metals

MDPI

Article

Defect Recognition in High-Pressure Die-Casting Parts Using Neural Networks and Transfer Learning

Georgia Andriosopoulou [1,2,†], Andreas Mastakouris [1,2,†], Dimosthenis Masouros [2], Panorios Benardos [1,*], George-Christopher Vosniakos [1] and Dimitrios Soudris [2]

[1] Manufacturing Technology Laboratory, School of Mechanical Engineering, National Technical University of Athens, Heroon Polytechniou 9, GR15772 Athens, Greece; vosniak@central.ntua.gr (G.-C.V.)
[2] Microprocessors and Digital Systems Laboratory, School of Electrical and Computer Engineering, National Technical University of Athens, Heroon Polytechniou 9, GR15772 Athens, Greece; dmasouros@microlab.ntua.gr (D.M.); dsoudris@microlab.ntua.gr (D.S.)
* Correspondence: pbenard@mail.ntua.gr; Tel.: +30-210-772-1799
† These authors contributed equally to this work.

Abstract: The quality control of discretely manufactured parts typically involves defect recognition activities, which are time-consuming, repetitive tasks that must be performed by highly trained and/or experienced personnel. However, in the context of the fourth industrial revolution, the pertinent goal is to automate such procedures in order to improve their accuracy and consistency, while at the same time enabling their application in near real-time. In this light, the present paper examines the applicability of popular deep neural network types, which are widely employed for object detection tasks, in recognizing surface defects of parts that are produced through a die-casting process. The data used to train the networks belong to two different datasets consisting of images that contain various types of surface defects and for two different types of parts. The first dataset is freely available and concerns pump impellers, while the second dataset has been created during the present study and concerns an automotive part. For the first dataset, Faster R-CNN and YOLOv5 detection networks were employed yielding satisfactory detection of the various surface defects, with mean average precision (mAP) equal to 0.77 and 0.65, respectively. Subsequently, using transfer learning, two additional detection networks of the same type were trained for application on the second dataset, which included considerably fewer images, achieving sufficient detection capabilities. Specifically, Faster R-CNN achieved mAP equal to 0.70, outperforming the corresponding mAP of YOLOv5 that equalled 0.60. At the same time, experiments were carried out on four different computational resources so as to investigate their performance in terms of inference times and consumed power and draw conclusions regarding the feasibility of making predictions in real time. The results show that total inference time varied from 0.82 to 6.61 s per image, depending on the computational resource used, indicating that this methodology can be integrated in a real-life industrial manufacturing system.

Keywords: quality control; high-pressure die-casting; defect recognition; computer vision; deep learning; transfer learning

check for updates

Citation: Andriosopoulou G.; Mastakouris A.; Masouros D.; Benardos P.; Vosniakos G.-C.; Soudris D. Defect Recognition in High-Pressure Die-Casting Parts Using Neural Networks and Transfer Learning. *Metals* **2023**, *13*, 1104. https://doi.org/10.3390/met13061104

Academic Editor: Noé Cheung

Received: 2 May 2023
Revised: 1 June 2023
Accepted: 8 June 2023
Published: 12 June 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Automation of industrial quality control processes is a problem that has been extensively studied in recent years by researchers [1,2]. The term quality control refers to the process by which products are tested and measured to ensure they meet predefined specifications and/or standards. It is an important function in industrial production since, on one hand, it controls to what extent the goals set are achieved, and, on the other hand, it determines the changes that must be followed in the production system so as to correct any observed product defects or out-of-specification deviations.

In general, quality control in industry can be achieved using various methods and accompanying technological means. The most widely used include non-destructive testing, such as penetrant liquids, ultrasound, X-rays or γ-rays, which offer excellent results depending on the type of defect examined [3]. However, these tasks require personnel with the appropriate knowledge, they involve time-consuming procedures and the final decision regarding the quality of a part is linked to the subjective opinion and experience of the individual expert. Furthermore, although the human ability to visually inspect different objects is very high, the fatigue resulting from performing repetitive tasks can lead to human errors.

To overcome such limitations, lately, there has been a shift towards the application of machine learning (ML) to automate the process of quality control, thus eliminating the need for human presence in the loop [4]. Training a machine vision system to quickly and automatically assess the quality of the industrial parts can contribute to increasing productivity and reducing the operating costs of a factory [5–7]. In addition, in the context of the fourth industrial revolution, the integration of digital technology in industrial production [8] becomes necessary, as, in this way, the role of personnel is constantly upgraded, while repetitive and standardized tasks are gradually assigned to machines. Combined with high-speed wireless telecommunications [9], the quality control process can be automated to a significant degree, resulting in improved overall productivity.

This paper deals with automation of quality control using photos of industrial parts from two different datasets, the first being substantially larger than the second. Both datasets depict mechanical parts produced by the die-casting process, the difference being that the second dataset depicts high-pressure die-casting parts whilst the first one refers to low-pressure die-casting. The purpose is to develop a machine vision system that will automatically detect surface defects on cast parts and will be based on deep neural networks.

The contribution of this paper lies in the fact that, so far, the study of automating quality control in high-pressure die-casting products is at an early stage, with publications from the scientific community being limited. Moreover, the present work leverages transfer learning from pretrained models in a larger dataset, so as to investigate whether the surface defect detection in a smaller dataset can be improved. Finally, comparing four computational resources, the real-time deployment potential of neural networks for this purpose is systematically examined.

More specifically, the goals of the present paper concern the following:

- Identification and localization of defects in the defective parts of the first dataset using Faster R-CNN and YOLOv5 detection networks.
- Study of the transfer learning process for this specific application in the second dataset.
- Comparison of the two detection networks employed for each dataset and selection of the best one.
- Comparison of four deployment strategies regarding the inference times on the second dataset in order to examine whether the predictions can be performed in real time and consequently integrated into the production system.

The structure of this paper is as follows: In Section 2, a state-of-the-art review is carried out. In Section 3, the architectures of Faster R-CNN and YOLOv5 networks employed to detect the defects in the datasets are analyzed. In Section 4, the two datasets and the main hyperparameters of the networks are described, while, in Section 5, the results achieved are presented. In Section 6, the inference times using four different computational resources are compared, and, finally, Section 7 summarizes the conclusions and suggests future extensions of this work.

2. State of the Art

In die-casting products there are various types of surface defects that can be observed. Based on NADCA's standard report [10], the most common surface defects are (a) cold flow, (b) cold lap, (c) chill, (d) non-fill, (e) swirls, etc. Some of the causes for these surface defects

might be the low metal temperature, cold dies or lack of venting. Subsequently, some research works are briefly described regarding the problem of quality control in industrial parts that are made by casting-based processes.

A methodology for real-time defect detection in products manufactured by cast extrusion process was developed [11] using refraction of a collimated light source referred to as Mie light scattering. Then, a system was used to implement blob analysis aimed at detecting the contrasting dark areas of the defects. Blobs are a group of contiguous pixels with the same intensity, which could be caused by shadowing from defects. Based on the results of the proposed methodology, the accuracy was 90% or higher depending on the examined defect.

Subsequently, a methodology for locating surface defects in cast aluminium and cast iron components was proposed [12]. In the first stage, it was necessary to photograph the cast parts under suitable lighting conditions and then to preprocess them, so as to reduce noise and enhance quality. Image binarization followed, i.e., conversion from gray scale to black and white. Next, blob analysis was used to identify connected areas of pixels within an image, using the intensity information of the pixels. In this way, it was possible to find various types of surface defects, such as cracks, inclusions, spots, shrinkages, holes and porosity.

An inspection system was proposed [13] for the identification of surface defects in casting products, based on digital images analyzed with a hybrid processing algorithm. In the first stage, the basic parameters were extracted through a genetic algorithm and subsequently morphological operations were employed, yielding 99% accuracy for the defects examined.

A support vector machine (SVM) classifier was suggested to categorize defective images of strongly reflected metals [14]. Specifically, the products were photographed and a wavelet transform was applied to reduce noise. Subsequently, the digital images were converted into binary based on the Otsu threshold and, subsequently, the main characteristics in the spectral domain were extracted. These features were used as input to the SVM model, which classified the corresponding surface defects with an 85% success rate.

In [15–17], a number of ML algorithms were proposed for the automated inspection of surface quality in castings. In these specific publications, the part images were captured by a laser camera with 3D technology. Subsequently, certain features were extracted and used to train ML algorithms, such as Bayesian networks, SVM model, decision trees and, finally, the k-nearest neighbor algorithm. The methodology followed led to accuracy higher than 90% for all the models employed.

A comparison of different classifiers, such as RBF-SVM, BP and polynomial kernel SVM was conducted by the authors [18]. Their investigation focused on images taken by a camera for identifying scratches, black spots, and holes by employing a geometrical feature extraction algorithm. Following the analysis, the authors achieved an average recognition rate of 96%, indicating the effectiveness of the designed supervised learning classifier.

The authors of [19] used an advanced image-processing algorithm based on a modified Laplacian of Gaussian edge detection method and an advanced lighting system so as to extract important features from each image. Subsequently, a neural network was trained for the defect classification, achieving a success rate equal to 90%.

In more recent studies, convolutional neural networks (CNNs) for image processing have often been employed, since the extraction of the important features is automatically achieved through the deep architecture of such networks. Specifically, an 11-layer CNN was employed [20], achieving the classification of defective parts with an accuracy of 99.8%. Corresponding approaches were also followed for surface defect classification [21,22], with minor variations in the neural network architectures and the respective accuracies.

Other publications employed more innovative versions of CNNs, which automatically achieved surface defects recognition. More specifically, the authors of [23] examined the use of the single-shot detector (SSD) machine-vision algorithm for recognizing surface

defects in large-scale steel parts. The proposed method led to very satisfactory results with a mAP metric higher than 65% for both datasets examined.

The use of an improved Faster R-CNN architecture for defect detection in engine blocks and heads was proposed [24] in which scratches and sand inclusion could be observed. The accuracy of this approach reached 92%. Moreover, based on the architecture of Faster R-CNN, the authors of [25] detected defects in wheel hubs. The results were encouraging, with mAP values equal to 73% for the four types of defects examined. Finally, the Centernet model was employed by the authors of [26] in order to achieve the surface defect detection of metal castings. By expanding the dataset with various image-processing techniques, the developed model achieved a high mAP of over 0.9.

The aforementioned state-of-the-art techniques, including the investigated defect types, data collection/image processing techniques, and recognition approaches, are summarized in Table 1.

Table 1. Summarized state-of-the-art techniques.

References	Investigated Defect Types	Data Collection/Image Processing	Recognition Approach
[11,12]	Wrinkles, voids/pinholes, streaks/cold shuts	Photos under controlled lighting conditions/histogram, thresholding and particle analysis	Blob analysis
[13]	Various types	Part presented to camera by a robot/image parameters extracted by GA	Gray scale mathematical morphology
[14]	Burrs, oil stains, scratches, pits, perforations, peelings	Wavelet-based decomposition and denoising, thresholding and spectral feature extraction	SVM
[15–17]	Inclusions, cold laps, misruns	Robotic mounted laser camera for 3D representation/filtering, image segmentation and height maps	Bayesian ANNs, SVM, decision trees, k-NN
[18]	Scratches, black spots, holes	Photos under controlled conditions/geometrical feature extraction	Comparison between RBF-SVM, BP, polynomial kernel SVM
[19]	Blowholes, shrinkage porosity, shrinkage cavity	Photos under double illumination conditions/geometrical features extraction	ANN
[20]	Texture-related defects	DAGM public dataset	CNN

Table 1. *Cont.*

References	Investigated Defect Types	Data Collection/Image Processing	Recognition Approach
[21]	Scars, scratches, inclusions, seams	Image cropping and resizing	GoogLeNet
[22]	Scratches, bumps, foreign bodies	Image resizing, random rotation, mirroring and other operations to deal with imbalanced classes	ResNet18
[23]	Creases, inclusions, oil spots, pits, scratches	Comparison in 2 different public datasets (NEU-DET, GC10-DET)	SSD
[24]	Scratch, sand inclusions	Images taken from real production/image resizing and labelling	Faster R-CNN
[25]	Scratches, oil stains, blocks, grinning	Images taken from real production/added various types of noise for augmentation, labelling	Faster R-CNN
[26]	Scratches, pits	Image resizing, random image cropping, scaling and rotation, along with brightness/contrast adjustments	Pretrained Centernet (COCO dataset)

This paper makes a notable contribution by addressing the early stage of research in automating quality control for high-pressure die-casting products, where scientific publications on this topic are limited. Leveraging transfer learning from pretrained models on a larger dataset, the study investigates the potential improvement of surface defect detection in a smaller dataset. Additionally, the paper provides a systematic examination of the real-time deployment potential of neural networks for this specific purpose by comparing four different computational resources. A systematic flowchart of the methodology is depicted in Figure 1.

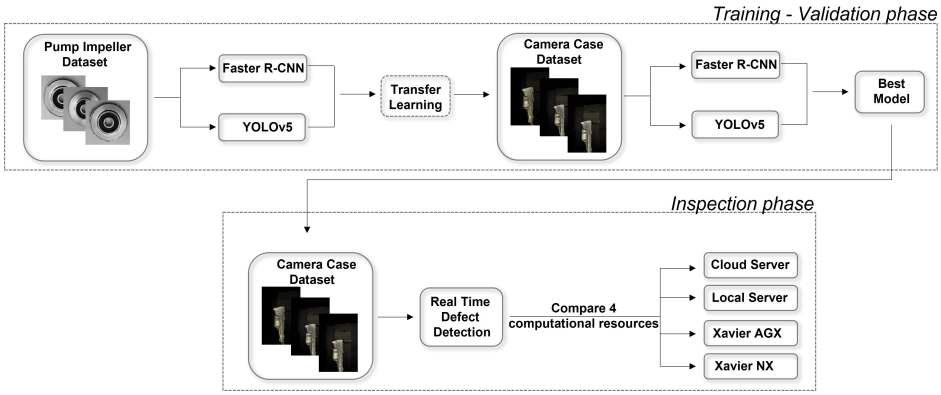


Figure 1. Systematic flowchart.

3. Image-Based Defect Detection Neural Networks

3.1. Faster R-CNN

The first detection network employed is the Faster R-CNN [27]. This network consists of three basic blocks: (a) the backbone network, (b) the region proposal network (RPN), and (c) the box head, which are briefly presented below. It should be noted that the networks RPN and box head are known as Fast R-CNN, as are parts of the previous version of this network.

3.1.1. Backbone Network

This network is the basis of the entire architecture and is responsible for creating feature maps from the original input image. Various architectures can be used as a backbone network, such as the feature pyramid network (FPN) [28], C4, DC5, etc. Focusing on the FPN, this network receives as input an image of dimensions $H \times W$ and then, using residual networks (ResNets) [29,30] and MaxPooling Layers, a total of five feature maps of the original image are created at different scales, thus ensuring that the network will be able to detect objects independent of size.

3.1.2. Region Proposal Network

The second part of Faster R-CNN is the RPN. Essentially, this network is responsible for extracting the regions of interest based on the feature maps of the backbone network. These feature maps are convoluted with suitable kernels and, for each map, the following are predicted:

- pred_objectness_logits: probability of object existence (0 or 1),
- pred_anchor_deltas: box shape containing the detected object.

The next step is the creation of the anchors, which are boxes that will be adjusted to each of the five feature maps of the image. The goal is to find those anchors associated with the true boxes of each image. For this reason, it is required to calculate the intersection over union (IoU) metric between all anchors and ground truths in order to retain those anchors with IoU higher than a threshold. Those anchors are marked with the label 1 (foreground) as they closely approximate the real boxes, while the anchors for which $0.3 <$ IoU $<$ threshold are labeled with 0 (background). In addition, the network is responsible for correcting the dimensions of these anchors, predicting four parameters $\Delta x, \Delta y, \Delta w$ and Δh, referring to the position, width and height of the anchor, respectively.

Subsequently, the cost function is estimated to extract the error. Specifically, the following two types of error are taken into account:

- Objectness loss function: L_{cls}. In this case, the binary cross-entropy function is used, estimated for all anchors labeled with 0 and 1.
- Localization loss function: L_{reg}. For the second case, the L_1 cost function is used only for the anchors with label 1.

According to [27], the cost function is defined based on Equation (1):

$$L(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \quad (1)$$

where the index i denotes a random anchor, p_i the predicted probability of i anchor containing an object (pred_objectness_logits), and p_i^* is the true probability of the anchor to contain an object (ground truth). In addition, t_i defines the predicted dimensions of the anchor and t_i^* the corresponding ground truth. For the regression cost, the existence of the term $p_i^* L_{reg}$ is observed, which ensures that only the anchors with label 1 are used. Moreover, the parameters N_{cls} and N_{reg} refer to the size of the mini-batch and the total number of anchors, respectively. Finally, using the non-maximum suppression algorithm, a certain number of anchors (e.g., 1000) is maintained, which finally represents the candidate regions of interest.

3.1.3. ROI (Box) Head

In the last part of Faster R-CNN, the following are taken as inputs:

- The five extracted FPN feature maps,
- The candidate interest boxes (proposal boxes) resulting from the RPN, each of which is labeled with 0 or 1,
- The real boxes of the dataset (ground truths).

Subsequently, the candidate regions of interest are sampled to ensure a population balance between the background (label 0) and the foreground (label 1). Then, the regions of interest are cropped from each feature map based on the proposal boxes retained.

After this process, the cropped regions are fed to the head network, which categorizes the object in that region and modifies the proposed box accordingly. This network consists of two fully connected networks, predicting the probabilities of each class (score) and the correction factors of the boxes. To achieve this, the definition of the objective function is obviously required, based on which this network will be trained. Specifically, the following two types of loss functions are used:

- Classification loss. In this case, the softmax function is used, estimating the probabilities of each class for all candidate boxes labeled with 0 and 1 (background and foreground).
- Localization loss. For this case, the L_1 cost function is used only for the candidate boxes designated as foreground.

A schematic diagram of the Faster R-CNN is depicted in Figure 2.

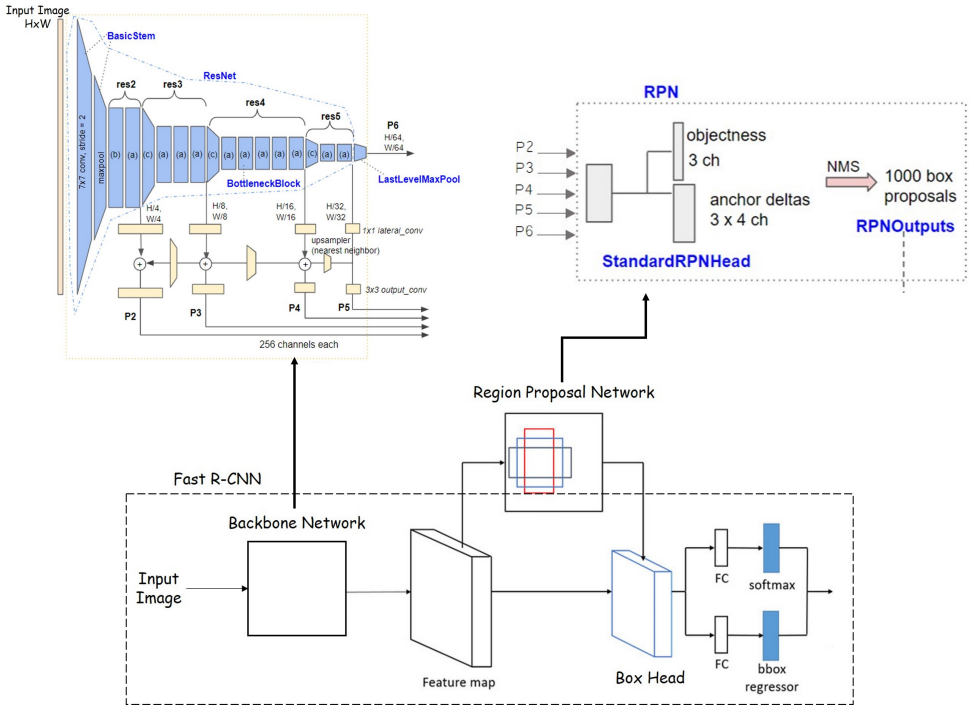


Figure 2. Faster R-CNN architecture. Adapted from [31,32] with modifications.

3.1.4. Faster R-CNN Training Algorithm

The training phase of Faster R-CNN in this paper is based on the alternating training method. Specifically, the RPN network is trained first and then follows the Fast R-CNN. Then, the Fast R-CNN weights are used to initialize the RPN weights and the training starts again. This procedure is followed until convergence of the weights between RPN and Fast R-CNN occurs.

3.2. YOLOv5 Network

The second network examined in the present paper belongs to the category of YOLO networks: You Only Look Once [33], which is a very popular model for solving such problems because of the speed and accuracy it offers. To date, various optimizations (YOLO9000, YOLOv2, etc.) have been proposed based on this architecture, increasing the accuracy and concurrently reducing the inference time of the model [34–36].

In this paper, YOLOv5 is employed provided by the company Ultralytics [37], the architecture of which is somehow similar to the YOLOv4 network, with the notable difference that, instead of Darknet software, the Python language was used for the development. The architecture of the YOLOv5 network can be divided into the following three parts: backbone, neck and head. The cross-stage partial connections (CSP) Darknet53 network [38] is used as a backbone, which is a CNN based on the logic of the DenseNet network [39].

In the next stage, namely the neck part, the spatial pyramid pooling (SPP) [40] layer and a modification of the path aggregation network (PANet) [41] are used. The SPP layer is generally used at the end of CNNs in order to extract fixed-size features regardless of the dimensionality of the input data. The PANet is an architecture that aims to more effectively disseminate information from the first layers to the next. In this way, low-level features (edges, corners, etc.) are propagated to layers that are responsible for extracting high-level features. This combination seems to be of crucial importance for the accurate localization of objects in this type of network.

The final part of this network is the head, which is composed of three convolution layers responsible for predicting the output vector that consists of the boxes corrections, the confidence score and the probabilities of each class in each of the feature maps entered by the neck part. A schematic diagram of the YOLOv5 is depicted in Figure 3.

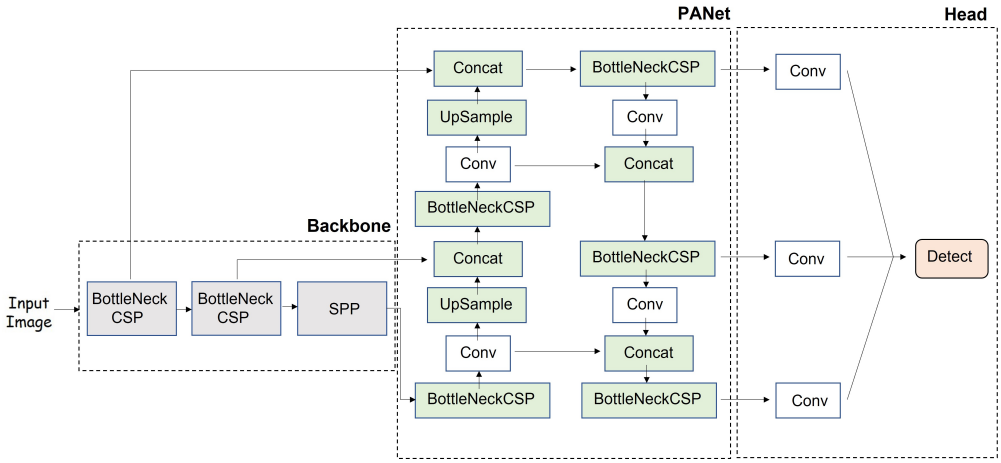


Figure 3. YOLOv5 architecture. Adapted from [42] with modifications.

A new addition to YOLOv5 is the introduction of a modified cost function on the location error of predicted boxes. Specifically, a complete intersection over union (CIoU) [43] quantity is employed, which is defined according to Equation (2). In this relationship, b denotes the coordinates of the center (x, y), d^2 the Euclidean distance, and c the diagonal length of the smallest enclosing box covering the two boxes. Furthermore, considering w and h as the width and height of a box, based on [43], the terms v and α are given by Equations (3) and (4), respectively.

$$L_{CIoU} = 1 - IoU(pred, groundtruth) + \frac{d^2[b_{pred}, b_{groundtruth}]}{l^2} + \alpha v, \quad (2)$$

$$v = \frac{4}{\pi^2}(arctan\frac{w_{groundtruth}}{h_{groundtruth}} - arctan\frac{w_{pred}}{h_{pred}})^2 \quad (3)$$

$$\alpha = \frac{v}{(1 - IoU) + v} \tag{4}$$

The total loss for YOLOv5 is estimated by Equation (5) and consists of three parts: classes loss (L_{cls}), objectness loss (L_{obj}) and location loss (L_{loc}). For the first two losses, the binary cross-entropy (BCE) is employed, while the location loss [37] is given by Equation (5).

$$Loss = \lambda_1 L_{cls} + \lambda_2 L_{obj} + \lambda_3 L_{loc} \tag{5}$$

As mentioned above, the development of YOLOv5 is written in the Python programming language and is specifically based on the PyTorch library [44]. However, this fifth release does not come with a published paper, but the code is freely available on Github [42].

4. Networks Set Up

4.1. Datasets Acquisition and Preparation

4.1.1. Pump Impeller Dataset

As mentioned in Section 1, two datasets are studied in the present paper. The first concerns photos depicting submersible pump impellers, manufactured by Pilot TechnoCast, India. These images are freely available on Kaggle [45], including a total of 1300 images without data augmentation, 781 of which refer to defective parts and the remaining 519 to healthy ones. The dimensions of all photos are 512×512 and are displayed in gray scale. In addition, a specific label corresponds to each image, meaning that it is known which photos refer to defective parts and which to healthy ones.

However, a more detailed categorization of the defects has not been carried out for each image and there is no information regarding the types of defects depicted. Thus, in the present work, the defects considered, observing each image separately, are manually categorized and labelled as follows:

- Roughness, denoting the excess metal on the outer perimeter of the pump,
- Holes, denoting the existence of holes observed on the surface of the pump or lack of material on the outer perimeter,
- Spots, that is, the stains in the surface of the pumps, and, finally,
- Creases, that is, cracks in the surface of the pumps.

As an example, Figure 4 showcases three photos, with (a) showing a healthy part and all the others defective ones. Based on the above information, Table 2 also provides a summary of the pump impeller dataset.

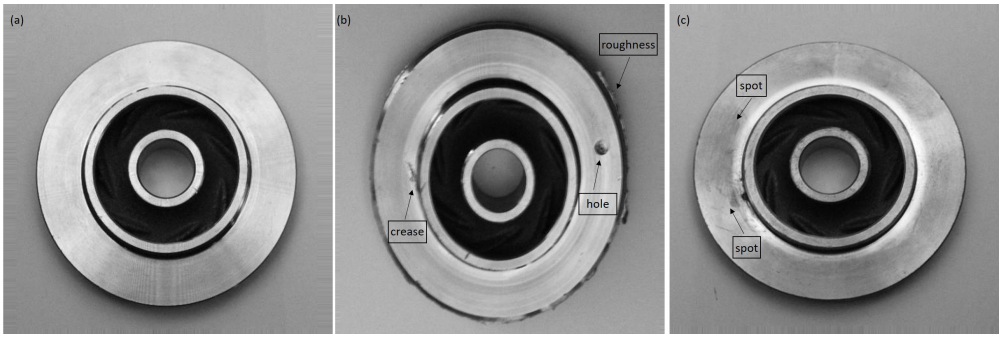


Figure 4. Pump impeller dataset: (**a**) healthy, (**b**) roughness, crease, hole and (**c**) spot [45].

Table 2. Summary of the Pump Impeller dataset.

Dataset Information	Value
Number of Images	1300
Image Dimensions	512 × 512
Image Type	Grayscale
Manufacturer	Pilot TechnoCast, Veraval, India
Availability	Kaggle
Healthy and Defective Parts	
Healthy Images	519
Defective Images	781
Defect Categories	
Roughness	Excess metal on the outer perimeter
Holes	Holes or lack of material on the outer perimeter
Spots	Stains on the surface
Creases	Cracks on the surface

4.1.2. Automotive Camera Case Dataset

The second dataset examined corresponds to a special car camera case, manufactured by Vioral S.A., Greece [46]. Specifically, it is a mechanical part that is located under the mirror inside the cabin of a car and serves both to control pedestrian traffic and to brake automatically in the event of an accident. The manufacturing method is high-pressure die-casting and the raw material is aluminum AC 46000 (79.7–90% Al, 8.0–11.0% Si, 2.0–4.0% Cu, 0–1.3% Fe, 0–1.2% Zn). A Canon EOS M camera was used to capture the images at a distance of 245 mm from each part, the technical specifications of which are presented in Table 3. During the experiments, the parts were held in a special jig with spacers and, in order to ensure appropriate lighting, a softbox with dimensions of 50 × 70 cm was used at a distance of 695 mm from each part.

Table 3. Digital camera specifications.

Analysis Pixels	Lens	Sensor	Shutter Speed	Aperture
5184 × 3456	EF M22	APS-C CROP 35 mm	1/5	f/11 η f/8

In Figure 5, three images are presented indicatively, (a) corresponding to a healthy part, while (b), (c) correspond to defective ones. As depicted, two specific types of defects are examined on the side of the parts, which are: (i) cold lap and (ii) shrinkage. The term cold lap refers to the cracks on the surface of the parts, while the term shrinkage refers to the points of local shrinkage, i.e., where a dent is observed due to the solidification process. In total, the available data amounts to 118 photos, 13 of which are healthy whilst the remaining 105 are defective. The details of the camera case dataset are summarised in Table 4.

Before training the detection networks, image segmentation is required, that is, the definition of boxes enclosing the objects of the images to be detected. However, such boxes are not available in either of the two datasets and, thus, in the present work, the segmentation process is achieved through the free platform Apeer [47].

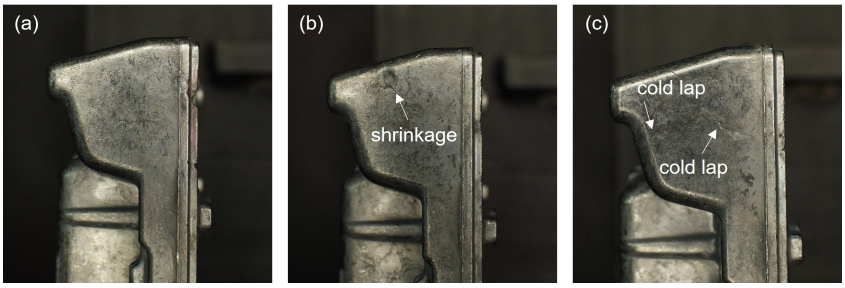


Figure 5. Camera case dataset: (**a**) healthy, (**b**) shrinkage and (**c**) cold lap.

Table 4. Summary of the Camera Case Dataset.

Dataset Information	Value
Number of Images	118
Image Dimensions	5184 × 3456
Image Type	Grayscale
Manufacturer	Vioral S.A., Aspropirgos, Greece
Healthy and Defective Parts	
Healthy Images	13
Defective Images	105
Defect Categories	
154 Cold Laps	Cracks on the surface of the parts
135 Shrinkages	Dents due to the solidification process

4.2. Detection Networks Hyperparameters

Regarding the pump impeller dataset, no normalization is performed and the images are imported directly into the networks. Additionally, to reduce computational cost, not all available pump images are used. Specifically, a total of 400 photos were selected, 300 of which belonged to the defective parts, while the remaining 100 belonged to the healthy ones. Regarding the camera case dataset, in order to reduce the computational cost, the dimensions of the images were reduced from 5184 × 3456 to 2592 × 1728 pixels. Finally, it should be mentioned that all available images were used (13 healthy 105 defective, which included 154 cold laps and 135 shrinkages).

Another critical hyperparameter that may significantly affect the performance of the models is the initialization of the networks' weights. So, the following methodology was followed:

- Pump impeller dataset: to train the networks of this dataset, the pretrained models in the COCO dataset [48] were used for the weights' initialization. It is one of the largest labeled datasets, consisting of thousands of images depicting objects of 80 different classes.
- Camera case dataset: to train the networks of this dataset, the pretrained models in the pump impeller dataset were employed, thus implementing the transfer learning procedure. Moreover, the results of this procedure were then compared with the case of using the pretrained models in the COCO dataset for the weights' initialization.

The two datasets were split into training, validation and test sets. The percentages of the pump impeller dataset were: (a) training set: 80% (320 photos), (b) validation set: 9% (35 photos), and (c) test set 11% (45 photos). At this point, it should be mentioned that, for the pump impeller dataset, a new class named background was introduced. During the segmentation process, this class was represented by a box placed on a characteristic point that appeared in all photos (healthy and defective). So, during the inspection phase, if the

networks predicted only the background box for a new image, this meant that the specific part had been categorized as healthy.

Accordingly, for the camera case dataset, the values selected were: (a) training set: 65% (76 damaged photos, including 138 cold laps and 110 shrinkages), (b) validation set: 15% (17 damaged photos, including 6 cold laps and 10 shrinkages), and (c) test set: 20% (25 photos: 13 healthy and 12 damaged, including 10 cold laps and 15 shrinkages). It should be noted that, during the segmentation process, boxes were placed only for the two examined defects (shrinkage and cold lap) and, thus, in the training phase the networks were fed only with damaged photos. Therefore, in this case, if the networks did not detect any object in the inspection phase, these images were categorized as healthy.

Finally, some basic hyperparameters selected for Faster R-CNN and YOLOv5 networks, as proposed by the corresponding authors [32,42], are listed in Table 5. The selection of these hyperparameters was also tested with a trial and error procedure, with the goal of maximizing the detection performance of the employed networks.

Table 5. Hyperparameters of Faster R-CNN and YOLOv5 networks.

Network	Optimizer	Learning Rate	Weight Decay	Gamma	Momentum	Number of Epochs
Faster R-CNN	SGD	10^{-3}	10^{-4}	0.05	0.9	20
YOLOv5	SGD	10^{-2}	$5 \cdot 10^{-4}$	0	0.937	150

5. Defect Detection Results

5.1. Pump Impeller Dataset

5.1.1. Faster R-CNN Network

This subsection summarizes the predictions of Faster R-CNN on the pump impeller dataset. Using the 45 images of the test set, Figure 6 depicts the precision–recall curve for each of the four defects, considering a threshold value for the IoU metric equal to 0.5. The area under the curve is known as the average precision and is the key performance evaluation metric. In Table 6, the values of average precision for each class and for different threshold values are presented in order to investigate whether this value affected the performance of the network.

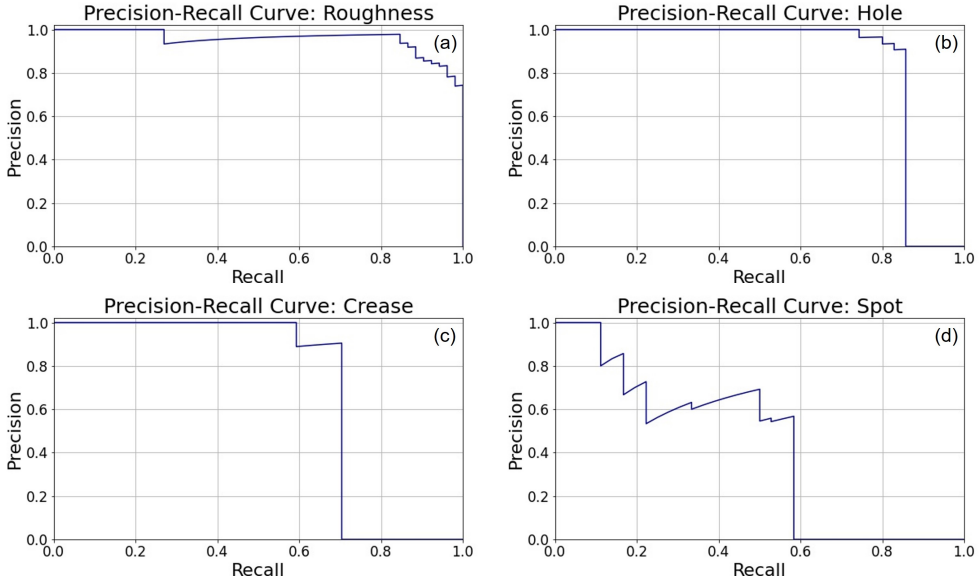


Figure 6. Precision–recall plots of Faster R-CNN for each class of pump impeller test set: (**a**) roughness, (**b**) hole, (**c**) crease, and (**d**) spot for an IoU threshold value equal to 0.5.

According to Table 6, the overall mAP metric equalled 0.77 for a threshold value of 0.5, indicating good performance. The highest AP value was observed for the background class and was equal to 1, meaning that this class was correctly detected in all photos of the dataset. Regarding the classes corresponding to defects, the highest value of the AP was 0.76 for the roughness class, while, correspondingly, the lowest value was observed for the spot class and was equal to 0.63. This result was expected since the roughness defect was the most distinguishable by bare eye, in contrast to the spot defect which was quite indiscernible. Additionally, increasing the threshold value to $IoU \geq 0.75$ or $IoU \geq 0.9$ led to an expected decrease in performance, since the predictions must satisfy a stricter criterion. Specifically, for a threshold equal to 0.75, the value of mAP did not differ significantly and was equal to 0.75, while for a threshold equal to 0.9, the metric was reduced to 0.56.

Table 6. AP values for the pump impeller test set with respect to Faster R-CNN.

IoU Threshold	Background	Roughness	Hole	Crease	Spot	mAP
0.5	1.0	0.76	0.75	0.69	0.63	0.77
0.75	1.0	0.73	0.75	0.69	0.58	0.75
0.9	0.94	0.24	0.60	0.61	0.42	0.56

Figure 7 depicts some predictions of Faster R-CNN for a threshold value equal to 0.5. As shown, the network predicts the coordinates of the boxes and, additionally, for each prediction, a percentage is extracted indicating the confidence of the network that the specific defect is included within the box. In general, it is observed that the network succeeds in identifying, with fairly high accuracy, the various types of defects examined in this dataset.

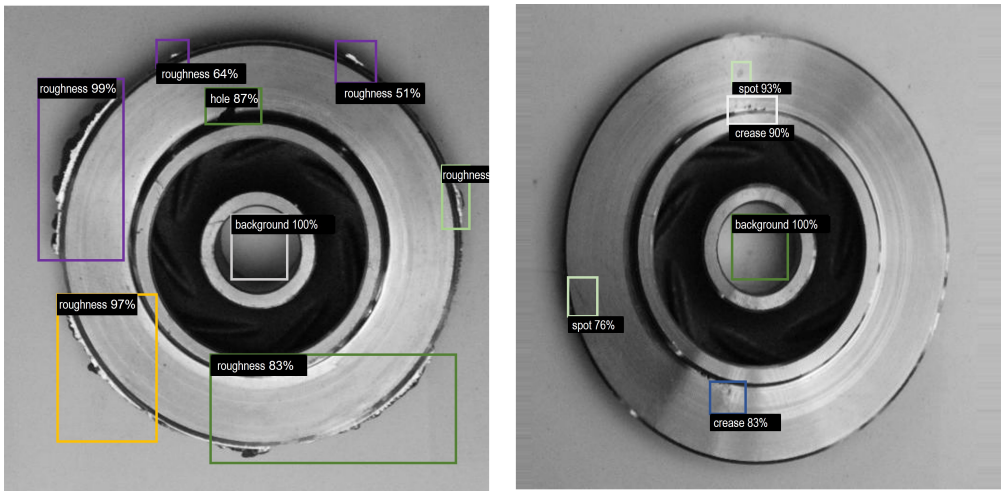


Figure 7. Indicative predictions of Faster R-CNN in the pump impeller test set.

Finally, it should be noted that Faster R-CNN did not predict any boxes indicating the existence of a defect for the photos that actually depicted a healthy product, and, at the same time, detected at least one defect in each defective product. Therefore, it is suitable, in principle, as a quality control tool.

5.1.2. YOLOv5 Network

The next network employed was YOLOv5. In order to evaluate its performance, in Figure 8, the precision–recall curve is depicted for each defect class of the pump impeller test set. Similar to the case of the Faster R-CNN, the value 0.5 was used as a threshold for the extraction of these plots. Based on Table 7, the mAP metric for a threshold value of 0.5 equalled 0.65, which was considered satisfactory. The lowest performance appeared,

as expected, for the spot defect, while, for the rest of the classes, the AP values were quite close, namely 0.59, 0.62 and 0.61 for the roughness, hole and crease defects, respectively. Moreover, increasing the threshold value led to a decrease in model performance; namely, for a threshold equal to 0.9, the mAP metric decreased to 0.45. Overall, these results indicate effective defects detection in the examined mechanical parts, but, compared to the Faster R-CNN, all AP values were lower and, thus, the overall performance of YOLOv5 was comparatively slightly inferior.

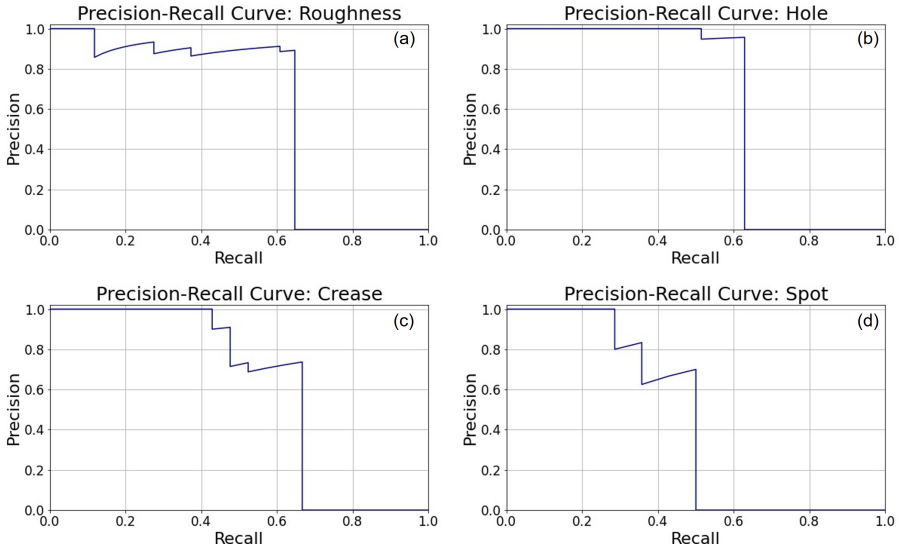


Figure 8. Precision–recall plots of YOLOv5 for each class of pump impeller test set: (**a**) roughness, (**b**) hole, (**c**) crease, and (**d**) spot for an IoU threshold value equal to 0.5.

Table 7. AP values for the pump impeller test set with respect to Yolov5.

IoU Threshold	Background	Roughness	Hole	Crease	Spot	mAP
0.5	1.0	0.59	0.62	0.61	0.44	0.65
0.75	1.0	0.43	0.57	0.62	0.44	0.61
0.9	0.93	0.23	0.45	0.26	0.39	0.45

Figure 9 indicatively showcases two defective photos of the test set and the corresponding predictions of YOLOv5. As depicted, the network detects all the surface defects observed in these parts with high accuracy. Finally, it should be noted that YOLOv5 correctly categorized both defective and healthy parts, proving its feasibility as a defect recognition tool.

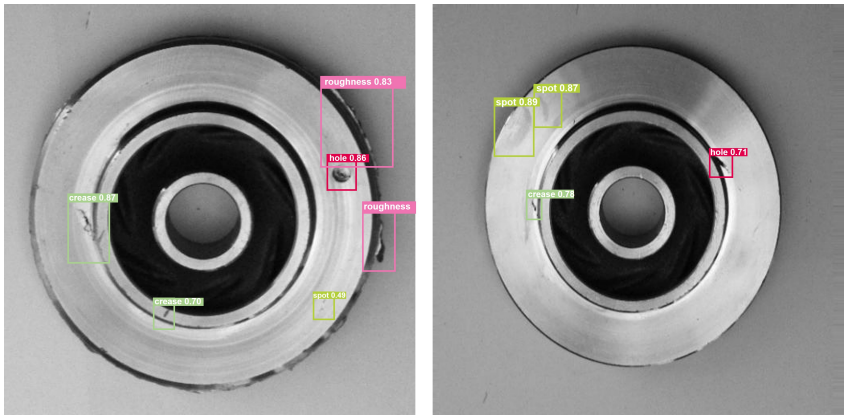


Figure 9. Indicative predictions of the YOLOv5 in the pump impeller test set.

5.2. Camera Case Dataset

5.2.1. Transfer Learning Procedure

This subsection examines whether the models trained in the pump impeller dataset can be employed for the camera case dataset while retaining similar performance characteristics. As mentioned above, this process is based on transfer learning and is used for datasets that consist of few data. Small datasets easily lead to overfitting, i.e., the network achieves correct predictions only on training data but is unable to generalize them to the test set [49].

Therefore, all the weights of the already pretrained Faster R-CNN and YOLOv5 models were used for the initialization of the weights of the two new networks, the architectures of which being the same with those of the two pretrained models. Subsequently, a new training phase for whole architectures took place using the hyperparameters of Table 5, so as to use these networks for surface defect detection in the camera case dataset.

The goal of the transfer learning process is to mitigate the fact that producing real-world casting process image datasets of sufficient size requires considerable investments in resources (equipment, personnel, cost and time). Therefore, using the pump impeller dataset to pretrain the models, and then reusing these models for the camera case dataset, would overcome the size limitation of the latter.

5.2.2. Faster R-CNN Network

The first network evaluated in the camera case dataset was the Faster R-CNN. In Figure 10, the precision–recall curves are presented for the 25 images of the test set regarding the two types of defects with a threshold 0.5. Based on these results, the values of the AP metrics for each class and the mAP value are summarized in Table 8. The results suggest that Faster R-CNN more effectively detected local shrinkages (AP = 0.78) than cold laps (AP = 0.63), which was somehow expected considering that the latter were more indiscernible. Furthermore, the mAP equalled 0.70 for threshold 0.5, while increasing the threshold value to 0.7 or 0.9 yielded reductions in the total performance to 0.64 and 0.60, respectively.

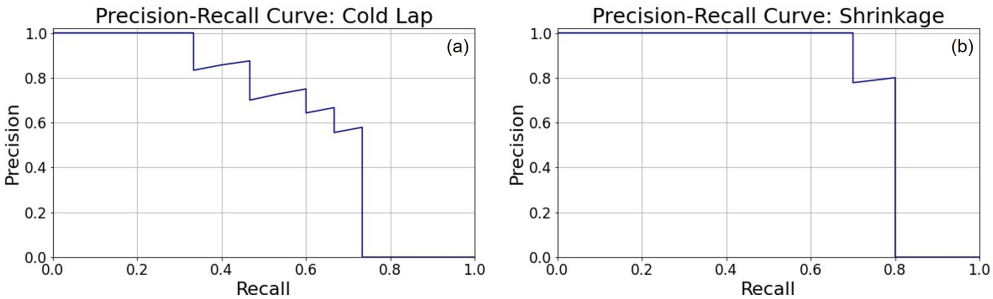


Figure 10. Precision–recall plots of Faster R-CNN for each class of the camera case test set: (**a**) cold lap and (**b**) shrinkage for an IoU threshold value equal to 0.5.

Table 8. AP values for the camera case test set with respect to Faster R-CNN.

IoU Threshold	Cold Lap	Shrinkage	mAP
0.5	0.63	0.78	0.70
0.75	0.63	0.7	0.64
0.9	0.50	0.7	0.60

Figure 11 shows three indicative randomly defective parts of the test set along with the corresponding predictions of the network, where it is depicted that the examined defects have been recognized effectively. Overall, from the 25 defects of the test set, 7 out of 10 cold laps and 13 out of 15 shrinkages were categorized correctly (i.e., 20 out of 25 defects were correctly categorized), yielding a correct classification rate of defects that equalled 80%.

The correct classification rate of the photos of the test set was also examined for the Faster R-CNN. In total, 11 out of 13 photos corresponding to healthy parts were categorized correctly and, thus, the false alarm rate for this network was 15%. Moreover, Faster R-CNN detected at least one defect in all images depicting defective parts, yielding a correct classification rate of defective products equal to 100%. Overall, the performance of the Faster R-CNN network in image classification was considered quite satisfactory since, on the one hand, it correctly categorized all the defective parts, and, on the other hand, the percentage of healthy products that were incorrectly categorized as damaged was quite small.

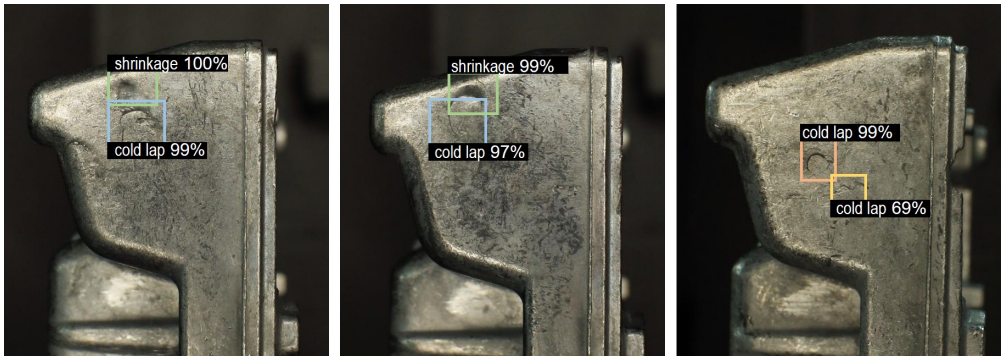


Figure 11. Indicative predictions of Faster R-CNN in the camera case test set.

5.2.3. YOLOv5 Network

The next network evaluated in the camera case dataset was YOLOv5. In Figure 12 the precision–recall curves for a threshold value of 0.5 are shown and, in Table 9, the corresponding mAP values are presented for different thresholds. Based on these results, the shrinkage defects were again detected with greater efficiency (AP = 0.67) compared to cold laps (AP = 0.53). Overall, the mAP metric was 0.60, which was slightly reduced compared to Faster R-CNN.

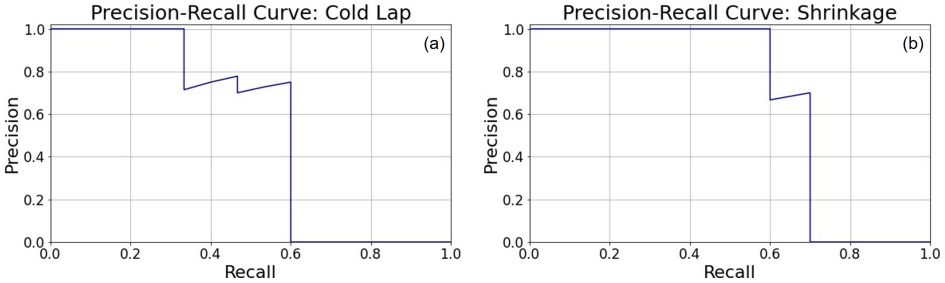


Figure 12. Precision–recall plots of YOLOv5 for each class of camera case test set: (**a**) cold lap and (**b**) shrinkage for an IoU threshold value equal to 0.5.

Table 9. AP values for the camera case test set with respect to Yolov5.

IoU Threshold	Cold Lap	Shrinkage	mAP
0.5	0.53	0.67	0.60
0.75	0.53	0.67	0.60
0.9	0.33	0.67	0.50

Figure 13 showcases three indicative photos of the test set combined with the predictions of YOLOv5. Regarding the probabilities extracted by the network, it was found that they had a lower value than their counterparts for Faster R-CNN, which was linked to the reduced values of the mAPs. Overall, from the 25 defects of the test set, 6 out of 10 cold laps and 11 out of 15 shrinkages were categorized correctly (i.e., 17 out of 25 defects were correctly categorized), yielding a correct classification rate of the defects equal to 68%.

Finally, examining the image classification problem, from the 25 photos of test set (13 healthy and 12 damaged), only one was wrongly categorized in each state. Therefore, the false alarm rate was 7.7%, which was lower compared to that of Faster R-CNN. Moreover, the percentage of correct classification of defective parts was also lower, namely, equal to 91.7%, given that one defective part was wrongly categorized as healthy.

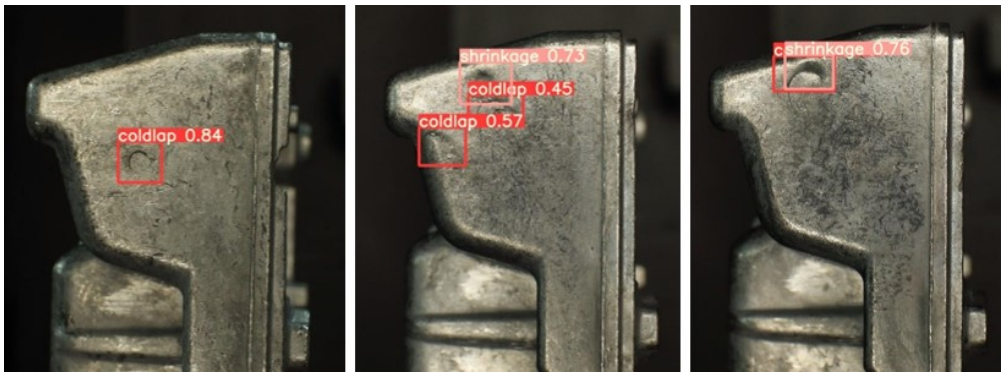


Figure 13. Indicative predictions of YOLOv5 in the camera case test set.

5.2.4. Models Comparison

Subsequently, a performance comparison for the camera case dataset was performed between the following cases: (a) training the Faster R-CNN and YOLOv5 networks using the pretrained models of the COCO dataset for the weights' initialization, and (b) training the networks using the respective pretrained models of the pump impeller set. In this way, it was determined whether the transfer learning from the pump impeller to the camera case dataset could lead to higher performance, given that both datasets depicted defects in the die-casting mechanical parts.

In this context, Figure 14 compares the AP metrics for the two types of defects (cold lap and shrinkage), as well as the mAP for the two cases mentioned above. As depicted, the use of the pretrained models based on the Kaggle dataset led to improved performance compared to the COCO dataset. Specifically, for the Faster R-CNN, the increase in mAP equalled 6%, while, for the YOLOv5, a smaller mAP increase was observed, equal to 5%. These results indicate the importance of transfer learning between datasets depicting roughly similar objects for detection.

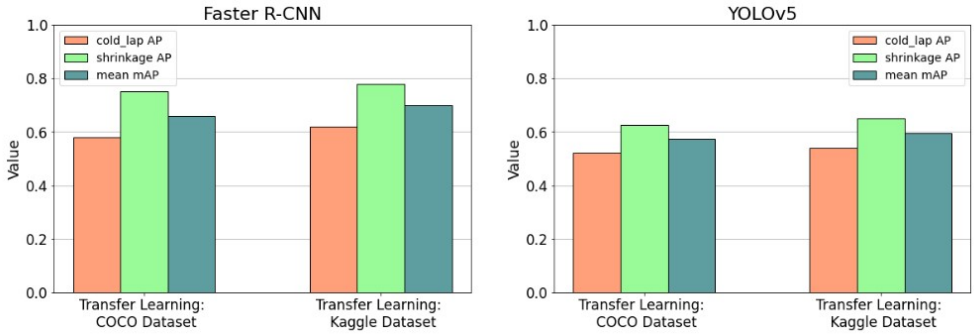


Figure 14. Performance comparison for different weights initializations of the networks: (**a**) Faster R-CNN and (**b**) YOLOv5.

6. Computational Resource Comparison

In this section, different computational resources are considered in order to determine whether the predictions in the camera case dataset can be completed in real time using the optimum Faster R-CNN of Section 5.2. In this context, two different computing approaches are compared, that is: cloud computing and edge computing.

- Cloud computing: the process of providing access to a cloud server, which runs all the results in a cloud computing environment [50,51]. Executing an algorithm in the cloud obviously increases performance due to high computing capacity, but the overhead of transferring the data (cloud offloading) and the high costs should be taken into account.
- Edge computing: process in which data is analyzed at the "edge" of the network, i.e., (a) near or (b) exactly where it is collected, using devices called edge devices [52]. In the first case, an edge device is considered an edge server to which it is necessary to send data for processing (edge offloading). In contrast, in the second case, the calculations are carried out exactly where the data is generated using embedded devices (e.g., embedded GPUs, FPGAs) and, therefore, offloading costs are avoided.

In this context, four different devices were employed, which were: a cloud server, a local server and two embedded Jetson Boards (NVIDIA Xavier AGX and NVIDIA Xavier NX). Table 10 summarizes the technical characteristics of the computational resources examined during the inspection phase, with the aim of selecting the more efficient devices in terms of speed and power consumption. Finally, regarding the Xavier AGX and NX devices, a bash script was utilized to read the power values from a specific file of the embedded devices at regular intervals. The corresponding power consumption of the NVIDIA GPUs of the cloud and local server was determined using the "nvidia-smi" command.

Table 10. Technical characteristics of computational resources.

Computational Resource	CPU	GPU
Cloud Server	Intel Xeon @ 2.00GHz	NVIDIA Tesla V100
Local Server	Intel(R) Xeon(R) @ 2.20GHz	NVIDIA Tesla T4
NVIDIA Xavier AGX	8-core Carmel ARM v8.2	512-core NVIDIA Volta
NVIDIA Xavier NX	6-core Carmel ARM v8.2	384-core NVIDIA Volta

6.1. Inference Time Comparison

Using the 25 images of the camera case test set, the average inference times for a new prediction of the Faster R-CNN model are presented in Table 11. As observed, the prediction for a new image was completed in under 10 s in all four devices, which is very important for this specific application since quality control is desired to be achieved in almost real time. As expected, the fastest execution of the algorithm was provided by the cloud server,

followed by the local server and then the two embedded systems. Furthermore, based on these results, it was observed that the average prediction time per image decreased in the case that the network was fed with more than one photo. For example, in the cloud server, the expected response time for 25 photos would be 9.5 s (i.e., 25 images × 0.38 s/image), but it turned out to be 4.40.

Table 11. Average inference times [in s] of Faster R-CNN for the camera case test.

	Cloud Server	Local Server	Xavier AGX	Xavier NX
per image	0.38	0.43	4.69	6.61
batch = 5	1.05	2.17	13.75	21.86
batch = 10	1.88	2.96	24.87	40.37
batch = 25	4.40	7.62	58.44	96.68

Finally, the end-to-end latency, consisting of the offloading time and the model's inference time, was examined. Based on Figure 15, the lowest total response time was observed for the local server, for which the offloading time was almost equal to the inference time and, on average, they summed to less than 1 s per image. As for the cloud server, although the inference time was lower than for the local server, the total response time was 2.3 s. This was due to the fact that the time for uploading data to the cloud was much longer than the corresponding transfer to an edge device. Finally, the Xavier AGX and NX accelerators led to longer average inference times per image, i.e., 4.69 s and 6.61 s, respectively, without requiring the transfer of each image to another device.

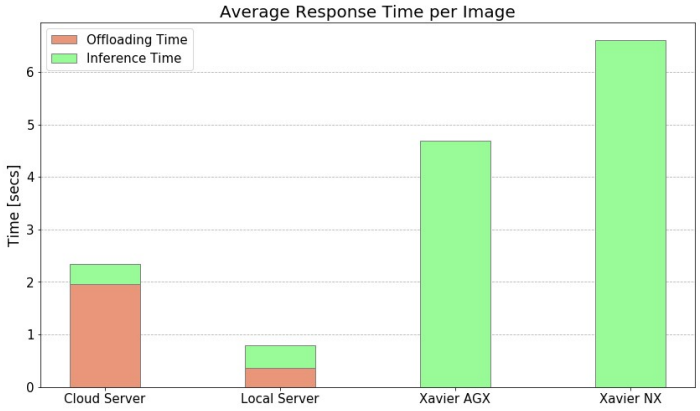


Figure 15. Comparison of total response time per image.

6.2. Power Consumption Comparison

Another important issue concerns the power consumed by each computational resource during the inspection phase of the camera case test set. For this purpose, Figure 16 shows the energy consumption during the inspection phase for the Faster R-CNN, using only one image at a time. As depicted, the maximum power value of the two integrated systems approximated 6 Watts, the cloud server 250 Watts, and the local server 75 Watts. Thus, it is obvious that the two integrated systems had very low energy consumption, reducing the maximum power by 98% and 92% compared to the cloud server and the local server, respectively.

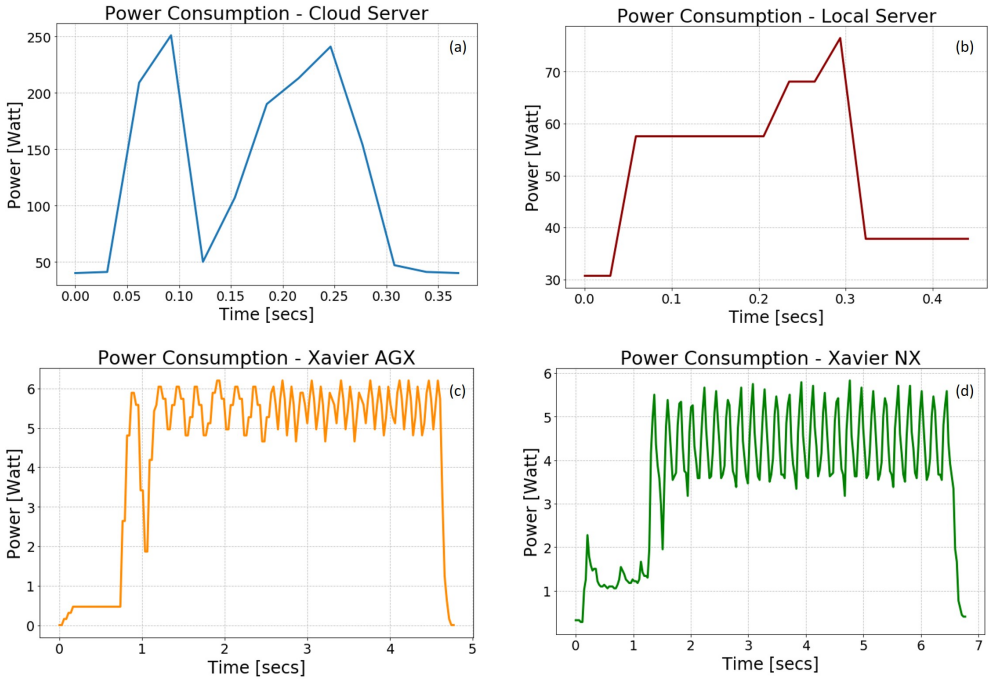


Figure 16. Power consumption of: (**a**) Cloud Server, (**b**) Local Server, (**c**) Xavier AGX and (**d**) Xavier NX for the camera case test set.

7. Conclusions and Future Work

This paper studied defect detection in the surface of mechanical parts produced in series by die-casting using innovative machine vision algorithms. Through the exploitation of a transfer learning procedure, the aim was to develop a system that would accomplish quality control of the products in real time, so as to integrate this system in the industrial production system.

The conclusions drawn for each of the objectives are:

- Regarding the defect detection in the pump impeller dataset, the results of the networks can be considered to be very satisfactory. Specifically, the mAP metrics were 0.77 and 0.65 for the Faster R-CNN and YOLOv5 networks, respectively. These networks also managed to detect at least one defect in all defective parts of the test set, while, at the same time, no false alarms were observed and, therefore, all healthy parts were correctly categorized.
- Regarding the camera case dataset, the mAP values of the Faster R-CNN and YOLOv5 were 0.70 and 0.60, respectively. The corresponding defect classification rates were 80% and 68% for the two networks, respectively. Compared to the previous dataset, the performance was slightly lower but was still satisfactory. This was mainly due to the fact that training detection networks requires a very large amount of data, which were unavailable for this dataset. Moreover, regarding the image classification problem, Faster R-CNN was able to detect at least one defect in the camera case photos that were defective, thus leading to a correct classification rate of defective parts equal to 100%. The corresponding value for the YOLOv5 network was slightly lower and equal to 92%.
- Training the networks on the camera case dataset using transfer learning from the pre-trained models of the pump impeller dataset led to improved performance compared to the case of using the COCO dataset. This indicates that the necessary information was successfully transferred from one dataset to the other, which was desirable since both datasets included images of die-casting mechanical parts.
- Regarding the response times of Faster R-CNN for the camera case dataset, it was found that all four different devices tested could ensure surface defect detection on a part in real time. The fastest total response was provided by the local server, with an

average inspection time per image of 0.82 s, followed by the cloud server, with 2.3 s, and NVIDIA's embedded systems, with 4.69 and 6.61 s.
- The maximum value of the power consumed by the two integrated systems (Xavier AGX and NX) during the inspection phase equalled 6 Watts, i.e., a reduction of more than 92% compared to the maximum energy required by the cloud and local servers.

As a future extension, a detection transformer (DETR) [53] network could be employed in order to examine whether the detection performance can be further improved. Finally, an important extension would be the implementation of the networks in FPGA devices, with the aim of reducing the inspection time of the surface defect detection to a minimum. This idea represents a difficult and demanding undertaking; howeve, the contribution to the production process would be highly beneficial.

Author Contributions: Conceptualization, G.A., A.M., D.M., P.B., G.-C.V. and D.S.; methodology, G.A., A.M., D.M., P.B., G.-C.V. and D.S.; software, G.A., A.M. and D.M.; data curation, G.A. and A.M.; writing—original draft preparation, G.A. and A.M.; writing—review and editing, D.M., P.B., G.-C.V. and D.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Camera case dataset available on request, due to restriction, on a case-specific basis.

Acknowledgments: VIORAL SA is acknowledged for providing the specimens of the second dataset. Kostas Kerasiotis of NTUA Manufacturing Technology Laboratory is gratefully acknowledged for creating the corresponding photo dataset.

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

References

1. Akundi, A.; Reyna, M. A machine vision based automated quality control system for product dimensional analysis. *Procedia Comput. Sci.* **2021**, *185*, 127–134. [CrossRef]
2. Michiels, S.; Schryver, C.D.; Houthuys, L.; Vogeler, F.; Desplentere, F. Machine learning for automated quality control in injection moulding manufacturing. *arXiv* **2022**, arXiv:2206.15285.
3. Ida, N.; Meyendorf, N. *Handbook of Advanced Nondestructive Evaluation*; Springer: Cham, Switzerland, 2019. [CrossRef]
4. Escobar, C.A.; Moralez-Menendez, R. Machine learning techniques for quality control in high conformance manufacturing environment. *Adv. Mech. Eng.* **2018**, *10*. [CrossRef]
5. Silveira, J.; Ferreira, M.J.; Santos, C.; Martins, T. Computer Vision Techniques Applied to the Quality Control of Ceramic Plates. Available online: https://hdl.handle.net/1822/16574 (accessed on 25 April 2023).
6. Aydin, I.; Karakose, M.; Hamsin, G.G.; Sarimaden, A.; Akin, E. A new object detection and classification method for quality control based on segmentation and geometric features. In Proceedings of the International Artificial Intelligence and Data Processing Symposium (IDAP), Malatya, Turkey, 16–17 September 2017; IEEE: Malatya, Turkey, 2017; pp. 1–6. [CrossRef]
7. Gadelmawla, E.S. Computer vision algorithms for measurement and inspection of spur gears. *Measurement* **2011**, *44*, 1669–1678. [CrossRef]
8. Khalil, R.A.; Saeed, N.; Masood, M.; Fard, Y.M.; Alouini, M.-S.; Al-Naffouri, T.Y. Deep learning in the industrial Internet of Things: Potentials, challenges, and emerging applications. *Internet Things J.* **2021**, *8*, 11016–11040. [CrossRef]
9. Peraković, D.; Periša, M.; Zorić, P.; Cvitić, I. Development and implementation possibilities of 5G in industry 4.0. In *DSMIE 2020: Advances in Design, Simulation and Manufacturing III, Part of Lecture Notes in Mechanical Engineering*; Ivanov, V., Trojanowska, J., Pavlenko, I., Zajac, J., Peraković, D., Eds.; Springer: Cham, Switzerland, 2020; pp. 166–175. [CrossRef]
10. Walkington, W.G. *Die Casting Defects*; North American die-casting Association: Arlington Heights, IL, USA, 2013.
11. Gamage, P.; Xie, S.Q. A real time vision system for defect inspection in a cast extrusion manufacturing process. In Proceedings of the 14th International Conference on Mechatronics and Machine Vision in Practice, Xiamen, China, 4–6 December 2007; IEEE: Xiamen, China, 2007; pp. 240–245. [CrossRef]
12. Naveen, S.; Mohan, G.; Manjunatha Rajashekar, R.; Rajaprakash, B.M. Analysis of casting surface using machine vision and digital image processing technique. *Int. J. Innov. Res. Sci. Eng. Technol.* **2017**, *6*, 19912–19918.
13. Frayman, Y.; Zheng, H.; Nahavandi, S. Machine vision system for automatic inspection of surface defects in aluminum die-casting. *JACIII* **2006**, *10*, 281–286. [CrossRef]

14. Zhang, X.-W.; Ding, Y.-Q.; Lv, Y.-Y.; Shi, A.-Y.; Liang, R.-Y. A vision inspection system for the surface defects of strongly reflected metal based on multi-class SVM. *Expert Syst. Appl.* **2011**, *38*, 5930–5939. [CrossRef]
15. Pastor-Lopez, I.; Santos, I.; Santamaria-Ibirika, A.; Salazar, M.; de-la-Pena-Sordo, J.; Bringas, P.G. Machine-learning-based surface defect detection and categorisation in high-precision foundry. In Proceedings of the 7th IEEE Conference on Industrial Electronics and Applications (ICIEA), Singapore, 18–20 July 2012; IEEE: Singapore, 2012; pp. 1359–1364. [CrossRef]
16. Pastor-Lopez, I.; Santos, I.; de-la-Pena-Sordo, J.; Salazar, M.; Santamaria-Ibirika, A.; Bringas, P.G. Collective classification for the detection of surface defects in automotive castings. In Proceedings of the 8th Conference on Industrial Electronics and Applications (ICIEA), Melbourne, VIC, Australia, 19–21 June 2013; IEEE: Melbourne, VIC, Australia, 2013; pp. 941–946. [CrossRef]
17. Pastor-Lopez, I.; Santos, I.; de-la-Pena-Sordo, J.; Garcia-Ferreira, I.; Zabala, A.G.; Bringas, P.G. Enhanced image segmentation using quality threshold clustering for surface defect categorisation in high precision automotive castings. In Proceedings of the International Joint Conference SOCO'13-CISIS'13-ICEUTE'13, Salamanca, Spain, 11–13 September 2013; Springer: Cham, Switzerland, 2014; pp. 191–200. [CrossRef]
18. Lin, J.; Wen, K.; Liu, Y.; Zhang, X. Recognition and classification of surface defects of aluminum castings based on machine vision. In Proceedings of the 2021 International Conference on Machine Learning and Intelligent Systems Engineering (MLISE), Chongqing, China, 9–11 July 2021; IEEE: Chongqing, China, 2021; pp. 10–15. [CrossRef]
19. Swillo, S.J.; Perzyk, M. Surface casting defects inspection using vision system and neural network techniques. *Arch. Foundry Eng.* **2013**, *13*, 103–106. [CrossRef]
20. Wang, T.; Chen, Y.; Qiao, M.; Snoussi, H. A fast and robust convolutional neural network-based defect detection model in product quality control. *Int. J. Adv. Manuf. Technol.* **2018**, *94*, 3465–3471. [CrossRef]
21. Liu, Y.; Geng, J.; Su, Z.; Zhang, W.; Li, J. Real-time classification of steel strip surface defects based on deep CNNs. In Proceedings of the 2018 Chinese Intelligent Systems Conference, Wenzhou, China; Lecture Notes in Electrical Engineering; Jia, Y., Du, J., Zhang, W., Eds.; Springer: Singapore, 2018; pp. 257–266. [CrossRef]
22. Jiang, H.; Zhu, W. Defect detection method for die-casting aluminum parts based on RESNET. In Proceedings of the 3rd International Conference on Artificial Intelligence and Advanced Manufacture (AIAM2021), Manchester, UK, 23–25 October 2021; Association for Computing Machinery: New York, NY, USA, 2021; pp. 3051–3055. [CrossRef]
23. Lv, X.; Duan, F.; Jiang, J.; Fu, X.; Gan, L. Deep metallic surface defect detection: The new benchmark and detection network. *Sensors* **2020**, *20*, 1562. [CrossRef] [PubMed]
24. Liyun, X.; Boyu, L.; Hong, M.; Xingzhong, L. Improved Faster R-CNN algorithm for defect detection in powertrain assembly line. *Procedia CIRP* **2020**, *93*, 479–484. [CrossRef]
25. Sun, X.; Gu, J.; Huang, R.; Zou, R.; Palomares, B.G. Surface defects recognition of wheel hub based on improved Faster R-CNN. *Electronics* **2019**, *8*, 481. [CrossRef]
26. Li, Y.; Jiang, H.; Zhao, H.; Li, X.; Wang, Q. Surface defect detection of metal castings based on Centernet. In Proceedings of the International Conference on Optical and Photonic Engineering (icOPEN 2022), Online, 24–27 November 2022. [CrossRef]
27. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef]
28. Lin, T.Y.; Dollar, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; IEEE: Honolulu, HI, USA, 2017; pp. 936–944. [CrossRef]
29. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]
30. Khan, R.U.; Zhang, X.; Kumar, R.; Aboagye, E.O. Evaluating the performance of ResNet model based on image recognition. In Proceedings of the 2018 International Conference on Computing and Artificial Intelligence, Chengdu, China, 12–14 March 2018; Association for Computing Machinery: New York, NY, USA, 2018; pp. 86–90. [CrossRef]
31. Zhang, H.; Deng, Q. Deep learning based fossil-fuel power plant monitoring in high resolution remote sensing images: A comparative study. *Remote Sens.* **2019**, *11*, 1117. [CrossRef]
32. Detectron2. Available online: https://github.com/facebookresearch/detectron2 (accessed on 25 April 2023).
33. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, real-time object detection. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; IEEE: Las Vegas, NV, USA, 2016; pp. 779–788. [CrossRef]
34. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; IEEE: Honolulu, HI, USA, 2017; pp. 6517–6525. [CrossRef]
35. Redmon, J.; Farhadi, A. YOLOv3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
36. Bochkovskiy, A.; Wang, C.-Y.; Liao, H.-Y.M. YOLOv4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
37. Ultralytics. Available online: https://docs.ultralytics.com/yolov5/ (accessed on 25 April 2023).
38. Wang, C.-Y.; Liao, H.-Y.M.; Wu, Y.-H.; Chen, P.-Y.; Hsieh, J.-W.; Yeh, I.-H. CSPNet: A new backbone that can enhance learning capability of CNN. In Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Seattle, WA, USA, 14–19 June 2020; IEEE: Seattle, WA, USA, 2020; pp. 1571–1580. [CrossRef]

39. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; IEEE: Honolulu, HI, USA, 2017; pp. 2261–2269. [CrossRef]
40. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial Pyramid Pooling in deep convolutional networks for visual recognition. In *Computer Vision—ECCV 2014*; Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Springer: Cham, Switzerland, 2014; pp. 346–361. [CrossRef]
41. Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J. Path Aggregation Network for instance segmentation. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; IEEE: Salt Lake City, UT, USA, 2018; pp. 8759–8768. [CrossRef]
42. Ultralytics-Yolov5. Available online: https://github.com/ultralytics/yolov5 (accessed on 25 April 2023).
43. Zheng, Z.; Wang, P.; Liu, W.; Li, J.; Ye, R.; Ren, D. Distance-IoU loss: Faster and better learning for bounding box regression. *arXiv* **2019**, arXiv:1911.08287.
44. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An imperative style, high-performance deep learning library. In Proceedings of the 33rd International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; Curran Associates Inc.: Red Hook, NY, USA, 2019; pp. 8024–8035. [CrossRef]
45. Pump Impeller Dataset. Available online: https://www.kaggle.com/ravirajsinh45/real-life-industrial-dataset-of-casting-product (accessed on 25 April 2023).
46. Vioral. Available online: https://www.vioral.eu/ (accessed on 25 April 2023).
47. Apeer. Available online: https://www.apeer.com/app/dashboard (accessed on 25 April 2023).
48. Coco-Dataset. Available online: https://cocodataset.org/#home (accessed on 25 April 2023).
49. Ying, X. An overview of overfitting and its solutions. *J. Phys. Conf. Ser.* **2019**, *1168*, 022022. [CrossRef]
50. Okoro, U.R.; Idowu, S.A. On the cloud web services: A review. *Int. J. Comput. Technol.* **2013**, *9*, 1020–1027. [CrossRef]
51. Turab, N.M.; Taleb, A.A.; Masadeh, S.R. Cloud computing challenges and solutions. *Int. J. Comput. Netw. Commun.* **2013**, *5*, 209–216. [CrossRef]
52. Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge computing: Vision and challenges. *IEEE Internet Things J.* **2016**, *3*, 637–646. [CrossRef]
53. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-end object detection with transformers. In *Computer Vision—ECCV 2020*; Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M., Eds.; Springer: Cham, Switzerland, 2020; pp. 213–229. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.