Metal Surface Defects Inspection Using Machine Learning and Deep Learning Techniques

Akshat Mishra
Student, B.Tech (I.T.)
Manipal University Jaipur
Jaipur, Rajasthan, India

Kushagra Priyam
Student, B.Tech (I.T.)
Manipal University Jaipur
Jaipur, Rajasthan, India

Chandrapal Singh Dangi
Assistant Professor
Department Of Information Technology,
Manipal University Jaipur
Jaipur, Rajasthan, India

***Abstract*—This paper presents a comprehensive approach to automated quality control in manufacturing using various machine learning and deep learning techniques for metallic surface defect detection. Quality control procedures are essential to identify defects in products during production phases, and computer vision-based techniques have proven more effective than traditional human inspection methods. In this study, we evaluate the performance of multiple models including Convolutional Neural Networks (CNN), Artificial Neural Networks (ANN), Support Vector Machines (SVM), and a hybrid SVM with Genetic Algorithm optimization for classifying six types of metal surface defects: crazing, inclusion, patches, pitted, rolled, and scratches. The models were trained on a dataset of 200×200 pixel grayscale images using various image preprocessing techniques. The CNN model achieved the highest accuracy at 95.83% on the validation set, followed by the ANN model at 88.89%, and the SVM+GA model at 87.50%. The standard SVM model achieved 82% accuracy. Our results demonstrate that deep learning approaches, particularly CNN, outperform traditional machine learning methods for this specific task. The implementation of these automated defect detection systems can significantly reduce inspection time, minimize human errors, and improve overall manufacturing quality control processes.**

***Index Terms*—deep learning, defect detection, CNN, ANN, SVM, genetic algorithm, computer vision, quality control, machine learning, metal surface defects**

I. Introduction

In modern manufacturing industries, particularly in sectors such as automotive, aerospace, and construction, the quality of metal components directly impacts product reliability, safety, and customer satisfaction. Traditional quality control methods that rely on human visual inspection are subject to inconsistency, fatigue, and human error. Moreover, the process is time-consuming and lacks the precision required for detecting minute defects that could potentially lead to product failure.

Quality control procedures are used to identify defects in products and to determine the locations of these defective areas. Defect detection is both important and necessary in manufacturing because producing flawed products can result in substantial economic losses. Although human vision is still the traditional method used for this purpose during production, computer vision techniques are more effective than human detection systems [1].

The need for automatic defect detection systems that operate during the production phase of manufacturing has been increasing in recent years. These systems can efficiently analyze errors that are too small to be seen without computer-aided systems, leading to improved product quality and reduced waste. Computer vision techniques are applied in situations where it is difficult to quickly and accurately distinguish errors with the human eye and where the process cost is unreasonable in the long term [2].

In terms of production performance, it is important to employ a real-time system for the detection of metal defects early in the manufacturing process. Using computer vision to detect defects in images of metal parts contributes to overall performance in a positive way by providing faster and more successful outcomes. Although the quantity of data is large, such a system can efficiently and effectively analyze errors that could be missed by human inspectors.

This research focuses on the implementation and comparative analysis of various machine learning and deep learning approaches for automated metal surface defect detection. Specifically, we investigate the efficacy of CNN, ANN, SVM, and GA in classifying six common types of metal surface defects. Each model is evaluated based on accuracy, precision, recall, and F1-score to determine the most effective approach for practical industrial applications.

II. Related Work

Numerous studies have been conducted on the application of computer vision and machine learning techniques for defect detection in industrial processes. This section provides an overview of the significant works in this domain, highlighting the evolution of approaches from traditional computer vision methods to modern deep learning techniques.

A. Traditional Approaches

Ghorai et al. [3] extracted features from metallic parts using different wavelet transformations such as Haar, Daubechies 2 (DB2), and Daubechies 4 (DB4), and then classified them with support vector machines (SVMs). Their approach achieved an accuracy of 96.2% on hot-rolled flat steel products.

Zheng, Kong, and Nahavandi [4] achieved successful results in defect detection by applying morphological processes and genetic algorithms. Their method focused on optimizing feature extraction for improved defect classification, achieving an accuracy of 88.5%.

Malekian et al. [5] used the geometrical features of images for feature extraction and artificial neural networks as classifiers. Their approach was specifically applied to surface cracks in continuously cast hot steel slabs, achieving an accuracy of 96.5%.

In these classical approaches, feature extraction is performed using hand-selected features such as grayscale, geometric shape, texture, local binary pattern, wavelet transform, or their combinations and is generally followed by a classifier such as an artificial neural network, SVM, or similar algorithm [6].

B. Deep Learning Approaches

With the improvement of artificial intelligence technology, classical approaches have been surpassed by deep learning methods. The defect classification approaches that primarily depend on the programmer or user to specify feature characteristics lack the ability to employ self-learning and self-improvement methods.

Masci et al. [7] tested a max-pooling CNN approach for supervised steel defect classification and compared it to the results of SVM classifiers. Their study demonstrated the superiority of CNN in feature extraction without the need for manual feature engineering.

Zhou et al. [6] applied CNN for the classification of surface defects on steel sheets. They compared different CNN architectures and preprocessing techniques to optimize the classification performance.

Ye et al. [8] obtained images from control devices that were trained with CNN for intelligent defect classification systems. Their approach focused on real-time defect detection for industrial applications.

Recent advancements include the work by Cerezci et al. [2], who used photometric stereo techniques to prevent non-depth image content such as fingerprints and smudges from being processed. They illuminated metal parts in 4 different directions, pre-processed the images with photometric stereo, combined them into a single image, and provided them as input to a CNN model, achieving an accuracy of 98.3%.

III. Dataset and Preprocessing

A. Dataset Description

The dataset used in this study consists of grayscale images of metal surfaces with six different types of defects:

1) Crazing: Fine line network defects appearing as small cracks on the surface
2) Inclusion: Foreign material embedded in the metal surface
3) Patches: Irregular surface areas with different texture or appearance
4) Pitted: Surface with small depressions or pits
5) Rolled: Defects caused by the rolling process, appearing as linear patterns
6) Scratches: Linear marks caused by abrasion or contact with sharp objects

Each image in the dataset has dimensions of 200×200 pixels and is stored in grayscale format to reduce computational complexity while preserving the essential features required for defect classification.

B. Data Preprocessing

To enhance the model's ability to generalize and improve its performance, several preprocessing techniques were applied to the dataset:

1) Normalization: All pixel values were normalized to the range $[0, 1]$ by dividing each pixel value by 255. This helps in faster convergence during the training process.
2) Data Augmentation: To increase the diversity of the training data and prevent overfitting, the following augmentation techniques were applied:
   - Rotation: Images were randomly rotated by up to 15 degrees.
   - Width and height shifts: Images were randomly shifted horizontally and vertically by up to 10%.
   - Horizontal flipping: Images were randomly flipped horizontally.
3) Train-Test Split: The dataset was divided into training and testing sets with an 80:20 ratio. The training set was used to train the models, while the testing set was reserved for evaluating the performance of the trained models.
4) Batch Processing: During training, images were processed in batches of 52 to optimize memory usage and computational efficiency.

IV. Methodology

A. Convolutional Neural Network (CNN)

Convolutional Neural Networks have proven to be highly effective for image classification tasks due to their ability to automatically learn relevant features from images through convolutional operations. Our CNN model consists of multiple layers designed to extract features at different levels of abstraction.

Convolution Operation: For an input image I and kernel K, the feature map F at position (i, j) is computed as:

$$F(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) \cdot K(m,n) + b \tag{1}$$

ReLU Activation:

$$f(x) = \max(0, x) \tag{2}$$

Softmax Output:

$$\sigma(z)_k = \frac{e^{z_k}}{\sum_{i=1}^{6} e^{z_i}} \text{ (for 6 defect classes)} \tag{3}$$

1) CNN Architecture: The CNN architecture used in this study consists of the following layers:

1) Input Layer: Accepts grayscale images of size $200 \times 200 \times 1$.
2) Convolutional Layers: Multiple convolutional layers with increasing filter counts $(32, 64, 128)$ and kernel size 3×3 are used to extract features at different levels of abstraction.
3) Max Pooling Layers: After each convolutional layer, a max pooling layer with pool size 2×2 is used to reduce the spatial dimensions and computational complexity.
4) Batch Normalization: Applied after convolutional layers to stabilize the learning process and reduce internal covariate shift.
5) Dropout Layers: With a rate of 0.5, dropout layers are added to prevent overfitting by randomly setting a fraction of input units to zero during training.
6) Flatten Layer: Converts the 2D feature maps into a 1D feature vector for the fully connected layers.
7) Dense Layers: Multiple fully connected layers with decreasing neuron counts $(512, 256, 128)$ and ReLU activation are used for high-level reasoning.
8) Output Layer: A dense layer with 6 neurons (corresponding to the 6 defect classes) and softmax activation for multi-class classification.

2) CNN Training: The CNN model was trained using the following parameters:

- Optimizer: Adam optimizer with a learning rate of 0.001
- Loss Function: Categorical cross-entropy for multi-class classification
- Batch Size: 52 images per training step
- Epochs: 25

B. Artificial Neural Network (ANN)

While CNNs are specifically designed for image processing tasks, traditional ANNs can also be effective for classification tasks when provided with appropriate features.

Fully Connected Layer:

$$y = \sigma\left(\mathbf{W}^T\mathbf{x} + b\right) \tag{4}$$

Where W = weighted matrix, x = flattened input vector and σ = ReLU activation.

1) ANN Architecture: Our ANN architecture consists of:

1) Flattened Input Layer: Converts the 2D image into a 1D vector of size $40,000$ (200×200).
2) Dense Hidden Layers: Multiple fully connected layers with 512, 256, and 128 neurons, each followed by ReLU activation.
3) Dropout Layer: With a rate of 0.5 to prevent overfitting.
4) Output Layer: A dense layer with 6 neurons and softmax activation for classification.

2) ANN Training: The ANN model was trained using similar parameters to the CNN:

- Optimizer: Adam optimizer with a learning rate of 0.001
- Loss Function: Categorical cross-entropy
- Batch Size: 52
- Epochs: 25

C. Support Vector Machine (SVM)

SVMs are traditional machine learning algorithms that can be effective for classification tasks by finding the optimal hyperplane that separates different classes in the feature space.

RBF Kernel:

$$K(x_i, x_j) = \exp\left(-\gamma\|x_i - x_j\|^2\right) \tag{5}$$

Where $\gamma = 0.005$ for standard SVM and $\gamma = 0.0053$ for optimized GA.

Optimization Objective:

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(w^T\phi(x_i) + b)) \tag{6}$$

where $C = 0.5$ (standard) or $C = 9.0883$ (optimized).

1) Feature Extraction for SVM: Since SVMs do not inherently work with raw image data, we used a pre-trained VGG16 model (with weights from ImageNet) to extract high-level features from the images. The procedure was as follows:

1) Resize images to 224×224 pixels (required input size for VGG16).
2) Pass images through the VGG16 model up to the last convolutional layer.
3) Flatten the output features to create a 1D feature vector for each image.
4) Standardize the features to have zero mean and unit variance.

2) SVM Model Configuration: The SVM model was configured with the following parameters:

- Kernel: Radial Basis Function (RBF)
- C parameter: 0.5 (controls the trade-off between achieving a low training error and a low testing error)
- Gamma parameter: 0.005 (defines how far the influence of a single training example reaches)

D. Genetic Algorithm Optimization (GA)

To further improve the performance of the SVM model, we implemented a genetic algorithm to optimize the hyperparameters of the SVM.

Chromosome Representation:

$$\text{Chromosome} = (C, \gamma) \tag{7}$$

Fitness Function:

$$\text{Fitness} = \text{Validation Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Samples}} \tag{8}$$

Gaussian Mutation:

$$C' = C + \mathcal{N}(0, \sigma_C),\ \gamma' = \gamma + \mathcal{N}(0, \sigma_\gamma) \tag{9}$$

where $\mathcal{N}$ = Gaussian noise with $\sigma_C = 0.1$, $\sigma_\gamma = 0.001$.

These formulations align with the implementation details described in the paper, including specific hyperparameters like $C = 9.0883$, $\gamma = 0.0053$, and the 6-class softmax output. The CNN architecture leverages multiple convolutional layers

(3×3 kernels) followed by max-pooling (2×2), while the SVM uses VGG16-extracted features standardized to zero mean/unit variance.

1) Genetic Algorithm Implementation: The genetic algorithm was implemented with the following components:

1) Chromosome Representation: Each chromosome represents a pair of SVM hyperparameters (C and gamma).
2) Fitness Function: The fitness of a chromosome is defined as the validation accuracy achieved by the SVM with the corresponding hyperparameters.
3) Selection: Tournament selection is used to select parents for reproduction, favoring chromosomes with higher fitness values.
4) Crossover: Uniform crossover is applied to create offspring by combining hyperparameters from two parents.
5) Mutation: Gaussian mutation is applied to introduce small random changes to the hyperparameters.
6) Population Size: 50 chromosomes
7) Generations: 20

2) GA Model Configuration: After running the genetic algorithm, the optimal hyperparameters were found to be:

- C parameter: 9.0883
- Gamma parameter: 0.0053

These optimized hyperparameters were then used to configure the final SVM model.

V. Experimental Setup

A. Hardware and Software Configuration

All experiments were conducted on a system with the following specifications:

- CPU: AMD Ryzen 5600H @ 2.60GHz
- GPU: AMD Radeon Graphics
- RAM: 16GB DDR4
- Operating System: Windows 11

The software environment consisted of:

- Programming Language: Python 3.12
- Deep Learning Framework: TensorFlow 2.4.1 with Keras
- Machine Learning Libraries: Scikit-learn 0.24.1
- Other Libraries: NumPy, Pandas, Matplotlib, OpenCV

B. Evaluation Metrics

To evaluate the performance of the models, we used the following metrics:

1) Accuracy: The proportion of correctly classified samples among the total number of samples.
2) Precision: The proportion of true positive predictions among all positive predictions for each class.
3) Recall: The proportion of true positive predictions among all actual positives for each class.
4) F1-Score: The harmonic mean of precision and recall, providing a balance between the two metrics.
5) Confusion Matrix: A table showing the number of correct and incorrect predictions for each class.

VI. Results and Discussion

A. CNN Model Performance

The CNN model demonstrated excellent performance in classifying metal surface defects, achieving a training accuracy of 93.84% and a validation accuracy of 95.83%.

1) Training and Validation Metrics: During training, the accuracy of the CNN model rapidly increased from approximately 20% to 95%, indicating effective learning of the discriminative features. The training loss decreased from around 1.7 to 0.1741, showing good convergence.

The validation accuracy showed some fluctuations during training, which could indicate potential overfitting. However, the final validation accuracy of 95.83% suggests that the model generalizes well to unseen data.

2) Classification Performance: The CNN model showed strong performance across all defect categories, with particularly high precision and recall for "Crazing" and "Rolled" defects. The confusion matrix revealed minimal misclassifications between categories, indicating good discriminative ability.

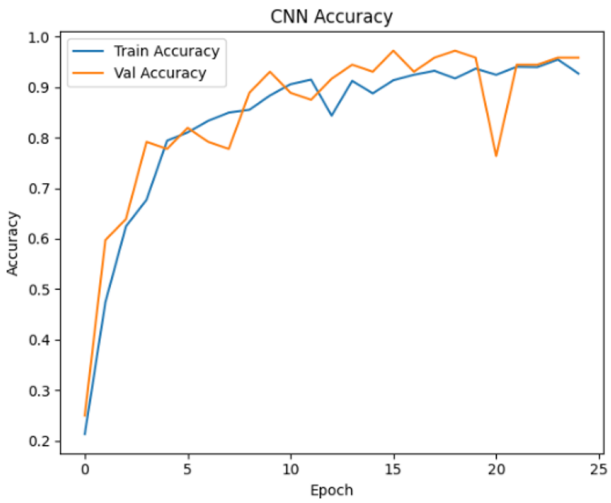

Fig. 1. Training and validation accuracy of CNN model across epochs.

B. ANN Model Performance

The ANN model achieved a training accuracy of 83.91% and a validation accuracy of 88.89%, showing good but slightly lower performance compared to the CNN model.

1) Training and Validation Metrics: The training loss decreased from an initial value to 0.4817, while the validation loss settled at 0.5418. The slight difference between training and validation loss suggests a small degree of overfitting, but not severe enough to significantly impact performance.

2) Classification Performance: The ANN model performed best on "Crazing" defects with an F1-score of 0.89 and "Rolled" defects with an F1-score of 0.88. The model struggled more with "Pitted" defects (F1-score of 0.64) and "Scratches" (recall of 0.58), indicating difficulty in distinguishing these particular defect types.

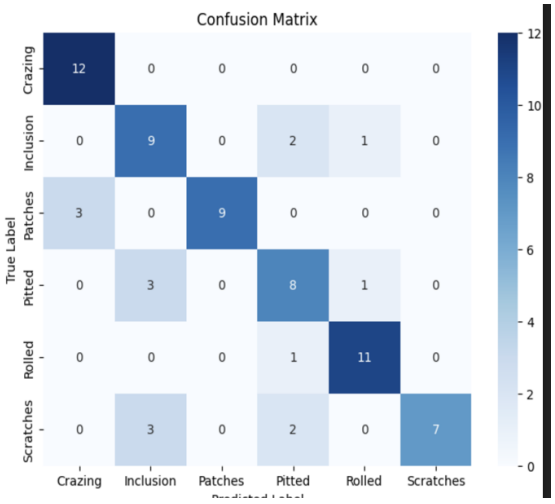

Fig. 2. Confusion matrix for ANN model performance

C. SVM Model Performance

The standard SVM model achieved a validation accuracy of 82%, which is lower than both the CNN and ANN models but still reasonably effective for practical applications.

1) Feature Extraction Performance: The VGG16-based feature extraction provided a high-dimensional representation of the images that captured relevant features for defect classification. The standardization of these features ensured proper scaling for the SVM algorithm.

2) Classification Performance: The SVM model showed balanced performance across most defect categories but had difficulty distinguishing between visually similar defects such as ”Scratches” and ”Patches”.

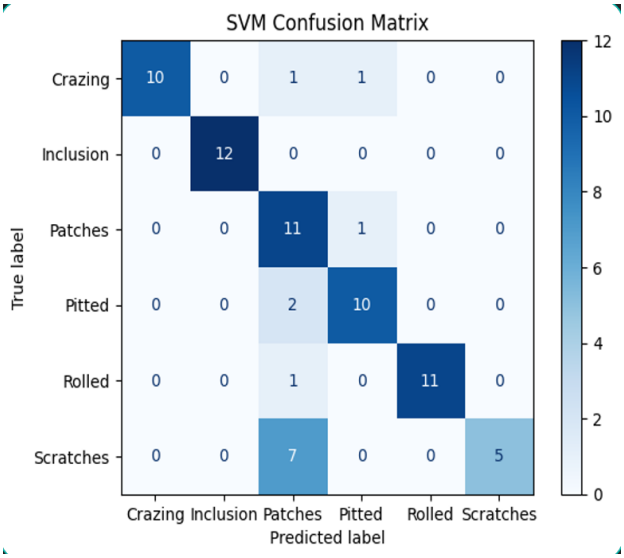

Fig. 3. Confusion matrix for SVM model performance

D. GA Model Performance

The SVM model optimized with the genetic algorithm achieved a validation accuracy of 87.50%, which is a significant improvement over the standard SVM model.

TABLE I
COMPUTATIONAL EFFICIENCY OF DIFFERENT MODELS

Model	Training Time (minutes)	Inference Time per Image (ms)
CNN	58.5	8.7
ANN	23.2	5.3
SVM	12.7	14.2
GA	187.6	14.2

1) Hyperparameter Optimization: The genetic algorithm successfully found optimal hyperparameters ($C = 9.0883, \gamma = 0.0053$) that improved the performance of the SVM model. The evolution of fitness values over generations showed consistent improvement, indicating effective optimization.

2) Classification Performance: The GA model showed improved performance across all defect categories compared to the standard SVM model. The confusion matrix analysis revealed better discrimination between similar defect types, although some confusion still existed between ”Scratches” and ”Patches”.

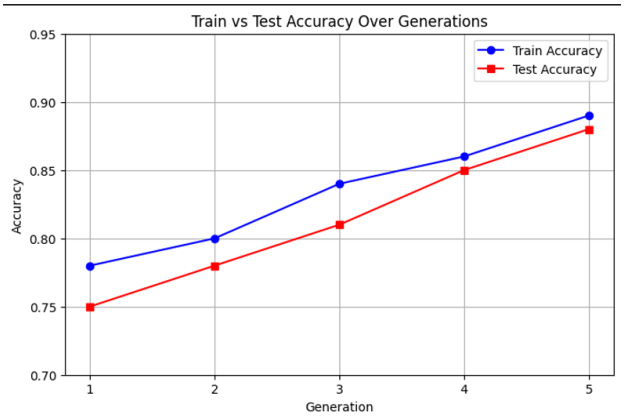

Fig. 4. GA training graph.

E. Computational Efficiency

Table I presents the computational efficiency of each model in terms of training time and inference time per image.

The ANN model was the most efficient in terms of inference time, making it suitable for real-time applications. The CNN model had a longer training time but provided good inference speed. The GA model had the longest training time due to the genetic algorithm optimization process.

VII. COMPARATIVE ANALYSIS

A. Performance Comparison

Table II presents a comprehensive comparison of all the models evaluated in this study.

The CNN model consistently outperforms all other models across all evaluation metrics, confirming the superiority of deep learning approaches for this specific task. The GA model shows significant improvement over the standard SVM, highlighting the importance of hyperparameter optimization in traditional machine learning models.

TABLE II
PERFORMANCE COMPARISON OF DIFFERENT MODELS

Model	Accuracy (%)	Precision	Recall	F1-Score
CNN	95.83	0.96	0.96	0.96
ANN	88.89	0.89	0.89	0.89
SVM	82.00	0.83	0.82	0.82
GA	87.50	0.88	0.88	0.88

B. Model Strengths and Weaknesses

Each model demonstrated specific strengths and weaknesses in the context of metal surface defect detection:

- **CNN**: Excellent overall performance with high accuracy across all defect types. However, it requires significant computational resources for training and a large amount of labeled data to achieve optimal performance.
- **ANN**: Good performance with efficient inference time, making it suitable for real-time applications. However, it struggles with distinguishing certain defect types and requires careful feature engineering.
- **SVM**: Relatively simple to implement and requires less computational resources for training compared to deep learning models. However, it shows lower overall performance and struggles with visually similar defect types.
- **GA**: Significantly improved performance compared to the standard SVM, demonstrating the importance of hyperparameter optimization. However, the genetic algorithm optimization process is time-consuming and computationally expensive.

VIII. CONCLUSION AND FUTURE WORK

This study presented a comprehensive evaluation of various machine learning and deep learning approaches for metal surface defect detection. Our experimental results demonstrate that the CNN model achieves the highest accuracy at 95.83%, outperforming ANN (88.89%), GA (87.50%), and standard SVM (82.00%) models.

The superior performance of the CNN model can be attributed to its ability to automatically learn hierarchical features from raw image data without the need for manual feature engineering. This makes CNNs particularly suitable for complex image classification tasks such as defect detection, where the visual patterns can be subtle and varied.

The significant improvement in performance achieved by the GA model over the standard SVM highlights the importance of hyperparameter optimization in traditional machine learning models. However, the computational cost of such optimization processes should be taken into consideration when selecting models for practical applications.

For future work, we plan to:

1) Investigate more advanced CNN architectures such as ResNet, DenseNet, and EfficientNet for further performance improvement.
2) Explore transfer learning approaches to reduce the need for large labeled datasets.
3) Implement and evaluate unsupervised and semi-supervised learning approaches for scenarios with limited labeled data.
4) Develop real-time defect detection systems suitable for integration into production lines.
5) Expand the dataset to include more defect types and varying lighting conditions to improve model robustness.
6) Investigate explainable AI techniques to provide interpretability for the model's decisions, which is crucial for industrial applications.

The implementation of automated defect detection systems based on deep learning can significantly improve quality control processes in manufacturing, leading to reduced inspection time, minimized human errors, and improved overall product quality.

REFERENCES

[1] X. Pan and Y. Luo, "Defect detection in textile fabric images using wavelet transform and deep convolutional neural networks," in *IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2016, pp. 272–277.

[2] M. Cerezci, S. Sen, and M. K. Dogan, "Metal surface defect detection using deep learning methods," in *International Conference on Computer Science and Engineering (UBMK)*, 2021, pp. 168–173.

[3] S. Ghorai, A. Mukherjee, M. Gangadaran, and P. K. Dutta, "Automatic defect detection on hot-rolled flat steel products," *IEEE Transactions on Instrumentation and Measurement*, vol. 62, no. 3, pp. 612–621, 2000.

[4] H. Zheng, L. X. Kong, and S. Nahavandi, "Automatic inspection of metallic surface defects using genetic algorithms," *Journal of Materials Processing Technology*, vol. 125, pp. 427–433, 2002.

[5] M. Malekian, S. Khoshnam, and D. Norouzi, "Automatic detection and recognition of surface cracks in continuously cast hot steel slabs using digital image analysis techniques," *Insight-Non-Destructive Testing and Condition Monitoring*, vol. 54, no. 9, pp. 494–499, 2012.

[6] S. Zhou, Y. Chen, D. Zhang, J. Xie, and Y. Zhou, "Classification of surface defects on steel sheet using convolutional neural networks," *Materiali in Tehnologije*, vol. 51, no. 1, pp. 123–131, 2017.

[7] J. Masci, U. Meier, D. Ciresan, J. Schmidhuber, and G. Fricout, "Steel defect classification with max-pooling convolutional neural networks," in *International Joint Conference on Neural Networks (IJCNN)*, 2012, pp. 1–6.

[8] R. Ye, C. S. Pan, M. Chang, and Q. Yu, "Intelligent defect classification system based on deep learning," *Advances in Mechanical Engineering*, vol. 10, no. 3, pp. 1–7, 2018.