IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 21 September 2022, accepted 11 October 2022, date of publication 19 October 2022, date of current version 25 October 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3215148

RESEARCH ARTICLE

A Smart System for Personal Protective Equipment Detection in Industrial Environments Based on Deep Learning at the Edge

GIONATAN GALLO, FRANCESCO DI RIENZO, FEDERICO GARZELLI, PIETRO DUCANGE, AND CARLO VALLATI, (Member, IEEE)
Information Engineering Department, University of Pisa, 56126 Pisa, Italy
Corresponding author: Gionatan Gallo (gionatan.gallo@phd.unipi.it)

This work was supported in part by the Italian Ministry of Education and Research (MIUR) in the Framework of the CrossLaboratory Project (Departments of Excellence).

ABSTRACT Real-time object detection is currently used to automate various tasks in industrial environments. One of the most important tasks is to improve the safety of workers by monitoring the correct use of Personal Protective Equipment (PPE) in dangerous areas. In this context, usually, a monitoring system analyzes the stream of videos from surveillance cameras to assess PPE usage in real time. When a worker not wearing the appropriate PPE is detected, an acoustic or visual alarm is triggered automatically to raise attention and awareness. The solutions proposed so far are mostly cloud-based systems: images from the site are continuously offloaded to the cloud for analysis. This centralized architecture requires significant network bandwidth to transmit the video feeds through an internet connection that must be reliable, as a network outage would disrupt the service. In this work, we propose a system for real-time PPE detection based on video streaming analysis and Deep Neural Network (DNN). We adopt the edge computing model in which the application for image analysis and classification is deployed on an embedded system installed in proximity of the camera and directly connected to it. The system does not require continuous image transmission towards a cloud system, thus ensuring bandwidth efficiency, reliability, and workers' privacy. A prototype of the proposed system is developed exploiting a low-cost commercial embedded system, i.e. a Raspberry PI, equipped with an Intel Neural Compute Stick 2. We tested the system with five different pre-trained convolutional neural networks (CNNs), fine-tuned to detect different PPEs, namely helmets, vests, and gloves. In our experimental evaluation, we first compared the five CNNs in terms of classification performance and inference latency. Then, we deployed each CNN on the real system and evaluated the system's throughput regarding the number of video frames analyzed each second.

INDEX TERMS Edge computing, industrial safety, personal protective equipment.

I. INTRODUCTION

High injury rates characterize many sectors ranging from construction to manufacturing [1]. However, many work accidents can be considered preventable by adopting Personal Protective Equipment (PPE), such as hard hats, shoes, vests, and masks. For instance, more than half (56%) of the accidents in construction sites are caused by the lack of PPE devices [2], while other studies [3] have shown that proactive safety measures, and among them continuous adoption of PPEs, are effective in reducing the accident rate.

Typical examples of PPEs (Figure 1) are helmets, vests, and gloves. Helmets are crucial to prevent fatal accidents in dangerous areas, e.g., the proximity of mechanical lifting devices, automated machinery for assembly line production, or areas with a high risk of falling objects. Vests are usually used to improve workers' visibility, thus reducing the likelihood of accidents in areas where forklifts operate. Finally,

The associate editor coordinating the review of this manuscript and approving it for publication was Zhongyi Guo.

FIGURE 1. Personal protective equipments (PPEs).

protective gloves can mitigate burns and abrasions in case of touching hot surfaces or using power tools. Industrial environments are usually a mix of high- and low-risk areas. In the former, the use of different kinds of protection is usually mandatory, while in the latter, the use of protection is discretionary. Despite their effectiveness, it is hard to guarantee compliance and correct use of PPEs by workers entering high-risk areas [4]. Monitoring performed by human supervisors is often implemented. However, human distractions and difficulties due to crowded areas might still result in a lack of compliance, leading to accidents. Risks could be significantly reduced by employing automated systems, which monitor workers in real-time and alert them or others. There are different ways to protect a worker from improper use of PPEs: acoustic alarms, visual signals, or shutting down dangerous machinery nearby.

Current PPE monitoring systems exploit real-time object detection, a technology already adopted to automate specific tasks, e.g., car recognition for road monitoring [5] or human recognition in video surveillance [6]. Most of the available monitoring systems adopt a cloud-based approach, where video frames are locally captured by cameras and transmitted to a cloud service, where object recognition, mainly based on machine learning, is performed [7]. Such centralized systems, however, require a reliable network connection, as a network outage interrupts the service. Moreover, these systems consume significant bandwidth, as data is continuously offloaded through the Internet to the cloud infrastructure. In addition to this, workers' privacy is not preserved since images are analyzed on an external infrastructure usually owned and operated by a third company.

In this work, we propose a real-time PPE detection system based on deep learning techniques to analyze video frames. The system adopts an edge-computing architecture [8]: the image analysis process is performed on an embedded system installed directly in the proximity of the cameras, i.e., on the same local network as the cameras, without requiring continuous image transmission to an external system. This edge-based architecture results from various advantages: i) it minimizes bandwidth requirements as images are analyzed locally, ii) it improves the overall system reliability as an Internet connection is not required, and iii) it preserves workers' privacy as an external system is not involved.

We developed a prototype of the proposed edge PPE monitoring system based on a low-cost commercial embedded system, i.e., a Raspberry PI 4, empowered through an Intel Neural Compute Stick 2 (NCS2) device, which supports the execution of the model for object detection. In order to select the most suitable Deep Neural Network (DNN), we analyzed and compared different pre-trained convolutional neural networks for object detection, namely the You Only Look Once (YOLO) network version 4 and its tiny version [9], the Single Shot Detector (SSD) with MobileNetV2 [10], the CenterNet with ResnetV2 [11], and the EfficientDet D0 [12]. We fine-tuned the networks above for our purposes using images from industrial environments.

Since the classification performance of the networks depends on both the quantity and quality of the training dataset, we selected images from different sources. In particular, we started our study exploiting an existing public dataset, selected for its large size and suitable for fine-tuning DNNs that require a substantial amount of samples to learn the adequate weights of multi-layered convolutional neural networks. Since some objects were not labeled, we enhanced the dataset by manually updating the labels to satisfy our requirements. Even after this enhancement, however, this first dataset still has some issues: several workers are far from the camera, the environment is generally outdoor, and helmets are over-represented compared with the other PPEs. For this reason, we gathered two additional datasets with images collected in a public event and in the laboratory, allowing us to add more images close to our target environment to our analysis. Specifically, we appropriately selected the scenarios in terms of constraints and features of a "dangerous" industrial environment (e.g. indoor, distance, poses, and obstacles).

We assess the classification performance of the models above in detecting the presence or absence of each PPE typology (helmet, vest, gloves) with a total of six classes of objects, as the absence of a PPE was treated as a distinct object class. In our experimental analysis, the capability of each model to discriminate each class was assessed incrementally: we extended, in two steps, the initial dataset by adding images from the two additional datasets. In addition to accuracy, we also evaluate the models in terms of the inference latency, namely the time required for providing the estimated classes, given an input image. Finally, after deploying all the analyzed models on our prototype, we evaluate their real-time image analysis performance. To this aim, we calculate the average number of video frames that the system can elaborate.

The results from different models highlight a trade-off between the classification performance and their capability to be used in real-time. Indeed, the most accurate model, namely the YOLO network, is characterized by a six times higher latency than its tiny version. However, the latter results in lower detection accuracy, i.e., less than 30% in recognizing small PPEs such as gloves. For this reason, we suggest different networks depending on the specific system requirements. For our PPE detection system, we prioritize the speed of the real-time image analysis. Consequently, the final version of our prototype incorporates the small YOLO network.

This work represents an extended version of our previous contribution presented in [13]. Specifically, the extension includes: (i) a greater set of PPEs involved in the experimental analysis, i.e. safety vests and gloves, in addition to helmets; (ii) a comparison of different families of DNNs; (iii) an additional latency analysis that we carried out for assessing the performance of different DNNs exhaustively.

The paper is structured as follows: in Section II, we overview the related work on systems to monitor workers and their usage of PPEs; in Section III, we describe our scenario and the architecture of our system. Section IV presents the methodology for training and evaluating the different models. In Section V, we discuss the experiment setup, the evaluation metrics, and the results of our experiments. Section VI evaluates the PPE detection system deploying the different models on real hardware. Finally, in Section VII, we draw some conclusions.

II. RELATED WORK

The popularity of video surveillance systems and low-cost cameras led to the creation of large datasets of images of industrial areas, thus enabling Computer Vision (CV) based algorithms for monitoring critical areas [14], [15]. CV-based algorithms are advanced algorithms that process images and return a specific output based on the visual features (e.g., colors, brightness, shapes, etc…) of the provided image. Obviously, the output of CV-based algorithms depends on the specific application.

Several applications based on object identification in industrial environments have been proposed in the literature. Examples of applications include tracking workers moving across construction sites [14], early detection of defects in factory products [16], detecting high-risk situations where workers could fall from construction scaffolding [17], or helmets identification on construction sites [15].

The ever-increasing computing power and the availability of large sets of images promoted the use of Deep Learning (DL) models characterized by a very high-performance level in several tasks that typically involve images and videos. Consequently, these models, especially Convolutional Neural Networks (CNNs) [18], have been widely adopted in the context of object detection. CNNs are very efficient at processing large image databases for supervised learning. Thus, different network typologies have been proposed in the specialized literature for object detection tasks, such as Region-based CNN (R-CNN) [19], Fast R-CNN [20], Faster R-CNN [21], SSD [22], and YOLO [23].

Recently, such poses have been extensively used to enforce workplace safety compliance. Wang et al. [24] exploited R-CNNs to predict collisions and detect workers on construction sites. Zhang et al. [25] designed a framework for monitoring the correct use of protective helmets based on Faster R-CNN. Authors in [26] designed a helmet-wearing detection system by integrating the Faster R-CNN with face detectors and CNNs to improve the performance of each algorithm.

In most works, it is common to use transfer learning techniques on pre-trained object detectors to recognize PPEs in an image. Recently, Wang et al. [27] compare different versions of YOLO detectors to identify vests and helmets, testing the results on images where the workers' heads are blurred. Nath et al. [28] study three different techniques to identify the correct use of helmets and vests. Specifically, the authors compare three configurations involving an object detector, an ML classifier, and a decision tree exploited for image analysis. Compared with [28], Iannizzoto et al. [29] add more PPEs such as helmets, headphones, vests, and masks. The authors propose a system based on DNN used with fuzzy filtering, specifically considering multiple versions of the YOLO and EfficientDet family of object detectors. However, the authors consider only a small dataset in their performance evaluation. Compared with our work, the two works in [27] and [28] concentrate only on the server- or cloud-based solutions estimating the values of inference time on GPU-enhanced machines. In contrast, the work in [29] provides some experimental analysis also considering the deployment of the models on real hardware (NVIDIA Jetson Nano). However, none of the three works on transfer learning consider small objects such as hands and gloves.

Other works in the literature propose PPE detection systems that exploit a pose estimator to detect the region of interest, i.e., in the human body, in which the PPE may be present or not. For instance, Chen et al. [30] propose a solution to identify improper use of PPE by combining DL-based object detection and geometric relationships with a pose estimator. Xiong et al. [31] use a pose estimator to identify the region of interest and then classify if those regions include hard helmets or safety vests.

CNN-based detection systems that process real-time video streams usually require substantial computing capabilities. Real-time video analysis continuously takes high-definition frames from cameras and needs high computation, high bandwidth, high privacy, and low latency to analyze the content of each frame. One viable approach that can meet these strict requirements is edge computing [32]: this paradigm considers computing capabilities as close as possible to devices producing data. Edge AI comprises solutions at different levels of edge, from almost everything on the cloud to all on-device [33]. Recently, systems based on Edge AI have been designed for video surveillance [34], [35], smart healthcare [36] and autonomous driving [37], [38].

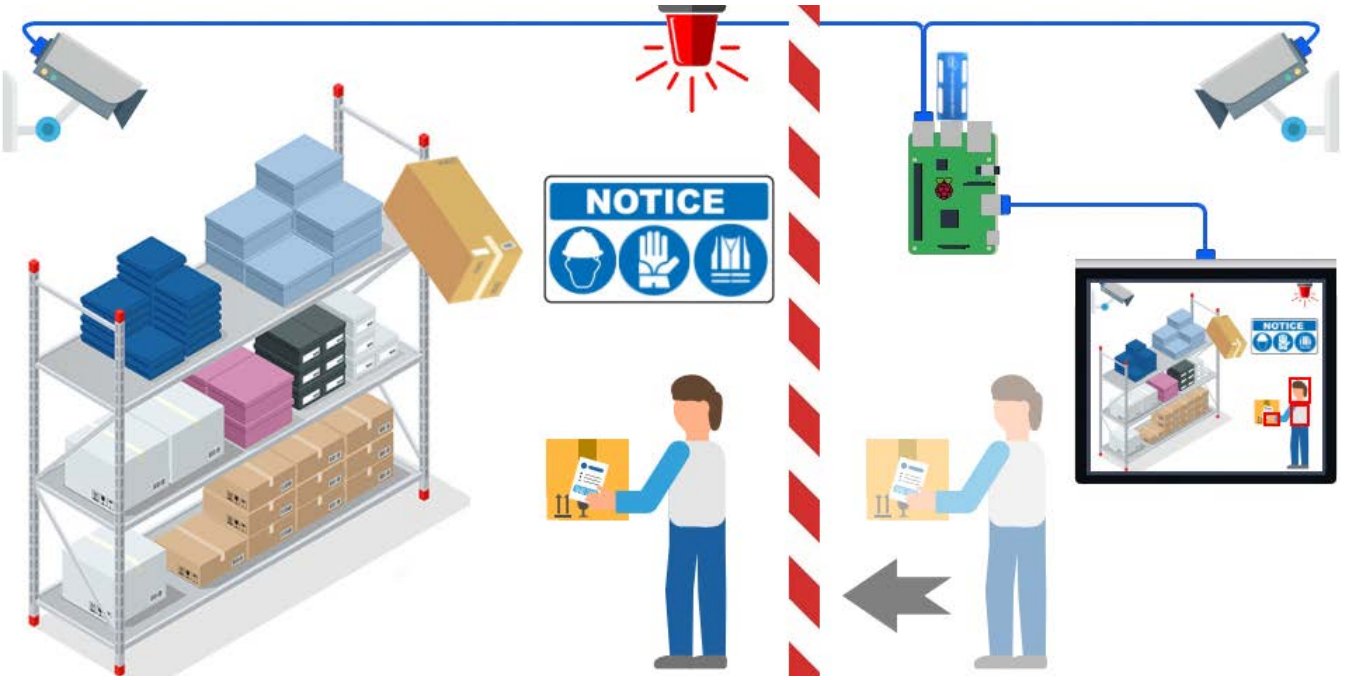

FIGURE 2. A possible scenario for monitoring the correct use of PPEs in an industrial environment through video stream analysis: a worker enters a dangerous area and the monitoring system alerts the worker not wearing the PPEs.

Most of the approaches described above do not consider the possibility of deploying their solution at the edge, i.e., on an embedded system characterized by limited resources. In this paper, instead, we propose a smart system for PPE detection and monitoring based on edge computing. Specifically, we discuss an approach designed to run on an embedded system installed near the cameras monitoring a working site. In order to select the most suitable model for the image classification task, different DL techniques are assessed in terms of model accuracy and latency for analyzing the frames of the video streams in real-time.

III. THE PROPOSED PPE DETECTION SYSTEM

A. PPE DETECTION SCENARIO

Different wearable protections are often mandatory in industrial areas for workers to prevent serious injuries.

Different body parts should be protected from injuries by adopting different PPEs (see Figure 1). Firstly, the head should be protected from falling objects using hard hats. Moreover, in the head, we can also protect our ears and eyes from noisy machines and shards of glass by wearing hearing protection and using safety goggles, respectively. Second, we can improve the visibility of the chest area by wearing a safety vest or ensure stability by wearing a harness. Finally, body limps are vulnerable to burns or scratches, so workers should wear gloves and safety shoes.

In our work, we select one PPE for each body area to protect. For the upper area, we choose safety helmets, as they prevent critical injuries to the head. Second, we consider safety vests since they are widely used in many industrial contexts. Finally, for the arms and legs, we focus on recognizing gloves because workers often use their hands to interact with machines.

The use-case considered in this paper is depicted in Figure 2. The workspace is a mixed space with low-risk and high-risk areas. In the latter, using PPEs, namely a helmet, a vest, and gloves, is mandatory to protect the worker. The proposed system aims to analyze real-time images captured by a surveillance camera to detect workers not wearing PPEs. If a worker enters a high-risk area without protection, the system raises a visual or acoustic alarm to alert the worker. Specifically, an alarm will be issued for each PPE not worn. The system could also be connected to a controller that shuts down potentially dangerous machinery in the high-risk area, thus preventing injuries and improving safety.

B. SYSTEM ARCHITECTURE

Our system deploys a DNN to analyze the images on an edge computing node, i.e., an embedded system with limited computing capabilities. The edge node is physically close to the supervised environment: it receives the images via a Local Area Network (LAN) and analyzes them without transmitting them to an external system or a cloud-based service, thus preserving workers' privacy. In addition, since the system works in isolation and does not transmit any data, it is resilient to network outages and does not consume any bandwidth to offload the images to a cloud service.

In Figure 3, the architecture of our prototype created by exploiting commodity hardware is shown. The system comprises three main components: an embedded computing node,

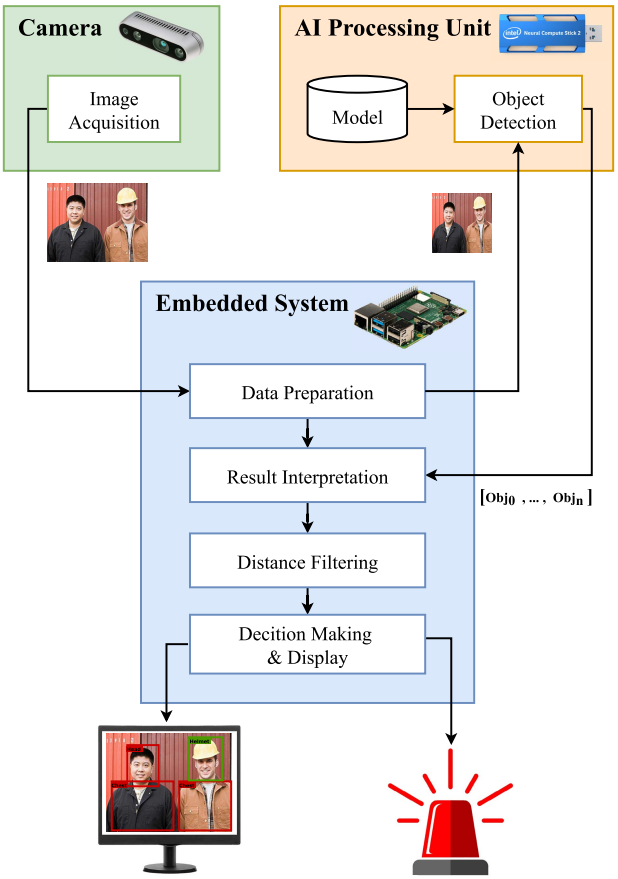

FIGURE 3. Overall architecture of the PPE detection system.

an AI processing unit, and a video camera. At first, the camera periodically sends RGB and depth information to the central node. The latter processes an RGB frame and sends it to the AI processing unit, which is in charge of detecting the presence or not of PPEs within the image. Upon the reception of a list of objects, the embedded system can filter out the objects outside the dangerous area by combining their locations and distances from the machinery (depth frame). Finally, the system displays the current state of detection on a monitor, showing each incorrect use of PPE as a red bounding box and as a green one otherwise. If the system detects at least one missing mandatory PPE, it can act upon such violation with visual or acoustic alarms.

We adopted a Raspberry Pi 4 board as an edge computing node for our PPE detection system. The embedded system is equipped with an Intel Neural Stick 2 (NCS2) as an AI processing unit. NCS2 is a USB device that can support hardware acceleration for inference-based computing. The Intel RealSese Depth Camera D435 was adopted as a video camera.

The Raspberry Pi 4 is the last revision of the Raspberry PI single-board computer. It costs around 35$ and it is very popular (approximately 30 million Raspberry Pi have been sold worldwide). Due to the low computing capabilities of the Raspberry PI, real-time image object detection is supported via the NCS2, which can support object detection via DL through hardware acceleration.

In order to measure the position of the worker in the area, we adopted the Intel RealSense depth camera D435, a low-cost camera equipped with infrared stereo support that can measure the distance between the detected objects and the camera. he system can use this additional information to enable/disable the alarm according to the worker's position, i.e., enable the alarm only when the worker is inside a high-risk area. It is important to notice that this is not a critical feature; the system can still work with cameras that do not provide distance estimation. In this case, however, alarms are triggered in the whole area without distinction between high- and low-risk.

C. SOFTWARE DESCRIPTION

We implemented Python software that runs on the Raspberry PI for the entire image processing pipeline, including acquisition, manipulation, and analysis. As regards the first two stages, we exploited two python libraries: the Intel SDK,[1] which continuously gets images from the Intel camera, and the OpenCV[2] library for image manipulation.

The image analysis process for PPE detection is carried out by exploiting a DNN for object detection. Image-based object detection is a computer vision task that consists of the following three sequential phases: (i) identify the presence of one or more objects on the image; (ii) locate the objects within the image boundaries; (iii) classify each object into one category, among the set of pre-defined categories. Specifically, the DNN initially identifies the regions of interest inside the image, considering all possible objects, and then labels each region to one of the possible classes of objects. A position is expressed through a bounding box, i.e., a rectangle surrounding the object and defining a specific region of interest. In other words, the object detection process consists of identifying the objects belonging to the categories of interest and associating each to a bounding box with the proper class label (i.e., the corresponding category).

As highlighted in Section II, CNNs are the most popular DNNs for object detection. These models allow to extract, at each level of the network, different image features characterized by an increasing level of specificity, i.e. from general to specific features. In this work, we consider a specific category of DNNs based on CNN, which adopts the one-stage regression-based detection approach: objects are detected by applying a regression process to predict simultaneously the target region and its category, i.e., the class. In order to select the most suitable model to deploy in our PPE detection system, we analyzed the following state-of-the-art object detectors: 'You Only Look Once version 4' (YOLOv4) [9] and its lightweight version (YOLOv4-Tiny), Single-Shot Detection (SSD) [10], CenterNet [11], and EfficientDet [12].

[1]https://dev.intelrealsense.com/docs/python2
[2]https://pypi.org/project/opencv-python/

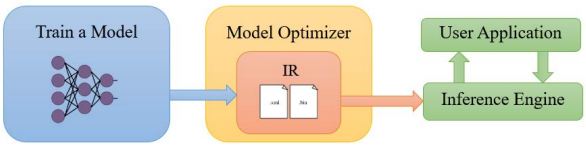

FIGURE 4. OpenVINO workflow for deploying a trained DNN.

As discussed in Section VI, in order to evaluate the capability of working in real-time in an actual device, we deployed all the analyzed DNNs on our PPE detection system. To this aim, we exploited Intel's OpenVINO [39] (Open Visual Inference and Neural network Optimization)[3] free toolkit that is widely adopted for deploying object detection solutions for both images and video frames, since it minimizes the overall deployment workflow. Figure 4 shows the OpenVINO workflow for preparing the model to be deployed for the NCS2. The OpenVINO framework includes a Model Optimizer and an Inference Engine, which simplifies the transition from the training phase to the deployment of the application. The Model Optimizer imports an already-trained DNN and optimizes it for execution on a target device by transforming it into an intermediate representation (IR), a representation optimized for the Inference Engine. The latter, at runtime, loads into memory the IR to infer the result of the network for a given input. The user application has to be developed to integrate the Inference Engine that eventually reads the model and runs the inference on the target device.

First, the neural network has to be trained. Usually, the training process is performed by exploiting a deep-learning framework running on a powerful server. To this aim, OpenVINO includes support for several popular formats of trained networks. For example, TensorFlow, PyTorch, and Caffe can be used to train networks that will then be deployed using OpenVINO. After training the model, the Model Optimizer tool is exploited to produce the IR used at runtime on the embedded device. This operation is lossless, so given an input, both versions of the model will produce the same result.

IV. DNNs TRAINING AND EVALUATION METHODOLOGY FOR PPE DETECTION TASK

In this section, we briefly describe the methodology adopted for training and evaluating the different DNNs that we considered in our work to implement the PPE detection task. This stage aims to identify the most suitable DNN to use in our PPE detection system. Specifically, we first introduce and describe the exploited datasets and then the training and evaluation workflow.

A. DNNs FOR PPE DETECTION TASK

Training a predictive model, such as DNNs for object detection in images, generally requires a set of suitable labeled training samples. In our case, we require a set of RGB images and the corresponding set of objects present on each image, with their position and label. A training dataset for our use case should include images from industrial workplaces, with a good mixture of the PPE objects to be detected, i.e., hard helmets, safety vests, and protective gloves. It is also important that the training dataset includes images with different ambient conditions, backgrounds, distances, and angles from the camera. This diversity improves the training of the DNNs to recognize new and heterogeneous scenarios.

When dealing with DNNs, especially for the object detection task in images, the higher the number of labeled training samples, the better the quality of the trained model. This aspect is due to the huge amount of parameters to be optimized in such models. Thus, a large set of labeled images is needed to train a DNN from scratch. However, collecting and labeling images are very time-consuming tasks. Thus, researchers and practitioners usually exploit the transfer learning methodology [40] in which pre-trained models for approaching general tasks, such as a general object detection task, are selected and then fine-tuned. The fine-tuning process consists of adapting a pre-trained model to a more specific task, i.e., recognizing a specific set of PPEs in our case. For the pre-training stage of DNNs for object detection in images, a collection of datasets containing millions of images labeled with thousands of categories is currently available and recognized as effective by the specialized literature [41], [42].

A set of very accurate pre-trained DNNs, including the object detection task, is available in several machine learning and artificial intelligence software libraries and on verified online repositories. In this work, we compared the performance of five well known DNNs for object detection, namely the YOLOv4 network [9] and its lightweight YOLOv4-Tiny version, SSD MobileNet V2 [10], CenterNet V2 [11], and EfficientDet D0 [12]. For each of them, first, the available pre-trained network generated considering the COCO dataset[4] was downloaded from the specific public code repository. A pre-trained implementation of YOLOv4 and YOLOv4-Tiny can be found in Alexey's Github repository,[5] while the others are available in the Tensorflow official repository.[6] Then, we fine-tuned each pre-trained network w.r.t. our specific task. To this aim, we considered three different datasets. The following describes the datasets and the procedure for fine-tuning and comparing the five DL networks.

B. DATASET

Generally, creating large labeled datasets is time-consuming: first, the set of suitable images for a specific task must be identified; subsequently, such images must be labeled. The latter is usually performed manually, which might also result in errors.

As already mentioned, our PPE detection system aims at recognizing the presence or the absence of three PPEs,

[3]https://https://docs.openvinotoolkit.org/latest/index.html

[4]https://cocodataset.org/

[5]https://github.com/AlexeyAB/darknet/releases/download/

[6]https://github.com/tensorflow/models/

namely helmets, vests, and gloves. Thus, the images of the dataset should include six different classes, namely *head* without an helmet, head with *helmet*, *chest* without vest, chest with *vest*, *hand* without glove, hand with *glove*. For the sake of simplicity, we will refer to these classes as *head*, *helmet*, *chest*, *vest*, *hand* and *glove*, respectively.

In our experimental analysis, we considered three datasets, namely, D_1, D_2, D_3. These datasets were used to fine-tune the pre-trained DNNs we analyzed and compared. This comparison aims at identifying the most suitable DNN to be deployed in our system.

For the creation of dataset D_1, a public dataset[7] was used to speed up the analysis and rapidly generate the first version of fine-tuned DNNs. The initial public dataset comprises images from industrial environments, including scenarios with workers wearing different PPEs. Initially, the dataset includes 7035 images with one or more workers performing different activities in construction sites and industrial environments. Both indoor and outdoor images are included in the dataset, which results in heterogeneous distances between objects and the camera. The quality of the images varies: in some images, workers are very close to the camera, thus showing only a portion of their bodies. In others, they are distant, so barely recognizable to the human eye. For this reason, the images of the dataset were filtered to remove those with low resolution (less than 200×200px) or where workers were too far away (long distance) or too close to the camera (very short distance).

After the filtering, duplicates were also removed. Duplicates were detected using the Mean Structural SIMilarity index (MSSIM) [43]. This index was adopted since it is resilient to image alterations, such as blurring and scaling, thus allowing for the detection of duplicates where one of the images was slightly altered. Images with an MSSIM index higher than 0.95 were marked as duplicates. Only the version with the higher resolution was kept for each pair of images detected as a duplicate. Overall, 203 images from the initial 7035 were removed, resulting in 6832 images for D_1.

Finally, the labeling of the images of D_1 was improved as the initial dataset included images only partially labeled. Indeed, the original dataset was composed by images tagged only with *head* and *helmet* classes. Thus, in order to use the dataset for training DNNs for also recognizing vests and gloves, we manually added the missing class labels to identify also the *vest*, *chest*, *glove*, and *hand*. An example of annotated image is shown in Figure 5. It is worth noticing that during the labeling process, we also verified and calibrated the bounding boxes of the pre-existing labeling to ensure precision.

Even though D_1 processing resulted in a rich set of images, it still has multiple issues. First, the diversity of the workers in terms of race and gender is low. Moreover, most outdoor images contain workers too far from the camera; consequently, the PPEs they wear are challenging to recognize. Even though we only kept high-resolution images, several ones still have poor lighting and artifacts. Finally, some PPEs are not common, e.g., some vests have non-standard colors, or they include stripes.

FIGURE 5. An example of an annotated image from the Roboflow dataset.

In order to mitigate the issues mentioned above of D_1, two additional datasets were created by collecting new images in a controlled environment: we labeled these datasets as D_2 and D_3.

Dataset D_2 was collected by setting up a green screen positioned at a fixed distance from the camera. Pictures were taken during a public event[8] where we asked volunteers to wear one PPE from a set of available PPEs, i.e., helmets, vests, and gloves, and strike a pose from a set of pre-defined ones, e.g., a worker using a power tool, like an electric drill or a hammer. As a result, we collected 215 images, and each picture included a varying number of volunteers from one to four. Images were processed to remove the green background and replace it with another more realistic one from different indoor industrial settings (see two examples in Figure 6).

Our laboratory's third dataset D_3 was created considering typical industrial scenarios. Specifically, several hazardous situations in which workers are required to wear one or more PPEs are considered, e.g., a worker operating a press. D_3 is composed of 236 images similar to those shown in Figure 7. As shown in Figure 7, we staged frontal images showing industrial machines and semi-hidden operators (Fig. 7a), and images collected at a longer distance to simulate workers in the background outside of the critical areas (Fig. 7b).

Table 1 summarizes the class distributions in each dataset. Specifically, we show the number of instances representing each class. It is worth to notice that in the three datasets all classes are well represented, even though some of them, such as *vest* in D_1, *head* and *chest* in D_2 and *vest* in D_2 are slightly under-represented. The table also shows, in the last columns, the class distributions of the dataset $D_{1,2,3}$, obtained by merging all the images of the three datasets (see Section IV-C for more details). Even though the dataset is not extremely imbalanced, we performed a preliminary analysis in which

[7]https://public.roboflow.com/object-detection/hard-hat-workers

[8]https://www.bright-night.it/

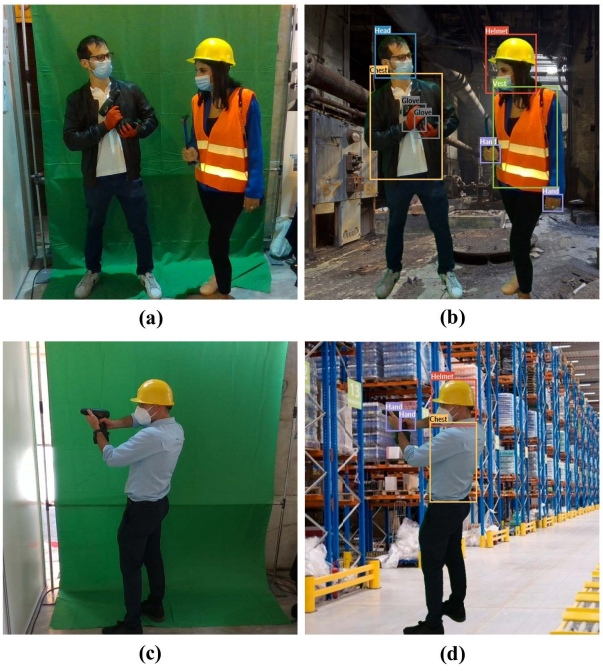

FIGURE 6. Examples of D_2: photos of people in front of a green screen (*a*, *c*). Processed images with a more suitable background (*b*, *d*).

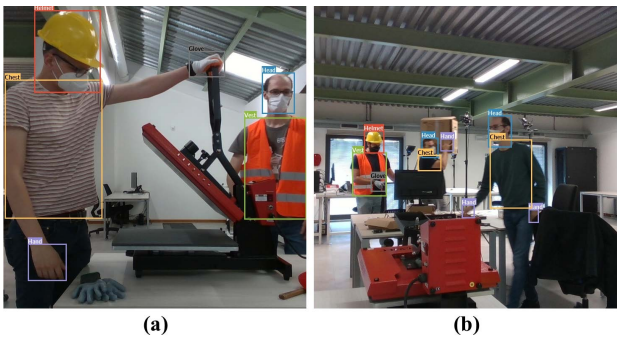

FIGURE 7. An example of annotated images from D_3.

TABLE 1. Number of instances per class in each dataset.

ID	Class	$\#D_1$	$\#D_2$	$\#D_3$	$\#D_{1,2,3}$
0	head	6906	136	181	7223
1	helmet	19305	190	153	19648
2	chest	19606	148	188	19942
3	vest	2574	182	154	2910
4	hand	11947	279	216	12442
5	glove	6561	254	247	7062

we applied some re-balancing techniques, omitted here for brevity, and did not show a significant improvement in the DNNs performance.

C. DNNs FINE TUNING AND COMPARISON WORKFLOW

The models' fine-tuning and comparison were carried out through three phases with a dataset D_x increased incrementally with data from D_1, D_2, and D_3. The goal is to analyze how additional images used in training influenced the accuracy of the models w.r.t. the different PPEs considered in this work. During each phase, a stratified five-fold cross-validation is used for training and testing. Figure 8 shows the schematizing of the cross-validation analyses we carried out.

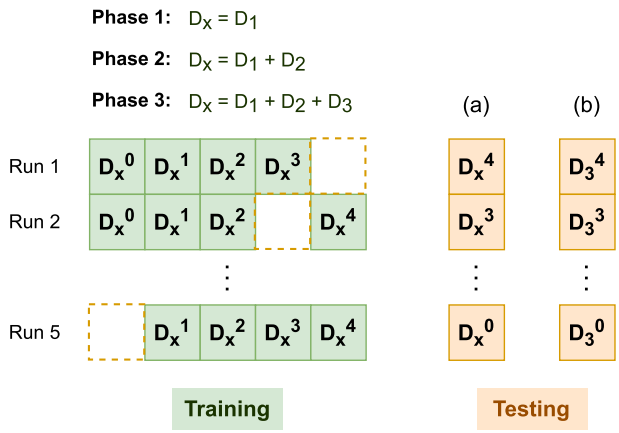

FIGURE 8. Schematizing of the cross-validation analyses.

In phase 1, D_1 was considered as D_x to perform a preliminary comparative analysis of the five DNNs five-tuned for the PPE detection task. We labeled this analysis as D_1 Cross-Validation. Subsequently, in phase 2 and 3 we extended the dataset to be considered in additional cross-validation experiments: in phase 2 we created the dataset $D_{1,2}$ composed by the instances of D_1 and D_2 and in phase 3 a dataset $D_{1,2,3}$ composed by the instances of D_1, D_2 and D_3. The former is hereafter labelled as $D_{1,2}$ cross-validation, the latter as $D_{1,2,3}$ cross-validation.

In the cross-validation analyses carried out in the three phases, we also calculated the generalization capability of each network generated at each training run i on the *i-th* test fold of D_3. Indeed, D_3 is the dataset containing the most realistic scenarios for the PPE detection task, as they were staged in a controlled environment using practical tools. The *i-th* test fold of D_3 has been extracted from the *i-th* test fold of $D_{1,2,3}$, considering only the instances belonging to D_3. Notice that the *i-th* test folds of D_3 represent reference test data on which we can evaluate the improvement of the different DL networks along the process of extending the training datasets with more realistic data, considering the specific contest of PPE detection in indoor industrial environments.

As discussed in the following sections, we first compare the performance of the DL in terms of their object detection performance and then their latency in providing the classification of each video stream image.

V. DNNs COMPARISON: EXPERIMENTAL RESULTS

In this section, we show the experimental analysis results, which we carried out to assess the performance of the different models we considered for the PPE detection task. In our experiments, we assume that: i) each model is pre-trained with the same set of images (i.e. COCO dataset) and

fine-tuned using D_1, D_2, and D_3 as discussed in Section IV-B, ii) D_1, D_2, and D_3 do not include any image from the COCO dataset, iii) only single-stage object detectors that can be converted to IR have been selected for the experimental comparison, iv) the selected pre-trained models have similar input layer sizes, and v) the model conversion from TensorFlow to OpenVINO is lossless.

A. EXPERIMENTAL SETUP

The processes of fine-tuning and evaluation have been carried out on a server powered by an 8-core AMD EPYC@2195MHz CPU, 32 GB of RAM, and equipped with an NVIDIA Tesla T4 GPU with 16 GB of memory. As regards the YOLO networks, we adopted the Darknet framework[9] for training and evaluating the models. As regards the other networks, we resorted to the Tensorflow Object Detection API.[10] The former is a framework specifically designed to build and test networks from the YOLO family. The latter is an open-source framework built on top of TensorFlow. It allows easy building, training, testing, and deploying object detection models.

Table 2 shows the parameters which describe the structure of the models and those adopted by the fine-tuning algorithms for each DNN. Specifically, we report the input size (*IS*) of each network, the number of millions of trainable parameters (*MParams*) and the number of Giga (10^9) of FLoating OPerations (*GFLOPs*) per single inference. All the above are structural parameters, which are strictly related to the complexity of the models, As regards the input size, although each model is available with different sizes of the input image in the repositories from which we downloaded the pre-trained networks, not all networks are available with the same input dimensions. Therefore, we selected the networks with input sizes most similar to YOLO networks among those available: as shown in the table, in our experiment, EfficientDet D0 and CenterNet Resnet50 V2 have slightly larger input sizes than YOLO networks. In comparison, SSD MobileNet V2 has slightly smaller input sizes than YOLO networks.

Table 2 also shows fine-tuning parameters such as the number of training steps in which the algorithm updates the weights according to a descending gradient algorithm (*NSteps*) and the batch size as the number of images presented to the network at each iteration (*BSize*). Finally, we also report the parameters of the training algorithm that adopts a variable learning rate with momentum. The latter, specifically, changes linearly from warm-up η_{init} to η_{base} every $n_{warm-up}$ steps, thus keeping the algorithm constant or increasing its speed towards the minimum if the gradient of the loss function keeps pointing in the same direction.

We recall that for each model, we run a grid search algorithm to find the optimal value of some hyper-parameters using dataset D_1.

[9]https://github.com/AlexeyAB/darknet

[10]https://github.com/tensorflow/models/tree/master/research/object_detection

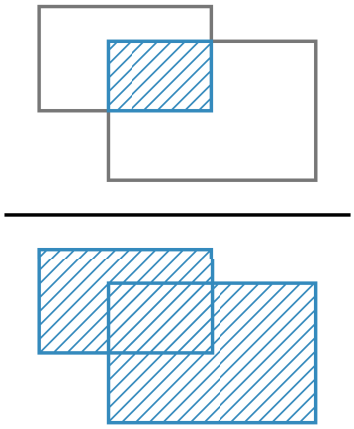

FIGURE 9. Intersection over union (IoU).

B. EVALUATION METRICS

The most common metric to assess an object detection system's performance in the specialized literature is the Average Precision (AP) [44]. Before introducing the specific version AP adopted in our experiments, we need to recall the following concepts:

- True Positive (TP): a correct detection of a ground-truth bounding box;
- False Positive (FP): incorrect detection of a nonexistent object or a misplaced detection of an existing object;
- False Negative (FN): an undetected ground-truth bounding box.

In object detection, the number of True Negatives (TN) is irrelevant because there is an infinite number of bounding boxes that could not be detected within any given image.

The above definitions of TP and FP are still incomplete because we did not give a distinct definition of what is a correct or an incorrect detection. A common way to distinguish a correct detection is using an Intersection Over Union (IoU) threshold. In object detection, the IoU measures the overlapping area between the predicted bounding box B_{pred} and the ground-truth bounding box B_{true} divided by the area of union between them, such as:

$$IoU = \frac{area(B_{pred} \cap B_{true})}{area(B_{pred} \cup B_{true})} \tag{1}$$

Figure 9 shows the concept of IoU intuitively. Given a certain threshold t ($0 \leq t \leq 1$), any detection is classified as *correct* if $IoU \geq t$, *incorrect* otherwise. It is common to evaluate the performance of an object detector adopting *Precision* and *Recall* defined as follows:

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{\text{all detections}} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{\text{all ground truths}} \tag{3}$$

Precision is the ability of a model to identify only relevant objects. It is the number of correct positive detections divided

TABLE 2. Structural and fine-tuning algorithm parameters values for each DNN.

	YOLOv4	YOLOv4-Tiny	SSD	CenterNet	EfficientDet
IS	416x416	416x416	320x320	512x512	512x512
MParams	64.0	6.1	16.8	17.9	3.9
GFLOPs	60.1	6.9	3.8	62.2	2.5
NSteps	12000	12000	24000	16000	24000
BSize	64	64	64	16	8
η_{init}	0	0	$1.6 \cdot 10^{-3}$	$2.5 \cdot 10^{-5}$	$2 \cdot 10^{-3}$
η_{base}	10^{-3}	$2.61 \cdot 10^{-3}$	$8 \cdot 10^{-3}$	10^{-4}	$8 \cdot 10^{-3}$
$n_{\text{warm-up}}$	1000	1000	960	320	400
Momentum	0.949	0.9	0.9	0.0	0.9

by the total number of predictions. Instead, recall measures how well a model is able to find all the ground-truth bounding boxes. It is the percentage of correct positive predictions among all given ground truths.

We can now build a precision-recall curve by changing the threshold for detecting positive detection. If we set a high threshold value, there will be few but precise detections, so generally, a high level of precision but a low recall. On the contrary, by lowering the threshold, the recall will increase, while the precision is expected to stay high. Hence, the higher the area under the Precision-Recall curve, the better the performance in detecting objects of a specific model.

In our study, we use the typical 11-point precision-recall interpolation. During this process, the shape of the precision-recall curve is summarized by averaging the maximum precision values at a set of 11 equally spaced recall levels [0, 0.1, 0.2,..., 0.9, 1]. In the following, we introduce the AP_{11} metric:

$$AP_{11} = \frac{1}{11} \sum_{R \in \{0, 0.1, \ldots, 0.9, 1\}} P_{int}(R) \tag{4}$$

which represents the average of all the 11 interpolated precision values $P_{int}(R)$ calculated as in formula (5)

$$P_{int}(R) = \max_{\tilde{R}: \tilde{R} \geq R} P(\tilde{R}) \tag{5}$$

where $P(R)$ is the precision for a given recall level R, the interpolated precision $P_{int}(R)$ is the maximum value of precision whose recall is greater than R.

In the following, when dealing with the AP of each class, we will adopt the notion AP@50, when considering an IoU threshold equal to 0.5. In literature, AP@50 is a well-established metric to compare the performance of different object detectors [10], [12], [28]. Moreover, in order to provide a global accuracy metric for a specific DNN, we will resort to the mean Average Precision (mAP), simply calculated as the mean value of AP among all classes:

$$mAP = \frac{1}{N} \sum_{h=1}^{N} AP^h \tag{6}$$

where N is the number of classes to be detected, also, in this case, we will use the notion of mAP@50 considering an IoU threshold equal to 0.5.

To assess the system's performance in real-time, we conduct two types of analyses. First, we evaluate the inference time, namely the time needed for a model to perform object detection on a single image. This first analysis runs the fine-tuned models on the previously introduced server. Subsequently, to measure the performance of the DNNs on the considered system, we have deployed all the DNNs on the proposed PPE detection system presented in Section III. In this analysis, we considered the number of video frames analyzed each second by the system, which is dual with latency. However, it provides a direct measurement of the performance, which can be used to assess the feasibility of adopting our system in the considered use case.

C. OBJECT DETECTION PERFORMANCE RESULTS

In this section, we show the results of the object detection performance achieved by the five DNNs previously introduced. Specifically, we discuss the results of the cross-validation analysis considering first D_1 (Sec. V-C1), then $D_{1,2}$ (Sec. V-C2) and, finally, $D_{1,2,3}$ (Sec. V-C3). As mentioned before, in each analysis, we also extracted the average results achieved on the test folds of dataset D_3, representing a realistic scenario for PPE detection in an industrial environment. On the test set, the object detection performance level has been calculated in terms of average values of AP@50 per class. Moreover, we provide some summarizing results on the test set in terms of average mAP@50 in percentage. We recall that, in the following, we always refer to the stratified k-fold cross-validation with $k = 5$.

In order to offer a more effective visualization of the results obtained with the different models, in every table, we report the best result in **bold** and the second best result in *italic*.

1) D_1 CROSS-VALIDATION RESULTS

Table 3 shows the average results of cross-validation on D_1. We can see that YOLOv4 outperforms the other models by a wide margin for all the classes. As expected, YOLOv4 achieves better results than YOLOv4-Tiny because the latter is designed to be faster. Indeed, it is characterized by a lower number of parameters, thus hindering its ability to recognize objects precisely. However, YOLOv4-Tiny still maintains good recognition levels for the *head*, *helmet*, *chest* and *vest*

TABLE 3. D_1 cross-validation: average values of AP@50 per class calculated on D_1.

	Head	Helmet	Chest	Vest	Hand	Glove
YOLOv4	**96.4**	**98.2**	**94.9**	**86.7**	**80.4**	**63.6**
YOLOv4-Tiny	*92.6*	94.6	*91.8*	*79.8*	57.0	35.9
SSD MobileNet V2	77.3	86.2	81.3	69.9	43.1	21.3
CenterNet Resnet50 V2	*92.6*	*95.4*	91.1	75.2	*64.5*	*39.9*
EfficientDet D0	83.8	86.6	90.1	79.3	39.5	34.1

TABLE 4. D_1 cross-validation: average values of AP@50 per class calculated on D_3 test sets (in percentage).

	Head	Helmet	Chest	Vest	Hand	Glove
YOLOv4	*91.7*	**97.5**	**96.7**	*98.4*	**70.1**	**44.9**
YOLOv4-Tiny	86.3	*96.7*	91.1	**99.5**	54.3	29.5
SSD MobileNet V2	86.8	93.5	85.9	96.9	47.0	29.9
CenterNet Resnet50 V2	**93.2**	95.1	*93.0*	97.1	*62.6*	*44.2*
EfficientDet D0	90.2	96.5	91.5	97.1	62.5	43.1

TABLE 5. $D_{1,2}$ cross-validation: average values of AP@50 per class calculated on $D_{1,2}$.

	Head	Helmet	Chest	Vest	Hand	Glove
YOLOv4	**96.4**	**98.2**	**95.0**	**87.4**	**80.9**	**63.8**
YOLOv4-Tiny	*93.5*	94.8	*92.0*	*82.5*	57.9	38.4
SSD MobileNet V2	77.5	86.2	81.2	72.4	42.8	21.3
CenterNet Resnet50 V2	92.6	*95.3*	91.2	77.7	*64.2*	*40.2*
EfficientDet D0	83.5	86.4	90.1	81.2	40.2	36.0

classes. In general, *head*, *helmet* and *chest* classes are characterized by the highest recognition level. Specifically, for the YOLOv4, YOLOv4-Tiny and CenterNet Resnet50 V2, the values of mAP@50 are higher than 90% (up to 96.4%, 98.2% and 94.9%, respectively for YOLOv4).

It is worth noticing that among the different networks, only CenterNet Resnet50 V2 achieves performances similar to the YOLO networks. Specifically, when comparing YOLOv4-Tiny and CenterNet Resnet50 V2, the two models have the same average accuracy for *head* (92.6%). Instead, the former performs better for the *vest* (+0.7%) and *chest* (+4.6%), but worse for *helmet* (−0.8%), *hand* (−7.5%), and *glove* (−4.0%).

The inferior results obtained with SSD MobileNet V2 can be partially explained by the fact that the model accepts images with lower resolution (320x320) than the others.

In Table 4, we show the average results of the D_1 cross-validation calculated on the test folds of D_3. YOLOv4 is still the most accurate among the five models, except for the *vest* class for which YOLOv4-Tiny achieves the highest AP@50 value of 99.5%, and *head* class for which CenterNet Resnet50 V2 achieves the highest AP@50 value of 93.2%. Apart from YOLOv4, CenterNet Resnet50 V2 achieves the second best results for *head*, *chest*, *hand*, and *glove* classes. For the remaining classes, namely *helmet* and *vest*, YOLOv4-Tiny is the second-ranked networks after YOLOv4. It is worth noticing that CenterNet Resnet50 V2 can better identify small objects than YOLOv4-Tiny, and this can be mainly because CenterNet Resnet50 V2 takes as input images with a higher resolution than YOLO networks. SSD MobileNet V2 still is the worst performing network, while EfficientDet D0 has a performance behavior similar to YOLOv4-Tiny, even better for *hand* and *glove* classes.

2) $D_{1,2}$ CROSS-VALIDATION RESULTS

This section discusses the results achieved by performing cross-validation with dataset $D_{1,2}$. We recall that this dataset has been created by merging D_1 and D_2.

As shown in Table 5, YOLOv4 achieves, also in this case, the highest performance levels. Data in Table 5 highlights that YOLOv4-Tiny and CenterNet Resnet50 V2 outperform the remaining two networks for the *head* (93.5%),*chest* (92.0%) and the *vest* (82.5%) classes, and the *helmet* (95.3%), *hand* (64.2%) and *glove* (40.2%) classes, respectively. It is worth noticing that CenterNet Resnet50 V2 can recognize better small objects (i.e., hands and gloves) than YOLOv4-Tiny. On the contrary, YOLOv4-Tiny performs better detecting large objects such as chests and vests.

Table 6 shows the average results of the $D_{1,2}$ cross-validation calculated on the test folds of D_3. If we exclude YOLOv4, which achieves the highest performance in all classes except for *head* class, as shown in Table 6, CenterNet Resnet50 V2 outperforms the other models in recognizing the *head* (96.8%), the *chest*(94.2%) and the *glove* (53.5%) classes. As regards *helmet* and *vest* classes, the best performance is achieved by YOLOv4-Tiny, with values of AP@50 equal to 97.7% and 98.0%, respectively.

In Table 7 we show the percentage of improvement of the average values of AP@50 in recognizing each class, on the D_3 test set, after adding D_2 to D_1 for creating $D_{1,2}$. Data in Table 7 highlights that the introduction of D_2 in training improved the detection of small objects, especially for the

TABLE 6. $D_{1,2}$ cross-validation: average values of AP@50 per class calculated on D_3 test sets (in percentage).

	Head	Helmet	Chest	Vest	Hand	Glove
YOLOv4	*96.5*	**98.5**	**95.3**	**98.4**	**83.6**	**65.7**
YOLOv4-Tiny	87.3	*97.7*	91.5	*98.0*	55.9	47.2
SSD MobileNet V2	88.6	96.8	85.5	95.8	49.8	36.1
CenterNet Resnet50 V2	**96.8**	97.1	*94.2*	97.2	63.9	*53.5*
EfficientDet D0	94.3	97.5	90.9	97.4	*64.2*	50.7

TABLE 7. $D_{1,2}$ cross-validation: average percentage of improvement of AP@50 in recognizing each class on D_3 test sets.

	Head	Helmet	Chest	Vest	Hand	Glove
YOLOv4	**+4.8**	+1.0	-1.4	0.0	**+13.5**	**+20.8**
YOLOv4-Tiny	+1.0	+1.0	*+0.4*	-1.5	+1.6	*+17.7*
SSD MobileNet V2	+1.8	**+3.3**	-0.4	-1.1	*+2.8*	+6.2
CenterNet Resnet50 V2	+3.6	*+2.0*	**+1.2**	*+0.1*	+1.3	+9.3
EfficientDet D0	*+4.1*	+1.0	-0.6	**+0.3**	+1.7	+7.6

TABLE 8. $D_{1,2,3}$ cross-validation: average values of AP@50 per class calculated on $D_{1,2,3}$.

	Head	Helmet	Chest	Vest	Hand	Glove
YOLOv4	**96.7**	**98.3**	**95.2**	**88.5**	**80.9**	**65.2**
YOLOv4-Tiny	*92.9*	94.8	*92.2*	*83.2*	57.7	*41.2*
SSD MobileNet V2	78.0	86.2	81.2	74.0	43.3	22.0
CenterNet Resnet50 V2	92.7	*95.3*	91.4	79.3	*63.9*	41.0
EfficientDet D0	83.7	86.4	90.1	82.8	40.6	36.7

gloves. It is worth noticing that YOLOv4-Tiny is the model that mainly improved its performance in recognizing the *glove* class (+17.7%).

3) $D_{1,2,3}$ CROSS-VALIDATION RESULTS

This section discusses the results achieved by performing cross-validation with dataset $D_{1,2,3}$. We recall that this dataset has been created by merging D_1, D_2, and D_3.

As we can see from Table 8, the YOLOv4 network outperforms all the other models in recognizing each class. It is worth noticing that the classes *hand* and *glove* are still the most difficult objects to detect, even for YOLOv4, which achieves average values of AP@50 equal to 80.9% and 65.2%, respectively. As regards the remaining models, these two classes are characterized by an average value of AP@50 not higher than 63.9% (CenterNet Resnet50 V2) and 41.2% (YOLOv4-Tiny), respectively. The low capability of recognizing these two classes may be due to the small size of the objects and to the fact that often the color of the gloves blends with the working clothes. Apart from YOLOv4, the networks with the highest performances are YOLOv4-Tiny and CenterNet Resnet50 V2. The two networks outperform the other models as follows. From one hand, YOLOv4-Tiny works better at identifying *head* (92.9%), *chest* (92.2%), *vest* (83.2%), and *glove* (41.2%). From the other hand, CenterNet Resnet50 V2 is capable of better identifying objects such as *helmets* (95.3%) and *hands* (63.9%).

In general, for all the networks, even those resulting in low performance, their overall performance in object recognition is improved over the results obtained in the D_1 and $D_{1,2}$ cross-validation analyses previously discussed.

Also for this last cross-validation analysis, in Table 9 we report the average results calculated on the test folds of D_3. Results clearly show, once again, that YOLOv4 achieves the best performance for all the classes, but *helmet*. CenterNet Resnet50 V2 and YOLOv4-Tiny are, respectively, the second and the third better models for object recognition. However, the differences between these two networks in recognizing different classes are irrelevant. Moreover, we carried out non-parametric statistical tests, namely the Friedman test followed by a Holm post-hoc procedure [45]. For each model, the tests have been carried out considering a distribution of values composed of the single values of AP@50 calculated on each D_3 fold for each class. Thus, we considered a distribution composed of 30 values for each model. Results of the tests, with a confidence level of 95%, confirmed that: i) YOLOv4 outperforms all the other networks, ii) YOLOv4-Tiny and CenterNet Resnet50 V2 are statistically equivalent and iii) YOLOv4-Tiny and CenterNet Resnet50 V2 outperforms both EfficientDet D0 and SSD MobileNet V2. For the sake of brevity, we omitted all the details of the test results.

As expected, it is worth to notice from Table 9, that for all the networks the average values of AP@50 per class are mostly higher than those shown in Tables 4 and 6. This is because realistic images from D_3 have been included in the training stage. This behaviour is clearly highlighted in Table 10, which shows the percentage of improvement of the averages values of AP@50 in recognizing each class, on D_3 test sets, after adding D_3 to $D_{1,2}$ for creating $D_{1,2,3}$. Once again, Table 10 highlights that introducing D_3 improved the detection of small objects, especially for the hands and the gloves. Also in this case, YOLOv4-Tiny is the model

TABLE 9. $D_{1,2,3}$ cross-validation: average values of *AP*@50 per class calculated on D_3 test sets (in percentage).

	Head	Helmet	Chest	Vest	Hand	Glove
YOLOv4	**98.5**	*98.4*	**98.4**	**99.0**	**89.3**	**88.2**
YOLOv4-Tiny	96.7	**98.6**	*96.2*	98.6	68.2	59.6
SSD MobileNet V2	93.3	96.4	89.2	96.4	51.0	41.7
CenterNet Resnet50 V2	*97.8*	97.4	95.0	*98.9*	*70.4*	*61.2*
EfficientDet D0	95.8	98.3	94.4	98.2	66.6	59.5

TABLE 10. $D_{1,2,3}$ cross-validation: average percentage of improvement of AP@50 in recognizing each class on D_3 test sets.

	Head	Helmet	Chest	Vest	Hand	Glove
YOLOv4	+2.0	-0.1	+3.1	+0.6	+5.7	**+22.5**
YOLOv4-Tiny	**+9.4**	**+0.9**	**+4.7**	+0.6	**+12.3**	*+12.4*
SSD MobileNet V2	*+4.7*	-0.4	*+3.7*	+0.6	+1.2	+5.6
CenterNet Resnet50 V2	+1.0	+0.3	+0.8	**+1.7**	*+6.5*	+7.7
EfficientDet D0	+1.5	*+0.8*	+3.5	*+0.8*	+2.4	+8.8

TABLE 11. Summarized comparison of the DNNs: average values of mAP@50 on the test set.

	cross-validation			on D_3		
	D_1	$D_{1,2}$	$D_{1,2,3}$	D_1	$D_{1,2}$	$D_{1,2,3}$
YOLOv4	**86.7**	**87.0**	**87.5**	**83.2**	**89.6**	**95.3**
YOLOv4-Tiny	75.7	76.3	76.6	76.2	79.6	86.3
SSD MobileNet V2	63.3	63.6	64.1	73.3	75.4	78.0
CenterNet Resnet50 V2	*76.5*	*76.9*	*77.3*	*80.9*	*83.8*	*86.8*
EfficientDet D0	68.7	69.6	70.1	80.2	82.5	85.5

that mostly improved its performance in recognizing the *hand* class (+12.3%) and *glove* class (+12.4%). Moreover, YOLOv4-Tiny improves considerably its performance also on the class *head* (+12.3%) and *chest* (+12.3%). Finally, it is worth noticing that the two YOLO networks, namely YOLOv4 and YOLOv4-Tiny, experienced the highest level of improvement, considering the average results on the test folds of D_3 when passing from D_1 to $D_{1,2,3}$. This improvement is noticeable especially for the *head* (up to +10.4% for YOLOv4-Tiny), *hand* (up to +19.2% for YOLOv4) and the *glove* (up to +43.3% for YOLOv4) classes.

In order to provide a summarized view of the results of our performance comparison between the different DNNs for object recognition, in Table 11, we show the average values of mAP@50 achieved by each model in the three cross-validation analyses. Specifically, in the table, we also show the average values of mAP@50 on the test folds of D_3.

As expected, YOLOv4 is the model characterized by the highest level of accuracy, whereas CenterNet Resnet50 V2 is the model characterized by the second highest level of accuracy. As regards YOLOv4-Tiny, we have to highlight that its performances are not far from those achieved by CenterNet Resnet50 V2, especially if we consider the average values of mAP@50 on D_3. In particular, we recall that the results of the non-parametric statistical tests carried for the $D_{1,2,3}$ cross-validation analysis considering the results achieved on D_3, confirms the statistical equivalence of YOLOv4-Tiny and CenterNet Resnet50 V2.

D. LATENCY ANALYSIS OF THE OBJECT DETECTION TASK

This section shows the results of the latency analysis we carried out for each of the five DNNs considered in our study. Specifically, this analysis aims to evaluate the inference time

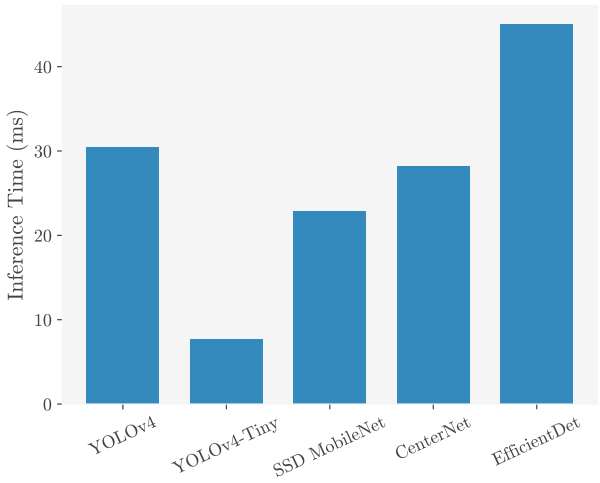

FIGURE 10. Average inference time for object recognition on a single image.

associated with each model, namely the time required to output the bounding boxes around detected objects. Indeed, this time is particularly important when such models for object detection are adopted in real-time contexts. This analysis has been carried out on the server described in SectionV-A, exploiting the TensorFlow Object Detection API.

Figure 10 shows a comparison of the average inference time, expressed in milliseconds (ms) per image. The average values have been calculated considering the time for performing the object detection task on a single image. We have run inference on 1458 images to each fine-tuned model, and the image presentation process has been repeated 30 times.

In our results, we also calculated the 95% confidence intervals, which are omitted in the figures since they are

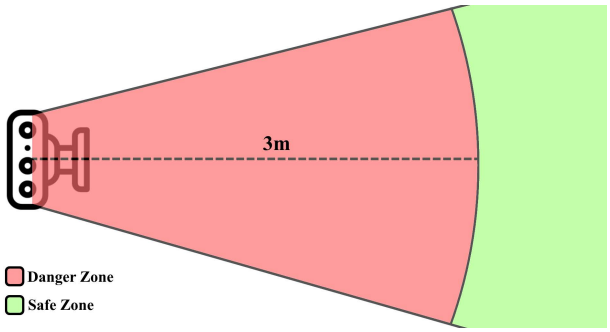

FIGURE 11. Schema of the areas to be monitored, in the danger zone in red where there is the obligation to wear PPE instead in the non-dangerous zone in green.

negligible. For each of the 30 repetitions, we ignore the inference times of the first 100 images, thus making sure that each model was at its steady state. Indeed, there may be some initial computations that could impact the average value.

As we can see from Figure 10, YOLOv4-Tiny is the fastest model in the object detection task, whereas EfficientDet D0 is the slowest one. Specifically, the average inference times per image associated with these two networks are 7.6ms and 45ms, respectively. As regards the other models, their average inference times are equal to 22.8ms, 28.2ms and 30.6ms for SSD MobileNet V2, CenterNet Resnet50 V2 and YOLOv4, respectively.

It is worth noting that, even though YOLOv4 is always the best performing network in the object recognition task, its average inference time is more than three times higher than that of its simplified version, namely YOLOv4-Tiny. Moreover, EfficientDet D0 is six times slower than YOLOv4-Tiny, but both the models are characterized approximately by the same level of performance in the object recognition task.

VI. FINAL DEPLOYMENT OF THE DNNs IN THE PPE DETECTION SYSTEM

In this section, we briefly report the performance obtained by the models when deployed on our testbed for PPE detection. Even though the final version of the system is equipped with the YOLOv4-Tiny model as it showed the best results, we also tested the other models deploying them on our system for the sake of comparison. To this aim, all the models have been converted their OpenVINO optimized representation (i.e., IR format).

In order to evaluate the capability of each model in carrying out the real-time detection of the PPEs, we created a 3 minutes demo video in our laboratory. The video includes the presence of workers randomly wearing and removing the three PPEs considered in this study. Specifically, we recorded the video in the area schematized in Figure 11. The red zone of the area may be considered a typical "dangerous zone" in an industrial plant.

The video is then exploited to run experiments on our testbed with the same conditions. At each experiment, the video is loaded into the memory of the Raspberry PI node and elaborated by the PPE detection application, which implements one of the considered models. For each model, the experiment is run five times, and in each one, we measured the average values of the Frame-Per-Second (FPS) metric, defined as the number of video frames that the whole system can analyze in one second.

TABLE 12. Average FPS values calculated for each DNNs deployed on the real system for the PPE detection task.

	FPS
YOLOv4	0.7
YOLOv4-Tiny	**6.8**
SSD MobileNet V2	6.3
CenterNet Resnet50 V2	0.9
EfficientDet D0	1.0

TABLE 13. Differences in percentage of mAP@50 and average FPS between YOLOv4-Tiny and the other analyzed CNN networks. Models are trained in cross-validation with $D_{1,2,3}$ and tested on D_3.

	mAP@50	FPS
YOLOv4	+10.4%	-89.7%
SSD MobileNet V2	-9,6%	-7.4%
CenterNet Resnet50 V2	+0.6%	-86.8%
EfficientDet D0	-0.9%	-85.3%

Table 12 shows, for each deployed model, the average value of FPS achieved by the PPE detection system. As expected, almost in line with the results on latency discussed in Section V-D, the PPE detection system equipped YOLOv4-Tiny network is associated with the highest value of FPS, whereas the system with YOLOv4 network is the slowest one. Even though SSD MobileNet V2 ensures a good level of analyzed FPS (higher than YOLOv4-Tiny), this network is characterized by a low level of performance in recognizing the different classes. As regards EfficientDet D0, it is worth noticing that, when deployed on our prototype, the system can process just one frame per second (almost 7 times less than the system equipped with YOLOv4-Tiny, which is characterized by the same level of performance than EfficientDet D0 in recognizing PPEs).

Based on the analysis we carried out before, we decided to implement in our PPE detection system prototype the YOLOv4-Tiny network, appropriately fine-tuned using all the images contained in $D_{1,2,3}$. Table 13 show an overview of the differences in percentage between the mAP@50 and the average FPS associated to the YOLOv4-Tiny network and each of the remaining networks.

Even though YOLOv4 is more accurate (up to +10.4% mAP@50 and up to +20% accuracy in recognizing the *hand* and *glove* classes with respect to YOLOv4-Tiny, see Table 9), the system equipped with YOLOv4-Tiny is able to recognize the remaining classes with a very high level of accuracy with the highest value of FPS. However, if the PPE detection application requires giving higher priority to accuracy in recognizing small objects rather than low latency, YOLOv4 can be used instead of YOLOv4-Tiny.

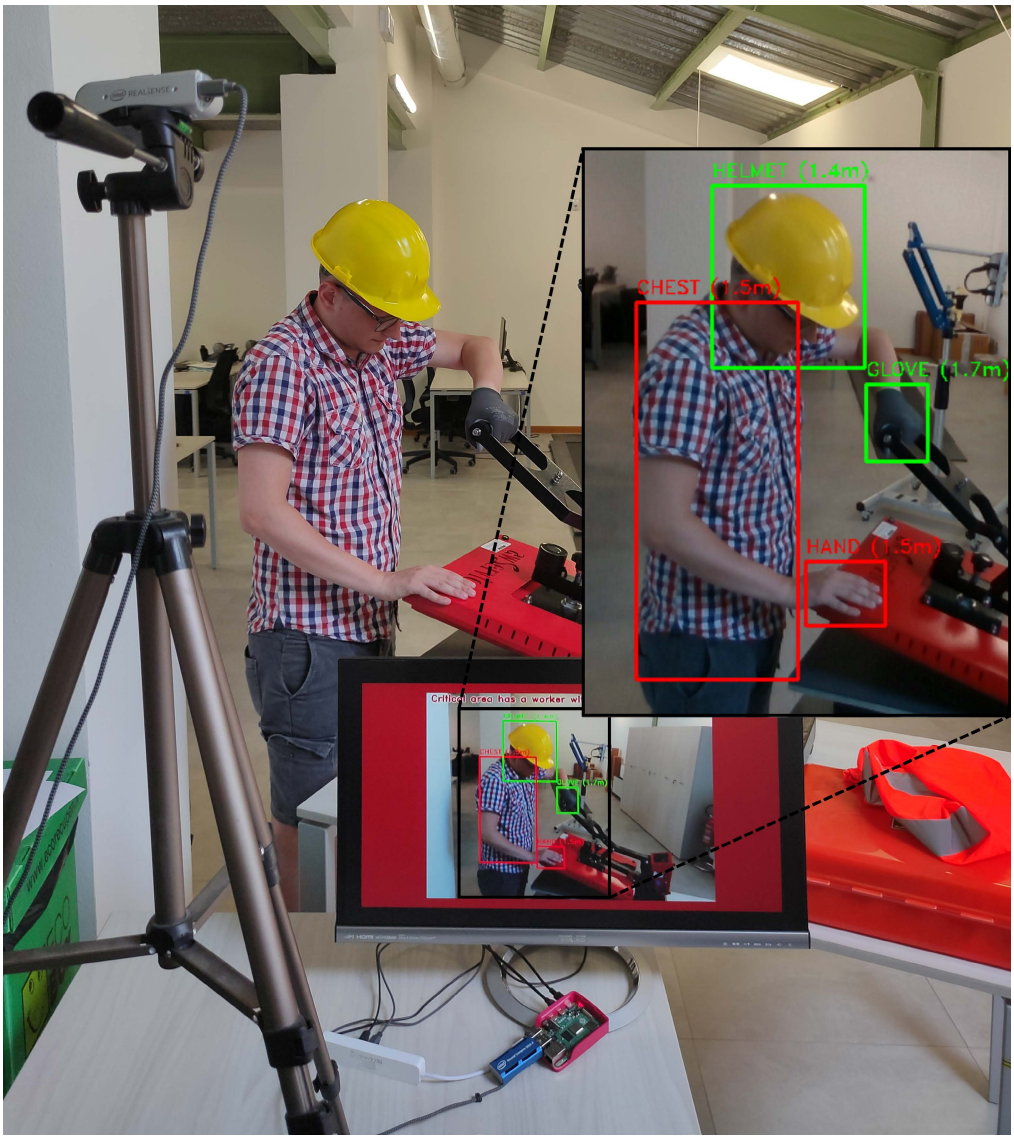

FIGURE 12. The PPE detection system: an example of usage in a dangerous scenario of managing a press.

Figure 12 shows an example of a real working scenario staged in our laboratory. In the example, a worker is operating a press: he is correctly wearing the helmet and only one glove, while he is not wearing a vest and the glove on the other hand. As can be seen, the system correctly identifies all the classes.

VII. CONCLUSION

This paper presents a system for real-time PPE detection based on video streaming analysis and deep learning. Based on the edge computing model, the system analyzes the images from a dangerous area in real-time on an embedded system placed to identify if workers wear or not protective equipment, e.g., helmet, vest, and gloves in our case.

In order to select the most suitable object detector to implement our system, different DNNs have been analyzed and compared. Specifically, in our intensive experimental campaign, we have exploited three different datasets with realistic images for carrying out different cross-validation analyses. The results of our analysis have shown that YOLOv4 is the model characterized by the highest level of classification performance, although with a small reduction w.r.t. gloves detection, due to their small size in the image. The simplified version of the model, namely the YOLOv4-Tiny, instead, results in slightly lower classification accuracy (it loses around 10.4% in mAP@50 to YOLOv4).

In order to assess the capability of our system to work in real-time in a real industrial scenario, we have also carried out a set of experiments deploying all the models on the system itself. We have measured the number of frames per second analyzed by the system for each model. The latter experiments have highlighted that the model resulting in the highest speed in image elaboration, in terms of frames per second, is YOLOv4-Tiny. Indeed, this model is associated with the highest values of FPS, and the most accurate model, namely YOLOv4, loses around 90% to YOLOv4-Tiny in terms of throughput.

The results of our experimental analyses have shown that different trade-offs between the accuracy of the DNNs and their capability of working in real-time can be achieved when comparing the different models. Since we have decided to give priority to the speed in detecting if workers are wearing or not the required PPEs, our final implementation included the YOLOv4-Tiny model. However, if the requirement of having a higher level of PPEs recognition will have higher importance than the speed in elaborating the video streams, more accurate DNNs than YOLOv4-Tiny may be deployed on our system.

REFERENCES

[1] (2018). *Eurostat Accidents at Work Statistics*. [Online]. Available: https://ec.europa.eu/eurostat/statistics-explained/index.php/Accidents_at_work_statistics#Analysis_by_activity

[2] R. A. Haslam, S. A. Hide, A. G. F. Gibb, D. E. Gyi, T. Pavitt, S. Atkinson, and A. R. Duff, "Contributing factors in construction accidents," *Appl. Ergonom.*, vol. 36, no. 4, pp. 401–415, 2005.

[3] S. Rowlinson, *Construction Safety Management Systems*. Evanston, IL, USA: Routledge, 2004.

[4] N. Cavazza and A. Serpe, "Effects of safety climate on safety norm violations: Exploring the mediating role of attitudinal ambivalence toward personal protective equipment," *J. Saf. Res.*, vol. 40, no. 4, pp. 277–283, Aug. 2009.

[5] J. Wu, N. Liu, C. Geyer, and J. M. Rehg, "C^4: A real-time object detection framework," *IEEE Trans. Image Process.*, vol. 22, no. 10, pp. 4096–4107, Jun. 2013.

[6] S. Y. Nikouei, Y. Chen, and T. R. Faughnan, "Smart surveillance as an edge service for real-time human detection and tracking," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2018, pp. 336–337.

[7] B. Balakreshnan, G. Richards, G. Nanda, H. Mao, R. Athinarayanan, and J. Zaccaria, "PPE compliance detection using artificial intelligence in learning factories," *Proc. Manuf.*, vol. 45, pp. 277–282, Jan. 2020.

[8] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, "Edge computing: A survey," *Future Gener. Comput. Syst.*, vol. 97, pp. 219–235, Aug. 2019.

[9] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," Tech. Rep., 2020.

[10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[11] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "CenterNet: Keypoint triplets for object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6569–6578.

[12] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10781–10790.

[13] G. Gallo, F. Di Rienzo, P. Ducange, V. Ferrari, A. Tognetti, and C. Vallati, "A smart system for personal protective equipment detection in industrial environments based on deep learning," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, Aug. 2021, pp. 222–227.

[14] X. Luo, H. Li, H. Wang, Z. Wu, F. Dai, and D. Cao, "Vision-based detection and visualization of dynamic workspaces," *Autom. Construct.*, vol. 104, pp. 1–13, Aug. 2019.

[15] B. E. Mneymneh, M. Abbas, and H. Khoury, "Vision-based framework for intelligent monitoring of hardhat wearing on construction sites," *J. Comput. Civil Eng.*, vol. 33, no. 2, Mar. 2019, Art. no. 04018066.

[16] J. Yang, S. Li, Z. Wang, and G. Yang, "Real-time tiny part defect detection system in manufacturing using deep learning," *IEEE Access*, vol. 7, pp. 89278–89291, 2019.

[17] Q. Fang, H. Li, X. Luo, L. Ding, H. Luo, and C. Li, "Computer vision aided inspection on falling prevention measures for steeplejacks in an aerial environment," *Autom. Construct.*, vol. 93, pp. 148–164, Sep. 2018.

[18] W. Zhiqiang and L. Jun, "A review of object detection based on convolutional neural network," in *Proc. 36th Chin. Control Conf. (CCC)*, Jul. 2017, pp. 11104–11109.

[19] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.

[20] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448.

[21] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[22] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2016, pp. 21–37.

[23] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[24] M. Wang, P. Wong, H. Luo, S. Kumar, V. Delhi, and J. Cheng, "Predicting safety hazards among construction workers and equipment using computer vision and deep learning techniques," in *Proc. Int. Symp. Autom. Robot. Construct. (ISARC)*, vol. 36, 2019, pp. 399–406.

[25] W. Zhang, C.-F. Yang, F. Jiang, X.-Z. Gao, and X. Zhang, "Safety helmet wearing detection based on image processing and deep learning," in *Proc. Int. Conf. Commun., Inf. Syst. Comput. Eng. (CISCE)*, Jul. 2020, pp. 343–347.

[26] Z. Fan, C. Peng, L. Dai, F. Cao, J. Qi, and W. Hua, "A deep learning-based ensemble method for helmet-wearing detection," *PeerJ Comput. Sci.*, vol. 6, p. e311, Dec. 2020.

[27] Z. Wang, Y. Wu, L. Yang, A. Thirunavukarasu, C. Evison, and Y. Zhao, "Fast personal protective equipment detection for real construction sites using deep learning approaches," *Sensors*, vol. 21, no. 10, p. 3478, May 2021.

[28] N. D. Nath, A. H. Behzadan, and S. G. Paal, "Deep learning for site safety: Real-time detection of personal protective equipment," *Autom. Construct.*, vol. 112, Apr. 2020, Art. no. 103085.

[29] G. Iannizzotto, L. Lo Bello, and G. Patti, "Personal protection equipment detection system for embedded devices based on DNN and fuzzy logic," *Expert Syst. Appl.*, vol. 184, Dec. 2021, Art. no. 115447.

[30] S. Chen and K. Demachi, "Towards on-site hazards identification of improper use of personal protective equipment using deep learning-based geometric relationships and hierarchical scene graph," *Autom. Construct.*, vol. 125, May 2021, Art. no. 103619.

[31] R. Xiong and P. Tang, "Pose guided anchoring for detecting proper use of personal protective equipment," *Autom. Construct.*, vol. 130, Oct. 2021, Art. no. 103828.

[32] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.

[33] I. Rodriguez-Conde, C. Campos, and F. Fdez-Riverola, "Optimized convolutional neural network architectures for efficient on-device vision-based object detection," *Neural Comput. Appl.*, vol. 34, no. 13, pp. 10469–10501, Jul. 2022.

[34] C.-C. Hung, G. Ananthanarayanan, P. Bodik, L. Golubchik, M. Yu, P. Bahl, and M. Philipose, "VideoEdge: Processing camera streams using hierarchical clusters," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2018, pp. 115–131.

[35] L. Ciampi, C. Gennaro, F. Carrara, F. Falchi, C. Vairo, and G. Amato, "Multi-camera vehicle counting using edge-AI," *Expert Syst. Appl.*, vol. 207, Nov. 2022, Art. no. 117929.

[36] R. Rajavel, S. K. Ravichandran, K. Harimoorthy, P. Nagappan, and K. R. Gobichettipalayam, "IoT-based smart healthcare video surveillance system using edge computing," *J. Ambient Intell. Hum. Comput.*, vol. 13, no. 6, pp. 3195–3207, Jun. 2022.

[37] S. Liu, L. Liu, J. Tang, B. Yu, and W. Shi, "Edge computing for autonomous driving: Opportunities and challenges," *Proc. IEEE*, vol. 107, no. 8, pp. 1697–1716, Jun. 2019.

[38] A. Ndikumana, N. H. Tran, D. H. Kim, K. T. Kim, and C. S. Hong, "Deep learning based caching for self-driving cars in multi-access edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 5, pp. 2862–2877, May 2020.

[39] A. Demidovskij, Y. Gorbachev, M. Fedorov, I. Slavutin, A. Tugarev, M. Fatekhov, and Y. Tarkan, "OpenVINO deep learning workbench: Comprehensive analysis and tuning of neural networks inference," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 783–787.

[40] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *Proc. Int. Conf. Artif. neural Netw.* Springer, 2018, pp. 270–279.
[41] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, 2014, pp. 1717–1724.
[42] K. He, R. Girshick, and P. Dollar, "Rethinking ImageNet pre-training," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 4918–4927.
[43] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.
[44] R. Padilla, S. L. Netto, and E. A. B. da Silva, "A survey on performance metrics for object-detection algorithms," in *Proc. Int. Conf. Syst., Signals Image Process. (IWSSIP)*, Jul. 2020, pp. 237–242.
[45] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, Mar. 2011.

GIONATAN GALLO received the bachelor's and master's degrees in computer engineering from the University of Pisa, in 2015 and 2019, respectively, where he is currently pursuing the Ph.D. degree in information engineering. His research interests include artificial intelligence (AI) methods and techniques applied to industrial (industry 4.0) and biomedical domains with particular attention to explainable (XAI) and deep AI.

FRANCESCO DI RIENZO received the bachelor's and master's degrees in computer engineering from the University of Pisa, in 2015 and 2019, respectively, where he is currently pursuing the Ph.D. degree in smart industry in cooperation with the University of Florence and the University of Siena. He has published a couple of papers in journals and contributed to international conferences. His research interests include wearable sensor, the IoT, near field communication, and fog computing. From 2015 to 2018, he was a member of the "E-Team Squadra Corse" at the University of Pisa as an Information Technology Manager. He is also serving as the WEB Co-Chair for IEEE SMARTCOMP 2020.

FEDERICO GARZELLI received the bachelor's and master's degrees from the University of Pisa, in 2018 and 2021, respectively. He is currently working as a Cloud Engineer in an IT consulting company. His research interests include deep learning applied to computer vision, cloud computing, DevOps, and site reliability engineering practices.

PIETRO DUCANGE received the M.Sc. degree in computer engineering and the Ph.D. degree in information engineering from the University of Pisa, Italy, in 2005 and 2009, respectively. He is currently an Associate Professor of information systems and technologies at the University of Pisa. He has been involved in a number of research and development projects in which data mining and computation intelligence algorithms have been successfully employed. He has coauthored over 80 papers in international journals and conference proceedings. His main research interests include explainable artificial intelligence, big data mining, social sensing, and sentiment analysis. He is a member of the Editorial Board of *Soft Computing* journal.

CARLO VALLATI (Member, IEEE) received the master's *(magna cum laude)* and Ph.D. degrees in computer systems engineering from the University of Pisa, in 2008 and 2012, respectively. In 2010, he visited the Computer Science Department, University of California at Davis. He is currently an Associate Professor at the Department of Information Engineering, University of Pisa. He has been involved in the project BETaaS, Building the Environment for the Things as a Service, funded by the European Union under the 7th Framework Program and in several research projects supported by private industries. He is the coauthor of more than 60 peer-reviewed papers in international journals and conference proceedings. He has served as a program committee member for more than 30 international conferences and workshops and as a Workshop Chair for the IEEE IoT-SoS and IEEE SmartSys Workshops. He is also serving as a TPC Co-Chair for IEEE SMARTCOMP 2020 and on the Editorial Board of two international journals, the *Journal of Reliable Intelligent Environments* (Springer) and *Applied Sciences* (MDPI). He is the Co-ordinator of the Cloud Computing, Big Data and Cybersecurity Crosslaboratory founded in the framework of the Departments of Excellence ("Dipartimenti di Eccellenza") funded by the Italian Ministry of Education, University and Research ("Ministero dell'Istruzione dell'Università e della Ricerca").