



**Pedro Henrique Lopes Torres**

**A robust real-time component for personal  
protective equipment detection in an industrial  
setting**

**Dissertação de Mestrado**

Dissertation presented to the Programa de Pós-graduação em  
Informática of PUC-Rio in partial fulfillment of the requirements  
for the degree of Mestre em Informática.

Advisor : Prof. Hélio Côrtes Vieira Lopes  
Co-advisor: Prof. Thuener Armando da Silva

Rio de Janeiro  
April 2021



**Pedro Henrique Lopes Torres**

**A robust real-time component for personal  
protective equipment detection in an industrial  
setting**

Dissertation presented to the Programa de Pós-graduação em  
Informática of PUC-Rio in partial fulfillment of the requirements  
for the degree of Mestre em Informática. Approved by the  
Examination Committee.

**Prof. Hélio Côrtes Vieira Lopes**

Advisor

Departamento de Informática – PUC-Rio

**Prof. Thuener Armando da Silva**

Co-advisor

Greensill

**Prof. Marcelo Gattass**

Departamento de Informática – PUC-Rio

**Dr. Guilherme Gonçalves Schardong**

Departamento de Informática – PUC-Rio

Rio de Janeiro, April 14<sup>th</sup>, 2021

All rights reserved.

## **Pedro Henrique Lopes Torres**

Bachelor's in Computer Science (2018) at the Federal University of Ceará (UFC), with emphasis in Computer Vision.

### Bibliographic data

Torres, Pedro Henrique Lopes

A robust real-time component for personal protective equipment detection in an industrial setting / Pedro Henrique Lopes Torres; advisor: Hélio Côrtes Vieira Lopes; co-advisor: Thuener Armando da Silva. – Rio de Janeiro: PUC-Rio, Departamento de Informática, 2021.

v., 61 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui bibliografia

1. Artificial Intelligence;. 2. Industrial Application;. 3. Computer Vision;. 4. Real-time System;. 5. Detection Equipment.. I. Lopes, Hélio Côrtes Vieira. II. Silva, Thuener Armando. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. IV. Título.

CDD: 004

## Acknowledgments

To make it easier for everyone I will do my acknowledgments in Brazilian Portuguese...

Agradeço imensamente aos meus pais, Maria José e José Martins, por sempre acreditarem em mim e fazerem o possível e o impossível para que eu chegasse até aqui. Valeu mãe por toda a paciência que teve em educar aquele moleque trabalhoso que vivia te dando preocupação. Valeu pai por me mostrar a partir de seu próprio exemplo o que é ser uma pessoa justa e leal.

Agradeço a minha avó, Constância Rabelo, a mulher mais forte e incrível que já conheci. Obrigado por todas as nossas conversas naquele balcão. Aprendi muito com você avó e tenho certeza que vou levar esse aprendizado para o resto da vida. Hoje você é uma estrela que continua me guiando por meio de sua linda luz. Essa é mais uma conquista dedicada para você vó. Obrigado.

Agradeço ao meu irmão, Alexandre Rabelo, que esteve sempre presente quando eu precisei, por torcer e acreditar em mim e por sempre ter me incentivado aos estudos. Valeu Alexandre, pode contar comigo, estamos juntos!

Agradeço ao meu orientador, professor Hélio Lopes, por sua valiosa orientação, por acreditar em mim e por todas as oportunidades que me proporcionou e que me ajudaram a crescer. Valeu Hélio, sou muito grato de ter sido seu aluno e de poder trabalhar com você.

Agradeço ao meu coorientador, Thuener Silva, por sua valiosa orientação, por acreditar em mim e por tudo o que me ensinou. Foi uma honra trabalhar com você. Essa jornada teria sido muito mais difícil se eu não pudesse contar com pessoas como você. Valeu!

Agradeço aos membros da banca avaliadora, professor Marcelo Gattass e Guilherme Schardong pelas contribuições dadas para o desenvolvimento deste trabalho.

Agradeço a todos os professores e demais funcionários do Departamento de Informática (DI) da PUC – Rio. É uma honra compartilhar esse espaço com tanta gente massa. Muito aprendizado, valeu DI!

Agradeço ao meu amigo Arthur, por estar sempre presente e torcendo por mim. Valeu meu brother, agora a meta é montar um carro de *drift* e sair por aí soltando pipa. R15.

Agradeço ao meu amigo Rodrigo, por ter encarado comigo o desafio de encarar um mundo novo e ir fazer mestrado longe de casa. Rodrigo, tu sabe que a gente é fechamento. Valeu, meu querido!



Agradeço ao Dalai, um grande amigo que fiz durante essa jornada. Você me ajudou muito nos mais diversos momentos, muito obrigado. Estamos juntos irmãozinho!

Agradeço a toda galera do laboratório DaSLab, especialmente ao André (*log n*), Rômulo e Vinicius. Foi muito massa trabalhar com vocês e espero levar essa parceria pra vida. Contem comigo!

Agradeço a minha musa, Maria Juliana, por sempre acreditar em mim e estar ao meu lado em todo e qualquer momento. Só tenho a agradecer por ter uma mulher tão incrível ao meu lado, o tanto que aprendi nessa caminhada com você não foi brincadeira. Te amo, Juliana. Obrigado por aguentar minhas loucuras.

Agradeço a arte, a música, que fez a trilha sonora de toda essa jornada. Obrigado por estar sempre fazendo barulho na minha cabeça.

Por último, mas não menos importante, fazendo das palavras de Snoop Dogg as minhas, quero me agradecer. Quero me agradecer por acreditar em mim. Eu quero me agradecer por fazer todo o trabalho duro. Eu quero me agradecer por não ter dias de folga. Eu quero me agradecer por nunca desistir. Eu quero me agradecer por sempre ser um doador e tentar dar mais do que eu recebo. Eu quero me agradecer por tentar fazer mais o certo do que o errado. Eu quero apenas me agradecer por ser eu o tempo todo.

To Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) for partially financing this research under grant 132135/2019-1.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

## Abstract

Torres, Pedro Henrique Lopes; Lopes, Hélio Côrtes Vieira (Advisor); Silva, Thuener Armando (Co-Advisor). **A robust real-time component for personal protective equipment detection in an industrial setting**. Rio de Janeiro, 2021. 61p. Dissertação de mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

In large industries, such as construction, metallurgy, and oil, workers are continually exposed to various hazards in their workplace. Accordingly to the International Labor Organization (ILO), there are 340 million occupational accidents annually. Personal Protective Equipment (PPE) is used to ensure the essential protection of workers' health and safety. There is a great effort to ensure that these types of equipment are used properly. In such an environment, it is common to have closed-circuit television (CCTV) cameras to monitor workers, as those can be used to verify the PPE's proper usage. Some works address this problem using CCTV images; however, they frequently can not deal with multiples safe equipment usage detection and others even skip the verification phase, making only the detection. In this paper, we propose a novel cognitive safety analysis component for a monitoring system. This component acts to detect the proper usage of PPE's in real-time using data stream from regular CCTV cameras. We built the system component based on the top of state-of-art deep learning techniques for object detection. The methodology is robust with consistent and promising results for Mean Average Precision (mAP) and can act in real-time.

## Keywords

Artificial Intelligence; Industrial Application; Computer Vision; Real-time System; Detection Equipment.

## Resumo

Torres, Pedro Henrique Lopes; Lopes, Hélio Côrtes Vieira; Silva, Thuener Armando. **Um componente robusto em tempo real para detecção de equipamentos de proteção individual em um ambiente industrial**. Rio de Janeiro, 2021. 61p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Em grandes indústrias, como construção, metalúrgica e petróleo, trabalhadores são continuamente expostos a vários tipos de perigos em seus locais de trabalho. Segundo a Organização Internacional do Trabalho (OIT), anualmente ocorrem cerca de 340 milhões de acidentes de trabalho. Equipamentos de Proteção Individual (EPI) são utilizados para garantir a proteção essencial da saúde e segurança dos trabalhadores. Com isto, há um grande esforço para garantir que esses tipos de equipamentos sejam usados de maneira adequada em ambientes de trabalho. Em tais ambientes, é comum ter câmeras de circuito fechado de televisão (CFTV) para monitorar os trabalhadores, pois essas podem ser usadas para verificar o uso adequado de EPIs. Alguns trabalhos presentes na literatura abordam o problema de verificação automática de EPIs usando imagens de CFTV como entrada; no entanto, muitos destes trabalhos não conseguem lidar com a detecção de uso seguro de múltiplos equipamentos e outros até mesmo pulam a fase de verificação, fazendo apenas a detecção. Neste trabalho, propomos um novo componente de análise de segurança cognitiva para um sistema de monitoramento. Este componente atua para detectar o uso adequado de EPIs em tempo real, usando fluxo de dados de câmeras de CFTV comuns. Construímos este componente do sistema com base nas melhores técnicas de Aprendizado Profundo voltadas para a tarefa de detecção de objetos. A metodologia proposta é robusta com resultados consistentes e promissores em termos da métrica *Mean Average Precision* (mAP) e pode atuar em tempo real.

## Palavras-chave

Inteligência Artificial; Aplicação Industrial; Visão Computacional; Sistema em tempo real; Detecção de Equipamentos.

## Table of contents

1	Introduction	14
1.1	Document Structure	16
2	Background	17
2.1	Machine Learning	17
2.1.1	Supervised Learning	17
2.1.2	Unsupervised Learning	18
2.2	Learning Algorithms	18
2.3	Deep Learning	20
2.3.1	How Learning Happens	22
2.4	Computer Vision	23
2.4.1	Convolutional Neural Networks	25
2.4.2	Deep Neural Networks for Object Detection	29
3	Related Works	33
3.1	Sensor-Based	33
3.2	Vision-Based	34
3.3	Vision-Based Methods Using Deep Learning	35
4	Methodology	39
4.1	Dataset Generation	40
4.2	Data Preprocessing	41
4.3	Data Augmentation	43
4.4	Approach-I	44
4.5	Approach-II	45
4.6	YOLO-v4 Model Architectures	46
4.7	Model Training	47
4.8	Mean Average Precision for Performance Evaluation	48
5	Experimental Results	50
5.1	Performance of the YOLO-v4 Models	50
5.2	Performance of Classifiers	51
5.3	Performance Comparison of Approaches	52
5.4	Benchmark of Results	53
6	Conclusion	55

## List of figures

Figure 1.1	Example of industrial system to monitor the use of PPEs.	15
Figure 1.2	Detection and verification stages.	16
Figure 2.1	Machine Learning algorithms.	19
Figure 2.2	Comparison of intuitive clusters versus clusters found by K-means (Developers, 2021).	20
Figure 2.3	Deep representations learned by a object-classification model.	21
Figure 2.4	Neural network learning process. Adapted from Francois (2017).	22
Figure 2.5	A rough timeline of some of the most active topics of research in computer vision. Adapted from Szeliski (2010).	24
Figure 2.6	Comparison of GPUs performance along the years (Sapunov, 2018).	24
Figure 2.7	Convolutional Neural Network example.	25
Figure 2.8	VGG-16 architecture (Neurohive, 2018).	26
Figure 2.9	Two $3 \times 3$ convolutions replacing one $5 \times 5$ convolution (Szegedy et al., 2016).	27
Figure 2.10	One $3 \times 1$ convolution followed by one $1 \times 3$ convolution replacing one $3 \times 3$ convolution (Szegedy et al., 2016).	27
Figure 2.11	Inception-v3 architecture (Szegedy et al., 2016).	28
Figure 2.12	A building block for residual learning (Zhang et al., 2020).	28
Figure 2.13	ResNet architecture (He et al., 2016a).	29
Figure 2.14	R-CNN flowchart (Girshick et al., 2014).	30
Figure 2.15	YOLO-v4 detection step.	32
Figure 3.1	IoT-based architecture of the smart hard-hat system (Zheng et al., 2019).	34
Figure 3.2	Comparison of the overall performance of each approach (Nath et al., 2020).	38
Figure 4.1	Possible classes to perform classification.	39
Figure 4.2	Distribution of instances per class for each data source.	41
Figure 4.3	Number of instances per class for each subset.	42
Figure 4.4	Corresponding anchor boxes for the nine centroids.	43
Figure 4.5	Data augmentation transformations examples.	43
Figure 4.6	Mosaic data augmentation (Bochkovskiy et al., 2020).	44
Figure 4.7	Approach-I.	45
Figure 4.8	Approach-II.	45
Figure 4.9	Ensemble of classifiers (majority vote).	46
Figure 4.10	Intersection over union (IoU).	48
Figure 5.1	Performance of the proposed models for each approach.	51
Figure 5.2	Confusion matrices for VGG-16, Inception, ResNet-50 and ResNet-101 classifier models.	51

Figure 5.3	Performance comparison of approaches implementation.	52
Figure 5.4	Processing time comparison for approaches implementation.	53
Figure 5.5	Example of detections obtained from YOLO-v4-AP1 model. The first two images are from the Crowd-sourced dataset. The third image is from the Web-scraped dataset that is in an industrial setting.	54

## List of tables

Table 3.1	YOLO-v3 fine-tuned model results (Bo et al., 2019).	36
Table 3.2	Detection results for each PPE type (Zheng et al., 2019).	37
Table 5.1	mAP, precision, recall and f1-score metrics for YOLO-v4-AP1 and YOLO-v4-AP2 models.	50
Table 5.2	Detailed result of each implementation for the values of mAP, precision, recall, and f-score.	53
Table 5.3	Comparison of mAP and FPS values with a baseline model.	54

## List of abbreviations

AI – Artificial Intelligence  
AP – Average Precision  
BN – Batch Normalization  
CCTV – Closed-circuit television  
CNN – Convolutional Neural Networks  
CV – Computer Vision  
DL – Deep Learning  
DNN – Deep Neural Networks  
FN – False Negative  
FP – False Positive  
FPS – Frames per second  
IoT – Internet of Things  
NN – Neural Network  
PCA – Principal Component Analysis  
PPE – Personal protective equipment  
SVM – Support Vector Machine  
TP – True Positive  
YOLO – You only look once



*Salve guerreiras e guerreiros  
Escolas libertárias e terreiros  
Amores de todos os tipos  
Verdades de todas as cores.*

*Salve nossos ancestrais e espíritos prote-  
tores  
Coragem, atenção, irmã, irmão  
Fé no coração, sempre siga em frente  
Respiração profunda oxigena corpo e mente  
Que a esperança renasça como a fênix.*

*Resistência, resistência  
Força, sabedoria, inteligência, coletividade  
Serenidade em tempos de tempestade  
Liberdade sempre.*

*Salve.*

**BaianaSystem**, *O futuro não demora - Salve.*

# 1

## Introduction

Workers, especially in an industrial setting, are continually exposed to various hazards in their workplace. Unfortunately, this susceptibility can lead to several occupational injuries. The Brazilian Protection Statistical Yearbook<sup>1</sup> shows an average of six hundred thousand occupational accidents and 2600 deaths per year, registered between 2010 and 2017. For this reason, regulatory agencies require that workers operating in areas that are subject to exposure to hazards make appropriate use of personal protective equipment (PPE).

A company, such as an oil and gas refinery, could avoid occupational injuries by monitoring its workers to prompt corrective measures when there is no use (or inadequate use) of personal protective equipment. This activity is often performed by a human from a constant visual local inspection or closed-circuit television (CCTV) but looking from a practical perspective; this task can be exhaustive since monitoring a large number of workers for PPE compliance takes a lot of time and resources (Mneymneh et al., 2019). In this scenario, an industry could benefit from a system powered by Machine Learning and Computer Vision techniques to automate this task to prevent accidents and minimize costs. Figure 1.1 illustrates a possible industrial system to monitor the use of PPE automatically and emit alarms when they are missing or not used appropriately. The system is supplied with RGB images from a CCTV, each image initially passes through the detection and verification component. This component is the fundamental basis of the system and is responsible for producing evidence of deviations from inappropriate use of workers' equipment. Finally, the ID association component can match those evidence with the worker's identity in the company database and alert the worker's identification and type of deviation.

In literature, the methods used for automated monitoring of PPE compliance can be categorized into two types: sensor-based and vision-based. The sensor-based methods usually install a sensor in each personal protective equipment (commonly just hard hat) and process emitted signals continuously to monitor PPE compliance (Barro-Torres et al., 2012; Naticchia et al., 2013). Al-

<sup>1</sup><https://bc.pressmatrix.com/pt-BR/profiles/1227998e328d/editions/0e55e8eba33a3ed62b2e/pages/page/40>

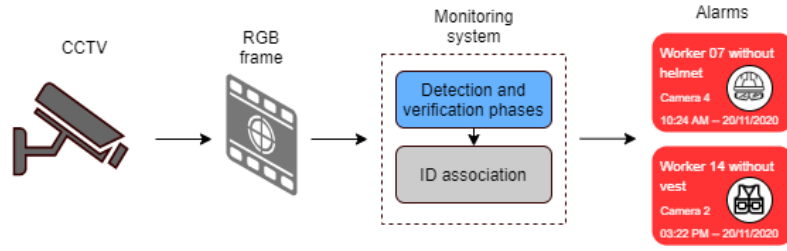


Figure 1.1: Example of industrial system to monitor the use of PPEs.

though, this approach requires the acquisition, installation, and maintenance of several sensors, which can demand a significant investment for a company. In contrast, the vision-based methods use cameras installed in the work environment to process images using Computer Vision techniques to provide a richer comprehension of the scene, enabling automated PPE compliance (Seo et al., 2015).

Regarding vision-based methods, most of the existing works for monitoring PPE compliance simply focus on identifying one type of PPE: hard hats (Fang et al., 2018; Wu et al., 2019; Mneymneh et al., 2019). This suggests that multiple PPE detection is a more challenging problem. But, despite being scarce, already possible thanks to the advancement of Deep Learning algorithms (Zheng et al., 2019; Nath et al., 2020). It is important to emphasize that identifying only the presence of PPE in an image is not enough. It is also necessary to recognize the contextual relationship of the use of PPE by a worker. This verification of proper PPE usage remains an open question and needs to be answered from a contextual analysis of the work environment (Nath et al., 2020). In this way, the PPE monitoring compliance problem can be addressed in two stages: detection and verification 1.2. The detection step consists of locating workers and equipment. The verification step analyzes if a worker is appropriately using PPE components. It is worth mentioning that when dealing with the problem of identifying multiple PPE, the level of complexity of the verification step grows exponentially; since there are  $2^n$  possible combinations of use for  $n$  different types of PPE. For this study, we choose to focus on two types of PPE, hard hat and protective clothing. Since these equipment are often used to ensure the safety of workers in the oil and gas industry. Thus, we have  $n = 2$ , which leads to four ( $2^2 = 4$ ) combinations, that is, a worker using no equipment, a worker wearing hard hat, a worker wearing protective clothing, and a worker wearing both types of equipment.

Although important in the scenario described in the Figure 1.1, we will not conduct an in-depth exploration of the challenges related to the ID Identification component, as they lay outside of the scope of this work. We shall



Figure 1.2: Detection and verification stages.

focus on exploring the two main approaches for implementing a detection and verification component (highlighted as the blue box of the Figure 1.1) that is both robust and capable of act in real-time for monitoring systems in an industrial environment, especially oil and gas refineries.

Our main contributions are:

- Evaluating two approaches to solve the PPE detection problem. In our first approach, we built a one-stage classifier. While in our second approach, we build a multi-stage classifier.
- A dataset for PPE detection that addresses multiple types of equipment.
- Exploring how ensemble classifiers performs for the verification stage of a multi-stage implementation.

## 1.1

### Document Structure

This work is structured as follows: Chapter 2 presents the theoretical background for Machine Learning, Deep Learning, and Computer Vision methods addressed in this work; Chapter 3 presents a literature review about works that address the problem of detecting personal protectives equipment automatically using sensor and vision-based methods; Chapter 4 describes the construction and exploration process adopted for the proposed approaches to automatically detect and verify the appropriate use of equipment by workers. The Chapter 4 also includes details regarding dataset generation; Chapter 5 presents the performance comparison of models and approaches; Chapter 6, the conclusion.

## 2 Background

### 2.1 Machine Learning

Machine Learning is a subfield of Artificial Intelligence (AI) that has been consolidated over time based on concepts explored in other areas, such as Statistics and Computer Theory. The studies carried out in the Machine Learning area are focused on building computer programs that can automatically improve from a set of experiences. As Mitchell (1997) defined: “A computer program is said to learn from experience  $E$  with respect to some class of tasks  $T$  and performance measure  $P$ , if its performance at tasks in  $T$ , as measured by  $P$ , improves with experience  $E$ ”.

Commonly, Machine Learning problems can be divided into two main branches: supervised learning and unsupervised learning. There is also a third branch (Reinforcement Learning) that can be included, but for the scope of this work we will not be addressing it.

#### 2.1.1 Supervised Learning

In Supervised Learning, the algorithms are designed to learn by examples. Given a data set made up of  $N$  pairs  $(x_i, y_i)$ , where  $y_i$  is the output corresponding to the entry  $x_i$ . The algorithm aims to learn a function that maps an input  $x$  to an output  $y$ , commonly called label. After training, the algorithm will receive new inputs out of the data set used to learn and determine which label the new inputs will be classified as based on prior training data. In a simplistic form, a supervised learning algorithm can be described as  $Y = f(x)$ . Where  $Y$  is the predicted output, determined by a mapping function ( $f$ ) that assigns a output value to an input value  $x$ . Supervised learning problems are categorized into two types: classification and regression. In a classification problem, the algorithm tries to predict results in a discrete output, such as animal classification. Given  $x$  find  $y \in \{1, \dots, k\}$  with  $k \in \mathbb{N}$ . Where  $k$  is the number of possible classes in the classification problem. In a regression problem, the algorithm tries to predict results within a continuous

output, such as predict the profit of a store during Christmas. Given  $x$  find  $y \in \mathbb{R}$ .

### 2.1.2

#### Unsupervised Learning

Unsupervised Learning is the branch of Machine Learning that is used to find underlying patterns in data. Since the data does not need to be labeled, classified, or categorized. This allows approaching problems with little knowledge of data variables. In Unsupervised Learning, instead of responding to feedbacks (as in Supervised Learning), the algorithms will often find subgroups or hidden patterns within the dataset that can not be easily observed.

Two types of tasks that stand out in Unsupervised Learning are clustering and dimensionality reduction. In clustering, the input data is automatically partitioned, creating groups according to the degree of similarity present in the data distribution. In general, some patterns on a two-dimensional are easy to see. But when dealing with real-world problems, we often deal with higher-dimensional data. For these type of problems, dimensionality reduction algorithms, like Principal Component Analysis (PCA) (Abdi and Williams, 2010), can be very effective. The goal is finding a subspace representing the data, in which turn possible the data projection, for a better analysis.

## 2.2

### Learning Algorithms

Once we identify the branch that a problem is in, it is time to choose an algorithm. The literature has a range of approaches that can be used in the most diverse types of problems. As there is no single solution, it is crucial to choose an algorithm that best suits the input data, giving the expected outputs. Here, we choose to briefly describe one of some commonly adopted Machine Learning approaches for each type of task (Figure 2.1):

- **Support Vector Machine (SVM):** A supervised learning algorithm that is often used for classification tasks but is also suitable for regression tasks. The algorithm builds a non-linear classifier by finding the hyperplane that best separates data points into two classes. The distance between the closest data points and the hyperplane is referred to as the margin. The larger the margin, the more optimal the hyperplane is. Despite being an accurate algorithm, especially when dealing with high dimensional spaces where the number of dimensions is greater than

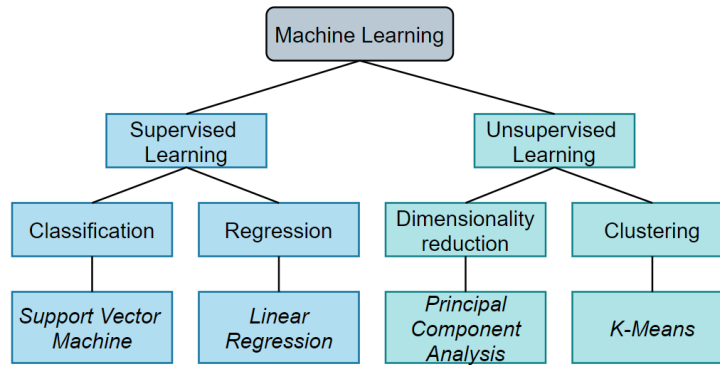


Figure 2.1: Machine Learning algorithms.

the number of samples, SVMs do not scale well over a large dataset; because the required training time is considerably high.

- **Linear Regression:** A supervised learning algorithm that models the relationship between the set of input variables ( $x$ ) and an output variable ( $y$ ). This output is a continuous variable and has a constant slope. There are two main types of Linear Regression, where both have simple representations:
  - **Simple regression:** When there is only one input, the Linear Regression is expressed as the traditional slope-intercept form  $y = mx + b$ . The algorithm finds out and evaluates the values of the coefficients  $m$  and  $b$  that produce the most accurate predictions.
  - **Multi-variable regression:** In practice, the problems have more than one input, in that case, we have the a multi-variable linear equation  $f(x_1, \dots, x_n) = w_1x_1 + \dots + w_nx_n$ , where  $w$  represents the coefficients (model weights), that the model will learn. The variables  $(x_1, \dots, x_n)$  represents the attributes that we have about each observation.
- **Principal Component Alaysis (PCA):** PCA is a linear dimensionality reduction algorithm that, in simple terms, derives a new set of features from the existing ones. The algorithm seeks to keep as much information as possible in a reduced dimension. These new derived features are called principal components. Each component is a linear combination of the original variables, the order of these components is determined according to the variance of the explainable data. Since each component is orthogonal to one another, there is zero correlation between these components.

The dimensionality reduction brings some gains, for example, the removal of highly correlated features. These features usually end up hindering

learning models in their decision making. Besides, the training time of the algorithms reduces significantly with less number of features. In terms of data visualization, when we reduce the number of features and bring the problem to a dimension where we can visualize, is easier to have a better understanding of the problem.

- **K-Means:** One of the simplest and popular unsupervised machine learning clustering algorithms. K-means implements a partition-based clustering technique that aims to partition data into  $k$  clusters in a way that points belonging to the same cluster are the most similar. The similarity of points  $p$  and  $q$  is determined by the distance between them. During partitioning, K-means tries to minimize distances between points that share the same cluster and maximize the distance between *centroids* of different clusters. One of the tasks that require attention is the definition of the number of clusters ( $k$ ) to be used. This task can be challenging since the K-means is not able to calculate the number of clusters that best fits the data. In this way, if there is a non-linear structure separating groups in the data, K-means will not work appropriately. K-means is also good in generalizes to clusters of different shapes and sizes but can have some problems with certain datasets. Figure 2.2 compares the intuitive clusters on the left side with the clusters actually found by K-means on the right side. Note that clusters with different densities and sizes can present some challenges.

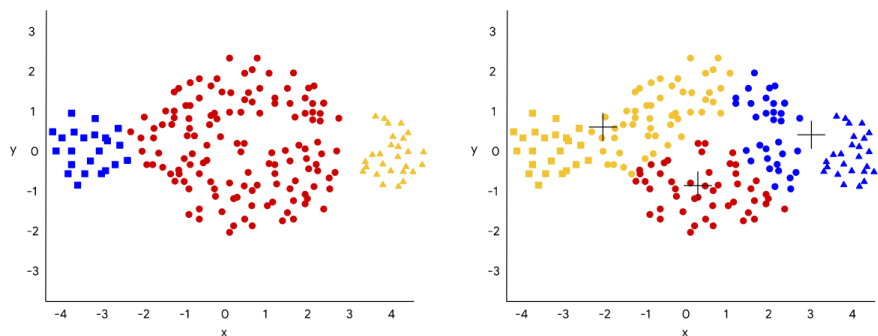


Figure 2.2: Comparison of intuitive clusters versus clusters found by K-means (Developers, 2021).

## 2.3 Deep Learning

In the previous section, we presented some of the principal Machine Learning approaches and examples of learning algorithms that can solve specific types of tasks. However, when using conventional Machine Learning



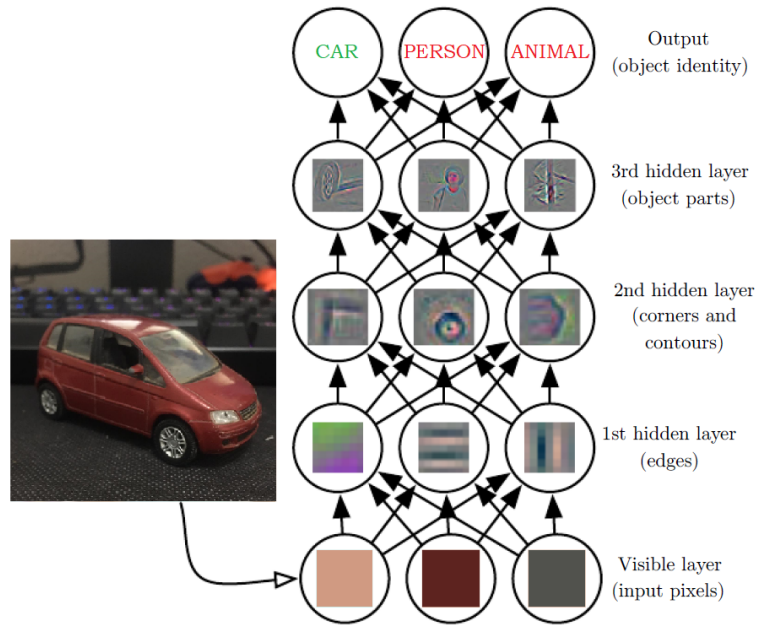


Figure 2.3: Deep representations learned by a object-classification model.

algorithms, performance depends heavily on the *representation* of the given data. When we know how to design the right set of features for solving a task it is possible to take great advantage of such algorithms. Nevertheless, sometimes, it is not easy to know what features should be extracted. One approach to get rid of this problem is using a *representation learning* algorithm. Such type of algorithm not only discovers the mapping from representation to output but also the representation itself. This technique makes possible to automatically discover a good set of features, saving precious time and effort for researchers (Goodfellow et al., 2016). It is in this context that the use of Deep Learning becomes extremely relevant.

Deep Learning is a specific subfield of Machine Learning based on learning data representations, allowing a computer to build complex concepts out of more straightforward concepts. The *deep* in Deep Learning comes from the idea of successive layers of representations. The modern deep learning architectures often have hundreds of successive layers, where a model, often called deep neural networks, learn these representations.

Figure 2.3 illustrates how a deep learning model can perform an image recognition task, combining simpler concepts, such as corners and contours, which are defined in terms of edges, to recognize the objects present in the image. What the network is doing is transforms the car image into representations. These representations are different from the original image, but they are more informative to reach the final result.

### 2.3.1 How Learning Happens

The purpose of deep neural networks is to do input-to-target mapping via a deep sequence of data transformations learned by exposure to examples. The transformations that occur when a layer receives its input data are implemented from a parametrization of the layer's weights (illustrated as the blue box in Figure 2.4). In this context, the learning process is based on finding the right values for all layer's weights. Such that the network will correctly map data inputs to their associated targets (Francois, 2017). But there is a problem. A deep neural network contains millions of parameters. Finding the correct values for all these parameters is not an easy task, since the modification of one parameter will affect the behavior of all the others.

To manage the output of a neural network, we have to be able to measure if the results (predictions) are matching what is expected. For this, we can use a *loss function*. The loss function takes the network's predictions and the true target and computes a distance score. With this, it is possible to know how well the network is going for a specific set of examples (illustrated as the green box in Figure 2.4). In addition, this score serves as feedback to update the network parameters (weight values). This update is the job of the *optimizer*, through the implementation of the central algorithm in deep learning, called *backpropagation*, that seeks to minimize the loss score, making the predictions of a set of examples made by the network closer to the real targets. Figure 2.4 illustrates how the learning process occurs.

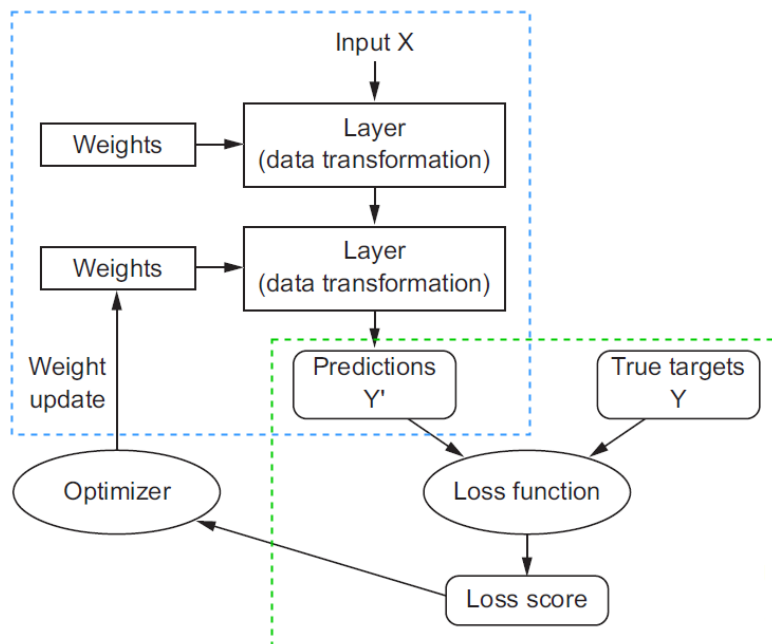


Figure 2.4: Neural network learning process. Adapted from Francois (2017).

It is worth mentioning, that initially the network merely implements a series of random transformations since all the weights are randomly assigned. With this, is expected that outputs are far from what it should be, which results in a very high loss score. During the network training phase, for each input example that net processes, the weights are adjusted, correcting the direction of predictions. Consequently decreasing the loss score. This process is repeated for a sufficient number of iterations and over thousands of examples, to make the network produce the outputs as close to the targets.

## 2.4

### Computer Vision

Computer Vision (CV) is an interdisciplinary field interested in making machines capable of understanding and interpreting the visual world from the extraction of information in digital images. The Computer Vision field emerged in the early 1970s with concepts similar to those approached by the Digital Image Processing area. It didn't take long for the two areas to become distinct since CV methods started to use information extracted from real-world images to take a new step, which sought to enable a total understanding of a scene (Szeliski, 2010). Briefly, Computer Vision works based on three basic steps:

- **Image acquisition:** in this step, cameras are used to collect images (2D or 3D) or videos in real-time;
- **Image processing:** in this step, it is common to use conventional image processing methods, such as linear and non-linear filters application, geometric transformations. To get an enhanced image or feature information that may be useful later;
- **Image understanding:** this final step is interpretative, in which it is sought to identify or classify different types of objects about an image.

Figure 2.5 presents some of the most active topics of research in Computer Vision until today. In a more actual context, it is possible to notice many CV applications that perform a visual recognition task combined with sophisticated Machine Learning techniques to solve real-life problems. For example, the use of CV in the manufacturing industry to automatically identify and in real-time defects in pieces resulting from a production line, using only images or video processing, enables cost reduction and safety.

The combination of Machine Learning and Computer Vision brought considerable gains and ended up being another factor that made AI become mainstream in the last decade (2010-20). Once, the Big Data field caused an

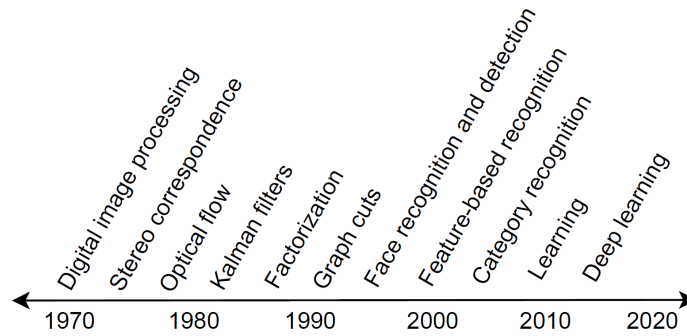


Figure 2.5: A rough timeline of some of the most active topics of research in computer vision. Adapted from Szeliski (2010).

explosion in the growth of data volume, which was a requirement to realize that sufficiently large models trained on sufficiently large data (a byproduct of the rise of the consumer internet) was all that was necessary for the emergence of Deep Learning (Francois, 2017). That led to the construction of significantly large datasets containing hundreds of thousands to tens of millions of examples, a crucial factor for the advancement of Deep Learning and Computer Vision. Some examples of these datasets are: ImageNet (Russakovsky et al., 2015), Sports-1M (Karpathy et al., 2014) and Street View House Numbers (Netzer et al., 2011). The need to develop larger models has also led to a series of studies regarding parallel GPU computing, bringing a considerable advance in GPU performance in recent years (Figure 2.6). This trend is expected to remain positive in the future (Goodfellow et al., 2016).

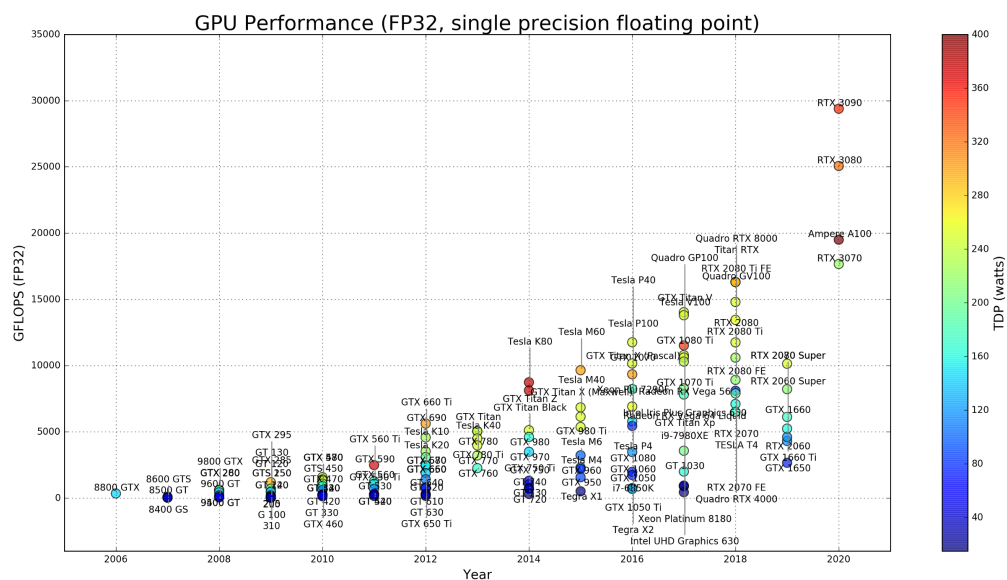


Figure 2.6: Comparison of GPUs performance along the years (Sapunov, 2018).

### 2.4.1 Convolutional Neural Networks

The use of Deep Learning methods has achieved state of the art in different Computer Vision tasks in recent years. Convolutional Neural Networks (CNN) have emerged as an important approach to perform a broad range of visual tasks (Krizhevsky et al., 2012; Toshev and Szegedy, 2014; Long et al., 2015; Ren et al., 2015). CNNs are composed of layers of filters that represent neighborhood spatial connectivity patterns. It is the use of convolutions, non-linear activation functions, and downsampling (pooling), commonly followed by one or more fully connected layers, that results in a hierarchical understanding of image features. Figure 2.7 illustrates a CNN that combines the extraction of different features along with the network, sought for the formation of high-level features that enable the classification of an image based on the attribution of a label.

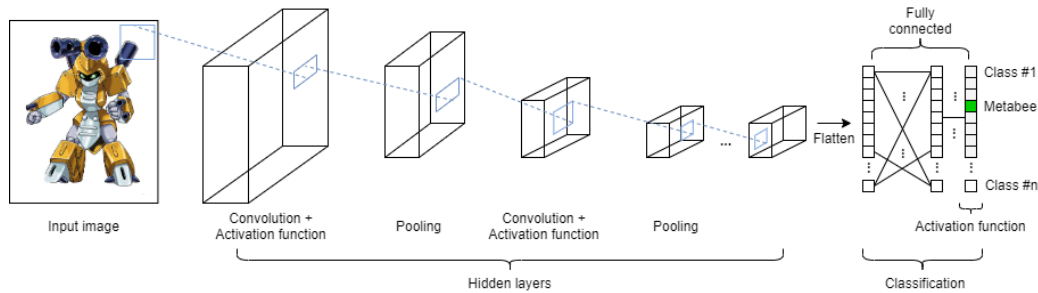


Figure 2.7: Convolutional Neural Network example.

In the following, we present four CNN architectures: VGG-16, Inception-v3, ResNet50, and ResNet101. These architectures are commonly adopted by the Computer Vision community due to the excellent results obtained in different competitions, such as ImageNet. In this work context, we use those architectures for the verification step implementation of a multi-stage approach.

**VGG-16** Simonyan and Zisserman (2014) proposed the VGG-16 model that won the localization task in ImageNet competition in 2014, reaching 92.7% of accuracy. The VGG-16 makes an improvement based on the settings of the AlexNet (Krizhevsky et al., 2012) by replacing large kernel-sized filters ( $5 \times 5$  and  $11 \times 11$ ) with multiple  $3 \times 3$  filters, one after another. The use of smaller kernels brings greater depth to the network and an increase in its non-linearity, which ends up being a positive aspect for Deep Neural Networks. Regarding the number of parameters, VGG-16 employed about  $3\times$  more parameters when

compared to AlexNet (which has 60 million parameters) and that brings a high cost: evaluating the network requires a lot of computation.

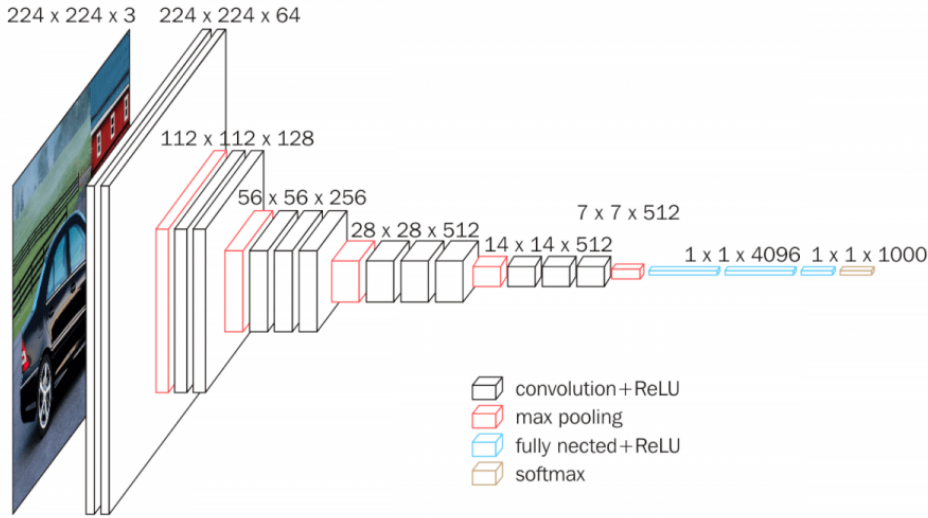


Figure 2.8: VGG-16 architecture (Neurohive, 2018).

Figure 2.8 illustrates the VGG-16 architecture. The input receives a fixed size  $224 \times 224$  RGB image. The image is passed through a combination of stacked convolutional layers followed by the ReLU (Equation 2-1) activation function. Downsamples are done using max-pooling layers. After the last stack of convolutional layers, there are three fully connected layers. The final layer uses the Softmax function (Equation 2-2), which in this case performs a 1000-way ILSVRC classification. The Softmax function returns probabilities of each class, with the target class having the highest probability.

$$R(x) = \max(0, x) \quad (2-1)$$

$$\sigma(x_i) = \frac{e^{x_i}}{\sum_{j=1}^K e^{x_j}} \quad (2-2)$$

**Inception-v3** Szegedy et al. (2016) proposed the Inception-v3 model, the third version of Google deep learning architecture series. The first Inception convolutional architecture was introduced as GoogLeNet (Szegedy et al., 2015). The Inception architecture was refined later with the Batch Normalization method (Ioffe and Szegedy, 2015), which became the second version of the architecture, namely Inception-v2. The Batch Normalization method seeks to coordinate the update of multiple model layers by standardizing the activations of each input variable per mini-batch. This method has the effect of stabilizing the learning process, reducing the number of iterations required to train deep neural networks.

Inception-v3 architecture implemented methods for factorization of convolutional kernels. Factorizing convolutions aim to reduce the number of connections and parameters of the network without decreasing the efficiency. Figure 2.9 illustrates the process of replacement of a  $5 \times 5$  convolution for two  $3 \times 3$  convolutions. In this case, the number of parameters is reduced by 28% ( $5 \times 5 = 25$  against  $3 \times 3 + 3 \times 3 = 18$ ). A similar technique was already mentioned in VGG networks (Simonyan and Zisserman, 2014).

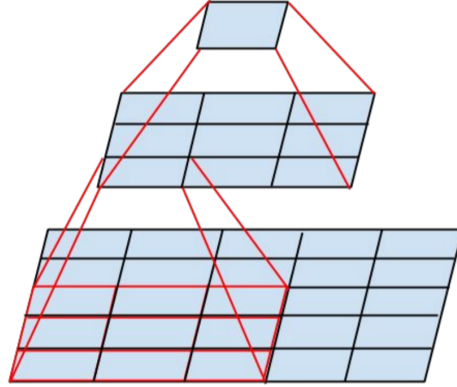


Figure 2.9: Two  $3 \times 3$  convolutions replacing one  $5 \times 5$  convolution (Szegedy et al., 2016).

Inception-v3 also implements another factorization method that does a factorization into asymmetric convolutions (Figure 2.10). In this case, one  $3 \times 3$  convolution is replaced by a  $3 \times 1$  convolution followed by one  $1 \times 3$  convolution. This causes a reduction in the parameter numbers of up to 33% ( $3 \times 3 = 9$  against  $3 \times 1 + 1 \times 3 = 6$ ).

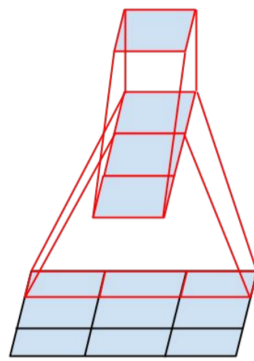


Figure 2.10: One  $3 \times 1$  convolution followed by one  $1 \times 3$  convolution replacing one  $3 \times 3$  convolution (Szegedy et al., 2016).

Figure 2.11 illustrates the Inception-v3 architecture. The model is made up of symmetric and asymmetric building blocks. The input receives a fixed size  $299 \times 299$  RGB image. Each convolutional layer is followed by Batch Normalization and ReLU activation function. Downsamples are done using max

and average pooling layers. The architecture also uses a Dropout layer (Srivastava et al., 2014) to mitigate overfitting. The fully connected layer has a shape of  $8 \times 8 \times 2048$ . Which is turned into 1001 output classes after the application of the Softmax function in the final layer.

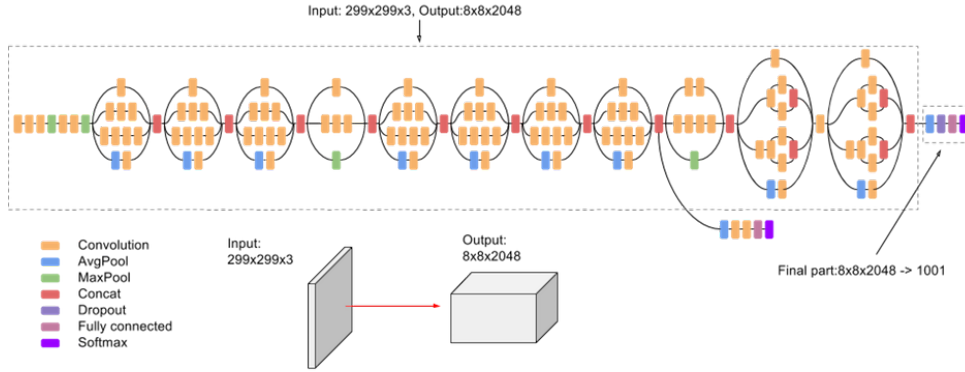


Figure 2.11: Inception-v3 architecture (Szegedy et al., 2016).

**ResNet** He et al. (2016a,b) proposed the ResNet model, which was designed to enable the increase of more convolutional layers. Since the conception of AlexNet, another CNN architectures were designed to go deeper and deeper (like VGG-16 and Inception architectures). However, increasing the number of layers in a network is not enough to achieve better performance; as the gradient is back-propagated to earlier layers, repeated computations can take the gradient infinitely small. Thus, as the network goes deeper, its performance can get saturated or even starts degrading quickly. This problem is well known as the *vanishing gradient* problem (Hochreiter, 1998).

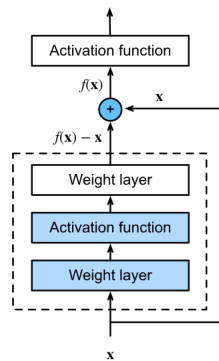


Figure 2.12: A building block for residual learning (Zhang et al., 2020).

The core idea of ResNet is introducing residual blocks which applies skip connections, also called identity shortcut connection (Figure 2.12). These blocks make it possible to skip one or more layers, taking advantage of residual



learning from the previous layers, allowing the faster propagation of inputs through layers.

Figure 2.13 illustrates the ResNet architecture. The model can treat different input image shapes but should not be smaller than  $32 \times 32$ . ResNet architecture has four modules made up of residual blocks, where each block has the same number of output channels. Similar to the Inception model, ResNet also uses Batch Normalization after each convolution. Since ResNet architecture is designed to enable hundreds or thousands of convolutional layers, this architecture has some size variations that directly influence time and resources available for computation. In this work, we make use of two alternatives, ResNet50 and ResNet101.

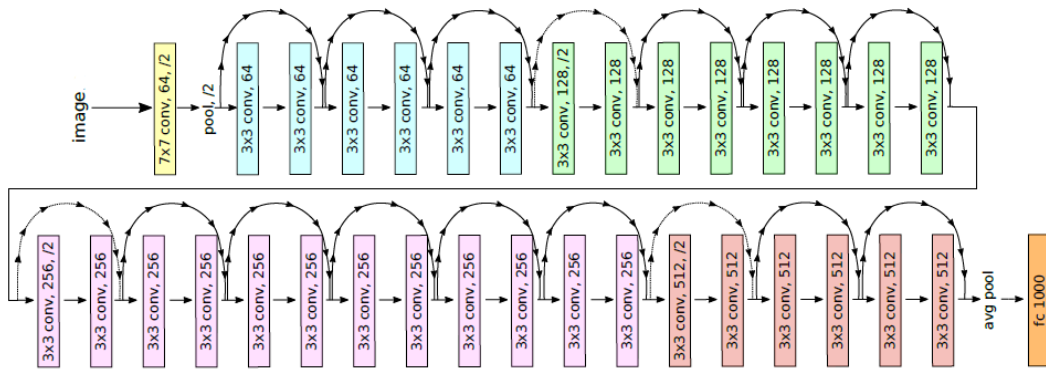


Figure 2.13: ResNet architecture (He et al., 2016a).

## 2.4.2

### Deep Neural Networks for Object Detection

Object detection is one of the fundamental Computer Vision problems. This task can provide valuable information for semantic understanding of images and video streams that go beyond the image classification task, precisely estimating the concepts and locations of objects in an image (Zhao et al., 2019). Object detection models aim to determine objects locations in a given image (object localization) and which label each object belongs to (object classification), using bounding boxes to show the confidences of objects existence. The object detection methods can mainly be categorized into two types:

- **Region proposal-based:** this method follows a traditional object detection pipeline, a two-step process, where the region proposals are generated at first, then each region is classified into different classes.
- **Regression or classification based:** this method regards objection detection as a regression or classification problem. In this case, one

framework is unified to achieve categories and locations of each object directly.

In the following, we present two object detection models: R-CNN and YOLO-v4. The R-CNN model implements the region proposal-based method, and YOLO-v4 adopts the regression-based method. In this work, we use the YOLO-v4 model as a basis for implementing two proposed approaches.

**R-CNN** Girshick et al. (2014) proposed The R-CNN model, which improved significantly the quality of candidate bounding boxes using a Convolutional Neural Network to extract high-level features. The R-CNN model obtained a mean average precision (mAP) of 53.3%, which was an improvement of more than 30% when compared with the previous best result (DPM HSC proposed by Ren and Ramanan (2013)) on PASCAL VOC objection detection dataset (Everingham et al., 2010).

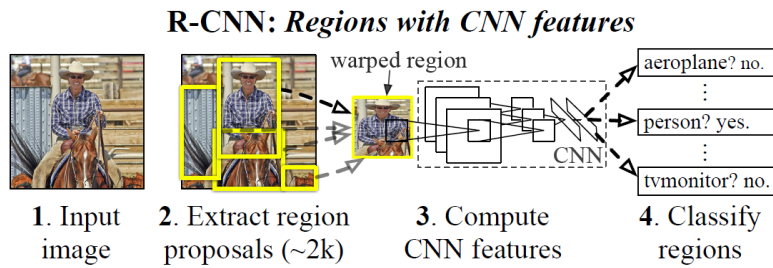


Figure 2.14: R-CNN flowchart (Girshick et al., 2014).

Figure 2.14 illustrates the flowchart of R-CNN. The R-CNN model consists of three stages:

1. **Region proposals generation:** the R-CNN adopts a selective search method (Uijlings et al., 2013). This method relies on a simple bottom-up grouping strategy that generates about two thousand region proposals for an image, providing more accurate candidate boxes of arbitrary sizes that reduces considerably the searching space;
2. **CNN based feature extraction:** in this stage, each generated region proposal is warped (or cropped) into a fixed resolution. Then a CNN is used to extract a high dimensional feature as the final representation;
3. **Region classification:** In this last stage, the different region proposals are scored positively or negatively (which indicates the background) using a set of pre-trained linear SVMs for multi-classification. Then, all positive regions are adjusted using bounding box regressions and filtered

with non-maximum suppression (Neubeck and Van Gool, 2006), producing the final bounding boxes.

**YOLO-v4** Redmon et al. (2016) proposed You only look once (YOLO), a one-stage object detection model that has continuously improved and is now in its fourth version, namely YOLO-v4 (Bochkovskiy et al., 2020). YOLO-v4 achieved state-of-the-art results (65.7%  $AP_{50}$ ) at a real-time speed ( $\sim 65$  FPS) on the COCO dataset (Lin et al., 2014). The YOLO-v4 architecture is mainly composed of three parts:

- **Backbone:** a Convolutional Neural Network, usually trained on ImageNet dataset that is used for feature extraction. YOLO-v4 uses the CSPDarknet53 model (Wang et al., 2020). The CSPDarknet53 contains 29 convolutional layers (using  $3 \times 3$  filters) and counts with 27.6 millions of parameters;
- **Neck:** a component composed of several bottom-up paths and several top-down paths that seeks to collect and prepare information (feature maps) from different layers of the backbone to the detector. For this component, YOLO-v4 implements Spatial Pyramid Pooling (He et al., 2015) and Path Aggregation Network (Liu et al., 2018);
- **Head (detector):** YOLO-v4 implements the same detector as the YOLO-v3 (Redmon and Farhadi, 2018), called dense prediction. This component is responsible for the final prediction, which is composed of a vector containing the coordinates of the predicted bounding box, the confidence score of the prediction, and the label.

The YOLO-v4 performs detections in three scales (Figure 2.15), for this, initially, it does downsamplings of size 32, 16, and 8 in the input dimensions. Thus, assuming that an input RGB image with  $416 \times 416$  size, will result in three feature maps: one with  $13 \times 13$  cells (FM-1), another with  $26 \times 26$  cells (FM-2) and another with  $52 \times 52$  cells (FM-3). Each of these is associated with three anchor boxes. Where the three largest anchor boxes are assigned to FM-1, the three smallest anchor boxes are assigned to FM-3, and the remaining three (medium-sized) anchor boxes to FM-2. These associations allow objects of larger scales to be detected by FM-1, medium scales by FM-2 and small scales by FM-3. To perform the prediction detection kernels are applied to each of the cells of these feature maps.

A detection kernel has the format  $1 \times 1 \times (B * (5 + C))$ , where  $B = 3$ , this value corresponds to the number of bounding boxes that a cell in the

feature map predicts. The value of  $C$  corresponds to the number of classes to be detected. Finally, we have four more coordinates  $(t_x, t_y, t_w, t_h)$  of the bounding boxes location next to the object's confidence score ( $p_o$ ).

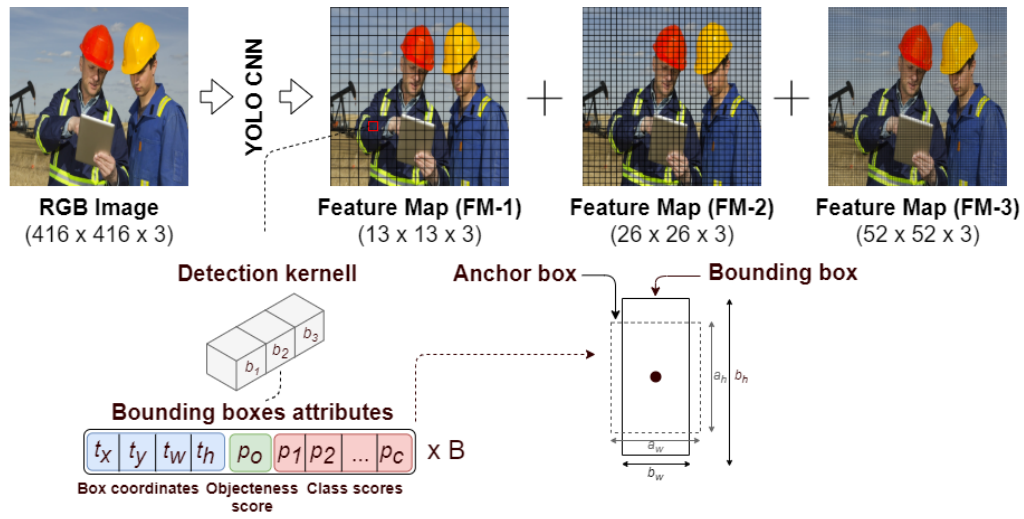


Figure 2.15: YOLO-v4 detection step.

## 3

## Related Works

In this chapter, it will be presented some works proposed in the literature to address the personal protective equipment (PPE) detection problem. The methods used for automated monitoring of PPE compliance can be categorized into two types: sensor-based and vision-based. The sensor-based methods usually install a sensor in each equipment (commonly just hard hat), process, and analyze its emitted signals continuously to produce PPE compliance evidences. Although, this approach requires the acquisition, installation, and maintenance of several sensors which can demand a significant investment for a company. Another factor that can be problematic in this kind of approach is that some types of sensors (e.g. pressure sensors) need to be installed inside the equipment and are in direct contact with the skin of a worker, causing discomfort after long hours of use, especially in a hot and humid construction environment (Zheng et al., 2019). In contrast, the vision-based methods use cameras installed in the work environment to process images using Computer Vision techniques to provide a richer comprehension of the scene, enabling automated PPE compliance (Seo et al., 2015). This type of approach can be a convenient solution, thanks to its practical potential and cost-effective implementation.

### 3.1

#### Sensor-Based

Zhang et al. (2019) creates a smart hard hat system using an Internet of Things (IoT)-based architecture. The proposed architecture includes: the installation of an infrared beam detector, a thermal infrared sensor, and radio-frequency identification (RFID) in each worker hard hat; a smartphone application for workers that communicates with a cloud server, sending hardhat uses information and receiving personalized warnings; and a web application for data visualization and alarms for managers.

Figure 3.1 illustrates the proposed IoT-based architecture. The devices installed in the hard hat communicates with a smartphone application via Bluetooth, sending evidence (worker identifier, time, and location) of the inappropriate use of the equipment by a worker. Then, the smartphone

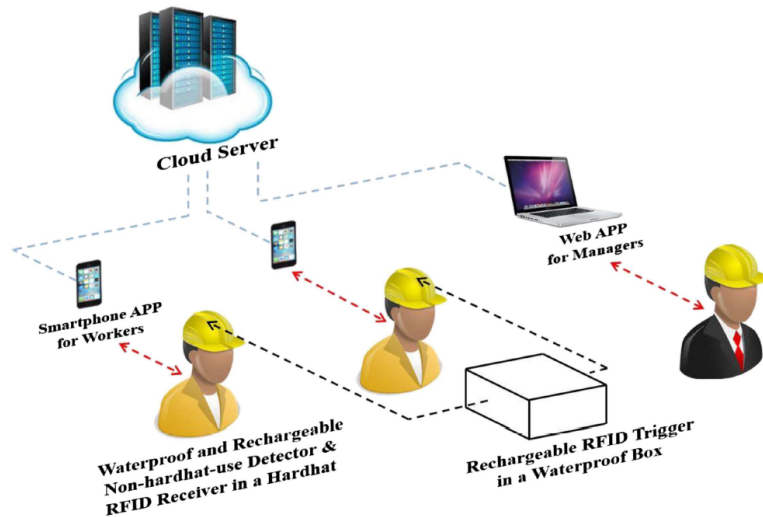


Figure 3.1: IoT-based architecture of the smart hard-hat system (Zheng et al., 2019).

communicates with the cloud via a 4G wireless network. Since the cloud service records all evidence, the web application retrieves this information and alerts a manager when a deviation occurs. This approach shows the potential to promote safety in workplace environments. But it is common to present some limitations, like communication bottlenecks, causing a delay between sensors and web or smartphone applications. Also, there is the possibility of internet unavailability in some areas. In terms of evidence production, there is no visual record, such as an image that points out the inappropriate equipment use, which not allows a complete assessment of the deviation by the manager.

### 3.2 Vision-Based

Li et al. (2017) proposed a vision-based framework for hard hat detection that consists in three stages: background modeling, pedestrian classification, and hard hat detection. The application is used in a power substation scenario and receives images extracted from a surveillance camera. This camera has a fixed install position to ensure that the image's background undergoes minimal modifications along a sequence. The first stage of the proposed solution consists of modeling the background. Hence, the ViBe algorithm (Barnich and Van Droogenbroeck, 2010) is applied to segment moving objects from a video sequence, seeking to isolate static objects, leaving only the worker as an object of interest in the scene. Then, each moving foreground object is classified by a Support Vector Machine (SVM) into two classes: pedestrians and non-pedestrians. The SVM input is a feature description of a pedestrian calculated using the Histogram of Oriented Gradients (HOG) (Dalal and Triggs, 2005).

Finally, to determine if a worker is wearing a hard hat, a simple color feature recognition is applied in the top region (one-fifth of the total height) of each classified pedestrian. The experimental results showed that the proposed framework obtained an accuracy of 80.19%, running at up to 7 frames per second. What is a satisfactory outcome for the application context. However, the implemented methods show some limitations. For example, different light intensities have affected the classifier's performance, leading to negative results. Also, the choice of a fixed region of interest assumes that workers will always be standing, and the method can fail if the worker performs a different movement, like bending down.

### 3.3

#### Vision-Based Methods Using Deep Learning

Fang et al. (2018) proposed one of the first works that use deep neural network architectures for dealing with the PPE detection problem. The proposed method uses a Faster R-CNN model (Ren et al., 2015) for non-hard hat detection in far-field surveillance video frames. Compared to other methods, an object detection model empowered with Deep Learning, like Faster R-CNN, has some advantages. One of the main benefits is the capability of learning features automatically. In the context of PPE detection, it is possible to use those benefits to automatically establishing various workers' posture models (like standing, bending). Also, the model offers robust worker detection despite weather and illumination, which was also a problem in some previous works.

To train the learning model to automatically detect the non-hard hat use by a worker, the authors built a dataset with more than one hundred thousand images from security cameras installed in a construction scenario. The dataset covers a variety of visual conditions that may occur on construction sites. Including visual range, weather, illumination, individual posture, and occlusions. In these images, workers who are not wearing a hard hat are labeled with a bounding box. Thus, during the inference stage, the R-CNN model will provide the location (bounding box) and a confidence value for each detected object, which in this case, is the probability of a worker being a non-hard hat use. The experimental results indicate that the proposed method could successfully detect non-hard hat use by workers with a precision and recall rate of 95.7% and 94.9%, respectively. Regarding the processing time, Faster-RCNN has a frame rate of 5 FPS. In this way, the proposed method offers an opportunity to contribute to real-time site monitoring, improving the workers' safety on construction sites.

Bo et al. (2019) use the YOLO-v3 model (Redmon and Farhadi, 2018)

for hard hat detection in an electric power construction scenario. For this, the proposed model was re-trained using the fine-tuning technique from the pre-computed weights of the network in the COCO dataset to detect two classes: hard hat and head. The dataset built in this work includes 1555 images, annotated with 2253 and 2346 instances of the hard hat and head classes, respectively. For the model evaluation stage, the authors adopted three metrics: precision, false alarm (Equation 3-1), and missing alarm (Equation 3-2). Table 3.1 presents YOLO-v3 fine-tuned model results. The experimental results indicate that the proposed method obtained promising results. However, the method has no verification stage, which does not guarantee that a worker is making appropriate use of his hard hat. For example, if a worker is holding his helmet in his hand, the model would probably detect that hard hat. But not its improper use.

$$\text{False alarm} = \frac{FP}{TP + FP} \quad (3-1)$$

$$\text{Missing alarm} = \frac{FN}{TP + FN} \quad (3-2)$$

	Precision	False alarm	Missing alarm
<b>Hard hat</b>	96.58%	0.89%	3.42%
<b>Head</b>	97.12%	0.69%	2.88%

Table 3.1: YOLO-v3 fine-tuned model results (Bo et al., 2019).

The works presented above aims to construct a model for hardhat wearing detection. However, in some cases, we are interested in detect multiple PPE, such as hard hats, gloves, and worker vests. In this sense, Zheng et al. (2019) and Nath et al. (2020) proposed methods to handle a higher amount of PPE.

Zheng et al. (2019) implement a simplified version of YOLO-v2 architecture (Redmon and Farhadi, 2017) aiming at a low computational complexity network for PPE detection in real-time. The proposed architecture consists only of nine convolutional layers and six pooling layers (maximum). The authors also embed a dense connection block in the convolutional layer with the lowest resolution ( $16 \times 16$ ), which allows the following layer to receive multi-layer convolutions, aiming to minimize information losses and strengthen feature propagation.

To address the multiple PPE detection, which in this work are: hard hat, safety glove, and safety vest. The authors build three different models (using the same architecture), where each model is responsible for one type of equipment. In this way, each model is trained separately to detect the presence



or absence of a single piece of personal protective equipment. During the inference stage, the three models receive the same input image and perform the detections. Table 3.2 presents the detection results for the test dataset, which consists of 526 images of three categories: safety gloves samples (211 images), hard hat samples (182 images), and safety vest samples (133 images). Despite the qualities of the results obtained for the precision and recall metrics, the proposed method has some limitations. The main disadvantage comes from the three different models use (one for each PPE) that share the same architecture. In this way, there is no sharing of information between the different models about the features learned. Thus, the Deep Neural Network capacity for multi-classification problems is not well used, bringing unnecessary computational complexity to the task.

	<b>Precision</b>	<b>Recall</b>
<b>Hard hat</b>	98.90%	93.67%
<b>Safety glove</b>	94.79%	91.22%
<b>Safety vest</b>	94.74%	91.10%

Table 3.2: Detection results for each PPE type (Zheng et al., 2019).

Nath et al. (2020) proposed three approaches to detect multiple PPE (hard hat and safety vest) in a construction scenario. All approaches use a different implementation of YOLO-v3 (Redmon and Farhadi, 2018) model as the basis of the method. Approach-I is a multi-stage method that performs the detection and verification phases using two models. First, given an input image, a YOLO-v3 model detects three objects: worker, hard hat, and safety vest. Next, a simple Machine Learning classifier receives the object detection model outputs (bounding box coordinates) as inputs and determines what equipment is being used properly by a worker. In approach-II, the detection and verification phases are performed by a single model, so it is a one-stage method. In this case, a YOLO-v3 localizes workers in the input image; and direct classifies each worker into four classes: worker without PPE (W), a worker using a hard hat (WH), a worker using a safety vest (WV), and a worker using a hard hat and a safety vest (WHV). Finally, the approach-III is a multi-stage method. First, a YOLO-v3 detects only workers in an input image. Then, each worker detection is cropped and given as input to a CNN (or a Bayesian framework of CNNs). This CNN is responsible for performing the verification stage, which in this case, given the cropped image, the model classifies it in one of the four classes previously mentioned (W, WH, WV, WHV). The CNN architectures investigated in this approach implementation are VGG16, ResNet50, and Xception (Chollet, 2017). The dataset used to train

and evaluate the models contains 1472 annotated images with approximately 4700 instances of workers wearing different combinations of PPE components. The dataset is named Pictor-v3, and it is composed of two sources: Crowdsourcing (Yuen et al., 2011) and web-mined images. Figure 3.2 presents the comparison in terms of Average Precision (AP) and Mean Average Precision (mAP) of the overall performance obtained for each approach.

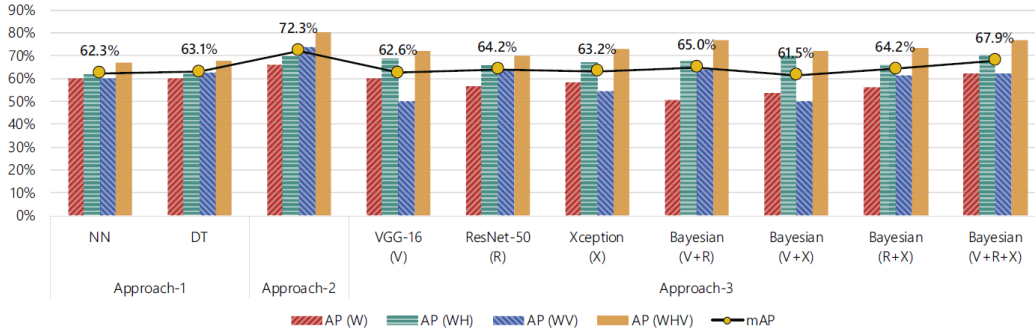


Figure 3.2: Comparison of the overall performance of each approach (Nath et al., 2020).

Deep neural networks have been revealed to be quite useful for solving object identification tasks (Nath et al., 2020). Despite the meaningful results described in the literature, in the industrial domain, this approach has limitations considering the time of inference and precision. In this work, we propose robust approaches that can be used as a fundamental component of a monitoring system for PPE detection that uses YOLO-v4 model as a basis for implementing the proposed approaches. We also experiment with new adaptations, in terms of models and parameters for the multi-stage approach, where we create a solution that uses ensemble classifier. As one will see in the next sections, our approaches present promising results (in terms of mAP) and still capable of act in real-time.

## 4 Methodology

This chapter presents the construction and exploration process for the two proposed approaches, including dataset generation, used to carry out model training and evaluation. In this work, we evaluate whether these approaches are ready to act in real-time and with robustness when implemented as a fundamental component of a monitoring system that seeks to detect which workers are making the appropriate use of personal protective equipment (PPE).

For this study, we choose to focus on two types of PPE, hard hat and protective clothing. These types of equipment are often used to ensure the safety of workers in the oil and gas industry and civil construction. Our approaches are based on techniques for detecting and classifying objects in images. In this way, we have four possible classes to perform the classification (Figure 4.1), which are: worker with no PPE (W), worker wearing a hardhat (WH), worker wearing protective clothing (vest) (WV), and worker wearing a hard hat and protective clothing (WHV). Note that implemented approaches can be extended to any number of PPE. However, the complexity level of the verification step (PPE compliance) grows exponentially. Once there are  $2^n$  possible combinations of use for  $n$  different types of PPE.



Figure 4.1: Possible classes to perform classification.

The detection and verification component is based on models that empower Deep Learning methods capable of being executed in real-time. In Nath et al. (2020) work, which also addresses the problem of detecting multiple PPE, the authors adopt as the definition for real-time system one that can process at least five frames per second (FPS). When this is not possible,

values  $\geq 1$  FPS are considered "near real-time". These definitions emerge from previous work (Redmon et al., 2016) which also raise this concern. In this work, we follow the same definition since the context is quite similar. We expect that our implementation can act in real-time with a prompt response when a worker's life is exposed to risk.

In particular, both approaches employ YOLO-v4 architecture and carry some of the steps in a similar manner, such as preparing data and the models' training. In Approach-I, for each image, we annotate a bounding box for each worker. The class of this bounding box informs which PPE the worker is wearing. This way, we create a single model based on YOLO-v4 architecture for detection and verification phases. Approach-II uses one model for the detection stage and another model for the verification stage. First, we detect workers' locations (bounding boxes) from the input image using a model based on the YOLO-v4 architecture. For the PPE compliance verification stage, a Convolutional Neural Network receives cropped-image regions (for each worker detected) and classifies them according to possible states (W, WH, WV, WHV). The details for both approaches are clarified in Section 4.4 and Section 4.5.

## 4.1

### Dataset Generation

Supervised Machine Learning applications requires a large annotated dataset to provide the learning model a way to create and recognize patterns through the data. As we did not have access to a large image dataset available for multiple PPE detection, it was necessary to create a annotated dataset to feed the learning model.

The dataset used in this work is composed of images from the following sources: Crowd-sourced (as used by Nath et al. (2020)), GDUT-Hardhat wearing detection (GDUT-HWD) (Wu et al., 2019), Web-scrapped, and images captured by the authors. Of the datasets that already had annotation (Crowd-sourced and GDUT-HWD), the only one that deals with multiple PPE detection is Crowd-sourced, where the annotation of classes is the same as that used in this work. For GDUT-HWD, only the images were used, since the dataset annotations are only for the individual identification of safety helmets. To compose the Web-scrapped source, images were obtained from public databases using search engines that perform searches by keywords, for example, "workers in refinery", "workers in platforms". After the complete collection of these images, there was a visual inspection to remove images that were out of context or that had low quality. The images captured by

the authors were taken in a controlled environment that sought to reproduce the conditions of an industrial environment. Figure 4.2 shows the number of instances for each data source.

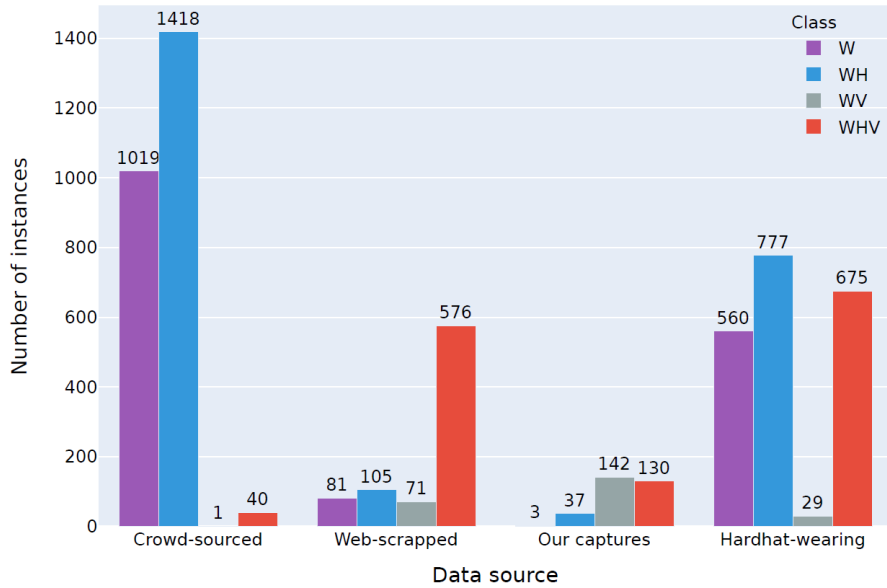


Figure 4.2: Distribution of instances per class for each data source.

The annotation procedure to mark bounding boxes of objects in images for training the models was carried out in the YOLO mark tool<sup>1</sup>, an open-source library for image annotation. Aiming to minimize the annotation bias, the whole dataset was divided into batches of equal sizes and assigned to one of the annotators (two participated). After the completion of a batch of images, a second annotator checked all annotations. If there was any divergence regarding an annotation (concerning an annotated class or a bounding box region), this instance was separated and discussed among the annotators until they reached a consensus for the real ground truth definition.

## 4.2

### Data Preprocessing

In the dataset preparation, we divided the images into three subsets: training (70%), validation (20%), and testing (10%). Since the dataset used is composed of four different sources. The images of each source were proportionally distributed (randomly) among the subsets. i.e., training, testing, and validation sets have the same percentage of each data source. This kind of division ensures that the subsets have similar distributions, making them more homogeneous. As an image can have several instances and it is not possible to distribute the instances of a single image to different subsets. The proportion

<sup>1</sup>[github.com/AlexeyAB/Yolo\\_mark](https://github.com/AlexeyAB/Yolo_mark)

of the number of instances may end up being affected. Thus, to verify whether the instances proportion for each subset is similar, we analyzed the distribution of the number of instances, as shown in Figure 4.3. According to the figure, the proportion of instances remained close to expected for all subsets. The same can be observed for the distribution of the number of classes per subset. Moreover, it is possible to observe that the dataset has few examples of the WV (worker with a safety vest) class compared with other classes. This unbalance may hinder the learning model from generalizing that class.

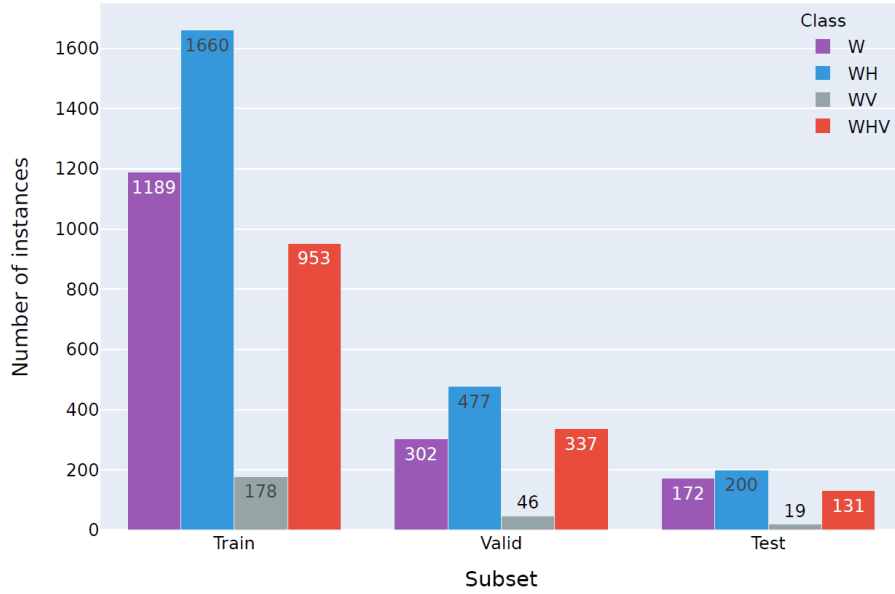


Figure 4.3: Number of instances per class for each subset.

To improve the ability to detect objects accurately, YOLO-v4 uses nine anchor boxes, which must be defined for use during the training and inference phases. With the anchor boxes defined, the model can specialize in objects of certain sizes and with particular aspect ratios (height  $\times$  width).

In practice, during the training phase, each cell of the feature maps of the network's output layers has an associated anchor box. Thus, the model learns how to shift and scale a predicted bounding box based on anchor box references so that the output coordinates fit an object of interest. The K-Means clustering algorithm was used with training set bounding boxes as input to define the anchor boxes. Since YOLO-v4 architecture uses nine anchor boxes as a reference, the number of centroids in K-Means is equal to nine. In this way, we have one correspondent centroid that defines the characteristic patterns (height and width) of each anchor box (Figure 4.4).

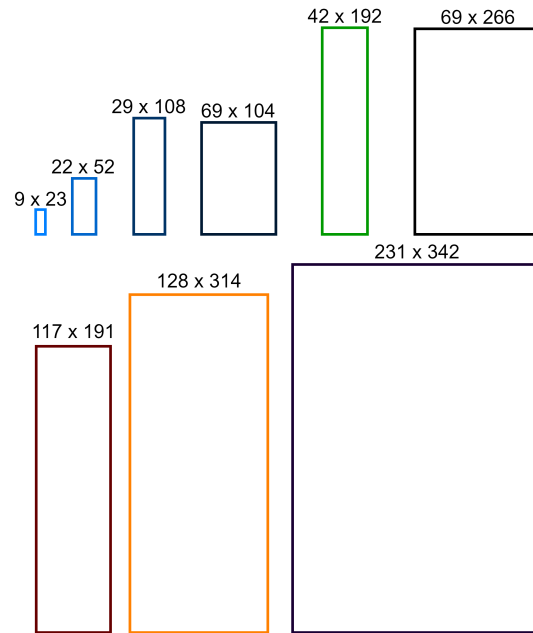


Figure 4.4: Corresponding anchor boxes for the nine centroids.

### 4.3

#### Data Augmentation

To avoid the problem of lack of data, which can lead to the overfitting of Machine Learning models during the training stage, we use different data augmentation techniques. Data augmentation techniques seek to apply transformations to the input images to increase the diversity of examples in the training dataset. Some conventional data augmentation techniques use simple transformations, such as rotation, translation, and scale. The application of data augmentation requires a lot of care. Some transformations may end up not preserving the labels of the input data, which would end up causing an opposite effect, hindering the training stage of a model. Figure 4.5 illustrates examples of applications of this technique in a fictitious input image.

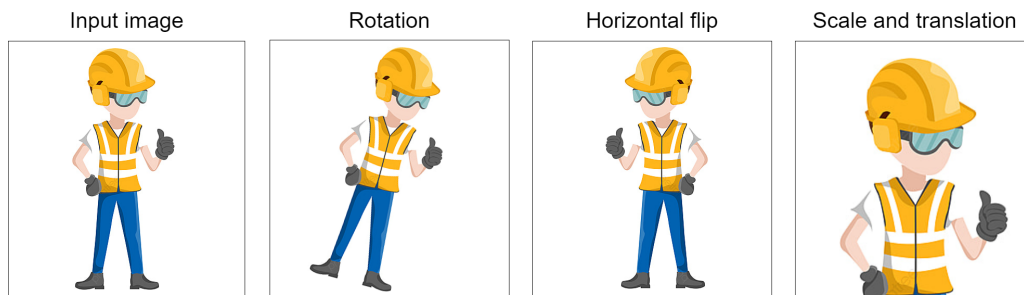


Figure 4.5: Data augmentation transformations examples.

In this work, we implemented two data augmentation techniques during the training stage of object detection models. The first technique consists of

an HSV color space change, where the saturation and exposure values are randomly adjusted by up to a factor of 1.5. The second technique is the Mosaic method (Bochkovskiy et al., 2020). This method brings a new data augmentation technique that mixes four training images into a single image (Figure 4.6). Hence, four different contexts are mixed and could allow the model to learn to detect objects outside of their usual context.



Figure 4.6: Mosaic data augmentation (Bochkovskiy et al., 2020).

Data augmentation techniques have also been used during CNNs training stage. In this case, the following transformations are adopted:

- Random zoom in the range  $[0.80, 1.20]$ ;
- Shear range with a factor of up to 20 degrees (counter-clockwise direction);
- Horizontal flip in 50% of images.

#### 4.4

##### Approach-I

The first approach uses a single YOLO-v4 model, which we will call YOLO-v4-AP1, to perform the detection and verification steps in a one-stage implementation. The YOLO-v4-AP1 model detects workers' location and directly classify each worker according to the PPE attire (W, WH, WV, WHV). Figure 4.7 illustrates how Approach-I is performed. One of the main advantages of this approach is that it takes advantage of the YOLO-v4 architecture's capabilities to make predictions with objects' locations and their respective classes using a single network. What makes it a simple and effective implementation.



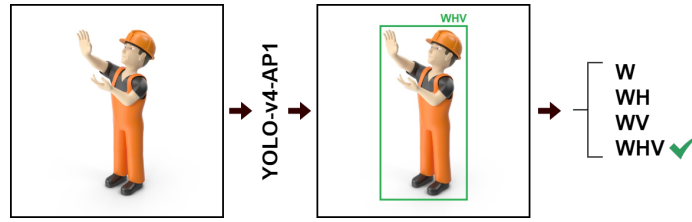


Figure 4.7: Approach-I.

During the inference phase, only predictions with a confidence score above 50.0% are admitted as final predictions. It is worth mentioning that the model can produce duplicate detections, presenting different classes for the same worker. But following the problem definition, a worker can belong to only one class. For example, if there is a worker detection with the WHV class, this detection supersedes any W, WH, or WV detections for the same worker. To avoid duplicate detections, the model uses non-maximum suppression. Usually, non-maximal suppression adds 2-3% in mAP (Redmon et al., 2016).

## 4.5

### Approach-II

For this approach, we use a multi-stage method to perform the identification and verification stages. Initially, a YOLO-v4 model is used (which we will call YOLO-v4-AP2) only to locate the worker (detection step). Thus, there is only the worker class (W). Then, a CNN is used to carry out the verification step. This CNN receives a cropped image for each worker detected by the YOLO-v4-AP2 model in the input image and classifies it according to one of the four possible classes (W, WH, WV, WHV). Figure 4.8 illustrates how this multi-stage approach is performed.

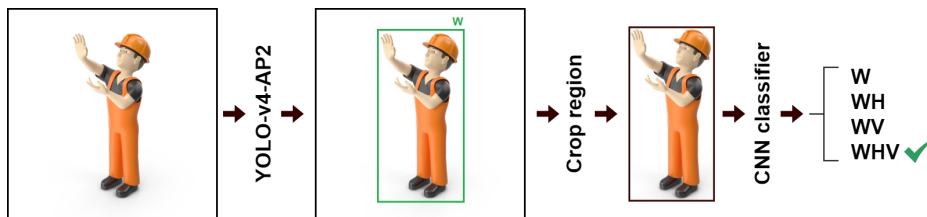


Figure 4.8: Approach-II.

The CNN architectures investigated in the implementation of this approach are VGG-16, Inception-v3, ResNet50, and ResNet101. The following modifications have been made (in order) to all architectures to address the verification stage:

- Change in the input layer to receive  $150 \times 150$  size images;

- Addition of a dropout layer with a probability of drop  $p = 0.3$ ;
- Addition of a fully connected layer with 256 nodes and regularization  $L1$  and  $L2$  with penalty value of 0.001 followed by the ReLU activation function;
- Addition of an output layer (dense) with four nodes (one for each output class) followed by the Softmax activation function.

We also improve Approach-II by using an ensemble of classifiers with the majority vote method (Figure 4.9). That is, each classifier assigns his vote to the class with the highest probability. The final prediction is given to the class that received the most votes. Thus, when performing the verification stage for PPE compliance, the learning of an entire set of classifiers (VGG16, Inception-v3, ResNet50, and ResNet101) is used. We believe that combining multiple classifiers predictions may reduce the variance and the classification bias, making the classification less dependent and the class distinction criteria more expressive. As an even number of classifiers have been adopted, a tie can happen. In that case, as implementation detail issues, the first class that obtained the most votes in order of labels indices (W, WH, WV, WHV) is chosen. However, we believe that there are better strategies to resolve the cases of ties, for example, to exclude the vote of the classifier that had the lowest confidence score.

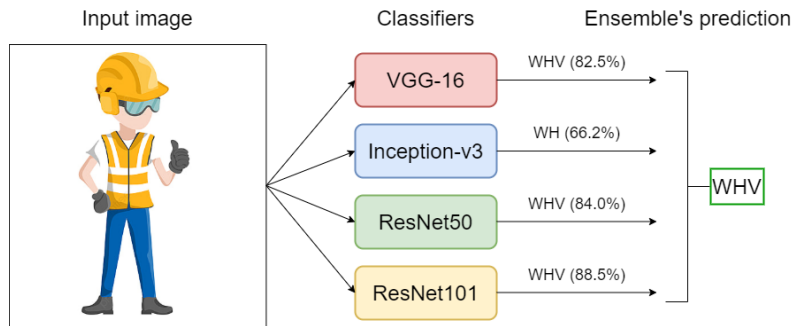


Figure 4.9: Ensemble of classifiers (majority vote).

## 4.6

### YOLO-v4 Model Architectures

For the two proposed approaches, the architecture of YOLO-v4 models is modified based on the number of considered classes by each approach. These modifications consist of changing the dimensions of the network's output layers. In Approach-I, the YOLO-v4-AP1 model directly classifies each worker according to the PPE attire. Since we focus on two types of equipment (hardhat and safety vest), there are four possible classes: W, WH, WV, WHV. As already

seen in Subsection 2.4.2, the dimensions of the three output layers of YOLO-v4 architecture are  $13 \times 13 \times (3 * (5 + C))$ ,  $26 \times 26 \times (3 * (5 + C))$ , and  $52 \times 52 \times (3 * (5 + C))$ , respectively. Where the  $C$  value corresponds to the number of classes. Thus, for the model implemented in Approach-I, we have three output layers with the following dimensions:  $13 \times 13 \times 27$ ,  $26 \times 26 \times 27$ , and  $52 \times 52 \times 27$ , respectively. In Approach-II, the YOLO-v4-AP2 model detects only workers (only W class). Therefore, for the model implemented in Approach-II, we have three output layers with the following dimensions:  $13 \times 13 \times 18$ ,  $26 \times 26 \times 18$ , and  $52 \times 52 \times 18$ , respectively.

## 4.7

### Model Training

**Object detection models** To carry out the training process, we adopted the transfer learning method. For this, it was used a pre-trained YOLO-v4 model that is state-of-the-art in the COCO dataset (Bochkovskiy et al., 2020). In that way, we can take advantage of the model's ability to detect up to 80 classes (person, bicycle, car, motorcycle, plane, etc.) from the COCO data set and apply the knowledge in our domain. Since this model has already been trained from a more significant number of images, it can generalize its learning ability to distinguish resources for our task, which has a much smaller number of images. It may seem that these two tasks have no evident intersection (in addition to the fact that a worker is also a person). Although in problems that address classification or detection tasks, some low-level characteristics, for example, edges, shapes, contours, and intensities, can be shared between tasks, thus allowing the spread of knowledge between them.

The training process of both models (YOLO-v4-AP1 and YOLO-v4-AP2) took place in a very similar way. As the models implemented for the two proposed approaches share almost the same architecture, all layers have their weights initialized with pre-computed weights from the YOLO model trained in the COCO dataset. The only exception is in the three output layers of each model, which had their dimensions modified. These layers, in turn, have their weights initialized at random. Regarding the hyperparameters definitions, both models followed the same outlines adopted in Bochkovskiy et al. (2020). That is, 30 epochs using the stochastic gradient descent (SGD) optimizer (Qian, 1999) to accelerate the learning convergence process, with a learning rate = 0.0013, momentum = 0.949, and decay = 0.0005.

**Classification models** Approach-II involves additional training of the classifiers models (VGG-16, Inception-v3, ResNet50, and ResNet101). In this case, the first change consists of the training data, where this data is constructed from each instance present in the training set of the object detection models. During the training process, we also use the transfer learning method. The difference is that the model weights are initialized from the pre-trained weights for the Imagenet dataset. The models were trained during 30 epochs using the Adam optimizer with a learning rate of  $10^{-5}$  with the categorical cross-entropy loss function (Equation 4-1). Where  $t_i$  is the ground truth class,  $p_i$  is the Softmax probability for  $i^{th}$  class, and  $n$  is the number of classes.

$$L_{CE} = - \sum_{i=1}^n t_i \log(p_i) \quad (4-1)$$

## 4.8

### Mean Average Precision for Performance Evaluation

The performance of the YOLO models are evaluated using the mAP (Mean Average Precision) metric, which is used in several object detection models (for example, Faster R-CNN, R-CNN, SSD). One of the advantages of the mAP is that we can quantify how well an object detection model is performing in a data set using a single numerical representation (Turpin and Scholer, 2006).

To calculate the mAP, initially, all detections must be ordered in descending order based on the confidence level. For each prediction, we calculate the precision and recall metrics using the accumulated value of valid (true positives) and non-valid (false positives) detections for the current class. To determine whether detection is valid or not, the Intersection over Union (Figure 4.10) of the bounding box resulting from the prediction ( $B_p$ ) is calculated with the bounding box of the ground truth ( $B_{gt}$ ).

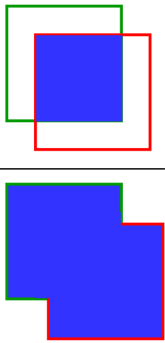
$$IoU = \frac{\text{area}(B_p \cap B_{gt})}{\text{area}(B_p \cup B_{gt})} = \frac{\text{(area of overlap)}}{\text{(area of union)}}$$


Figure 4.10: Intersection over union (IoU).

If the value of intersection over union (IoU) is greater than a certain threshold and the predicted class is equal to the ground truth class, the detection is a true positive (TP). Otherwise, a false positive (FP). The confidence IoU threshold adopted to classify a prediction as a true positive for all experiments in this work is  $\text{IoU} \geq 0.50$ .

Each class calculates its Average Precision (AP) by applying the Equation 4-2, where  $n$  represents the total amount of detections;  $i$  denotes, in the list of ordered detections, the rank of a singular detection;  $p(i)$  is the calculated with the cumulative values for false positives and true positives from the first to the  $i$ th detection, ordered by confidence; and  $\Delta r(i)$  is the variation in recall from  $(i - 1)$ th to  $i$ th detection.

$$AP = \sum_{i=1}^n p(i) \Delta r(i) \quad (4-2)$$

## 5 Experimental Results

### 5.1 Performance of the YOLO-v4 Models

Figure 5.1 presents the performance results of the object detection models used in Approach-I (YOLO-v4-AP1) and Approach-II (YOLO-v4-AP2) for the test set. The YOLO-v4-AP2 model obtained the highest value of mAP (88.67%), which was expected since there is only one class to be detected. A factor that may have contributed considerably to the model's performance was the use of transfer learning. Thus, the model was able to use the previous knowledge obtained from the training in the COCO dataset and to identify classes that present similar features with more precision. In this case, we have a notable similarity between the person class of the COCO dataset and the worker class (W). For Approach-I, we have an mAP of 80.65%, which is a good result since this approach uses only a single model to perform the detection and verification stages for PPE compliance. It is worth mentioning that this is the final mAP value for this approach, in contrast to the result obtained for Approach-II, which will change when the verification stage is taken into account. Thus, the final mAP will be directly affected according to the performance of the classifiers. Table 5.1 shows, in addition to the mAP value, the precision, recall, and f1-score metrics obtained by each model. The f1-score is the harmonic mean of the precision and recall (Equation 5-1).

$$\text{F1-score} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} \quad (5-1)$$

	mAP	Precision	Recall	F1-score
<b>YOLO-v4-AP1</b>	80.65%	75.46%	70.11%	72.69%
<b>YOLO-v4-AP2</b>	88.87%	91.23%	83.72%	87.31%

Table 5.1: mAP, precision, recall and f1-score metrics for YOLO-v4-AP1 and YOLO-v4-AP2 models.

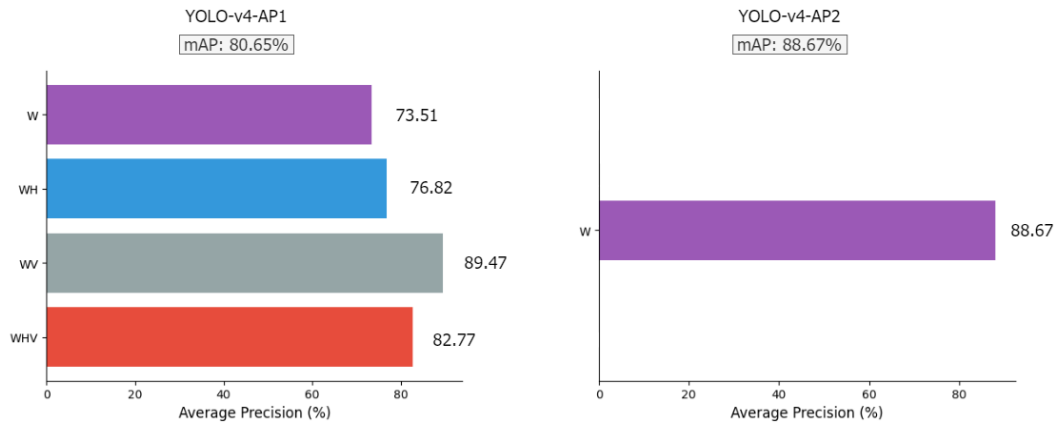


Figure 5.1: Performance of the proposed models for each approach.

## 5.2

### Performance of Classifiers

The instances (bounding boxes) of the test set were used to evaluate the performance of the classifiers that worked during the approach-II verification stage. The accuracy of VGG-16, Inception-v3, ResNet50, and ResNet101 for classifying images into W, WH, WV, and WHV classes are 82.85%, 78.35%, 84.02%, and 82.75%, respectively. Figure 5.2 presents the confusion matrix for each model. Note that the WHV class has the lowest accuracy in all classifiers. One reason for that is the confusion with the WH class, which has an average value of 20.05% for false negatives. That is, once a worker is being detected with a hard hat, it is difficult for the classifier to distinguish whether a given worker is wearing protective clothing or not. The individual accuracy for the rest of the classes (W, WH, WV) presents values higher than 80%, except for class W when evaluated in Inception, which had a value of 6.6% below the average accuracy of class W (82.83%).

VGG16					INCEPTION				
	W	WH	WV	WHV		W	WH	WV	WHV
W	83.1%	14.5%	0.6%	1.7%	W	76.2%	18.6%	0.6%	0.47%
WH	4%	85.5%	0%	10.5%	WH	5%	83.5%	0%	11.5%
WV	0%	0%	84.2%	15.8%	WV	0%	5.3%	84.2%	10.5%
WHV	2.3%	19.1%	0%	78.6%	WHV	6.1%	23.7%	0.8%	69.5%

RESNET50					RESNET101				
	W	WH	WV	WHV		W	WH	WV	WHV
W	86%	11.6%	1.2%	1.2%	W	86%	11.6%	1.2%	1.2%
WH	6.5%	83.5%	0%	10%	WH	7%	83%	0%	10%
WV	0%	5.3%	89.5%	5.3%	WV	0%	5.3%	89.5%	5.3%
WHV	3.1%	16.8%	3.1%	77.1%	WHV	5.3%	20.6%	1.5%	72.5%

Figure 5.2: Confusion matrices for VGG-16, Inception, ResNet-50 and ResNet-101 classifier models.

### 5.3

#### Performance Comparison of Approaches

In this subsection, we will evaluate the final performance of approaches I and II. Note that for Approach-II, the final mAP value strongly depends on the classifiers' performance (or ensemble of classifiers) used during the verification stage. Figure 5.3 shows that Approach-I obtained the best mAP (80.65%) value, even when compared with the different model combinations used in Approach-II (including ensemble). Although the YOLO-v4-AP2 model presented an mAP of 88.67% in the detection stage, the errors of the classifiers for the verification step end up reducing the mAP. Since some of the instances will be classified incorrectly, generating false positives. The best result for Approach-II presents an mAP value of 71.74%, given from the YOLO-v4-AP2 model plus the ensemble with the classifiers VGG16, Inception-v3, ResNet50, and ResNet101. However, when the Inception-v3 model is removed from the ensemble, the result is almost the same (71.73%). That may be because the Inception-v3 model was the model that presented the least accuracy of all classifiers (78.35%). In the scenario where only one classifier was used in the verification stage, the best result is obtained using ResNet101, which is 69.44% of mAP. Table 5.2 shows, in addition to the mAP value, the precision, recall, and f1-score metrics obtained by each model.

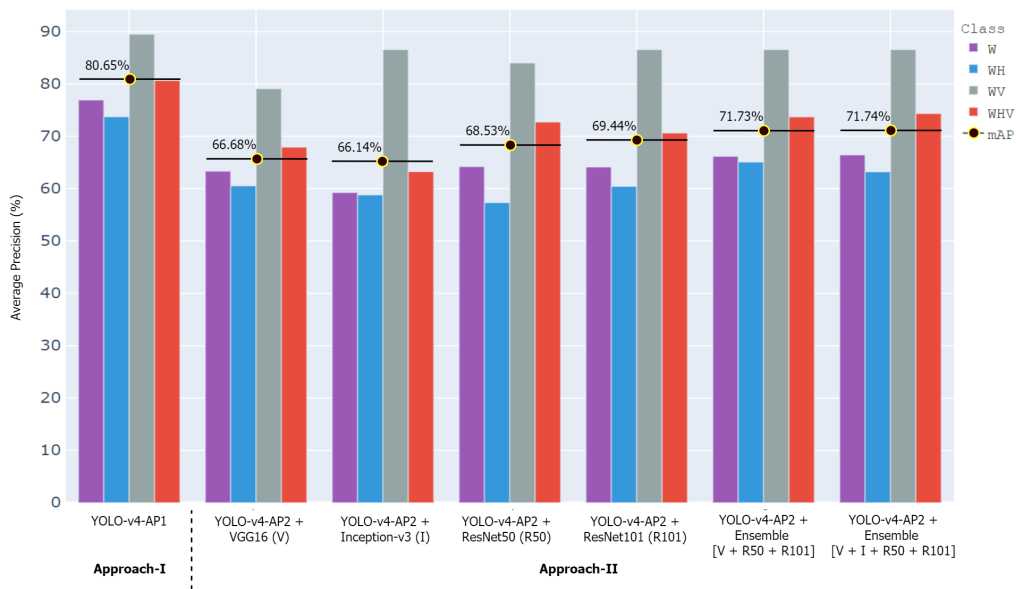


Figure 5.3: Performance comparison of approaches implementation.

We also compared the processing time spent by each approach. We run all models on the same machine, which has the following configurations: Intel Core i9-7900X, 128 GB RAM, TITAN RTX GPU with 24 GB memory, and operating system Ubuntu 19.10. Figure 5.4 presents the average processing



	mAP	Precision	Recall	F1-score
YOLO-v4-AP1	80.65%	75.46%	70.11%	72.68%
YOLO-v4-AP2 + VGG-16	66.68%	74.68%	75.16%	74.91%
YOLO-v4-AP2 + Inception-v3	66.14%	73.04%	74.24%	73.63%
YOLO-v4-AP2 + ResNet50	68.53%	73.07%	76.31%	74.65%
YOLO-v4-AP2 + ResNet101	69.44%	75.87%	76.96%	76.41%
YOLO-v4-AP2 + [VGG16, ResNet50, ResNet101]	71.73%	78.20%	78.48%	78.34%
YOLO-v4-AP2 + [All classifiers]	71.74%	78.15%	78.64%	78.39%

Table 5.2: Detailed result of each implementation for the values of mAP, precision, recall, and f-score.

time for a test image in each approach. Approach-I has the best processing time (12.55ms). For Approach-II, the YOLO-v4-AP2 model processing time is slightly less (12.38ms) than the model implemented in Approach-I. But when adding the classifiers' processing time into account, that time increases significantly, making Approach-II the slowest. With times in the interval of 53.4ms to 224.67ms. As the proposed models run at a rate of at least 5 FPS (frames per second), following the definition adopted in this work (same as Redmon et al. (2016) and Nath et al. (2020)), we can say that these approaches are capable to process videos in real-time applications and can be implemented as the base component of a monitoring system.

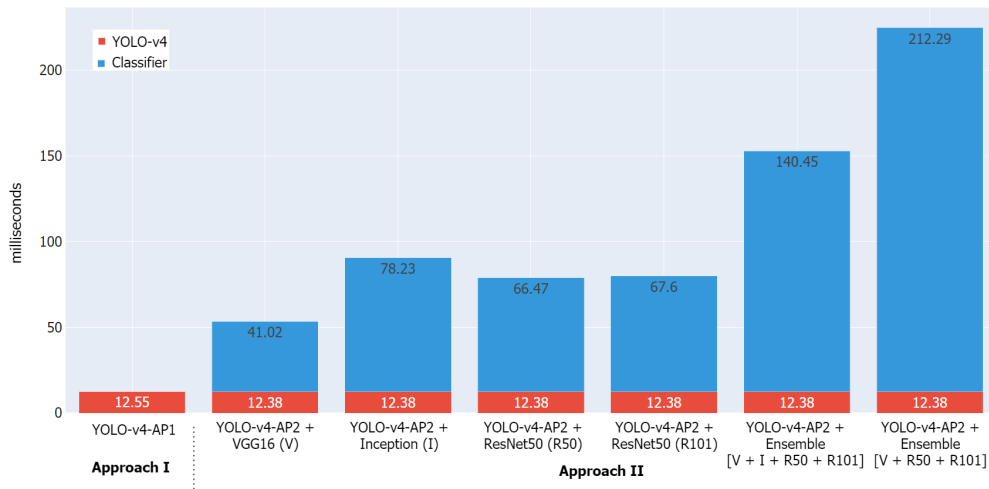


Figure 5.4: Processing time comparison for approaches implementation.

## 5.4 Benchmark of Results

Since the dataset or models used in the work of Nath et al. (2020) were not made available by the authors, we have to find other alternatives. To perform a fair comparison of the methodology adopted in this work and verify if the approaches were effective, we establish a comparison with the results obtained

Criteria	Baseline model (YOLO-v3)	One-stage approach (I) (YOLO-v4-AP1)	Multi-stage approach (II) (YOLO-v4-AP2 + Ensemble [VGG16, ResNet50, ResNet101])
mAP	58.67%	80.65%	71.73%
FPS	104	80	7

Table 5.3: Comparison of mAP and FPS values with a baseline model.

from the YOLO-v3 model. This model was trained and tested in a similar way to the YOLO-v4-AP2 model. Table 5.3 shows the comparison between models in terms of mAP and FPS. Concerning mAP, we have an increase of 21.52% and 13.96% when comparing the YOLO-v4-AP1 and YOLO-v4-AP2 (with ensemble) models with baseline (YOLO-v3), respectively. On the other hand, there is an increase in FPS. However, since the models of approaches I and II are already able to operate in real-time, this gain is not very significant.



Figure 5.5: Example of detections obtained from YOLO-v4-AP1 model. The first two images are from the Crowd-sourced dataset. The third image is from the Web-scraped dataset that is in an industrial setting.

## 6

## Conclusion

This work explores the implementation of two different approaches based on Deep Learning to perform the task of detecting the usage of PPEs by workers. We aimed to develop and evaluate approaches that are robust and capable of acting in real-time, so that they can be implemented as a fundamental component (detection and verification) of a monitoring system.

Although in this paper, we focus on two types of PPE (hard hat and safety vest), our results show that we can employ our methodology to any number of equipment (for example, goggles, gloves, and masks) changing the networks' output layers. The effect of this, is an increase in complexity due to the number of different combinations of equipment.

To carry out the models' training, we built a dataset from four different sources (Crowd-sourced, GDUT-HWD, Web-scrapped, and images captured by the authors) to supply our models with a more significant number of images from different devices, angles, lighting, and environments. In order to enhance our models' generalization capabilities.

For Approach-I, we built a single model based on YOLO-v4 implemented in one-stage; that is, the same model is responsible for identifying and verifying the use of PPE. Hence, when receiving an image as input, the model classifies each founded region that displays a worker, with one of the following classes: W, WV, WH, and WHV. In contrast, Approach-II is a multi-stage, with at least two distinct models for the identification and verification stages. Initially, a YOLO-v4 based model detects the regions in which workers are located. Then, a Convolutional Neural Network receives the clipping from each region and performs the verification stage, which consists of classifying the image into W, WV, WH, and WHV classes.

Both proposed approaches I and II outperforms the baseline results relating to mAP. Our results show Approach-I presenting the best mAP for detecting PPEs (80.65%). Although YOLO-v4-AP2 mAP display superior results (88.87%), the classifiers' errors in the verification stage decrease the final mAP. This effect is evident even for our best implementation, which employs an ensemble of classifiers VGG16, Inception-v3, ResNet50, and ResNet101, producing a final mAP of 71.74% . These results may indicate that superior

results may be obtained from the individual improvement of the classifiers or methods proposed in this work. The ensemble method achieved an increase of up to 2.3% compared to the best single classifier (ResNet101) mAP (69.44%) of Approach-II. Regarding the processing time, Approach-I proved to be more effective because of its one-stage implementation, which avoids bottlenecks between the processing phases. Although slower, our results demonstrate that Approach-II is still feasible to use in real-time, both for the use of a single classifier or with the use of an ensemble.

From the implementation carried out for Approach-I, it is possible to build a monitoring system that has a robust detection and verification component. Since the approach proved to be more efficient, not only in terms of mAP (80.65%) but also in processing time, reaching up to 11x faster (80 FPS) when compared to Approach-II. Because of this, we believe that the one-stage approach has a high potential for the construction of an effective monitoring system that can contribute to the safety of workers, minimizing the number of accidents and occupational injuries.

Regarding ID association component mentioned in Figure 1.1, we believe that tracking algorithms such as DeepSORT (Wojke et al., 2017) may present good results when employed along with the component explored in this work. This happens due to those algorithms working well with robust detection models to track real-time custom objects and assign unique identities for each object.

## Bibliography

- Abdi, H. and Williams, L. J. (2010). Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459.
- Barnich, O. and Van Droogenbroeck, M. (2010). Vibe: A universal background subtraction algorithm for video sequences. *IEEE Transactions on Image processing*, 20(6):1709–1724.
- Barro-Torres, S., Fernández-Caramés, T. M., Pérez-Iglesias, H. J., and Escudero, C. J. (2012). Real-time personal protective equipment monitoring system. *Computer Communications*, 36(1):42–50.
- Bo, Y., Huan, Q., Huan, X., Rong, Z., Hongbin, L., Kebin, M., Weizhong, Z., and Lei, Z. (2019). Helmet detection under the power construction scene based on image analysis. In *2019 IEEE 7th International Conf. on Computer Science and Network Technology (ICCSNT)*, pages 67–71. IEEE.
- Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.
- Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258.
- Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee.
- Developers, G. (2021). K-means advantages and disadvantages. <https://developers.google.com/machine-learning/clustering/algorithm/advantages-disadvantages>, Last accessed on 12-02-2021.
- Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338.
- Fang, Q., Li, H., Luo, X., Ding, L., Luo, H., Rose, T. M., and An, W. (2018). Detecting non-hardhat-use by a deep learning method from far-field surveillance videos. *Automation in Construction*, 85:1–9.

- Francois, C. (2017). *Deep learning with Python*. Manning Publications Company, USA, 1st edition.
- Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.
- Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.
- He, K., Zhang, X., Ren, S., and Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916.
- He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
- He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer.
- Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.
- Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.
- Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *CVPR*.
- Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.
- Li, J., Liu, H., Wang, T., Jiang, M., Wang, S., Li, K., and Zhao, X. (2017). Safety helmet wearing detection based on image processing and machine learning. In *2017 9th International Conf. on Advanced Computational Intelligence (ICACI)*, pages 201–205. IEEE.
- Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

- Liu, S., Qi, L., Qin, H., Shi, J., and Jia, J. (2018). Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8759–8768.
- Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.
- Mitchell, T. (1997). Introduction to machine learning. *Machine Learning*, 7:2–5.
- Mnemyneh, B. E., Abbas, M., and Khoury, H. (2019). Vision-based framework for intelligent monitoring of hardhat wearing on construction sites. *Journal of Computing in Civil Engineering*, 33(2):04018066.
- Nath, N. D., Behzadan, A. H., and Paal, S. G. (2020). Deep learning for site safety: Real-time detection of personal protective equipment. *Automation in Construction*, 112:103085.
- Naticchia, B., Vaccarini, M., and Carbonari, A. (2013). A monitoring system for real-time interference control on large construction sites. *Automation in Construction*, 29:148–160.
- Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning.
- Neubeck, A. and Van Gool, L. (2006). Efficient non-maximum suppression. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 3, pages 850–855. IEEE.
- Neurohive (2018). Vgg16 – convolutional network for classification and detection. <https://neurohive.io/en/popular-networks/vgg16/>, Last accessed on 12-02-2021.
- Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151.
- Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conf. has on computer vision and pattern recognition*, pages 779–788.
- Redmon, J. and Farhadi, A. (2017). Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271.

- Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.
- Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*.
- Ren, X. and Ramanan, D. (2013). Histograms of sparse codes for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3246–3253.
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.
- Sapunov, G. (2018). Hardware for deep learning. part 3: Gpu. <https://blog.inten.to/hardware-for-deep-learning-part-3-gpu-8906c1644664>, Last accessed on 19-02-2021.
- Seo, J., Han, S., Lee, S., and Kim, H. (2015). Computer vision techniques for construction safety and health monitoring. *Advanced Engineering Informatics*, 29(2):239–251.
- Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.
- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.
- Szeliski, R. (2010). *Computer vision: algorithms and applications*. Springer Science & Business Media.
- Toshev, A. and Szegedy, C. (2014). Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1653–1660.



- Turpin, A. and Scholer, F. (2006). User performance versus precision measures for simple search tasks. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 11–18.
- Uijlings, J. R., Van De Sande, K. E., Gevers, T., and Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*, 104(2):154–171.
- Wang, C.-Y., Liao, H.-Y. M., Wu, Y.-H., Chen, P.-Y., Hsieh, J.-W., and Yeh, I.-H. (2020). Cspnet: A new backbone that can enhance learning capability of cnn. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 390–391.
- Wojke, N., Bewley, A., and Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE.
- Wu, J., Cai, N., Chen, W., Wang, H., and Wang, G. (2019). Automatic detection of hardhats worn by construction personnel: A deep learning approach and benchmark dataset. *Automation in Construction*, 106:102894.
- Yuen, M.-C., King, I., and Leung, K.-S. (2011). A survey of crowdsourcing systems. In *2011 IEEE third international conference on privacy, security, risk and trust and 2011 IEEE third international conference on social computing*, pages 766–773. IEEE.
- Zhang, A., Lipton, Z. C., Li, M., and Smola, A. J. (2020). *Dive into Deep Learning*. <https://d2l.ai>.
- Zhang, H., Yan, X., Li, H., Jin, R., and Fu, H. (2019). Real-time alarming, monitoring, and locating for non-hard-hat use in construction. *Journal of construction engineering and management*, 145:04019006.
- Zhao, Z.-Q., Zheng, P., Xu, S.-t., and Wu, X. (2019). Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11):3212–3232.
- Zheng, X., Yao, J., and Xu, X. (2019). Violation monitoring system for power construction site. In *IOP Conf. Series: Earth and Environmental Science*, volume 234, page 012062. IOP Publishing.