SETSCI Conference Proceedings
21, 23-32, 2024

International Trend of Tech Symposium
7-8 December 2024, İstanbul - Türkiye

https://doi.org/10.36287/setsci.21.5.023

Hyperparameter Optimization of a Faster R-CNN Model For Fault Detection In Quality Control

Abdullah Yüksel [1], Oğuzhan Karahan [2]

[1] Department of Electronics and Communication Engineering, Kocaeli University, Kocaeli, Turkey
[2] Department of Electronics and Communication Engineering, Kocaeli University, Kocaeli, Turkey

Abstract – Quality control in the industry is becoming increasingly important as a result of increasing market competition and the need for quality products. The increasing complexity of industrial processes and the increase in the amount of data available have promoted the development of intelligent systems for automated error prediction/detection, mainly based on Industry 4.0 technologies and especially deep learning methodologies. Therefore, deep learning has an important position for quality control processes. In this study, we propose an intelligent error/fault detection system that will use the Faster Region-based Convolutional Neural Network (Faster R-CNN) and integrates deep neural networks with the particle swarm optimization (PSO) to self-tune the system hyperparameters and improve its performance. The model will be evaluated on different performance metrics such as accuracy, recall, precision, false alarm rate, false negative rate, F1 score and error rate. Finally, the mean average precision of the proposed PSO based Faster R-CNN model is 97.96%. The experimental results illustrated that the PSO based Faster R-CNN has a good accuracy and fast detection ability by a large margin. In conclusion, it can be inferred that the hyperparameter optimization in fault detection systems based on DL model has the high importance and effects.

Keywords – Deep learning, Fault Detection Systems, Particle Swarm Optimization, Hyperparameter optimization

I. Introduction

With the development of production technology, production numbers continue to rise rapidly. This means that the duration of quality control processes is extended. As production quantities increase and process times are shortened, the risk of production errors increases at the same rate. Visual or manual inspection processes with the help of quality control personnel cannot completely prevent these errors. As a result of this error capture process with the human eye, it is only proportional to how much the person can concentrate his focus and open his perceptions during the work. Quality control personnel are located at the end of the production line and try to catch the produced errors visually. There are also differences in the interpretation of errors in quality control processes based on human decisions. Many factors have an impact on the result, including the current working conditions, environmental conditions and psychological status in the interpretation of the error by the person who checks the error.

The transformation from traditional to smart industries proposed by the Fourth Industrial Revolution (I4.0) has taken on particular significance in various settings such as industry, education, and science [1]. This highlights the need for technological and methodological solutions demanded by the context so that integrating traditional physical systems with a modern digital system does not generate resistance in industries and provides smart industries with flexible tools that allow progressive evolution. Decision Making Processes (DMPs) are of great benefit by emphasizing the importance of information availability, access, and analysis to improve the relevant industrial system and access solutions. Decision-making is a broad term found in all industrial sectors and fields, which can be understood as a sequential process from the definition of the problem to the verification and effectiveness of the adopted decision. As is known, decision-making is based on available information. This has led to the development of multiple and innovative data processing and analysis solutions, generating more interest in various industrial sectors for its application. In this sense, in addition to production functions, maintenance and quality are the most sensitive functions in terms of production. Therefore, the effects of decisions and their consequent actions cannot be ignored because of their impact on continuity and synchronicity [2]. Usually, at these checkpoints we find people performing visual inspection tasks of the product or part of it. The accuracy and efficiency of decision-making human decisions will not always be at their best, so the error-based decision-making process, the large continuous flow of data, the heterogeneity and the variability of the available data make it very difficult to use classical data analysis techniques for decision-making, as in the field of failure risk. As a result, the development of data analysis systems based on Artificial Intelligence (AI) techniques can be increased by using machine learning and deep learning algorithms.

Current AI enhancements allow for the selection of a specific methodology depending on the problem, data types, complexity, and expected responses. However, the same complexity and variety of options have created difficulties in the industrialization and implementation of solutions in modern production systems, resulting in even greater resistance precisely in small and medium-sized companies, which can significantly benefit from the use of these technologies. In particular, today, the use of deep learning techniques for online fault detection allows data or image analysis to be performed automatically immediately and with successful results, preventing process interruptions and wasted time [3]. The need for expertise in building and configuring

intelligent solutions, such as those based on deep neural networks and advanced deep learning algorithms, is challenging due to the variety of possible applications and the complexity of industrial systems. Additionally, there is a gap in developing comprehensive proposals to incorporate these methodologies into manufacturing processes, facilitating the industrialization of deep learning solutions. From now on, we refer to the term industrialization as the stage of scaling up any AI-based prototype to a manufacturing facility.

The Convolutional Neural Network (CNN) is a high-fidelity deep learning model used for recognition and classification tasks with short training times. With the help of filters or cores, they can automatically extract features without human intervention and are more successful in obtaining the best features compared to other models [2].

The development of the CNN model to Faster R-CNN is a transition to a structure that combines object detection and classification. While CNN basically classifies by extracting features from the image, Faster R-CNN also performs object detection and develops through the following steps:

- Backbone (Feature extraction): Faster R-CNN extracts features from the image using the base layers of CNN (e.g. VGG, ResNet). This stage produces the feature maps necessary for classification, as in CNN.
- Region Proposal Network (RPN): Unlike CNN, the RPN layer added in Faster R-CNN suggests regions on the image where possible objects are located. RPN identifies regions that contain objects by filtering out regions that are not objects.
- RoI Pooling (Region of Interest): RPN's suggested regions (suggested boxes) are identified and matched with the feature map extracted from CNN. These zones are used for classification and positioning by bringing them to a fixed size.
- Final layers (Classification and Regression): Unlike CNN, Faster R-CNN predicts the object's position (bounding box) as well as determining the object class.

With this process, Faster R-CNN can both detect objects and classify them correctly. RPN is much faster and more efficient than the methods used before. A single network performs both zone recommendations (RPNs) and classification of these zones, which speeds up the process. The classification of objects and the determination of their location are done together [19].

The success of deep learning techniques compared to other traditional techniques is recognized. However, they have many hyperparameter values, which significantly affect their performance. Manually determining the optimal values is difficult and time-consuming. Therefore, various optimization methods are used to determine the optimal value. Optimization methods are used to develop models with acceptable performance and proximity to the best. They improve the generalization ability of models and achieve higher training and testing performance [4]. Meta-heuristic optimization algorithms allow for a more dynamic and comprehensive exploration of the search field. Therefore, it is more advantageous to use metaheuristic optimization algorithms.

In this study, he presents an intelligent approach that contributes to the industrialization of deep learning solutions. It involves self-tuning of hyperparameters with adaptive capabilities that require negligible human experience. Our method integrates a module of deep neural networks of different natures that apply the Particle Swarm Optimization (PSO) optimization technique.

In this study, he presents an intelligent approach that contributes to the industrialization of deep learning solutions. Particle Swarm Optimization (PSO) is chosen as the optimization method in the study. PSO is a well-known meta-heuristic optimization technique used for hyperparameter optimization in deep learning models. PSO mimics the behaviour of flocks, such as flocks of fish and birds. In PSO, a group of particles adjust their position around the hyperparameter space based on their own and their neighbours' best performance [4].

Galindo-Salcedo et al. [6]. Fault detection is a special case of DMP that has a very important role in the high efficiency of the production processes of many production systems. Advances in artificial intelligence methodologies and computational resources have made it possible to develop different proposals, including artificial intelligence algorithms in their formulations, for online fault detection in the workshop of different industrial systems.

Likewise, multiple applications of artificial intelligence are available for the control and quality assurance processes of different production systems.

Chen et al. [7] used a convolutional neural network to visualize manufacturing defects in textiles by analyzing images obtained directly from the production line.

Rožanec et al. [8] illustrate the practice of quality control of brand printing on the products produced by a company in a real-world use case, highlighting the rapid growth that visual inspection has had in recent years, accompanied by intelligent methodologies.

Lu et al. [9] use a real-time surface defect (wear and misalignment) detection system connected to a robotic arm equipped with an additive extrusion system.

Similarly, Li et al. [10], in this study, describe various types of defects (pores, grooves, and blockages) in a metal additive manufacturing process that combines metallic wire and arc welding.

Furthermore, Zhang and Zhao [11] use a neural network applied to the machines used in such processes, known as LPBF (Laser Powder Bed Fusion), to identify visual defects in 3D printed metal products.

The above-described studies for fault diagnosis and quality determination offer solutions to various problems and applications that are of great interest today.

However, they do not mention the structuring process of methods and developments, but only reveal the hyperparameter values used for the specific application, which casts doubt on the adaptability and generalization of these methods to other situations of modern industry. Since the success of smart models in various applications depends largely on the initial configuration given, the model will be self-defeating in a different situation that may occur. The model ultimately involves the selection of several hyperparameters that determine the performance and convergence of the proposed solutions in combination.

Related to the above, when defining a neural model, it is necessary to define a series of hyperparameters that, on the one hand, determine the architecture of the neural network used, such as the size of the kernel or hidden layer, and on the other hand, a series of hyperparameters that define factors specific

to the training algorithms, such as the learning factor and the size of the training group. This will determine the behaviour of the learning algorithm during the training phase.

Said hyperparameters determine and control the learning process that takes place in IA. Therefore, the hyperparameter search and selection process is crucial to achieving the effectiveness that models need for their applications, and it is necessary to precisely identify each of these values. These are determined prior to training and should be configured for algorithmic and/or neural models, following some selection methodology. Manual, grid or random search are some of the methodologies that have been widely used in various studies for hyperparameter search in recent years [12].

The use of machine learning and supervised learning algorithms for defect recognition, diagnostics and fault prediction is one of the sectors that has developed strongly in recent years through the digitization and use of data for the creation of representative models that allow the development of quality and maintenance policies aligned with the business objectives of the industries.

Artificial intelligence techniques combined with optimization methods facilitate the industrialization of self-adjusting smart solutions based on deep learning, enabling objective information to be obtained without the intervention of psychological, social or personal factors to reduce the risk of decisions.

Interest in these expert systems with low human knowledge requirements has been growing in recent years due to the improvements in efficiency and effectiveness they provide, contributing to the ultimate goals of organizations that use them. At the same time, independence from human expertise gives them the characteristics of objectivity and precision, reducing the risk involved in DMP.

In this study, an error detection system algorithm based on hyperparameter optimization is proposed instead of a deep learning model with manually adjusted parameters. The CNN model is used as a classifier. The goal is to optimize methods to improve the performance of the model.

The contributions of this study can be summarized as follows:

- Instead of investigating the properties on the image with the CNN model, the focus is on the features on the object in the image with the Faster R-CNN.
- In model training, two different features were used with the Callback method, these are EarlyStopping and ReduceLROnPlateau.
- EarlyStopping: Stops training when the model's validation loss has not recovered over a period of time.
- ReduceLROnPlateau: Reduces the learning rate when the model's validation loss doesn't improve.
- With these callback features, it is ensured that the training time is reduced to the optimum level.
- An algorithm is proposed that automatically solves the problems of manually tuning the hyperparameters of deep learning models. Improves the performance of models.
- The proposed model will be trained specifically with data collected from the field.
- Different evaluation criteria were used in the evaluation of the proposed models using the optimization method.
- Other deep learning models created in the literature were examined to evaluate the approaches.

II. Materials and Method

In this study, a deep learning-based error detection system tuned using particle swarm optimization (PSO) for error detection is introduced. Although Faster R-CNN excels at automatically extracting features from the dataset, it requires a number of parameters to extract features on the image. The detection algorithm starts with the pre-processing of the dataset through data cleansing and processing steps. The choice of hyperparameter values is very important for DL techniques. Therefore, a metaheuristic optimization method, PSO, is used for hyperparameter selection. Hyperparameter optimization improves the performance of models. Finally, the evaluation step of the created models is performed. Figure 1 shows the proposed algorithm.

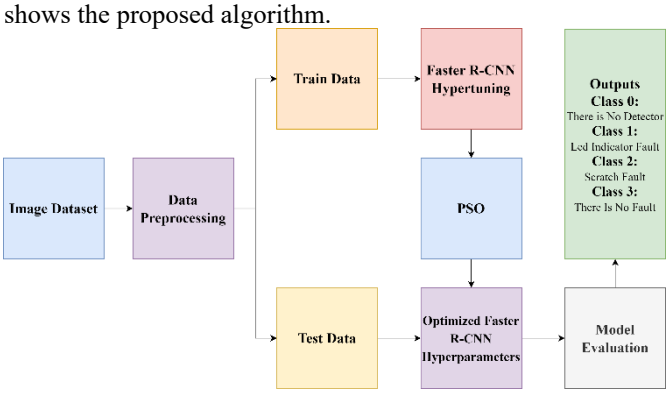


Fig. 1 Proposed workflow scheme for product fault detection

A. Data Description and Preprocessing

Preparing the dataset is a critical step for the effectiveness of deep learning models. This stage is one of the important processes that directly affects the success performance of the model and is necessary to make the training process more efficient. A well-prepared dataset allows the model to be trained faster and also allows the model to produce results with higher accuracy. Therefore, working meticulously at every step of the dataset preparation stage is vital for the success of deep learning projects.

In this study, the dataset collected from the field was carefully examined and necessary precautions were taken to prevent missing or erroneous data from damaging the model. The dataset was carefully checked for missing values that could negatively affect the analysis. Examining the error conditions means correctly detecting the missing data and removing this data from the dataset or filling it in appropriately. Thus, the model was trained more healthily and the overall success was increased.

The dataset was also evaluated in terms of classification processes. In order for the classification algorithms to work effectively, the data must be categorized correctly. In this study, the classifications made on the dataset were analyzed in detail in order to increase the performance of the model. Detailed information and data regarding the classifications are shown in Table 1 below. This table shows the classification results and the distribution of erroneous data for the dataset used in the study. The classifications are in Table 1 below.

Table 1. Sample images of the classes from dataset

Fault Name	Type1	Type2	Type3
There is no detector			
Led Indicator Fault			
Scratch Fault			
There is no fault			

The process of converting categorical features into numerical values performed using label encoding. The dataset divided into three clusters: training, validation, and testing, with ratios of 60%, 28%, and 12%, respectively. The test and training image numbers according to error class names are in Table 2 below.

Table 1. Dividing the datasets into training and testing

Class Name	**Fault Type**	**Amount of Training images used**	**Amount of Test images used**
0	There is no detector	972	195
1	Led Indicator Fault	972	195
2	Scratch Fault	972	195
3	There is no fault	972	195

Standardization for numerical features is necessary to ensure consistency and regular scales. However, in this study, normalization is used instead. The primary purpose of normalization is to preserve the data's behaviour by transforming all features to lie between 0 and 1. In this study, the normalization step was performed using min-max scaling, as shown in equation (1).

$$x_i = \frac{x_i - Min}{Max - Min} \quad (1)$$

In this study, the dataset collected from the field used. It contains 3 types of errors and is the state that should be in 1 case.

B. Deep Learning Models

Machine learning includes deep learning as a subfield. It forms the basis of artificial neural networks. It is used to solve complex problems. It has an important place in the field of data science due to its ability to learn data representations. With its layered structure, it enables feature representations to be learned automatically, eliminating the need for time-consuming feature engineering and contributing to time efficiency. Today, deep learning is widely used in many different fields such as natural language processing, sentiment analysis, image recognition, etc.

Faster Region-Based Convolutional Neural Network (Faster R-CNN): Faster R-CNN is a deep learning model used for object detection. The term "R-CNN" is an acronym for "Region-based Convolutional Neural Networks", and this model developed to perform object detection tasks faster and more accurately [19].

Convolutional Neural Networks (CNN), a popular deep learning model, is inspired by the innate visual perception abilities observed in living organisms [13]. For quality assurance and defect identification status, a CNN is recommended as it provides visual processing capability from the images given as an input dataset. Convolution, pooling, activation layers, fully connected layers for classification, and output layers make up the architecture of convolutional neural networks. CNN is a type of artificial neural network that is widely used in image processing and computer vision tasks. CNNs are designed to automatically learn features in images.

CNN layers:

- Convolutional Layer: Creates feature maps by applying filters (kernels) to the image.
- Pooling Layer: It is used to reduce the size of feature maps, which reduces computational overhead and prevents overfitting.
- Fully Connected Layer: Used to classify results; it is usually found at the end of CNN.

For feature learning, the convolution and pooling layers are used, while the fully connected layers and the output layer are used for classification. The convolutional layer is the most basic layer used in architecture and focuses on the use of cores or filters. The kernel is rotated over the spatial dimensions of the input to create a feature map. The main advantage of the feature map is that it can retain all the distinctive and important features. By using the pooling layer, the dimensions of the feature map are reduced, only important features are preserved, and data errors are reduced. The activation layer is typically a non-linear layer that follows each layer. Sigmoid, hyperbolic tangent, softmax, and ReLU are all activation models that are commonly used in deep learning. The fully connected layer is where the classification and recognition processes take place. The matrix is flattened before reaching this layer, which resembles the arrangement of neurons in a conventional neural network. Therefore, all nodes in a fully linked file are directly linked to all nodes in the layers that precede and follow them. CNNs can quickly capture complex features of images.

R-CNN (Regions with CNN features) developed in 2014 and is considered an important step in object detection.

R-CNN consists of the following phases:

- Region Proposal: Potential object regions (region proposals) are determined on the image. This stage is usually done using an algorithm (for example, Selective Search).
- Feature Extraction: These identified regions are fed to a CNN and features are extracted from each region.
- Classification: The extracted features are classified by a classifier (usually SVM).
- Regression: The bounding boxes of regions can be corrected for more precise object positions.

This structure of R-CNN requires a lot of calculations and is a slow process, because it is necessary to run CNN separately for each region [20].

Fast R-CNN is a faster and more efficient version of R-CNN. Fast R-CNN acts on the entire image with a single CNN, and then takes action on region recommendations using feature

maps from that image. This is faster because the entire image is processed only once.

Faster R-CNN builds on top of Fast R-CNN and further optimizes the region recommendation phase [21].

Key components of Faster R-CNN:

- Region Proposal Network (RPN): A network that proposes object regions in the image. The RPN is embedded within a specific CNN network and quickly creates potential object zones.
- Feature Map: RPN uses and operates on feature maps issued by CNN.
- RoI Pooling: Using the designated zones (RoI), the characteristics of these zones are extracted and processed in a similar way to Fast R-CNN.
- Classification and Regression: Specified regions are used to predict classes of objects and more accurate boundary box positions.

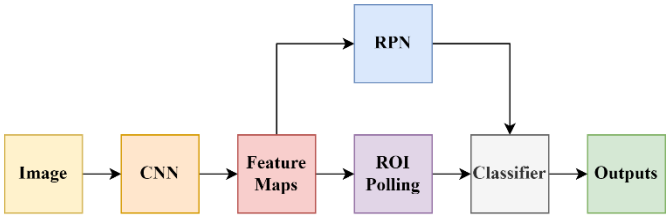


Fig. 2 Proposed workflow scheme for Faster R-CNN

Region Proposal Network (RPN): The RPN is used to identify potential object areas (region proposals) in the image.

- RPN conv: It is a convolution layer for rendering the feature map.
- RPN clss: A Conv2D layer to classify whether the proposed regions are objects or not.
- RPN reg: It is another Conv2D layer to improve the accuracy of the proposed regions.

Faster R-CNN is an improved version of R-CNN and Fast R-CNN. RPN enables real-time object detection by increasing the speed of the model. In this way, both high accuracy is achieve and processing time is significantly reduced. This structure has led to significant progress in the field of object detection and is used in many applications today.

C. Optimization Tasks

Deep learning (DL) models have complex structures. Before learning begins, there are model-specific hyperparameters that need to be tuned. These hyperparameters significantly affect the training, behaviour, and performance of the model. Manual tuning of model hyperparameters can be time-consuming due to the complexity of the models. For this reason, optimization methods have emerged in which models are automatically optimized. Metaheuristic algorithms are techniques used to solve optimization problems across large search domains. They have the ability to efficiently search the area for optimal or approximate solutions. Therefore, due to their efficiency, they are suitable for use in optimization problems with large configuration areas. In this study, a metaheuristic algorithm called PSO used. PSO allows each particle to communicate with other particles at each iteration to determine and update the current global optimum.

Particle Swarm Optimization (PSO): The PSO algorithm is a swarm-based meta-heuristic optimization algorithm proposed by Eberhart and Kennedy (1995). Animal social behaviours, such as the behaviour of insects, flocks, birds, and fish, are simulated by the PSO algorithm [14]. The primary goal of PSO is to mimic the behaviours that individuals perform in order to survive the longest. In PSO, each individual is considered a particle and interacts with other particles. Each particle acts according to its own past decisions and the decisions of the herd. During this process, the particles evaluate their performance and compare it to their previous state, trying to choose the best decision for themselves.

The PSO algorithm is a multi-threaded, swarm-based search method. The number of particles in the swarm is denoted by the integer S. In a given set, each particle has two vectors of length N. The size of the problem is determined by N. The position vector, which indicates the exact position at that moment, is the first vector. The second vector, the velocity vector, controls the direction and velocity of the particle in the next iteration. A particle remembers the optimal position and speed of the herd, in addition to its own position. Based on his previous experience, he adjusts each particle position to the best position. In each iteration, the information of the particles is combined, the velocity of each dimension is adjusted, and this velocity is used to calculate the new position of the particle. The particles constantly change their position until they reach their optimal position.

Equations (8) and (9) describe how the PSO algorithm updates the particle velocity and position of the swarm:

$$V_{id}^{k+1} = WV_{id}^{k} + c_1 r_1^k (pbest_{id}^{k} - x_{id}^{k}) + c_2 r_2^k (gbest_{id}^{k} - x_{id}^{k}) \quad (8)$$

$$x_{id}^{k+1} = x_{id}^{k} + V_{id}^{k+1} \quad (9)$$

Recommended Optimization Framework: This section outlines the steps of the proposed optimization framework, which aims to determine the hyperparameters for the Faster R-CNN deep learning model. The hyperparameters targeted for optimization in this study include the number of epochs, learning rate, number of dense layers, number of neurons within those layers, drop rate, batch size, and learning rate. The search ranges for these parameters are provided in Table 2. The Particle Swarm Optimization (PSO) algorithm is employed to select optimal hyperparameter values, with the goal of maximizing accuracy on the training set. Below is a detailed outline of the proposed algorithm's steps:

Step 1: Initialization

The first step involves initializing the algorithm. Key parameters are determined, including population size, stopping criteria, maximum number of iterations, and the necessary hyperparameters. The initial population is established, where the positions of individual particles correspond to the hyperparameters that require optimization.

Step 2: Evaluation

In the second step, the model is trained solely on the training set. The performance of the trained model is then assessed using the validation set to evaluate its accuracy.

Step 3: Update Personal Best Position (pbest)

The best position of each particle, referred to as personal best (pbest), is updated based on the conformity value obtained from the evaluation in Step 2.

Step 4: Update Global Best Position (gbest)

Among the available personal best positions, the one with the highest fitness value is updated as the global best position (gbest). This step ensures that the algorithm retains the most optimal solution identified thus far.

Step 5: Update Particle Positions and Velocities

For each particle in the swarm, equations (8) and (9) are applied to update the positions and velocities, enabling the particles to explore the hyperparameter space more effectively.

Step 6: Iteration or Termination

The process continues until either the maximum number of iterations is reached or the stopping criterion is met. Once the stopping criterion is satisfied, the optimized hyperparameters are derived from the global best position (gbest). If the stopping criterion is not yet met, the algorithm loops back to Step 2, repeating the evaluation and optimization process.

In summary, the PSO algorithm systematically searches for optimal hyperparameter values, with a specific focus on maximizing accuracy on the training set. This iterative approach not only enhances model performance but also ensures a robust exploration of the hyperparameter space.

Table 2. The defined hyperparameters and their range

Hyperparameter	Ranges
Filter1	24-64
Kernel1	4-6
Filter2	52-76
Kernel2	2-4
Filter3	100-148
Kernel3	2-4
RPN Filter	450-525
RPN Kernel	2-5
Dense Unit	500-600
Drop Out Rate	0.3-0.5
Learning Rate	0.001-0.0001
Batch Size	5-60
Epoch Size	50-80
Anchor Box	8-11

In the PSO algorithm, the population size is set to 100 and the maximum number of iterations is set to 10. In addition, other parameters of the PSO, i.e., scaling factors =1.5 and =1.5 and inertial weight W=0.5, are configured.$c_1 c_2$

III. Results

The error detection results of the Faster R-CNN model, which works with hyperparameters optimized for the PSO, are presented in this section. A dataset consisting of images collected from the field is used in the experiments.

D. Evaluation Criteria

The performance of the proposed models evaluated using the same evaluation criteria applied to most of the previous research on error detection systems. These metrics consist of specificity, error rate, false negative rate (FNR), recall, false alarm rate (FAR), accuracy, precision, recall and F1 score [15].

Equations (10)-(16) provide mathematical formulas of metrics

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \quad (10)$$

$$Precision = \frac{TP}{TP + FP} \quad (11)$$

$$Recall = \frac{FP}{TN + FP} \quad (12)$$

$$F1\_Score = \frac{2 * Precision}{Precision + Recall} \quad (13)$$

$$Specificity = \frac{TN}{TN + FP} \quad (14)$$

$$FAR = \frac{FP}{TN + FP} \quad (15)$$

$$FNR = \frac{FP}{TN + FP} \quad (16)$$

$$ErrorRate = \frac{FP + FN}{TP + TN + FP + FN} \quad (17)$$

Where:

- TP-True Positive
- TN-True Negative
- FP-False Positive
- FN-False Negative

Test loss is a value that measures how well the model performs on the test data. A low loss indicates that the model is highly capable of making accurate predictions.

The accuracy stated here is often used in the same sense and indicates the overall performance of the model. Precision indicates how many of the samples that the model predicts as positive are actually positive. That is, it is the ratio of the model's true positive predictions to the total positive predictions. Recall measures how much of the true positive samples are correctly predicted by the model. That is, it is the ratio of true positives among true positives. The F1 score is a measure that combines precision and sensitivity. It combines these two metrics together to provide balance. A high F1 score indicates that the model performs well in terms of both precision and responsiveness. False Acceptance Rate refers to the proportion of cases where the model incorrectly makes a positive prediction. The False Rejection Rate indicates the rate at which the model incorrectly predicts true positives as negative.

These metrics indicate the model's success in the object detection or classification task. In general, high accuracy, precision, sensitivity, and an F1 score indicate that the model is effective; Low false acceptance and false rejection rates increase its credibility.

E. Performance Analysis and Results

The experiments in the study were carried out using a dataset created from images collected from the production site. Advanced callback methods were used in model training. These are Early Stopping and Learning Rate Scheduler callbacks. With these callbacks, the model was trained in the most optimal time and speed. In trainings performed with traditional methods, large time losses and unnecessary learning results can occur.

A deep learning model is used to classify errors. PSO is used to determine the best hyperparameter values for these models. The PSO algorithm selects the best hyperparameter values that maximize accuracy in the dataset. PSO streamlined the time-consuming and complex process of hyperparameter tuning and had an impact on classification performance.

The optimal hyperparameter values obtained for the model are presented in Table 4.

Table 3. Optimal hyperparameters

Hyperparameter	Optimal value
Filter1	34
Kernel1	6
Filter2	54
Kernel2	3
Filter3	105
Kernel3	2
RPN Filter	484

RPN Kernel	3
Dense Unit	600
Drop Out Rate	0.5
Learning Rate	0.0001
Batch Size	5
Epoch Size	69
Anchor Box	8

Models are trained using a training set. The accuracy of the training set is verified by the validation set. The performance of the models is evaluated on test data that has not been used before. The results of the Faster R-CNN model can be tracked using a variety of metrics. Figure 4 shows the Confusion matrix for the proposed model. The confusion matrix is a table to evaluate the performance of a classification model and to reveal which classes it succeeds and fails. This table allows the model's predictions to be compared with real classes and allows for the calculation of various metrics (accuracy, precision, recall, F1 Score). In the confusion matrix in Figure 4, it can be concluded that the proposed model identifies errors of all classes at a very high rate. Figure 5 shows the state of the error rate of the model with respect to the epoch number. Figure 6 shows the state of the accuracy rate of the model with respect to the number of epochs.

The results show that Faster R-CNN has superior performance. Specifically, the Faster R-CNN model's higher accuracy, precision, recall, F1 score, and specificity indicate that it discriminates more effectively between positive and negative classes. In addition, the lower false alarm rate, false negative rate, and error rate of the Faster R-CNN model demonstrate its reliability in reducing false predictions. This highlights Faster R-CNN's balanced and reliable performance in both accurately identifying positive class and reducing false alarm and false negative rates.

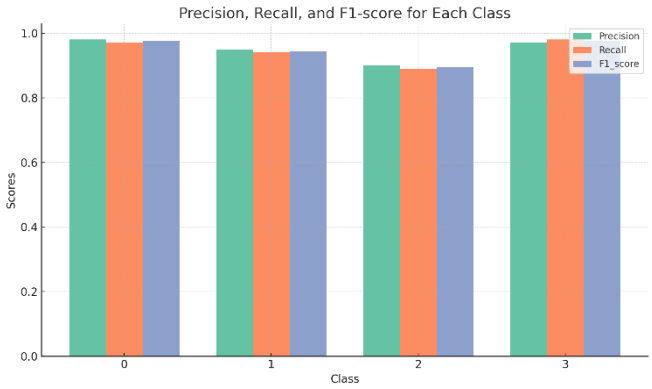


Fig. 3 Precision, recall f1_score is represented in each class

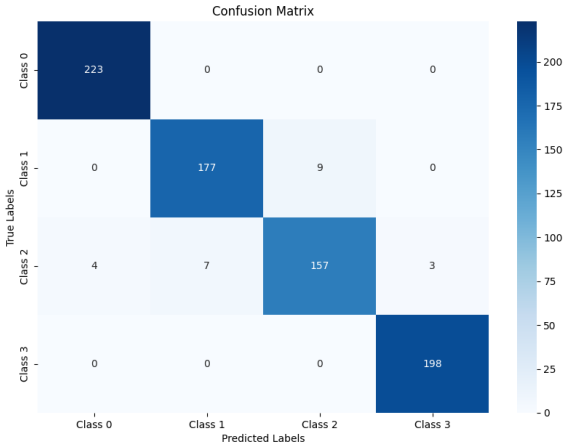


Fig. 4 The fault detection results using the confusion matrix

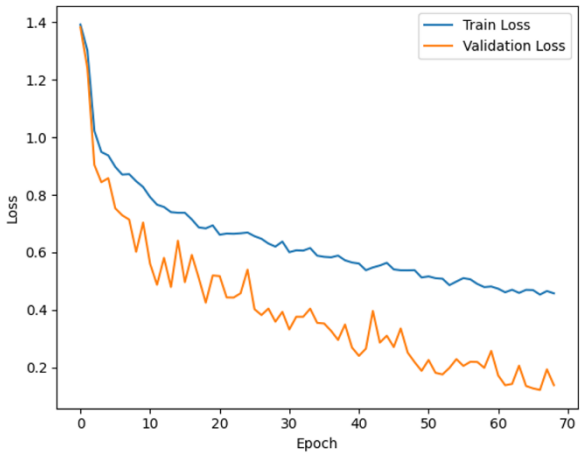


Fig. 5 Training and validation loss of the proposed model

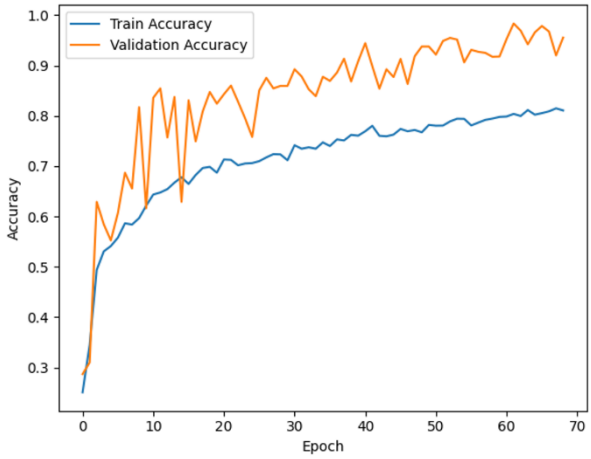


Fig. 6 Training and validation accuracy of the proposed model

F. Real-Time Classification Results

This section will show the classification results with real-time images taken using optimal hyperparameters and the faster R-CNN trained model with PSO optimization. In total, there are 4 classes of images that are included in the training. In addition to these, another fault that is not included in the training but is expected to be converged as an fault by the model will be shown and expected to be classified as an fault. Thus, it will show that our proposed model can adapt to newly formed errors that are not taught to it.

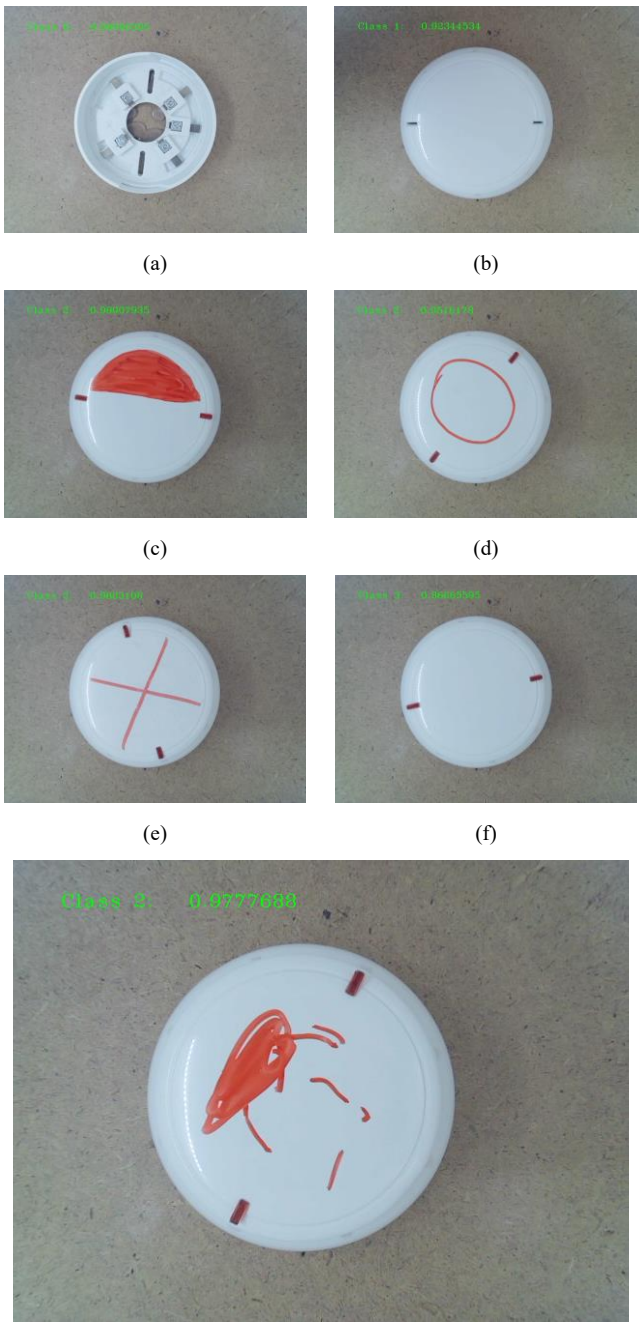
Situations to be detected in real-time product tracking:

1. Detection of the absence of a detector

2. Detection of the situation where there is no led indicator in the detector
3. If three types of error condition that may occur in the detector are perceived as stain errors
4. Detection of the absence of errors in the detector.

In addition, there is another error condition that does not enter the training, but the model is expected to detect that there is an error.

The model reveals the probability that this fault is an fault as a result of fault detection. This fault probability and class are showing at the top of the screen.



(a) (b) (c) (d) (e) (f) (g)

Fig. 7 Real-time classification results: (a) Class 1 There is no detector (b) Class 2 led indicator fault (c) Class 3 stain 1.fault (d) Class 3 stain 2.fault (e) Class 3 stain 3.fault (f) Class 4 there is no fault (g) Detection of a different error that does not fall

In the image in Figure 7, a specific error that is not used in the training of the model is expected to converge as an error by the model. The model decided that this specific error had a error of 0.97. In the image in Figure 7, a certain error that was not used in the training of the model is expected to be approximated as an error by the model. Thus, the model will be resistant to a newly occurring error that did not exist before. The need to update the dataset with each new error will be eliminated.

Since a machine learning model usually learns through a specific training set, it creates a general structure based on the data it encounters during this training process. The model learns patterns, relationships, and rules in the data and develops a function to apply to future data. However, when confronted with data that is not used or underrepresented in the training of the model, it may have difficulty identifying that data.

The fact that the model identifies an error that it did not encounter during training and evaluates it as an error 97% of the time points to several important aspects:

1. Divergence of Data Distribution: The model may have struggled to recognize this specific error that is underrepresented in the training set or has a different distribution. This shows the limits of the model's ability to generalize. Compared to the training data, this new state is a far cry from the model's previous learning processes.
2. Error Types and Incorrect Decision Making: The model's 97% evaluation of a particular error as an "error" indicates that the model may have created a false generalization or a wrong conceptual framework that it learned about that situation. A model that is not subject to such errors during training, when faced with this new situation, may reach an erroneous conclusion based on the available data to make a specific decision.
3. Model Overfitting Problem: A model that has achieved a high accuracy on the training data can sometimes lose its ability to generalize by overfitting the training data. In this type of situation, the model's ability to identify errors when it encounters new data is reduced. This indicates that the model only works well on training data, but may fail when it encounters real-world data.

As a result, the model's evaluation of an unused error type during training with a high accuracy rate of 97% highlights its current limitations and areas for improvement. This outcome suggests that while the model performs well in certain scenarios, it may still be prone to misclassifications or overfitting, especially when faced with unseen data. Addressing such weaknesses requires refining the training process, possibly by incorporating more diverse data or employing advanced techniques such as regularization to improve generalization. Ultimately, this insight serves as a valuable guide for enhancing model performance in future iterations and ensuring more robust real-world applications.

IV. DISCUSSION

This section discusses the superiority of the proposed method over existing methods. Traditional optimization methods often run into local optimum problems. This means that the algorithm only comes up with a "good enough" solution, rather than finding the best possible outcome. For example, in the process of training, some combinations of

hyperparameters can get stuck at local optimums. This negatively affects the overall performance of the model and leads to loss of time.

Another problem is population diversity. In particular, methods such as evolutionary algorithms or particle swarm optimization are dependent on initial population diversity. If the initial population is selected homogeneously, this may lead to a narrowing of the research area and a lack of exploration of different solution areas. This can increase computational costs while also limiting the model's ability to learn.

As a result, it is not preferred to find optimum set points with manual hyperparameter adjustments and to calculate well, it will never come close to perfection.

The proposed PSO-based Faster R-CNN model overcomes these limitations. Performing hyperparameter optimization with PSO allows a more optimal model to be created. However, the shortcoming of this model is the inadequacy of the dataset. Diversity can be achieved by using the Generative Adversarial Network (GAN) model to develop the dataset. GAN is a type of deep learning model proposed by Ian Goodfellow and colleagues in 2014. GANs are based on the principle that two neural network models, a generator and a discriminator, work in competition with each other. These two models try to produce more realistic data by improving each other within the framework of a game theory [18].

V. CONCLUSION

In this study, a CNN-based model optimized with Particle Swarm Optimization (PSO), referred to as PSO-Improve-CNN, is proposed for intelligent fault detection in quality control processes. The strong global search capability of PSO in non-linear problems is utilized to automatically optimize the hyperparameters of the Faster R-CNN model, enhancing fault diagnosis accuracy. This enhanced model provides a more effective and automated fault detection in quality control processes. The significant findings of this study are as follows:

Fault Detection and Classification Performance: The Faster R-CNN model optimized with PSO successfully classified detector faults. The model was trained using 32x32 grayscale images and demonstrated high performance in metrics such as accuracy, precision, recall, and F1 score. The optimized model improved validation and test accuracy, along with improvements in error metrics such as false alarm rate (FAR) and false negative rate (FNR).

Improvement of the Learning Process: The use of callback functions like Early Stopping and ReduceLROnPlateau prevented unnecessary epochs during the training process, reducing the risk of overfitting. ROC and PR curves indicate high classification performance of the model. Hyperparameters optimized by PSO enhanced overall model performance while also improving processing time.

Elimination of Uncertainty in Hyperparameter Optimization: The model's hyperparameters were optimized using PSO, bypassing the need for manual adjustment and eliminating associated uncertainties. This allowed the model to find the optimal hyperparameters and complete the training without human intervention.

Dynamic Structure and General Applicability: The model was trained with four classes: there is no detector, led indicator fault, scratch stain fault, and a there is no fault. The model exhibited resistance to different stain faults that may arise during production, accurately classifying them and demonstrating a dynamic structure.

In conclusion, this study provides a solid foundation for the automatic optimization of deep learning model hyperparameters and intelligent fault detection in quality control processes. In the era of big data, where manual inspection is limited, the proposed model offers a fast, precise, and adaptable solution, making a significant contribution to quality control processes.

REFERENCES

[1] Javaid, M., Haleem, A., Singh, R. P., Suman, R., & Gonzalez, E. S. (2022). Understanding the adoption of Industry 4.0 technologies in improving environmental sustainability. Sustainable Operations and Computers, 3, 203–217. http://dx.doi.org/ 10.1016/j.susoc.2022.01.008.

[2] Guo, D., Ling, S., Rong, Y., & Huang, G. Q. (2022). Towards synchronizationoriented manufacturing planning and control for industry 4.0 and beyond. IFAC-PapersOnLine, 55(2), 163–168. http://dx.doi.org/10.1016/j.ifacol.2022.04. 187.

[3] Ahmad, H. M., & Rahimi, A. (2022). Deep learning methods for object detection in smart manufacturing: A survey. Journal of Manufacturing Systems, 64, 181–196. http://dx.doi.org/10.1016/j.jmsy.2022.06.011.

[4] Erden E, Demir İE, Kökçam AH. "Hiper Parametre Optimizasyonu ile Makine Kaldırma Modeli Performansının Artırılması", 2023

[5] M. Choraś ve M. Pawlicki, "Optimize edilmiş yapay sinir ağına dayalı hata tespit yaklaşımı", Neurocomputing,vol. 452, pp.705-715, Sep., 2021

[6] Galindo-Salcedo, M., Pertúz-Moreno, A., Guzmán-Castillo, S., Gómez-Charris, Y. ve Romero-Conrado, A. R. (2022). Denetim ve kalite güvence süreçleri için akıllı üretim uygulamaları. Procedia Bilgisayar Bilimi, 198, 536–541. http: dx.doi.org/10.1016/j.procs.2021.12.282.

[7] Chen, M., Yu, L., Zhi, C., Güneş, R., Zhu, S., Gao, Z., Ke, Z., Zhu, M., & Zhang, Y. (2022). Genetik algoritma optimizasyonuna sahip gabor filtresine dayalı kumaş kusur tespiti için geliştirilmiş daha hızlı R-CNN. Endüstride Bilgisayarlar, 134, Madde 103551. http://dx.doi.org/10.1016/j.compind.2021.103551.

[8] Rožanec, J. M., Zajec, P., Trajkova, E., Šircelj, B., Brecelj, B., Novalija, I., Dam, P., Fortuna, B., & Mladenić, D. (2022). Endüstri 4.0* için kapsamlı bir görsel kalite kontrolüne doğru. IFAC-PapersOnLine, 55(10), 690–695. http://dx.doi. org/10.1016/j.ifacol.2022.09.486.

[9] Lu, L., Hou, J., Yuan, S., Yao, X., Li, Y. ve Zhu, J. (2023). Sürekli elyaf takviyeli polimer kompozitlerin eklemeli üretimi için derin öğrenme destekli gerçek zamanlı kusur tespiti ve kapalı döngü ayarı. Robotik ve Bilgisayarla Bütünleşik İmalat, 79, Madde 102431. http://dx.doi.org/10.1016/j.rcim.2022.102431.

[10] Li, W., Zhang, H., Wang, G., Xiong, G., Zhao, M., Li, G. ve Li, R. (2023). Tel ve ark eklemeli imalat için derin öğrenme tabanlı çevrimiçi metalik yüzey kusur tespit yöntemi. Robotik ve Bilgisayarla Bütünleşik İmalat, 80, Madde 102470. http://dx.doi.org/10.1016/j.rcim.2022.102470.

[11] Zhang, Y. ve Zhao, Y. F. (2022). Lazer toz yatağı füzyon süreçlerinin görsel kusurlarının üretilebilirliğini tahmin etmek için hibrit seyrek evrişimli sinir ağları. Üretim Sistemleri Dergisi, 62, 835–845. http://dx.doi.org/10.1016/j.jmsy. 2021.07.002. 2021.

[12] Kunang, Y.N., Nurmaini, S., Stiawan, D., & Suprapto, B.Y. (2021). Derin öğrenme ve hiper parametre optimizasyonu kullanarak bir hata tespit sisteminin hata sınıflandırması. Bilgi Güvenliği ve Uygulamaları Dergisi, 58, Makale 102804. http://dx.doi.org/10.1016/j.jisa.2021.102804.

[13] J. Gu, Z. Wang, J. Kuen, L. Ma,, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, T. Chen, "Evrişimli sinir ağlarında son gelişmeler", Örüntü Tanıma, cilt.77, ss.354-377, Mayıs, 2018.

[14] Y. Shi, "Parçacık sürüsü optimizasyonu: gelişmeler, uygulamalar ve kaynaklar", 2001 evrimsel hesaplama kongresi bildirileri, 2001, s. 81-86).

[15] Z. Vujović, "Sınıflandırma modeli değerlendirme metrikleri", Uluslararası İleri Bilgisayar Bilimi ve Uygulamaları Dergisi, cilt 12(6), s. 599-606, 2021.

[16] J. Jose ve D. V. Jose, "CIC-IDS 2017 veri setini kullanarak nesnelerin internetinde hata tespit sistemleri için derin öğrenme algoritmaları", Uluslararası Elektrik ve Bilgisayar Mühendisliği Dergisi (IJECE), cilt 13(1), s. 1134-1141, 2023.

[17] A. Yulianto, P. Sukarno ve N. A. Suwastika, "CIC IDS 2017 veri setinde AdaBoost tabanlı hata tespit sistemi (IDS) performansının iyileştirilmesi,", Journal of Physics Conference Series, 2019.
[18] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Derin öğrenme. MIT Press, http: www.deeplearningbook.org.
[19] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.
[20] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). R-CNN: Rich feature hierarchies for accurate object detection and semantic segmentation. Proceedings of the IEEE International Conference on Computer Vision (ICCV).
[21] Girshick, R. (2015). Fast R-CNN. Proceedings of the IEEE International Conference on Computer Vision (ICCV).