Advances in Science and Technology Research Journal

Advances in Science and Technology Research Journal, 18(8), 403–415
https://doi.org/10.12913/22998624/194891
ISSN 2299-8624, License CC-BY 4.0

Received: 2024.08.22
Accepted: 2024.10.19
Published: 2024.11.01

Application of Neural Networks for Defect Detection in Rotationally Symmetric Components

Andrzej Chmielowiec[1*], Paweł Żurawski[2], Sylwia Sikorska-Czupryna[1], Leszek Klich[1], Patryk Organiściak[3]

[1] Faculty of Mechanics and Technology, Rzeszow University of Technology, ul. Kwiatkowskiego 4, 37-450 Stalowa Wola, Poland

[2] Doctoral School of the Rzeszów University of Technology, al. Powstańców Warszawy 12, 35-959 Rzeszów, Poland

[3] Faculty of Electrical and Computer Engineering, Rzeszow University of Technology, ul. Wincentego Pola 2, 35-959 Rzeszów, Poland

* Corresponding author's e-mail: achmie@prz.edu.pl

ABSTRACT

Industrial quality control systems in mass production facilities must exhibit a very high level of defect detection efficiency. The continuous increase in quality control and process automation requirements is leading companies to increasingly experiment with artificial intelligence methods to boost efficiency. One potential application area for AI is visual inspection, which is an essential element of almost every quality control process. In this article, we propose the use of neural networks for the visual inspection of rotationally symmetric components. The presented method leverages the existence of symmetry to represent images in a polar coordinate system and to implement the learning process on data modified in this way. An undeniable advantage of the proposed algorithm is also the transition from a two-dimensional to a one-dimensional representation, which significantly reduces the demand for memory resources and the required computational power. This is particularly important in mass production processes, where the time for data analysis is relatively short. The high repeatability of images due to the mass production nature makes this model exceptionally effective, allowing not only to confirm the presence of defects but even to locate them. The obtained results are compared with the results achieved using a convolutional neural network operating on two-dimensional images.

Keywords: defect detection; quality control; neural networks; computer vision; mass production.

INTRODUCTION

Short production cycle times and the necessity to deliver products that meet technical specifications at the lowest possible cost present challenges for enterprises. Delivering parts that are crucial for the safety of machinery and equipment requires special attention. Personnel working in visual inspection are flexible, easy to train, and creative, but at the same time, they constitute a cost for the enterprise, become fatigued during long hours of work, and operate slower than vision systems [1, 2]. On the other hand, quality control systems, while essential for ensuring high-quality products and services, also come with significant implementation and maintenance costs, including hardware, software, and other operational expenses [3]. Quality involves more than just the final inspection of a product; it encompasses the entire production process. Even a small oversight or lapse in attention at any stage can lead to defects, disrupting production and making it more difficult to uphold consistent high standards efficiently [4, 5]. The quality control process should be economically justified, ensuring that the costs involved do not outweigh the benefits of maintaining high product standards [6, 7].

In some cases, delivering a single defective product to a customer can result in multimillion penalties. Detecting such defects requires the withdrawal of the entire series of products from the market at the manufacturer's expense and the payment of compensation. An example is the detection of malfunctioning airbags by Takata, which could have led to servicing as many as 100 million cars. Consumers, upon discovering a defective product, can assert their rights based on national regulations and directives, such as the EU Directive 85/374/EEC, often resulting in multimillion-dollar compensations [8].

Vision systems in large-scale production applications offer several advantages over visual inspection, including faster product inspection, increased defect detection effectiveness, real-time data analysis capability, and the prevention of human errors [9–12].

In modern industrial enterprises, vision systems are integrated with tracking and control systems for production parameters. This integration is realized through systems such as TQM and MES, which increasingly use machine learning [13] alongside standard statistical methods [14]. Additionally, the use of vision systems for industrial quality control measurements is continuously advancing. Therefore, this process is increasingly supported by artificial intelligence algorithms. Modern production lines are equipped with a significant amount of automation and electronics that provide information on operating parameters and the technical conditions of machines and equipment. These algorithms can correlate this information with production efficiency and quality thanks to advanced vision systems. Information exchange between the vision system and other systems such as ERP, MES, TQM, and TPM is also possible [15]. Although computer vision-based vision systems are still the most popular in industrial environments, AI-based algorithms, including deep learning and convolutional neural networks, are gaining prominence. The accuracy of such algorithms is close to 100%, which perfectly aligns with the expectations of large-scale production [12]. Moreover, the integration of machine learning and AI is further advanced by improving traditional methods, such as feature-based image stitching, as demonstrated in the study by Schlagenhauf et al., where an innovative stitching algorithm for rotationally symmetric components was developed [16].

Therefore, quality control plays a key role in production processes, ensuring the detection of defects and non-conformities before products are put into use [17, 18]. Traditional visual inspection methods, involving human factors, are labour-intensive, prone to errors, and suffer from the limitations of human perception. In recent years, advanced machine learning techniques, especially neural networks, have revolutionized the quality control process, offering automated, accurate, and scalable solutions for defect detection [19].

Neural networks, inspired by the architecture of biological neurons, exhibit unparalleled ability in pattern recognition and data classification [20, 21], making them an ideal tool for defect identification in quality control processes. Through deep learning, neural networks can learn distinctive features of defects from large training datasets and detect them in new examples with high precision [22]. Architectures such as Convolutional Neural Networks (CNN) [23] and ResNet [24] have revolutionized image recognition and object detection, enabling effective model training on large datasets.

A particular challenge in quality control is the inspection of symmetrical components, where detecting minor deviations from the correct form is crucial. Traditional methods often prove insufficient in identifying subtle defects hidden within repetitive patterns [25]. Neural networks, with their ability to capture complex dependencies in data, present a promising solution for automated defect detection in symmetrical components [26].

Machine learning and AI has recently become a crucial tool for surface defect detection in industries, gaining significant attention from researchers and professionals [27, 28]. Cumbajin et al. identifies that metal surfaces are the most frequently studied, representing 62.71% of the research, with image classification being the most common problem type. It also highlights the widespread use of performance-enhancing techniques like transfer learning (83.05%) and data augmentation (59.32%), especially when datasets are small or difficult to obtain. Ameri et al. highlights that techniques like transfer learning and data augmentation are essential for improving performance, especially when dealing with limited or hard-to-obtain datasets. Pyramid networks and CNN models are frequently employed, with metal surfaces being the most studied, representing the majority of the research.

The use of machine learning (ML) and deep learning (DL) is an effective approach for detecting and predicting defects, particularly within the context of Industry 4.0, where data-driven techniques enhance automation and predictive maintenance [29]. Awtoniuk et al. applied neural networks to detect defects in engine head castings, using data collected during the production stage, rather than data related to the final products, such as images. The authors explain that choosing measurements during the production process allows for an assessment of the casting immediately after the process is completed, eliminating the need for additional quality checks, such as X-ray or microscopic examinations [30]. The Oborski et al. study presents the development of an intelligent visual quality control system based on Convolutional Neural Networks (CNNs) within the requirements of Industry 4.0. This system integrates machines, operators and process monitoring, enabling the automation of information flow between the management and production levels. The algorithm achieved precision of (99.82%) in defect identification, demonstrating the potential for quality control automation in complex production environments [31].

Wen et al. discuss a machine vision system designed for inspecting the surface quality of bearing rollers using Convolutional Neural Networks (CNNs). The paper outlines the challenges of detecting defects such as scratches, corrosion, and missing material, comparing the proposed machine vision system with traditional methods. The results demonstrate that the system achieves higher accuracy, with precision scores of about 93% and a recall rate of around 90%, outperforming traditional inspection methods, which achieved precision scores between 86.40% and 86.94% [32].

Hena et al. focused on the detection of flaws in aluminium components using digital X-ray radiography. They applied machine learning with k-dimensional trees to evaluate defects according to the ASTM 2973-15 standard. This approach enabled the effective detection, classification, and grading of defects in aluminium castings, supporting automated quality control decisions in manufacturing [33].

Engine components are often a focus of quality control research. In the study by Abagiu et al., attention was given to the cylinder chambers within engine blocks, specifically targeting machining defects on the cylinder walls. The system was developed to automatically detect these defects during production, replacing manual inspection. The CNN model, using RGB images, achieved an accuracy of 100% [34]. Yoon et al. applied machine and deep learning methods to detect defects on piston rods and steering racks using impact response tests. The analysis focused on shifts in natural frequencies, achieving 100% accuracy in identifying defects. They used time and frequency domain data, extracted through Fast Fourier Transform (FFT), and employed feature vectors such as resonance frequencies and energy to train PCA, SVM, and BiLSTM models. This approach promises effective automated quality control in manufacturing [35]. Oeckl et al. in their study found that the use of computed tomography (CT) allowed the detection of piston defects with very high precision. This enabled early detection of problems, such as internal porosity, with an accuracy of less than 0.1 mm. The results also showed that integrating the CT system into the manufacturing process contributed to a significant improvement in quality control and a reduction in the number of defective components [36]. Utami & Leni in their study focused on using a Convolutional Neural Network (CNN) to detect piston damage based on digital images. The dataset consisted of 285 images classified into three categories: pistons with defects (Defected1), pistons with oil stains (Defected2), and normal pistons. The CNN model achieved an accuracy of 72.2% during training and 68.9% during validation, with precision and recall values around 70% [37]. The study by Brambilla et al. presented in this document focuses on the quality control of circularly symmetric components (CSC), such as bearings, clutch discs, gears, and washers, using vision-based algorithms. The authors compared traditional feature-based methods, such as Fourier and signal processing techniques, with deep learning (DL) approaches (MobileNetV2, ResNet50). The results demonstrated high accuracy for both methods, with DL models achieving over 99% accuracy [38].

This article presents a comprehensive study on the application of various neural network architectures for defect detection in rotationally symmetrical components. Key aspects such as data preparation, neural architecture design, training processes, and performance evaluation will be discussed. Particular emphasis will be placed on comparing classical image processing methods [39, 40] with modern deep learning techniques in the context of defect detection. Additionally, the effectiveness of knowledge transfer between different domains [41] will be examined to enhance model performance in

scenarios with limited training data, to effectively identify even subtle visual defects [42]. Case studies and experimental results will be presented, illustrating the effectiveness of the proposed methods in real-world production scenarios.

MATERIALS AND METHODS

A large part of the products produced in various types of factories are characterized by symmetries of various kinds. A special case is products that are invariant due to rotation by a certain preset angle about a certain axis. Sometimes this can be only one angle, sometimes there are several such angles, and sometimes rotation by any angle keeps the product invariant. Thus, if the product is represented by a set of points $A = \{P_i \in R^3\}$, an $\in$ d the function $f: R^3 \rightarrow R^2 \times V$ is used to represent these points, then by rotational symmetry we will mean the property $f(P_i) = f(P_i')$, where P_i' is formed from P_i by rotation about a given angle. The above definition is quite abstract, but in this article all attention will be focused on images of rotationally symmetric objects, which will allow us to demonstrate its operation from the practical side. An example of a rotationally symmetric product is shown in Figure 1. If rays are routed from the centre of symmetry of such an object at different angles, it is possible to reconstruct the image on each ray separately. Such rays can then be grouped into stacks, which will form images with a height given by the number of rays combined. This method will be used as part of the preparation of input data for the machine learning model.

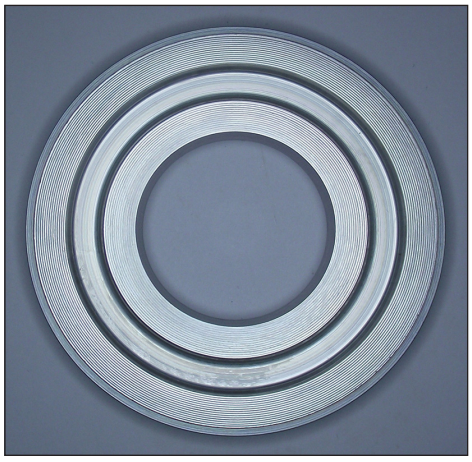


Figure 1. Piston part as an example of a rotationally symmetric semi-finished product

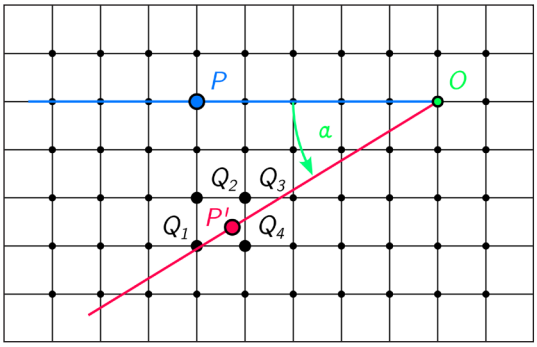


Figure 2. Scheme for determining the new value of a point after performing a rotation by a given α angle based on the values in neighboring points Q_1, Q_2, Q_3 and Q_4

Assume that the point $P = (x, y)$ after performing a rotation by an angle α goes to the point $P = (x', y')$ as shown in Figure 2. The new point P' is surrounded by four points Q_1, Q_2, Q_3 and Q_4, where the imaging function f is defined. Since the action takes place directly on the pixels of the finished image, we can refer to the f function as the f: $R^2 \rightarrow V$, where V is the given color space. In this case, the points Q_i have the following properties:

- $Q_1 = (x_1, y_1), f(Q_1) = (v_1)$: the nearest pixel to the left-down from P',
- $Q_2 = (x_1, y_2), f(Q_2) = (v_2)$: the nearest pixel to the left-up from P',
- $Q_3 = (x_2, y_2), f(Q_3) = (v_3)$: the nearest pixel to the right-up from P',
- $Q_4 = (x_2, y_1), f(Q_4) = (v_4)$: the nearest pixel to the right-down from P'.

where $x_1 = \lfloor x' \rfloor$, $x_2 = \lceil x' \rceil$, $y_1 = \lfloor y' \rfloor$, $y_2 = \lceil y' \rceil$, and v_i' define the color of the point. For the coordinates thus given, we determine the distances $d_x = x' - x_1$ and $d_y = y' - y_1$, which define the horizontal and vertical distances of point P' from Q_1, respectively. The imaging value of a point P' is calculated as the weighted sum of the imaging for the four surrounding points Q_1:

$$f(P') = f(Q_1) \cdot (1 - d_x)(1 - d_y) + f(Q_2) \cdot (1 - dx)d_y + \\ + f(Q_3) \cdot d_x d_y + f(Q_4) \cdot d_x(1 - d_y) \quad (1)$$

If the values of the imaging function are determined for all points of the ray, a ray image will be created, with dimensions of one pixel per ray length. Such images can then be grouped together to form input datasets for the machine learning model. Examples of conversions in which all radial images have been assembled side by side horizontally are shown in Figure 3. They clearly

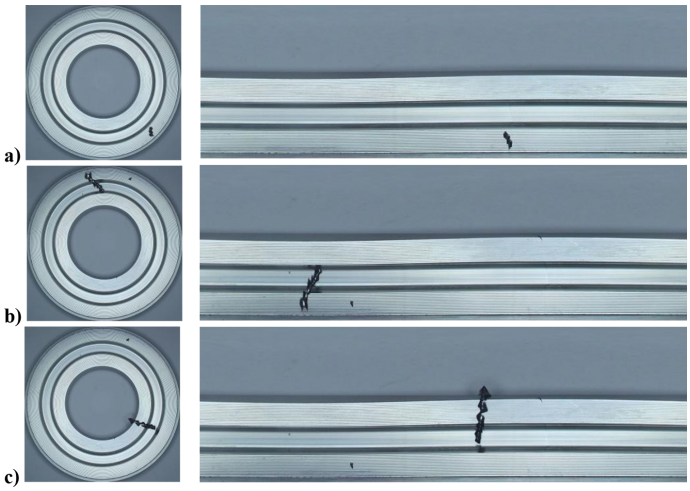


Figure 3. Examples of extracting rays from an image and assembling them horizontally as new images

show how rotational symmetry contributes to making the image more regular.

Data preparation and vector generation process

In order to automate the process of generating vectors for machine learning models, each defect found in the input image was manually marked with the colour green RGB = (0, 255, 0). Figure 4 illustrates how the images were modified to mark the defect. The colour used to mark the defect is not present in any of the original images and can be used to read the position of the defect on individual rays. In doing so, it is important to remember that the process of determining the colours for individual rays is approximative in nature. In practice, this means only that after transformation, the colour of the defect may differ slightly from the base colour RGB = (0, 255, 0). Therefore, the OpenCV library and the HSV palette were used for identification, for which the condition was assumed that a given point contains a defect if its colour meets the condition H∈[50; 70] and $S + V \geq 255$. If this condition is satisfied, the intensity of the defect is expressed by the relation $\frac{S+V}{2}$. It should be noted here that for the OpenCV library, the HSV palette values are in the following ranges: $H \in [0; 180]$, $S, V \in [0; 255]$.

Images of pistons with defects and images of pistons with defects marked in green were taken as input for the process of generating machine learning vectors. Thus, each image was provided in two versions: the original version and the version with a marked defect (see Figure 4). Then, using the implemented software, sets of vectors were generated for machine learning. Each vector consisted of an input tensor containing image data and an output tensor containing information about the location of the defect. For the purpose of verifying the effectiveness of specific machine learning methods, it was decided to group individual rays into so-called stacks of 1, 2, 4, 8, 16 and 32 rays. Therefore, individual tensors represented stacks of rays of a given size. These tensors were developed in two input variants (grayscale palette and HSV palette) and one output variant (intensity of defect in grayscale palette). Figures

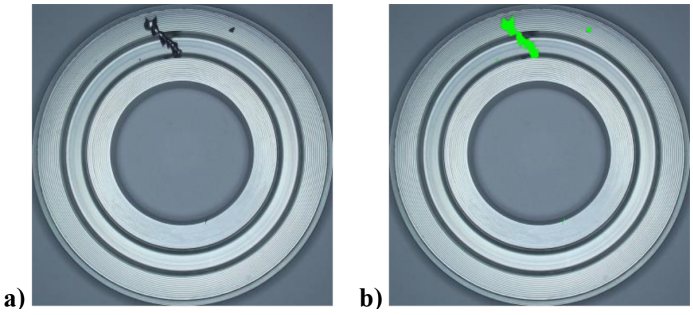


Figure 4. An image of a piston with a visible defect (a) and an image in which the defect was manually marked with the green color RGB = (0, 255, 0) (b) to automate the process of generating vectors for the machine learning model

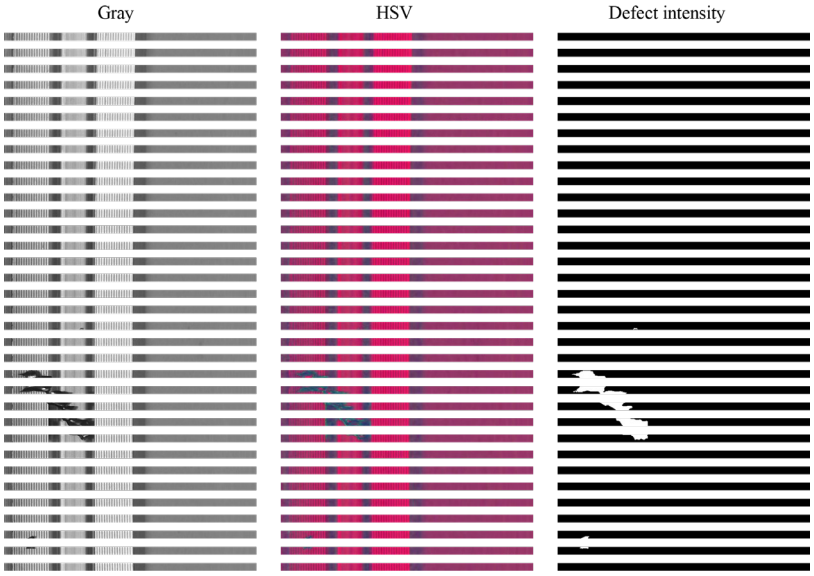


Figure 5. Stacks numbered 0 to 33, which group 16 rays each from the sample image

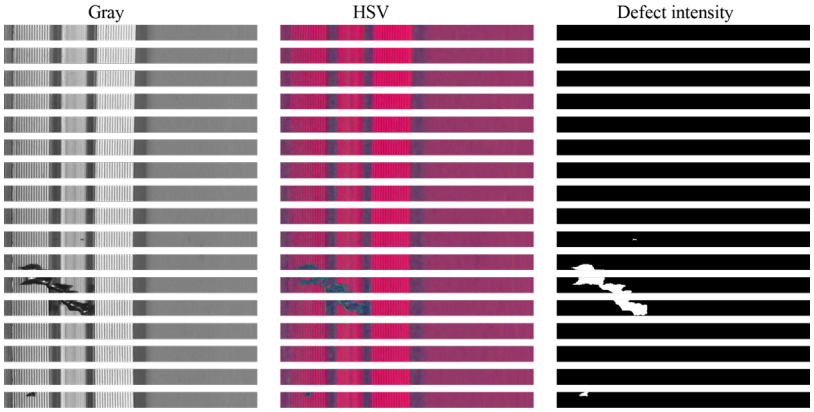


Figure 6. Stacks numbered from 0 to 16, which group 32 rays each from the sample image

5 and 6 graphically show example tensors for ray stacks of size 16 and 32, respectively. In them, it can be seen that in areas where there is a defect, the tensor named Defect intensity is clearly white, while in areas where there is no defect it remains black. For both the grayscale and HSV formats, an identical number of individual rays composed of 512 pixels was prepared. These rays were determined based on 63 images from the quality control process. The images were selected due to the defect they represented, which was the presence of a chip originating from the machining process. The extracted rays formed stacks of size 1. Subsequently, the rays were combined to create stacks of sizes 2, 4, 8, 16 and 32. Table 1 provides information on the number of machine learning vectors depending on the stack size. This table also indicates the number of vectors containing defects. The increase in the percentage of vectors with defects is due to the fact that stacking creates objects containing a significant number of defect-free rays. This is therefore a natural effect that occurs as a result of combining individual rays.

Table 1. The number of vectors in the dataset depending on the size of the rays' stack, along with the number of vectors containing defects in both the gray format and the HSV format

Ray stack size	Number of stacks	Number of stacks with defects	Percent of stacks with defects
1	103 320	5312	5.1%
2	51 660	2712	5.2%
4	25 830	1420	5.5%
8	12 915	763	5.9%
16	6 489	437	6.7%
32	3 276	271	8.3%

Model architecture and operation

The primary goal when designing the architecture of the machine learning model was to achieve a classification that divides objects into two categories: defective and non-defective. The objects to be classified were individual pixels located within a given stack of rays. Typical machine learning models responsible for classification assume that the input data will be assigned to one of several predefined classes. However, in this case, it was decided to build a model that classifies each pixel individually, with the assumption that the input to the model is the entire set of pixels. This means that we are dealing with a situation where each pixel is classified individually, but neighbouring pixels (within the selected stack of rays) are also taken into account.

The main idea of the machine learning model architecture is presented in Figure 7. The first part of the model consists of affine narrowing transformations, followed by a linear expansion transformation, and the entire process is concluded with a sigmoid function. The linear layers are supplemented with additional components as needed to prevent overfitting and the introduction of values that do not correspond to the pixel classification process. For example, the model for grayscale data uses the ReLU (Rectified Linear Unit) function, while the model for HSV data utilizes BatchNorm1d (Batch Normalization 1D), the ReLU function, and dropout. The implementation details of both versions of the model are discussed further in the results section.

RESULTS

The rays grouped into stacks were converted into numerical form and saved in compressed NumPy archive files (NPZ). These served as input values for neural network models. Additionally, as described earlier, for each input tensor, an output tensor was developed that contained information about the location of production defects. The output tensors were also converted into numerical form and saved in NPZ format. The process of preparing tensors for rays defined in grayscale is illustrated with the example code provided below.

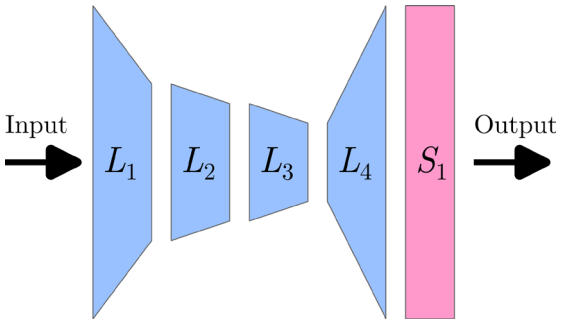


Figure 7. The architecture of the machine learning model and its key components consist of narrowing affine transformations L1, L2, and L3, an expanding affine transformation L4, and the sigmoid function S1

```
# Loading data set.
dataset = np.load('dataset_
   gray_0001.npz')
# Input data.
data = dataset['data']
# Output target.
target = dataset['target']
# Input tensor preparation.
X_tensor = torch.tensor(data,
   dtype=torch.float32)
X_tensor = X_tensor.view(-1,
   ray_stack_size*512)
# Output tensor preparation.
scaler = MinMaxScaler()y_scaled =
   scaler.fit_transform(target.re-
   shape(-1, 1)).reshape(target.shape)
y_tensor = torch.tensor(y_
   scaled, dtype=torch.float32)
y_tensor = y_tensor.view(-1,
   ray_stack_size*512)
```

The preparation of tensors for the HSV colour palette is very similar, with the difference that the hue value has three independent components (an additional tensor dimension is added).

The neural network model for grayscale ray stacks was designed based on four linear (Linear) and activation (ReLU) layers, along with a Sigmoid layer at the output. After creating the model, it is trained on the training data using the MSE Loss function and the Adam optimizer. The training data for the network comprises 80% of all vectors. The following code illustrates the definition of the neural network model used for grayscale rays.

```
class MultiOutputNeuralNetwork(nn.Module:
 def __init__(self, ray_stack_size):
  super(MultiOutputNeuralNetwork,
   self).__init__()
   self.flatten = nn.Flatten()
   self.fc1 = nn.Linear(ray_
    stack_size*512, 128)
   self.fc2 = nn.Linear(128,
   self.fc3 = nn.Linear(64, 32)
   self.fc4 = nn.Linear(32,
    ray_stack_size*512)
   self.relu = nn.ReLU()
   self.sigmoid = nn.Sigmoid()
```

```
 def forward(self, x):
  x = self.flatten(x)
  x = self.relu(self.fc1(x))
  x = self.relu(self.fc2(x))
  x = self.relu(self.fc3(x))
  x = self.sigmoid(self.fc4(x))
 return x
```

In the case of the HSV palette, the complexity of the model was increased by adding additional layers and multiplying the number of neurons. Additionally, to prevent overfitting, dropout was applied during the training phase, which randomly zeros out some of the input units at each update step of the training process. The introduced normalization of tensors before each of the input layers improves the learning quality, stabilizes, and accelerates the training process. Similar to the grayscale case, the training data also constituted 80% of the total available data. The following code snippet illustrates the neural network architecture developed for the HSV palette.

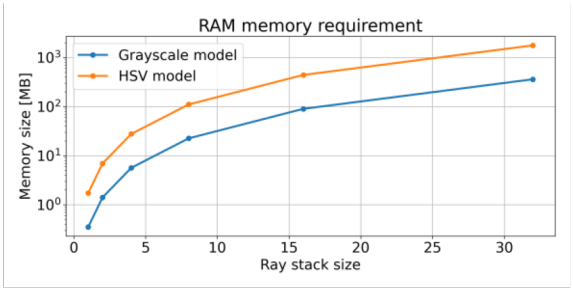
```
class MultiOutputNeuralNetwork(nn.Module):
 def __init__(self, ray_stack_size):\
  super(MultiOutputNeuralNetwork,
   self).__init__()
  self.flatten = nn.Flatten()
  self.fc1 = nn.Linear(ray_stack_size
   * 512 * 3, 256)
  self.bn1 = nn.BatchNorm1d(256)
  self.fc2 = nn.Linear(256, 128)
  self.bn2 = nn.BatchNorm1d(128)
  self.fc3 = nn.Linear(128, 64)
  self.bn3 = nn.BatchNorm1d(64)
  self.fc4 = nn.Linear(64, 32)
  self.fc5 =
   nn.Linear(32,ray_stack_size * 512)
  self.relu = nn.ReLU()
  self.dropout = nn.Dropout(0.5)
  self.sigmoid = nn.Sigmoid()
 def forward(self, x):
  x = self.flatten(x)
  x = self.relu(self.bn1(self.fc1(x)))
  x = self.dropout(x)
  x = self.relu(self.bn2(self.fc2(x)))
  x = self.dropout(x)
  x = self.relu(self.bn3(self.fc3(x)))
  x = self.dropout(x)
  x = self.relu(self.fc4(x))
  x = self.sigmoid(self.fc5(x))
 return x
```

The neural network model for grayscale was designed as a simplified architecture compared to the HSV model. The data inputs have a size of 512 · ray_stack_size, where ray_stack_size denotes the number of rays in a single tensor. The linear layers fc1 – fc4 are four layers with different numbers of neurons in each layer (128, 64, 32, 512) · ray_stack_size. The ReLU activation function is placed after each of the first three linear layers, and Sigmoid on the output layer. The output is a tensor with dimensions 512 · ray_stack_size.

For the HSV palette, the input data has a size of 3·512 · ray_stack_size. There are five linear layers fc1 – fc5 with different numbers of neurons in each layer: (256, 128, 64, 32, 512) · ray_stack_size. Batch normalization is applied after each of the first three linear layers. Regularization with dropout at a rate of 0.5 is applied after each of the first three linear layers. The activation function is ReLU after each of the first four linear layers, and Sigmoid in the output layer. The output is a tensor with dimensions 512 · ray_stack_size.

All computations were performed on a computer equipped with an AMD Ryzen 5 5600H processor (6 cores and 12 threads) and 32 GB of RAM. The calculations utilized the resources of the CPU, and the computational power of the GPU was not used. The implementation of both models was done using the PyTorch library version 2.1.1+cu118. The total memory and time resources consumed by the developed models during training are presented in Table 2 and

a)



b)

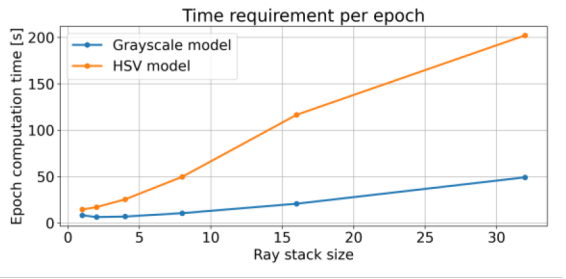


Figure 8. Graphs showing the computational resource usage by the model for grayscale and HSV models depending on the ray stack size: a) RAM usage, b) training time per epoch for the dataset specified in Table 1

Table 2. Computational resource usage by the model for grayscale and HSV models depending on the ray stack size

Ray stack size	RAM requirement [MB]		Epoch training time [s]	
	Gray	HSV	Gray	HSV
1	0.35	1.73	8.397	14.631
2	1.41	6.92	6.419	17.119
4	5.64	27.65	7.004	25.531
8	22.52	110.56	10.617	49.782
16	90.04	442.12	20.784	116.546
32	360.09	1768.23	49.212	202.119

illustrated in Figure 8. The memory demand of the HSV model is always 4.9 times greater than that of the grayscale model for the given stack size of rays. Moreover, for both models, a two-fold increase in the stack size results in a fourfold increase in memory consumption. The relationship between the reserved memory and the ray stack size for both models is depicted in Figure 8a). It should be noted that the vertical axis is on a logarithmic scale. The average training time per epoch for each model was also measured during the research. However, it is difficult to establish similar general patterns for time consumption as those observed for memory usage. The average training time for the examined models is illustrated in Figure 8 b). It is worth highlighting here that for the grayscale model, the computation time for a stack consisting of a single ray is longer than for stacks consisting of two and four rays. This is most likely due to optimizations implemented by numerical libraries. Nevertheless, it indicates that such details should be taken into account when developing solutions where computational efficiency is of particular importance.

Figure 9 presents the learning process graphs of the model for different stack sizes of rays generated based on grayscale. The graphs show the learning accuracy as a function of the number of epochs. As can be observed, the highest accuracy was achieved for stacks composed of one ray and reaching 99.88%. It is also noteworthy that the learning process is exceptionally fast, with the model achieving very good accuracy after just a

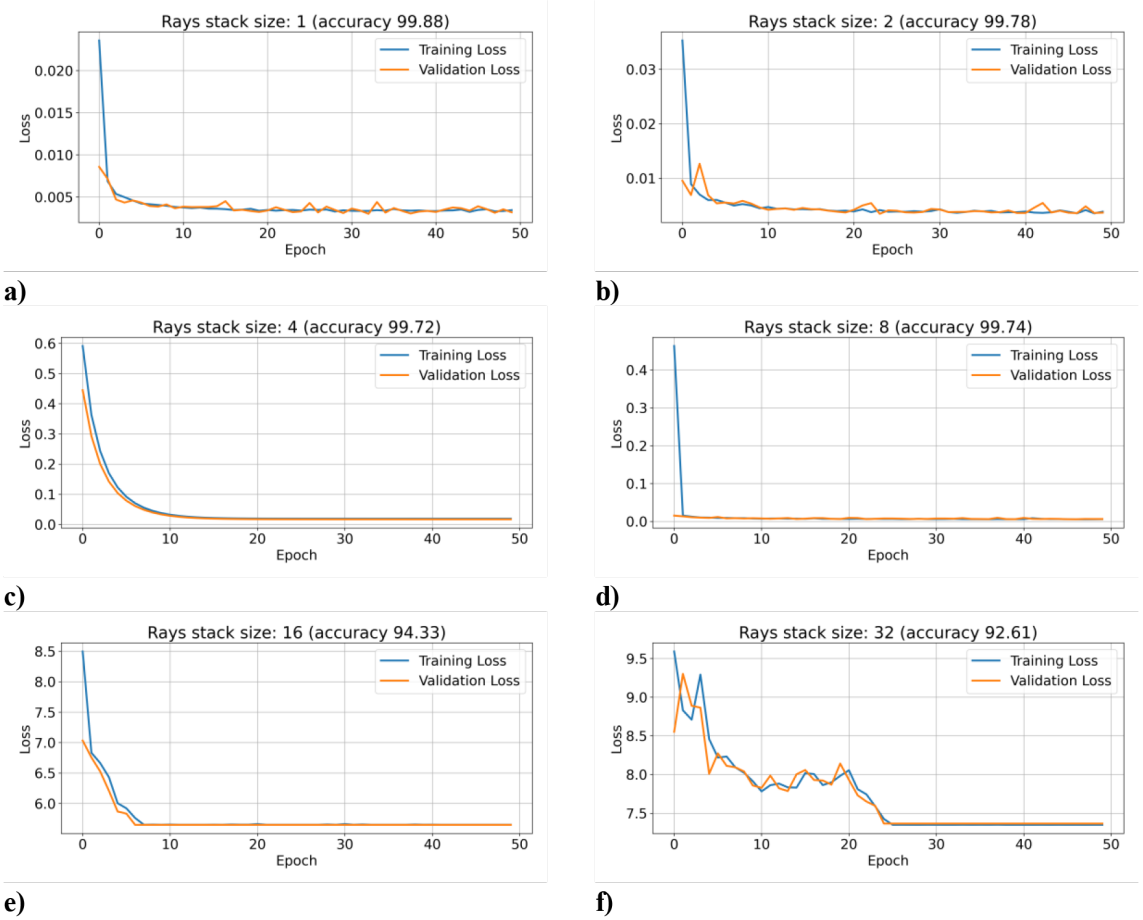


Figure 9. Graphs of learning accuracy as a function of the number of epochs for different ray stack sizes using grayscale imaging

few epochs. It should also be noted that for ray stacks of size 32, the proposed neural network architecture is completely ineffective, as the model's accuracy stabilizes at 92.61%. Models operating based on the grayscale format exhibit very good convergence and provide stable results after just twenty epochs for stacks of 1, 2, 4, 8, and 16 rays. In the case of a stack with 32 rays, stabilization occurs after approximately 30 epochs.

Figure 10 presents the learning process graphs of the model for different stack sizes of rays based on the HSV palette. Similar to the previous palette, the graphs show the learning accuracy as a function of the number of epochs. Analysing the graphs clearly shows that the learning process was significantly extended compared to the model for tensors defined based on the grayscale palette. The best learning result was achieved for double ray stack, where the model exhibited an accuracy of 99.91%. However, it should be emphasized that for stacks of sizes 1, 2, and 4 rays, a very similar accuracy result was obtained. It should be noted, however, that for almost all tested cases (except for the stack of 4 rays), there is a significant difference between the validation loss and the training loss, which may indicate overfitting of the model. In this context, it is also difficult to determine a specific number of epochs necessary to train the model, as there is no convergence between the validation loss and the training loss. Additionally, considering the significantly greater resources consumed by the HSV model compared to the model operating in the grayscale format, its use in this particular case does not seem to be particularly appropriate.

Additionally, two other types of neural networks were compared. The first was an RNN (Recurrent Neural Network). It employed the architecture of a standard fully connected feed-forward neural network. RNN is a type of neural network designed to process sequences of data. RNNs have feedback loops for at least one layer of neurons. The key feature of RNNs is that they have internal states (memory) that allow them to retain information about previous steps in the sequence. The algorithm used the

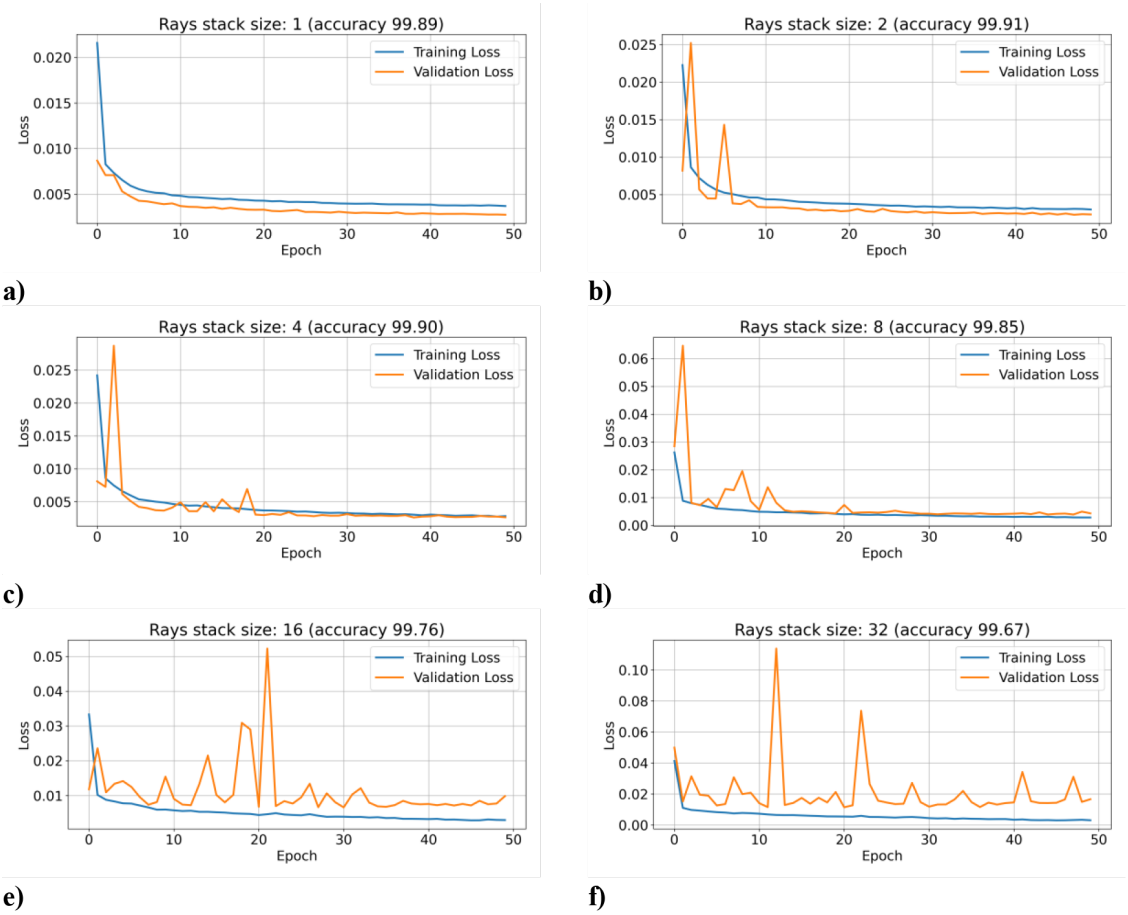


Figure 10. Graphs of learning accuracy as a function of the number of epochs for different ray stack sizes using HSV imaging

parameter batch_size = 20, meaning that the data are taken in batches of 20. The network was built with hidden state vectors of size hidden_size = 200 and num_layers = 15. The Sigmoid function was used as the activator. The optimizer for this network was the Adam algorithm, and the best results were obtained for a training process consisting of 20 epochs. This neural network model achieved an accuracy of 99.75%.

The second neural network model used was an LSTM (Long Short-Term Memory) network, which is used for processing sequential data. The number of units in the hidden layer was set to hidden_size = 700, and the number of LSTM layers was set to num_layers = 30. Binary Cross Entropy Loss was used as the loss function with the Adam optimizer. The accuracy of this neural network model was 99.74%.

CONCLUSIONS

The model built for grayscale is characterized by simplicity and efficiency due to the relatively small number of neurons it comprises. As a result, training is fast and requires minimal computational and memory resources. The architecture of the model improves the generalization of new data and reduces the risk of overfitting. However, the downside of this model is the lack of colour information, which may limit its ability to recognize colour-related features, making it less effective in tasks where colour is a key aspect. This model can be used even on computational platforms with highly limited resources. A particularly potential application is in small integrated circuits mounted directly on optical devices.

The model for the HSV palette uses 3 channels (Hue, Saturation, Value), which allows for capturing more complex colour-related features. Therefore, it can be more effective in tasks related to detection, classification, or segmentation, where colour plays a crucial role. It also allows for better colour selection during the learning process. The model is more complex, enabling better capture of nuances in the input data. This comes at the cost of requiring more computational and memory resources, but the usage is still relatively smaller compared to a typical convolutional network for processing complete images. It should be emphasized, however, that this model requires modifications to ensure convergence between the validation loss and the training loss, which will eliminate the overfitting effect.

Experiments conducted on both models described above, as well as experiments performed on RNN and LSTM, show that the best accuracy is achieved when training models on small number of rays. This is particularly interesting because such images are completely unreadable to the human eye. It is evident, therefore, that a neural network can achieve very high accuracy on data that are very problematic for human analysis. The achieved level of model accuracy is so high that the proposed method can make a significant contribution to enhancing the effectiveness of quality control systems in mass production facilities.

REFERENCES

1. Smith S.J. and Adendorff K. Advantages and limitations of an automated visual inspection system. The South African Journal of Industrial Engineering, 2011, 5(1). https://doi.org/10.7166/5-1-423
2. Pillet M., Baudet N., Maire J.L. The visual inspection of product surfaces. Food Quality and Preference, 2012, 27, 153-160. https://doi.org/10.1016/j.foodqual.2012.08.006
3. Chmielowiec A., Sikorska-Czupryna S., Klich L., Woś W. and Kuraś P. Optimization of quality control processes using the NPGA genetic algorithm. Advances in Science and Technology Research Journal, 2024, 18(7), 264–276. https://doi.org/10.12913/22998624/193195
4. Daneshjo N., Malega P. and Drábik P. Techniques for production quality control in the global company. Advances in Science and Technology Research Journal, 2021, 15(1), 174–183. https://doi.org/10.12913/22998624/131558
5. Imai M. Der Schlüssel zum Erfolg der Japaner im Wettbewerb. Langen Müller, 1992, München, German.
6. Burghardt A., Kurc K., Szybicki D., Muszyńska M. and Szczęch T. Monitoring the parameters of the robot-operated quality control process. Advances in Science and Technology Research Journal, 2017, 11(1). http://dx.doi.org/10.12913/22998624/68466
7. Huang S. and Pan Y. Automated visual inspection in the semiconductor industry: A survey. Computers in industry, 2015, 66, 1–10. http://dx.doi.org/10.1016/j.compind.2014.10.006
8. Davies A. Defective products and product recall issues for businesses. Pitmans LLP, 2014.
9. Minwoo P. and Jongpil J. Design and implementation of machine vision-based quality inspection

system in mask manufacturing process. Sustainability, MDPI, 2022, 14(10): 6009. https://doi.org/10.3390/su14106009

10. Seitz H. Quality inspection with AI vision. Vision & Sensors. System Integration, 2023.
11. Akundi A., Reyna M. A machine vision based automated quality control system for product dimensional analysis. Procedia Computer Science, 2021, 185, 127-134. https://doi.org/10.1016/j.procs.2021.05.014
12. Lewis J. Machine vision system counts, identifies and verifies highly-regulated objects. Vision system design, 2018.
13. Chmielowiec A., Klich L., Woś W., Błachowicz A. Implementation of the Maintenance Cost Optimization Function in Manufacturing Execution System. In J. Filipe, M. Śmiałek, A. Brodsky, and S. Hammoudi, editors. Enterprise Information Systems, Cham, Springer Nature Switzerland. 2023, 133–154, http://dx.doi.org/10.1007/978-3-031-39386-0_7
14. Chmielowiec A. and Klich L. Application of python libraries for variance, normal distribution and Weibull distribution analysis in diagnosing and operating production systems. Diagnostyka, 2021, 22(4), 89– 105. http://dx.doi.org/10.29354/diag/144479
15. Sioma A. Vision system in product quality control systems. Applied sciences, 2023, 13(2), 751. https://doi.org/10.3390/app13020751
16. Schlagenhauf T., Brander T. and Fleischer J. A stitching algorithm for automated surface inspection of rotationally symmetric components. CIRP Journal of Manufacturing Science and Technology, 2021, 35, 169–177. https://doi.org/10.1016/j.cirpj.2021.05.013
17. Baranovskyi D., Myamlin S., Bulakh M., Podosonov D., Muradian L. Determination of the filler concentration of the composite tape. Applied Sciences, 2022, 12(21), 11044. https://doi.org/10.3390/app122111044
18. Baranovskyi D., Bulakh M., Michajłyszyn A., Myamlin S., Muradian L. Determination of the risk of failures of locomotive diesel engines in maintenance. Energies, 2023, 16(13), 4995. https://doi.org/10.3390/en16134995
19. Qi X., Chen G. Y., Li G., Cheng X. and Li C. Applying neural-network-based machine learning to additive manufacturing: Current applications, challenges, and future perspectives. Engineering, 2019, 5(4), 721–729. https://doi.org/10.1016/j.eng.2019.04.012
20. Vladov S., Yakovliev R., Bulakh M., Vysotska V. Neural network approximation of helicopter turboshaft engine parameters for improved efficiency. Energies, 2024, 17(9), 2233. https://doi.org/10.3390/en17092233
21. Bulakh M., Klich L., Baranovska O., Baida A., Myamlin S. Reducing traction energy consumption with a decrease in the weight of an all-metal gondola car. Energies, 2023, 16(18), 6733. https://doi.org/10.3390/en16186733
22. Ren S., He K., Girshick R., Sun J. Faster R-CNN: towards real-time object detection with region proposal networks. Advances in Neural Information Processing Systems, 2015, 91-99. https://doi.org/10.48550/arXiv.1506.01497
23. LeCun Y., Bottou L., Bengio Y., Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998, 86(11), 2278–2324. http://dx.doi.org/10.1109/5.726791
24. He K., Zhang X., Ren S., Sun J. Deep residual learning for image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, 770–778. http://dx.doi.org/10.1109/CVPR.2016.90
25. Xie X. A review of recent advances in surface defect detection using texture analysis techniques. Electronic Letters on Computer Vision and Image Analysis, 2008, 7(3), 1–22. https://doi.org/10.5565/rev/elcvia.268
26. García Peña D., García Pérez D, Díaz Blanco I. and Marina Juárez J. Exploring deep fully-convolutional neural networks for surface defect detection in complex geometries. The International Journal of Advanced Manufacturing Technology, 2024, 134(a), 97–111. https://doi.org/10.1007/s00170-024-14069-7
27. Ameri R., Hsu C. and Band S. A systematic review of deep learning approaches for surface defect detection in industrial applications. Engineering Applications of Artificial Intelligence, 2024, 130, 107717. http://dx.doi.org/10.1016/j.engappai.2023.107717
28. Cumbajin E., Rodrigues N., Costa P., Miragaia R., Frazão L., Costa N., Fernández-Caballero A., Carneiro J., Buruberri L. and Pereira A. A systematic review on deep learning with CNNs applied to surface defect detection. Journal of Imaging, 2023, 9(10), 193. https://doi.org/10.3390/jimaging9100193
29. Johanesa T.V.J., Equeter L. and Mahmoudi S. A. Survey on AI applications for product quality control and predictive maintenance in industry 4.0. Electronics, 2024, 13(5), 976.
30. Awtoniuk M., Majerek D., Myziak A. and Gajda C. Industrial application of deep neural network for aluminium casting defect detection in case of unbalanced dataset. Advances in Science and Technology Research Journal, 2022, 16(5), 120–128. https://doi.org/10.12913/22998624/154963
31. Oborski P. and Wysocki P. Intelligent visual quality control system based on convolutional neural networks for holonic shop floor control of industry 4.0 manufacturing systems. Advances in Science and Technology Research Journal, 2022, 16(2), 89–98. https://doi.org/10.12913/22998624/145503

32. Wen S., Chen Z. and Li C. Vision-based surface inspection system for bearing rollers using convolutional neural networks. Applied Sciences, 2018, 8(12), 2565. https://doi.org/10.3390/app8122565
33. Hena B., Ramos G., Ibarra-Castanedo C. and Maldague X.. Automated defect detection through flaw grading in non-destructive testing digital x-ray radiography. NDT, 2024, 2(4), 378–391. https://doi.org/10.3390/ndt2040023
34. Abagiu M. M., Cojocaru D., Manta F. and Mariniuc A. Detecting machining defects inside engine piston chamber with computer vision and machine learning. Sensors, 2023, 23(2), 785. https://doi.org/10.3390/s23020785
35. Yoon Y., Woo J. and Oh T. A study on the application of machine and deep learning using the impact response test to detect defects on the piston rod and steering rack of automobiles. Sensors, 2022, 22(24), 9623. https://doi.org/10.3390/s22249623
36. Oeckl S., Gruber R., Schön W., Eberhorn M., Bauscher I., Wenzel T. and Hanke R. Process integrated inspection of motor pistons using computerized tomography. Microelectronic Systems: Circuits Systems and Applications, 2011, 277–286. http://dx.doi.org/10.1007/978-3-642-23071-4_26
37. Utami L. P. and Leni D. Modelling piston damage detection using a convolutional neural network based on digital image. International Journal of Multidisciplinary Research and Literature, 2024, 3(2), 172–183. https://doi.org/10.53067/ijomral.v3i2.189
38. Brambilla P., Conese C., Fabris D. M., Chiariotti P. and Tarabini M. Algorithms for vision-based quality control of circularly symmetric components. Sensors, 2023, 23(5), 2539. https://doi.org/10.3390/s23052539
39. Gonzalez R. C. and Woods R. E. Digital image processing. Prentice Hall, Upper Saddle River, 2008.
40. Szeliski R. Computer Vision - Algorithms and Applications. Texts in Computer Science. Springer, 2011.
41. Tan C., Sun F., Kong T., Zhang W., Yang C., Liu C. A survey on deep transfer learning. In: Artificial Neural Networks and Machine Learning – ICANN, Cham. Springer International Publishing, 2018, 270–279. https://doi.org/10.1007/978-3-030-01424-7_27
42. Bergmann P., Löwe S., Fauser M., Sattlegger D., Steger C. Improving unsupervised defect segmentation by applying structural similarity to autoencoders. In Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, SCITEPRESS - Science and Technology Publications, 2019, 372–380. http://dx.doi.org/10.5220/0007364503720380