machines

MDPI

Article

Deep-Learning-Based Cyber-Physical System Framework for Real-Time Industrial Operations

Vatsal Maru *, Saideep Nannapaneni, Krishna Krishnan and Ali Arishi

Department of Industrial, Systems and Manufacturing Engineering, Wichita State University, 120 Engineering Building, 1845 Fairmount St., Box 35, Wichita, KS 67260-0035, USA
* Correspondence: vkmaru@shockers.wichita.edu

Abstract: Automation in the industry can improve production efficiency and human safety when performing complex and hazardous tasks. This paper presented an intelligent cyber-physical system framework incorporating image processing and deep-learning techniques to facilitate real-time operations. A convolutional neural network (CNN) is one of the most widely used deep-learning techniques for image processing and object detection analysis. This paper used a variant of a CNN known as the faster R-CNN (R stands for the region proposals) for improved efficiency in object detection and real-time control analysis. The control action related to the detected object is exchanged with the actuation system within the cyber-physical system using a real-time data exchange (RTDE) protocol. We demonstrated the proposed intelligent CPS framework to perform object detection-based pick-and-place operations in real time as they are one of the most widely performed operations in quality control and industrial systems. The CPS consists of a camera system that is used for object detection, and the results are transmitted to a universal robot (UR5), which then picks the object and places it in the right location. Latency in communication is an important factor that can impact the quality of real-time operations. This paper discussed a Bayesian approach for uncertainty quantification of latency through the sampling–resampling approach, which can later be used to design a reliable communication framework for real-time operations.

Keywords: cyber-physical system; deep learning; robotics; real time; industrial operations

Citation: Maru, V.; Nannapaneni, S.; Krishnan, K.; Arishi, A. Deep-Learning-Based Cyber-Physical System Framework for Real-Time Industrial Operations. *Machines* **2022**, *10*, 1001. https://doi.org/10.3390/machines10111001

Academic Editors: Shuai Li, Dechao Chen, Mohammed Aquil Mirza, Vasilios N. Katsikis, Dunhui Xiao and Predrag S. Stanimirovic

Received: 20 September 2022
Accepted: 23 October 2022
Published: 31 October 2022

1. Introduction

The fourth industrial revolution (Industry 4.0), which refers to the increased usage of Internet-based applications and its digitization of processes in the industry [1], has reorganized the control of the product life cycle [2]. This Internet-based digitization has provided the opportunity for these applications to be implemented in real time [3] and also to be self-learning [4]. The real-time and self-learning applications can connect different fragments of the industry and improve overall functionalities in various stages of a product life cycle [1]. In addition to increasing the efficiency in industrial operations, it is also important to reduce wastage leading to sustainable operations. Sustainability is recognized as a core factor for businesses by the United Nations (UN) Sustainability 2030 agenda [5].

For these reasons, the techniques emerging from Industry 4.0, such as the Internet of Things (IoT) and cyber-physical systems (CPSs), offer key contributions to the sustainability of businesses [6]. Cyber-physical systems refer to the integration of computing and physical systems to satisfy desired functional operations. The Internet of Things refers to the interconnection of several entities (both cyber and physical) over the Web that facilitates the transmission of information across those entities, thereby enabling automation. Ref. [7] developed an evaluation method to understand the impacts of Industry 4.0 on sustainability and concluded that Industry 4.0 improved the sustainability dynamics of the industry. They used UN sustainable development goals as metrics to derive a conclusion in their study.

The IoT and CPS combined have a promise for introducing improved operations (both in terms of efficiency and sustainability) in a variety of domains such as smart

production systems, smart transportation, and infrastructure systems [8]. In this paper, we used a CPS to develop an intelligent framework to perform real-time operations in the physical world [9,10]. The reason for using a CPS over IoT is more from the perspective of reduction of cost. There are various industrial applications of this framework, and a primary application is in quality control (identifying faulty parts moving on a conveyor belt using machine vision and a robotic arm).

A cyber-physical system can be broadly decomposed into four components—a physical component, a cyber component, sensors, and an actuation component [11]. The physical component refers to a physical entity that is being monitored and controlled by the cyber component, which can be a single computing node or a collection of interconnected computing nodes. Sensors collect data regarding the physical component (e.g., operational characteristics), and the actuation component implements the appropriate control action on the physical component as determined by the cyber component. According to [12], the idea behind the use of IoT and CPS was the reduction in human interaction in data gathering and processing and proposed the increased ability of cyber systems in data gathering and communication among each other using the Web. In other words, different components of a CPS operate in synchronization to achieve similar objectives with the assistance of data transmission over the Web [13].

The three components of the CPS are selected for the paper in such a way that they are capable of intelligent operations [14]. There is a wide range of actuation components that can be included while developing a CPS [15,16]. In this paper, we used a robotic system (specifically a collaborative robotic arm) as the primary actuation component. Robotic systems have increasingly been used in the industry to perform repetitive tasks for increased operational efficiency as repetitive manual operations can lead to operator fatigue and can result in accidents.

Robots can perform a task on a repetitive basis over a specified period and can produce similar outputs. Hence, robotic systems possess better repeatability when compared against manual operations [17]. Therefore, robots working on repetitive tasks help in improving the accuracy and precision of products, reduce defects, reduce human fatigue, and increase workplace safety overall. Moreover, there could be variation across several human operators when performing the same task [18,19]. Human fatigue varies in different industries and depends upon the tasks. For example, a pick-and-place operation of a plastic toy and an airplane wing is a significantly different operation. Human operators can easily perform a pick-and-place operation of plastic toys; however, a pick-and-place operation of an airplane wing increases safety concerns, as it is heavier to lift. Nevertheless, plastic toys' pick-and-place operation will cause human fatigue when performed for long working periods. Due to weight and time considerations, robots have become a go-to solution for industrial operations, which is one of the motivations for this paper.

However, one of the challenges of the manufacturing industry is variability. For example, an automobile manufacturing plant will require the assembling of multiple different models of an automobile. Different products and models suggest variability in manufacturing operations. This variability will require highly automated plant operations using robots to change robot codes. The amount of hard coding required to program a robot for a task is time-consuming and expensive. In the IoT's case, as the definition suggests, the amount of devices and sensors that are connected to the Internet might increase the inherent cost. These additional time delays and cost increment negatively affect the plant's ability to reduce costs and efficiently operate. Therefore, robots that possess intelligence and that can handle variability are an important solution to the variability in industrial operations [20].

Intelligent robotic systems can have a huge impact on the manufacturing industry. Zhong et al. (2017) [21] stated that automation in manufacturing needs new technological concepts to make it intelligent, which can be applied as a part of the design, production, and management. The goal of this paper was to create an intelligent CPS framework, which functions over the Web, performs real-time object detection, and implements appropriate

control actions based on the detected objects. We detailed the proposed framework and demonstrated it with real-world experiments using a universal robot (UR5) for pick-and-place operations based on object detection. There are three main components of the framework, namely object detection by a computer, a robot operating based on the gathered and processed data, and updating the computer's memory. The entire communication is automated on the Internet by using TCP/IP standards between these three components of the framework. The resulting system from integrating these framework components requires certain intelligence. To be intelligent, the artificial (cyber and physical) systems require extracting patterns from the available data in such a way that they can be used as information.

Advanced intelligent systems extract patterns from the available data in such a way that they can be used as information. Several machine-learning techniques have been adopted in the literature such as linear regression, tree-based models (decision trees and random forests), support vector machines, artificial neural networks (ANNs) to perform different types of analyses such as regression, classification, and pattern recognition [22,23]. Depending on the type of available data, some types of machine-learning models are preferred to other types of models. In the case of image data, ANN-based techniques are predominantly used for image processing, object detection, and pattern recognition [23–25]. The recent advances in techniques based deep learning (DL) such as convolutional neural networks (CNNs) have resulted in more sophisticated and accurate models for classification.

In this paper, we demonstrated the proposed framework using image data obtained through machine vision, and therefore, CNN-based methods were used. This paper extended the working paper content of [26] and provided an applied approach of the framework with results and latency considerations. A review of several ANN- and CNN-based methods for image processing is later discussed in Section 2.2. The proposed algorithm for the CPS framework operates on real-time data gathering and processing with detection in a time frame of 0.2 s per image [27]. The proposed CPS works on real-time data exchange (RTDE) [28], which is operating on the Internet using TCP/IP standards. The proposed framework improves the operator's safety and simplifies the operational process in general by keeping the control in the hands of an operator.

Section 3 details a framework of concepts that are offered in the literature and defined by its innovators and provides a conceptual CPS framework for object detection using machine vision and performing appropriate real-time control. Since the proposed framework relies on communication between various entities over the Web, it is also essential to understand and quantify the latency in communication and also in analysis. The quantification of latency enables us to design reliable industrial control systems. The latency is not constant but a variable value. For example, the communication time between the same entities for repetition of the same tasks can be different.

This paper considered two types of latency—latency in the cyber system and latency in the networking element for data gathering. The latency in a cyber system can be due to the system's computational capabilities (hardware capabilities). Moreover, the latency in the cyber system can also be due to the deep-learning models that are being implemented on them. The required computational capabilities increase with the complexity of the deep-learning models (such as the number of layers). As the term DL itself suggests the depth of the models, these multilayered models require higher computational capabilities. Ref. [28] reviewed that the DL model latency varies based on the cyber system used. That is even more crucial depending on the application. For example, in the case of self-driving cars, the DL models require to operate with low latency as the traffic situations rapidly change in small time frames. Due to this dynamic environment, latency is a crucial factor while developing a model for these applications. Ref. [28] proposed a modeling design approach for CNNs on FPGAs considering latency. They claimed big improvements because of latency-driven optimization.

Another important consideration for a CPS is latency in data gathering or due to the networking element. Ref. [29] suggested that most system elements are focused on data integrity, connection, and throughput without a temporal mechanism. The temporal semantics are important when an application system requires a higher degree of control. They provided models and methodology for these applications suggesting suitable solutions. Ref. [30] proposed an incremental model-based approach for the temporal analysis of a CPS. Their research focused on decreasing the reengineering aspect of a CPS when a high-level latency specification can be performed. This helps in reducing costs over multiple media. Ref. [31] studied the architecture analysis and design language (AADL) for a CPS, which is a modeling language for CPS. The AADL is widely used in avionics and space programs. Their focus was on flow latency in a model design of the CPS model. Their case study was on a simulation. However, they claimed to have refined model performances for the driverless aircraft with improved results on achieving the given objectives.

A few other notable research papers include probabilistic end-to-end path latency in a wireless sensor network [32]. Ref. [32] proposed a model of providing an end-to-end estimation of distribution functions from the system's current state to intermediate states by decreasing the computational requirements. One other emerging area where the networking medium is used and has high usage, however peculiar for the research of a CPS, is online cloud-based gaming. Ref. [33] studied the effects of latency on player performance in cloud-based games. Therefore, this paper used a model-based approach to predict the communication time; the parameters of the model are estimated through Bayesian inference using the sampling–resampling algorithm [34]. In this way, a probabilistic prediction (prediction and also the uncertainty associated with it) of the communication time can be obtained, which can later be used for designing and analyzing real-time industrial cyber-physical systems.

The key contributions of this paper are as follows: (1) an intelligent cyber-physical system framework operating over the Internet and (2) a Bayesian model-based approach for a probabilistic prediction of the latency in the proposed cyber-physical system framework. The proposed methods are illustrated using real-world experiments of pick-and-place operations using a collaborative robotic arm and machine vision system.

The rest of the paper is organized as follows. Section 2 provides a brief review about robot programming, convolutional neural networks and its variants, and Bayesian inference through the sampling–resampling approach. Section 3 discusses the proposed framework along with real-world experimentation using a UR5 robot for pick-and-place operations. Section 4 discusses the estimation of latency in the CPS framework followed by concluding remarks in Section 5.

2. Background

The objective of the paper was to create an intelligent CPS that implements different types of physical control actions [35,36]. We consider a classic pick-and-place operation based on the detected objects using a vision system. For implementation, a conventional robot operation, pick and place of parts is considered. The motivation behind the objective is to improve the operator's safety and simplify the operational process on the production floor. There is a research scope in the existing literature for improvement using a deep-learning-based CPS for better production fit [37–40].

Some background information on the terminologies and concepts used in the experiment is explained in this section. The following subsections provide a comprehensive review of each of the methods, which will be used in the methodology section. Furthermore, there is a literature review provided for the networking element and protocol standards as well.

2.1. Robot Programming

The robot used for the research is a UR5 robot from Universal Robotics. This is a six-degree-of-freedom (DoF) robot with a payload capacity of 5 Kgs. The robot can be pro-

grammed using its software called Polyscope or by using SCRIPT Language or Python [28]. Collaborative robot arms have applications in different industrial operations such as pick-and-place, assembly, welding, quality inspection, and additive manufacturing [41–43].

Robot programming can be categorized into two types: (a) online programming and (b) offline programming [44–46]. Online programming is when the robot is programmed while functioning, and offline programming is when the robot is simulated in the programming software before deploying the real program to the robot [45]. Offline programming is a helpful way of avoiding malfunctions and dangerous incidents, which a programmer may not have anticipated while building the solution approach [46]. However, offline programming adds an extra step of simulation and testing the programs, which will then be extended as online programming [19].

The usual robot-based manufacturing systems consider minimal variability in the work environment. To counter any variability, manufacturing systems (e.g., robots) contain sensors that are responsible for detecting variable actions in an environment. The use of sensors in a system increases the cost of automation, which would be the case in implementing the IoT. Therefore, to increase the speed of control actions (by directly deploying the control actions via online programming) and decrease the costs (by decreasing the use of sensors), the use of the Internet and DL processes is a welcome progress.

The Tensorflow and Keras toolboxes [47,48] available in the Python programming language were used to implement the object detection and control algorithm (ODCA); this is later used in Section 4. The Python code creates a socket communication to send SCRIPT codes to control the motion of the robot as well as receive joint data from the robot. For object detection, a webcam was used for collecting and processing image data.

2.2. Neural Networks

An artificial neural network (ANN) is one of the most widely used machine-learning models for image processing and object detection [24,49]. Various deep-learning models have been developed based on the ANN architecture. One of the widely used deep-learning models is a convolutional neural network (CNN). A CNN is an artificial neural network (ANN) paradigm for object detection where the detected objects are used as input to the network [49]. Several variants of CNNs have also been developed to improve the efficiency of model training. A brief review of CNNs and their variants such as R-CNN [50], fast R-CNNs [51], and faster R-CNNs [27] are given below.

2.2.1. Convolutional Neural Networks (CNNs)

In a CNN, there are convolutional and pooling layers to extract data in metric form, which then are fed to the fully connected (FC) layers as shown in Figure 1. The FC layers, in the end, classify the data in the classified output [52].

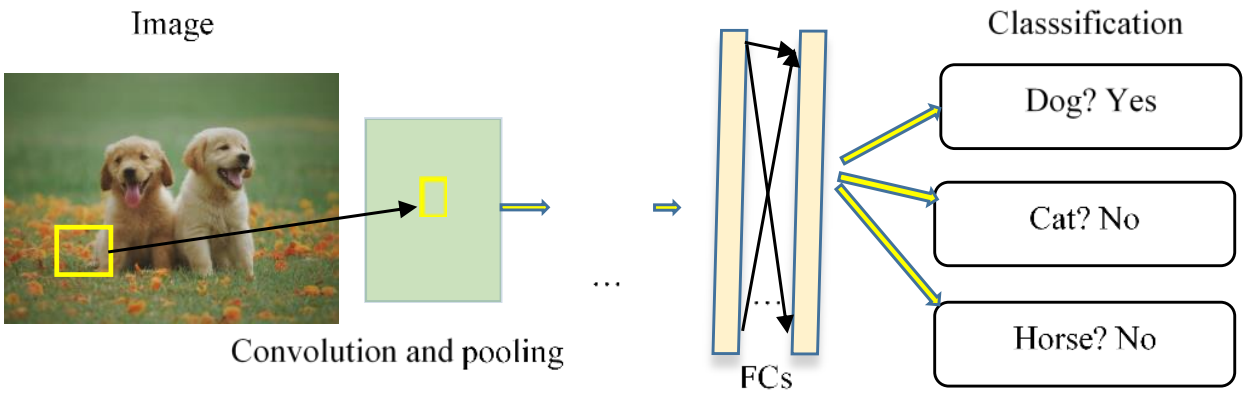

Figure 1. CNN architecture showing convolution layers and fully connected layers for image classification.

2.2.2. Proposed Region-Convolutional Neural Networks (R-CNNs)

To improve the computational speed and establish real-time object detection, a variation of a CNN called the R-CNN was developed, where R stands for the region proposals (or proposed region). Proposed regions are the regions that have a higher probability of containing an object within the image. Figure 2 below shows that CNNs are computed for the proposed regions instead of classifying all the pixels in the image [50,53]. Ref. [53] recorded 50 s per image for image classification.

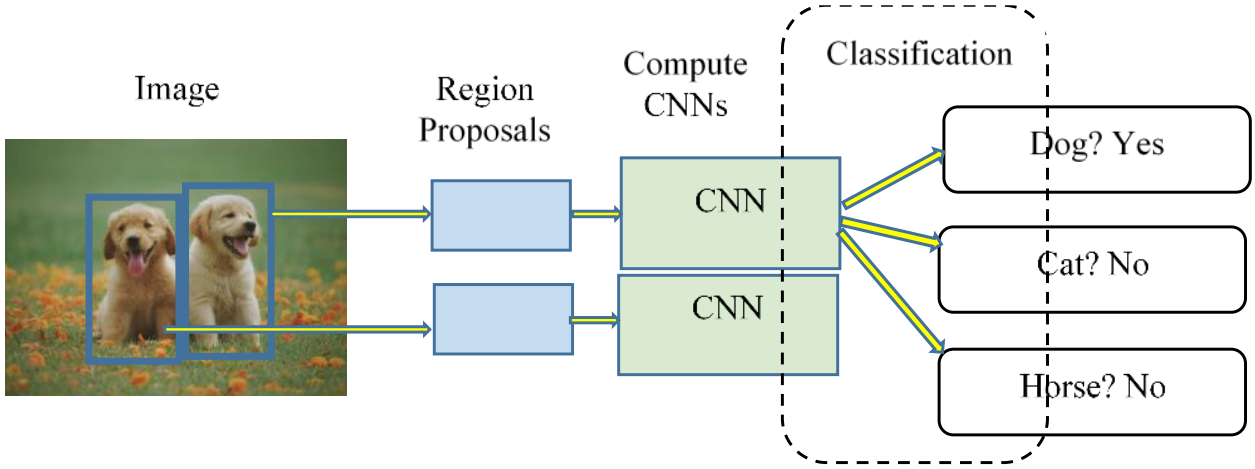

Figure 2. R-CNN architecture where CNNs are computed only for proposed regions.

2.2.3. Fast Proposed Region-Convolutional Neural Networks (Fast R-CNNs)

Ref. [51] proposed another development in the R-CNN architecture to improve the runtime of the algorithm. In a fast R-CNN, only one CNN is computed based on the region of interest (RoI). Then the RoI pooling layers feed the data to the FC layers, and the classification is performed as shown in Figure 3. The runtime for the fast R-CNN was 2 s per image [51].

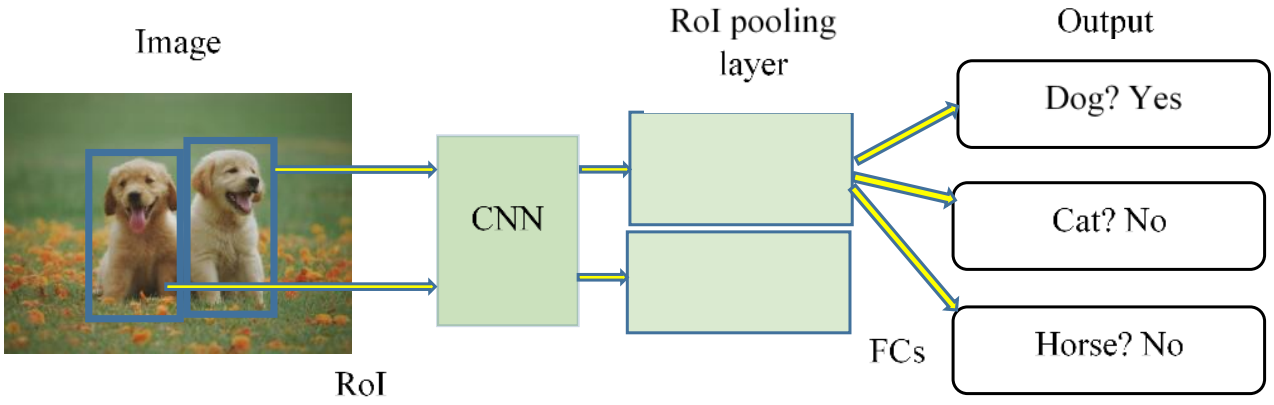

Figure 3. Fast R-CNN architecture where CNN is computed only for the proposed regions followed with similar FC layers classifying in the end. CNN is computed only for proposed regions followed by similar FC layers classifying in the end.

2.2.4. Faster Proposed Region-Convolutional Neural Networks (Faster R-CNNs)

In the faster R-CNN architecture shown in Figure 4, there is only one CNN trained for the image, which sends the data to the feature maps. Furthermore, region proposals are created regarding the region of interest (RoI) [27]. The RoI provides an interesting area of the image that requires classifying. This process significantly reduces the computational time to 0.2 s per image [27], while the video feed is detecting objects.

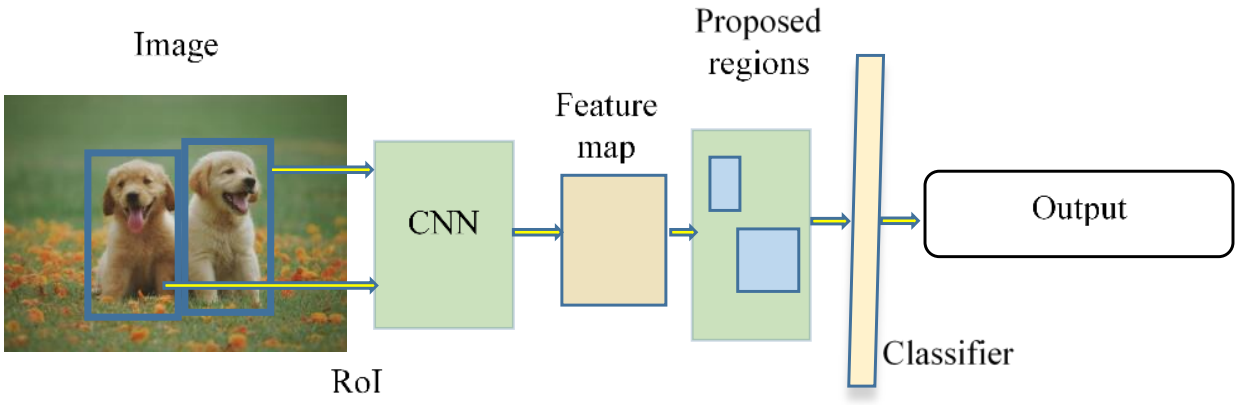

Figure 4. Faster R-CNN architecture where CNN is computed only for proposed regions. This process is repeated in the form of feature maps and region proposals followed by similar FC layers classifying in the end. CNN is computed only for proposed regions. This process is repeated in the form of feature maps and region proposals followed by similar FC layers classifying in the end.

2.3. Bayesian Model Development

As stated in Section 1, we used a predictive model approach to estimate the total latency in both data transmission and analysis. In this paper, we defined Bayesian model development as the process of creating a predictive model whose parameters are inferred using Bayesian inference. Traditionally, the point values of model parameters are estimated using the maximum likelihood approach. In the case of Bayesian model development, we obtain the probability distributions of model parameters as opposed to pointing values, which result in a probabilistic prediction of the output. Consider the relationship $y = f(x;\theta)$, where y, x, and θ represent the model output, inputs, and model parameters, respectively, and $f(.)$ represents the predictive model. If $D = \{x_D, y_D\}$ represents the dataset available on the input and output variables, then the model parameters can be estimated using Bayes theorem as $f(\theta|D) \propto f(D|\theta)f(\theta)$.

In the paper, we used the sampling–resampling method to estimate the unknown parameters θ [11,54].

3. Proposed Methodology and Experimentation

The proposed framework components are provided in Figures 5 and 6. The framework consists of a UR5 (it can be any robotic or actuation system, however), data exchanges over networking elements, and a cyber system. The laptop is used, which represents the object detection processing phase and respective control action corresponding to the object detection. These control actions are transmitted to the robotic system through a real-time data exchange (RTDE) system operating on TCP/IP standards. The cyber system therefore operates on the Internet for data exchange with the physical system in contrast to other CPSs such as self-driving cars.

The proposed framework, provided in Figure 7, shows that the cyber system is trained over a class of objects that are subject to detection. These detected classes are transferred to the physical system—a robot in this case. The data transfer from the laptop is using RTDE to the UR5 mini ITX computer in real time. In other words, RTDE is the socket communication, in which the laptop is the client and UR5 is the server [28]. The server is accessed by the controller's (laptop's) IP address. Therefore, the ODCA is operating in such a way that it creates a control system that is intelligent to detect objects, gather and process data, and send the data to the physical system to operate in real time.

In the experiment, we used the playing cards as objects to be detected for an important reason. There are many robotic arms in the industry that do not have 3D high-resolution cameras. We aimed to develop a framework that offers a solution to this problem. Moreover, some operations require humans in the loop. Therefore, this system is letting humans show a sign in the form of playing cards, which are the control signals to the physical systems.

The ODCA operates on a faster R-CNN. We developed a dataset consisting of images of playing cards.

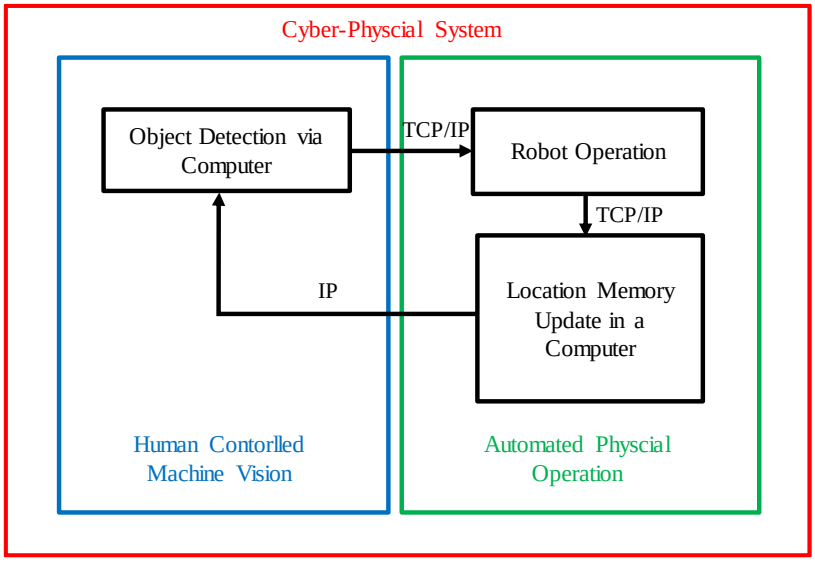

Figure 5. Conceptual cyber-physical system that uses machine vision and a robotic system. Data exchange protocols between various components are also shown.

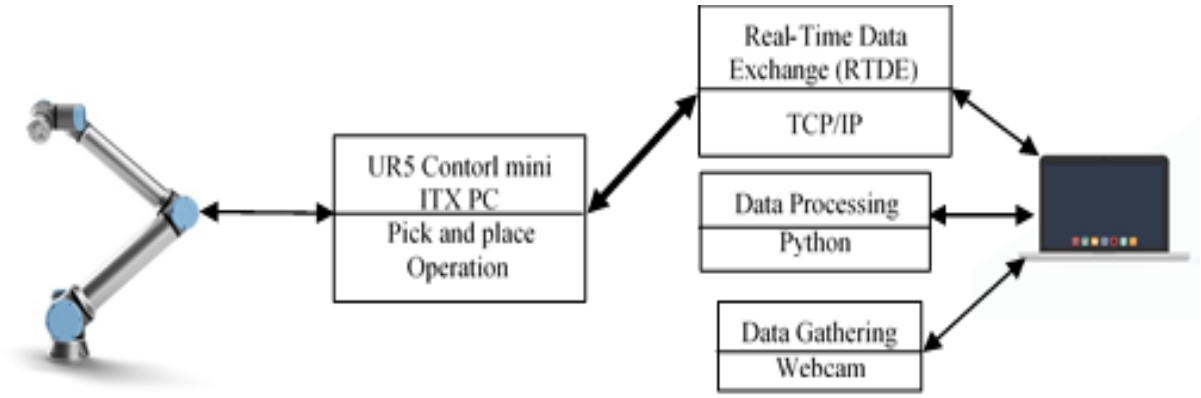

Figure 6. Actuation system capable of implementing control actions. Framework involves laptop computer (computing system), cable for socket communication (follows TCP/IP standards), and UR5 robot.

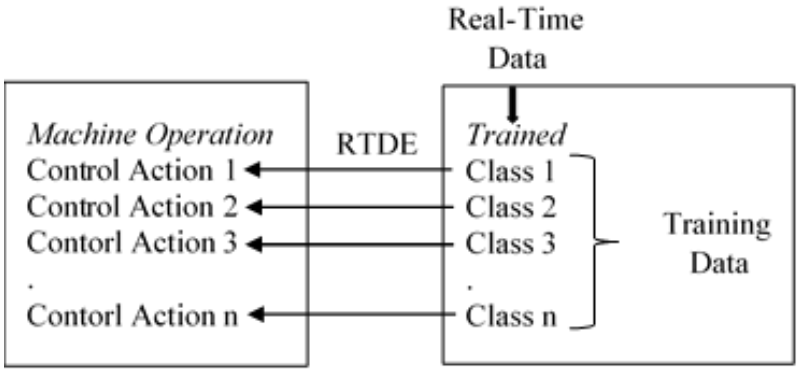

Figure 7. CPS framework that can operate on object detection and control algorithm (ODCA).

The Microsoft COCO (Common Objects in Context) dataset [55] is used, which has the structured layout of the faster R-CNN model directory. The manual creation of objects is carried out with the help of label image software. For example, there were 4 different classes

(playing cards) that were added with 296 images of the playing cards. These 296 images were taken in different environments with different backgrounds and positions in the image. This process increased the likelihood of accurate object detection. After having set the data directory, the programming is performed on Python. Python code is built with the help of the Anaconda platform where various deep-learning toolboxes were used. The major toolboxes were Tensorflow and Keras.

The dataset was trained with 296 mages for almost 3197 computational iterations. The training process accounted for almost 9 h of training to achieve 0.05 root-mean-square (RMS) error. In training, the optimization of the loss (or cost) function was performed by using the gradient descent algorithm. The gradient descent algorithm finds the local or global minima of a function by learning the appropriate gradient to reduce errors [56]. This means the gradient descent enables the model to gradually converge toward the minimum change in the parameter changes in the loss [57], in other words, a convergence of the model. The optimization results are shown in Section 3. After training the faster R-CNN model, the information was fed to the ODCA. Algorithm 1 (ODCA) is also explained in the next section. In the testing process, the ODCA performed successful object detection of the playing cards, showing more than 80% probability of correctly identifying the objects.

Dataset is structured in the famous Microsoft COCO dataset. We considered playing cards as the classes in the algorithm. There were 4 classes on which the faster R-CNN was trained. For training purposes, the dataset images were taken over varying degrees of brightness and different camera angles. To provide a certain structure in training, we used two brightness levels and 37 different angles for 4 playing cards (a jack, a queen, a king, and a number 8). This combination yields 296 images (4 cards, 2 brightness levels, 37 angles). We used 216 images for training and 80 for testing to avoid overfitting. The gradient descent algorithm was used for training to optimize the loss function. To improve detection accuracy, the model was trained over 3197 iterations of the gradient descent algorithm to achieve a root-mean-square (RMS) error of less than 5%. The RMS error is shown in Section 3. In our training and testing approach, we observed a detection accuracy of over 90%.

Algorithm 1 is broadly divided into 3 functions. The main functions include 'get_data' and 'get_image_data'. The program's primary function is to recognize the objects in the image and command the robot through socket communication (following the TCP/IP protocols), which performs a pick-and-place operation. For the experiment, three objects were considered: an orange block, a thread reel, and a plastic bottle. All three objects had two placing locations, one for picking and one for placing locations in the memory. Based on the operation, the location is updated. Since the UR5 robot does not have an inbuilt camera, the laptop webcam was used, which resulted in creating one more step in which the playing cards were assigned as keys substituting for the object themselves. Therefore, 4 playing cards were used as a human-controlled sign to command the robot for an operation. Figure 8 shows the experimental setup.

The data received from the robot are used to ensure that it starts and ends the operation from its home position. If the robot is already at the home position, it does not waste time and proceeds further with other operations.

The UR5 robot allows us to communicate with it by creating a socket communication with port no. 30003. This port sends data from the robot in the form of byte packets of size 1108 bytes [58]. Each byte in the packet contains certain significant information about the robot. A function called 'get_data' in the Python program was created to receive these data. A laptop computer, in networking terms, works as the client, and the UR5 control unit works as the server. This server can be accessed by the IP address of the controller (the laptop). In data transfers, the control lies in the laptop and can be operated based on the command options [58,59]. The pseudocode given below explains Algorithm 1 (ODCA) developed for the proposed framework.

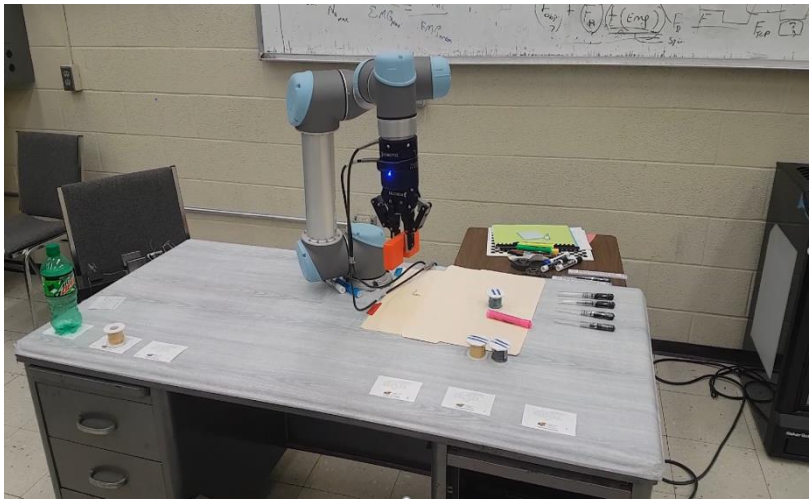

Figure 8. Experimental setup showing the UR5 and three objects.

Algorithm 1: ODCA for a robot pick and place operation operating on DL

start *the robot operation;*
Collect robot joint data by get_data to help plan robot motion;
if *robot at home position*
then *initialize the variables*
else *move to the home position*
then *initialize the variables*
end
Update the image data with get_image_data;
if *the class matches with the image*
then *pick and place the part from the memory location*
else *stay at home position*
Update the location in memory;
end

The histogram of the standard deviation of the weights of neural networks used in Algorithm 1 over the number of iterations can be seen in Figure 9. The network generated random weights in the training process, and from the histogram, it is visible that there was a large standard deviation of the weights in the initial stages. The resulting histogram provided in Figure 9 shows the training process of a DL neural network. Another important result is shown in Figure 10. Figure 10 shows the loss function optimization by the gradient descent algorithm. At the end of the training process, the RMS error was almost 0.05. The loss function graph in Figure 10 shows improvement in the accuracy of Algorithm 1 (detecting correct cards) over time and with the number of iterations [60]. Therefore, the objects were successfully recognized, and the robot operated based on the recognized objects, which can be seen in Figure 8. The use of playing cards in the research was for demonstration of the framework shown in Figure 11. In the real world, image processing and object detection analysis will be performed on industrial parts during quality control operations.

One of the challenges of this proposed methodology is picking and placing of moving objects. Moving objects here means the objects whose positions change with respect to time (dynamically). One such example is picking and placing moving objects. This requires the development of faster real-time object detection and also robust mechanics models to predict the positions of moving objects.

The paper showed that DL-based models have latency, which affects the system performance. In the paper, we successfully propagated the existing latency while performing object detection. This latency will help develop better DL-based CPS models that can have

applications in a dynamic setting. The future work is to propagate latency in data transfer or networking in general.

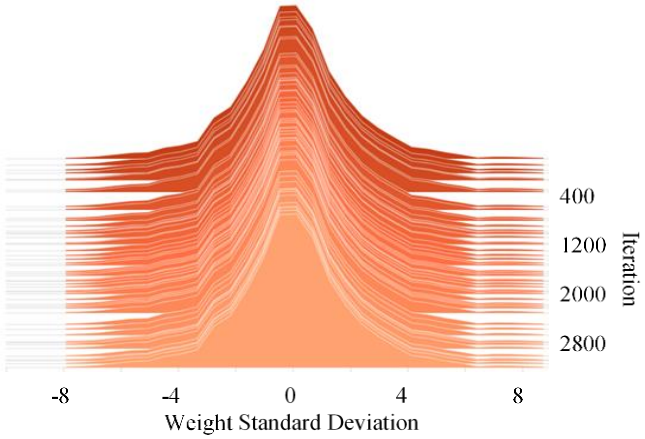

Figure 9. Histogram of standard deviation neural network weights over the number steps.

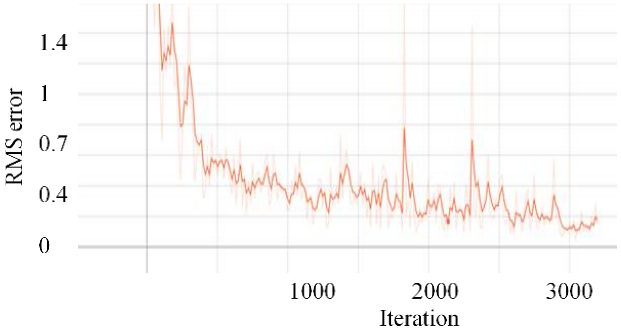

Figure 10. Loss function minimization using the gradient descent Algorithm 1 over number of steps.

Figure 11. Experimental setup in which laptop is controlling the robot as shown in Figure 8. A playing card is shown to the laptop webcam to perform object detection.

4. Latency in the CPS

To study latency in the CPS framework, we performed experiments and collected data, which are later used to estimate overall latency in the computing and communication systems to facilitate the design of effective real-time industrial systems. For the proposed CPS framework, the important part of performing the physical operations will depend on the cyber-system's information processing and the networking element's capability of sending the control signals in real time such that the operations can be performed in a dynamic setting. Therefore, the important areas to propagate latency are the time taken for information processing by a cyber system and the time taken by a responsible networking element between the cyber system and a physical component of the system. Data points are collected using a built-in timer following Algorithm 2 given below.

Algorithm 2: Algorithm that is used for data collection for the sampling and resampling algorithm to study latency in a CPS

```
start the time;
if algorithm 1 is running;
while true:
        Collect activation time (x1)
                Collect the detection time (x2)
                Calculate the Total elapsed time (y)
break
end
print x1, x2, and y
end
```

The collected data points are used to build a model to predict the overall system latency using the latency of individual components. A Bayesian model calibration approach is used to train the model and estimate the model parameters. Using the Bayesian calibration approach, we model the probability distributions of the model parameters and not just deterministic (single-point) values of the model parameters; the distributions of the model parameters are used to obtain probabilistic predictions of the overall latency. In this paper, we demonstrated the use of sampling–resampling algorithm for Bayesian model calibration. A brief review of the algorithm was provided in Section 2.3.

For model calibration, we collected 11 data points on different days and times to avoid any biases and dependence in the data gathering process. We collected data on three different variables—activation time, detection time, and the total elapsed time associated with Algorithm 1 (pick-and-place operations). Activation time (x_1) is the time between the start of the operation and opening of the webcam. Detection time (x_2) is the time taken by the Web cam to detect the object. The collected data points are given below in Table 1.

Since the total latency depends on both the activation time and the object time, we trained the model given below:

$$\boldsymbol{y} = \boldsymbol{a}\boldsymbol{x}_1 + \boldsymbol{b}\boldsymbol{x}_2 + \boldsymbol{c} + \boldsymbol{\epsilon} \tag{1}$$

$$\boldsymbol{\epsilon} \ \sim \ \boldsymbol{N}(0, \ \boldsymbol{\sigma}) \tag{2}$$

In Equation (1), a, b, and c represent the model parameters, while ϵ represents the measurement error, which is traditionally assumed to be a Gaussian distribution centered at 0. A Gaussian distribution is assumed as it is symmetric about the mean resulting in an equal likelihood of observing both the positive and negative measurement error values. In Equation (2), σ represents the standard deviation of the measurement error ϵ, which is an unknown and is therefore calibrated along with the model parameters. To perform Bayesian model calibration, prior distributions need to be assumed for all the parameters (a, b, c, and σ). The prior distributions represent our knowledge about the parameters before the experimentation process. Since the total elapsed time is the combination of the activation time and detection time, we assumed Gaussian distributions as prior distributions for

parameters a and b with a mean of 1. We assumed 25% coefficient of variation resulting in the standard deviation values of 0.25. Therefore, the prior distribution of parameters a and b is $N(1,\ 0.25)$. Since parameter c represents the intercept, we assumed a Gaussian distribution centered at 0, and assumed a standard deviation of 1. Since any knowledge regarding parameter σ was unavailable, we assumed a uniform distribution between 0 and 1 as the prior distribution.

Table 1. Experimental Data for the Operational Time.

#	Activation Time (x_1)	Detection Time (x_2)	Total Elapsed Time (y)
1	1.56	24.38	26.38
2	2.13	24.45	27.03
3	1.57	24.10	26.14
4	2.25	24.46	27.18
5	1.53	25.06	27.13
6	2.10	24.19	26.43
7	2.07	24.16	26.28
8	2.07	24.43	26.56
9	1.58	24.08	26.19
10	2.06	24.16	26.26
11	2.02	24.22	26.27

Using these prior distributions and available eleven data points, we performed Bayesian model calibration using the sampling–resampling algorithm. The steps of the sampling–resampling algorithms are as follows:

1. Generate N samples from the prior distribution, $f(\theta)$. The samples are denoted as $\theta_i,\ i = 1, 2 \ldots N$.
2. Compute the likelihood of observing the available data for each θ_i as $L(\theta_i) = f(y_D|x_D,\ \theta_i)$.
3. Compute a weight w_i corresponding to each θ_i as $w_i = \frac{L(\theta_i)}{\sum_{i=1}^{N} L(\theta_i)}$.
4. Generate N samples of θ_i according to their weights w_i.

The generated samples in step 4 above form the posterior distribution of θ, i.e., $f(\theta|D)$. The continuous probability distribution of θ_i uses the generated N samples through kernel density estimation [61].

We used 20,000 samples in the sampling–resampling algorithm. The mean and standard deviation values of the prior and posterior distributions of all the four parameters are available in Table 2. The prior and posterior histograms are available in Figure 12. From Table 2 and Figure 12, comparing the prior and posterior standard deviation values, we can observe that the values associated with parameter b had the maximum change; therefore, we can conclude that parameter b observed the maximum information gain through the experimental data.

Table 2. Mean and standard deviation values of prior and posterior distributions of calibration parameters.

		Mean	Standard Deviation
a	Prior	1	0.25
	Posterior	0.689	0.208
b	Prior	1	0.25
	Posterior	1.038	0.038
c	Prior	0	1
	Posterior	−0.062	0.906
σ	Prior	0.5	0.289
	Posterior	0.207	0.067

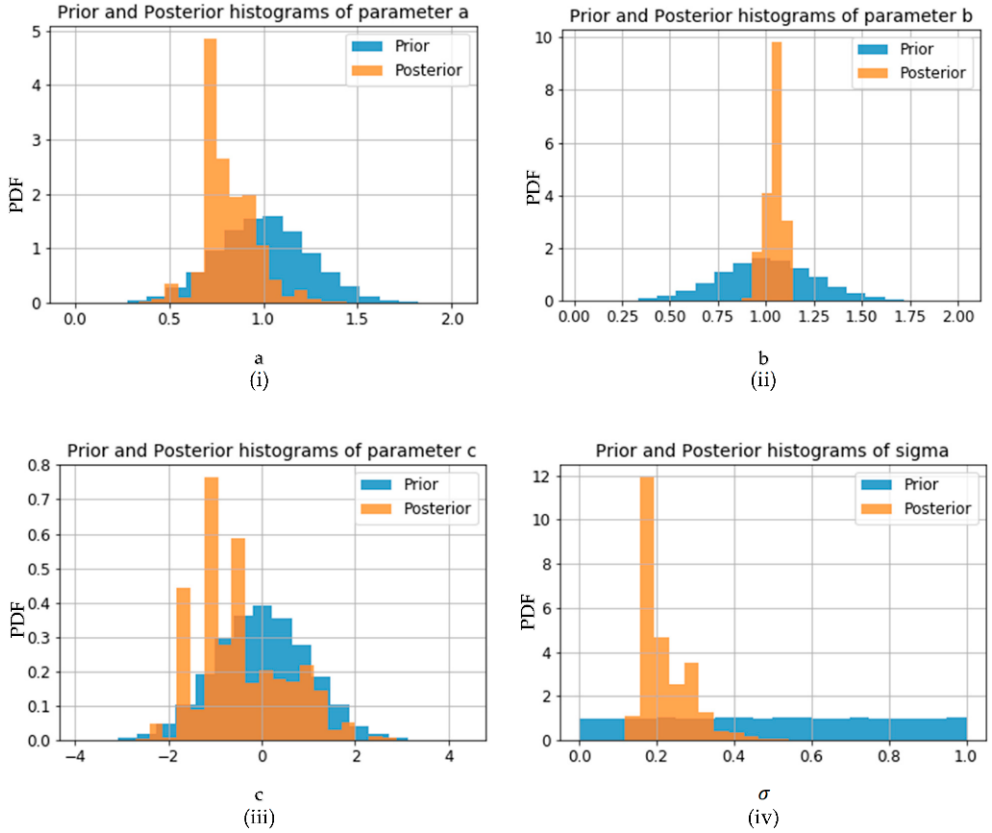

Figure 12. Prior and posterior histograms of (**i**) parameter a, (**ii**) parameter b, (**iii**) parameter c, and (**iv**) standard deviation associated with error (σ).

5. Conclusions

This paper discussed an intelligent cyber-physical system framework that operates over collaborative UR5 robotic arms, machine vision, image processing, and object detection for real-time industrial applications. Proposed real-time object detection and control algorithm (ODCA) suitably meets the practical requirements and can be used for various industrial applications such as a pick-and-place operation, one of the most common industrial applications using a collaborative robotic arm. An external Web camera that captures real-time images is used, which is connected to a robotic arm for pick-and-place operations. The research provides a concrete algorithmic setup to develop CPS for industrial operations including real-time and real-life criteria such as latency. Latency can be a crucial factor for real-time operations. This paper employed a model-based approach for the estimation of the total latency; the paper employed Bayesian inference analysis using the Bayesian sampling–resampling algorithm. Resulting latency propagation for the given framework using sampling–resampling algorithm provides an increased reproducibility and repeatability. However, a limitation of the experimental setup was that the robot did not have an inbuilt camera.

Future work can consider implementing the proposed methodology for dynamic systems such as picking and placing moving objects. This requires the development of faster real-time object detection and also robust mechanics models to predict the positions of moving objects.

Author Contributions: V.M. performed conceptualization, methodology, software, validation, formal analysis, investigation, and writing original draft preparations. S.N. and K.K. contributed in conceptualization, writing—review and editing, supervision, and project administration. A.A. contributed in writing—review and editing, visualization, and data curation. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Sample data can be found from the Microsoft COCO dataset website –https://cocodataset.org/#home (accessed on 10 October 2022).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lasi, H.; Fettke, P.; Kemper, H.-G.; Feld, T.; Hoffmann, M. Industry 4.0. *Bus. Inf. Syst. Eng.* **2014**, *6*, 239–242. [CrossRef]
2. Rüßmann, M.; Lorenz, M.; Gerbert, P.; Waldner, M.; Engel, P.; Harnisch, M.; Justus, J. Future of Productivity and Growth in Manufacturing. Boston Consulting. 2015. Available online: https://scholar.google.com/scholar?hl=en&as_sdt=0%2C44&q=Future+of+Productivity+and+Growth+in+Manufacturing+boston+consulting&btnG=&oq=Future+of+Productivity+and+Growth+in+Manufacturing+boston+consult#d=gs_cit&t=1667019003402&u=%2Fscholar%3Fq%3Dinfo%3Ak1ZrpaIBKXQJ%3Ascholar.google.com%2F%26output%3Dcite%26scirp%3D0%26hl%3Den (accessed on 20 November 2019).
3. Almada-Lobo, F. The Industry 4.0 revolution and the future of Manufacturing Execution Systems (MES). *J. Innov. Manag.* **2015**, *3*, 16–21. [CrossRef]
4. Lee, J.; Kao, H.-A.; Yang, S. Service Innovation and Smart Analytics for Industry 4.0 and Big Data Environment. *Procedia CIRP* **2014**, *16*, 3–8. [CrossRef]
5. Jamwal, A.; Agrawal, R.; Sharma, M.; Giallanza, A. Industry 4.0 Technologies for Manufacturing Sustainability: A Systematic Review and Future Research Directions. *Appl. Sci.* **2021**, *11*, 5725. [CrossRef]
6. Jamwal, A.; Agrawal, R.; Sharma, M.; Kumar, V.; Kumar, S. Developing A sustainability framework for Industry 4.0. *Procedia CIRP* **2021**, *98*, 430–435. [CrossRef]
7. Bai, C.; Dallasega, P.; Orzes, G.; Sarkis, J. Industry 4.0 technologies assessment: A sustainability perspective. *Int. J. Prod. Econ.* **2020**, *229*, 107776. [CrossRef]
8. Vaidya, S.; Ambad, P.; Bhosle, S. Industry 4.0—A Glimpse. *Procedia Manuf.* **2018**, *20*, 233–238. [CrossRef]
9. Pivoto, D.G.S.; de Almeida, L.F.F.; Da Rosa Righi, R.; Rodrigues, J.J.P.C.; Lugli, A.B.; Alberti, A.M. Cyber-physical systems architectures for industrial internet of things applications in Industry 4.0: A literature review. *J. Manuf. Syst.* **2021**, *58*, 176–192. [CrossRef]
10. Jamwal, A.; Agrawal, R.; Manupati, V.K.; Sharma, M.; Varela, L.; Machado, J. Development of cyber physical system based manufacturing system design for process optimization. *IOP Conf. Series: Mater. Sci. Eng.* **2020**, *997*, 012048. [CrossRef]
11. Nannapaneni, S.; Mahadevan, S.; Dubey, A.; Lee, Y.-T.T. Online monitoring and control of a cyber-physical manufacturing process under uncertainty. *J. Intell. Manuf.* **2021**, *32*, 1289–1304. [CrossRef]
12. Kevin, A. That ' Internet of Things' Thing. *RFiD J.* **2010**, *22*, 4986.
13. Lee, E.A. Cyber Physical Systems: Design Challenges. In Proceedings of the 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC), Orlando, FL, USA, 5–7 May 2008; pp. 363–369.
14. Radanliev, P.; De Roure, D.; Nicolescu, R.; Huth, M.; Santos, O. Artificial Intelligence and the Internet of Things in Industry 4.0. *CCF Trans. Pervasive Comput. Interact.* **2021**, *3*, 329–338. [CrossRef]
15. Mazumder, S.K.; Kulkarni, A.; Sahoo, S.; Blaabjerg, F.; Mantooth, H.A.; Balda, J.C.; Zhao, Y.; Ramos-Ruiz, J.A.; Enjeti, P.N.; Kumar, P.R.; et al. A Review of Current Research Trends in Power-Electronic Innovations in Cyber–Physical Systems. *IEEE J. Emerg. Sel. Top. Power Electron.* **2021**, *9*, 5146–5163. [CrossRef]
16. Wolf, W.; Tech, G. Cyber-Physical Systems. *Impact Control. Technol.* **2009**, *12*, 88–89. [CrossRef]
17. Zupančič, J.; Bajd, T. Comparison of position repeatability of a human operator and an industrial manipulating robot. *Comput. Biol. Med.* **1998**, *28*, 415–421. [CrossRef]
18. Heyer, C. Human-robot interaction and future industrial robotics applications. In Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 18–22 October 2010; pp. 4749–4754. [CrossRef]
19. Alvarez-De-Los-Mozos, E.; Renteria, A. Collaborative Robots in e-waste Management. *Procedia Manuf.* **2017**, *11*, 55–62. [CrossRef]
20. Luo, S. Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. *Appl. Soft Comput.* **2020**, *91*, 106208. [CrossRef]
21. Zhong, R.Y.; Xu, X.; Klotz, E.; Newman, S.T. Intelligent Manufacturing in the Context of Industry 4.0: A Review. *Engineering* **2017**, *3*, 616–630. [CrossRef]
22. Mahesh, B. Machine Learning Algorithms-A Review. *Int. J. Sci. Res.* **2018**, *9*, 381–386. [CrossRef]
23. Ayodele, T. Types of Machine Learning Algorithms. In *New Advances in Machine Learning*; Zhang, Y., Ed.; IntechOpen: London, UK, 2010. [CrossRef]
24. Weckman, G.R.; Ganduri, C.V.; Koonce, D.A. A neural network job-shop scheduler. *J. Intell. Manuf.* **2008**, *19*, 191–201. [CrossRef]
25. Venieris, S.I.; Bouganis, C.-S. Latency-driven design for FPGA-based convolutional neural networks. In Proceedings of the 2017 27th International Conference on Field Programmable Logic and Applications (FPL), Ghent, Belgium, 4–8 September 2017. [CrossRef]
26. Maru, V.; Nannapaneni, S.; Krishnan, K. Internet of Things Based Cyber-Physical System Framework for Real-Time Operations. In Proceedings of the 2020 IEEE 23rd International Symposium on Real-Time Distributed Computing (ISORC), Nashville, TN, USA, 19–21 May 2020.

27. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef] [PubMed]
28. Universal Robots. Real-Time Data Exchange (RTDE) Guide-22229. Universal Robots: Support. Available online: https://www.universal-robots.com/how-tos-and-faqs/how-to/ur-how-tos/real-time-data-exchange-rtde-guide-22229/ (accessed on 5 May 2021).
29. Cardoso, J.; Derler, P.; Eidson, J.C.; Lee, E. Network latency and packet delay variation in cyber-physical systems. In Proceedings of the 2011 IEEE Network Science Workshop, West Point, NY, USA, 22–24 June 2011; pp. 51–58. [CrossRef]
30. Delange, J.; Feiler, P. Incremental Latency Analysis of Heterogeneous Cyber-Physical Systems. In Proceedings of theThird International Workshop on Real-time and Distributed Computing in Emerging Applications, Rome, Italy, 2 December 2014; pp. 21–27.
31. Zhu, Y.; Dong, Y.; Ma, C.; Zhang, F. A Methodology of Model-Based Testing for AADL Flow Latency in CPS. In Proceedings of the 2011 Fifth International Conference on Secure Software Integration and Reliability Improvement-Companion, Jeju, Korea, 27–29 June 2011; pp. 99–105. [CrossRef]
32. Oliver, R.S.; Fohler, G. Probabilistic estimation of end-to-end path latency in Wireless Sensor Networks. In Proceedings of the 2009 IEEE 6th International Conference on Mobile Adhoc and Sensor Systems, Macau, China, 12–15 October 2009; pp. 423–431. [CrossRef]
33. Claypool, M.; Finkel, D. The effects of latency on player performance in cloud-based games. In Proceedings of the 2014 13th Annual Workshop on Network and Systems Support for Games, Nagoya, Japan, 4–5 December 2014; pp. 1–6. [CrossRef]
34. Van Trees, H.L.; Bell, K.L. A Tutorial on Particle Filters for Online Nonlinear/NonGaussian Bayesian Tracking. In *Bayesian Bounds for Parameter Estimation and Nonlinear Filtering/Tracking*; John Wiley: New York, NY, USA, 2007; pp. 10–1109. [CrossRef]
35. Derler, P.; Lee, E.A.; Vincentelli, A.S. Modeling Cyber–Physical Systems. *Proc. IEEE* **2011**, *100*, 13–28. [CrossRef]
36. Mörth, O.; Emmanouilidis, C.; Hafner, N.; Schadler, M. Cyber-physical systems for performance monitoring in production intralogistics. *Comput. Ind. Eng.* **2020**, *142*, 106333. [CrossRef]
37. Saufi, S.R.; Bin Ahmad, Z.A.; Leong, M.S.; Lim, M.H. Challenges and Opportunities of Deep Learning Models for Machinery Fault Detection and Diagnosis: A Review. *IEEE Access* **2019**, *7*, 122644–122662. [CrossRef]
38. Müller, J.M.; Kiel, D.; Voigt, K.-I. What Drives the Implementation of Industry 4.0? The Role of Opportunities and Challenges in the Context of Sustainability. *Sustainability* **2018**, *10*, 247. [CrossRef]
39. Jamwal, A.; Agrawal, R.; Sharma, M.; Kumar, A.; Kumar, V.; Garza-Reyes, J.A.A. Machine learning applications for sustainable manufacturing: A bibliometric-based review for future research. *J. Enterp. Inf. Manag.* **2021**, *35*, 566–596. [CrossRef]
40. Miskuf, M.; Zolotova, I. Comparison between Multi-Class Classifiers and Deep Learning with Focus on Industry 4.0. In Proceedings of the 2016 Cybernetics and Informatics, K and I 2016-Proceedings of the 28th International Conference, IEEE, Levoca, Slovakia, 2–5 February 2014; pp. 1–5. [CrossRef]
41. Beaupre, M. Collaborative robot technology and applications. International Collaborative Robots Workshop. 2014, p. 41. Available online: https://www.robotics.org/userAssets/riaUploads/file/4-KUKA_Beaupre.pdf (accessed on 20 October 2022).
42. Dumonteil, G.; Manfredi, G.; Devy, M.; Confetti, A.; Sidobre, D. Reactive Planning on a Collaborative Robot for Industrial Applications. In Proceedings of the 2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO), Colmar, France, 21–23 July 2015; pp. 450–457. [CrossRef]
43. Brandstötter, M.; Komenda, T.; Ranz, F.; Wedenig, P.; Gattringer, H.; Kaiser, L.; Breitenhuber, G.; Schlotzhauer, A.; Müller, A.; Hofbaur, M. Versatile Collaborative Robot Applications Through Safety-Rated Modification Limits. *Adv. Intell. Syst. Comput.* **2020**, *980*, 438–446. [CrossRef]
44. Biggs, G.; Macdonald, B. A Survey of Robot Programming Systems. In Proceedings of the Australasian Conference on Robotics and Automation; 2003; pp. 1–3.
45. El-Zorkany, H. Robot Programming. *INFOR: Inf. Syst. Oper. Res.* **1985**, *23*, 430–446. [CrossRef]
46. Zhou, Z.; Xiong, R.; Wang, Y.; Zhang, J. Advanced Robot Programming: A Review. *Curr. Robot. Rep.* **2020**, *1*, 251–258. [CrossRef]
47. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. TensorFlow: A System for Large-Scale Machine Learning. Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation. *OSDI* **2016**, *2016*, 265–283. [CrossRef]
48. François, C. *Keras: The Python Deep Learning Library*; Astrophysics source code library, 2015; ascl:1806.022; Available online: https://ascl.net/1806.022 (accessed on 20 November 2019).
49. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
50. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587. [CrossRef]
51. Girshick, R. Fast R-CNN. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1440–1448. [CrossRef]
52. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *NIPS* **2017**, *60*, 84–90. [CrossRef]
53. Gkioxari, G.; Girshick, R.; Malik, J. Contextual Action Recognition with R CNN. In Proceedings of the IEEE international conference on computer vision, Santiago, Chile, 7–13 December 2015; 2015; pp. 1080–1088. [CrossRef]

54. Smith, A.F.M.; Gelfand, A.E. Bayesian Statistics without Tears: A Sampling–Resampling Perspective. *Am. Stat.* **1992**, *46*, 84–88. [CrossRef]
55. Lin, T.Y.; Maire, M.; Belongie, S.; Bourdev, L.; Girshick, R.; Hays, J.; Perona, P.; Zitnick, C.L.; Dollár, P. Microsoft COCO: Common Objects in Context. Available online: https://cocodataset.org/ (accessed on 20 November 2019).
56. Mandic, D.P. Descent Algorithm. *Signal Processing* **2004**, *11*, 115–118.
57. Cauchy, M.A. Méthode Générale Pour La Résolution Des Systèmes d'équations Simultanées. *C R Hebd Seances Acad. Sci.* **1847**, *25*, 536–538.
58. Xue, M.; Zhu, C. The Socket Programming and Software Design for Communication Based on Client/Server. In Proceedings of the 2009 Pacific-Asia Conference on Circuits, Communications and Systems, Chengdu, China, 16–17 May 2009; pp. 775–777. [CrossRef]
59. Espinosa, F.; Salazar, M.; Valdes, F.; Bocos, A. Communication architecture based on Player/Stage and sockets for cooperative guidance of robotic units. In Proceedings of the 2008 16th Mediterranean Conference on Control and Automation, Ajaccio, France, 25–27 June 2008; pp. 1423–1428. [CrossRef]
60. Bengio, Y.; LeCun, Y. Scaling learning algorithms towards AI. *Large-Scale Kernel Mach.* **2007**, *34*, 1–41.
61. Terrell, G.R.; Scott, D.W. Variable Kernel Density Estimation. *Ann. Stat.* **1992**, *20*, 1236–1265. [CrossRef]