NÖHÜ Müh. Bilim. Derg. / NOHU J. Eng. Sci., 2025; 14(2), 649-658

Niğde Ömer Halisdemir Üniversitesi Mühendislik Bilimleri Dergisi
Niğde Ömer Halisdemir University Journal of Engineering Sciences

Araştırma makalesi / Research article

www.dergipark.org.tr/tr/pub/ngumuh / www.dergipark.org.tr/en/pub/ngumuh

Deep learning on the production line: A novel lightweight CNN model approach for efficient and fast defect detection

Üretim hattında derin öğrenme: verimli ve hızlı kusur tespiti için yeni bir hafif CNN modeli yaklaşımı

Hakan Tatar[1,*], Muhammed Furkan Küçük[2]

[1] *Fırat University, Academy Of Civil Aviation, Department of Aircraft Electrics and Electronics, 23200, Elazığ, Türkiye*
[2] *Fırat University, Electrıcal and Electronıcs Engıneerıng Department, 23200, Elazığ, Türkiye*

Abstract

This paper presents an optimized lightweight CNN model developed using a unique dataset introduced here for the first time to detect defects in manufacturing processes in a factory. The model performance was analyzed comparatively with widely used large-scale deep learning architectures such as VGG16 and ResNet50. All models were trained on the same original dataset, followed by the same approach in tuning hyperparameters such as learning rate, optimization algorithm, and data augmentation strategies. Performance analyses were conducted using fundamental metrics such as accuracy, precision, and F1 score, along with confusion matrices and randomly selected test images. Our proposed model attained high accuracy while reducing computational cost and significantly shortening training time compared to traditional architectures. The results demonstrate that the proposed CNN model achieves a competitive level of accuracy comparable to large-scale deep learning models while serving as a more suitable alternative for low-power hardware systems.

Keywords: CNN, Fault detection, Solar panel, By-Pass Diode, Classification methods

Öz

Bu makale, bir fabrikadaki üretim süreçlerinde ortaya çıkan kusurların tespiti için burada ilk kez sunulan benzersiz bir veri kümesi kullanılarak geliştirilen, optimize edilmiş hafif bir CNN modelini tanıtmaktadır. Model performansı, VGG16 ve ResNet50 gibi yaygın kullanılan büyük ölçekli derin öğrenme mimarileriyle karşılaştırmalı olarak analiz edilmiştir. Tüm modeller, aynı özgün veri kümesi üzerinde, öğrenme oranı, optimizasyon algoritması ve veri artırma stratejileri gibi sabit hiperparametrelerle eğitilmiştir. Performans analizleri doğruluk, kesinlik ve F1 skoru gibi temel metriklerin yanı sıra, karmaşıklık matrisleri ve rastgele test görüntüleri üzerinden gerçekleştirilmiştir. Önerdiğimiz model, geleneksel mimarilere kıyasla daha düşük hesaplama maliyeti ve çok daha kısa eğitim süresi ile yüksek doğruluk elde etmiştir. Elde edilen sonuçlar, önerilen CNN modelinin büyük ölçekli derin öğrenme modelleriyle rekabet edebilecek düzeyde doğruluk sunarken, düşük güçlü donanıma sahip sistemler için daha uygun bir alternatif olduğunu göstermektedir.

Anahtar kelimeler: CNN, Hata tespiti, Güneş paneli, Baypas diyotu, Sınıflandırma metotları

1 Introduction

Today, image-based defect and error detection systems provide great advantages in terms of both time and cost by making great contributions to the automation of quality control systems in industrial production [1]. Traditional methods are used in product defect detection today, and these systems progress based on human observations. Therefore, human-dependent systems have disadvantages such as fatigue, carelessness and variables caused by human factors [2]. Contrary to these disadvantages, deep learning-based models can be trained on large datasets and detect error outputs faster and with higher success rates; thus, they can help to significantly increase efficiency in a production line [3].

In recent years, numerous studies have been conducted to detect output defects during production stages using various deep learning-based architectures, particularly convolutional neural networks (CNNs) [4]. For instance, an examination of the study conducted by Elmas and Korkmaz reveals that a deep learning-based model was developed for detecting socket cable defects, achieving a test accuracy of 97.25% [5]. Similarly, in the study of Tan et al., applications for object detection and tracking utilizing deep learning algorithms were examined and incorporated into the literature [6]. In another study, Yıldırım et al. employed image processing and deep learning algorithms to classify assembly parts used in a manufacturing company, achieving a highly accurate automation system for assembly processes [7]. A separate study by Lei and Sui developed a Faster R-CNN-based model for detecting faults in high-voltage electrical transmission lines, accurately identifying insulator breaks and the presence of bird nests using the ResNet-101 architecture [8].

In addition, literature studies have implemented original model designs and conducted comparative analyses with traditional deep learning models. For example, the

* Sorumlu yazar / Corresponding author, e-posta / e-mail: htatar@firat.edu.tr (H. Tatar)
Geliş / Received: 17.02.2025 Kabul / Accepted: 13.03.2025 Yayımlanma / Published: 15.04.2025
doi: 10.28948/ngumuh.1641247

DEA_RetineNet model proposed by Cheng and Yu employs a channel attention mechanism and adaptive spatial feature fusion methods for detecting defects on the surface of steel parts. According to the comparative analysis, this model increased the mAP value to 78.25% and provided a performance improvement of 2.92% compared to the traditional RetinaNet model [9]. Another investigation by He et al. proposed a CNN-based model for detecting steel surface defects. The model aimed to determine defect locations on surfaces by combining a multi-level feature fusion network (MFN) with feature maps produced by the CNN. Subsequently, regions of interest were identified using a Region Proposal Network (RPN), and a model incorporating both a classifier and a bounding box regression module generated the final detection results. The model was evaluated on the NEU-DET dataset, achieving mAP accuracies of 74.8% and 82.3% with 300 proposals using ResNet34 and ResNet50, respectively. Furthermore, it was noted that the model reached 92% of its performance while operating at a speed of 20 ft/s on a single GPU with only 50 proposals. Consequently, the researchers concluded that the proposed method is suitable for real-time defect detection [10].

Based on the findings obtained from the literature, both analytical studies and original deep learning-based model designs addressing the problem are frequently integrated into industrial automation. It is anticipated that the requirements of systems intended to restore production to its most efficient state will be fulfilled by deep learning models.

A review of recent studies reveals a notable emphasis on real-time defect detection applications in industrial production processes. A study by Özcan et al. utilised machine learning and image processing methodologies to identify contamination and deformations on the surfaces of LPG cylinders[11]. In a similar vein, Ozan and Ceylan have proposed a Raspberry Pi-based image processing system for the detection and classification of defective eggs in egg production facilities[12].

A literature review on studies emphasizing the significance of datasets and data quality reveals that class imbalance and data scarcity pose considerable challenges. These challenges hinder the generalization capability of the model, thereby leading to errors in defect detection [13]. An imbalance in datasets directly affects the model's reliability, leading to critical issues such as misclassification and disruptions in the production process [14].

In this context, the study aimed to classify output products labeled as either faulty or intact at a factory producing bypass diodes for solar panels in the Elazığ Organized Industrial Zone, and an original dataset was created by collecting data directly from the production site. Concurrently, data augmentation techniques were applied to prepare this data for the model, thereby enhancing its suitability for diverse inputs and improving its performance. Finally, a unique, lightweight, and efficient CNN-based defect detection model was implemented to detect and classify physical defects in both intermediate and final products within the production process.

This paper makes four main contributions to the field, as summarized below:

- We develop a model designed to rapidly achieve high accuracy in detecting faulty intermediate and final products while minimizing the misclassification of non-defective items. Our proposed model achieves an overall accuracy of 94.2% in defect detection and classification.
- We evaluate the effectiveness of the newly designed model by comparing it against traditional architectures. These comparative tests provide an understanding of the strengths and limitations of our approach, emphasizing the specific areas where the proposed model demonstrates superiority.
- We conduct performance comparisons in later stages by benchmarking our model against pre-trained deep learning architectures, including ResNet50 and VGG16. This evaluation ensures an objective analysis of its performance relative to well-established models.
- Finally, we assess the models using a consistent evaluation framework, where all models, including pre-trained architectures, are trained on the same custom dataset. The results are analyzed using key performance metrics such as accuracy, loss, and training time, providing a comprehensive understanding of the trade-offs between model complexity and efficiency.

The rest of the paper is organized as follows.

Section 2 provides key methodological and analytical components in a systematic manner. This section includes a comprehensive discussion of the dataset characteristics, model architecture, training strategies, and computational methodologies employed to evaluate success criteria. Section 3 rigorously assesses the proposed model's performance through both quantitative and qualitative analyses, accompanied by a comparative evaluation against conventional approaches to ensure a critical discussion on the significance of the results. Finally, the paper concludes with the findings presented in Section 4.

2 Materials and methods

In this study, the aim was to systematically process 5751 high-resolution diode images labeled as A, B, and C types, which were originally generated for the first time in this study, to develop a deep learning-based classification model for detecting final and intermediate product defects at an industrial production site manufacturing bypass diodes for solar panels. These images consist of samples collected periodically from the production line that reflect various production defects.

2.1 Creating a data set

In our study, we utilized images collected from the factory operating in the Elazığ Organized Industrial Zone in Elazığ province to train our model. These images depict junction boxes, which are key components of solar energy systems, and within these boxes, bypass diodes are installed.

The components referred to as junction boxes are categorized into three types A, B, and C. Bypass diodes are installed inside the junction boxes and then secured by

riveting. During this process, visually detectable errors occur in these products both before and after the riveting procedure.

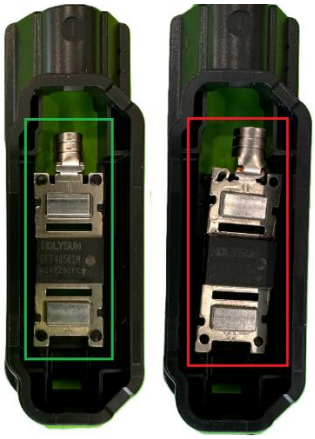

Figure 1. Type A, B and C empty junction box

The dataset we created to detect these errors, as provided by the manufacturer, contains a total of 5751 images. In our dataset, 3245 product images are labeled "intact" and 2506 are labeled "faulty."

Table 1. Example of "faulty" status

Types of Error
Diode that is not suitable for the body
Reverse riveting of the diode to the body
Wrong body selection.
By-pass diode not fully seated on rivets

Table 1 lists the reasons for "faulty" situations. For example, Figure 1 shows an example of a bypass diode that is not fully seated on the rivets.

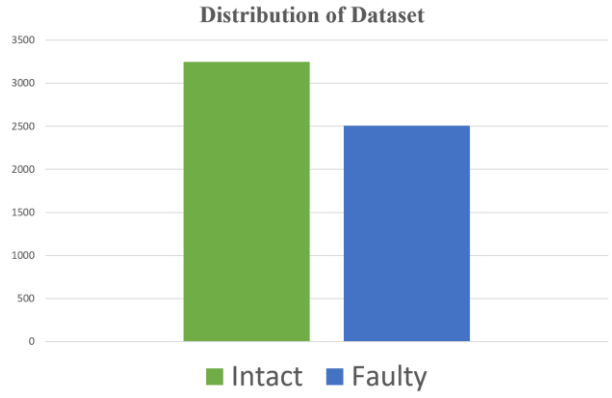

Figure 2. Distribution rate of dataset

Figure 2 explains the statistical distribution of the data set.

In the process of creating a dataset, it is essential to remember that data quality directly affects model performance. In this regard, careful attention was paid to ensuring high image quality, with photographs captured from various angles, under different lighting conditions, and at varying distances. Figure 3 illustrates examples of junction boxes containing diodes.

Figure 3. Type A, B and C with By-Pass diode junction box

2.2 Labeling the dataset

Our dataset comprises images of products classified as either "intact" or "faulty," with the latter exhibiting physical defects. This dataset was collected directly from the production site by the researchers, and no public dataset was utilized. The labeling process was conducted by the researchers as well. Consequently, the objective was to train and test the developed model to address challenges encountered in a real production environment.

2.3 Data pre-processing

In the Microsoft Visual Studio Code environment where the research was conducted, the Python programming language was utilized along with the TensorFlow and Keras libraries. Additionally, the OpenCV (cv2) library was employed for visual processing and preliminary preparation steps. In the initial stage of model training, the 5751 images created by the researchers were consolidated into a single dataset and organized into two folders, labeled as "faulty" and "intact" according to their classifications. Subsequently, the resolution of the images was standardized to 55×55 pixels for consistency and normalization, and the pixel values were rescaled to the range [0, 1] to make them suitable for the model. Prior to training, the dataset was partitioned into training (80%) and test (20%) sets, with the parameter "random_state=42" selected to prevent overfitting.

2.4 Data augmentation techniques

Data augmentation techniques were employed to utilize the dataset more comprehensively during training and to enhance the generalization ability of the developed model. The Keras class, ImageDataGenerator, is widely used for both data augmentation and preprocessing in the training of deep learning models. During the augmentation process, various transformations were applied to all training data using ImageDataGenerator to increase its diversity and generalizability. Table 2 shows data augmentation techniques.

Table 2. Data augmentation methods

Data Augmentation Techniques	Parameter
Random Rotation -20, +20	rotation_range = 20
Shifting in Horizontal and Vertical Directions	width_shift_range = 0.2, height_shift_range = 0.2
Shear or Shift	shear_range = 0.2
Zoom Range	zoom_range = 0.2
Horizontal Flip	True
Fill Mode	nearest

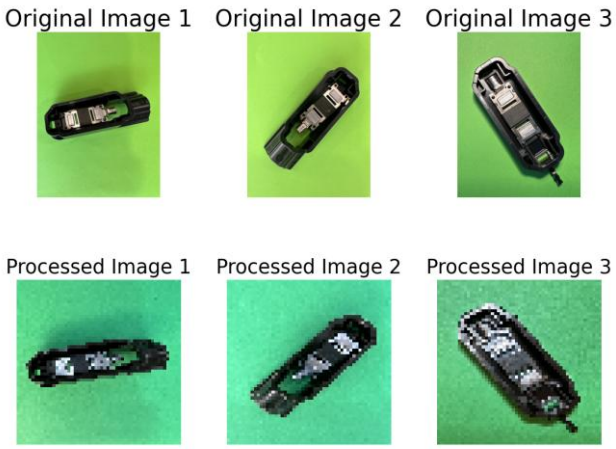

Figure 4. Examples of augmented images

Employing data augmentation techniques ensures that the model is not limited solely to the data it encounters directly in the dataset and enhances its generalization ability by learning how defects appear in different sizes, angles, and positions. Figure 4 shows the intermediate images after these data augmentation techniques are applied, demonstrating the transformations introduced to the dataset.

Convolutional Neural Networks exhibit high performance, particularly in image processing and object recognition. In the study by Şeker et al., it was stated that CNNs yielded superior results compared to other deep learning methods in image processing [15]. The feature extraction and classification capabilities of Convolutional Neural Networks are frequently employed in data analysis. For example, in the work of Oyucu and Herdem, the performance of the CNN-LSTM model in capturing long-term dependencies and complex features was emphasized [16].

In our study, Convolutional Neural Networks (CNNs) were employed because they can accurately recognize complex features in visual data.

2.5 Proposed CNN architecture

Our model architecture, illustrated in Figure 5, is constructed in a multi-layered manner based on the Convolutional Neural Network (CNN) model. The first convolutional layer employs 32 filters of size 3×3, while the subsequent convolutional layer utilizes 64 filters of the same size to extract higher-quality information.

After each convolutional layer, a MaxPooling layer is applied to perform spatial downsampling and reduce the model's size. Additionally, dropout layers, used in conjunction with pooling layers, temporarily disable certain neurons to prevent memorization issues.

After the convolutional layers, the two-dimensional data is flattened into one-dimensional vectors using a flattening layer. These vectors are then fed into a Dense layer with 128 neurons, which employs a ReLU activation function along with L2 regularization. Subsequently, a dropout layer with a rate of 0.5 is applied to enhance regularization. Finally, the output is passed through a Sigmoid activation function to address the binary classification problem of “faulty” and “intact.”.

In our study, the Binary Crossentropy loss function, which is frequently employed in two-class classification problems, was used. The purpose of this function is to minimize the difference between the probability distribution predicted by the model and the actual class labels. The function is based on the principle that the model's predictions for the correct class should be as high as possible while those for the incorrect class should be close to zero by evaluating the probability that each example belongs to the correct class in binary scenarios. In this way, it aims to improve the model's ability to distinguish between classes and enhance its generalization capability.

During the compilation phase, the Adam optimizer was employed for model optimization, and the "ReduceLROnPlateau" callback function was utilized to automatically adjust the learning rate based on the validation loss performance. This approach enabled continuous parameter updates and prevented unnecessary computational expenditure by avoiding an excessively high training rate.

To enhance the continuity of the model and improve the reliability of the output, the training process was conducted through five independent trials, with each trial consisting of 300 epochs. This approach allowed for a comparison of the effects of varying initial weight configurations and random conditions on the results. The statistical metrics obtained at the conclusion of each training session were recorded, and their averages were calculated. Additionally, the test data accuracy at the end of each trial was measured and documented separately. Consequently, detailed insights into the temporal dependencies of the model's performance, the learning process trajectory, and the associated performance statistics were obtained.

Our proposed CNN architecture is designed to achieve an optimal balance between complexity and computational efficiency. Unlike deeper networks that require extensive computational resources, our model employs only two convolutional layers, making it lightweight and suitable for real-time or resource-limited applications.

The inclusion of dropout layers after each pooling layer is a deliberate choice to mitigate overfitting, ensuring the model generalizes well across different data distributions.

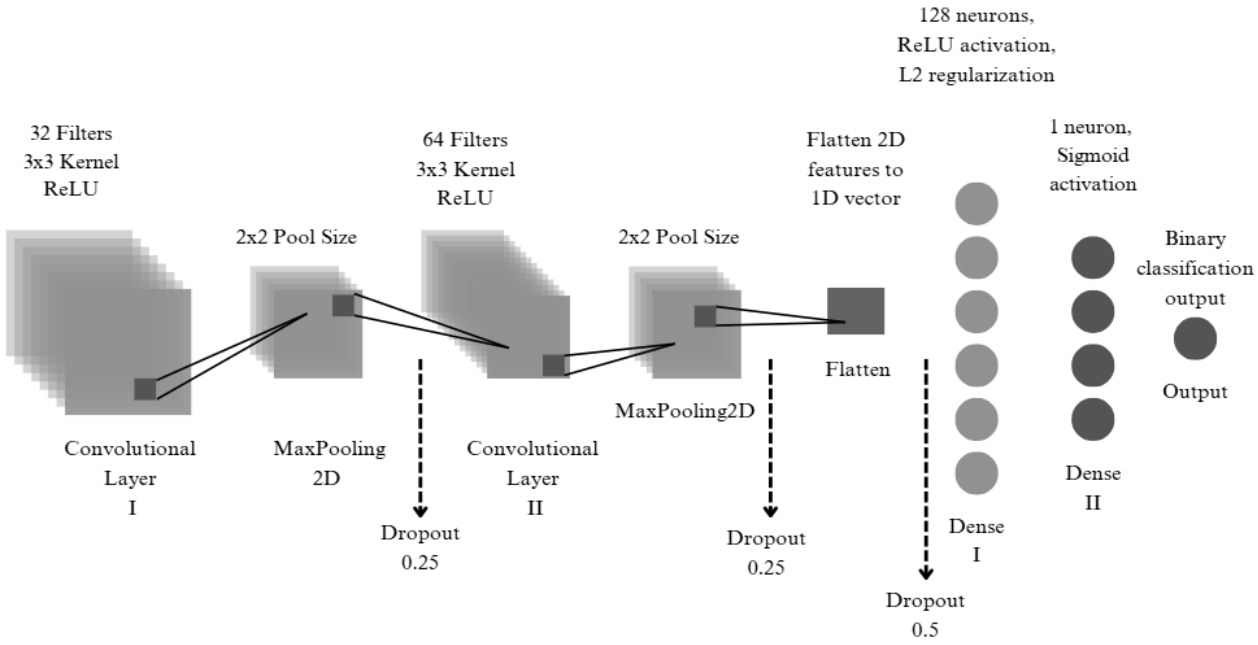

Figure 5. Proposed model with explanation

In the implemented CNN model, dropout layers with a rate of 0.25 are strategically placed after each max-pooling operation, reducing the risk of over-reliance on specific neurons and improving the model's ability to extract robust features from the input data. Additionally, a higher dropout rate of 0.5 is applied before the final dense layers to further enhance regularization, preventing co-adaptation of neurons and leading to better generalization.

The L2 regularization in the dense layer further prevents excessive weight magnitudes, leading to better stability in training. This regularization technique is particularly crucial in the fully connected layers, where a large number of parameters can increase the risk of overfitting. By penalizing large weight values, L2 regularization ensures a controlled optimization process, resulting in improved model robustness.

Hyperparameter tuning was performed systematically to optimize the model's performance. The initial learning rate was set to 0.001, a standart choice for the Adam optimizer. This adaptive adjustment helps prevent convergence to suboptimal solutions by reducing the learning rate when performance improvements stagnate.

The batch size was experimentally determined to be 16, as it provided a balance between training stability and computational efficiency. A smaller batch size allows for more frequent weight updates, leading to faster convergence, while still maintaining a stable training process. Data augmentation techniques, including rotation, width and height shifts, shear transformation, zoom, and horizontal flipping, were applied to the training data to artificially increase its diversity, further enhancing generalization.

The architecture was designed to maximize feature extraction efficiency while maintaining a compact model size, making it suitable for real-world applications requiring rapid inference. The use of two convolutional layers with 32 and 64 filters, respectively, ensures progressive feature extraction, capturing both low- and high-level patterns in the input images. The choice of 3×3 kernel size and ReLU activation function enables effective non-linear transformations, crucial for deep learning-based visual recognition tasks.

2.6 Other models

By comparing our proposed model with pre-trained large-scale models such as ResNet50 and VGG16, we examined the pursuit of a model that is particularly suited for embedded systems and mobile devices, considering the high parameter counts and processing power requirements.

Figure 6 highlights the visual that shows the mutual aspects of the proposed model with deep learning.

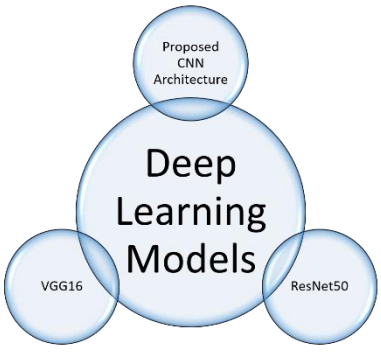

Figure 6. Used models on study

The ResNet50 architecture is a member of the ResNet family introduced by He et al. in 2015 for addressing the gradient vanishing and optimization challenges that arise with deeper neural networks, this architecture enhances gradient flow in deep layers by incorporating residual connections, thereby streamlining the training process. With

a total depth of 50 layers, ResNet50 has demonstrated superior performance on the ImageNet dataset and is widely employed in computer vision applications such as object detection and image classification [17].

The VGG16 model, developed by Simonyan and Zisserman, is a CNN-based model comprising 16 deep layers. A significant outcome of VGG16 is that increasing network depth can improve accuracy, and, in this context, the use of convolutional layers with smaller filters has proven to be effective [18].

2.7 Success parameters

In our study, various success parameters of the proposed model, such as accuracy, loss, sensitivity, precision, and F1 score, were analyzed, and the model's performance evaluation was explained in detail. Additionally, a complexity matrix analysis was performed to comprehensively assess the model's classification success after training. Moreover, to objectively analyze the model's real-world outputs, a probabilistic estimation of its output classification was conducted using randomly selected test images.

2.7.1 Confusion matrix

The complexity matrix is a critical analysis tool for evaluating the success of a machine learning model. This matrix compares the model's prediction results with the actual labels to determine which classifications are made correctly and to what extent. Figure 7 details the contents of the confusion matrix.

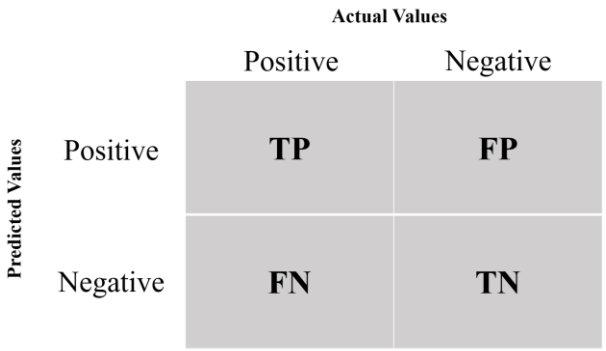

Figure 7. Confusion Matrix Interpretation

True Positive: Data that the model predicts as positive and is actually positive.

True Negative: Data that the model predicts as negative and is actually negative.

False Positive: Data that the model classifies as positive but is actually positive.

False Negative: Data that the model predicts as negative but is actually positive.

- Accuracy

Gives the total correct prediction rate of the model. In general, it is a metric that shows the total success of the model. It is expressed by Equation 1.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

- Recall

Essentially, it shows how accurately the data belonging to the positive class are predicted. It is expressed as in Equation 2.

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

- Precision

It is the metric that measures how accurate the positive predictions made by the model are. It is shown in Equation 3.

$$Precision = \frac{TP}{TP + FP} \quad (3)$$

- F1 Score

Precision alone is not a fully sufficient metric. For precision to be meaningful, it must be balanced with sensitivity. The F1-Score is used to measure the balance point of these two metrics. It is expressed as in Equation 4.

$$F1 = 2 \times \frac{Precision \times Recall}{Precison + Recall} \quad (4)$$

- Macro Average

It is the success parameter that calculates the average of the metrics of each class by taking them equally weighted. It is expressed as in Equation 5.

$$Macro\ Avg = \frac{Precision_1 + \cdots + Recall_N}{N} \quad (5)$$

N: Number of class

- Weighted Average

After calculating the metrics of each class, the average is obtained by weighting it with the total number of examples of the class. It is shown in Equation 6.

$$W\ Avg = \frac{(P_1 \times NE_1) + \cdots + (P_N x NE_N)}{Total\ Number\ of\ Samples} \quad (6)$$

P: Precision, NE: Number of examples, N: Number of class

3 Findings and discussions

The present study trained a CNN-based model on an uniqe dataset of 5751 images, classifying the products as either "faulty" or "intact.".

The study's results were analyzed using fundamental metrics, including the model's classification performance, recall, precision, and F1 score. In addition, classification errors were examined in detail with the assistance of a confusion matrix.

3.1 Classification performance

Table 3. Classification report

Classification Report				
	Precision	Recall	F1– Score	Support
Intact	0.99	0.90	0.95	3245
Faulty	0.89	0.99	0.94	2506
Accuracy	-	-	0.94	5751
Macro Avg	0.94	0.95	0.94	5751
Weighted Avg	0.95	0.94	0.94	5751

(The values are rounded to closest value.)

Table 3 presents the output metrics of the proposed model. The overall accuracy was determined to be 94%. In class-based evaluations, the precision for "intact" products was 99%, and the recall was 90%. These results indicate that the model is highly accurate in detecting "intact" products; however, in certain cases, products classified as "intact" were misidentified as "faulty," which is undesirable. The recall and precision values for "faulty" products were found to be 99% and 89%, respectively. Consequently, the model almost completely detects defective products correctly, rarely misclassifying defect-free products as faulty.

Class-based evaluation entails computing performance metrics (e.g., accuracy, precision, F1-score) separately for each class, enabling a detailed analysis of the model's performance across different categories. This approach is particularly crucial for imbalanced datasets, as it highlights class-specific variations. In contrast, overall evaluation provides a single aggregated measure of the model's effectiveness across all classes, offering a holistic assessment. This is achieved through macro, micro, or weighted averaging methods. However, overall evaluation may obscure disparities in class-wise performance. Therefore, to ensure a comprehensive assessment, both overall and class-based results are presented in detail in Table 3.

3.2 Confusion matrix

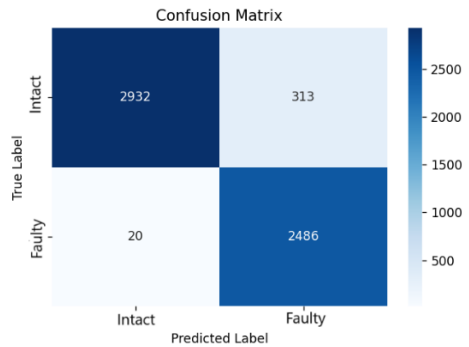

Figure 8. Confusion matrix result

The confusion matrix of the model is given in Figure 8, which enables the analysis of the samples taken by our model in incorrect classification. According to the confusion matrix obtained in our study:

- Approximately 90.3% of error-free products were predicted correctly.
- Approximately 99.2% of defective products were predicted correctly.
- 313 error-free labeled products were classified as faulty due to incorrect prediction.
- Only 20 of the faulty products were classified as faultless.

3.3 Validation on real images

Figure 9. Test on real images

The real world labeled outputs of the model are given in Figure 9. During the model design phase, outputs were also validated using randomly selected test images. An examination of these examples indicates that the model produces very high prediction probabilities and accurately classifies error-free products with a proportional accuracy ranging from 95% to 100%.

Based on the analysis of these images, it was observed that the proposed model can successfully distinguish even subtle differences in detail.

3.4 Comparison of models

In the field of deep learning, large-scale models trained using transfer learning techniques are frequently favored, particularly in object recognition and classification tasks. Models such as ResNet50 and VGG16, which achieve high performance in image-based tasks owing to their deep and expansive architectures, have been well established in the literature and are commonly employed in contemporary studies. However, the inherent disadvantages of large-scale models may render specialized, compact models more advantageous in certain contexts.

Within this scope, comparative analyses were conducted using traditional deep learning methods to contextualize the proposed model. The study results were presented in detail, outlining the characteristics of each model based on tests performed with the created dataset on traditional models.

3.4.1 *Proposed model vs ResNet50*

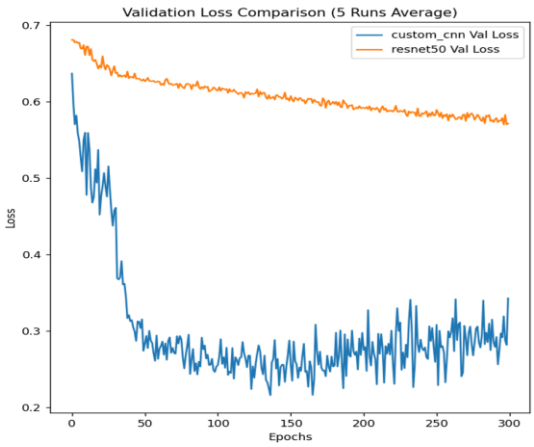

Figure 10. Proposed model and ResNet50 validation loss analysis

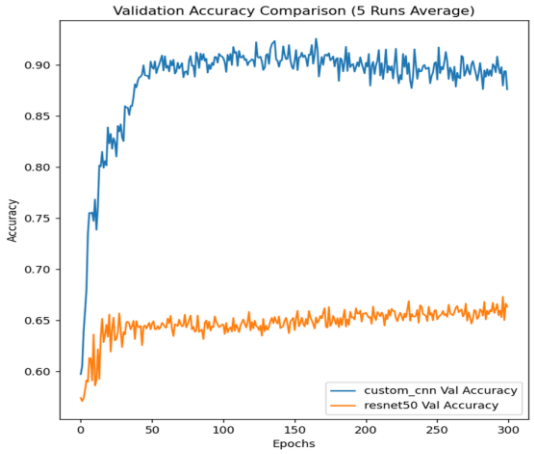

Figure 11. Proposed model and ResNet50 validation accuracy analysis

For the first application, Figure 10 and 11 show the comparative analysis of accuracy and loss values of the CNN architecture proposed in this study and the ResNet50 model. The information in the graph shows the average performance of the models after 300 epochs of training over five different iterations.

Upon examining of Figure 10, it is observed that the proposed model reduces the validation loss more rapidly during the initial epochs and approaches stability after approximately 100 epochs. Conversely, an analysis of the validation loss of the compared ResNet50 model reveals that its loss value decreases at a significantly slower rate and remains higher than that of the proposed model after approximately epoch 10 in each iteration. This indicates that the proposed model exhibits a faster learning capacity and achieves a lower validation loss rate.

An examination of the Figure 11 reveals that proposed model reached to 90% accuracy approximately in 30 epochs and largely maintained this level. In contrast, the compared ResNet50 model remained at 65-70% levels on our dataset and did not show significant improvement in the later stages of training. These results indicate that the specially designed CNN architecture generalizes much better than ResNet50 on our dataset and can successfully distinguish between target classes.

Considering another parameter like a training time our proposed CNN model completes training in 1260 seconds (21 minutes) per iteration on average, whereas the ResNet50 architecture requires 10,558 seconds (175 minutes). In CPU-based training, our model exhibits a significant advantage in updating parameters compared to ResNet50, which is hindered by extended parameter update durations and increased memory management overhead during mini-batch operations.

These results show that the transfer learning-based ResNet50 model cannot achieve optimal results on the uniqe dataset we created and that the CNN model we proposed has a much more suitable, lightweight, fast and reliable architectural design for certain tasks.

3.4.2 *Proposed model and VGG16*

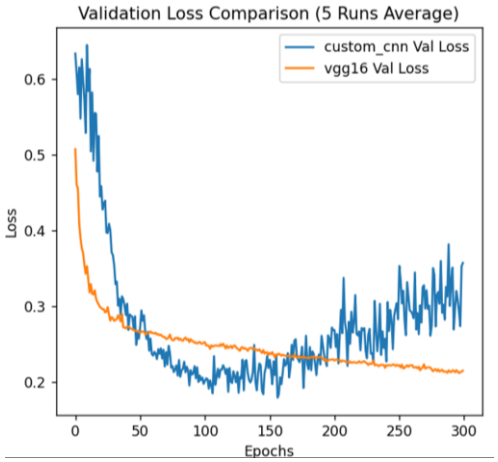

Figure 12. Proposed model and VGG16 validation loss analysis

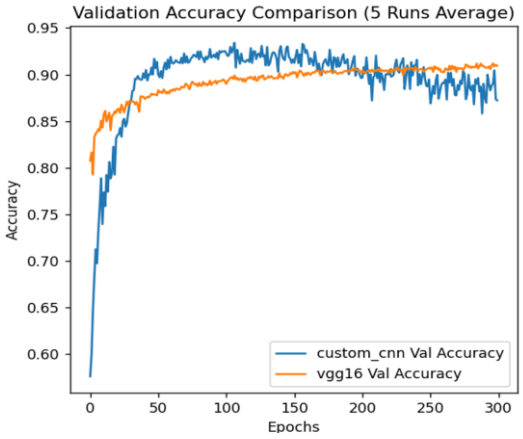

Figure 13. Proposed model and VGG16 validation accuracy analysis

In the second application, the performances of the proposed model and the VGG16 model, which is frequently used in the deep learning literature, were compared. In this

study phase, both models were trained five times with 300 epochs and the graphs in Figure 12 and 13 were created based on the average values.

An examination of the graphs reveals that VGG16 can stabilize the validation loss during training on the dataset up to a certain level, owing to its deep and complex structure. The VGG16 model demonstrates a balanced learning curve with low loss values, benefiting from the advantage of pre-trained weights. However, in terms of validation accuracy, our proposed model achieves nearly the same accuracy levels as VGG16. Moreover, although our proposed model is not pre-trained like VGG16, it has nonetheless attained a high accuracy rate and exhibited a stable, steadily improving learning process on average.

Compared to the proposed model, the primary disadvantage of VGG16 lies in the weight and computational cost associated with its deep structure. During training, VGG16 requires significant memory due to its large number of parameters, resulting in intensive CPU resource consumption in local training environments. During the training process on the same hardware as the proposed model, the high resource consumption of VGG16 becomes particularly apparent. While our proposed model completes one training iteration in approximately 450 seconds, the VGG16 model requires around 2560 seconds for the same process. Consequently, large-scale models such as VGG16 become less suitable for environments with hardware limitations, such as embedded systems and mobile devices. In contrast, our proposed model trains much faster due to its lighter and simpler structure, and it requires substantially less computational cost to achieve a comparable level of accuracy. These results support the evaluation of the proposed model as a more suitable alternative for practical applications.

Based on this comparative analysis, we concluded that traditional models pre-trained on large datasets, such as VGG16, may encounter limitations and exhibit disadvantages in flexibility when adapting to new datasets. In contrast, our proposed model demonstrates a more targeted and adaptable performance, as it is directly optimized for our originally created dataset.

The model proposed in the study is regarded as a strong alternative to VGG16 due to its shorter training time, lower parameter requirements, and computational efficiency. Although the literature demonstrates that VGG16 performs well on large-scale, general-purpose datasets, a lighter and more problem-oriented model may be preferable in scenarios that demand rapid and frequent optimization.

In future studies, the objective is to test the proposed model on more comprehensive datasets, integrate it into various production areas, and enable real-time object and defect detection. Specifically, detailed analyses of the model's performance metrics on these datasets are planned, along with efforts to enhance its generalization ability by testing it on diverse data. Additionally, hyperparameter optimization aimed at increasing the hardware efficiency of the proposed model—as well as research and implementation of techniques to minimize memory management and computational costs—will be considered key areas for further investigation.

Integration evaluation will be made in different systems in order to increase the portability and effectiveness of the model in practical use areas. In particular, parameter and hyperparameter optimization studies will continue to be carried out to further reduce the processing power requirements of our model.

3.5 Future studies

One of the main goals expected to be achieved as a result of future studies is to make this supported model operable in embedded systems, mobile devices and devices with low hardware power and to be used for real-time applications.

In the light of such an approach, it is expected that the proposed model will contribute to digital industrialization by increasing its share in industrial production sites, IoT devices and platforms with low power consumption.

At the same time, increasing the energy efficiency of the model, improving data processing times and accelerating detection capabilities will be the main focus of future research.

As a result, a comprehensive development process is carried out to increase the applicability potential of the developed model for academic and industrial use, and it is aimed to provide an efficient, low-cost and highly generalizable artificial intelligence solution that appeals to a wide range of uses.

4 Results

In this study, a novel convolutional neural network (CNN) model was developed to detect defects in intermediate and final products within a factory production line. The proposed model was trained using an original dataset, systematically processed and labeled from data collected at regular intervals from the production site. The model's performance was statistically evaluated based on key success metrics. Additionally, to ensure a comprehensive analysis, the proposed model was compared with widely used deep learning architectures, focusing on computational load, training time, and parameter optimization. To maintain objectivity in the comparative analysis, all models were trained using the same dataset, and hyperparameters such as data augmentation techniques, learning rate, and optimization algorithm were kept constant. The findings demonstrate that the proposed model not only achieves high accuracy and a low loss rate but also offers efficient processing capabilities with reduced computational costs. Furthermore, it represents a viable alternative for deployment on resource-constrained devices, outperforming large-scale deep learning architectures in terms of efficiency. The results indicate that the proposed model is a strong candidate for real-time defect detection in production processes.

Acknowledgement

This paper is produced from the part of Hakan TATAR's Master Thesis. In addition, we would like to express our sincere gratitude to all the management of Hatko - Hatsun Elazig PV-JB Production Factory for their valuable

assistance in providing the necessary facilities and permissions to create the dataset used in this study.

Conflict of Interest

The authors declare that they have no conflict of interest.

Similarity Rate (iThenticate): %13

References

[1] I. D. Apostolopoulos and M. Tzani, Industrial object, machine part and defect recognition towards fully automated industrial monitoring employing deep learning. The case of multilevel VGG19, arXiv preprint arXiv:2011.11305, 2020. https://doi.org/10.48550/arXiv.2011.11305

[2] D. Ever and E. N. Demircioğlu, Yapay zekâ teknolojilerinin kalite maliyetleri üzerine etkisi, Çukurova Üniversitesi Sosyal Bilimler Enstitüsü Dergisi, 31 (1), 59-72, 2022. https://doi.org/10.35379/cusosbil.1023004

[3] E. Oğuzay and M. Balta, Rulman titreşim verilerinden derin öğrenme tabanlı arıza tespiti, Karadeniz Fen Bilimleri Dergisi, 14 (3), 1159-1175, 2024. https://doi.org/10.31466/kfbd.1434595

[4] E. Akın and M. E. Şahin, Derin öğrenme ve yapay sinir ağı modelleri üzerine bir inceleme, EMO Bilimsel Dergi, 14 (1), 27-38, 2024.

[5] B. Elmas and H. Korkmaz, Derin öğrenme ile soket kablo sıralama hata tespiti, Politeknik Dergisi, 1–1, 2025. (Early Access). https://doi.org/10.2339/politeknik.1500454

[6] F. G. Tan, A. S. Yüksel, E. Aydemir, and M. Ersoy, Derin öğrenme teknikleri ile nesne tespiti ve takibi üzerine bir inceleme, Avrupa Bilim Ve Teknoloji Dergisi, 25, 159-171, 2021. https://doi.org/10.31590/ejosat.878552

[7] B. Yıldırım and G. Cagıl, Bir Montaj Parçasının Derin Öğrenme ve Görüntü İşleme ile Tespiti, Journal of Intelligent Systems: Theory and Applications, 3 (2), 31-37, 2020. https://doi.org/10.38016/jista.710144

[8] X. Lei and Z. Sui, Intelligent fault detection of high voltage line based on the Faster R-CNN, Measurement, 138, 379-385, 2019. https://doi.org/10.1016/j.measurement.2019.01.072

[9] X. Cheng and J. Yu, RetinaNet With Difference Channel Attention and Adaptively Spatial Feature Fusion for Steel Surface Defect Detection, IEEE Transactions on Instrumentation and Measurement, 70, 1-11, 2021. https://doi.org/10.1109/TIM.2020.3040485.

[10] Y. He, K. Song, Q. Meng, and Y. Yan, An End-to-End Steel Surface Defect Detection Approach via Fusing Multiple Hierarchical Features, IEEE Transactions on Instrumentation and Measurement, 69 (4), 1493-1504, 2020. https://doi.org/10.1109/TIM.2019.2915404

[11] H. Özcan, H. T. Gençtürk, G. Genç, T. E. Yıldırım, F. Durmuş, and A. Gürleyen, Gerçek zamanlı kusur tespiti: LPG tüplerinin yüzeylerinde kirlilikleri tanımlama için görüntü işleme ve makine öğrenimi teknikleri ile yenilikçi bir yaklaşım, Afyon Kocatepe Üniversitesi Fen ve Mühendislik Bilimleri Dergisi, 24 (2), 330-340, 2024. https://doi.org/10.35414/akufemubid.1364153

[12] M. Ozan and M. Ceylan, Endüstriyel Üretim Tesislerinde Yumurtaların Görsel Analizi Ve Sınıflandırılması İçin Raspberry Pi Tabanlı Gerçek Zamanlı Bir Uygulama, SETSCI Conference Proceedings, 3, pp. 727-731, Samsun, Turkey, 2018.

[13] M. Buda, A. Maki, and M. A. Mazurowski, A systematic study of the class imbalance problem in convolutional neural networks, Neural Networks, 106, 249-259, 2018. https://doi.org/10.1016/j.neunet.2018.07.011

[14] R. van den Goorbergh, M. van Smeden, D. Timmerman, and B. Van Calster, The harm of class imbalance corrections for risk prediction models: illustration and simulation using logistic regression, Journal of the American Medical Informatics Association, 29 (9), 1525-1534, 2022. https://doi.org/10.1093/jamia/ocac093

[15] A. Şeker, B. Diri, and H. H. Balık, A Review about deep learning methods and applications, Gazi Journal of Engineering Sciences, 3 (3), 47-64, 2017.

[16] S. Oyucu and M. S. Herdem, Hibrit derin öğrenme algoritmaları kullanılarak biyogaz reform süreçlerinin optimizasyonu: cnn-lstm modeli ile çıktı parametrelerinin tahmini, Adıyaman Üniversitesi Mühendislik Bilimleri Dergisi, 11 (23), 301-316, 2024. https://doi.org/10.54365/adyumbd.1488710

[17] K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, pp. 770-778, 2016. https://doi.org/10.1109/CVPR.2016.90.

[18] K. Simonyan and A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv preprint arXiv:1409.1556, 2015. https://doi.org/10.48550/arXiv.1409.1556