*electronics*

MDPI

*Article*

# Defective Product Classification System for Smart Factory Based on Deep Learning

**Huy Toan Nguyen [1], Gwang-Huyn Yu [1], Nu-Ri Shin [1], Gyeong-Ju Kwon [2], Woo-Young Kwak [2] and Jin-Young Kim [1,*]**

[1] Department of ICT Convergence System Engineering, Chonnam National University, Gwangju 61186, Korea; nguyenhuytoantn@gmail.com (H.T.N.); sayney1004@gmail.com (G.-H.Y.); nulnul07@jnu.ac.kr (N.-R.S.)
[2] LINUXIT, Gwangju 61186, Korea; gyeongju@linuxit.co.kr (G.-J.K.); wooyoung@linuxit.co.kr (W.-Y.K.)
* Correspondence: beyondi@jnu.ac.kr

**Abstract:** Smart factories merge various technologies in a manufacturing environment in order to improve factory performance and product quality. In recent years, these smart factories have received a lot of attention from researchers. In this paper, we introduce a defective product classification system based on deep learning for application in smart factories. The key component of the proposed system is a programmable logic controller (PLC) artificial intelligence (AI) embedded board; we call this an AI Edge-PLC module. A pre-trained defective product classification model is uploaded to a cloud service from where the AI Edge-PLC can access and download it for use on a certain product, in this case, electrical wiring. Next, we setup the system to collect electrical wiring data in a real-world factory environment. Then, we applied preprocessing to the collected data in order to extract a region of interest (ROI) from the images. Due to limitations on the availability of appropriate labeled data, we used the transfer learning method to re-train a classification model for our purposes. The pre-trained models were then optimized for applications on AI Edge-PLC boards. After carrying out classification tasks, on our electrical wire dataset and on a previously published casting dataset, using various deep neural networks including VGGNet, ResNet, DenseNet, and GoogLeNet, we analyzed the results achieved by our system. The experimental results show that our system is able to classify defective products quickly with high accuracy in a real-world manufacturing environment.

**Keywords:** smart factory; defective product classification; deep learning; transfer learning



**Citation:** Nguyen, H.T.; Yu, G.-H.; Shin, N.-R.; Kwon, G.-J.; Kwak, W.-Y.; Kim, J.-Y. Defective Product Classification System for Smart Factory Based on Deep Learning. *Electronics* **2021**, *10*, 826. https://doi.org/10.3390/electronics10070826

Academic Editors: Chang Wook Ahn and Pankoo Kim

Received: 12 March 2021
Accepted: 29 March 2021
Published: 31 March 2021



## 1. Introduction

It has been suggested that we are entering the Fourth Industrial Revolution (Industry 4.0); one of the characteristics of Industry 4.0 is the boosted productivity and increased efficiency that will be seen in factories [1,2] and achieved by modern technologies such as the internet of thing (IoT), artificial intelligence (AI), cloud computing, robotics, sensors, and integrated systems. In Industry 4.0, large-scale IoT systems and machine-to-machine communications (M2M) will be integrated for enhanced automation, communication, and self-monitoring without the need for human intervention. The smart factory is a step beyond traditional automated manufacturing environments to factories where we will see fully automotive systems in which the machines are connected with sensors and other devices via wired or wireless networks and controlled by advanced computational intelligence [3]. In a study [4], the integration of all physical components and digital technologies into one system, to give us what is called cyber-physical systems (CPSs), is analyzed in detail. In addition, in modern smart factories, the number of sensors being used to collect data is increasing exponentially [5]. Among the various kinds of sensor systems found in factories, vision-based systems are the most popular and effective, when it comes to estimating and classifying product quality. A comprehensive review of automated vision-based defect detection approaches that look at numerous kinds of materials such as ceramics, textiles, and metals were introduced in [6].

In the manufacturing environment, a critical part of product quality control is defective product classification on the product line. Currently, numerous methods are deployed to tackle this task. Basically, these defective product classification systems must match challenging requirements such as the need to work in real-time with highly accuracy and robust performance in noisy environments like those seen in real-world factories. With the development of machine learning and sophisticated vision systems, feature-based defect classification algorithms are starting to be investigated and applied to classify defective products [7,8] using various classifiers such as Bayesian network classifiers [9], artificial neural networks (ANN) [10], and support vector machines (SVM) [11]. However, these feature-based classification systems can be sensitive to noise such as illumination changes or shadows in images when used in real-world factories. Moreover, in some factories, wide ranges of diverse products are produced over various product lines, making these factories unsuitable for the use of feature-based methods.

Over the past few years, deep learning-based techniques have achieved impressive results in various computer vision tasks such as image classification, object detection, object recognition. These successes have shown their great potential for application in the manufacturing environment [12]. A multitask convolution neural network (CNN) was proposed in [13] to integrate wire defect region detection and defective product classification. Other quality inspection tasks that use CNNs have been suggested to monitor various products such as printed circuit boards (PCBs) [14,15], metal surfaces [16], bottled wine [17], casting products [18,19], semiconductor fabrication [20], and light emitting diode (LED) cup apertures [21], mobile phone screen [22], cover glass of display panels [23], bearings [24], optical film [25], and leather defect [26]. In the aforementioned research on defect classification, authors only concentrated on developed the software model and implement on personal computer (PC) or server computer with graphics processing unit (GPU). In fact, using another computer for classifying product is not suitable with practical factory in term of upgrade existed system. Therefore, in this research, we have integrated the programmable logic controller (PLC) and AI embedded modules into the AI Edge-PLC module for classifying products and to control the product conveyor system. Our integrated board can be used and connected with other existing machines and sensors in practical factory as an additional part in automation system.

As it is well known, one of the biggest advantages to using deep learning models is the ability to update and automatically learning new features from new data or from different datasets. Therefore, if we continuously training the network when new data is available, the system will continue to improve by learning different characteristics from new data. Deep learning-based systems are powerful because of the various types of data available that includes ground truth information to train these systems. However, in practice, it is difficult to collect and label millions of images of defective and satisfactory examples of one product to train a deep learning model. Therefore, in this paper, we implement and re-train various deep learning networks using the transfer learning method [27] with certain pre-trained weights from training on the ImageNet dataset [28]. We chose four well-known and well-used networks, namely VGGNet [29], ResNet [30], DenseNet [31], and GoogLeNet [32,33] to re-train and evaluate their performance on the embedded board. First, we re-trained these models on our electrical wire dataset and evaluated their performance. However, because the original images were taken in a practical manufacturing environment, they contain a lot of unnecessary information and noise, such as inconsistent shadow or lighting levels, which can affect system performance. To mitigate these issues, we extracted a region of interest (ROI) from each image that includes only the electrical wire before carrying out training and testing. The best performing model will then be selected for adopting in factory. For comparison, we re-train models on our defective product dataset and upload to cloud service. This means that, for each product, the user will be able to load the appropriate pre-trained model or retrain the deep network with new collected images from the server. Moreover, the system also uploads images from AI module to the cloud in order to train new model further or to re-train old models to enhance system performance.

In practical applications, the classify step in predict phase is done on the AI Edge-PLC module. Our contributions in this research are as follows:

(1) We propose the new AI Edge-PLC module for smart factory applications.
(2) We introduced a ROI extraction method based on an image's saturation channel.
(3) We conducted various experiments to make comparisons between VGGNet, ResNet, DenseNet, and GoogLeNet and evaluated their classification abilities; for our database, GoogLeNet was found to have the best performance and most stable network.
(4) We tested our system in a practical factory environment. The experimental results show that our system achieved remarkable performance up to 99.34% of accuracy on electrical wire dataset.

The rest of the paper is constructed as follows. In Section 2, all materials and methods including those related to the AI Edge-PLC module, smart factories, data collection, ROI extraction, and deep neural networks are given. The experimental and analysis details are provided in Section 3. Finally, we present the conclusions and future works in Section 4.

## 2. Materials and Methods

In this section, we first describe the AI Edge-PLC hardware configuration. After that, we introduce the smart factory where the AI Edge-PLC module was applied. Then, the data collection system and ROI extraction methods are explained. Finally, we introduce the deep learning networks and model deployment using TensorRT.

### *2.1. AI Edge-PLC Board*

In most manufacture and industrial environments, the PLC, a digital and analog input/output device that is programmable to control mechanical tools, is the brain of the control system. Therefore, it is necessary to localize artificial intelligence technology on a PLC. We customized deep learning models to communicate with a PLC allowing them to receive and send commands. Our aim was to develop an edge PLC model that is compatible with built-in artificial intelligence (AI) models based on the vision solution shown in Figure 1. The AI calculation module shown here uses the Jetson Nano [34] module based on the ARM A57 Core released by NVIDIA. The module is based on NVIDIA's Maxwell architecture and has 138 NVIDIA CUDA cores capable of 0.5 TFLOPs with 4 GB of RAM, this makes it ideal for IoT applications. The GPU architecture is suitable for large number of calculation such as neural network and AI applications. Program code can be downloaded and executed on the PLC module by the user. The module supports functions to control and monitor external devices according to their needs. The processor of the main module is the SAMA5D2 processor released by Microchip. The SAMA5D2 processor is based on the ARM Cortex-A5 Core and has a 500 MHz Core Clock Speed, it is responsible for operating the PLC program.


**Figure 1.** AI Edge-PLC Board.

### 2.2. Smart Factory

The overall architecture of the smart factory is shown in Figure 2. To inspect the products moving down the product line, we used IR sensors to stop the conveyor bell when the product is in the correct position for the camera to capture its image before the conveyor belt is restarted. A suitable webcam was carefully selected to allow us to reliably capture images of all the products; these images have two purposes: (1), as input to the AI Edge-PLC; (2), to be stored on the cloud the dataset for further training of the system if necessary or to be available for review by specialists. The classification results are sent to the Edge-PLC via a socket protocol, as previously mentioned in Section 2.1.

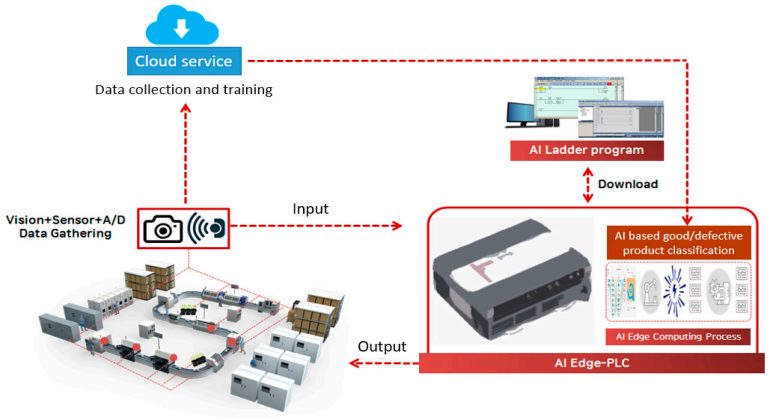


**Figure 2.** Architecture of the smart factory.

The framework of the defective product classification system is given in Figure 3. First, data on a particular product is collected directly from the factory. The collected data is processed to extract the region of interest (ROI) before each image is labelled by an operator or specialist as either defective or satisfactory. Then, the images are normalized and saved to the database where they can be used for training deep learning models. After using the collected labelled data to train the deep-learning models, the models themselves are evaluated to understand which model performs best for each product. This model is then considered as the pre-trained model that is then tested in a corresponding factory. After checking the product quality, the prediction analysis results are sent to the PLC that controls the actuators. The AI Edge-PLC board can also communicate with the cloud to upload any collected images or download a new pre-trained deep learning model.

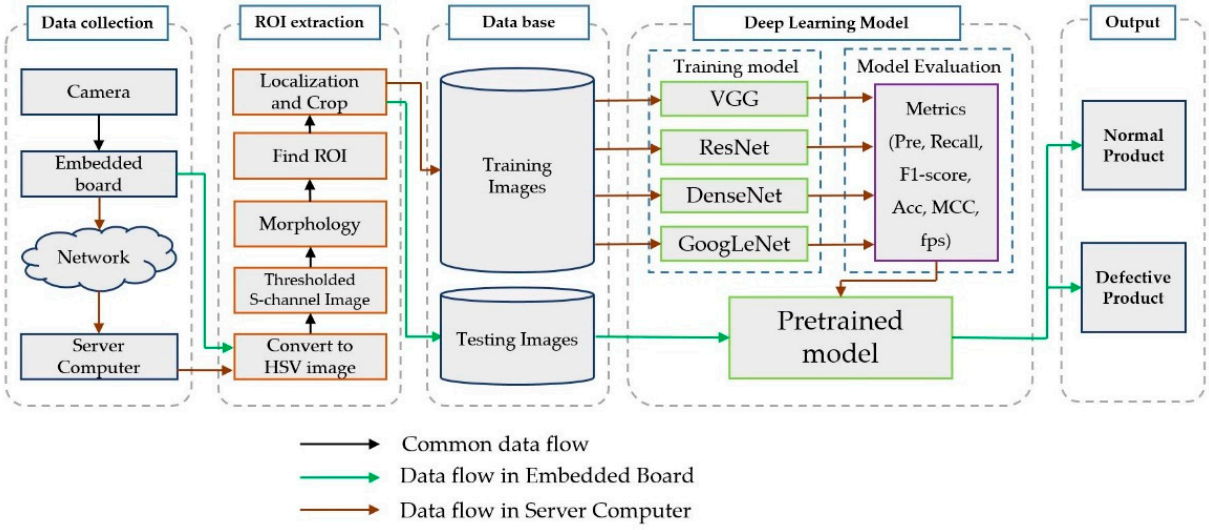


**Figure 3.** The framework of proposed method.

### 2.3. Data Collection

To collect the data for training, we set up a system consisting of a camera, AI embedded board, and server computer, as depicted in Figure 4. The captured images were sent to the server and saved in a database via a Wifi LAN network using a socket protocol. The Jetson nano board was used to control the webcam to capture product images from the conveyor belt. The resolution of each image was 1280 × 720 with high quality settings. The software for image acquisition and preprocessing was implemented using a Python-based OpenCV library.

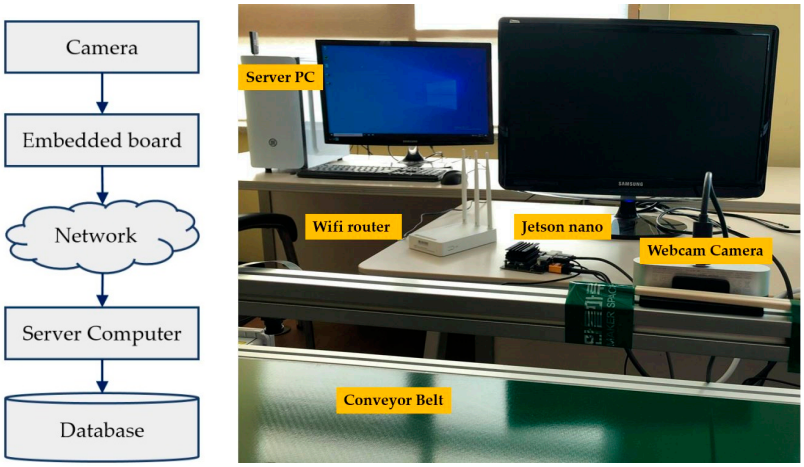


**Figure 4.** Data collection system: (**a**) Data collection diagram; (**b**) Data collection system setup.

Images of satisfactory electrical wiring samples are shown in Figure 5. As we can see in Figure 5, the images were captured under various lighting conditions to match those that might be encountered in real-world manufacturing situations. Under low levels of lighting, the shadows and reflections do not significantly affect the collected images. However, under higher levels of lighting, the reflections introduce unpredictable noise to the images which can result in classification errors. We collected and annotated 3879 images as samples of satisfactory results.

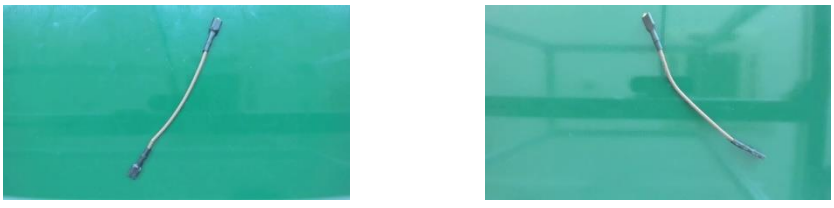


**Figure 5.** Satisfactory electrical wire connectors under different levels of lighting. (**a**) Low level of lighting; (**b**) High level of lighting.

Images of defective electrical wire samples are depicted in Figure 6. These defects can cause poor connections or even electrical equipment failure. The defective wire connectors are separated into four types, as shown in Figure 6a–d: (a) electrical wiring without connector cover, (b) electrical wiring without one connector, (c) electrical wiring with connector partially uncovered, and (d) broken electrical wire. Due to the differences in appearance and in wire connector position from image to image, it is difficult to successfully apply shallow-learning-based feature in this kind of classification problem.

Looking at the examples of the collected images in Figures 5 and 6, we can see that the captured images do not only show the electrical wires but also contain irrelevant

background details that actually take up most of the pixels in the image. Using these images as input to our system would result in unnecessary extra calculations in training and in classification. In practice, whatever the product, irrelevant lighting and background details could easily lead to classification errors. Therefore, in the next section, we propose a method to extract a ROI from the original image.

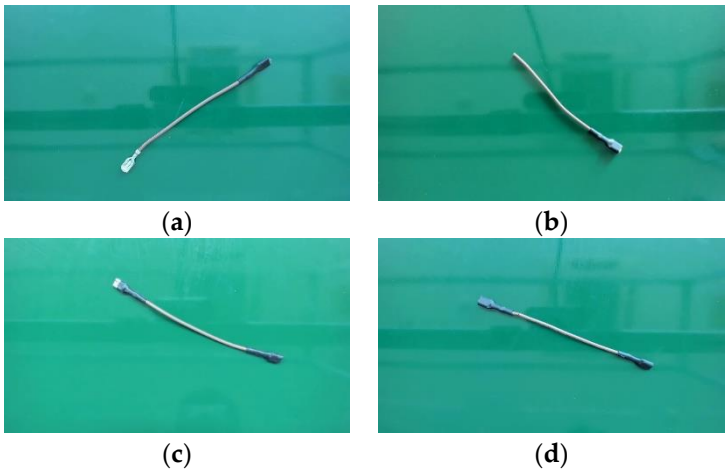

**Figure 6.** Examples of defective electrical wires. (**a**) Electrical wiring without connector cover; (**b**) Electrical wiring without one connector; (**c**) Electrical wiring with connector partially uncovered; (**d**) Broken electrical wire.

### *2.4. Region of Interest (ROI) Extraction on Images from Electrical Wire Dataset*

As mentioned above, not everything in the captured images is useful for estimating product quality. In previous work on a casting dataset with augmented data [35], the author has already explored various preprocessing methods to remove unrelated noise from images. These casting images captured using special arrangements to ensure the image was captured in stable lighting conditions. However, in a real-world manufacturing environment, the lighting condition obviously changes over time leading to classification errors if the vision system relies on stable lighting conditions. In this part, we investigate two methods to extract the ROI from captured images based on hue saturation value (HSV) color and saturation channel information.

#### 2.4.1. ROI Extraction Based on HSV Color

The ROI extraction method based on HSV color is presented in Figure 7. We assume that the color level of the electrical wire is different from the background.


**Figure 7.** ROI extraction based on HSV color.

First, the RGB image is converted to the HSV color space based on commonly used equations. Assume that $R, G, B \in [0, 1]$; $MAX := \max(R, G, B)$; $MIN := \min(R, G, B)$

$$H : \begin{cases} 0, & \text{if } R = G = B \\ 60^o \times \left(0 + \frac{G - B}{MAX - MIN}\right), & \text{if } MAX = R \\ 60^o \times \left(2 + \frac{B - R}{MAX - MIN}\right), & \text{if } MAX = G \\ 60^o \times \left(4 + \frac{R - G}{MAX - MIN}\right), & \text{if } MAX = B \end{cases} \quad (1)$$

$$S : \begin{cases} 0, \text{ if } R = G{=}B \\ \frac{MAX - MIN}{MAX}, \text{ else} \end{cases} \quad (2)$$

$$V := MAX \quad (3)$$

Subsequently, we selected a threshold based on the range of pixel values in the HSV color space according to the histogram of each channel. Next, we applied a morphological method to eliminate noise from the threshold image. The biggest ROI is assumed to be the region containing the electrical wire. This ROI is located and cropped from the original image. Representative results for each stage are presented in Figure 8. For low light level images, such as the ones shown in Figure 8a, the electrical wire is located and cropped correctly. However, with high light level images, such as the ones shown in Figure 8b, reflection from the conveyor belt means using a simple threshold with a HSV image does not result in a suitable ROI being extracted. Therefore, in the next section, we propose another method to extract the ROI that is based on only the saturation channel information.

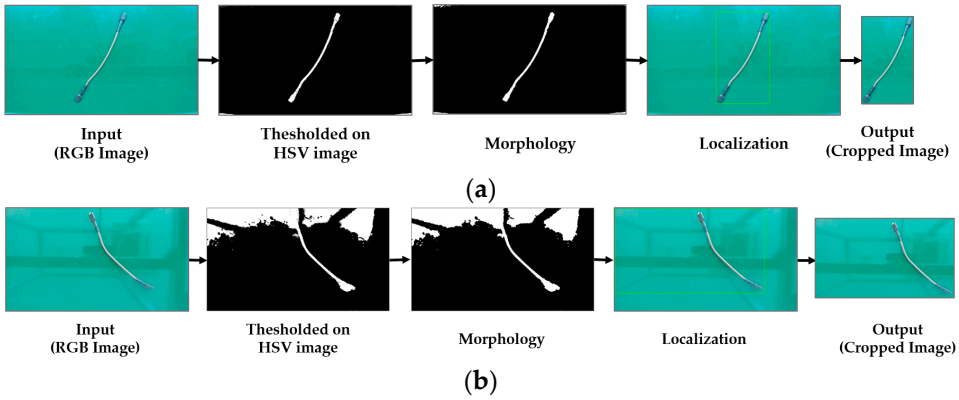


**Figure 8.** ROI extraction examples. (**a**) Low light level result; (**b**) High light level result.

#### 2.4.2. ROI Extraction Based on Saturation Channel Information

After converting the original to an HSV image, we analyzed the characteristics of each HSV channel. The H-channel (Hue) gives the basic attribute of color, this information is useful in image processing to segment an object based on color. The S-channel (Saturation) gives the purity of the color. The V-channel (Value) describes the brightness or the intensity of the color. The proposed method to extract the ROI based on the S channel information is shown in Figure 9.


**Figure 9.** ROI extraction based on saturation channel information.

First, the RGB image is converted to the HSV color base and each channel is separated. The S-channel is further processed for thresholding using Equations (4) and (5) as follows.

$$\text{Threshold} = \alpha \times \text{mean}(I_{S\_channel}) \quad (4)$$

$$I_{S_{threshold}}(x, y) = \begin{cases} 1, & \text{if } I_{S_{channel}}(x, y) > \text{Threshold} \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

where $I_{S\_channel}$ is the Saturation channel image, and $\alpha$ is a coefficient which is selected based on experiments of 0.8. The threshold value is equal to average value of all pixel in the Saturation channel image multiple by coefficient $\alpha$.

From the threshold image, the binary output is applied morphologically to remove noise and find the biggest region that could be considered as the ROI. Representative experimental results for each stage are presented in Figure 10. In Figure 10, we show the results of applying this method to the high light level image that the previous method failed to process correctly in Figure 8b. We can see that the proposed method is able to locate and crop the electrical wire accurately. A performance comparison between the two ROI extraction methods is provided in Section 3.

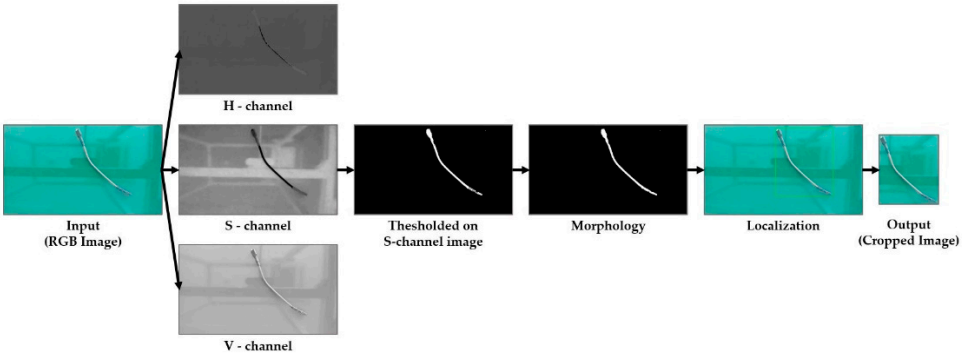


**Figure 10.** Representative ROI extraction result.

### *2.5. Defective Product Classification*

The aim of this stage is to recognize the class which the extracted ROI image belongs to. The defective products classification module goes through two main steps: (1) Train a deep neural network whose feature extractor portion was pre-trained on the ImageNet dataset, so that the neural network weights are already in place and ready to be used; (2) Optimize and customize the re-trained network, so that it is suitable for being deployed on an embedded board such as the Jetson Nano board.

#### 2.5.1. Training Deep Neural Model

Recently, deep learning-based methods have achieved remarkable performance in various computer vision tasks such as image classification, object detection, and semantic segmentation. Among the deep learning-based methods, the convolution neural network is the most representative architecture and so, it has been extensively studied. Based on their excellent performance, deep learning-based methods have been deployed in various applications, including on embedded systems. However, to train a deep neural network from scratch requires a dataset with millions of labeled entries, high computational power, and a long time. Transfer learning was introduced to tackle these issues, transfer learning is a technique to re-train a deep neural network (DNN) model on a new, small dataset by reusing its feature extractor portion which was trained using an existing large dataset such as the ImageNet dataset and re-training only the classification functionality to save training time. With transfer learning, the weights of a pre-trained model are fine-tuned to classify a new dataset. Moreover, this knowledge transfer can critically improve learning performance without requiring work-intensive data labeling efforts. The transfer learning approach has achieved state-of-the-art results on many datasets despite only requiring a small amount of training time. In this paper, four types of network architecture were investigated: VGGNet, ResNet, DenseNet, and GoogLeNet. The architectures of these four CNN networks are presented in Figure 11. The VGGNet consists of 16 convolution layers with small-sized kernels ($3 \times 3$), five max-pooling layers, three fully connected layers, and an output layer with a softmax nonlinear activation function. However, VGGNet contains a large number of parameters, up to 144 million, resulting in it being more expensive computationally because of the extensive amount of memory required. On the other hand, ResNet (Residual Network), in which the residual connections which significantly reduce optimization difficulties, has been enabled to train much deeper networks. DenseNet was introduced to ensure maximum information flow

between layers in a network by connecting all layers directly with each other in a dense block. DenseNet has an advantage in that it encourages feature reuse and alleviates the vanishing gradient problem. GoogLeNet uses an "inception module" which concatenates feature-maps produced by filters of different sizes. In this paper, we investigate using these networks after they are pre-trained on the famous ImageNet dataset which contains 1.2 million high-resolution labeled images divided into 1000 classes.


**Figure 11.** ROI extraction example result. Architecture of four convolutional neural network (**a**) VGGNet; (**b**) ResNet; (**c**) DenseNet; (**d**) GoogLeNet.

### 2.5.2. Deep Learning Model Deployment

An overview of our end-to-end deep learning workflow is shown in Figure 12. In the previous section, we discussed how the deep learning model is re-trained for our purposes using a GPU and is then saved before it is optimized for deploying on an embedded board, such as a Jetson Nano or Jetson AX. As mention in Section 2.1, we selected the Jetson Nano on which to deploy the pre-trained models and integrated this with a PLC board to make our AI Edge-PLC board. We are able to import and deploy every deep learning framework using TensorRT. According to a study [36], TensorRT-based applications perform up to 40× faster than CPU-only platforms during inference. With TensorRT, researchers only need to focus on creating novel AI-powered applications rather than worrying about performance tuning for inference deployment. In order to run pre-trained models with TensorRT on the Jetson Nano for real-time applications, we have to convert the pre-trained models to the Open Neural Network Exchange (ONNX) format [37]. These processes are completed on the server. A TensorRT model in the ONNX format can then be downloaded from the cloud and loaded onto the Jetson Nano ready for the prediction phase. The experimental results show that the deployed model is able to run in real-time. The output results are

sent directly to the PLC via socket protocol to control the actuators based on how a sample is classified.


**Figure 12.** Deep learning system's end-to-end flow.

## 3. Experiment and Analysis

This section presents details of the experimental settings for preprocessing, the re-training process, and the test results. Additionally, we also provide analysis based on the results of each section of testing.

### *3.1. Experimental Environment and Parameter Setting*

To train the CNN-networks, a high-performance server with an Intel Corporation Xeon E5/ Core i7 DMI2 CPU, 16 GB of RAM, and a Quadro K600 GPU running the Ubuntu 16.04.6 LTS Operating system was used. For training, we used weighted cross-entropy as the loss function and Stochastic Gradient Descent (SGD) as the optimization method. The training parameters used for all networks are shown in Table 1. In the prediction phase, as mentioned in Section 2.1, the Jetson Nano, a small AI computer, was selected to run the neural networks while we evaluated system performance.

**Table 1.** Parameters used for the CNNs in this research.

| Parameter | Setting |
|---|---|
| Training epoch | 50 |
| Batch size | 8 |
| Optimizer | SGD |
| Loss function | Cross-Entropy |
| Learning rate | 0.01 |

### *3.2. Dataset Description*

To evaluate and compare the performance of the proposed method, a publicly available dataset related to product classification, the casting dataset, was selected [30]. The casting product data includes two datasets; one includes 1300 images that are 512 × 512 pixels in size, the other one contains 7348 augmented, 300 × 300 pixel gray-scale images. We chose the second dataset that includes augmented images for training and testing the system. These images were captured in stable lighting using special arrangements. Examples of defective and good products in the casting product dataset are given in Figure 13.

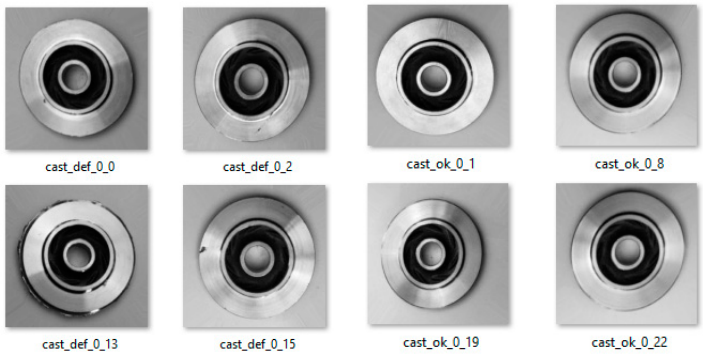


**Figure 13.** Casting product dataset. (**a**) Defective products; (**b**) Satisfactory products.

In addition, the original electrical wire and cropped electrical wire datasets was use to examine the effect of applying the ROI extraction step. The final number of cropped images is smaller than the number of original electrical wire images due to not including images when the ROI extraction process failed. For all dataset, we randomly choose 80% for training (70% for training and 10% for validation) and remaining 20% is for testing. The details of each dataset and the distributions used for training, validation and testing are provided in Table 2.

**Table 2.** Dataset distributions for training, validation, and testing.

| Dataset | Class | Total | Training | | Testing |
|---|---|---|---|---|---|
| | | | Train | Validation | |
| Casting dataset | Good | 3137 | 2196 | 314 | 627 |
| | Defective | 4211 | 2948 | 421 | 842 |
| Original electrical wire | Good | 3879 | 2715 | 388 | 776 |
| | Defective | 4000 | 2800 | 400 | 800 |
| Cropped electrical wire | Good | 3699 | 2589 | 370 | 740 |
| | Defective | 3903 | 2732 | 390 | 781 |

### *3.3. Evaluation Metrics*

The evaluation metrics in this research for product classification are precision, recall, F1_score, accuracy, and Matthews correlation coefficient (MCC) [38]. Precision is calculated using the number of true positives divided by the number positive samples predicted. Recall (sensitivity) is computed by taking the number of true positives divided by all positive samples. F1_scoreis the weighted average of precision and recall. Accuracy is the ratio between the number of correct predictions to the total number of predictions. The problem of defect classification in product is a binary one. Therefore, a more reliable statistical rate evaluation metric that considers both positive elements and negative elements was adopted for this study. That is the Matthews correlation coefficient (MCC) [39]. These five evaluation metrics were computed using the true positive (TP), true negative (TN), false positive (FP), and false-negative (FN) results from a confusion matrix between the prediction and the ground truth as follows.

$$\text{Precision} = \frac{\text{TP}}{(\text{TP} + \text{FP})} \tag{6}$$

$$\text{Recall} = \frac{\text{TP}}{(\text{TP} + \text{FN})} \tag{7}$$

$$\text{F1\_score} = \frac{(2 \times \text{Precision} \times \text{Recall})}{(\text{Precision} + \text{Recall})} \tag{8}$$

$$\text{Accuracy} = \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{FP} + \text{TN} + \text{FN})} \tag{9}$$

$$\text{MCC} = \frac{\text{TP} \times \text{TN} \; - \; \text{FP} \times \text{FN}}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}} \tag{10}$$

### *3.4. Experimental Results and Analysis*

#### 3.4.1. Performance of ROI Extraction

We first evaluated the performance of the two ROI extraction methods based on HSV data and S-channel data. The cropped images were carefully checked by our data team members. The performance was evaluated based on the original electrical wire dataset that contains 7879 images. ROI extraction based on the S-channel data achieved a success rate of up to 96.48% correctly cropped images. Conversely, the ROI extraction method based on HSV data only achieved a success rate of 84.44%, in Table 3, as this method struggled with

the lighting conditions changes in the real-world manufacturing environment when the images were taken. The 7602 correctly cropped images from ROI extraction based on the S-channel ROI extraction method were considered as the cropped electrical wire dataset and were used in the later experiments.

**Table 3.** Comparison of ROI Extraction methods.

| Parameter | ROI Extraction Based on HSV | ROI Extraction Based on S-Channel |
|---|---|---|
| Total images | 7879 | 7879 |
| Correctly cropped | 6653 | 7602 |
| Incorrectly cropped | 1226 | 277 |
| Overall accuracy | 84.44% | 96.48% |

#### 3.4.2. Performance Analysis with Electrical Wire Dataset

In order to highlight the importance and effectiveness of the ROI extraction step in the training and testing phases, in this section, we evaluate and analyze the performance of the transfer learning method with both the original electrical wire dataset and the cropped electrical wire dataset (Figure 14). We can see from Figure 14 that the training process with all pre-trained models using the cropped electrical wire dataset has faster convergence than when using the original electrical wire dataset. After training for 17 epochs, all networks trained using cropped images achieved over 95% of accuracy. However, using the original electrical wire dataset, only GoogLeNet was able to achieve 95% training accuracy, the other networks tested achieved around 85%. After 30 epochs, all networks trained using cropped images converge to the best possible performance; this is not the case when using the original images. The average training time in seconds of each network for one epoch are provided in Table 4. We can see that VGGNet needed the longest time (~255 s) to complete one epoch of training; this is due to the number of parameters in the pre-trained model. ResNet took the shortest time (~57 s) to re-train for one epoch. In practical applications in a smart factory, this difference could have a big effect on manufacturing performance when using this kind of system.

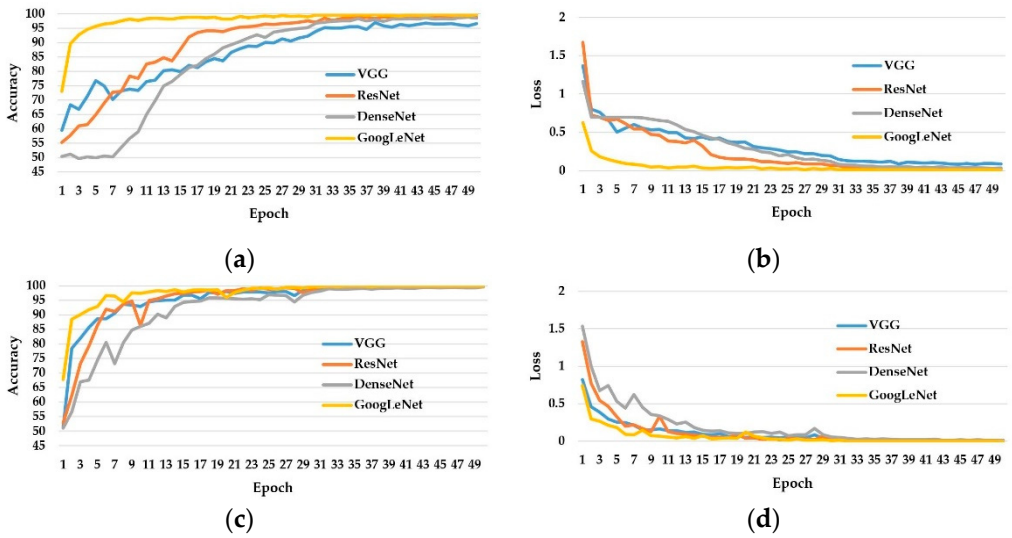


**Figure 14.** Training process. (**a**,**b**) Model accuracy and loss when trained on original electrical wire image dataset; (**c**,**d**) Model accuracy and loss when trained on cropped electrical wire image dataset.

**Table 4.** The average training time for one epoch.

| Dataset | VGGNet | ResNet | DenseNet | GoogLeNet |
|---|---|---|---|---|
| Original electrical wire | 265.2 | 59.2 | 152.6 | 86.0 |
| Cropped electrical wire | 255.9 | 57.4 | 151.4 | 82.8 |

In the testing phase, we compared the performance between all the networks and between the two datasets. The average performance details are presented in Table 5. From here, we see that DenseNet achieves the best performance using either electrical wire dataset. While this average performance difference between DenseNet and GoogLeNet is small, there is a large network speed difference between DenseNet (~30 fps) and GoogLeNet (~52 fps). ResNet, on the other hand, has a worse performance than both DenseNet and GoogLeNet, but its prediction speed (~54 fps) is the best. VGGNet is the slowest (~11 fps). For this kind of product in a real-world manufacturing environment, we would choose GoogLeNet as the best combination of speed and performance.

**Table 5.** The average performance while using the two electrical wire datasets.

| Dataset | Metrics | VGGNet | ResNet | DenseNet | GoogLeNet |
|---|---|---|---|---|---|
| Original electrical wire | Precision (%) | 98.20 | 97.69 | 98.72 | 98.59 |
| | Recall (%) | 98.58 | 97.94 | 99.10 | 98.84 |
| | F1-score (%) | 98.39 | 97.81 | 98.91 | 98.71 |
| | Accuracy (%) | 98.41 | 97.84 | 98.92 | 98.73 |
| | MCC (%) | 96.83 | 95.68 | 97.84 | 97.46 |
| | Speed (fps) | 11.68 | 54.73 | 29.40 | 52.19 |
| Cropped electrical wire (Ours) | Precision (%) | 98.65 | 98.77 | 99.86 | 99.59 |
| | Recall (%) | 98.78 | 97.97 | 99.46 | 99.05 |
| | F1-score (%) | 98.72 | 98.37 | 99.66 | 99.32 |
| | Accuracy (%) | 98.75 | 98.42 | 99.67 | 99.34 |
| | MCC (%) | 97.50 | 96.84 | 99.34 | 98.69 |
| | Speed (fps) | 11.65 | 53.23 | 30.80 | 53.42 |

#### 3.4.3. Performance Analysis on Casting Dataset

The number of parameters used and average training time results for 1 epoch are shown in Table 6. We can see that VGGNet takes the longest to train one epoch with a time of ~287 s. In contrast, ResNet needs only ~65 s to train one epoch. The training accuracy and training loss curve are depicted in Figure 15. This shows that all four networks achieved good, robust performance after training for at least 33 epochs. We deployed the pre-trained models on the Jetson Nano and found that DenseNet had the best on classification performance, as shown in Figure 16. GoogLeNet achieved the second best performance, while in recall, GoogLeNet achieved the best performance. The numerical results are presented in Table 7, the gap in MCC between GoogLeNet and DenseNet is only 0.3%. However, the training time for re-training one epoch with DenseNet is 1.5 higher than with GoogLeNet. In addition, the average network speed of GoogLeNet is 52.35 fps, while for the same metric, DenseNet achieves only 29.8 fps. For this kind of casting classification task, we would also choose GoogLeNet to be applied on the practical factory.

**Table 6.** The average training time for one epoch.

| VGGNet | ResNet | DenseNet | GoogLeNet |
|---|---|---|---|
| 265.2 | 59.2 | 152.6 | 86.0 |
| 255.9 | 57.4 | 151.4 | 82.8 |

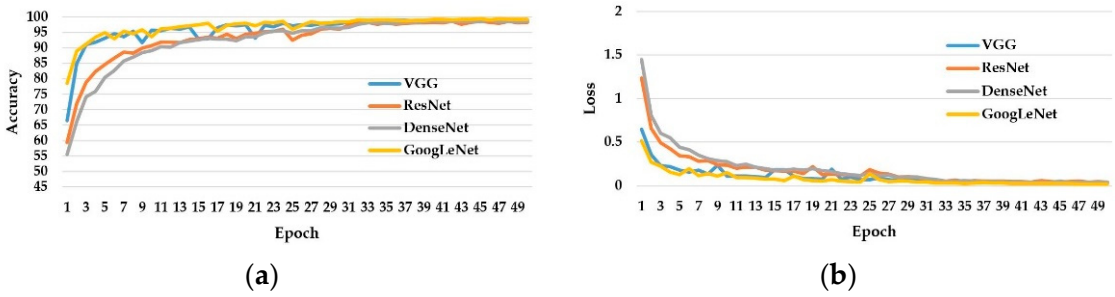


**Figure 15.** Training curve for casting dataset. (**a**) Model Accuracy, (**b**) Model Loss.

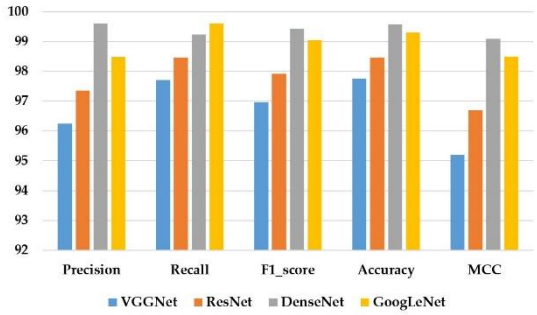


**Figure 16.** Casting product classification performance in testing phase.

**Table 7.** Casting product classification performance.

| Metrics | VGGNet | ResNet | DenseNet | GoogLeNet |
|---|---|---|---|---|
| Precision (%) | 96.24 | 97.36 | 99.62 | 98.49 |
| Recall (%) | 97.71 | 98.47 | 99.24 | 99.62 |
| F1-score (%) | 96.97 | 97.91 | 99.43 | 99.05 |
| Accuracy (%) | 97.76 | 98.46 | 99.58 | 99.30 |
| MCC (%) | 95.20 | 96.70 | 99.10 | 98.50 |
| Speed (fps) | 11.63 | 54.91 | 29.8 | 52.35 |

## 4. Discussion

In previous studies on defective product classifications [14–26], researchers have proposed methods based on deep learning which developed and run on personal computer or server GPU. In practical aspects, installing an additional new computer or a server to running on automation system such as conveyor belts is infeasible. The main reason is that the classify system have to communicate with other parts, such as other sensors or other controllers. In this research, an AI Edge-PLC board is proposed to solve the above-mentioned problems. Our board is not only able to classify good or defective product, but also connect with other system such as other PLC via communication ports. Our AI Edge-PLC board is small, low-power consumption, and installed easily with the existing system.

To confirm the board performance on deep learning-based method, we conducted the experiment with electrical wire dataset. We proposed to apply ROI extraction method to achieve better performance by using the same network architecture (GoogLeNet). The experimental results, in Figure 17, show that the proposed method with ROI extraction method outperformed the baseline GoogLeNet without ROI extraction. There are two main reasons for this improvement. First, by applying ROI extraction method before giving images to the classify system, we were able to remove unnecessary part of image where false information existed. Second, after apply ROI extraction method, we ensured that the system focuses on ROI part for learning more detailed information and the high-level descriptive features in ROI to boost the overall system performance.

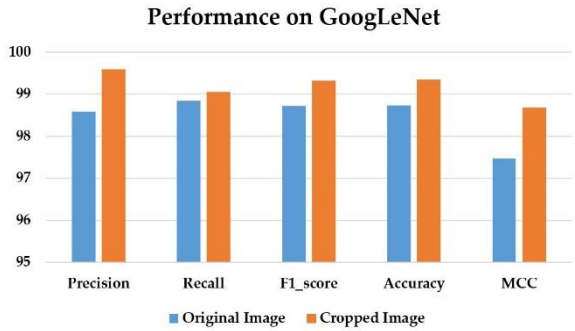


**Figure 17.** Performance comparison of two datasets on GoogLeNet.

## 5. Conclusions and Future Work

This paper introduces a defective product classification system for smart factories that is based on deep learning. The proposed system has three main parts, an AI embedded module (Jetson Nano hardware), a PLC module to control any mechanical devices, and a cloud service. We integrated the AI embedded board and PLC board into an AI Edge Module. Then, we implemented a transfer learning method to re-train deep learning networks on a public casting product dataset and on two of our own electrical wire datasets, one with and one without ROI extraction based on S-channel data. Four well-known deep learning models were selected to investigate a system performance. Each pre-trained model was deployed on our AI embedded module using the Jetson Nano board. The experimental results show that the system is able to classify input images with high accuracy in real-time using this AI Edge-PLC embedded board. These results are very impressive and promising for a wide range of potential applications. In the future, the proposed method will be further tested with other products in realistic factory environments to confirm the system performance in new areas.

**Author Contributions:** Conceptualization, H.T.N., G.-J.K. and J.-Y.K.; methodology, H.T.N. and J.-Y.K.; software, H.T.N. and G.-H.Y.; validation, G.-J.K., G.-H.Y. and J.-Y.K.; formal analysis, W.-Y.K. and N.-R.S.; investigation, H.T.N. and W.-Y.K.; resources, W.-Y.K. and G.-J.K.; data curation, G.-H.Y. and N.-R.S.; writing—original draft preparation, H.T.N. and G.-H.Y.; writing—review and editing, G.-H.Y. and J.-Y.K.; visualization, H.T.N. and G.-H.Y. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by NIPA (National IT Industry Promotion Agency), Korea, under the GITCT (Gwangju Information & Culture Industry Promotion Agency) funded by the Ministry of Science and ICT (MSIT, Korea). [Project Name: AI based Implementation of PLC Module and Controller on Production Site, Project Number: ITAS042620011009000100030020 0].

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

## References

1. Osterrieder, P.; Budde, L.; Friedli, T. The Smart Factory as a key construct of Industry 4.0: A systematic literature review. *Int. J. Prod. Econ.* **2019**, *221*, 107476. [CrossRef]
2. Büchi, G.; Cugno, M.; Castagnoli, R. Smart factory performance and Industry 4.0. *Technol. Forecast. Soc. Chang.* **2020**, *150*, 119790. [CrossRef]
3. Alcácerac, V.; Cruz-Machadoab, V. Scanning the Industry 4.0: A Literature Review on Technologies for Manufacturing Systems. *Eng. Sci. Technol. Int. J.* **2019**, *22*, 899–919.
4. Sinha, D.; Roy, R. Reviewing Cyber-Physical System as a Part of Smart Factory in Industry 4.0. *IEEE Eng. Manag. Rev.* **2020**, *48*, 103–117. [CrossRef]
5. Kalsoom, T.; Ramzan, N.; Ahmed, S.; Ur-Rehman, M. Advances in Sensor Technologies in the Era of Smart Factory and Industry 4.0. *Sensors* **2020**, *20*, 6783. [CrossRef]
6. Czimmermann, T.; Ciuti, G.; Milazzo, M.; Chiurazzi, M.; Roccella, S.; Oddo, C.M.; Dario, P. Visual-Based Defect Detection and Classification Approaches for Industrial Applications—A SURVEY. *Sensors* **2020**, *20*, 1459. [CrossRef]
7. Kuo, C.F.J.; Lo, W.C.; Huang, Y.R.; Tsai, H.Y.; Lee, C.L.; Wu, H.C. Automated defect inspec-tion system for CMOS image sensor with micro multi-layer non-spherical lens module. *J. Manuf. Syst.* **2017**, *45*, 248–259.

8. Aminzadeh, M.; Kurfess, T. Automatic thresholding for defect detection by back-ground histogram mode extents. *J. Manuf. Syst.* **2015**, *37*, 83–92. [CrossRef]
9. Pernkopf, F. Detection of surface defects on raw steel blocks using Bayesian network classifiers. *Pattern Anal. Appl.* **2004**, *7*, 333–342. [CrossRef]
10. Kuo, C.F.J.; Lee, C.J.; Tsai, C.C. Using a neural network to identify fabric defects in dynamic cloth inspection. *Text. Res. J.* **2003**, *73*, 238–244. [CrossRef]
11. Samy, M.P.; Foong, S.; Soh, G.S.; Yeo, K.S. Automatic Optical & Laser-Based Defect Detection and Classification in Brick Masonry Walls. In Proceedings of the 2016 IEEE Region 10 Conference (TENCON), Singapore, 22–25 November 2016; pp. 3521–3524.
12. Wang, J.; Ma, Y.; Zhang, L.; Gao, R.X.; Wu, D. Deep learning for smart manufacturing: Methods and applications. *J. Manuf. Syst.* **2018**, *48*, 144–156. [CrossRef]
13. Tao, X.; Wang, Z.H.; Zhang, Z.T.; Zhang, D.P.; Xu, D.; Gong, X.Y.; Zhang, L. Wire defect recognition of spring-wire socket using multitask convolutional neural networks. *IEEE Trans Compon. Packag. Manuf. Technol.* **2018**, *8*, 689–698. [CrossRef]
14. Adibhatla, V.A.; Chih, H.-C.; Hsu, C.-C.; Cheng, J.; Abbod, M.F.; Shieh, J.-S. Defect Detection in Printed Circuit Boards Using You-Only-Look-Once Convolutional Neural Networks. *Electronics* **2020**, *9*, 1547. [CrossRef]
15. Zhang, E.; Li, B.; Li, P.; Chen, Y. A Deep Learning Based Printing Defect Classification Method with Imbalanced Samples. *Symmetry* **2019**, *11*, 1440. [CrossRef]
16. Yun, J.P.; Shin, W.C.; Koo, G.; Kim, M.S.; Lee, C.; Lee, S.J. Automated defect inspection system for metal surfaces based on deep learning and data augmentation. *J. Manuf. Syst.* **2020**, *55*, 317–324. [CrossRef]
17. Wang, J.; Fu, P.; Gao, R. Machine vision intelligence for product defect inspection based on deep learning and Hough transform. *J. Manuf. Syst.* **2019**, *51*, 52–60. [CrossRef]
18. Nguyen, T.P.; Choi, S.; Park, S.-J.; Park, S.H.; Yoon, J. Inspecting Method for Defective Casting Products with Convolutional Neural Network (CNN). *Int. J. Precis. Eng. Manuf. Green Technol.* **2020**, *8*, 583–594. [CrossRef]
19. Nguyen, H.T.; Shin, N.-R.; Yu, G.-H.; Kwon, G.-J.; Kwak, W.-Y.; Kim, J.-Y. Deep Learning-Based Defective Product Classification System for Smart Factory. In Proceedings of the International Conference on Smart Media and Applications (SMA2020), Jeju, Korea, 17–19 September 2020.
20. Tello, G.; Al-Jarrah, O.Y.; Yoo, P.D.; Al-Hammadi, Y.; Muhaidat, S.; Lee, U. Deep-structured machine learning model for the recognition of mixed-defect patterns in semiconductor fabrication processes. *IEEE Trans. Semicond. Manuf.* **2018**, *31*, 315–322. [CrossRef]
21. Yang, Y.X.; Lou, Y.T.; Gao, M.Y.; Ma, G.J. An automatic aperture detection system for LED cup based on machine vision. *Multimed. Tools Appl.* **2018**, *77*, 23227–23244. [CrossRef]
22. Li, C.; Zhang, X.; Huang, Y.; Tang, C.; Fatikow, S. A novel algorithm for defect extraction and classification of mobile phone screen based on machine vision. *Comput. Ind. Eng.* **2020**, *146*, 106530. [CrossRef]
23. Park, J.; Riaz, H.; Kim, H.; Kim, J. Advanced cover glass defect detection and classification based on multi-DNN model. *Manuf. Lett.* **2020**, *23*, 53–61. [CrossRef]
24. Lu, M.; Mou, Y. Bearing defect classification algorithm based on Autoencoder neural network. *Adv. Civil Eng.* **2020**, *2020*, 6680315. [CrossRef]
25. Le, N.T.; Wang, J.-W.; Shih, M.-H.; Wang, C.-C. Novel framework for optical film defect detection and classification. *IEEE Access* **2020**, *8*, 60964–60978. [CrossRef]
26. Liong, S.-T.; Zheng, D.; Huang, Y.-C.; Gan, Y.S. Leather defect classification and segmentation using deep learning architecture. *Int. J. Comput. Integr. Manuf.* **2020**, *33*, 1105–1117. [CrossRef]
27. Liu, L.L.; Yan, R.J.; Maruvanchery, V.; Kayacan, E.; Chen, I.M.; Tiong, L.K. Transfer learning on convolutional activation feature as applied to a building quality assessment robot. *Int. J. Adv. Robot. Syst.* **2017**, *14*. [CrossRef]
28. Deng, J.; Dong, W.; Socher, R.; Li, L.; Li, K.; Li, F.-F. ImageNet: A Large-Scale Hierarchical Image Database. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Miami, FL, USA, 20–25 June 2009; pp. 248–255.
29. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2014**, arXiv:1409.1556.
30. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
31. Huang, G.; Liu, Z.; Maaten, L.; Weinberger, K. Densely Connected Convolutional Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–36 July 2017; pp. 2261–2269.
32. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going Deeper with Convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 1–9.
33. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception Architecture for Computer Vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 2818–2826.
34. Available online: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/ (accessed on 19 June 2020).
35. Available online: https://www.kaggle.com/ravirajsinh45/real-life-industrial-dataset-of-casting-product (accessed on 26 July 2020).

36. Available online: https://github.com/dusty-nv/jetson-inference (accessed on 5 May 2020).
37. Available online: https://onnx.ai (accessed on 6 August 2020).
38. Matthews, B.W. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochim. Biophys. Acta (BBA) Protein Struct.* **1975**, *405*, 442–451. [CrossRef]
39. Chicco, D.; Jurman, G. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genom.* **2020**, *21*, 6. [CrossRef]