Green Intelligent Systems and Applications

Research Article
Volume 5(1), 2025, 93–105
https://doi.org/10.53623/gisa.v5i1.581

Classification of Metal Surface Defects Using Convolutional Neural Networks (CNN)

Dhika Wahyu Pratama[1], Muchammad Ismail[1], Restu Nurraudah[1], Achmad Pratama Rifai[1]*, Nguyen Huu Tho[2]

[1]Department of Mechanical and Industrial Engineering, Universitas Gadjah Mada, Yogyakarta, Indonesia
[2]Faculty of Engineering and Technology, Nguyen Tat Thanh University, Ho Chi Minh City, Vietnam

*Correspondence: achmad.p.rifai@ugm.ac.id

SUBMITTED: 7 January 2025; REVISED: 13 March 2025; ACCEPTED: 14 April 2025

ABSTRACT: Metal surface quality inspection is an important step in ensuring that products meet predetermined industry standards. The manual methods used were often slow and prone to errors, so more efficient solutions were needed. The application of Machine Learning (ML)-based technologies, especially Convolutional Neural Networks (CNN), offered an innovative approach to overcome these challenges. CNN had the ability to automatically extract visual features from images with high accuracy, making it an effective tool in defect classification. This research used several CNN architectures, including MobileNetV2 and InceptionV3, as well as a model developed in-house, the K3 Model. Data augmentation, such as rotation and lighting adjustments, was applied to increase variation in the dataset and aid the model in generalization. The research results showed that the K3+Augmentation model achieved the highest accuracy of 100% in testing, with a very low loss of 0.0009. While MobileNetV2 offered better training speed, K3+Augmentation showed superior performance in detecting and classifying metal defects. These findings confirmed the potential of CNN in improving the efficiency of quality inspection in modern industry.

KEYWORDS: Metal surface inspection; CNN; MobileNetV2; K3 model; InceptionV3; augmentation

1. Introduction

Product quality was a very important aspect in the manufacturing process, both to maintain product function and aesthetics. Therefore, quality inspection was a key step in ensuring that products met the desired standards. This process included the inspection of various raw materials, such as plastics, liquids, and metals, each of which had different risks of defects. Metal products, for example, had a higher risk of defects due to reduced machine performance, improper storage, and temperature fluctuations during production [1].

The main problems affecting the quality of metal products often stemmed from machines not working optimally, causing defects in the final product. To reduce the number of defective products, visual inspection was an important step. Manual methods were often inefficient and prone to errors made by human operators. In this case, the application of Machine Learning

(ML)-based technology offered a better solution. By using ML, inspections could be carried out automatically with high accuracy and in shorter times [2].

Several Convolutional Neural Networks (CNN) architecture models, such as MobileNetV2 and InceptionV3, demonstrated superior performance in visual inspection tasks, including flaw detection on metal surfaces. MobileNetV2, as a lightweight CNN model, was designed for applications with limited computing resources, providing high efficiency without sacrificing accuracy. This model used inverted residual blocks to reduce memory and computational requirements, making it an ideal choice for performing visual inspection tasks [3]. On the other hand, InceptionV3 relied on a modular architecture with multi-scale parallel convolutions to extract complex features. This advantage made it capable of handling texture variations on metal surfaces that were difficult to detect by traditional methods [4]. In a recent study, Jiangyun et al. (2018) developed an improved YOLO model for surface defects detection of steel strips. The model was used to detect six types of defects, namely scar, scratch, inclusion, burr, seam, and iron scale. Additionally, the model was designed for real-time detection, with an average inference time of only 0.012s to detect a strip surface image.

Apart from pre-trained models such as MobileNetV2 and InceptionV3, this research also developed a special model called the "K3 Model." K3 Model was named after the group members working on this model to overcome specific challenges in metal surface defect inspection. Another model comparison involved adding data augmentation techniques (K3 Model + Augmentation). This model increased generalization by creating variations in the training data. Augmentations such as rotation, flipping, and changing light intensity proved effective in improving model performance on limited datasets [6].

This research focused on the multi-category classification of metal surface defects using the NEU Surface Defect dataset, which came from the database (https://www.kaggle.com/datasets/kaustubhdikshit/neu-surface-defect-database). This dataset included six categories of defects: crazing, inclusions, patches, pitted surfaces, rolled-in scale, and scratches, each of which had unique detection challenges [7]. By comparing several models, namely MobileNetV2, InceptionV3, and self-developed models (K3 Model and K3 Model + Augmentation), this research aimed to evaluate the efficiency and accuracy in detecting metal defects. The results obtained were expected to provide new insights into the application of AI-based technology to improve the quality of manufactured products.

2. Methods

In this research, a Convolutional Neural Networks (CNN)-based architecture was used to recognize patterns and classify visual data, by comparing its effectiveness using the transfer learning method. The transfer learning technique applied involved pre-trained models such as MobileNetV2 and InceptionV3, as well as testing special models designed in this research. The process and steps for applying CNN to overcome this problem are explained in Figure 1, which includes the feature extraction and classification stages using the model being tested.

2.1. Data collection.

Data collection is a systematic process of obtaining information from various sources, aimed at supporting data-based analysis and decision making [8]. In the context of Machine Learning (ML), data collection is a key stage that determines model quality [9]. This research uses datasets available from the Kaggle database, a public data platform that provides a variety of

well-structured datasets for machine learning purposes [10]. To increase variation and overcome dataset limitations, data augmentation techniques such as mirroring, rotation, and lighting adjustment are applied [11]. This step is important to improve the generalization and accuracy of the model. The dataset obtained via Kaggle was further explored with cleaning and normalization to ensure relevance and quality of the data [12]. The dataset chosen is a dataset in the form of images that will be processed and then classified based on the type of product defect. The following are examples of images used in data processing in research. An overview of the image data is available in Table 1.

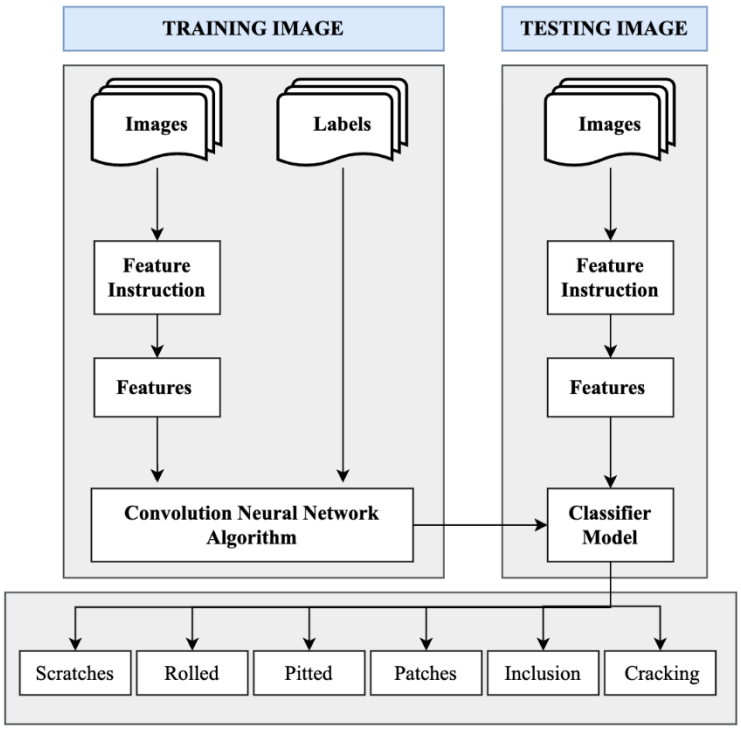
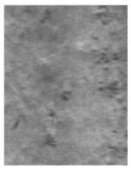
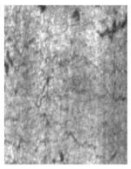
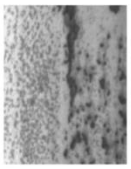
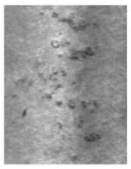
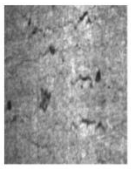
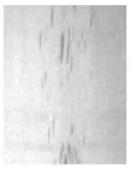
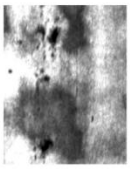
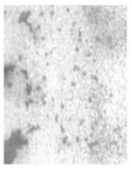
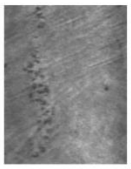
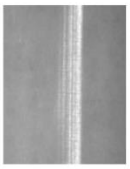


Figure 1. Model working scheme.

Table 1. Six types of metal surface defect images.

Crazing	Inclusion	Patches	Pitted	Rolled	Scratches
					
					
					

The dataset obtained consisted of images with different classifications for each image. The classification of the images was based on the form of product defects. These defects took various forms, namely crazing, inclusions, patches, pitted surfaces, rolled-in scale, and scratches. The dataset contained 1800 images, with 300 images for each classification, each having a size of 200 x 200 pixels. In this study, the dataset was split into 1656 images (92%) for the training dataset, 72 images (4%) for the validation dataset, and 72 images (4%) for the test dataset. The images in the validation and test datasets were drawn from different specimens than those in the training dataset to ensure the fairness of the testing procedures.

2.2. Image Augmentation.

Image augmentation is an important technique in deep learning to artificially increase data variation, especially when the dataset is limited [6]. This technique helps prevent overfitting and improves model generalization through transformations such as rotation, mirroring, and color adjustment [13]. Complex methods like mixup and random erasing also contribute to data variation [14]. Augmentation has been proven to be efficient in applications such as image classification and object detection [15]. However, irrelevant transformations can significantly alter image characteristics, confusing the model, so they need to be applied with caution [11].

To achieve optimal performance, deep learning models generally require large amounts of data. This is essential to improve model quality and reduce the risk of overfitting. However, obtaining large datasets is often hindered by several challenges, such as cost, time, and other factors. To address this issue in image classification, image augmentation can be used to increase the number of training images. During data augmentation, small random transformations are applied to images without altering the main content or key features of the image. Examples of these transformations include mirroring, resizing, and lighting adjustments. Figure 2 shows several examples of images transformed through data augmentation. However, the data augmentation process requires careful consideration to ensure that the augmented data does not significantly differ from the original data. Excessive transformations can lead to confusion in the model, making it difficult to understand the true meaning of the image. This could result in the model making incorrect predictions.

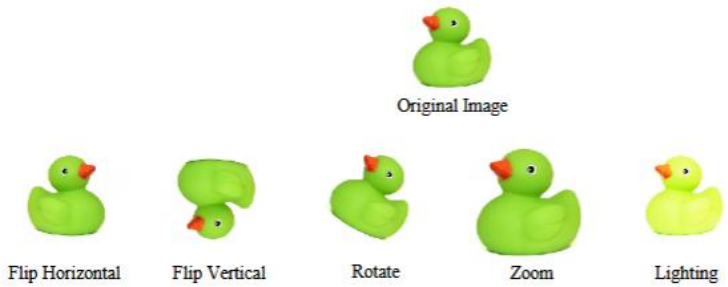


Figure 2. Example of image transformation orientation given by augmentation [16].

In this study, random transformations were applied to each image in the training dataset. This approach was adopted to ensure that the classification model being built is more robust, enabling it to better handle varied training data, prevent overfitting, and improve its accuracy when dealing with previously unseen images. For this research, with a total of 1656 images in the training dataset, several augmentation techniques were applied to each image, as shown in Table 2.

Table 2. Image augmentation parameters

Transformation	*Value*
Rotation	10°
Zoom	0.2
Brightness	[0.8, 1.2]
Horizontal Flip	True
Vertical Flip	True

Table 2 presents the image augmentation parameters used for data processing. Several indicators are included: Rotation, which involves randomly rotating the image by a maximum of 10 degrees to make the model more robust to rotation; Zoom, with a value of 0.2, meaning the image will be enlarged or reduced by up to 20% to address variations in object size in the original data; Image Brightness, which is adjusted randomly between 80% and 120% of the original value to make the model more resilient to lighting variations; Horizontal Flip, set to True, which allows the image to be mirrored during augmentation; and Vertical Flip, indicating that the image can be flipped vertically to provide a variety of vertical orientations.

2.3. Model architecture.

In this research, a simple architecture named K3 was developed, with the detailed architecture shown in Figure 3. The K3 architecture is compared with two widely used architectures, MobileNetV2 and InceptionV3, which are both transfer learning models. These two models have been pre-trained using the ImageNet database to expedite the learning process, and they were subsequently modified to address the specific challenges of this research. Google Colab was used to conduct the model training, utilizing the available Tesla P4 GPU to accelerate the training process.

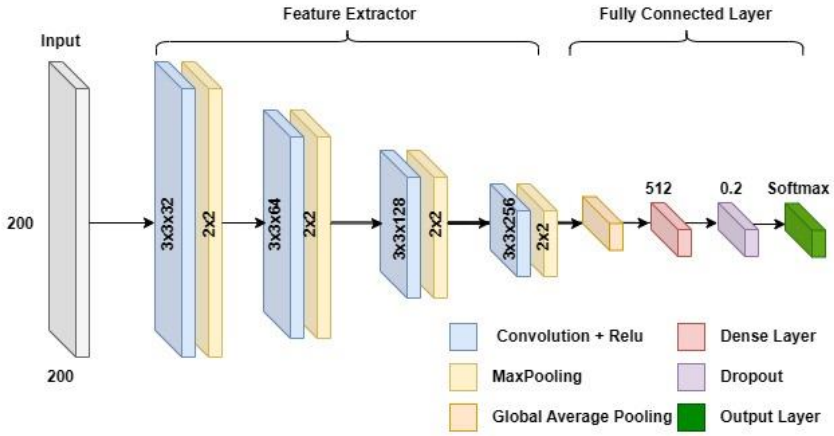


Figure 3. K3 Architecture.

This research compared the K3 architecture with MobileNetV2 and InceptionV3, and further evaluated the results after the K3 architecture was augmented with data. Table 3 below shows the composition of the K3 architecture, along with the parameters used. Table 3 shows the K3 Model Architecture and Parameters. The K3 model was developed to address specific challenges in metal surface defect classification, while balancing accuracy, computational efficiency, and generalization ability. This model is designed to handle image data with input dimensions of 200x200 pixels.

The K3 model consists of four convolutional layers, each followed by a max pooling operation. The decision to use four layers was based on empirical testing and a literature

review, where deeper architectures often improve feature extraction but may lead to overfitting or increased training time. Shallower models with one or two layers failed to extract sufficient hierarchical features, leading to poor defect classification, while deeper models with more than four layers increased computational complexity without providing significant accuracy improvements. The choice of four layers provided an optimal trade-off, effectively capturing edge details, textures, and patterns characteristic of metal defects. Each convolutional layer in K3 uses a 3×3 kernel size, a widely used setting in deep learning models such as VGG and ResNet. This kernel size balances computational efficiency and feature extraction capability. Larger kernels, such as 5×5 or 7×7, were tested but led to increased computational costs without significant accuracy gains. Stacking 3×3 convolutions mimics the effect of larger receptive fields while maintaining efficiency. Following each convolutional layer, a 2×2 max pooling operation is applied to reduce dimensionality and retain critical features. Max pooling was chosen over average pooling because it helps retain dominant defect features while reducing noise. Average pooling was tested but blurred critical edge details, lowering classification performance. The 2×2 window was selected as it provides a balance between feature reduction and preserving spatial details.

Table 3. K3 architecture and parameters.

Layer	Parameter
Convolution 1	32, 3, activation = ‘relu’, input_shape = [200,200,3]
Max Pooling	2
Convolution 2	64,3, activation= ‘relu’
Max Pooling	2
Convolution 3	128, 3, activation= ‘relu’
Max Pooling	2
Convolution 4	256, 3, activation = ‘relu’
Max Pooling	2
Global Average Pooling	-
Dense	512, activation= ‘relu’
Dropout	0.2
Dense Output	6, activation= ‘Softmax’

Before the dense layers, a global average pooling (GAP) layer is used to efficiently summarize spatial features. GAP was chosen over direct flattening as it reduces overfitting by minimizing the parameter count, improving model generalization. Empirical tests showed that GAP improved classification performance over direct flattening, acting as a regularization technique to ensure the model does not rely on localized patterns alone. The model includes a fully connected dense layer with 512 neurons before the output layer. Lower values, such as 128 or 256, resulted in lower accuracy due to insufficient feature extraction, while higher values, such as 1024, led to increased computational cost without proportional accuracy improvements. The choice of 512 neurons provided the best balance of expressiveness and efficiency. A dropout layer with a rate of 0.2 is applied before the final dense layer to prevent overfitting. Higher dropout rates above 0.3 degraded the learning process, leading to slower convergence, while lower dropout rates below 0.2 were insufficient to mitigate overfitting in smaller datasets. The dropout rate of 0.2 was selected based on cross-validation results, showing improved generalization.

ReLU (Rectified Linear Unit) activation is applied to all convolutional and dense layers, except the output layer. ReLU prevents vanishing gradients, ensuring stable model training. Other activations, such as Leaky ReLU and Swish, were tested, but ReLU showed the best

convergence speed. The output layer uses Softmax activation to convert logits into probabilities for the six defect categories. The K3 model was designed considering the strengths and limitations of existing architectures. MobileNetV2 provided high efficiency but slightly lower accuracy, while InceptionV3 demonstrated strong feature extraction but was computationally expensive. The K3 model was designed as a middle-ground approach for performance and efficiency, and the augmented version, K3+Augmentation, achieved the highest accuracy but required the longest training time. The design of the K3 model was optimized to achieve a balance between depth and computational efficiency, ensuring effective feature extraction without excessive training time. The choice of kernel size, pooling strategy, and activation functions were made based on empirical results and theoretical justifications. By integrating these design choices, the K3 model provides a robust approach for metal defect classification. Future work could explore further hyperparameter tuning, attention mechanisms, and lightweight model optimization to improve real-time defect classification performance.

2.4. Model training.

Model training was the process of training a machine learning model to learn data patterns and produce accurate predictions [9]. This research used two CNN architectures, MobileNetV2 and InceptionV3, with a transfer learning approach. Transfer learning allowed pre-trained models, such as ImageNet, to be applied to specific tasks with small datasets, speeding up training and reducing the need for large data [17]. The training process involved data augmentation techniques, such as rotation and flipping, to increase variation and prevent overfitting [11]. The "Adam" optimizer was used because of its stability and convergence efficiency [18]. The other hyperparameters of the training process were set as follows: a batch size of 32, 100 epochs, and a loss function using categorical crossentropy. Model evaluation was carried out using metrics such as accuracy, precision, recall, and the confusion matrix, which provided a complete picture of model performance.

2.5. Model testing.

Model testing was an important stage to evaluate the model's ability to recognize new data. Each trained model was tested using test data, and the test results were summarized in a confusion matrix, as shown in Figure 4. Based on this matrix, metrics such as accuracy, precision, recall, and F1-score were calculated to assess the overall performance of the model [7]. This process provided an understanding of the effectiveness of the model and areas that needed improvement. At this stage, Google Colab was also used to test the model by utilizing the available Tesla 4 GPU to speed up the model testing process.

$$Accuracy = \Sigma TP/n$$

$$Recall_{class} = TPclass/(TPclass + \Sigma FNclass)$$

$$Precision_{class} = TPclass/(TPclass + \Sigma FPclass)$$

Where: TP = True Positive, FN = False Negative, FP = False Positive, n = Amount of Data.

Table 4. Confusion matrix example.

Actual Class	C1	C2	C3	C4	C5	C6
C1	**TP_1**	FP_2/FN_1	FP_3/FN_1	FP_4/FN_1	FP_5/FN_1	FP_6/FN_1
C2	FP_1/FN_2	**TP_2**	FP_3/FN_2	FP_4/FN_2	FP_5/FN_2	FP_6/FN_2
C3	FP_1/FN_3	FP_2/FN_3	**TP_3**	FP_4/FN_3	FP_5/FN_3	FP_6/FN_3
C4	FP_1/FN_4	FP_2/FN_4	FP_3/FN_4	**TP_4**	FP_5/FN_4	FP_6/FN_4
C5	FP_1/FN_5	FP_2/FN_5	FP_3/FN_5	FP_4/FN_5	**TP_5**	FP_6/FN_5
C6	FP_1/FN_6	FP_2/FN_6	FP_3/FN_6	FP_4/FN_6	FP_5/FN_6	**TP_6**

3. Results and Discussion

The experiments (training and testing) were executed in Google Colab (standard license) using the TensorFlow library. Figures 4 to 7 show graphs of the training performance of the models that were trained on the data provided in the dataset. The model performance is displayed through graphical visualization, and the data can be extracted to find several parameters for processing, as shown in Table 5. Table 5 presents the parameters and training times of the four models run on the problems addressed in this research.

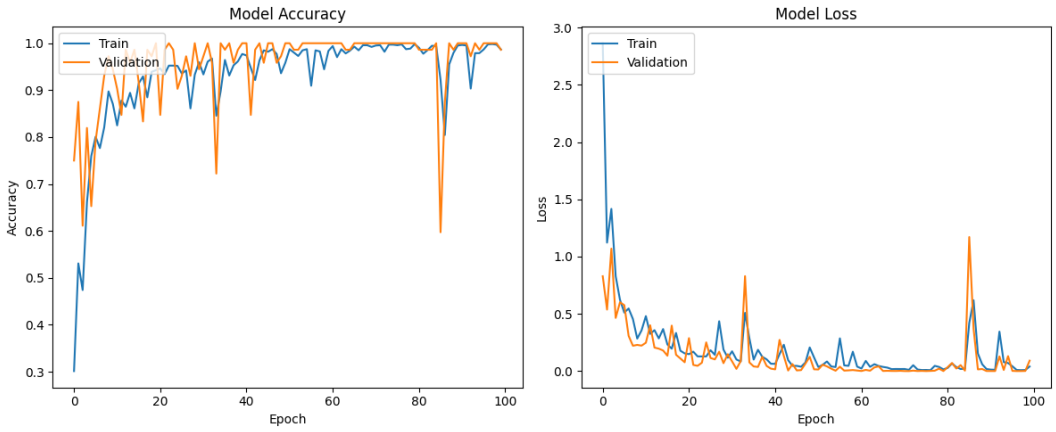


Figure 4. K3 training accuracy and loss graph.

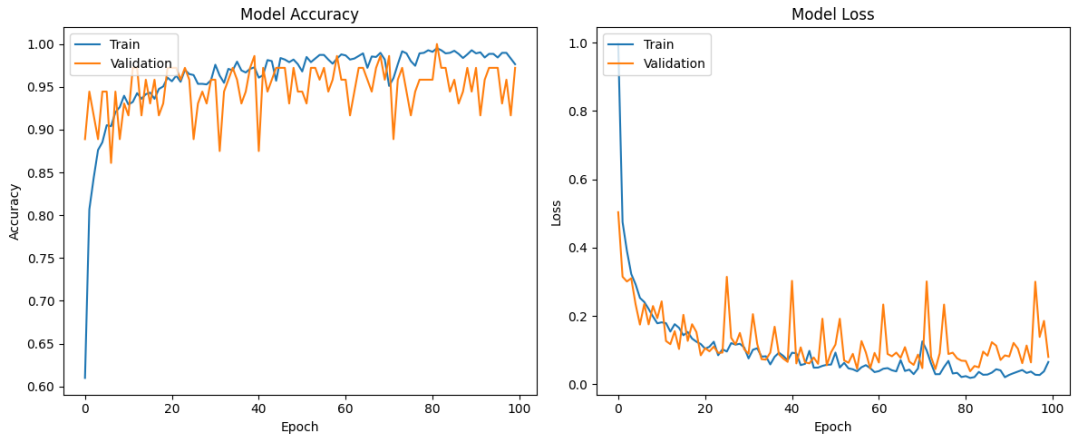


Figure 5. MobileNetV2 training accuracy and loss graph.

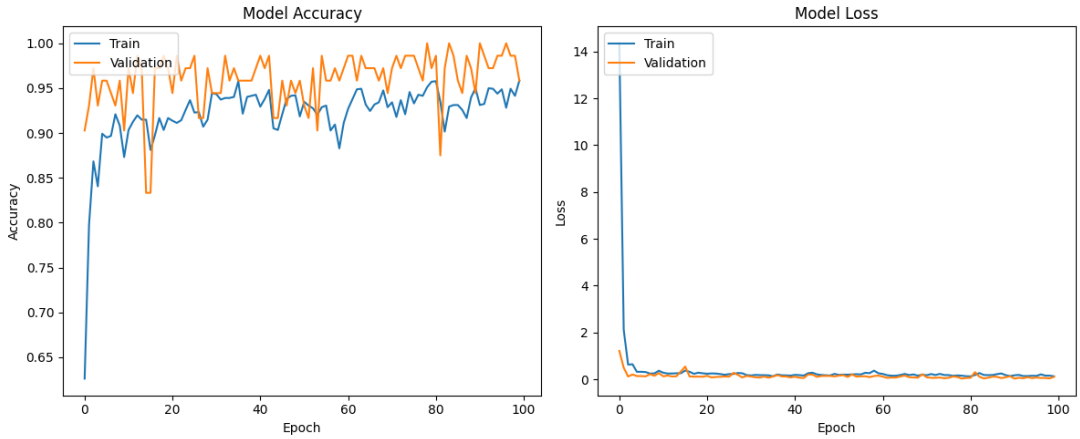


Figure 6. InceptionV3 training accuracy and loss graph.

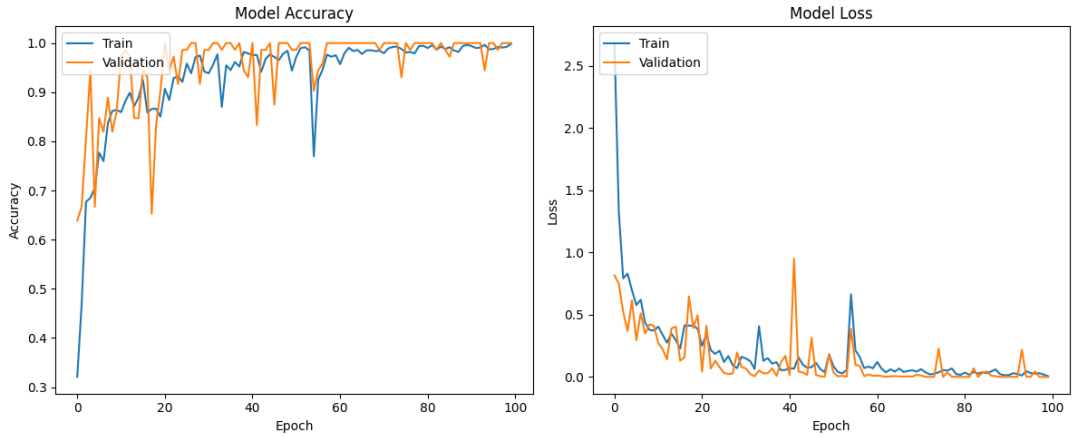


Figure 7. K3 training accuracy and loss graph + augmentation.

Table 5. Architectural model training parameters and time

Parameter	Architecture			
	K3	**MobileNetV2**	**InceptionV3**	**K3+Augmentation**
Training time	4 Mins 29 Sec	3 Mins 36 Sec	7 Mins 10 Sec	34 Mins 11 Sec
Accuracy	0.98	0.998	0.986	0.9968
Loss	0.057	0.0118	0.0373	0.0110

Table 5 explains that the time taken to carry out the training process for each model shows that the MobileNetV2 model has the fastest time, with a duration of 3 minutes and 36 seconds, while the K3+Augmentation model took the longest time, with a duration of 34 minutes and 11 seconds, due to the added data augmentation process. The highest accuracy is achieved by MobileNetV2 at 0.998, indicating the best performance for making predictions, while the lowest accuracy is observed with K3 at 0.98. However, the K3+Augmentation model has a value close to K3, indicating that data augmentation can enhance model generalization. The K3+Augmentation model also has the smallest loss value of 0.0110, indicating that the prediction error rate is very low compared to the others. Based on these results, it can be said that MobileNetV2 is the best choice, offering high performance and short training time, while K3+Augmentation has a small loss but requires a much longer training time compared to the other models.

Based on Figure 7, it is also evident that the model reached its peak accuracy at approximately 80 epochs, after which no further significant improvement occurred. Therefore, incorporating an early stopping mechanism based on validation loss would have optimized

computational efficiency. Implementing such a criterion could have automatically halted training once performance gains became negligible, preventing unnecessary computation. This approach will be considered in future experiments to enhance efficiency while maintaining model performance. Figures 8 to 11 show the performance of the model that was trained on the data provided in the dataset for testing. The model's performance is displayed through the Confusion Matrix table. The data obtained from the Confusion Matrix can then be reprocessed to determine accuracy, precision, and recall values. Table 6 shows the overall testing results of the four models run on the problems addressed in this research.

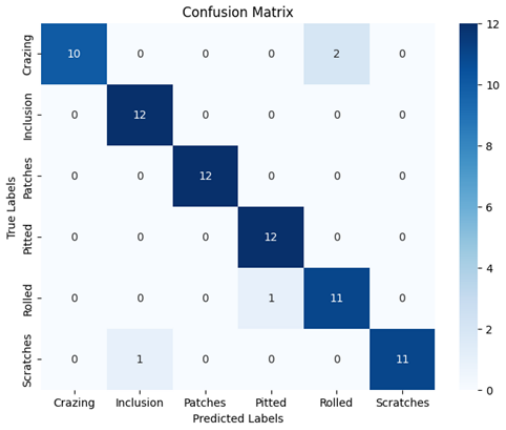


Figure 8. K3 training accuracy and loss graph.

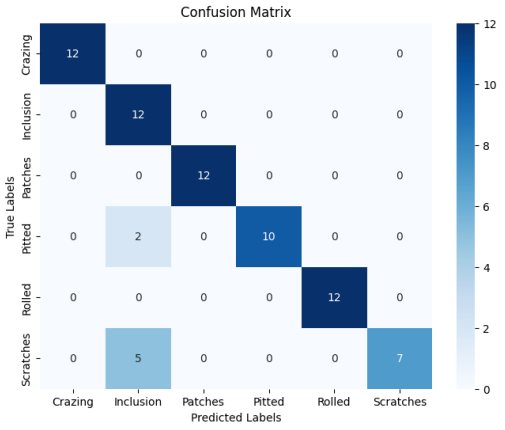


Figure 9. MobileNetV2 training accuracy and loss graph.

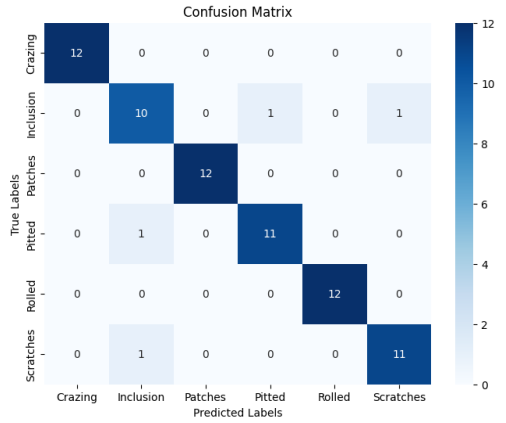


Figure 10. InceptionV3 training accuracy and loss graph.

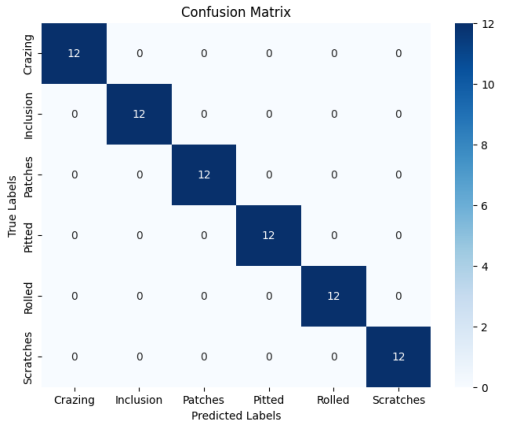


Figure 11. K3 training accuracy and loss graph + Augmentation.

The results of the K3 model confusion matrix in Figure 8 show that the model has high performance, with the majority of correct predictions on the diagonal of the confusion matrix. All classes have dominant correct predictions, but there are a few small errors. For example, the crazing class was incorrectly predicted as scratches in 2 samples, and the rolled class was incorrectly predicted as pitted in 1 sample. Figure 9 for the MobileNetV2 model shows that the generalization of the data is good, as the majority of predictions are on the right diagonal. However, the errors are more significant. For example, in the scratches class, it was wrongly predicted as crazing in 5 samples, and the pitted class was wrongly predicted as crazing in 2 samples. Figure 10 for the InceptionV3 model shows that this model performs well and has high accuracy, as most predictions fall on the correct diagonal. However, it still needs additional data to improve its accuracy, as it has several errors. Specifically, 1 sample in the inclusion class was incorrectly predicted as scratches, and 1 sample in the pitted class was

incorrectly predicted as crazing. Meanwhile, Figure 11 shows the results of the K3+Augmentation model, which demonstrates perfect performance, as all predictions were correct. This indicates that data augmentation can improve the model's ability to recognize various patterns.

Table 6. Architectural model testing parameters and time.

Parameters	Architecture			
	K3	**MobileNetV2**	**InceptionV3**	**K3+Augmentasi**
Inference time	12ms/step	17ms/step	53ms/step	13ms/step
Accuracy	0.9444	0.90278	0.9444	1.0
Loss	0.170	0.268	0.268	0.0009
Avg Precision	0.9487	0.9026	0.9487	1
Avg Recall	0.9444	0.9385	0.9444	1

Based on the results of processing using several models, the testing results for the four architectural models described in Table 6 show that the fastest time is achieved by the K3 model, which is very efficient and has the best performance. The highest accuracy is found in the K3+Augmentation model, which has a perfect value of 1.0 or 100%. The highest loss value in the MobileNetV2 and InceptionV3 models indicates that the prediction error made by these models is greater than in the other models. The highest Average Precision, with a perfect value of 1.00, is achieved by the K3+Augmentation model, demonstrating that it produces very accurate predictions, while the MobileNetV2 model has the lowest value compared to the others. The Average Recall for the K3+Augmentation model, with a perfect value of 1, indicates that the model has a perfect ability to detect all class samples. Meanwhile, previous research using the NEU Surface Defect Database, such as Islam et al. (2018), which developed a CNN model producing an accuracy of 64.7%, and Majeed et al. (2024), which developed the VGG16-LR and InceptionV3-LR models with accuracies of 98% and 96%, respectively, shows that the K3+Augmentation model we built outperforms these models in terms of accuracy

4. Conclusions

Based on the findings of this study, it is evident that CNNs can be effectively utilized for classifying defects on iron surfaces, demonstrating promising results. For future research, we aim to refine and extend the developed model to not only classify defects but also detect their precise locations and categorize the identified defect regions more accurately. A key challenge in defect classification is that, even within the same category, defects may exhibit varying patterns. This can lead to misclassification, particularly when defects resemble those from a different category. Regarding model performance, the training results indicate that MobileNetV2 and K3+Augmentation achieve comparable accuracy and loss values. However, MobileNetV2 trains significantly faster than K3+Augmentation. In testing, K3+Augmentation outperforms the other three tested models, achieving 100% accuracy with an extremely low loss of just 0.09%. Therefore, among the evaluated models—K3, MobileNetV2, and InceptionV3—the K3+Augmentation model demonstrates the best overall performance for this classification task.

Acknowledgment

We would like to express our sincere thanks to the staff of the Department of Mechanical and Industrial Engineering, UGM for providing the facility and assisting during the data collection and experiments.

Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this manuscript.

Authors Contribution

Dhika Wahyu Pratama: Conceptualization, Formal analysis, Investigation, Methodology, Experiment, Validation, Visualization, Writing - original draft; Muchammad Ismail: Conceptualization, Formal analysis, Investigation, Methodology, Experiment, Validation, Visualization, Writing - original draft; Restu Nurraudah: Conceptualization, Formal analysis, Investigation, Methodology, Experiment, Validation, Visualization, Writing - original draft; Achmad Pratama Rifai: Conceptualization, Formal analysis, Investigation, Methodology, Resources, Supervision, Validation, Visualization, Writing - original draft, Writing - review & editing; Nguyen Huu Tho: Resources, Writing - review & editing

References

[1] Cuan-Urquizo, E.; Barocio, E.; Tejada-Ortigoza, V.; Pipes, R. B.; Rodríguez, C. A.; Roman-Flores, A. (2019). Characterization of the mechanical properties of FFF structures and materials: A review on experimental, computational, and theoretical approaches. *Materials*, *12*, 895. https://doi.org/10.3390/ma12060895.

[2] Zaman, U.K.; Boesch, E.; Siadat, A.; Rivette, M.; Baqai, A.A. (2018). Impact of fused deposition modeling (FDM) process parameters on strength of built parts using Taguchi's design of experiments. *The International Journal of Advanced Manufacturing Technology*, *101*, 1215–1226. https://doi.org/10.1007/s00170-018-3014-6.

[3] Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 4510–4520. https://doi.org/10.1109/CVPR.2018.00474.

[4] Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2818–2826. https://doi.org/10.1109/CVPR.2016.308.

[5] Li, J.; Su, Z.; Geng, J.; Yin, Y. (2018). Real-time detection of steel strip surface defects based on improved YOLO detection network. *IFAC-PapersOnLine*, *51*, 76–81. https://doi.org/10.1016/j.ifacol.2018.09.412.

[6] Shorten, C.; Khoshgoftaar, T.M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, *6*, 60. https://doi.org/10.1186/s40537-019-0197-0.

[7] Guan, S.; Lei, M.; Lu, H. (2020). A steel surface defect recognition algorithm based on improved deep learning network model using feature visualization and quality evaluation. *IEEE Access*, *8*, 49885–49895. https://doi.org/10.1109/ACCESS.2020.2979755.

[8] Babbie, E. R. (2020). The Practice of Social Research; Cengage Au: Boston, USA.

[9] Goodfellow, I.; Bengio, Y.; Courville, A. (2016). Deep Learning; MIT Press: Cambridge, USA.

[10] Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A.C.; Fei-Fei, L. (2015). ImageNet Large Scale Visual

Recognition Challenge. *International Journal of Computer Vision*, *115*, 211–252. https://doi.org/10.1007/s11263-015-0816-y.

[11] Zhao, Z.Q.; Zheng, P.; Xu, S.T.; Wu, X. (2019). Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, *30*, 3212–3232. https://doi.org/10.1109/TNNLS.2018.2876865.

[12] Han, J.; Pei, J.; Tong, H. (2022). Data Mining: Concepts and Techniques, 3rd ed.; Morgan Kaufmann: Burlington, USA.

[13] Perez, L.; Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. https://doi.org/10.48550/arXiv.1712.04621

[14] Zhang, H.; Cisse, M.; Dauphin, Y. .; Lopez-Paz, D. (2017). Mixup: Beyond empirical risk minimization. https://doi.org/10.48550/arXiv.1710.09412.

[15] Wong, S.C.; Gatt, A.; Stamatescu, V.; McDonnell, M.D. (2016). Understanding data augmentation for classification: When to warp? *2016* International Conference on Digital Image Computing: Techniques and Applications (DICTA), 1–6. https://doi.org/10.1109/DICTA.2016.7797091.

[16] Rasyidi, M.A.; Bariyah, T. (2020). Batik pattern recognition using convolutional neural network. *Bulletin of Electrical Engineering and Informatics*, *9*, 1430–1437. https://doi.org/10.11591/eei.v9i4.2385.

[17] Tan, M.; Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. International Conference on Machine Learning, 6105–6114.

[18] Kingma, D.P.; Ba, J. (2014). Adam: A method for stochastic optimization. https://doi.org/10.48550/arXiv.1412.6980.

[19] Islam, M.F.; Rahman, M.M. (2018). Metal surface defect inspection through deep neural network. 2018 International Conference on Mechanical, Industrial and Energy Engineering (ICMIEE), 258. https://api.semanticscholar.org/CorpusID:235365832.

[20] Majeed, A.P.A.; Abdullah, M.A.; Nasir, A.F.A.; Razman, M.A.M.; Chen, W.; Yap, E.H. (2024). Surface defect detection: A feature-based transfer learning approach. *Journal of Physics: Conference Series*, *2762*, 012088. https://doi.org/10.1088/1742-6596/2762/1/012088.

© 2025 by the authors. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).