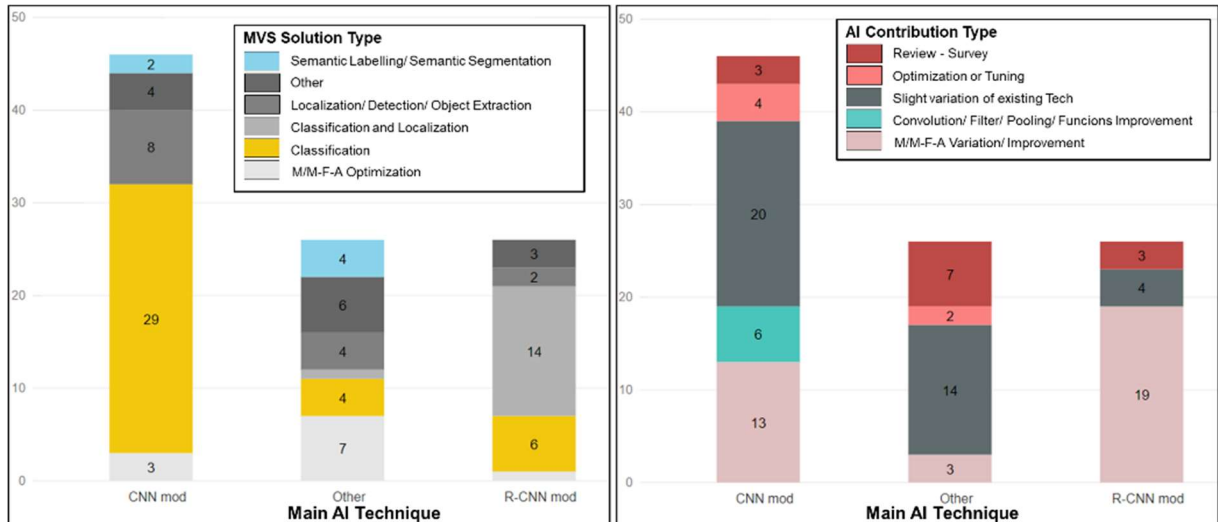
Table 8 - Detailed fields of how the papers were evaluated.

A TECHNIQUE FOR MVS	Paper Contribution Type	Review - Survey	HYPER-PARAMETERS	Layer Parameters	Informed or Non-informed/ Non-Applicable
		Minor Improvements of existing Technique		Workflow/ Diagram	
		Optimization or Tuning			
	Main Technique	Convolution/Filters /Pooling/Functions Improvement	VALIDATION METRICS	Classification Accuracy	Informed or Non-informed/ Non-Applicable
		M/M-F-A Variation/ Improvement		Localization Accuracy	
		R-CNN		Time to Train	
	Supervision Type	CNN	MVS SOLUTION TYPE	Deployed Solution Performance	Classification and Localization
		Other		Classification	
		Supervised		Semantic Labelling/Semantic Segmentation	
AI TRAINING	Dataset Source	Unsupervised	HARDWARE	Training	Informed
		Semi-supervised		Embedded/ Inference	Non-informed/ Non-Applicable
		Non-informed/ Non-Applicable			Informed - Same as Training
	Dataset Improvement	Standard and Custom	Other		Informed - Different from Training
		Standard			Non-informed/ Non-Applicable
		Custom			
		Non-informed/ Non-Applicable			
		Informed			
		Non-informed/ Non-Applicable			

Source: The Author.

The main AI techniques used was CNN with 46 papers. Other AI techniques and R-CNN resulted in 26 articles each. It can be observed that CNN is the core technique, and R-CNN has significant importance for MVS applications. Despite that, other AI techniques are still being researched for image applications, and these methods can also be combined with the existing techniques to provide new frameworks with better performance of the existing ones. The most common MVS solution type is Classification for CNN and Localization, followed by Classification for R-CNN. The main type of AI contribution detected is focused in slight variations of existing techniques. M/M-F-A with large variations and new proposals for improvements is the second most published AI contribution type. Figure 17 resumes the AI and MVS contributions type.

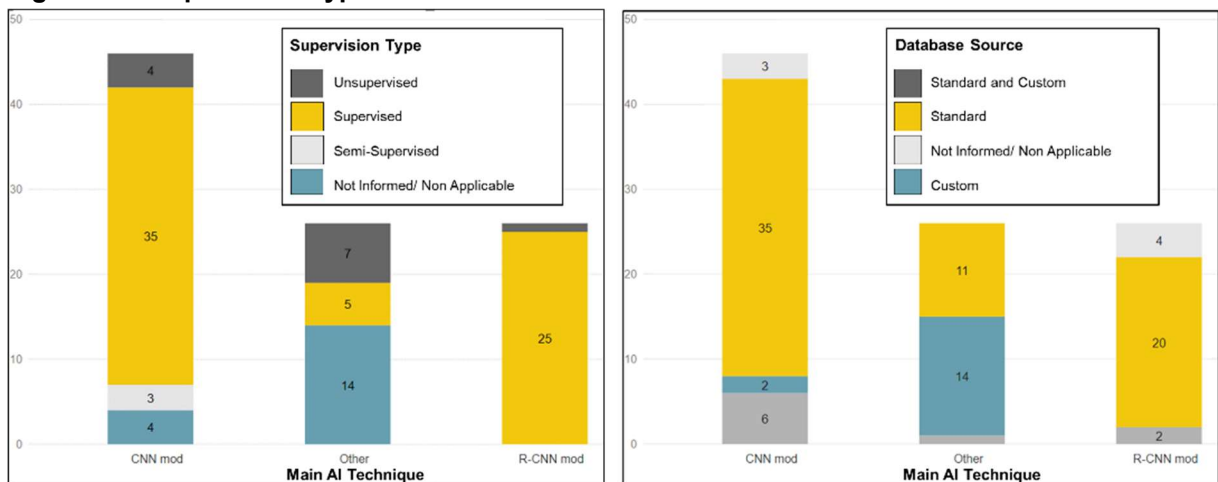
Figure 17 - AI contribution (a) and MVS solution type (b) results.



Source: The Author.

Regarding AI training, supervised training with standard image dataset without any data augmentation process is the most common method. But there are still articles which did non-informed any information related training. A standard dataset is used in online challenges so users can evaluate and compare their solution performance with other existing methods. Custom datasets with augmentation and improvement techniques are used in case of the standard dataset does not meet the desired detection. These techniques are used in case of the standard dataset does not meet the desired detection capacities. Figure 18 resumes the AI training evaluation for supervision type and dataset source.

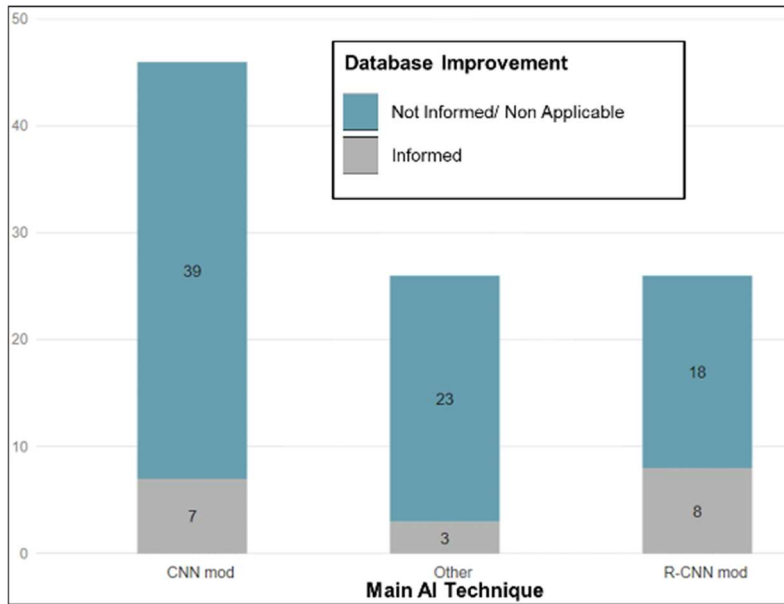
Figure 18 - Supervision Type and data source results.



Source: The Author.

Figure 19 shows the result for dataset improvement during AI training evaluation.

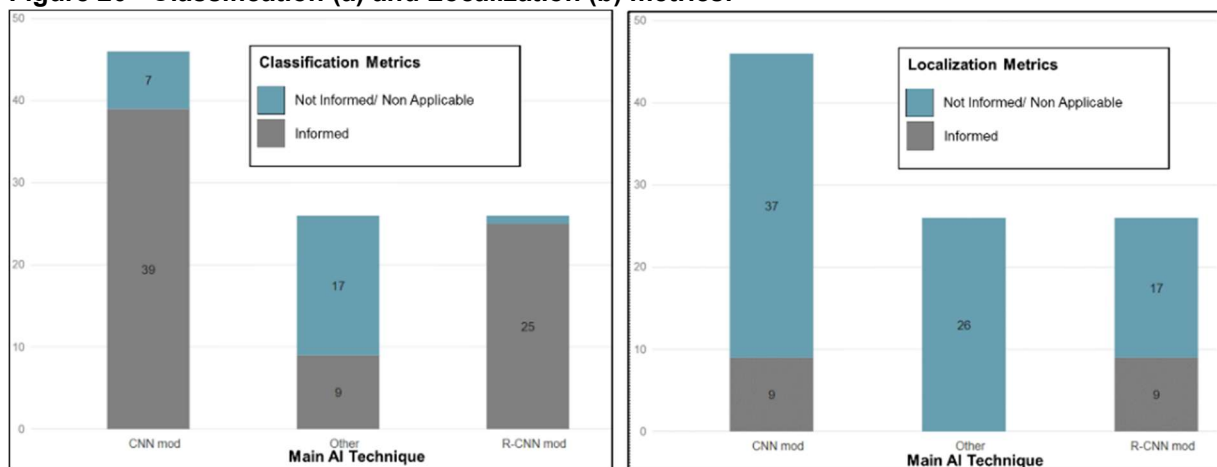
Figure 19 - Dataset improvement results.



Source: The Author.

Classification metrics are informed in most of the analyzed cases. Only in some types of AI techniques which the main goal was not classification the evaluation considered them non-applicable. In the other way, Localization metrics were not found in most of the cases. This can be observed in a CNN architecture which is used only for classification of the whole image so localization metric cannot be applied. Some R-CNN solutions consider an in-built localization algorithm or method, but it does not evaluate its performance later. Figure 20 resumes if classification and localization metrics were applied in the evaluated papers.

Figure 20 - Classification (a) and Localization (b) metrics.

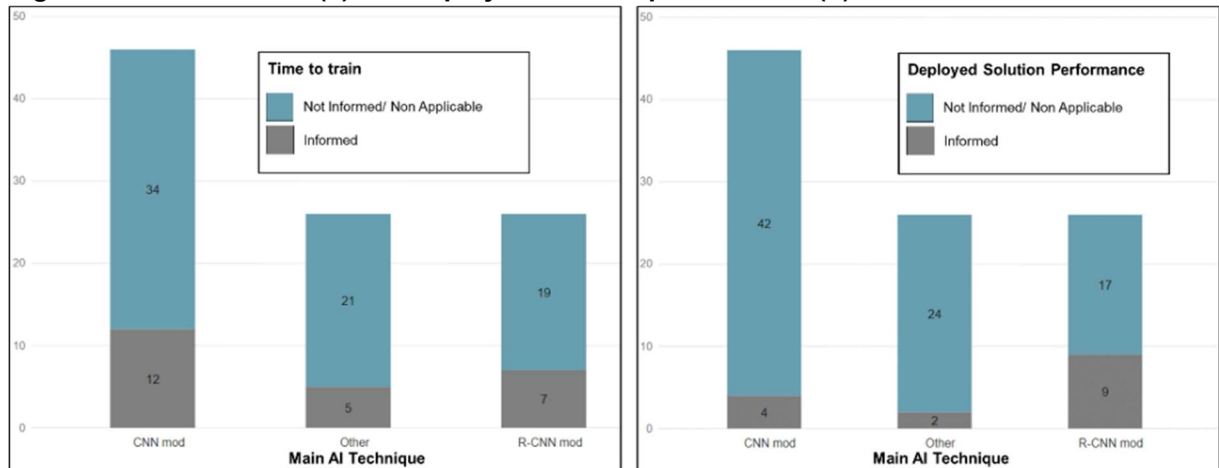


Source: The Author.

Performance metrics related to time to train and the deployed solution performance was not detected in most of the cases. The time to train may affect the

capacity of replicating a newly developed model which may rely on powerful training hardware or an expensive cloud-based solution so it can be properly trained. The performance of a deployed solution is also important because depending on how complex the model is, it would require a better processing capacity to operate in real time inference solution. Figure 21 resumes training and inference metrics were applied in the evaluated papers.

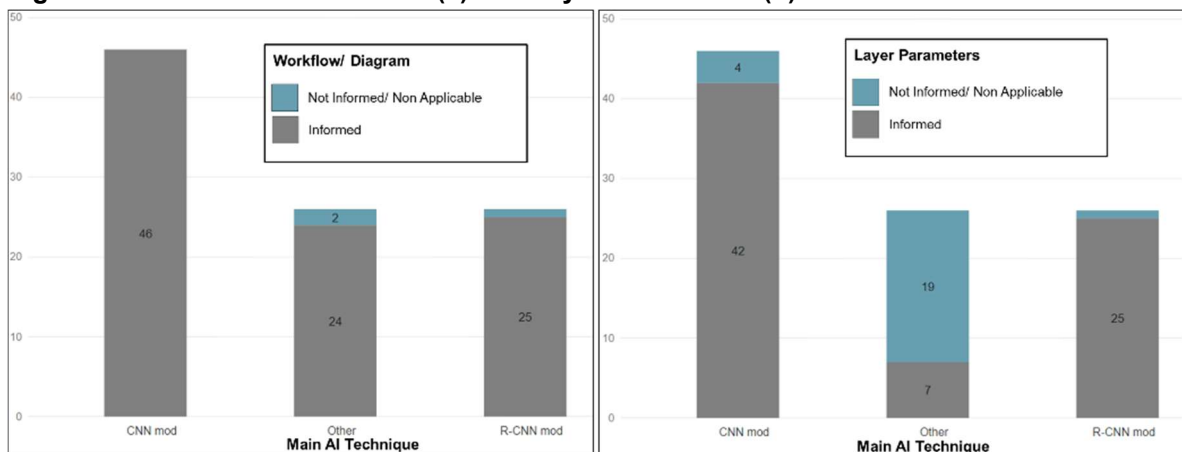
Figure 21 - Time to train (a) and deployed solution performance (b) metrics results.



Source: The Author.

Most of the analyzed studies provide workflow or diagram of the chosen AI solution. The presence of these diagrams reinforces the need for a clear overview of how an AI complex method or framework is organized. Layer hyperparameters are also well described in most of the cases, except for other AI techniques that are not based on ANN which organize neurons into layers. Figure 22 shows the results for layer hyperparameters and diagram or workflow presence.

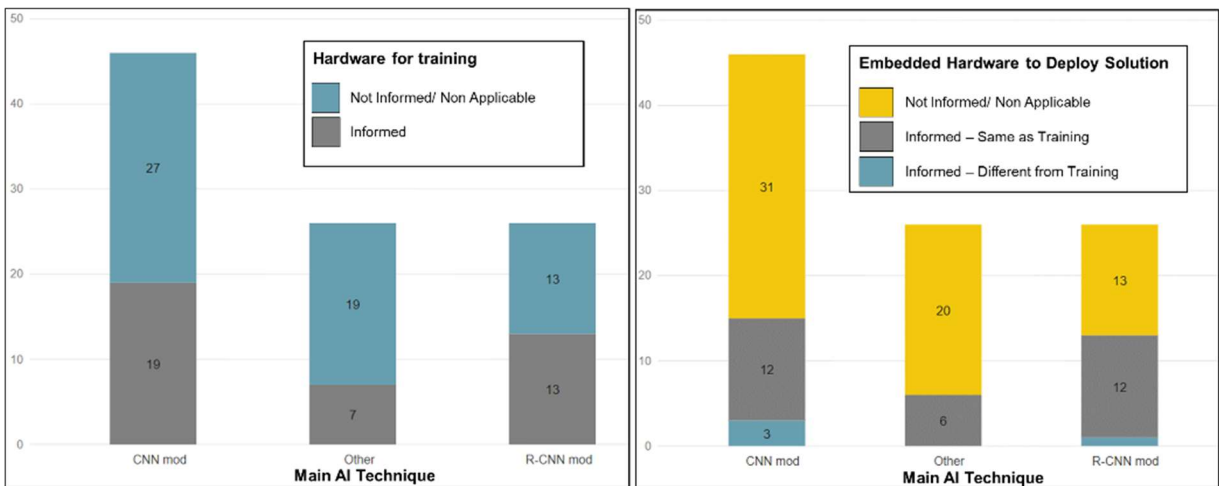
Figure 22 - Presence of Workflow (a) and Layer Parameters (b) results.



Source: The Author.

It can be verified that 59 from the 98 cases did not inform the hardware used for training the AI algorithm. Training a complex AI algorithm normally requires better hardware with GPU acceleration, so the lack of this information in the papers increases the difficulty to replicate a solution. Embedded hardware for the AI algorithm deployment, also known as inference, was not mentioned in 64 cases. Just like the training hardware, embedded hardware will limit the complexity of the deployed solution. Although the processing power needed for inference is lower than the training hardware, the absence of this information affects the ability of the solution be validated. Figure 23 resumes training and embedded hardware were applied in the evaluated papers.

Figure 23 - Presence of training hardware (a) and embedded hardware (b).



Source: The Author.

3.3 Content Analysis Contributions

Step 7. The final step resumes the 17 relevant papers extracted from the 98 analyzed papers that met most of the criteria's applied above or new approaches that provide better solutions to AI MVS. Table 9 resumes the content analysis final step, considering each contribution found in the selected paper and some insight for future developments that were not considered in the papers.

Table 9 - Content Analysis Resume.

Ref.	Contributions	Main Insights for Future Development
Yan <i>et al.</i> (2018)	<p>Propose a Spatial Alignment Network (SAN) which replaces the Region Proposal Network (RPN) used in R-CNN techniques.</p> <p>Increase training speed and inference without major loss classification accuracy</p> <p>Use the concept of Atrous Convolution</p> <p>Fulfill almost all criteria proposed by this evaluation model.</p>	<p>Used the same type of hardware for training and inference.</p> <p>Embedded hardware was not used.</p> <p>Mentions state-of-the-art techniques, but its performance comparison was not performed.</p>
Liu <i>et al.</i> (2016)	<p>Propose the SSD framework for fast object detection without reducing classification accuracy.</p> <p>Provide inference performance and classification accuracy comparison with state-of-the-art approaches and different datasets</p> <p>Use Atrous Convolution and data augmentation</p> <p>Fulfill almost all criteria proposed by this evaluation model.</p>	<p>Used the same type of hardware for training and inference.</p> <p>Embedded hardware was not used.</p>
Puttemans <i>et al.</i> (2018)	<p>Use transfer learning concept along with a pre-trained model based on a standard dataset.</p> <p>Use fine-tuning, custom dataset, and data augmentation to create new refined object class.</p> <p>Use the Recall-Precision curve as classification metrics</p> <p>The solution is structured for embedded applications.</p>	<p>Localization metrics was not applied.</p> <p>Training time and inference performance were not measured.</p> <p>Used the same type of hardware for training and inference.</p> <p>Embedded hardware was not used.</p>
Li <i>et al.</i> (2017)	<p>Propose a model that reduces the RoI generation time required for a trained R-CNN during inference.</p> <p>Use Recall-Intersect of Union (UoI) curve as classification and localization metric.</p> <p>The solution is structured for embedded applications.</p>	<p>Used the same type of hardware for training and inference.</p> <p>Embedded hardware was not used.</p>
Yu, S. <i>et al.</i> (2017)	<p>Use 3D models to automatically create a custom multi-pose dataset 2D dataset.</p> <p>Integrate several customized datasets.</p> <p>Perform a two-stage object classification: Coarse and fine-grained</p>	<p>Used the same type of hardware for training and inference.</p> <p>Embedded hardware was not used.</p> <p>Inference performance was not measured should be validated for the two-stage classification</p> <p>Localization metrics was not applied.</p>
Chen <i>et al.</i> (2018)	<p>Propose new improvements for the standard RPN.</p> <p>Unify classification layer and regression layer to reduce model complexity and accelerating training and testing speed.</p> <p>Use Recall-Intersect of Union (UoI) curve as classification and localization metric.</p>	<p>Used the same type of hardware for training and inference.</p> <p>Embedded hardware was not used.</p> <p>Mention high-speed inference state-of-the-art techniques, but its comparison with state-of-the-art techniques was not performed.</p>
Yao <i>et al.</i> (2018)	<p>Propose a network to learn the coexistence feature of multi-class objects detection to improve the classification.</p>	<p>Embedded hardware was not used.</p>

Ref.	Contributions	Main Insights for Future Development
	<p>May be used in an environment with many overlapping objects or pair of objects that are usually nearby each other, improving the classification.</p> <p>Use data augmentation</p>	<p>Inference performance must be validated for the two-stage classification</p> <p>Could be used for fined-grained classification.</p>
He (2017)	<p>Propose a channel pruning method to remove redundant filters during the convolutions.</p> <p>Channel pruning that accelerates inference speed without a great loss in classification accuracy CNN and R-CNN</p>	<p>Requires fine-tuning to achieve maximum acceleration performance with a lower error rate.</p> <p>Used the same type of hardware for training and inference.</p> <p>Embedded hardware was not used.</p>
Sun, Zhun; Ozay, Mete; Okatani (2016)	<p>Analyze the effects of quasi-hexagonal shapes of convolution kernels instead of square shape.</p> <p>Propose a feature visualization method for visualization of pixel-wise classification score maps of learned features.</p> <p>Reduce the number of parameters and computational time of CNN models.</p> <p>Improve the robustness of the baseline models to occlusion for classification of partially occluded images.</p>	<p>R-CNN frameworks were not evaluated with this method.</p> <p>Did not inform hardware for training and inference.</p> <p>The method improves detection accuracy, but localization metrics were not applied in this paper.</p>
Zhanquan; Fox (2012)	<p>Propose a Support Vector Machine with the customized dataset to replace the softmax layer and improve classification accuracy.</p> <p>Consider the effects of training time for each change performed in the framework and its effects on accuracy.</p>	<p>R-CNN frameworks were not evaluated using this method.</p> <p>Used the same type of hardware for training and inference.</p>
Kolesnikov ; Lampert (2016)	<p>Improve object localization cues (seeds) by incorporating a CNN with a semantic segmentation network.</p> <p>Uses a globally weighted rank pooling that combines max-pooling and average pooling.</p> <p>Use conditional random fields (CRF) to reduce imprecise boundaries.</p>	<p>R-CNN frameworks performance were not compared with the proposed method.</p> <p>Inference performance was not measured</p>
Hu <i>et al.</i> (2017)	<p>Combine CNN Saliency with a CNN localization with data augmentation into a single framework.</p> <p>Use Recall-Intersect of Union (UoI) curve as classification and localization metric.</p>	<p>Did not inform hardware for training and inference.</p>
Zhang <i>et al.</i> (2018)	<p>Propose the Rotation Invariant Local Binary Convolutional Neural Network.</p> <p>Use binary weights and rotating convolution filters to increase detection capacity and decrease processing power.</p> <p>The concept can be adapted to CNN and R-CNN to reduce the number of learned parameters</p>	<p>Did not inform hardware for training and inference.</p> <p>Did not provide localization metrics or performance metrics during inference and training.</p>
Yin <i>et al.</i> (2018)	<p>Evaluate inference classification precision of several object detection R-CNN techniques in embedded hardware different from the hardware used for training the model</p> <p>Compare the classification precision, mode, the image input size, model hyperparameters size and the embedded hardware performance during inference</p>	<p>Localization metrics was not measured.</p> <p>The time required for training of each model was not provided, although the lighter model is usually faster to train.</p>

Ref.	Contributions	Main Insights for Future Development
	Use the pruning filter and then fine-tune the model.	
Li <i>et al.</i> (2018)	Propose an ensemble R-FCN model by combing multi-scale inference, with more feature extractors and data augmentation Propose model that adapts to multi-scale object detection	Did not inform hardware for training and inference. Did not provide localization metrics or performance metrics during inference and training.
Zhiqiang; Jun (2017)	Provide an overview of CNN and R-CNN main techniques up the first half of 2017. Give a brief explanation of the structure of each network and the back propagation learning mechanism Provide a Precision-Recall classification metric Give the most common standard datasets used for model evaluation. Provide details for the Intersection Over Union (IOU) localization metric	Did not provide information about hardware for inference and training Did not provide performance metrics of each evaluated technique Did not provide detailed hyperparameter information.
Zhao <i>et al.</i> (2017)	Provide an overview of different frameworks using CNN and R-CNN techniques for fine-grained classification and semantic segmentation with a detailed workflow and accuracy validation for each proposed solution	Did not provide information about hardware for inference and training Did not provide performance metrics of each evaluated technique Did not provide detailed hyperparameter information.

Source: The Author

The most relevant AI technique for MVS is CNN for image classification and R-CNN for object detection [5], but research trends show that those techniques are being combined with other AI techniques to optimize hyperparameters (Li *et al.*, 2018), improve classification accuracy (Zhanquan; Fox, 2012), object localization precision and reduce the computational processing power (Li *et al.*, 2017), low weight frameworks and pruning techniques to remove redundant parameters (He, 2017). Multi-class detection and classification of complex images challenges are being overcome by two-step classification models with coarse and fine-grained classification (Yu. *et al.*, 2017) (Zhao *et al.*, 2017) and by using correlation information of nearby objects (Yao *et al.*, 2018). To increase classification accuracy and reduce model complexity, some solutions perform significant changes in convolution filter shapes and weights. Semantic segmentation techniques (Kolesnikov; Lampert, 2016), (Zhao *et al.*, 2017) are also being incorporated to standard CNN and R-CNN to provide better classification and object localization

Supervised training using standard dataset available in internet communities is the most common procedure adopted by researchers, mostly because they provide a common ground to evaluate algorithms performance. Researchers who desire to implement their solution in a specific application, such as machine vision inspection, may use pre-trained models with a standard dataset and fine tune them by using data augmentation to achieve better results, creating their own custom dataset. Yu *et al.* (2017) proposed a solution which uses an algorithm that transforms 3D files to create a multi-pose 2D custom image dataset to increase model classification robustness and can be used in an industrial scenario which already has a 3D digital version of its products.

The most common metric used to validate a model is Mean Average Precision (mAP) for simple classification algorithms when inference speed is not required. Classification only algorithms may also use the Precision-Recall curve, which provides a wider overview of the algorithm classification performance (Puttemans *et al.*, 2018), (Zhiqiang; Jun, 2017). Object detection solutions use Recall-UoI curve, which evaluates its capacity to detect the object location and its class (Chen *et al.*, 2018), (Hu *et al.*, 2017). The time required for training models and the number of images analyzed per second during inference are metrics that are beginning to be used by researches (Yan *et al.*, 2018), (Liu *et al.*, 2016) due to its necessity to make solutions

more readily available and match video frame rate speeds for real-time detections. Yin *et al.* (2018) were one of the few researchers that provided different embedded hardware for inference instead using the same hardware for training. MVS integrated with AI solutions should be considered this specific embedded hardware approach since existing MVS solutions rely on a plug and play solutions without sophisticated hardware or training servers spread over the factory.

During the paper reading, it was not found an implementation model that provides information to integrate AI models in the current Machine Vision Inspection scenario regardless of the chosen AI technique. Based on the main contributions and insights of the chosen papers, an implementation model is proposed to integrate these new AI solutions into the MVS.

The proposed model gathers all the visual inputs and results in outcomes from other inspection and creates a merged dataset. To integrate the new AI functionalities, a stable training environment, with validated programming libraries, must be created that is compatible with GPU-powered hardware either for training or for inference.

A machine vision expert must be trained to incorporate the AI programming new functionalities or a team with machine vision experts and AI programming must be built. This staff or team can be also be obtained from a third-party company depending on the size and budget availability of the industry. This team will analyze the visual dataset and will select the most suitable algorithm and apply any necessary modification that is compatible with the available training and embedded hardware. This algorithm should be validated in the training hardware considering all the classification, localization, and performance metrics already mentioned in this paper.

After the algorithm is validated in the training hardware, it should be validated in the industrial environment, using embedded hardware integrated with the camera. Algorithm refinement, hyperparameters optimization, and fine-tuning may be required to make the whole solution available for the embedded hardware. The solution must be validated in the industrial environment in order to achieve satisfactory output and to be easily integrated with the assembly line or inspection routines. The outputs from the inspections performed by the algorithms should also be stored together with the training image inputs to allow new architecture creations and evolve the existing ones.

Due to the constant evolving of algorithms, method, frameworks, and architectures the proposed model suggests the creation of a platform that will integrate

all the functionalities already mentioned into a single AI Machine Vision Management System. This platform will allow users to create the AI solutions, modify existing ones, evaluate and compare solutions performance, integrate new frameworks with new AI techniques, perform fine tuning, create ensemble architectures and apply transfer learning of already trained models into new ones. It should also allow users that are not familiar to machine vision or AI programming to interact with the system, facilitating its integration with quality expert users and plant managers.

4 PROPOSED MODEL

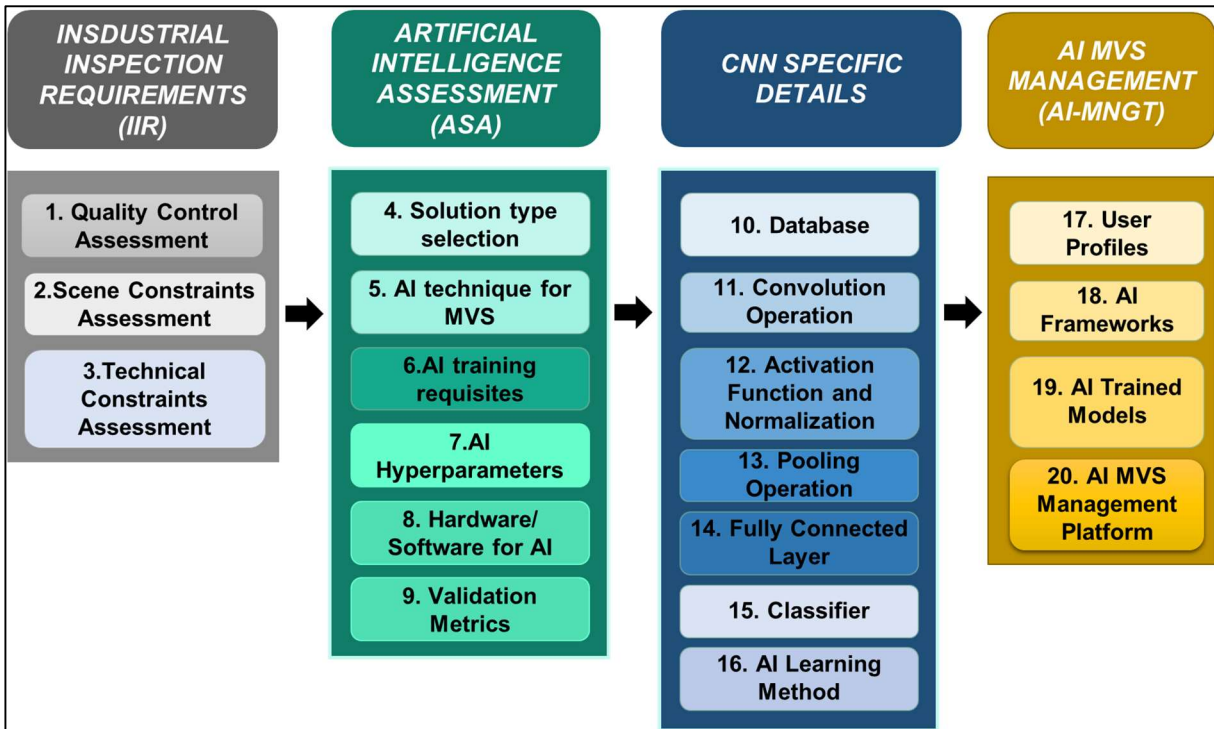
The implementation model proposed by this document is based on the systematic review and content analysis discoveries. It is developed in a straightforward workflow composed of four main groups. The main groups were divided into specific subgroups describing the main aspects of technical background to develop user knowledge and reduce process uncertainties. Each subgroup contains relevant information that allows the reader to understand what should be considered in each one of the implementation steps. The purpose of the model is not an automatic selection of AI MVS technologies since each industrial inspection scenario has its own characteristics, requirements, and specification.

The model should be considered as an iterative workflow, despite its straightforward development. It is recommended that the user go back and forward in the model while gathering requisites, selecting possible solutions, and validating into a real case scenario. Many technical requisites or new premises are found during the implementation steps. Every new requisite and premise should be evaluated and included in the AI MVS, improving its robustness of the complete solution.

4.1 AI MVS Implementation Model

The proposed AI MVS implementation model, as shown in Figure 24, is divided into four major groups.

Figure 24 - The proposed AI MVS implementation model divided into the main group and its subcategories.



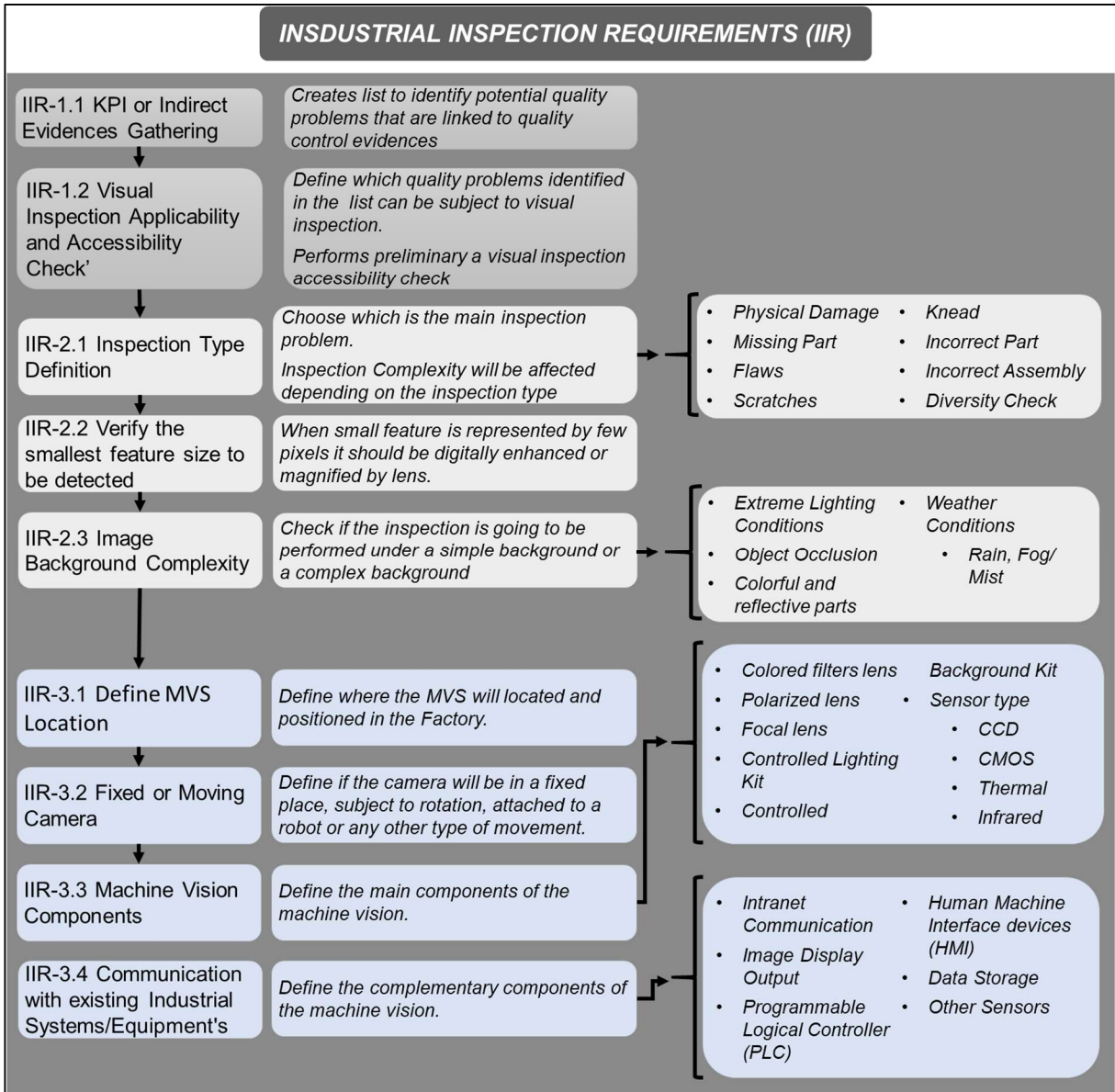
Source: The Author.

The first group is related to industrial inspection requirements (IIR). The second one is related to the artificial intelligence assessment (ASA). The third group is CNN specific details. This group is referred to only one type of AI possible applications, but it is the most relevant techniques currently being applied to image-based inspections. The last group is the AI MVS Management (AI-MNGT) that describes how the solution should be managed after the first three AI groups are implemented and validated.

4.1.1 Industrial inspection requirements (IIR).

The purpose of the IIR group is to assure that all requisites are correctly gathered and analyzed. These steps help the solution to fulfill the requisites from the quality control perspective (1. Quality Control Assessment), MVS technical constraints and industrialization requirements (2. Scene Constraints and 3. Technical Constraints Assessment). Figure 25 shows each activity of the IIR group with its description and details.

Figure 25 - IIR group with its main activities and simplified descriptions.



Source: The Author.

IIR-1. Quality Control Assessment

Quality control assessment is the first stage that a user must consider before selecting, installing, or purchasing any MVS solutions. This stage defines which are the quality control priorities, whether there are quality indicators or indirect evidence, whether these problems can be visually inspected.

IIR-1.1. KPI or Indirect Evidences Gathering

The main purpose of this activity is to relate to KPI's or indirect quality evidence with potential inspections. One must gather evidence of quality problems that affect the outcomes of a product or a process, whether it is a problem of high impact on the

final product or a problem that occurs frequently. A list must be created to identify potential quality problems that may be subject to visual inspection.

This step is not mandatory, but it increases the probability of the MVS investment payback and effectiveness. It also avoids random MVS placement around the factory without any valid pieces of evidence that the inspection may improve product quality outcomes.

IIR-1.2. Visual Inspection Applicability and Accessibility Check

Right after the potential quality problems list is created, it is necessary to check if there is a link between the source of the quality problem and possible visual inspection. There are quality problems whose source or the primary cause cannot be subject to a visual inspection to mitigate or reduce the incidence of this problem. After this link between the problem and visual inspection is made, the potential candidate's list is narrowed down to problems that can be visually inspected.

Another important aspect is the visual accessibility for inspection to be performed. Sometimes the problem can be visually identified, but there is no access for the equipment to be positioned or the right angle necessary to check that specific characteristic. This is a preliminary accessibility check because later in this model, there are some steps which will act as significant accessibility restraints and may turn the visual inspection technically unfeasible. Some MVS components, such as robots and complementary illuminations equipment's may not fit into the available inspection space.

In the case of multiple potential candidates, further investigation must be performed considering the MVS implementation costs, potential quality gains, and technological limitations.

IIR-2. Scene Constraints Assessment

Scene constraints assessment is the second stage in the model. This stage defines which are the main inspection problem types, identifying the smallest feature size to be detected and how the image background complexity may affect the MVS performance.

IIR-2.1. Inspection Type Definition

The objective of this objective is to define which are the main types of inspection problem. There are several types of inspections, and each one of them has its own

characteristics, which may affect the MVS hardware selection, image processing techniques, and AI technique. Common inspection problems are, e.g., physical damage, missing part, flaws, scratches, knead, incorrect part, incorrect assembly, product/part diversity check. Complex inspections may have more than one type of inspection to be performed with the same image or MVS equipment.

It is recommended to take some images of each detection type and link with each item in the visual inspection problem list.

IIR-2.2. Verify the smallest feature size to be detected

Some defects and quality problems can be so small that they require a magnifying lens or digitally enhanced images to make the inspection possible. Its import to provide the approximate size of the feature to be detected. This will allow an appropriate lens selection and what should be the distance between the MVS equipment and the inspected part. When a small feature is represented only by a few pixels, some MVS sensors or image processing techniques might not behave correctly.

IIR-2.3. Image Environment and Background Complexity

Depending on where the MVS will be installed, it is essential to know what it's the environment the image will be taken along with the rest of the image, as known as background information, that will not be subject to any inspection but will affect in the outcomes. In this step, one must check the potential location for the image MVS location, whether it will be internal or external.

Lighting environment conditions must also be verified because most of the MVS sensors are light sensitive, and most of them are not so adaptable as the human eye. Controlled lighting environment together with complementary illumination system can be an option for an internal MVS. Even in indoor facilities are subject to slight light fluctuations causing system malfunctioning (Gonzalez; Woods, 2008). External MVS must consider environmental effects of rain, fog, daylight, and night sensitive sensor depending on the application type.

Background complexity such as object occlusion, a color similarity between the inspected object and the background, and reflective parts must also be verified. Background information, if not considered, may also affect the detection performance. Light reflection problems may be solved by using a polarized lens. One way to reduce background complexity is to apply a known background through a controlled

environment solution. When this is not possible, the user must look for alternative in the image processing and available AI techniques.

IIR-3. Technical Constraints Assessment

Technical Constraints Assessment is the last stage of the IIR group. This stage helps to define where the MVS will be located, whether the MVS will be on a fixed or moving location, define the main MVS components and its complementary functionalities.

IIR-3.1. Define MVS Location

MVS location will be based on the scene constraints definitions and the visual inspection applicability and accessibility check. The user must evaluate all the gathered data and choose a location which makes visual inspection possible, minimizing environmental and background adverse effects. This location must be validated with the several departments of the factory because the MVS equipment may interfere with an existing process, existing equipment or it may require factory modifications which could turn MVS technically or financially unfeasible.

IIR-3.2. Fixed or Moving Camera

To define if an MVS will be stationary or subject to movement may affect in the number of the cameras required to perform an inspection. Using stationary cameras reduces the necessity of camera integration with other equipment responsible for the movement, but it provides a limited view. In some cases, multiple fixed cameras may be used instead of one moving camera.

Using a camera attached to moving equipment may enable the new field of views with improved detection possibilities at the exchange of increasing integration complexity and complementary equipment cost. Cameras may be attached to, e.g., camera lens zoom, a servo motor or stepper motor or robot. Moving cameras that are subject to sharp movement may suffer from equipment vibration or a blurring effect related to extended exposure time while the camera was in motion. All these constraints must be dealt with before equipment purchase.

IIR-3.3. Machine Vision Component (MVC)

A machine vision is composed of several components, such as sensor type, lens, controlled lighting, and environment kits. Defining the correct MVCs will directly

affect the inspection outcomes. Specialized companies in camera systems usually provide manuals and technical specifications to help final users to choose the best option. They also have a communication channel with a specialized staff which provides technical assistance during this phase. When most of the technical requisites are correctly gathered, it will allow a better selection of the MVCs.

Sensors. Sensors receive light from the inspected object. This light passes a light-sensitive sensor such as a charge-coupled device (CCD) or a complementary metal-oxide-semiconductor (CMOS). These sensors can be manufactured to detect specific light wavelengths. There is a sensor which captures visible light spectrum (Red, Green, Blue) or a specific infrared spectrum. Night vision and thermal imaging sensor are an application using specific wavelengths of the infrared spectrum. [41]

These sensor receives light and measures luminous packages from time to time with that information a numeric pixel intensities value is given, this phenome is known as quantization. Typically the pixel is 8 bit long, which allows 256 combinations of light. [41]

Lens. There is three application to a lens in an MVS. The first one is to modify the lens magnification effect and its field of view. The second application is to filter a determined type of wavelength, also known as colored filters. They allow only a specific light wavelength range. The third application is the polarity lens filter.

Controlled Lighting Kit's. When environmental light does not deliver satisfactory outputs, controlled lighting kits are necessary to control the lighting conditions locally. These kits can provide specific light inputs to highlight some defect type , to provide inspection conditions for low light scenarios or to combine the effect of some light spectrum with a colored filter lens. Light exposure time, intensity, and specific wavelength output are some of the control options in a kit.

Controlled Background Kit's. There are solutions to provide a known color background can be applied to combine with image processing techniques to remove any negative influence. Another option of controlled background is to enclosure the region of inspection into a dark chamber, minimizing effects of external light.

IIR-3.4. Communication with existing Industrial Systems/Equipment's

Every MVS must consider its integration with other industrials system, so they can communicate with them and company staff. To their integration, it is necessary to gather communication protocols of each system and assure that information is not lost,

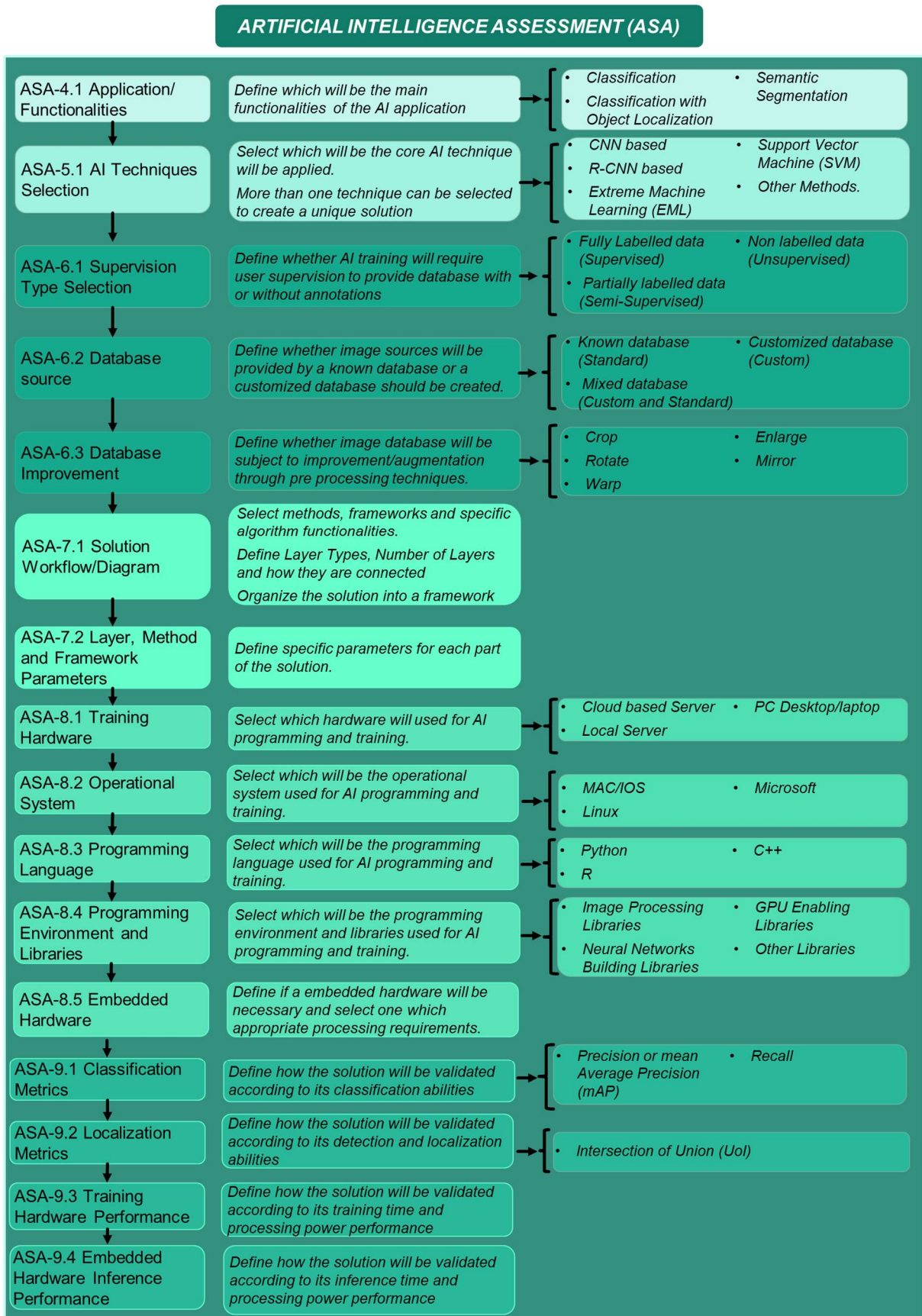
corrupted, or misunderstood. Typically, an MVS must communicate with an image display output, human-machine interfaces, programmable logic controllers, intranet, data storage systems, and other sensors. The activity goal is not to provide details on how to create communication between equipment's and systems, but to inform that communication is an essential technical requisite. Depending on the communication protocols complexity, the existing communication hardware may not meet *all* the requirements, and it will become a project constraint.

4.1.2 Artificial Intelligence Assessment (ASA).

The purpose of the ASA group is to assist the user in defining the most appropriate AI solution that meets the requirements gathered in the previous group phase. New requirements will be determined in this phase, which may require IIR requirements update to make them compatible.

The ASA group was based on the evaluation model applied during the content analysis in early in this document. The first part is to define what type of solution will be applied (4. Solution type selection). This selection narrows down the selection of the AI technique (5. AI technique for MVS) because there is architecture which is optimized according to the solution type required. Defined the AI technique, the user will be able to select the most appropriate way to train the AI model (6. AI training requisites) and define which are the starting hyperparameters for the solution (7. AI Hyperparameters). Given the AI complexity, one can select the most appropriate programming environment and libraries along with appropriate hardware to train and deploy the final solution (8. Hardware/Software for AI). The whole solution must be submitted to a validation process given the required performance metrics (9. Validation Metrics). Figure 26 shows each activity of the ASA group with its description and details.

Figure 26 - ASA group with its main activities and simplified descriptions.



Source: The Author.

ASA-4. Solution type selection

Solution type selection is directly related to the Inspection Type Definition. There are several solutions and multiple functionalities which can be applied. It depends on what should be detected and how complex the inspection context is (Yu *et al.*, 2017). Since new solutions and functionalities are constantly being developed, and the existing ones are still evolving, four main types of solutions will be described as examples.

ASA-4.1. Application/ Functionalities

Classification. Classification applications analyze the input image and provide a probability output which determines what the model is seeing. The output could be a simple one, as a binary output (e.g., The solution will evaluate if one component is good or defective.) or a more complex one with multiple output classes (e.g., The solution must evaluate if which component is being applied in a list of several possibilities). Classification complexity increases with the number of desired classes to be detected. This solution is not recommended when there are multiple classes that must be classified simultaneously in the same image input (e.g., classification of multiple components being assembled). This type of inspection does not consider the localization of the detection since it evaluates the whole image. The user must provide labeled images to create a classification system.

Classification with Object Localization. This application is derived from the pure classification model. It is a more complex model because it requires the object localization functionality. The input image is analyzed, and the localization of the detected object in the image is also given along with the classification probability output. Its complexity also increases with the number of classes to be detected, and it can perform multiple classification tasks simultaneously in the same image. When there are different classes overlapping each other or object occlusion, the system may not perform accordingly. Object localization is typically shown through a colored rectangle over the detected object, also known as the bounding box. Another limitation is that inside the bounding box, there will be image pixels that may not belong to the detected object, which in some cases are not the desired outcome. The user must provide images with labeled bounding boxes to create a classification with an object localization system (Yu *et al.*, 2017).

Coarse and Fine Classification. Depending on the complexity of the application, the number of classes and image features will increase. This increased

complexity may turn a single model to become computational extensive and with its performance significantly reduced. It is possible to create a combination of NN working in series with different purposes to reduce the complexity of a single NN. The first NN will provide a coarse classification to detect objects groups (e.g., detect cars and trucks). A second coarse classification can be applied only over the detected objects, providing more details for the object group (e.g., detect what the car brand is). The last NN could be used to perform information a fine-grained classification to extract very specific information about the object (e.g., detect what the car model) (Zhang *et al.*, 2017).

Semantic Segmentation. It's a pixel-wise classification and localization model. Its localization performance tends to be more precise than the standard object localization. In the other hand, the system requires an improved labeling process which consists in provide annotations at the pixel level. Image annotation is one of the drawbacks of semantic segmentation since annotation at this level of precision is usually human-made and it may be time-consuming. This system can be used to detect very small types of defects and provide a more precise localization. If the image annotation is not performed properly, the system may not work properly also.

ASA-5. AI technique for MVS

AI technique is directly related to the solution type selection. Each technique can be considered as a building block of the whole solution, with each one of them providing specific functionality. This is one of the reasons that AI MVS are usually tailor-made because there are several possible combinations of functionalities which could be converted into a unique solution. As verified early CNN and R-CNN based solutions are the most used techniques to MVS applications. They consist of a pack of several AI techniques arranged into a workflow that extracts relevant image features, find a correlation between these features, and allows a proper classification.

ASA-5.1. AI Techniques Selection

CNN based. CNN is a deep multi-layered ANN which is related to pure classification models (Goodfellow *et al.*, 2016). CNN can be divided into four steps:

- I- Convolution: Receives images inputs and extract relevant image features.

- II- Fully Connected: Receives extracted features and search for correlation between them.
- III- Classifier: It receives correlated features outputs and converts them into classification probabilities.
- IV- Learning: Compares the label inputs with the classification probabilities given by the model and updates it.

This step cycle is repeated until the error between label input and classification output, also known as model loss, achieves a minimum satisfactory value. A CNN solution will be detailed in the last group of the implementation model. CNN can also be used for semantic segmentation solutions, but it requires some modifications in the base model.

R-CNN based. R-CNN based solutions are related to classification with localization models. R-CNN has an additional step when compared to a CNN framework. This step usually happens between the convolution and the fully connected step, and it is responsible for providing Region of Interest (RoI) for object location. This new step will provide a predicted bounding box over the detected object in the image and its classification probability. Semantic segmentation with R-CNN is also possible given the adequate modifications in the base model.

Other Methods. There several other techniques that can be added to the core CNN and R-CNN techniques providing additional functionalities, or they can be used to create a unique solution. A few examples are stated below with a potential application into AI MVS:

- Support Vector Machines (SVM) can be used in the classification phase of a CNN, replacing the softmax functions.
- Extreme Machine Learning (EML) can also be used to replace the softmax layer, but it can also be used to replace the hole CNN architecture, given specific input requirements and classification output limitations.
- Generative Adversarial Networks (GAN) can be used to improve image dataset through the creation of synthetic data, or it can be used to reconstruct CNN features output.
- AI bioinspired techniques can be used to optimize model parameters, improving model performance and reducing the time required to optimize and fine-tune models.

ASA-6. AI training requisites

AI training defines how the selected technique will be trained. Training is related to the selected AI technique and its solution type. It is also related to the image dataset availability, what kind of image annotation is required, how many images are necessary, and whether the images will be required preprocessing operations.

ASA-6.1. Supervision Type Selection

Selecting supervision type is related to how data is provided to the AI solution. Data can be provided in two forms, raw data or labeled data. Raw data is obtained directly from machines or equipment's (e.g., camera), without it being modified or analyzed by a human. While labeled data can be obtained through the interpretation of raw data (e.g., search for specific objects in the image) followed by an annotation process based on the extracted information (e.g., add classification label and draw a bounding box around the object) (Goodfellow *et al.*, 2016).

Based on these definitions, supervision type can be divided into three groups: supervised, semi-supervised, and unsupervised.

Supervised Training. All dataset provided for training is labeled. CNN requires a label class for each image, without informing where the label is located in the image. Besides the class label, R-CNN also requires the corresponding bounding box with its coordinates .

Semi-supervised Training. Part of the dataset provided for training is labeled, while the rest of the dataset is not labeled. This type of training usually requires an in-built algorithm which can automatically label images without human aid. This labeling process could be performed by an already trained AI solution, or a complementary dataset could be created through a GAN solution.

Unsupervised Training. This type of training uses raw data only. This kind of solution extracts features from each image in the dataset and find patterns that allow the solution to split and classify images (Lin *et al.*, 2016).

ASA-6.2. Dataset source

Image dataset source is essential to create a reliable AI solution with a common base to evaluate them and still be adaptable to suit its final use. The dataset that is available on the internet is considered standard, while a unique dataset that is created for a particular application is called custom.

Standard. These datasets are preprocessed, and most of them are already labeled. They are commonly used to analyze a new algorithm's performance because they provide a standard base of evaluation. Another application for them is to use them to pre-train an ANN and create an initial feature extraction of a more complex solution. Microsoft COCO, PASCAL VOC, ImageNet Large Scale Visual Recognition Competition (ILSVRC) datasets are some type of standard dataset that is available on the internet[52]. Typically a standard dataset has a balanced number of images for each detection class.

Custom. These datasets are created when the standard ones may not have the required information to train the model. In order to fit them into an AI solution, they must be subject to preprocessing steps (e.g., cropping, resizing, rotation, color map operations), labeling steps (Guan *et al.*, 2017). The amount of image may vary from case to case, but a dataset with few images or an unbalanced number of images per class may create an unreliable model.

Standard and Custom. It is usual to use both types of dataset source to create a unique and robust solution. The customized dataset must be adapted to have the same inputs as the standard one (Sun; Xiao, 2016).

ASA-6.3. Dataset Improvement

Dataset improvement, also known as data augmentation, is a technique that creates new images based on an existing image dataset. There are several purposes which this improvement is required. It can be used to solve the dataset with unbalanced classes or simply to verify if the model performance increases with a more extensive source of images. Another application to dataset improvement is to increase model sensibility to the rotation. An example of this case is when the input training data considers the object in one possible rotation, but in the real case, the object may appear rotated. (Yao *et al.*, 2018). There cases where 3D models are used to create a 2D dataset from multiple views, which also increase model robustness (Puttemans *et al.*, 2018).

ASA-7. AI Hyperparameters

AI Hyperparameters covers the steps needed to organize several programming methods and specific algorithm functionalities into a unique framework. A solution framework is composed of several different layers, methods and algorithms, and specific hyperparameters.

ASA-7.1. Solution Workflow/Diagram

To better visualize how the AI solution will work, a workflow or diagram of the framework must be provided. It facilitates solution replicability and future changes. The diagram must provide the framework layout and how layers are connected.

Layer Connection and Framework Layout. The layers are connected to each other and how they are positioned affects the framework will work (e.g., a CNN is composed by multiple convolution layers, and its outputs are connected to fully connected layers.).

ASA-7.2. Layer, Method and Framework Parameter

Method and Algorithms. These are the main programming functions that compose a solution. (e.g., the convolution operation over an image is a method). Combining and sequencing different methods in NN creates layers (Young *et al.*, 2018).

Method and Algorithms Hyperparameters. Each method and algorithm has input values which must be defined by the user (e.g., kernel size of the convolution or number of neurons in the fully connected layer). Most of the known methods contain pre-set or recommended starting values to avoid an incorrect selection of parameters. In the other hand, AI solution fine tuning can be achieved by changing and optimizing these starting values.

Layers. They are composed of methods (e.g., the first convolution layer can be composed by a convolution, an activation function, and a pooling while the last convolution layer may not have the pooling functionality).

ASA-8. Hardware/Software for AI

All AI solutions rely on adequate hardware and software. Hardware gives the necessary processing power to allow complex algorithms, parallel processing, faster training, and faster model inference. Software is the base to create new AI algorithms

and implement existing programming methods. Both represent core elements of an AI solution, and their technical requisites are interconnected. There are two types of hardware and three software requisites that should be considered while selecting an AI solution.

ASA-8.1. Training Hardware

Training hardware is the first part of an AI MVS solution. Either Multi-Core Computer Processing Units (CPU) or Graphic Processing Units (GPU) can be used to train AI solutions. GPU architecture was designed to work image data, and it can be used to perform more complex calculations. Shi *et al.* (2016) performed a benchmark of existing DL frameworks with different configurations of hardware. It was observed that GPU computing is faster than models trained by CPU only.

The first version of GPU's were not designed for the purpose of training and deploying AI models at first, but newer architectures are developed for this purpose.

Training hardware can be a PC or Laptop with a GPU, a local server with multiple GPU's or a cloud server.

PC or Laptop with GPU. They can be used to create first AI models, programming new algorithms, technological experiments, and applying new Proof of Concept (PoC). Training a complex AI model can be time-consuming and, sometimes it might not be possible to run due to lack of memory. Training AI with less powerful PC's may require days so that the model can be completed. A simple PC can be used if the application does not require a constant model update or newer training. It can be used when training time is not a constraint.

Local Server. A local server can be built or purchased using multiple GPUs. A local server is a robust option to train very complex models without worrying with model complexity or extended training time. Since it does not rely on the internet for training, it has an advantage over cloud-based servers. There are companies which already provide ready to go local servers with the necessary support to integrate it with existing communication protocols. It is also possible to use a customized local server.

Cloud Server. A cloud server can be a solution when a PC is not powerful enough or acquiring a local server is not a viable option, or the cost of processing power and data storage are satisfactory. Model complexity and training time are not technical restraints, but rental costs may increase with longer time to train, or more processing power are needed. It is possible to combine PC/Laptop with a cloud solution

when a once-off complex training is promptly needed. Another possible combination is cloud and local servers, where the cloud is used when the training queue exceeds a certain threshold (Agrawal *et al.*, 2015).

ASA-8.2. Operational System

The operational system (OS) should be considered during the AI software programming phase, AI software deployment phase, and the management of the inspection outputs provided by the AI solution. The OS affects the availability of programming tools, libraries availability, and compatibility. The last phase should match with the operational system, which is currently being used. It is possible to create a solution with a different OS in each phase, but a compatibility analysis must be made. The three most common OS are Windows, Linux, and macOS, but each one of them has many versions of its OS so that must be considered during software integration.

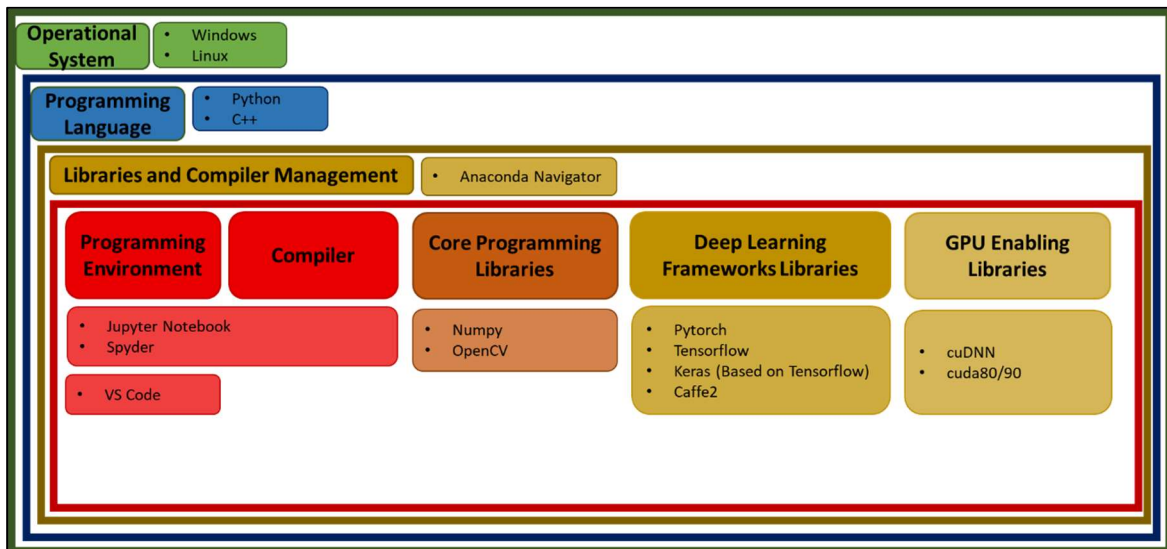
ASA-8.3. Programming Language

The programming language is the backbone of any programming activities. Each programming language has its own characteristics which provide different processing performance with the same method. There are several programming languages available, but Python, C, and C++ have a broader programming repository already available for ML/DL and computer vision. A company normally choose its programming language for creating an AI solution according to the availability of trained professionals, ongoing support for the existing and development of new programming tools, and stability of the current versions. Since programming languages are always changing and evolving, a company should constantly be checking for newer and better options without dismantling existing solutions.

ASA-8.4. Programming Environment and Libraries

The programming environment and its libraries are the programmer user interface tool which allows him to create new algorithms and test them. Figure 27 shows an example of how the programming environment and its programming libraries are related to the OS and the programming language.

Figure 27 - Correlation between the main components for AI programming.



Source: The Author.

Programming environment. It is created by the programmer with the desired programming libraries. Just like the programming language, the programming environment usually is a personal choice of the programmer. A programming department may define a standard environment for management purposes. Some programming environments help to organize the management of the current programming language version and

Programming library. It has built-in methods and algorithms which facilitates the development of new solutions. One programming library may be compatible with different OS and programming languages, but they usually are specific to an OS and programming languages. Libraries are constantly evolving, and newer versions are being released. Keeping track of libraries version is very important when developing AI solutions because a simple version update may affect the whole solution.

Several programming libraries have been developed to allow DL models to communicate with CPU and GPU, such as cuDNN (Chetlur *et al.*, 2014), Caffe (Jia *et al.*, 2014), and Tensorflow (Abadi *et al.*, 2016). They allow researchers and developers to build new deep learning models without being concerned with scheduling data processing, on-chip memory placement, register blocking, and basic linear algebra subprograms (BLAS) calculations. GPU accelerating libraries have its specific version compatibility according to hardware architecture; any change in one of them may cause algorithm malfunction (Abadi *et al.*, 2016).

Developed AI algorithms should also be submitted to version control. Otherwise, any non-validated updates may compromise an existing solution.

ASA-8.5. Embedded Hardware

Embedded hardware is the last part of an AI MVS solution. They are used to perform model inference using smaller hardware with reduced performance, but with reduced costs. It is possible to use the same hardware to train and deploy an AI solution. A deployed solution requires less processing power, so it will consume only a fraction of the processing power used during training (Abadi *et al.*, 2016). To avoid unused processing power, it is better to use simpler hardware for less complex applications.

Embedded hardware is recommended in the following cases:

- When latency between AI model inference and Local/Cloud servers exceeds the time required for inspection.
- When multiple simultaneous inspections are required, and the existing hardware does not support it.
- When MVS has limited working space and the training hardware will not fit in.
- When implementation costs are limited and using multiple training hardware will be expensive in a multiple inspection scenario.

Complex AI models are still a challenge for embedded hardware. These complex models can achieve better detection performance, but they require a lot of processing power. Embedded hardware usually does not have enough processing power for complex models. This model can be simplified, but they will eventually reduce detection performance. Applications using embedded hardware should be validated in accordance with each type of inspection and its specific requirements.

ASA-9. Validation Metrics

Validation metrics are used to verify if the solution outcomes will meet the desired quality requirements. Each new algorithm, framework, or solution must be validated to assess if there was any performance improvement. Image-based solutions can be validated through classification metrics and object localization (Zhang *et al.*, 2013). Training time and algorithm deployed performance are metrics, which also appears in Deep Learning industrial applications.

ASA-9.1. Classification Metrics

Classification problems use the concept of the Confusion Matrix (CM), which compares the real data input with the classification model output. There are four possibilities of outputs in a confusion matrix. Figure 28 shows the four types statement which a confusion matrix can provide (Zhiqiang; Jun, 2017)

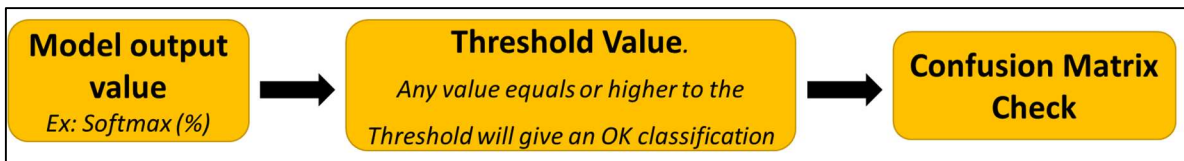
Figure 28 - Confusion Matrix Example.

		Real Data Input	
		Positive	Negative
Model Output	Positive	True Positive (TP) (Models correctly detects/classificate a “good” input)	False Positive (FP) (Models wrongly detects/classificate as “good” a “bad” input)
	Negative	False Negative (FN) (Models wrongly detects/classificate as “bad” a “good” input)	True Negative (TN) (Models correctly detects/classificate “bad” input)

Source: Adapted from Zhiqiang; Jun (2017)

The threshold value is a parameter set by the AI MVS user which will modify the confusion matrix output without retraining a whole model. Figure 29 shows that the threshold value is applied right after the classification output value.

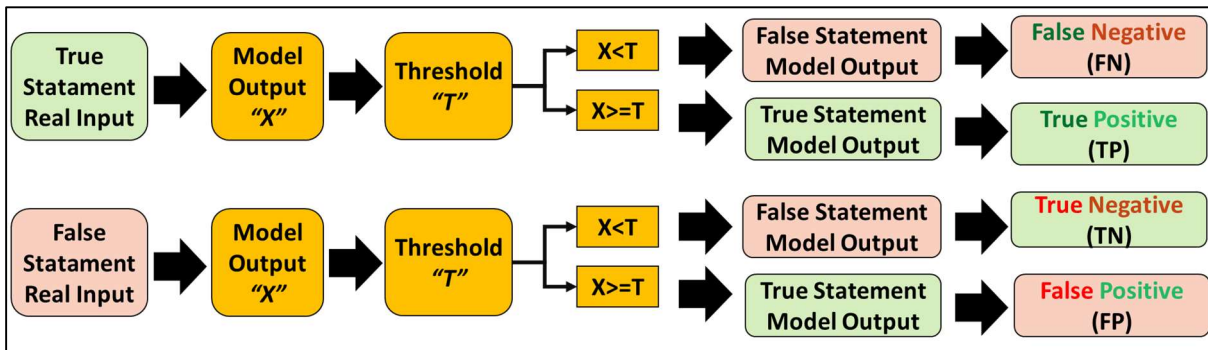
Figure 29 - Relationship between the model output value and the confusion matrix given a threshold value.



Source: The Author.

The threshold value affects directly in the four main outputs of the confusion matrix. Figure 30 shows how an upper limit threshold value can affect the false negative (FN), false positive (FP), true negative (TN) and true positive (TP).

Figure 30 - The four types of possibilities of a confusion matrix given threshold value.



Source: The Author.

If the threshold value "T" is larger than the average value of true statements given by the model output "X", there will be increasing value of the false negative (FN) outputs. An example of this scenario is an AI MVS quality control to detect defective parts. The system will be incorrectly informing that good parts may have defects when they do not have any. In the other hand, a high threshold value decreases the chance of false positive (FP) outputs. This trade-off between FP and FN should be evaluated by the user depending on the final application of the AI MVS classification model.

Classification accuracy considers the number of correct statements the model performs compared with the whole data input. Equation 17 and 18 shows how accuracy can be calculated.

$$TT = TP + FP + FN + TN \quad (17)$$

$$\text{Accuracy}\% = \frac{TP + TN}{TT} \quad (18)$$

Receiver operating characteristics (ROC) graph is another option to use CM information to evaluate classification performance. It uses FP and TP values to plot performance curves for each algorithm (Fawcett, 2006).

Precision-Recall graphs can also be used to evaluate classification performance. Recall evaluates how many true statements the model achieved from the total possible true statements. Values closer to one mean that the model is detecting all true statements, so FN tends to zero. Equation 19 shows how the recall is calculated. (Davis; Goadrich, 2006).

$$\text{Recall}\% = \frac{TP}{TP + FN} \quad (19)$$

Precision evaluates how many true statements were given along with wrong statements. Values closer to one mean that the model is providing good statements, so FP tends to zero. Equation 20 shows how precision is calculated (Davis; Goadrich, 2006)

$$\text{Precision}\% = \frac{TP}{TP + FP} \quad (20)$$

There are other classification metrics (e.g., F-Score, F1-Score, OTA, OTP) that combines precision, recall, and accuracy outputs into new metrics (Sunkara *et al.*, 2018) that can be used to evaluate an AI solution.

AI MVS solutions usually require a model that classifies multiple objects or products. Achieving a maximum classification metrics value is very challenging when multiple objects must be classified by the same model. During the learning process, a model classification performance may be optimized to some specific object while it lacks for the other objects. To check the performance of the solution for all objects, an average value must be calculated. Equation 21 shows an example of the Mean Average Precision (mAP) value that can be used for multiple object classification solution.

$$(\text{mAP})\% = \frac{\text{Precision}_{\text{obj.1}} + \text{Precision}_{\text{obj.2}} + \dots + \text{Precision}_{\text{obj.n}}}{\text{Number of Object or Classes (n)}} \quad (21)$$

ASA-9.2. Localization Metrics

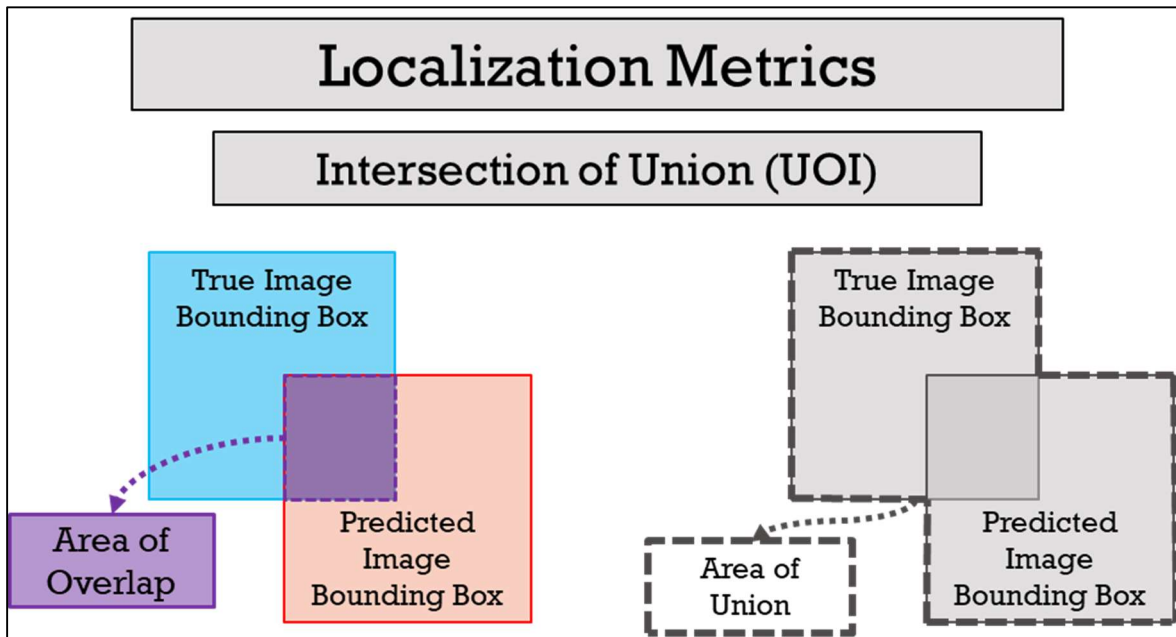
Localization metrics were already used in traditional machine learning techniques, such as SIFT, feature matching, and others. It became more significant with the region based CNN (R-CNN) architecture proposed by Girshick *et al.* (2014). This new type of algorithm architecture requires not only an accurate classification but also that a bounding box to place over the detected object.

To validate the proposed bounding box location, a comparison must be made with the true bounding box, also known as ground truth. The combined area of the ground truth and the predicted box is known as the area of union. The intersected area of the ground truth and the predicted box is known as the area of overlapping. The Union of Intersects (UOI) is the relation between the area of overlapping is divided by the area of the union, as in Equation (5).

$$\text{UOI} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \quad (22)$$

The closer UOI is to one, more accurate localization your algorithm will have. Figure 31 shows the main variables of this localization metric.

Figure 31 - Intersect of Union localization metric.



Source: Adapted from Girshick *et al.* (2014)

ASA-9.3. Training Hardware Performance

Training hardware performance for this document was considered as the time required to train the same AI model comparing them with available hardware. Normally, modern hardware's performs better than older ones, training a model much faster.

Normally, for academical purposes, training time is not a restriction, but in an industrial scenario, multiple instances must be trained simultaneously. Higher training times or a large training queue may delay new inspection demands and become a blocking point for AI MVS. Training time for industrial will affect in the hardware selection cost's or cloud computing cost's which is normally charged by consumed computational time.

ASA-9.4. Embedded Hardware Inference Performance

Deployed Inference Performance considers the time needed to analyze an image or a video in real time given an existing trained model. The inference performance should consider the speed required for the inspection to be performed, the frames per second input (FPS) and the camera resolution height and width. Higher resolution models normally require more time to perform inference. If the FPS input is higher than the inference performance, the model will not be able to process every input given by the camera. The model may not achieve the desired performance if the required time for inference is higher than the inspection time.

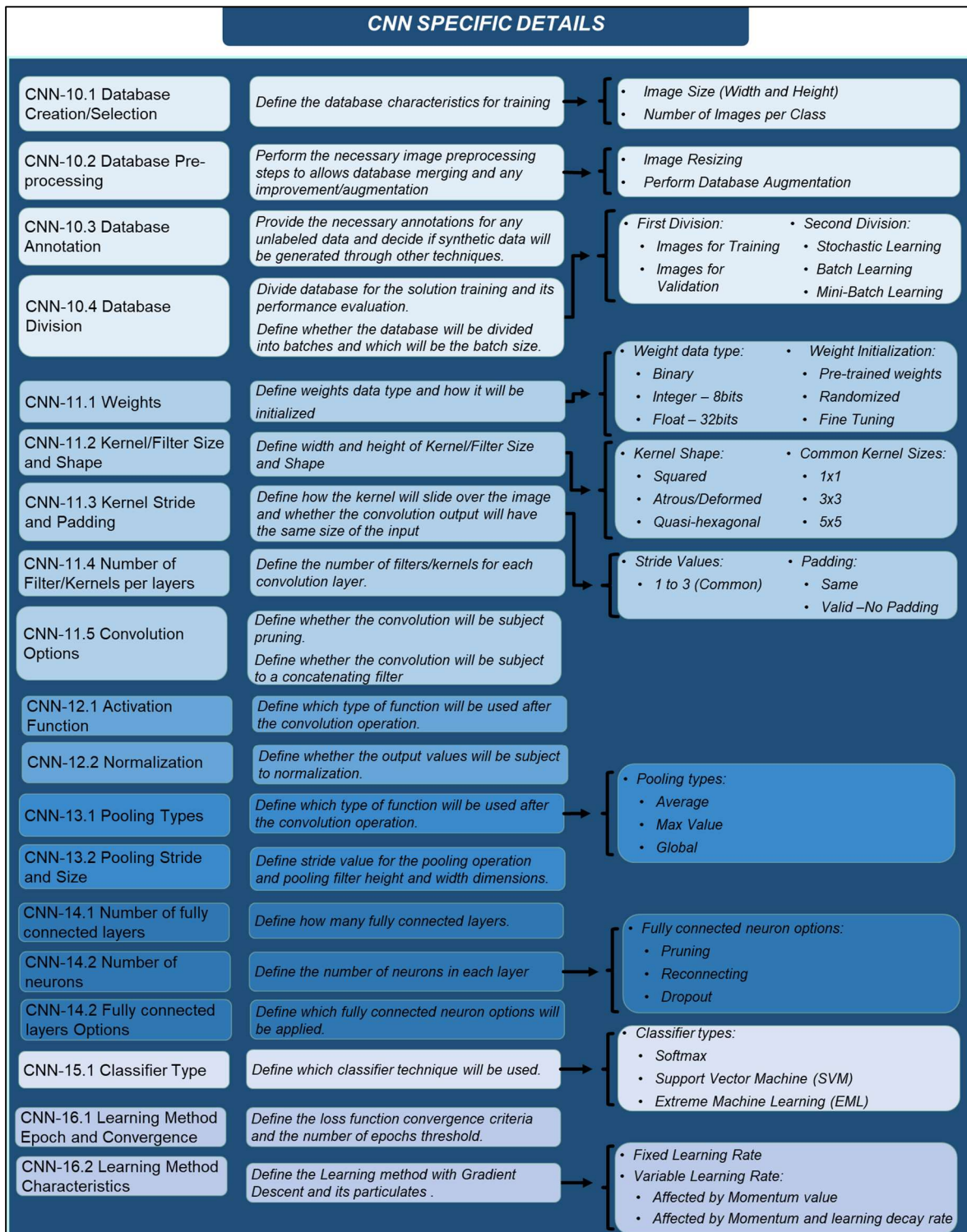
The same hardware adopted during the training phase can be used for inference, and they usually have better inference time performance than embedded hardware. Older R-CNN algorithms were capable of processing only 1 FPS, while You only look once (YOLO) achieved 45 FPS (Redmon *et al.*, 2016) and its newer version achieved up to 90 FPS without losing accuracy (Redmon; Farhadi, 2018).

Embedded hardware is an option to perform inference when training hardware is not viable. Compared to the training hardware, they have slower performance, but they are a more affordable option with better mobility and scalability when multiple simultaneous inspections are required. Inference time with recent embedded hardware can achieve 3 to 4 FPS with SSD and YOLO algorithms (Sunkara *et al.*, 2018). Research and development of specific embedded hardware is an ongoing subject. This new hardware should allow faster inference time with smaller hardware and meeting industrial requirements.

4.1.3 CNN specific details (CNN)

The purpose of this group is to provide the user with basic knowledge of the CNN main requisites, defining how the final solution will work. Figure 32 shows each activity of the CNN group with its description and details.

Figure 32 - CNN specific details group with its main activities and simplified descriptions.



Source: The Author

The group begins with the dataset requirements (10. Dataset), then it is followed by the convolution layer requirements (11. Convolution Operation, 12. Activation Function and Normalization, 13. Pooling Operation). After the convolution, the fully connected layer (14. Fully Connected Layer) along with the classifier (15. Classifier) are defined. The last part of this group is the learning method (16. AI Learning Method).

CNN-10. Dataset

Image dataset is the main input to CNN's learning process. The learning mechanism will depend on how the data is provided to the AI solution. The user can define if a customized dataset will be created or an existing data will be used, deciding if the dataset requires image preprocessing, data annotation and how the data will be divided during training and model validation. This section will give details on how to structure a dataset to be properly fed into a CNN.

CNN-10.1. Dataset Creation/Selection

The first step of a dataset is to define if it will be created, selected, or both options. Regardless of its origin, a dataset input must have a standard image height and width input to allow CNN models properly. The number object/classes and the number of images for each class are also important to assure the model has enough data to learn the characteristics of each class.

Image input size could be as small as the MNIST dataset with a 32x32 input, an intermediate input with 300x200 as the Caltech101 dataset or a larger input with 800x600. The smaller the dataset width and height, the faster they are to train, but larger images can provide more relevant information depending on how the CNN is built.

The number of classes and **number of images per class** should be defined by the user while constructing or selecting a dataset.

Images per class. A larger number of images per classes increases the chances of the model, learning more relevant data and increasing its classification performance. Datasets with fewer images per class can be subject to two problems. The first problem is overfitting. Overfitting in CNN happens when the model learns specific information of the images given by the dataset, but it lacks the ability to recognize any other images of the same trained class during model validation and inference. The second problem may happen with unbalanced datasets. AI solutions with a dataset with some classes with very few images than other classes may not learn properly. Depending on how the model updates its learning, an unbalanced class may be treated as an outlier, where the relevant characteristic may be discarded by the model or its relevant features will not be learned. It is recommended that the number of images for each class is almost the same.

The number of images will vary depending on the application, and they can be increased or modified while the model is being trained and validated. The CIFAR-10 is an image dataset with 60,000 images divided into ten classes, and each image has a size of 32x32. MS-COCO dataset has more than 330,000 images divided into 80 classes.

Number of classes. The more classes a model has, the more complex it is to learn image features of each class. Optimized CNN's for a certain number of classes may suffer from decreasing mAP when the number of classes is increased.

Selection or Creation. For industrial inspections applications, it is recommended to create its own dataset because each inspection will involve images of a part, component, assemblies or process which are not common in existing datasets. Selecting and using existing image datasets are useful for other purposes such as algorithm performance evaluation and optimization due to available benchmarks in the academic community. A selected dataset could be also used for pre-training models, where learned features could accelerate the fine-tuning process with a customized dataset.

CNN-10.2. Dataset Pre-processing

Raw images gathered from cameras or images from existing datasets are normally submitted to preprocessing which involves cropping and resizing. These preprocessing steps are used to normalize image inputs, selecting the relevant information required for training. Preprocessing can also be used to create augmented images, though color changes, image rotation, warping, and scaling operations. Augmented images are an alternative for balancing datasets that has fewer images inputs. Preprocessing steps are important to assure that trained data will gather relevant image features without harming the detection performance (Guan *et al.*, 2017)

CNN-10.3. Dataset Annotation

Dataset annotation is the process to add labels and other relevant information for the learning step. There are three major types of annotation that can be added to an image depending on the AI solution. Image classification models require that the image input has a label describing which class that image belongs to. Object detection requires the bounding box image coordinate along with the class it belongs. Semantic

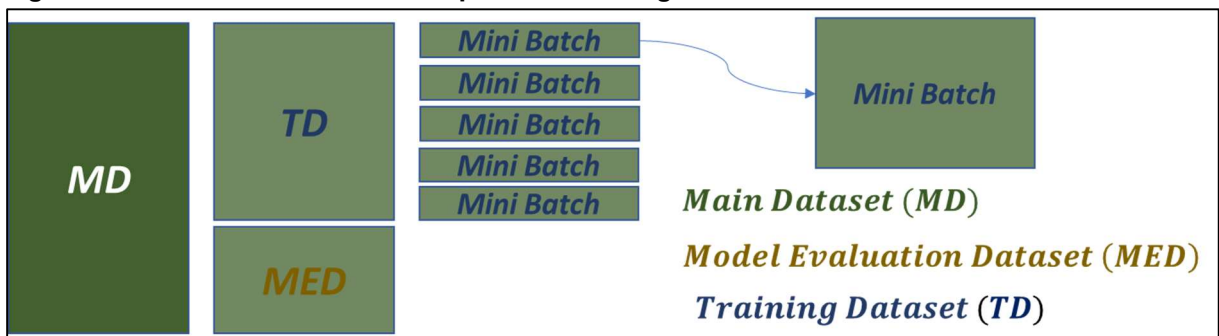
segmentation models require a precise drawing of the contours of the desired detection class along with the class it belongs.

The annotation process is normally performed by the user, but there are non-AI algorithms and AI trained solutions that could be used to accelerate the annotation process. This annotation algorithm should be submitted to an evaluation of the user because an incorrect labeling may impair the learning process.

CNN-10.4. Dataset Division

Dataset division is the last part of the process where data can be composed of several datasets split into smaller ones. Figure 33 shows the main dataset (MD) is divided into the training dataset (TD) and model evaluation dataset (MED), then the training dataset is divided into several mini-batches.

Figure 33 - Dataset division smaller parts for training and model evaluation.



Source: The Author

The first step divides the main dataset into two, the training dataset and the validation dataset. They are important to assure that the model is trained and validated properly. During splitting the user must check if unbalanced datasets were created during the splitting activity. The validation dataset should be composed of images that were not used during the training phase. This is important to compare model performance during training and validation. AI solution overfitting can be detected if the model achieves satisfactory performance in the training phase, but it achieves a poor performance during validation. In this case, the dataset must be verified and improved along with some modifications in the model hyperparameters.

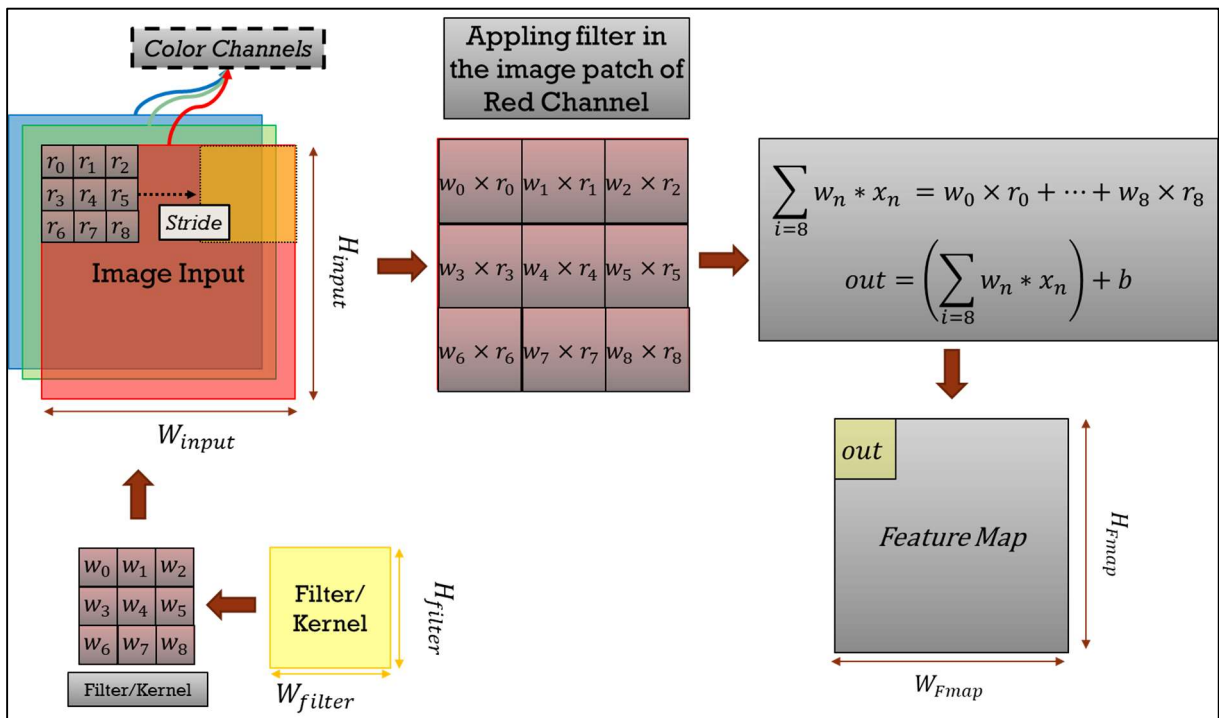
The second step is to divide the training dataset into smaller parts, also known as batches. Current CNN models and data used for training are fully uploaded into the hardware, but if the hardware does not have enough memory, the model will crash. To avoid model crashing, the user may use more powerful hardware, or it could divide the

dataset into batches. This division affects the learning process, and it will be discussed later in a specific topic.

CNN-11. Convolution Operation

The convolution operation process an image input from the training dataset will be subject to. It is important to distinguish the convolution layer between the convolution operation. The convolution layer is normally composed by the convolution operation, the activation and normalization functions, and the pooling operations. Figure 34 shows the initial part of a convolution operation over an image.

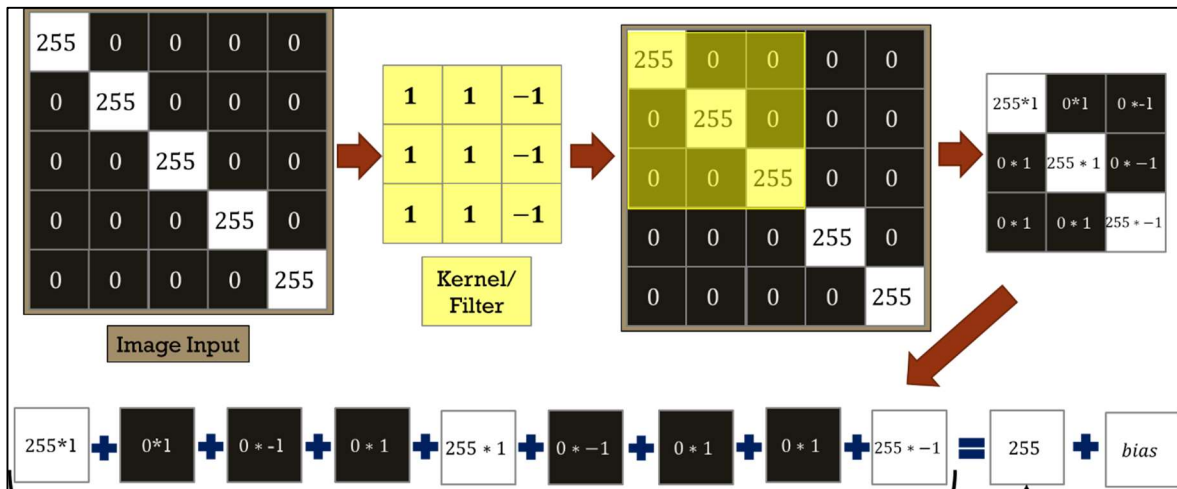
Figure 34 - Example of a generic kernel applied over an image input and its respective feature map.



Source: The Author

The convolution operation consists of a kernel, also known as a filter, applied over an image input or feature map coming from a previous convolution layer. The kernel is a small matrix with a given height and width, and each value of the matrix is known as weights (w_n). The output values of a features maps are composed by the summation of the multiplication of kernel weights and the inputs values, and a bias value is added to the result of this summation. Figure 35 shows an example of the convolution operation during the first stride

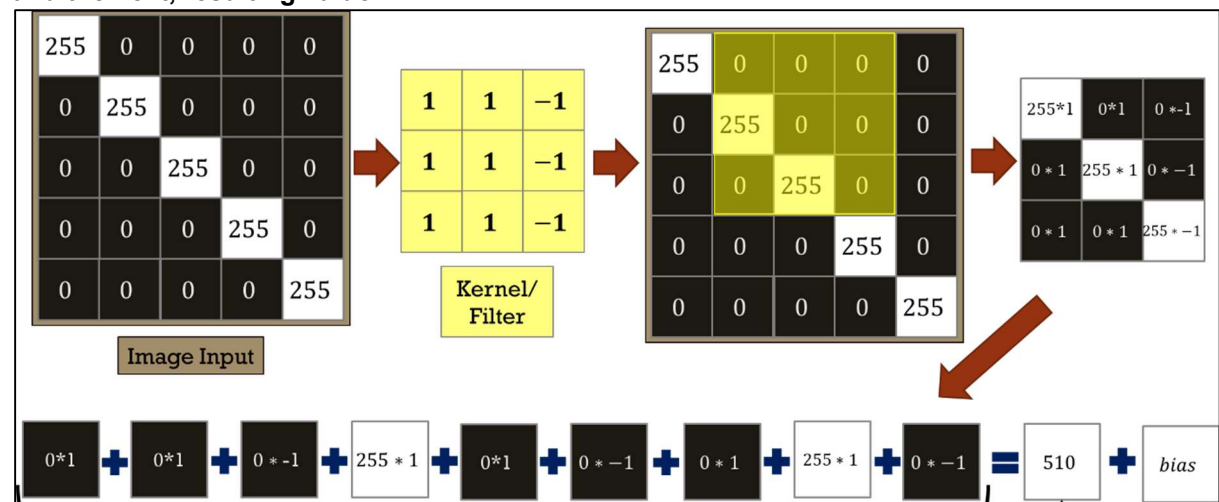
Figure 35 - Example the mathematical operation of the kernel over the input and the first resulting value.



Source: The Author

This process is repeated as the kernel sweeps over the image input given a stride value, where the image inputs will change according to the stride value, but the weights and the bias remains constant. Figure 36 shows the following stride over the same image.

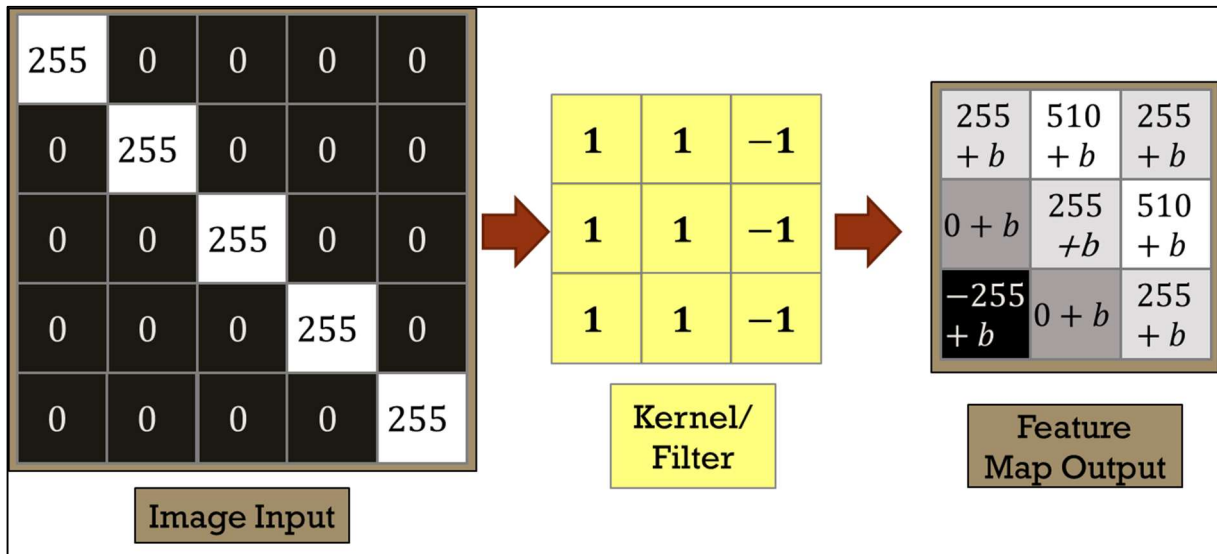
Figure 36 - Example the mathematical operation of the kernel over the input given a stride of one and the next, resulting value.



Source: The Author

The feature map output size will be affected by the kernel size, the stride value, and the padding (Goodfellow *et al.*, 2016). The feature map is a matrix containing the output of a convolution operation, and each filter will provide a different feature map give the same input. Figure 37 shows a full feature map values from a convolution filter.

Figure 37 - Example the complete mathematical operation of the kernel over the input and all the resulting values.



Source: The Author

The values obtained in each feature map represents how relevant they are to the learning process. Positive values can be considered as important features, while a negative value is not.

CNN-11.1. Weights and bias

Filter weights are one of the variables which are optimized during the learning process. These values can be started randomly, imported from previous training, or be set manually by the user.

The values of this weight are normally assigned to 32 bits floating point, but recent CNN research shows that 8 bits value can be assigned instead of the previous one (Truong *et al.*, 2018). To change from 32 to 8 bits, data type can reduce mAP, but it increases processing speed performance. According to a research, it is also possible to replace 32 bits for a binary (2 bits) weight data type with greater loss in the mAP with faster processing speed, but it should be properly evaluated to be applied into a rigorous industrial inspection scenario.

CNN-11.2. Kernel/Filter Size and Shape

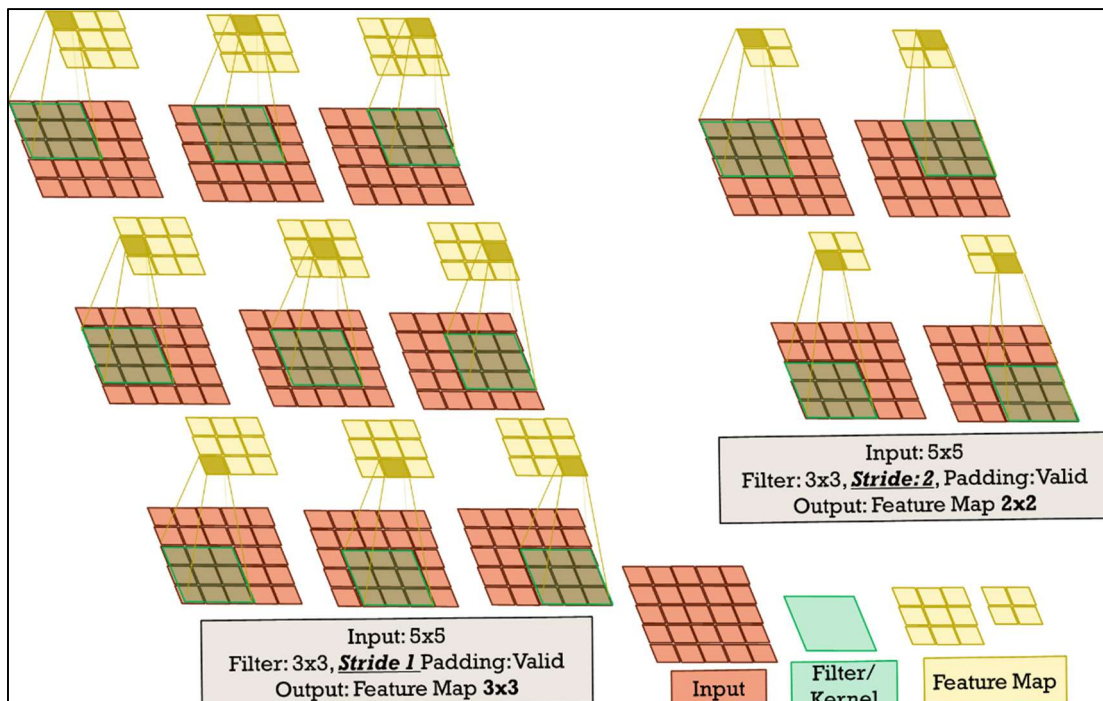
The size and shape of the filter determine how the model will process the image inputs. Common kernel sizes of standard square filters range from 1x1 to 11x11 but that does not mean larger sizes cannot be used. The computational power requirements may increase with larger filter sizes since it must store a bigger matrix with more weights within.

The standard shape for a filter is a square, but newer studies propose quasi-hexagonal shapes (Sun, Zhun; Ozay, Mete; Okatani, 2016), dilated squares (Wang, P. *et al.*, 2018), deformed (Dai *et al.*, 2017), and atrous convolution. Squared shaped considers the influence of each nearby feature value or image pixel given the center of the kernel. Dilated shapes do consider nearby features because they have blank spaces between filter weights. Quasi-hexagonal shapes consider nearby features but not all of them. Deformed convolution has an irregular shape which or may not consider nearby features. The newer shapes tend to decrease model complexity, and they can be used to optimize existing standard shape models.

CNN-11.3. Kernel Stride and Padding

Kernel stride and padding are parameters of the convolution operation. The stride value defines how the filter will be sweep over the input. The minimum value for a stride is one, but larger values can be used. Figure 38 shows an example of the effects of a fixed size filter over the same image, considering a different stride value and the difference in the feature map size caused by that change.

Figure 38 - Example of Valid Padding with different stride values.



Source: The Author

Higher strides values tend to compress the resulting feature map and can be used for image and feature map input that have a larger size. One drawback of a higher

stride value is that it can lose information on small or scaled objects during the compression process.

Padding parameter changes the input size by adding zero value around the image. Padding can be used for two purposes: to assure that the feature map output will have the same size as the input and to allow the filter to extract relevant information located near the feature map edges. When a padding value is equal to zero, it means the image input size will not be changed before the convolution operation. This is known as valid padding. When padding is higher than zero, it means that the input will be changed. The same padding happens when the padding value assures that the output will have the same size as the input. Figure 39 shows an example of a convolution operation with a stride value of 1 and padding same.

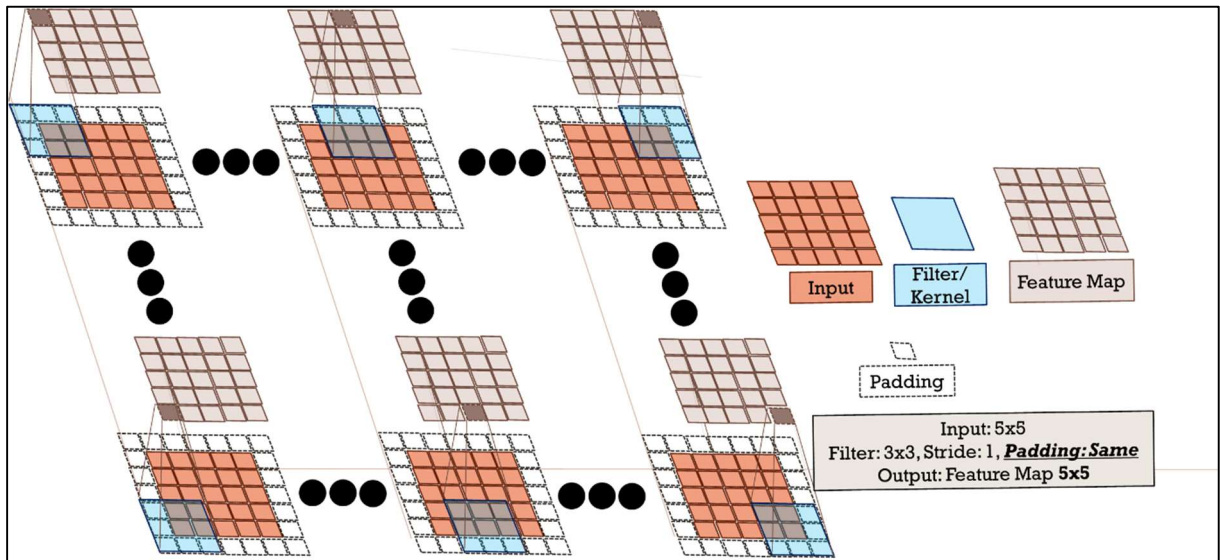


Figure 39 - Example of Same Padding with stride value equals two.

Source: The Author

The feature map output depends on the input size, stride value, padding value, and kernel size. Equation 22 shows how the feature map width can be calculated, and equation 23 shows how to calculate the padding value for the same padding option.

$$W_{\text{output}} = \frac{W_{\text{input}} + 2 * (\text{Padding Value}) - W_{\text{filter}}}{\text{Stride Value}} + 1 \quad (22)$$

$$\text{Padding Value} = \frac{(W_{\text{input}} - 1) * \text{Stride Value} + W_{\text{filter}} - W_{\text{input}}}{2} \quad (23)$$

CNN-11.4. Number of Filter/Kernels per layers

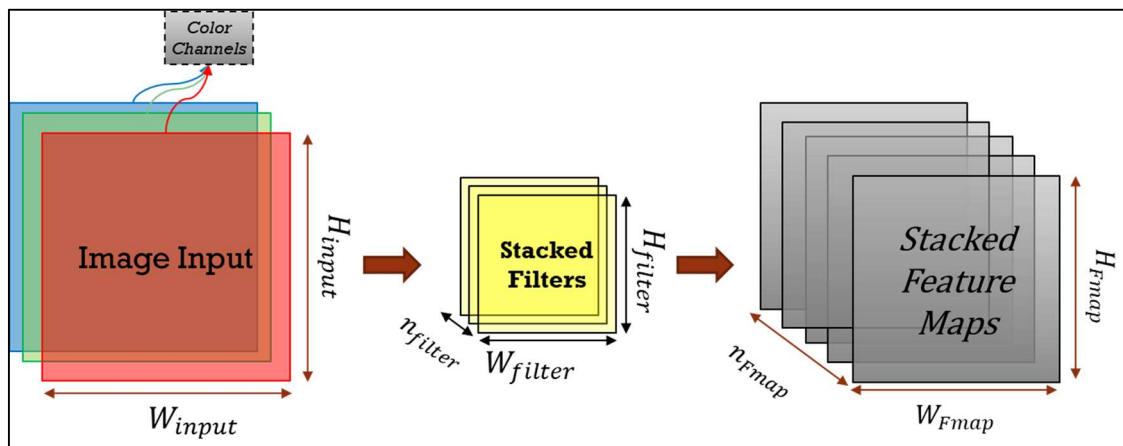
The convolution operation is composed of several filters applied over the same input. Each filter can learn a different characteristic of the image input. Equation 24

shows how to calculate the number of feature maps output based on the number of inputs and the number of filters.

$$n_{\text{fmap}} = n_{\text{filter}} * n_{\text{input}} \quad (24)$$

The more filters are applied, the higher the image features learned during the convolution operation. Increasing the number of filters increases the demands for computational power because the number of feature maps will increase, and they will need to be stored in the hardware memory. Figure 40 shows stacked filters applied over an image input with the standard RGB color channels and its resulting stacked feature maps.

Figure 40 - Example of stacked feature maps gives an image input and a pack of convolution filters.



Source: The Author

The deep learning concepts for image processing is related to how many convolution layers are applied over an image input. Recent researches introduced the wide concept, which increases the number of filters of a single convolution layer and allows a reduced number of convolution layers without losing mAP performance (Lee *et al.*, 2017), (Szegedy *et al.*, 2016)

CNN-11.5. Convolution Options

Besides the standard hyperparameters described above for the convolution operation, there are new options that improve its performance. The first option is the concatenating filter, which speeds up the setup process and selection of the adequate filter sizes. The process is known as the inception layer, which applies multiple filter sizes and concatenates into a single feature map output (Szegedy *et al.*, 2016)

Another option for the convolution operation is to prune filters during the learning process. During the learning process, some filters may have little effect over the output

generating unused feature maps output, which consumes hardware memory. The filter pruning technique detects those unused filters and remove them from, saving memory and speeding up the learning process. The pruning technique can also be used to remove filter which a very high influence during the learning process. The presence of these filters may severe the learning of other filters. They can be suppressed during a certain stage of the training, allowing other filters to compensate for its absence. The pruned filter can be reactivated in a later moment if necessary (He, 2017), (Luo *et al.* 2017).

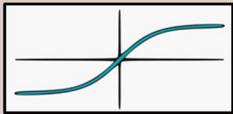
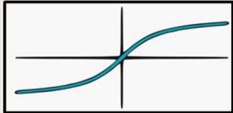
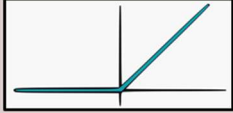
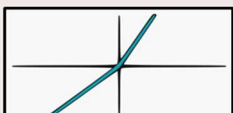
CNN-12. Activation Function and Normalization

Input normalization, also known as batch normalization (Ioffe; Szegedy, 2015), and activation functions are mathematical operations that can be performed with the feature maps. Both are important to assure that the learning process will converge.

CNN-12.1. Activation Function

The activation function effects over the learning process were already detailed previously in this document. Rectified Linear Unit (ReLU) Nair; E. Hinton (2010), Leaky-ReLU (L-ReLU), and Parametric-ReLU(P-ReLU) (He *et al.*, 2015) are being used instead of the sigmoid functions such as hyperbolic tangent and arctangent functions. Table 10 shows four common types of activation functions which are commonly applied in ANN's (Gupta; Duggal, 2018).

Table 10 - Example of commonly activation function for image-based solutions.

Activation Function	Graph	Function Details
TANH		$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
ArcTan		$f(x) = \tan^{-1}(x)$
Rectified linear unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Leaky Rectified linear unit (Leaky - ReLU)		$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$

Source: Adapted from Gupta; Duggal (2018)

Sigmoid functions are more sensitive to the vanishing gradient, thus, they increase the chance of the harming the learning process. ReLU transforms all the negative inputs into zero, while P-ReLU and L-ReLU multiply the negative value by a small variable. This small variable is a learning parameter while using P-ReLU, but L-ReLU consider it like a fixed hyperparameter (Zhang *et al*, 2017).

CNN-12.2. Normalization

The normalization process can be applied before the activation function, and through this method, the mean of the inputs of each layer is standardized, limiting its variance to one. Normalization contributes to the learning process by adding a well-behaved input, which enables a faster and effective optimization. It also turns the model robust to hyperparameter slight changes and helps minimize the vanishing gradient problem (Santurkar *et al.*, 2018).

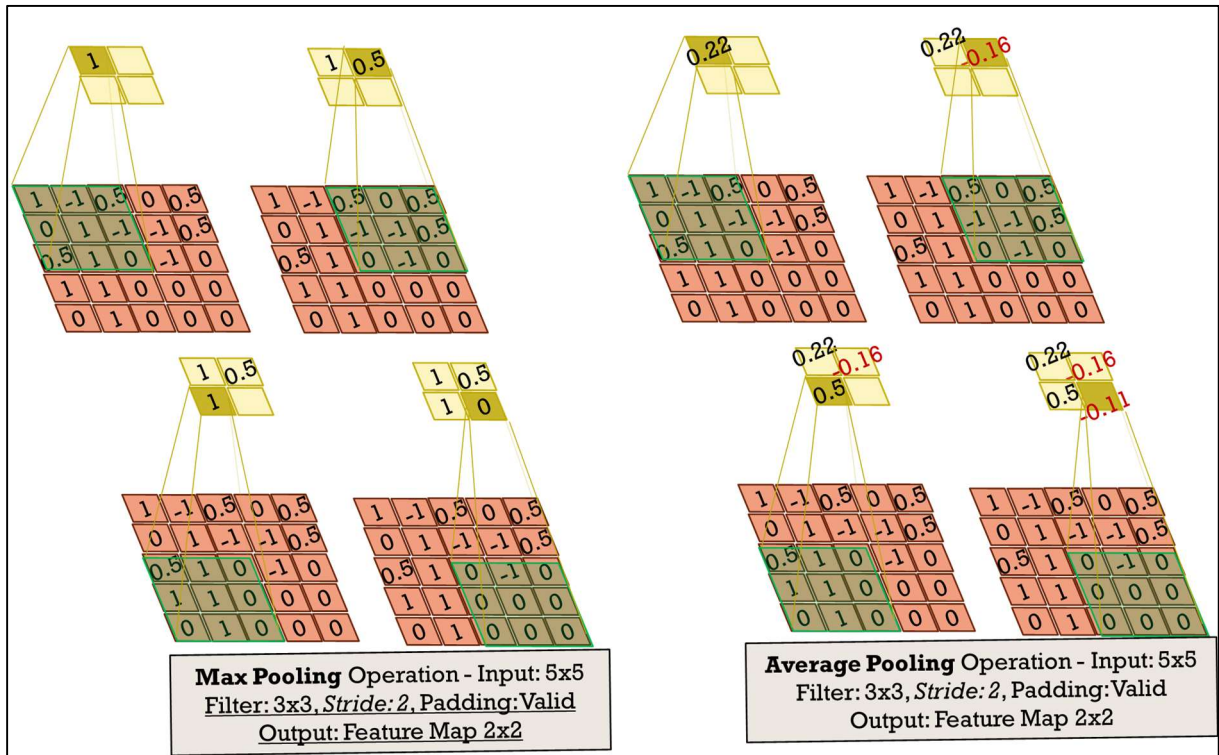
CNN-13. Pooling Operation

The last step of a convolution stage is pooling. Pooling is used in CNN to reduce the dimensionality of feature maps, making them less susceptible to data variation and perturbation, and still preserving the main image features. The two main pooling types: averaging, maximum. Pooling also represented by a square matrix of size $W_{pool} \times H_{pool}$ (Lee *et al.*, 2015). Pooling works like a convolution operation, which requires a stride and padding parameters.

CNN-13.1. Pooling Types

The first pooling types applied to CNN were the average and the max pooling. Figure 41 shows an example of the max and average pooling.

Figure 41 - Example of max and average pooling.



Source: The Author

Average pooling sums the values inside the and finds the average value based on the filter size. Max pooling extract the highest value of the analyzed filter. Feature output reduction due to pooling can decrease training time. One aspect that should be taken into account when choosing the pooling type is that max-pooling tends to underestimate the size of objects while average-pooling tends to overestimate it (Kolesnikov; Lampert, 2016).

CNN-13.2. Pooling Stride and Size

Pooling stride and size function similarly to the convolution equivalent parameters. Stride equal one is the minimum value, but it can be higher if necessary. The higher the stride value, the smaller the feature output it will be. Feature output reduction due to pooling can decrease training time. While excessive feature output reduction performed by a single pooling operation may cause relevant information to be discarded, and consequently decreasing detection accuracy (Yu, Q. *et al.*, 2017).

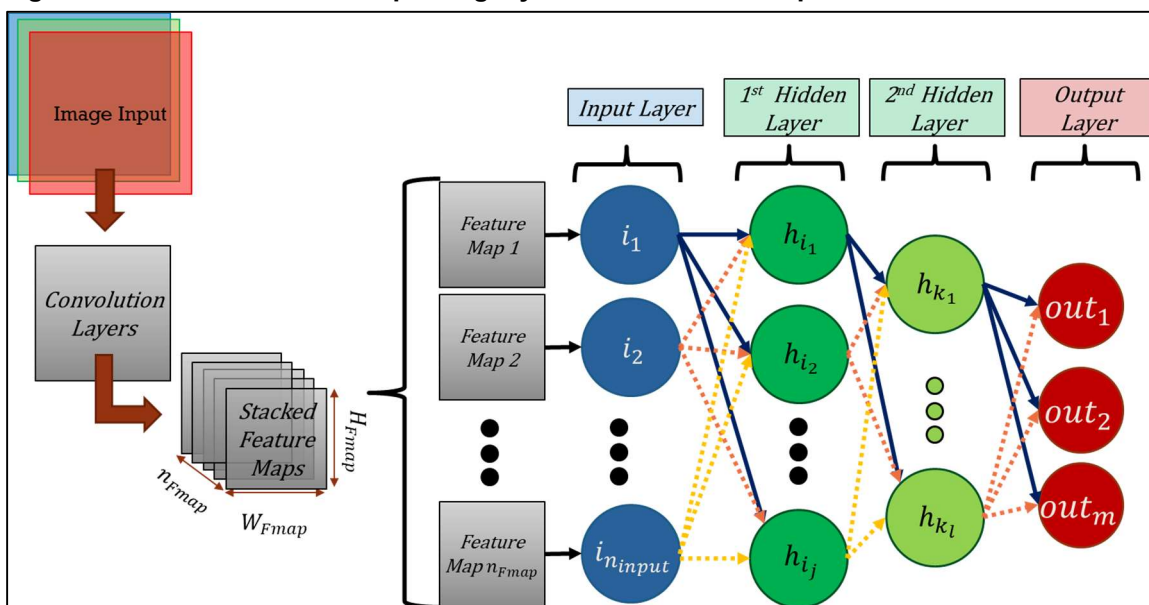
High pooling size values tend to increase the evaluation window of the feature map. It will consider more information for the max or average pooling. A high size value may not differentiate two important features located at the pooling extremities with the

same values. While a high size value using averaging filter may combine the effects of a high feature map value with a nearby low importance feature map.

CNN-14. Fully Connected Layer

The term 'fully connected' means that every neuron from the previous is connected to each neuron layer in the current layer. A fully connected (FC) layer in CNN is located at the end of the convolution layers which provide processed feature maps, as shown in Figure 42.

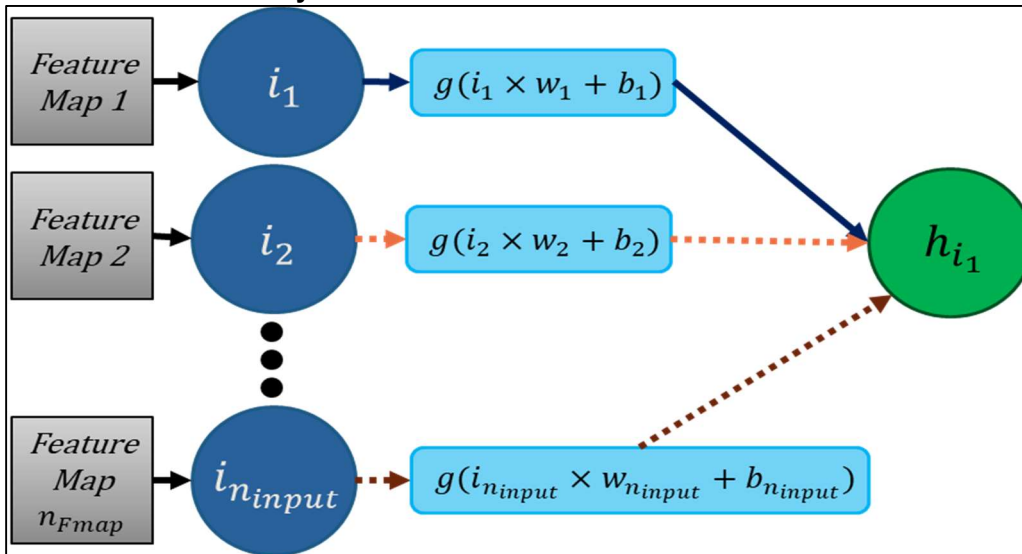
Figure 42 - Localization of the pooling layer and its relationship with the stacked feature maps.



Source: The Author

The feature maps will be considered as input neurons in the FC layer. The FC layer is composed by the input layer, the hidden layers, and an output layer. These layers work the same way as described previously in the AI simple learning. Figure 43 shows that the input of the FC is a feature map and all values will be fed into each neuron of the next layer, given its specific weight, bias and activation function given by 'g'.

Figure 43 - Feature maps being fed into the input layer and their relationship with a single neuron in the hidden layer.



Source: The Author

CNN-14.1. Number of layers in FC

The number of layers in an FC depends on how many hidden layers it will have. Increasing the number of hidden layers will increase the ability of CNN to evaluate the influence of combined feature maps, furthermore, improving the ability of learning complex data. Models are normally composed of three hidden layers Krizhevsky *et al.* (2012), but it is possible to use FC with one, two, or any value larger than three. A side effect of several hidden layers is the increased chance of overfitting, where the models memorize the training data and lose the ability to generalize the classification. The increased number of layers will increase the number of neurons and consequently, the number of weight and bias to be learned, which may slow down the training speed.

CNN-14.2. Number of neurons

The number of neurons can be defined by the user, and a different value for each layer in the FC can be set by the user. Usually, the number of neurons of the FC input layer has the same amount the number of feature maps. The output neuron will depend on the number of classes or objects to be detected. The number of neurons in the hidden layer is variable

Increasing the number of neurons in each hidden layer does not necessary means that the FC performance will increase proportionally. When the number of neurons achieves a certain threshold, some neurons may not be activated. Activated neurons mean that the feature is relevant for that kind of input. Neurons that are not activated can mean two things, either the neuron is not relevant to that specific input,

but it is relevant for another input, or the neuron is not relevant at all. These irrelevant neurons are considered as neurons with zero or near zero activation values.

CNN-14.3. Fully connected layers Options

Dropout, pruning are techniques for FC layers which can aid in excessive irrelevant neuron and overfitting problems. Each technique affects the neurons in the hidden layers.

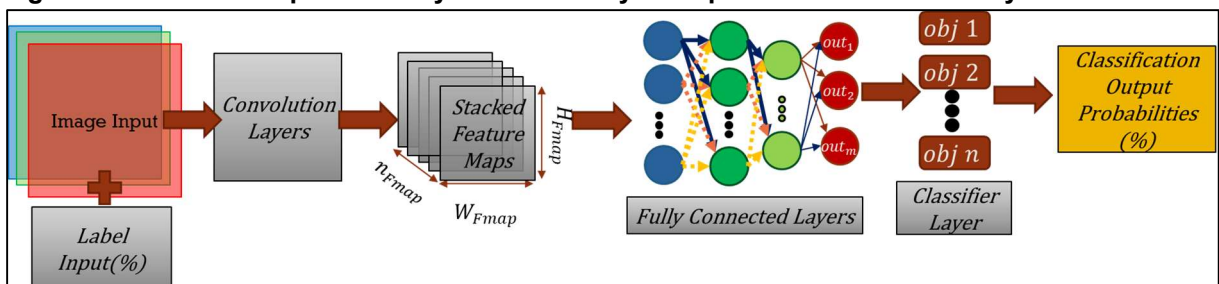
Dropout is a technique that creates a probability of shutting off a hidden neuron randomly. If a neuron is shut off, any incoming and outgoing weights and bias related to that neuron will also be dismissed (Srivastava *et al.*, 2014). The dropout rate is the hyperparameter, which defines the probability of a neuron to be active. The lower the rate, the higher the chance of neurons being turned off. Dropout may also reduce the number of neurons with low activation values and decrease the chance of overfitting. Excessive dropout rates could cause underfitting, where the model loses the ability to learn from data.

Pruning (Bondarenko *et al.*, 2015) has the same effect of the dropout, but instead of removing neurons randomly, it can select the neurons. The main challenge of the pruning is to implement an efficient algorithm which can carefully select which neurons are relevant or not.

CNN-15. Classifier.

The classifier is the last layer of a CNN and its fed by the output of the FC layer. Its purpose to provide a probability class or object output given the input data provided by the FC layer (Hu *et al.*, 2017). Figure 44 shows how the classifier is connected to the previous layer.

Figure 44 - Relationship of the fully connected layer output and the classifier layer.



Source: The Author

The output of the classification layer is a vector containing the probabilities of each possible class or object. The equation describes the classification output vector, where P_n is the probability value of the 'n' object.

$$\text{Classification Output} = [P_1, P_2, P_3, \dots, P_n]$$

This output vector is compared with the label input given with the image input. The label input should inform what object class was fed into the CNN. To match the input with the classification output, the label must be converted to one hot encoding. Figure 45 shows a label input is converted into one hot encoding, where the initials 'OBJ' are referred to the object class and its respective one hot encoding.

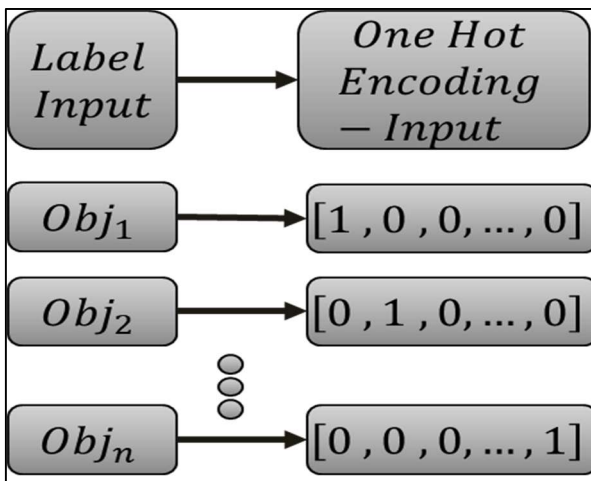
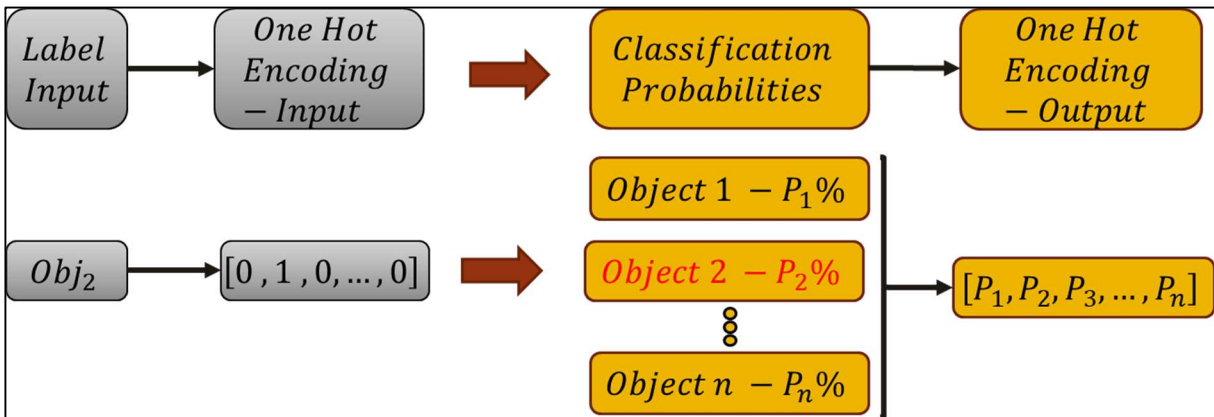


Figure 45 - Transformation of the label input into a one-hot encoding form.

Source: The Author

For example, OBJ₂ means that the image input belongs to the object class two and the all the values of the one hot encoding will be zero except the second position in the vector, which will be one. The position of the one value will determine which object class input will be. Figure 46 shows an example of an object label input converted into one hot encoding and the comparison with the classification output probabilities vector form.

Figure 46 - Relationship of the label input and the classification output and its one hot encoding vector.



Source: The Author

The one hot encoding input value is compared with the probabilities given by the output vector. For example, in Figure 46, has the input vector with the one value in the second position, and the output vector value P_2 should be close to one for a good classification score. In the other hand, the remaining probabilities values should be close to zero. The error value given by the comparison between the input and output vectors will be used as the input for the AI learning method.

CNN-15.1. Classifier Type

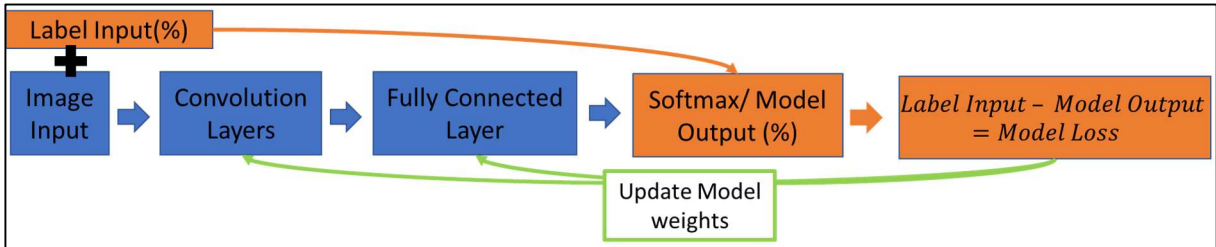
There are several classifier methods that can be applied to CNN. The main classifier methods are SoftMax (Duan *et al.*, 2007), Support Vector Machine (SVM) (Li *et al.*, 2016), Adaptive Boosting (AdaBoost) (Song *et al.*, 2015) (Wu; Nagahashi, 2015), Linear discriminant analysis (LDA) (Liu *et al.*, 2019). There are studies to determine which classifier has the most accurate results, but there no major difference between them (Tang, 2013) (Liu *et al.*, 2019).

CNN-16. AI Learning Method

The learning method is the last part of the solution which uses the mathematical equations early in chapter 2 of this document. The learning method will depend on the number of images in the dataset, how these datasets are divided into smaller batches, when the model will update the learned parameters (e.g., weights and bias) and which are the convergence criteria. The number of cycles of the learning process, also known as an epoch, is a way to count how many times all the images or batches are where fed into the CNN and its learned parameters are updated.

Stochastic, batch, and mini-batch learning are three main types of learning strategies that can be applied into CNN's. The difference between each one of them is when the weights and bias are updated, and what is the batch size input. Figure 47 shows the first type of strategy, known as stochastic learning.

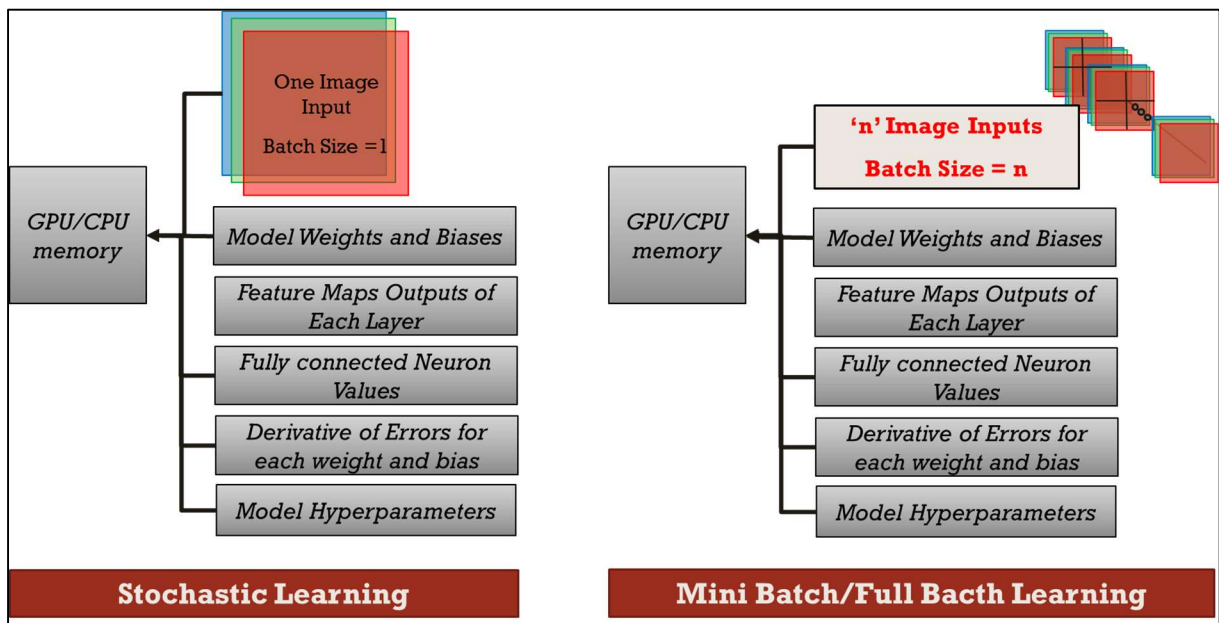
Figure 47 - Example of stochastic learning on CNN.



Source: The Author

Stochastic learning uses a batch size of one, and the learned parameters updates happen right after each input is processed by the CNN. One learning cycle in stochastic learning is when all the images from the trained dataset are processed by CNN. This type of learning strategy is the one who demands less computational power because only one image at the time must be load into the system memory. Figure 48 shows an example of what kind of data is allocated into the GPU/CPU memory and the differences between stochastic and batch learning.

Figure 48 - Differences between stochastic learning and batch learning regarding memory consumption.

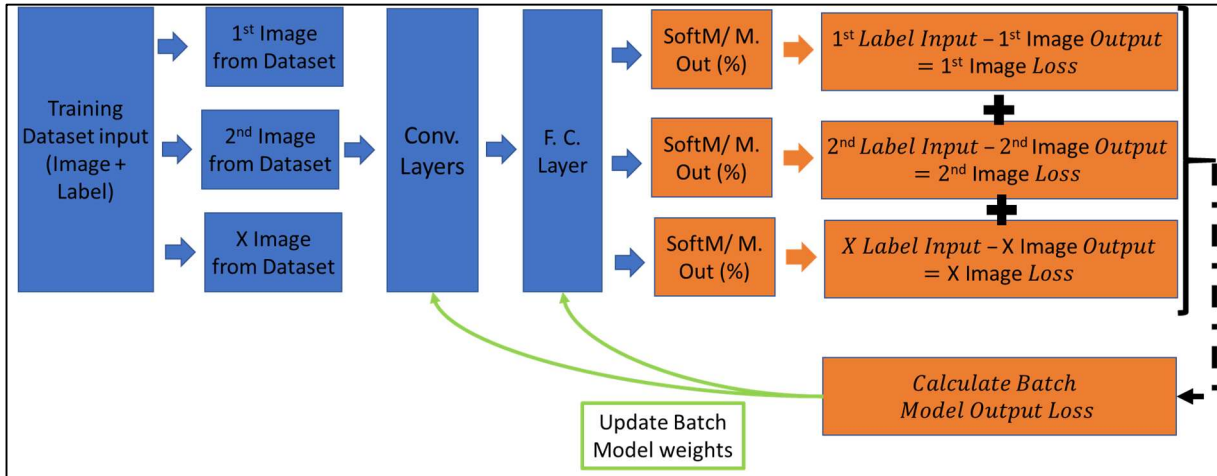


Source: The Author

Batch learning strategy is different from the stochastic type. In batch learning the model upload more images into the memory. The weights and bias updates are

calculated using the average values of the errors of each image input in the batch, and they are updated only after a whole batch is processed. Full batch learning happens when the batch size value is equal to the number of images given in the training dataset. Figure 49 shows the second type of strategy, known as batch learning.

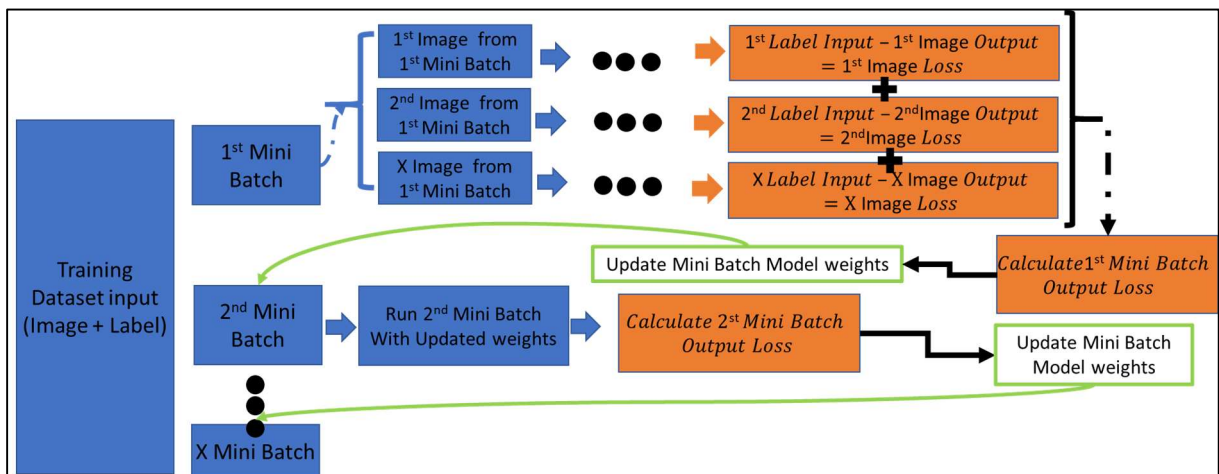
Figure 49 - Example of full batch learning in CNN.



Source: The Author

Mini-batch learning is the third possible strategy. The training image dataset is divided into several small batches and learned parameters updates are performed at the end of each mini batch. Epoch update happens when all the mini batches are processed. Figure 50 shows this type of strategy.

Figure 50 - Example of mini-batch learning on CNN.

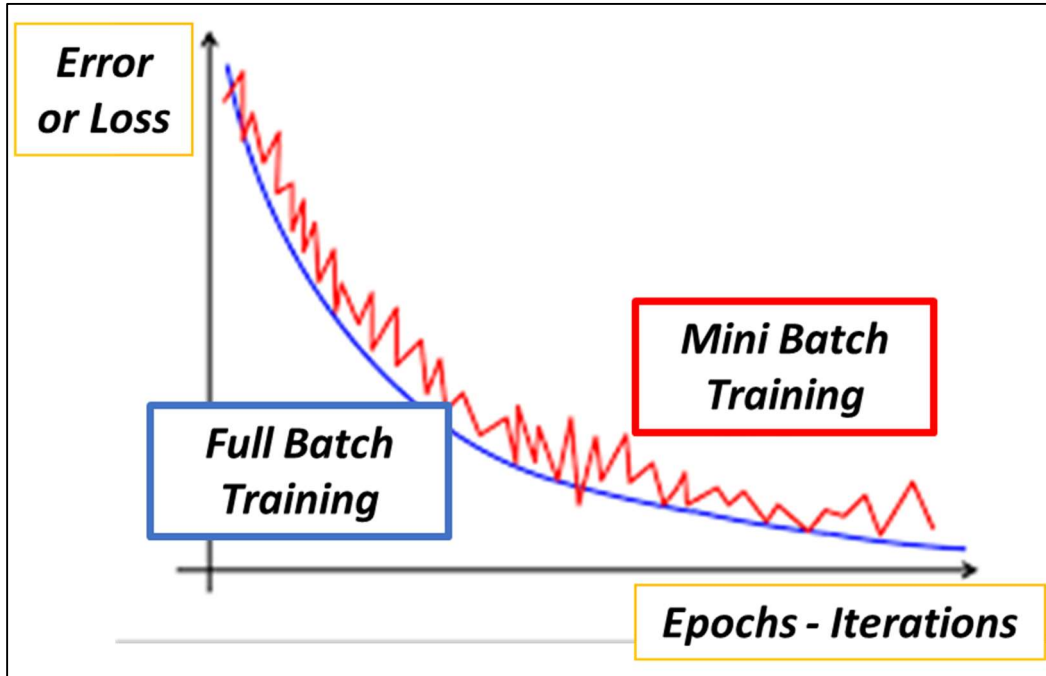


Source: The Author

Full batch training is considered the best training option to achieve a generalized solution, but using this strategy is not always possible. Memory limitation to upload the full batch is the main challenge for this strategy, and when there are memory shortage

problems, the mini batch is recommended. Figure 51 shows the noisy error behavior of mini-batch learning and the smoother behavior of the full batch learning.

Figure 51 - Error vs. Epoch curve for full batch and mini-batch behavior comparison.

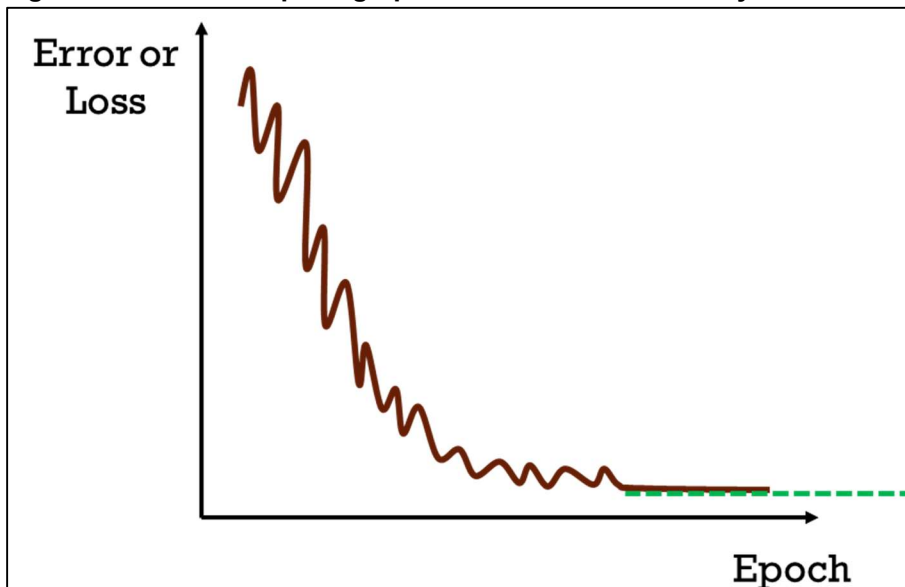


Source: The Author

CNN-16.1. Learning Method Epoch and Convergence

The number of epochs required for a model to achieve the minimum error and achieve convergence will vary case to case. Convergence is achieved when the difference between the updated error value is almost the same as the previous value. Figure 52 shows an example of model convergence.

Figure 52 - Error vs. Epoch graph with a model that already achieved convergence.



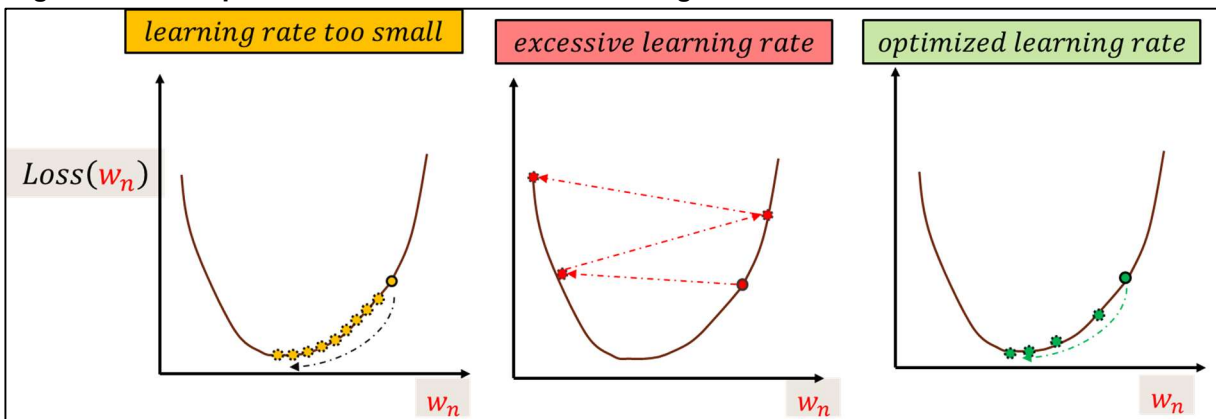
Source: The Author

To avoid unnecessary training, it is possible to use an algorithm which detects the difference between errors, and it achieves a certain threshold stopping the learning process. If the loss function is too noisy, this error difference may not be achieved. A second parameter may be set to limit the number of epochs regardless of the model achieving the desired minimum error.

CNN-16.2. Learning Method Characteristics

Model convergence depends on how the weights and bias are updated. These learned parameters depend on the learning rate parameter value. Setting this parameter is a challenging task due to its effects on weights and bias update. Figure 53 shows an example of how the loss function in the function of a weight value changes given different learning rates values.

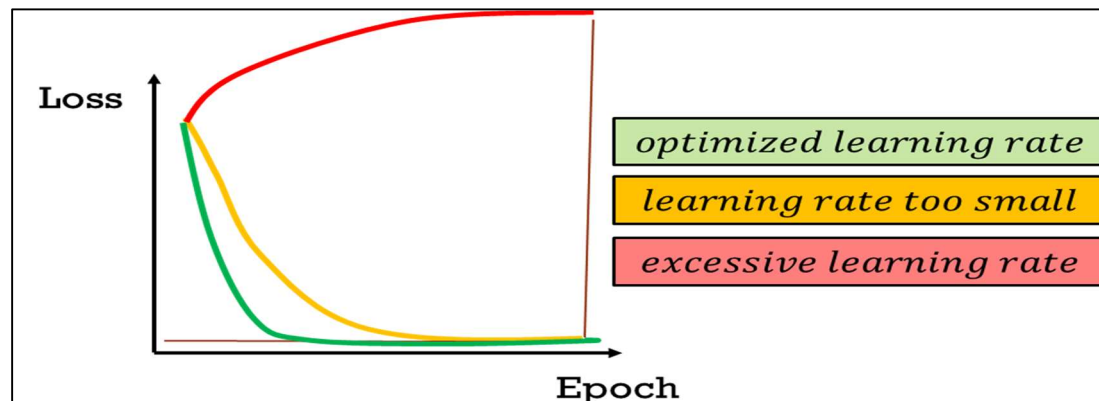
Figure 53 - Example of the effects of different learning rate values.



Source: The Author

Smaller learning rates makes models longer for training, while excessive values cause a non-convergence. Through an optimized learning, rate can make the model converge with fewer epochs. Figure 54 shows an example of these three scenarios.

Figure 54 - Example of different learning rate values in the loss/error vs. epoch curve.



Source: The Author

Equation 24 and 25 show how the weights and bias are updated using a fixed learning rate.

$$\mathbf{w}_{n_{\text{new}}} = \mathbf{w}_n - \alpha_{\text{fixed l.rate}} * \frac{\partial C}{\partial \mathbf{w}_n} \quad (24)$$

$$\mathbf{b}_{n_{\text{new}}} = \mathbf{b}_n - \alpha_{\text{fixed l.rate}} * \frac{\partial C}{\partial \mathbf{b}_n} \quad (25)$$

There are optimization methods which accelerate the learning process. The first of them add a momentum variable which accelerates the gradient descent by replacing the derivative of the cost function relative to a weighted derivative value. Equation 26 a 27 shows the weighted derivate value formula and 28 and 29 shows how they are used to calculating the new parameters.

$$\mathbf{V}_{\partial \mathbf{w}_{n_{\text{new}}}} = \beta_{\text{momentum}} * \mathbf{V}_{\partial \mathbf{w}_n} + (1 - \beta_{\text{momentum}}) * \frac{\partial C}{\partial \mathbf{w}_n} \quad (26)$$

$$\mathbf{V}_{\partial \mathbf{b}_{n_{\text{new}}}} = \beta_{\text{momentum}} * \mathbf{V}_{\partial \mathbf{b}_n} + (1 - \beta_{\text{momentum}}) * \frac{\partial C}{\partial \mathbf{b}_n} \quad (27)$$

$$\mathbf{w}_{n_{\text{new}}} = \mathbf{w}_n - \alpha_{\text{fixed l.rate}} * \mathbf{V}_{\partial \mathbf{w}_n} \quad (28)$$

$$\mathbf{b}_{n_{\text{new}}} = \mathbf{b}_n - \alpha_{\text{fixed l.rate}} * \mathbf{V}_{\partial \mathbf{b}_n} \quad (29)$$

An unoptimized value of momentum may cause an excessive weight update, and the convergence is not achieved properly requiring more epochs.

There are other optimization algorithms which consider a variable learning rate such as Adagrad (Duchi *et al.*, 2011), Adam (Kingma; Ba, 2014) and their implantation are already available over the existing programming libraries. Ruder (2016) provided an evaluation of learning functions optimizer's which aid the user to select the best optimization strategy for gradient descent.

4.1.4 AI MVS Management (AI-MNGT)

The purpose of last this group is to provide minimum management requirements to keep AI MVS systems working and updated. The group begins with the type of users (17. User Profiles) which will interact with the AI MVS. The second part of this group is related to the management of the framework of an AI MVS (18.AI Frameworks). The last part provides basic functionalities for the AI MVS models (19.AI Models). Figure 55 shows each activity of the CNN group with its description and details

Figure 55 - AI-MNGT group with its main activities and simplified descriptions.



Source: The Author

Since the concepts of framework and models may have similar interpretations, for this document, an AI Framework will be treated as the final AI algorithm. This final algorithm can be composed by a combination of methods and algorithms where each one of them provides complementary functionality. The AI model will be considered as the application of an AI framework for a given inspection purpose. Thus, an AI

framework serves as the backbone of a solution which can be deployed for different AI models.

AI-MNGT 17. User Profiles

User profiles define the basics type of user which will interact with the AI MVS in each phase of the solution lifecycle and which are their interactions the whole solution. The user profiles were divided into four groups based on their required skills, their main tasks, and what are their permission to interfere with the system. The main suggested users are the quality inspector, quality engineer, AI MVS expert, and other users. This task division is not mandatory because each company has its own hierarchy, job positions names, and responsibilities. Companies will have to choose if the proposed task division meet their corporate requirement. Otherwise they will have to adapt some of these new tasks.

AI-MNGT 17.1. Quality Inspector

The first user proposed by this method is the quality inspector. This user will directly interact with AI MVS providing it with new images for inspections or giving feedback of an inspection output given by the system. The role of the quality inspector is work as a supervisor of the AI MVS or to work together with it aiding in tasks where the AI MVS has a technical limitation.

The quality inspector can be one of the main sources of input data to train or retrain a system. Their feedback to the system, either for the training phase or the deployed phase is very important to assure if the system is performing detections correctly. Also, the quality inspector has skills which do not rely on vision. They can gather complementary data to perform a complete analysis of defects and a root cause analysis.

The AI MVS is always trained with a fixed number of objects classes and detection outputs, but newer defects which were not included in the first AI model can happen. A quality inspector can inform the other users of the AI MVS platform to add theses update the system.

Depending on the size of the company and the inspection complexity, the task assigned to the quality inspector could be incorporated into the quality engineer/supervisor.

AI-MNGT 17.2. Quality Engineer or Supervisor

The second user proposed by this method is the quality Engineer or supervisor. This user will also directly interact with AI MVS, and they have the main responsibility of managing the system and its respective inspection outcomes.

In addition to the quality inspector responsibilities, the quality supervisor will have the permission to create new image datasets and select a validated AI framework to run the training. After the training, the responsible must check if the trained AI model meets the validation metrics and required performance. If the model shows satisfactory results, it will be deployed for inspection activities. If the model does not meet the validation metrics, the supervisor can either select a different AI framework or request the AI MVS expert to perform an assessment. This assessment can be related to AI framework hyperparameters fine-tuning or AI model dataset analysis.

The supervisor will be responsible for the inspection of KPI's monitoring and control. These KPI's could be related to the AI MVS performance such as periodic classification and localization metrics check or related to the process performance, such as the number of detected defects, type of defects. In the case of process updates or production evolution, the responsible must evaluate if the AI model should be updated or not.

AI-MNGT 17.3. AI MVS Expert

The third user proposed by this method is AI MVS Expert. This user must have skills in image processing techniques and AI intelligence programming. Their role will be to create new AI frameworks, add new AI techniques and functionalities, optimize and fine-tune AI trained models, perform minor AI framework and updates, and evaluate the performance of existing AI models.

Due the constant evolution of AI techniques, the AI MVS Expert must be constantly evaluating the updates of these newer frameworks, which can increase detection speed and classification performance, or reduce the need of computational processing power.

Newer AI frameworks do not necessarily mean the addition of newer AI techniques. There are several techniques which already exist in the literature, but there are not fully integrated into an industrial solution. The AI MVS expert will be responsible for creating and testing the performance of these newer frameworks.

Another role of the specialist is to optimize and fine-tuning trained models when requested by the supervisor or the engineer. This task is normally related to AI models that do not meet the validation metrics. To fine tune models, one must understand how image processing and AI programming works. The specialist must determine what is cause non-convergence, where the source of the problem can be the insufficient images, unbalanced datasets, low-quality images, incorrect annotation, wrong initial hyperparameters, incorrect selection of AI framework and so on. Upon training, a quality engineer may learn how to identify some those problems, but to avoid problems of AI MVS malfunctioning, it is recommended that this task is performed by an expert.

The AI MVS expert has a very specific set of skills which can be uncommon for some industries to have this kind of staff. One option for this type of user is to look for third-party companies which have this kind off staff. It is also possible to train existing MVS expert with AI programming skills to allow them to operate these kinds of solutions.

AI-MNGT 17.4. Other Users

Inspection outcomes may be distributed to users who are neither from the quality area nor from the specific AI MVS expert group. This other user's can receive outcomes in the form of images, for technical reports, KPI dashboards, or any other type of information the AI MVS can provide. Their level of access to modify how the AI MVS should be restricted in order to avoid any malfunctioning from external sources.

Inspection outcomes data can also be sent for other machines, to raise alarms or even to perform minor modifications in the manufacturing process. How this integration will work will not be discussed in this document, but it an important factor for factories to achieve a higher degree of maturity regarding 4.0 industry concepts.

AI-MNGT 18. AI Frameworks

AI framework is the fully working algorithm with the standard image processing functionalities and AI techniques integrated into one solution. The CNN group discussed earlier in this document is an example of an AI framework. Several frameworks were shown in the content analysis and they can be used or adapted to the industrial inspection scenario. Managing AI framework its essential to keep AI MVS working properly and to allow system improvements.

AI-MNGT 18.1. Standard AI Framework

Standard frameworks will be the first AI solutions adapted to the industrial scenario. They could be simple classification framework, object detection framework, or semantic segmentation framework. Each one of them must have all the hyperparameters pre-set and easily customizable for different images dataset. After each one is validated, they will be uploaded to the AI MVS platform so users can access and use them to create new inspections.

Each framework must have a version control system, so the AI MVS expert can manage the most stable version and perform the necessary modifications without harming existing inspections. Since the AI framework is the core of multiple AI model inspections, one must be very careful when updating them.

AI-MNGT 18.2. Create a new framework or modify an existing one.

The AI MVS platform must consider a developer option to create new AI framework or perform slight changes in the existing standard frameworks. The creation of a new framework must consider the requirements gathered in the ASA and CNN group. Other requirements which this method did not cover that AI MVS expert may find necessary can also be included in the creation of a new framework.

An example of newer frameworks is to use multiple AI framework to create an ensemble decision tree, where the same input is subject to multiple AI solutions, and they vote for the best result. Another example of a new framework is the coarse and fine-grained R-CNN. This network works with a generic or less specific R-CNN object detection, also known as coarse. The output of this coarse detection will be used for a more detailed classification, also known as fine-grained.

AI-MNGT 19. AI Trained Models

AI trained model is the combination of an image inspection data combined with an AI framework whose training is already validated and is ready to be deployed. Creating and managing AI models is one of the core activities to ensure that an AI MVS will work properly. First, an AI model must be trained, then it should be validated in with an ongoing inspection activity. Validated models can be subject to updates given new defects or the inclusion of the object classes. These models should be properly stored, and its backup managed to avoid corrupting existing models. This backup also serves as data to allow transfer learning from existing models.

AI-MNGT 19.1. Train Model

The first step of AI model is to train them. The user must create a new image database or upload existing ones along with its respective labels and annotations based on the required inspection type. The adequate AI framework must also be selected, and then training can be performed. The trained model will be subject to a validation using part of the images which were not used during the training phase to perform an initial validation. If the model does not achieve the required metrics, it must be subject to dataset improvements and corrections or, AI framework fine-tuning or, reselection the appropriate AI framework.

AI-MNGT 19.2. Validate Model

A model which has been trained and a pre-validated will be subject to real case inspections. During the validation process, the metrics should be constantly checked and be subject to the quality inspector or engineer feedbacks. These feedbacks will assure the ability of the model be flexible in real case inspections. When the model achieves is metrics, it will be validated and uploaded to the AI MVS platform. This validated version will be stored so the users can always have a stable inspection version.

AI-MNGT 19.3. Update Model

Industries, factories, and companies are subject to constant process changes and new product development. In addition to that, existing processes and its respective manufacturing processes are subject to new defects, which previously was not considered during the training phase of a model. The ability to update the model is essential to assure that the AI MVS inspection will be flexible to adapt to these modifications. Users will be able to send this new input to the AI MVS platform, and the responsibility for its management will be able to update the dataset and retrain an existing model. This update must be subject to the same validation metrics and if the new model will be updated only if they achieved the required performance. If they do not achieve the performance, the AI MVS expert will have to analyze these new changes and optimize the AI framework for this new scenario.

AI-MNGT 19.4. Model Backup

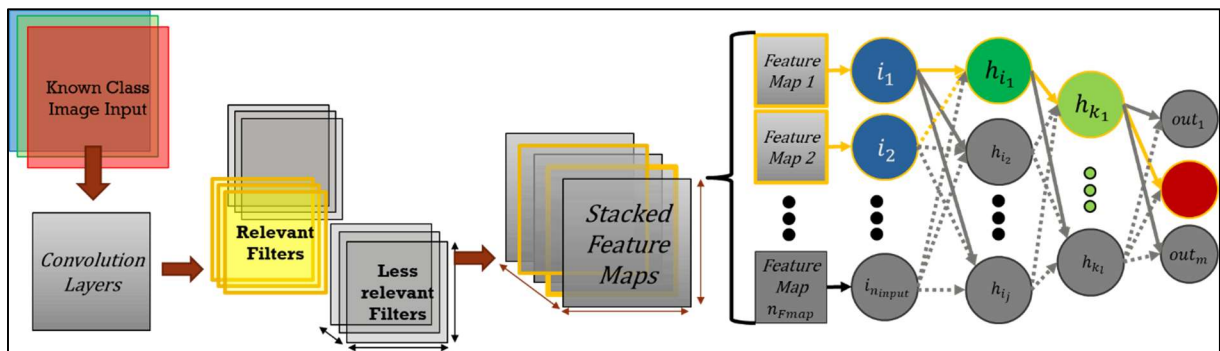
A backup of each new validated Model with be performed and stored in the AI MVS platform along with the respective images of the training and validation dataset.

These backups will assure that existing inspections will not be corrupted during their evolution process. The users responsible for the model creation, validation, and update must be registered in the AI MVS platform to keep track in who performed each activity. This register allows better management of what's have been done, monitor any problems in the process and allow the AI MVS correct evolution.

AI-MNGT 19.5. Model Transfer Learning

Training and optimizing new models with the current dataset, frameworks, and programming tools can be time-consuming. Model transfer learning is a technique proposed in this research that can transfer a specific knowledge about an existing class from a trained model and transfer them into a new or existing model. Figure 56 shows a diagram of how the transfer learning would be extracting data an existing model.

Figure 56 - Example of the extraction of relevant data in the proposed transfer learning technique.



Source: The Author

Based on a trained model, the user would select several images from a single known class and feed into the trained AI model. The output the model would fire all pack of neurons in the FC layers, but only a few of them would the most relevant values for that classification process. This allows the user to find which were the most relevant feature maps generated by the convolution layers. The same backtracking technique would be used to find the most relevant convolution filters.

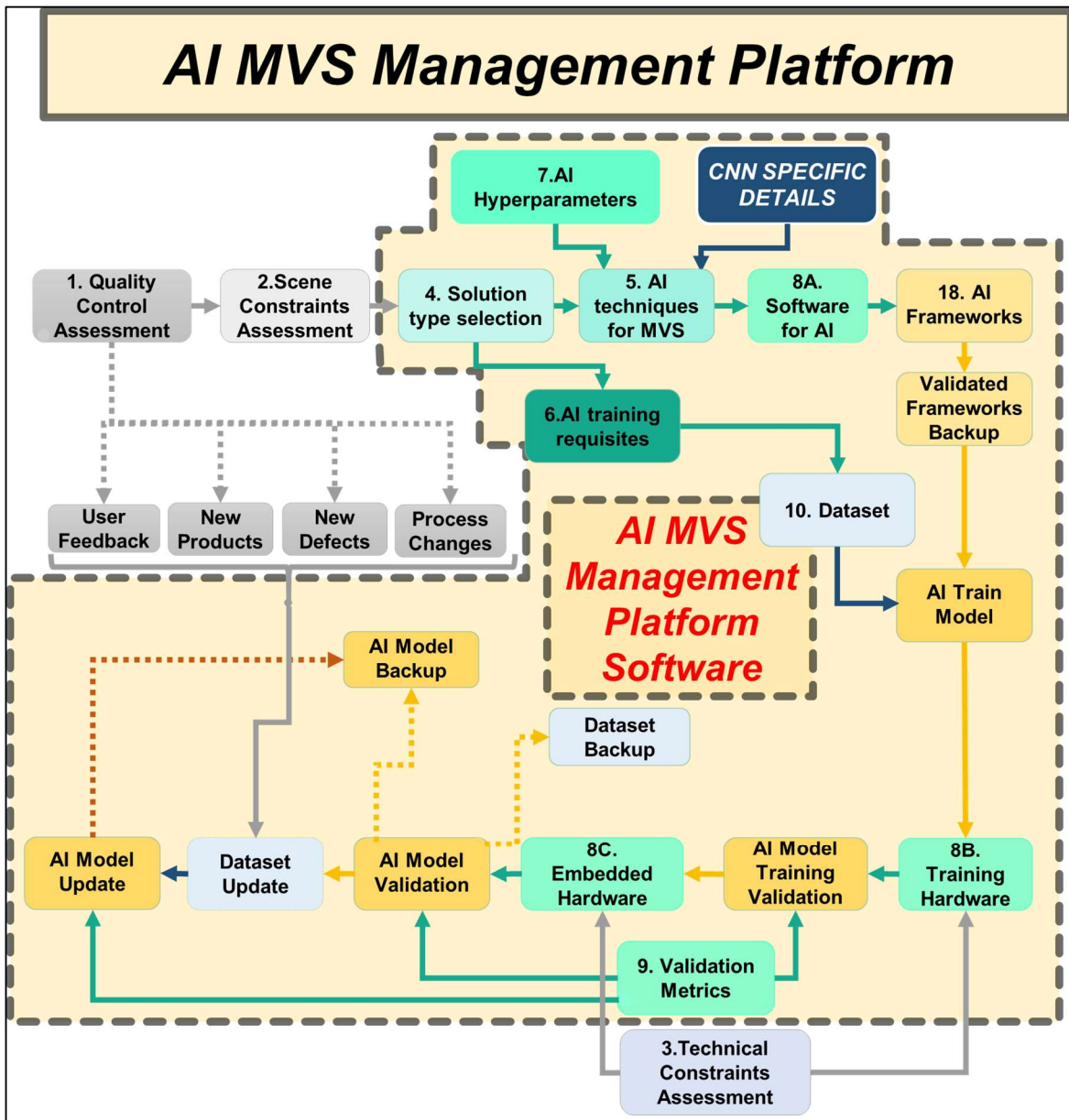
This proposed technique is different from fine-tuning, which feeds a new dataset into an existing AI trained model with a specific AI framework. The proposed technique must consider how this extracted data from a certain AI framework would be fed into a different AI framework with different convolution layers and FC configuration.

To validate if the extracted data is relevant, it is possible to pick the same dataset from the same desired class to train a single classification model and compared if the neurons and filters from both techniques are similar.

AI-MNGT 20. AI MVS Management Platform

The AI MVS management platform briefly mentioned in the previous section is a software that would integrate all the functionalities described in the proposed implementation model. Figure 57 shows how the AI MVS management platform works and its main functionalities.

Figure 57 - AI MVS Management Platform main workflow and its functionalities.



Source: The Author

Each element in the workflow has its correspondent description in the model so that the users can find how each of them works. Each new AI MVS inspection can begin in the Quality Control Assessment and follow the model in a clockwise flow. If the AI MVS is already implemented and functional, the user can start the process from

the AI Framework or the AI Train Model. This workflow can be adapted to each industrial scenario considering each specific requirement.

AI-MNGT 20.1. Management Software

All these workflow and functionalities can be converted into management software since most of the proposed functionalities are computer-based. The elements within the dashed line yellow box in Figure 57 are elements that would require to communicate with management software or to be directly accessed in the management software.

5 IMPLEMENTATION EXAMPLE OF THE PROPOSED FRAMEWORK MODEL

The implementation model validation was performed after each new functionality and activity was added. Four scenarios are presented to illustrate how the main elements of the model were validated and how the model evolved with each new scenario.

5.1 First Implementation Scenario – Non-AI-MVS Robotic Arm.

The first scenario is an example of non-AI MVS Robotic Arm project which was implemented in the of the assembly to perform image-based inspection. Since the model does not use any kind of AI technique, the implementation model can be used for the IIR group phase.

By using the IIR model the Quality Control Assessment was performed, and the warranty related cost expenditure KPI was used to define which product should be inspected. This KPI considers the cost that the factory had to indirectly pay to customers due products which showed some type of defect before its warranty expires. Components which presented the higher impacts on this KPI were selected for a deeper analysis. In this second analysis it was mapped the potential localization where the failures or defects origin source(s). Based on that information it was possible to determine the types of inspection that should be performed, the smallest feature size and where to place the MVS. The end of one assembly lines was selected as the best location, since the project consisted in a camera attached to robotic arm and it could detect most of the problems and it would have minor impacts on the existing assembly process. Table 11 resumes the outputs of the implementation model for this scenario.

Table 11- Summary of the implementation steps of the Robotic Inspection MVS

Sub Group		Activity	Scenario 01 - Robotic Inspection MVS
INDUSTRIAL INSPECTION REQUIREMENTS (IIR)	1. Quality Control Assessment	IIR-1.1 KPI or Indirect Evidences Gathering	Warranty related cost expenditure KPI Mapping Simplified root cause analysis to determine the potential localization of the failures or origin of defects
		IIR-1.2 Visual Inspection Applicability and Accessibility Check	Preliminary assesment of the potential localization of the failures or origin of defects to evaluate the accesibility check
	2. Scene Constraints Assessment	IIR-2.1 Inspection Type Definition	Several types of inspection: Diversity Check, Incorrect Part, Missing Part
		IIR-2.2 Verify the smallest feature size to be detected	10mm to 500mm depeding on the problem type
		IIR-2.3 Image Environment and Background Complexity	Industrial environment with low lighting region. No influence from sunlighth. Possible occlusion due worker interference
	3. Technical Constraints Assessment	IIR-3.1 Define MVS Location	In the end of the assembly line where most of the desired inspections could be performed simultaneously
		IIR-3.2 Fixed or Moving Camera	Camera attached to a Robotic Arm (Moving Camera)
		IIR-3.3 Machine Vision Components	Customized Camera with Controlled Lighting Kit
		IIR-3.4 Communication with existing Industrial Systems/Equipment's	HMI, Standardized PLC, Other Sensors

Source: The Author

5.2 Second Implementation Scenario – Non-AI – MVS.

The second implementation scenario is a non-AI MVS prototype that was used to quickly validate some concepts and activities of the ASA group. Most of the IIR group was imported from the previous implementation case, except for the following changes:

- Inspection type: Diversity check of assembled components.
- Smallest feature size: changed to 20 millimeters due component characteristics,
- Inspection camera: webcam on a fixed position.
- Equipment; Laptop and Raspberry PI

Most of the ASA activities could be applied on this scenario with the exception of the AI related data. The solution type chosen was the classification with object localization using a customized dataset. This dataset was created through images extracted from videos when the camera was positioned in the assembly line. No dataset improvement technique was applied in this case. The algorithm was programmed in Python 3.6 using the AWS Sagemaker online platform and the Jupyter Notebook the application to write the codes and validate the programming. Feature matching technique was selected because it can be programed to perform simple task of classification and object localization. The confusion matrix was applied to evaluate the algorithm classification performance.

Some difficulties were identified during the implementation process. One problem the absence of the solution workflow. The other one was to transfer the same programming environment created in the AWS Sagemaker to embedded hardware so it could run the designed solution. The last problem of this scenario was the low performance of the algorithm which was not optimized for the embedded hardware. The algorithm required almost four second for every image input image to evaluate correctly which was the component diversity, which is a very poor performance compared to recent techniques (Redmon; Farhadi, 2017),(Liu *et al.*, 2016). Those difficulties motivated upgrades in the implementation model which are already included in the version described early in this document. Table 12 resumes the main activities output of this scenario.

Table 12- Summary of the implementation steps of the Non-AI MVS Prototype

Sub Group		Activity	CASE 02 - Non-AI MVS Prototype
INDUSTRIAL INSPECTION REQUIREMENTS	2.Scene Constraints Assessment	IIR-2.1 Inspection Type Definition	One type of inspection: Diversity Check
		IIR-2.2 Verify the smallest feature size to be detected	20mm
		IIR-3.2 Fixed or Moving Camera	Fixed Camera
		IIR-3.3 Machine Vision Components	Simple Webcam only
		IIR-3.4 Communication with existing Industrial Systems/Equipment's	Laptop and Raspberry
ARTIFICIAL INTELLIGENCE ASSESSMENT (ASA)	4. Solution type selection	ASA-4.1 Application/ Functionalities	Classification with Object Localization
	5. AI technique for MVS	ASA-5.1 AI Techniques Selection	AI Technique was not applied
	6.AI training requisites	ASA-6.1 Supervision Type Selection	AI MVS was not applied - Training was not applied
		ASA-6.2 Dataset source	Customized Dataset (Custom)
		ASA-6.3 Dataset Improvement	No Dataset improvement was performed
	7.AI Hyperparameters	ASA-7.1 Solution Workflow/Diagram	Workflow or diagram was not used
		ASA-7.2 Layer, Method and Framework Parameters	AI MVS was not applied - MVS techniques: Automatized Feature Matching
	8. Hardware/ Software for AI	ASA-8.1 Training Hardware	AI MVS was not applied - Training was not applied
		ASA-8.2 Operational System	Windows
		ASA-8.3 Programming Language	Python 3.6
		ASA-8.4 Programming Environment and Libraries	AWS Sagemaker - Jupyter Notebook OpenCV, PIL
		ASA-8.5 Embedded Hardware	Raspebrry PI
	9. Validation Metrics	ASA-9.1 Classification Metrics	Confusion Matrix
		ASA-9.2 Localization Metrics	Localization Metrics was not applied
ASA-9.3 Training Hardware Performance		Training Hardware Performance was not applied	
ASA-9.4 Embedded Hardware Inference Performance		Frame processed per second - (4 seconds per input image)	

Source: The Author

5.3 Third Implementation Scenario – First AI MVS Prototype

The third implementation scenario considers the first AI MVS prototype to validate the remaining concepts and activities of the ASA group. Most of the IIR group was imported from the second implementation case, except for the equipment, which did consider the Raspberry PI due its lack of performance. This third scenario evaluates the algorithm performance only in the training hardware.

The solution type chosen was also the classification with object localization using the same dataset created previously. The SSD (Liu *et al.*, 2016) was the AI R-CNN technique selected to match the solution type functionalities. The programming platform was changed to the Anaconda Navigator so it would not depend on cloud computing and online services. The programming language was kept but new programming libraries has to be installed in order to use the GPU computing, such as cuDNN (Chetlur *et al.*, 2014), CUDA, and R-CNN framework libraries, such as Tensorflow (Abadi *et al.*, 2016) and Keras. The training hardware was the in-built GPU of the Laptop NVIDIA Geforce GTX 950M.

The images in the dataset were subject to cropping, changing the its size from 1280x720 to 300x300 which is the required input to the SSD framework. Bounding boxes a diversity classes labelling was also applied which each image of the dataset to allow supervised training. Seventy percent of the dataset was destined to the training phase, while the remaining was used for model validation. The remaining information of the CNN specifics details were not changed from the base SSD framework.

New difficulties were identified during the implementation process in this scenario. Since the available GPU of the laptop was not a early model, there was some challenges to find older programming libraries which would enable the programmed algorithm to work properly. Even with this problem solved, the SSD layers were not customized for this application which caused the AI model to crash during the training phase due lack of memory. After that, the SSD framework configuration and learning parameters, such as mini-batch size was modified. Those difficulties allowed more upgrades in the implementation model which are already included in the version described early in this document. Table 13 resumes the main activities output of this scenario.

Table 13 - Summary of the implementation steps of the First AI MVS Prototype

Sub Group		Activity	CASE 03 - First AI MVS Prototype
ARTIFICIAL INTELLIGENCE ASSESSMENT (ASA)	4. Solution type selection	ASA-4.1 Application/ Functionalities	Classification with Object Localization
	5. AI technique for MVS	ASA-5.1 AI Techniques Selection	Single Shot Multi Detector SDD- R-CNN based
	6.AI training requisites	ASA-6.1 Supervision Type Selection	Supervised Training
	7.AI Hyperparameters	ASA-7.1 Solution Workflow/Diagram	Used the SSD workflow
		ASA-7.2 Layer, Method and Framework Parameters	Layer and Framework Parameters according to SSD paper
	8. Hardware/ Software for AI	ASA-8.1 Training Hardware	GPU - NVIDIA Geforce GTX 950M
ASA-8.4 Programming Environment and Libraries		Anaconda Navigator - Jupyter Notebook OpenCV, CUDA, cuDNN, Tensorflow, Keras	
CNN SPECIFIC DETAILS	10. Dataset	CNN-10.1 Dataset Creation/Selection	Images with 1280x720 resolution extracted from static videos form the assembly line divided into 5 different classes
		CNN-10.2 Dataset Pre-processing	Dataset Image cropping from the original size to 300x300 to adjust data into the framework parameters
		CNN-10.3 Dataset Annotation	Manually applied Bounding Box and class type annotation
		CNN-10.4 Dataset Division	70% Dataset form training and 30% for validation
	11. Convolution Operation	CNN-11.1 Weights	Ramdomized Weights startup
		CNN-11.2 to CNN-11.5	Kernel and Filters were not changed from the SSD framework extracted from the paper
	13. Pooling Operation	CNN-13.1 Pooling Types	Pooling were not changed from the SSD framework extracted from the paper
		CNN-13.2 Pooling Stride and Size	
	14. Fully Connected Layer	CNN-14.1 Number of fully connected layers	Modified only the last FC layer to match the number of classes for the diversity check
		CNN-14.2 Number of neurons	
		CNN-14.3 Fully connected layers Options	Other fully connected parameters were not changed from the SSD framework extracted from the paper
	15. Classifier	CNN-15.1 Classifier Type	Modified only the classifier to match the number of classes for the diversity check
	16. AI Learning Method	CNN-16.1 Learning Method Epoch and Convergence	Learning Method were not changed from the SSD framework extracted from the paper
		CNN-16.2 Learning Method Characteristics	

Source: The Author

5.4 Fourth Implementation Scenario – Final AI MVS Prototype.

The fourth implementation scenario considers the final AI MVS prototype to validate the remaining concepts which were not considered in the previous scenario. Most of the IIR group, ASA group and CNN group configurations were imported from the second implementation case. For this final prototype the embedded hardware Raspberry PI were added to evaluate its performance with an optimized framework. The laptop used in this scenario has a modern GPU, NVIDIA Geforce GTX 1060, where there is no concern regarding to the version of the programming libraries, allowing a smoother evaluation the algorithm performance with the training hardware. This also allowed a faster optimization to deploy the trained model into the embedded hardware.

Tiny YOLO was used as the AI R-CNN technique, which is based on YOLOv3 (Redmon; Farhadi, 2018). Tiny YOLO requires an input of size 32x32, which allows a lighter AI model and it can run on embedded hardware. To allow a faster training, the starting weights were downloaded the YOLO website Redmon (2019) and they were pretrained based on the COCO dataset (Lin *et al.*, 2014).

Given all the customizations performed in the Tiny YOLO framework the model was trained and validated in less than 10 hours. Then it was upload into the embedded hardware. The embedded hardware managed to run the model processing three images per second, which is satisfactory performance given the available benchmarks (Yin *et al.*, 2018). Table 14 resumes the main activities output of this scenario.

Table 14 - Summary of the implementation steps of the Final AI MVS Prototype

Sub Group		Activity	CASE 04 - Final AI MVS Prototype
ARTIFICIAL INTELLIGENCE ASSESSMENT (ASA)	5. AI technique for MVS	ASA-5.1 AI Techniques Selection	Tiny Yolo - R-CNN based
	7.AI Hyperparameters	ASA-7.1 Solution Workflow/Diagram	Used the Tiny Yolo workflow
		ASA-7.2 Layer, Method and Framework Parameters	Layer and Framework Parameters according to Tiny Yolo
	8. Hardware/ Software for AI	ASA-8.1 Training Hardware	GPU - NVIDIA Geforce GTX 1080
		ASA-8.2 Operational System	Windows
		ASA-8.3 Programming Language	Python 3.6
		ASA-8.4 Programming Environment and Libraries	Anaconda Navigator - Jupyter Notebook OpenCV, PIL, CUDA, cuDNN
		ASA-8.5 Embedded Hardware	Raspebrry PI
	9. Validation Metrics	ASA-9.3 Training Hardware Performance	Time required for training
ASA-9.4 Embedded Hardware Inference Performance		Frame processed per second	
CNN SPECIFIC DETAILS	10. Dataset	CNN-10.2 Dataset Pre-processing	Dataset Image cropping from the original size to 32x32 to adjust data into the framework parameters
	11. Convolution Operation	CNN-11.1 Weights	Pretrained Weights imported from COCO Dataset
		CNN-11.2 Kernel/Filter Size and Shape	Kernel and Filters were customized to adapt the Tiny Yolo framework
		CNN-11.3 Kernel Stride and Padding	
		CNN-11.4 Number of Filter/Kernels per layers	
		CNN-11.5 Convolution Options	
	13. Pooling Operation	CNN-13.1 Pooling Types	Pooling was customized to adapt the Tiny Yolo framework
		CNN-13.2 Pooling Stride and Size	
	14. Fully Connected Layer	CNN-14.1 Number of fully connected layers	Modified only the last FC layer to match the number of classes for the diversity check
		CNN-14.2 Number of neurons	
CNN-14.3 Fully connected layers Options		Other fully connected parameters were customized to adapt the Tiny Yolo framework	
16. AI Learning Method	CNN-16.1 Learning Method Epoch and Convergence	Used mini-batch learning with six images per batch and ten thousand of epoch to achieve model convergence.	
	CNN-16.2 Learning Method Characteristics		

Source: The Author

6 CONCLUSION

The research provides an evaluation of current status of MVS for industrial inspection, identifying potential AI technique for MVS, and detailing how can they be implemented through the proposed model.

The most relevant limitations of traditional MVS identified by this research are the constant need of reprogramming the MVS for existing or newer inspection and, adapting it to newer inspections. They are more sensitive to environmental lighting changes, requiring more investments on controlled lighting background or special equipment to provide specific illumination conditions. Traditional MVS also lacks the ability to learn from previous inspections, which decreases its ability to adapt and the data from previous inspections are lost, if not stored properly.

The systematic review and content analysis allowed a deeper evaluation of the current scenario for AI MVS, identifying which techniques are being applied already, such as R-CNN. In addition to that, the review showed promising techniques like transfer learning, ensemble neural networks and automatic hyperparameters optimization architecture (PNAS) (Liu et al., 2017), which can used to further improve the existing AI MVS inspections functionalities. Despite the existing improvements of AI MVS, it was identified that the existing techniques and algorithms need to consider industrial requirements. Without these requirements and extensive validation in the industrial inspection scenario, the gap from what is being developed in the academical researches and industrial needs will not be fulfilled. Embedded hardware with the ability to run complex AI model in real-time inspection is one of the main challenge new AI MVS research must overcome. Specific validation metrics that fulfil not only algorithm performance, but industrial requirements must be included in these studies, to make these techniques more robust. Through the review it was possible to identify the absence of how industries can implement AI MVS, considering the most relevant requisites and what is the required knowledge to create technical specifications or allow negotiations with specialized companies that can provide this service.

The proposed model was built on the extracted elements from the systematic review and based on the experiments performed in an industrial environment which the author participated. The model division into steps helps non-AI specialist or non MVS specialists, to understand which are the main requisites to implement this solution

in an industrial scenario. It also provides a technical background and basic knowledges of the current AI techniques applied to machine vision. The approach was designed to use only the necessary mathematical operations without deepening into more complex operations and programming that each technique is based on.

The validation of the model was done parallelly as the model was built, identifying the limitations of each group and activities and improving it as new requirements or limitations were found. Since all the validation steps of the model was performed under the supervision of the author, some technical limitations, new requirements or newer groups may be required. These improvements would turn the proposed model more flexible to specific industrial scenarios and more robust in the validation process. The proposed management system and software were based on these difficulties and since their programming and implementation were complex, they could be further detailed during the two years of research. However, if companies have the intention to integrate AI MVS in their inspection routines, they will be soon needing some sort of management.

It is important to know that current AI MVS has the potential to perform many descriptive evaluations and in some cases some predictions based on the data given, but diagnostic and prescriptive functionalities, based on 4.0 industry concepts, are still to developed. Another important point to be developed and integrated into this study is the addition of 3D machine vision with AI features which were not covered. Besides that, this research and the proposed model can serve as a foundation for further improvements of AI MVS in academical research and allowing them to evolve into the cognitive machine vision inspection.

REFERENCES

- ABADI, M.; BARHAM, P.; CHEN, J.; et al. TensorFlow : A System for Large-Scale Machine Learning. **Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation**. p.265–283, 2016.
- AGRAWAL, H.; MATHIALAGAN, C. S.; GOYAL, Y.; et al. CloudCV: Large-Scale Distributed Computer Vision as a Cloud Service. **Mobile Cloud Visual Media Computing**, p. 265–290, 2015. Cham: Springer International Publishing. Available on: <http://link.springer.com/10.1007/978-3-319-24702-1_11>. .
- VAN BON, J.; DE JONG, A.; KOLTHOF, A.; et al. Foundations of IT Service Management based on ITIL V3. **ITIL Library**, v. 1, p. 234, 2007.
- BONDARENKO, A.; BORISOV, A.; ALEKSEEVA, L. Neurons vs Weights Pruning in Artificial Neural Networks. **Environment. Technology. Resources. Proceedings of the International Scientific and Practical Conference**, v. 3, p. 22, 2015.
- CHAUHAN, V.; SURGENOR, B. A Comparative Study of Machine Vision Based Methods for Fault Detection in an Automated Assembly Machine. **43rd North American Manufacturing Research Conference, NAMRC 43, 8-12 June 2015, UNC Charlotte, North Carolina, United States**, v. 1, p. 416–428, 2015. Available on: <<http://www.sciencedirect.com.ezproxy.javeriana.edu.co:2048/science/article/pii/S2351978915010513>>. .
- CHEN, Y.; LI, Y.; WANG, G.; XU, Q. A Multi-strategy Region Proposal Network. **Expert Systems With Applications**, v. 113, p. 1–17, 2018. Elsevier Ltd. Available on: <<https://doi.org/10.1016/j.eswa.2018.06.043>>. .
- CHETLUR, S.; WOOLLEY, C.; VANDERMERSCH, P.; et al. cuDNN : Efficient Primitives for Deep Learning. **arXiv preprint arXiv:1410.0759**, p. 1–9, 2014.
- DAI, J.; QI, H.; XIONG, Y.; et al. Deformable Convolutional Networks. **2017 IEEE International Conference on Computer Vision (ICCV)**. p.764–773, 2017. IEEE. Available on: <<https://github.com/msracver/Deformable-ConvNets>>. .
- DAVIS, J.; GOADRICH, M. The Relationship Between Precision-Recall and ROC Curves. **Proceedings of the 23rd international conference on Machine learning**. p.233–240, 2006.
- DRUZHKOVA, P. N.; KUSTIKOVA, V. D. A survey of deep learning methods and software tools for image classification and object detection. **Pattern Recognition and Image Analysis**, v. 26, n. 1, p. 9–15, 2016.
- DUAN, K.; KEERTHI, S. S.; CHU, W.; SHEVADE, S. K.; POO, A. N. Multi-category Classification by Soft-Max Combination of Binary Classifiers. , p. 125–134, 2007.
- DUCHI, J.; HAZAN, E.; SINGER, Y. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. v. 12, p. 1–40, 2011.
- FAWCETT, T. An introduction to ROC analysis. , v. 27, p. 861–874, 2006.
- FOIDL, H.; FELDERER, M. Research Challenges of Industry 4.0 for Quality Management. **Lecture Notes in Business Information Processing**. v. 245, p.121–137, 2016.
- GIRSHICK, R.; DONAHUE, J.; DARRELL, T.; MALIK, J. Rich feature hierarchies for

- accurate object detection and semantic segmentation. **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, p. 580–587, 2014.
- GOLNABI, H.; ASADPOUR, A. Design and application of industrial machine vision systems. **Robotics and Computer-Integrated Manufacturing**, v. 23, n. 6, p. 630–637, 2007. Institute of Water and Energy, Sharif University of Technology, Tehran, Iran.
- GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing**. 3rd ed. Pearson Education, Inc, 2008.
- GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. MIT press Cambridge, 2016.
- GUAN, J.; YI, S.; ZENG, X.; CHAM, W. K.; WANG, X. Visual Importance and Distortion Guided Deep Image Quality Assessment Framework. **IEEE Transactions on Multimedia**, v. 19, n. 11, p. 2505–2520, 2017.
- GUO, Z.; YE, S.; WANG, Y.; LIN, C. Resistance Welding Spot Defect Detection with Convolutional Neural Networks. **Computer Vision Systems. ICVS 2017**, v. 10528, p. 169–174, 2015. Available on: <<http://link.springer.com/10.1007/978-3-662-45129-8>>. .
- GUPTA, A.; DUGGAL, R. P-TELU: Parametric Tan Hyperbolic Linear Unit Activation for Deep Neural Networks. **Proceedings - 2017 IEEE International Conference on Computer Vision Workshops, ICCVW 2017**, v. 2018-Janua, p. 974–978, 2018.
- HE, K.; ZHANG, X.; REN, S.; SUN, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. **Proceedings of the IEEE International Conference on Computer Vision**, v. 2015 Inter, p. 1026–1034, 2015.
- HE, Y. Channel Pruning for Accelerating Very Deep Neural Networks. **2017 International Conference on Computer Vision.**, 2017.
- HITOMI, K. **Manufacturing systems engineering: A unified approach to manufacturing technology, production management and industrial economics**. 2017.
- HU, B.; LAI, J. H.; GUO, C. C. Location-aware fine-grained vehicle type recognition using multi-task deep networks. **Neurocomputing**, v. 243, p. 60–68, 2017. Elsevier B.V. Available on: <<http://dx.doi.org/10.1016/j.neucom.2017.02.085>>. .
- IOFFE, S.; SZEGEDY, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. , 2015. Available on: <<http://arxiv.org/abs/1502.03167>>. .
- VAN DER JEUGHT, S.; DIRCKX, J. J. J. Real-time structured light profilometry: A review. **Optics and Lasers in Engineering**, v. 87, n. 2016, p. 18–31, 2015. Elsevier. Available on: <<http://dx.doi.org/10.1016/j.optlaseng.2016.01.011>>. .
- JIA, Y.; SHELHAMER, E.; DONAHUE, J.; et al. Caffe : Convolutional Architecture for Fast Feature Embedding Categories and Subject Descriptors. **Proceedings of the 22nd ACM international conference on Multimedia**. p.675–678, 2014.
- KINGMA, D. P.; BA, J. Adam: A Method for Stochastic Optimization. , p. 1–15, 2014. Available on: <<http://arxiv.org/abs/1412.6980>>. .
- KOLESNIKOV, A.; LAMPERT, C. H. Seed, expand and constrain: Three principles for

weakly-supervised image segmentation. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 9908 LNCS, p. 695–711, 2016.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. ImageNet Classification with Deep Convolutional Neural Networks. **Advances in neural information processing systems**, p. 1097–1105, 2012.

LABUDZKI, R.; LEGUTKO, S.; RAOS, P. The essence and application fo machine vision. **Tehnicki Vjesnik (Technical Journal of Hardware and Vision Applications)**, v. 21, n. 4, p. 903–909, 2014.

LEE, C.-Y.; GALLAGHER, P. W.; TU, Z. Generalizing Pooling Functions in Convolutional Neural Networks: Mixed, Gated, and Tree. , v. 51, 2015. Availabe on: <<http://arxiv.org/abs/1509.08985>>. .

LEE, Y.; KIM, HUIEUN; PARK, E.; CUI, X.; KIM, HAKIL. Wide-residual-inception networks for real-Time object detection. **IEEE Intelligent Vehicles Symposium, Proceedings**, p. 758–764, 2017.

LEITÃO, P.; COLOMBO, A. W.; KARNOUSKOS, S. Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges. **Computers in Industry**, v. 81, p. 11–25, 2016.

LERONES, P. M.; FERNÁNDEZ, J. L.; GARCÍA-BERMEJO, J. G.; ZALAMA, E. Total quality control for automotive raw foundry brake disks. **International Journal of Advanced Manufacturing Technology**, v. 27, n. 3–4, p. 359–371, 2005.

LI, G.; LIU, J.; JIANG, C.; et al. Relief R-CNN: Utilizing convolutional features for fast object detection. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 10261 LNCS, p. 386–394, 2017.

LI, J.; QIAN, J.; ZHENG, Y. Ensemble R-FCN for Object Detection. **Advances in Computer Science and Ubiquitous Computing**, p. 400–406, 2018.

LI, M.; MA, L.; BLASCHKE, T.; CHENG, L.; TIEDE, D. A systematic comparison of different object-based classification techniques using high spatial resolution imagery in agricultural environments. **International Journal of Applied Earth Observation and Geoinformation**, v. 49, p. 87–98, 2016. Elsevier B.V. Availabe on: <<http://dx.doi.org/10.1016/j.jag.2016.01.011>>. .

LIN, H.; LUO, G.; ZHU, Y. A robust visual tracking method with restricted Boltzmann machine based classifier., 2016. Availabe on: <<http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.2244145>>. .

LIN, T. Y.; MAIRE, M.; BELONGIE, S.; et al. Microsoft COCO: Common objects in context. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 8693 LNCS, n. PART 5, p. 740–755, 2014.

LIU, C.; ZOPH, B.; NEUMANN, M.; et al. Progressive Neural Architecture Search. **arXiv preprint arXiv:1712.00559**, 2017.

LIU, W.; ANGUELOV, D.; ERHAN, D.; et al. SSD: Single shot multibox detector. **Lecture Notes in Computer Science (including subseries Lecture Notes in**

Artificial Intelligence and Lecture Notes in Bioinformatics), v. 9905 LNCS, p. 21–37, 2016.

LIU, X.; ZHANG, R.; MENG, Z.; HONG, R.; LIU, G. On fusing the latent deep CNN feature for image classification. **World Wide Web**, v. 22, n. 2, p. 423–436, 2019.

LUO, J. H.; WU, J.; LIN, W. ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression. **Proceedings of the IEEE International Conference on Computer Vision**, v. 2017-Octob, p. 5068–5076, 2017.

MONTGOMERY, D. C. **Introduction To Statistical Quality Control**. Wiely, 2005.

NAIR, V.; E. HINTON, G. **Rectified Linear Units Improve Restricted Boltzmann Machines**. 2010.

PAL, A.; DASGUPTA, R.; SAHA, A.; NANDI, B. Human-Like Sensing for Robotic Remote Inspection and Analytics. **Wireless Personal Communications**, v. 88, n. 1, p. 23–38, 2016. Springer US.

PÉREZ, L.; RODRÍGUEZ, Í.; RODRÍGUEZ, N.; USAMENTIAGA, R.; GARCÍA, D. F. Robot guidance using machine vision techniques in industrial environments: A comparative review. **Sensors (Switzerland)**, v. 16, n. 3, 2016.

PUTTEMANS, S.; CALLEMEIN, T.; GOEDEMÉ, T. Building Robust Industrial Applicable Object Detection Models using Transfer Learning and Single Pass Deep Learning Architectures. **Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications**. p.209–217, 2018. SCITEPRESS - Science and Technology Publications. Availabe on: <<http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0006562002090217>> . .

REDMON, J. YOLO: Real-Time Object Detection Website. Disponível em: <<https://pjreddie.com/darknet/yolo/>>. Visited on: 4/6/2019.

REDMON, J.; DIVVALA, S.; GIRSHICK, R.; FARHADI, A. You Only Look Once: Unified, Real-Time Object Detection. **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. p.779–788, 2016. IEEE. Availabe on: <<http://arxiv.org/abs/1506.02640>> . .

REDMON, J.; FARHADI, A. YOLO9000: Better, faster, stronger. **Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017**, v. 2017-Janua, p. 6517–6525, 2017.

REDMON, J.; FARHADI, A. YOLO v.3. , p. 1–6, 2018. Availabe on: <<https://pjreddie.com/media/files/papers/YOLOv3.pdf>> . .

RUDER, S. An overview of gradient descent optimization algorithms. , p. 1–14, 2016. Availabe on: <<http://arxiv.org/abs/1609.04747>> . .

RUSSELL, S.; NORVIG, P. **Artificial Intelligence A Modern Approach Third Edition**. 2010.

SANTURKAR, S.; TSIPRAS, D.; ILYAS, A.; MADRY, A. How Does Batch Normalization Help Optimization?, 2018. Availabe on: <<http://arxiv.org/abs/1805.11604>> . .

SATORRES MARTÍNEZ, S.; GÓMEZ ORTEGA, J.; GÁMEZ GARCÍA, J.; SÁNCHEZ GARCÍA, A. A machine vision system for defect characterization on transparent parts

with non-plane surfaces. **Machine Vision and Applications**, v. 23, n. 1, p. 1–13, 2012.

SCHMIDHUBER, J. Deep Learning in neural networks: An overview. **Neural Networks**, v. 61, p. 85–117, 2015. Elsevier Ltd.

SHI, S.; WANG, Q.; XU, P.; CHU, X. Benchmarking state-of-the-art deep learning software tools. **2016 7th International Conference on Cloud Computing and Big Data Benchmarking**. p.99–104, 2016.

SILVA, R. L.; RUDEK, M.; SZEJKA, A. L. Machine Vision Systems for Industrial Quality Control Inspections. **Product Lifecycle Management to support Industry 4.0**. p.1–10, 2018. Springer International Publishing.

SONG, X.; RUI, T.; ZHA, Z.; WANG, X.; FANG, H. The AdaBoost algorithm for vehicle detection based on CNN features. , p. 1–5, 2015.

SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. **Journal of Machine Learning Research**, v. 15, p. 1929–1958, 2014. Available on: <<http://jmlr.org/papers/v15/srivastava14a.html>>. .

SU, X.; ZHANG, Q. Dynamic 3-D shape measurement method: A review. **Optics and Lasers in Engineering**, v. 48, n. 2, p. 191–204, 2010. Elsevier. Available on: <<http://dx.doi.org/10.1016/j.optlaseng.2009.03.012>>. .

SUN, ZHUN; OZAY, METE; OKATANI, T. D. Design of kernels in convolutional neural networks for image classification. **European Conference on Computer Vision**, p. 51- 66., 2016.

SUN, J.; XIAO, Z. Potential fault region detection in TFDS images based on convolutional neural network. **Infrared Technology and Applications, and Robot Sensing and Advanced Control**, v. 10157, n. October 2016, p. 101571L, 2016.

SUNKARA, J. K.; SANTHOSH, M.; CHERUKURI, S. B.; KRISHNA, L. G. Object tracking techniques and performance measures - A conceptual survey. **IEEE International Conference on Power, Control, Signals and Instrumentation Engineering, ICPCSI 2017**, p. 2297–2305, 2018. IEEE.

SZEGEDY, C.; IOFFE, S.; VANHOUCHE, V.; ALEMI, A. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. **AAAI**, v. 4, p. 1–12, 2016. Available on: <<http://arxiv.org/abs/1602.07261>>. .

SZKILNYK, G. Vision-Based Fault Detection in Assembly. **Thesis**, 2012.

TANG, Y. Deep Learning using Linear Support Vector Machines. , 2013. Available on: <<http://arxiv.org/abs/1306.0239>>. .

TRUONG, T.-D.; NGUYEN, V.-T.; TRAN, M.-T. Lightweight Deep Convolutional Network for Tiny Object Recognition. **Proceedings of the 7th International Conference on Pattern Recognition Applications and Methods**, p. 675–682, 2018. Available on: <<http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0006752006750682>>. .

VERGARA-VILLEGAS, O. O.; CRUZ-SÁNCHEZ, V. G.; OCHOA-DOMÍNGUEZ, H. J.; NANDAYAPA-ALFARO, M. J.; FLORES-ABAD, Á. Automatic Product Quality Inspection Using Computer Vision Systems. **Lean Manufacturing in the**

Developing World, , n. September 2016, 2014. Available on:
<<http://link.springer.com/10.1007/978-3-319-04951-9>>. .

WANG, J.; MA, Y.; ZHANG, L.; GAO, R. X.; WU, D. Deep learning for smart manufacturing: Methods and applications. **Journal of Manufacturing Systems**, p. 1–13, 2018. Available on: <<https://doi.org/10.1016/j.jmsy.2018.01.003>>. .

WANG, P.; CHEN, P.; YUAN, Y.; et al. Understanding Convolution for Semantic Segmentation. **Proceedings - 2018 IEEE Winter Conference on Applications of Computer Vision, WACV 2018**, v. 2018-Janua, p. 1451–1460, 2018.

WU, S.; NAGAHASHI, H. Analysis of generalization ability for different AdaBoost variants based on classification and regression trees. **Journal of Electrical and Computer Engineering**, v. 2015, 2015.

YAN, C.; CHEN, W.; CHEN, P. C. Y.; S, K. A.; WU, X. A New Two-Stage Object Detection Network without RoI-Pooling. **2018 Chinese Control And Decision Conference (CCDC)**, p. 1680–1685, 2018. IEEE.

YAO, C.; SUN, P.; ZHI, R.; SHEN, Y. Learning coexistence discriminative features for multi-class object detection. **IEEE Access**, v. 6, n. c, p. 37676–37684, 2018. IEEE.

YIN, X.; CHEN, L.; ZHANG, X.; GAO, Z. Object Detection Implementation and Optimization on Embedded GPU System. **IEEE International Symposium on Broadband Multimedia Systems and Broadcasting, BMSB**, v. 2018-June, p. 1–5, 2018. IEEE.

YOUNG, S.; ABDU, T.; BENER, A. Deep Super Learner: A Deep Ensemble for Classification Problems. , 2018. Available on: <<http://arxiv.org/abs/1803.02323>>. .

YU, Q.; WANG, J.; ZHANG, S.; GONG, Y.; ZHAO, J. Combining local and global hypotheses in deep neural network for multi-label image classification. **Neurocomputing**, v. 235, n. August 2015, p. 38–45, 2017. Elsevier B.V. Available on: <<http://dx.doi.org/10.1016/j.neucom.2016.12.051>>. .

YU, S.; WU, Y.; LI, W.; SONG, Z.; ZENG, W. A model for fine-grained vehicle classification based on deep learning. **Neurocomputing**, v. 257, p. 97–103, 2017. Elsevier B.V. Available on: <<http://dx.doi.org/10.1016/j.neucom.2016.09.116>>. .

ZHANG, G.; CHEN, L.; DING, Y. A Multi-Label Classification Model Using Convolutional Natural Networks. , p. 2151–2156, 2017.

ZHANG, X.; LIU, L.; XIE, Y.; et al. Rotation Invariant Local Binary Convolution Neural Networks. **Proceedings - 2017 IEEE International Conference on Computer Vision Workshops, ICCVW 2017**, v. 2018-Janua, n. c, p. 1210–1219, 2018.

ZHANG, X.; YANG, Y.-H.; HAN, Z.; WANG, H.; GAO, C. Object class detection. **ACM Computing Surveys**, v. 46, n. 1, p. 1–53, 2013. Available on: <<http://dl.acm.org/citation.cfm?doid=2522968.2522978>>. .

ZHANQUAN, S.; FOX, G. Large Scale Classification Based on Combination of Parallel SVM and Interpolative MDS. , v. 3, p. 545–555, 2012.

ZHAO, B.; FENG, J.; WU, X.; YAN, S. A survey on deep learning-based fine-grained object classification and semantic segmentation. **International Journal of Automation and Computing**, v. 14, n. 2, p. 119–135, 2017.

ZHIQIANG, W.; JUN, L. A review of object detection based on convolutional neural

network. **Chinese Control Conference, CCC**, p. 11104–11109, 2017. Available on: <<http://ieeexplore.ieee.org/document/8029130/>>. .

ZHONG, R. Y.; XU, X.; KLOTZ, E.; NEWMAN, S. T. Intelligent Manufacturing in the Context of Industry 4.0: A Review. **Engineering**, v. 3, n. 5, p. 616–630, 2017. Elsevier LTD. Available on: <<http://dx.doi.org/10.1016/J.ENG.2017.05.015>>. .