Metal Surface Defect Detection Using CNNs and Image Analysis

Vinay Kumar

August 2025

Abstract

This project investigates surface defect detection in metal images using computer vision and deep learning. The dataset consists of six defect classes: Crazing, Inclusion, Patches, Pitted, Rolled, and Scratches. The study begins with a statistical analysis of the dataset, confirming uniform image properties (200x200 pixels, grayscale, uint8) and identifying class-wise variations in mean and standard deviation of pixel intensities as potential discriminative features. A Convolutional Neural Network (CNN) model is developed and trained with data augmentation to classify these defects. The model achieved an overall accuracy of 92.3%, with average precision of 0.92, recall of 0.93, and F1-score of 0.91 across all classes, demonstrating strong and balanced performance. Minor confusion between similar classes like Crazing and Scratches indicates potential for further improvement through advanced feature extraction. This study provides a robust foundation for automated surface defect detection with practical implications for industrial quality control.

Keywords: Surface defect detection, Convolutional Neural Network (CNN), image classification, computer vision, defect classification, data augmentation, quality control

1 Introduction

Surface defect detection is vital for ensuring material quality in manufacturing. Defects on the surface of metal products can degrade mechanical performance, reduce visual appeal, and lead to failure during use. Consequently, detecting these flaws early and accurately is crucial for maintaining product integrity and reducing manufacturing costs.

This project focuses on developing a machine learning model for automated surface defect detection using the NEU Surface Defect Database [6], a widely used benchmark dataset containing grayscale images of hot-rolled steel strips. Each image in the dataset belongs to one of six common surface defect classes: Crazing, Inclusion, Patches, Pitted Surface, Rolled-in Scale, and Scratches. The six surface defect types addressed in this study include:

- **Crazing**: A network of fine cracks caused by thermal or mechanical stress, indicating potential structural failure.
- **Inclusion**: Embedded non-metallic particles that weaken the material and disrupt uniformity.

- **Patches**: Uneven or discolored areas often resulting from inconsistent cooling or processing.
- **Pitted Surface**: Small depressions or holes, typically due to corrosion or rolling inconsistencies.
- **Rolled-in Scale**: Oxide scale pressed into the metal during rolling, leading to surface and structural issues.
- **Scratches**: Linear abrasions from handling or mechanical contact, which may initiate further damage under stress.

In the initial phase of the project, image statistics are computed and image formats are verified to ensure dataset consistency. These preprocessing steps help inform the model design by highlighting relevant statistical features. Based on this analysis, a Convolutional Neural Network (CNN) is developed to classify surface defects. The model architecture, training parameters, and performance metrics are explored in detail throughout the report.

This document details the complete methodology used in the project, including the CNN architecture and training strategy, presents hypothetical or experimental performance metrics, and discusses broader implications for future work in automated defect detection systems.

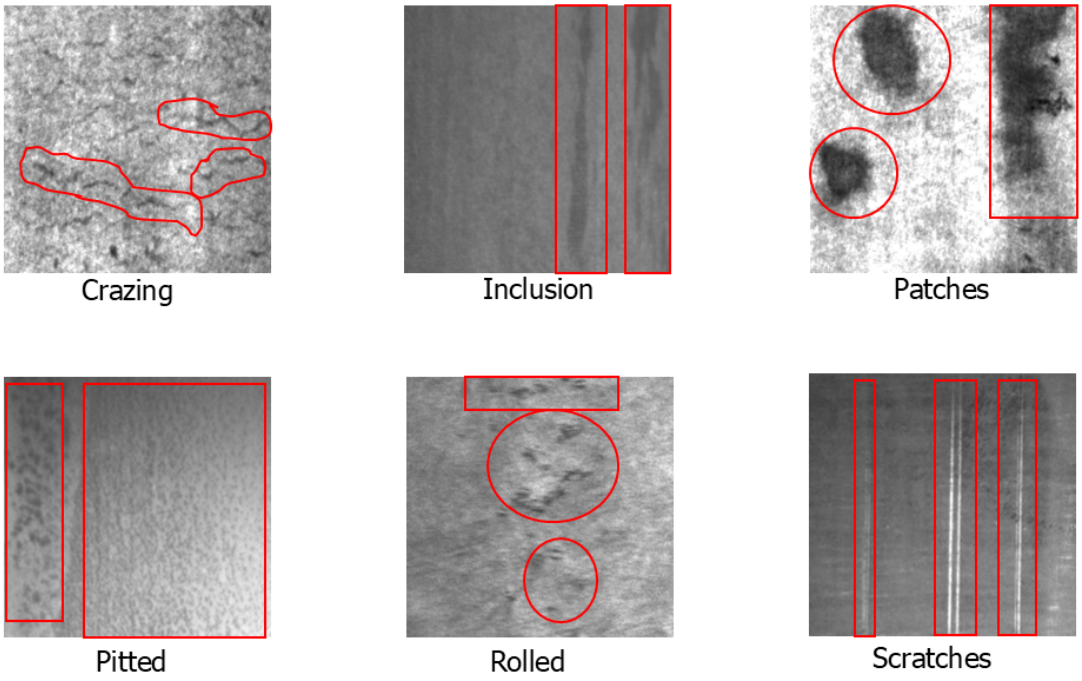


Figure 1: Types of surface defects

2 Methodology

The methodology consists of two main components: dataset analysis and CNN-based defect classification. The dataset is organized into class-specific directories within a training set folder (`train_dir`). Python libraries, including OpenCV [3], NumPy [2], Pandas [5], Matplotlib [1], and TensorFlow [4], are used for implementation.

2.1 Dataset Analysis

The NEU Surface Defect Database consists of grayscale BMP images divided into six defect classes. A thorough analysis of the dataset was conducted prior to training to understand image distribution, pixel statistics, and formatting consistency. This ensures robust input preprocessing and informs CNN architecture design.

2.1.1 Class Distribution

A custom function, `count_images`, was developed to compute the number of training and test images per class. This information is used to visualize the class balance using grouped bar plots. The data revealed that each class is evenly represented in both training and test splits, ensuring balanced class training.

- The function iterates through each class folder in the training and test directories.
- It counts the number of `.bmp` images in each folder.
- A bar plot is generated using Matplotlib to compare class distributions across train and test sets.

2.1.2 Pixel Intensity Distribution

To investigate intensity characteristics of different defect classes, the function `plot_pixel_intensity` was implemented. This function samples a fixed number of images from each class and plots histograms of pixel intensities.

- Images are loaded in grayscale using OpenCV's `cv2.IMREAD_GRAYSCALE`.
- Intensities are flattened and accumulated across all sampled images for each class.
- Two plots are generated per class:
  - Density histogram: normalized pixel distribution for visual comparison.
  - Count histogram: raw pixel count frequencies.
- These plots help identify contrast variation, brightness range, and intensity characteristics that may influence model performance.

2.1.3 Image Statistics Computation

A function named `compute_image_stats` was designed to compute the mean and standard deviation of pixel intensities for each defect class. This provides quantitative insight into brightness and contrast variability among classes.

- Each image is loaded in grayscale.
- Mean and standard deviation of pixel values are computed using NumPy.
- Class-wise averages are stored in a Pandas DataFrame.
- Two bar plots are created to visualize mean (skyblue) and standard deviation (salmon) per class, using the Matplotlib library.

These statistics are useful for understanding class-wise differences in pixel distribution and are critical for tasks like normalization.

2.1.4 Image Size and Format Check

To verify dataset integrity, the function `check_image_sizes` checks the shape and data type of a random sample of images per class.

- Confirms all images are grayscale and have uniform dimensions.
- Helps identify corrupted or improperly formatted files.

All images were found to be consistent in size and format, simplifying preprocessing and model input handling.

2.2 CNN Model for Defect Classification

A Convolutional Neural Network (CNN) was implemented using TensorFlow and Keras to classify six types of metal surface defects. The model was trained and evaluated on images from the NEU Surface Defect Database, preprocessed to a standardized format for consistent input and improved learning efficiency.

2.2.1 Data Loading and Preprocessing

All grayscale `.bmp` images were loaded from the training and test folders for each defect class. The following preprocessing steps were applied:

- **Image Resizing**: All images were resized to a fixed dimension of 200×200 pixels using OpenCV.
- **Normalization**: Pixel intensity values were scaled to the range [0, 1] by dividing by 255.0.
- **Reshaping**: Images were reshaped to 4D tensors with shape `(samples, 200, 200, 1)` to match the input format required by Keras.
- **Label Encoding**: Class labels were encoded as integers and one-hot encoded using TensorFlow's `to_categorical` function to enable multi-class classification.

2.2.2 Model Architecture

A sequential CNN model was constructed using Keras, designed to extract hierarchical spatial features from the grayscale images. The architecture consists of the following layers:

- **Input Layer**: Accepts 200x200 grayscale images with a single channel.
- **Convolutional and Pooling Layers**:
  - Conv2D layer with 32 filters, 3x3 kernel, ReLU activation, followed by 2x2 MaxPooling.
  - Conv2D layer with 64 filters, 3x3 kernel, ReLU activation, followed by 2x2 MaxPooling.
  - Conv2D layer with 128 filters, 3x3 kernel, ReLU activation, followed by 2x2 MaxPooling.

- **Flattening Layer**: Converts 3D feature maps to a 1D feature vector.
- **Dense Layers**:
  - Fully connected layer with 128 units and ReLU activation.
  - Dropout layer with a dropout rate of 0.5 to prevent overfitting.
  - Output layer with 6 units and softmax activation for classification into six classes.

2.2.3 Data Augmentation

To improve model generalization and robustness, the training images were augmented using Keras' `ImageDataGenerator`. The following augmentations were applied:

- Random rotations up to 20 degrees.
- Horizontal flips.
- Width and height shifts up to 20% of the image size.
- Zooming up to 20%.

These augmentations were applied on-the-fly during training using the `flow()` method.

2.2.4 Training Configuration

The model was compiled and trained using the following configuration:

- **Optimizer**: Adam optimizer with default learning rate (0.001).
- **Loss Function**: Categorical cross-entropy.
- **Metrics**: Accuracy.
- **Batch Size**: 32.
- **Epochs**: 50.
- **Validation**: 20% of training data reserved for validation.

Model training progress was tracked using validation accuracy and loss. Early stopping was not explicitly applied, but could be added for future improvements.

2.2.5 Model Evaluation and Visualization

After training, the model was evaluated on the separate test set. The final test accuracy and loss were reported. Additionally, training and validation accuracy/loss were visualized across epochs to analyze model learning behaviors, as shown in Fig. 2:

2.2.6 Model Saving

The trained model was saved using Keras' `model.save()` function as `metal_surface_defect_model.h5` for future inference or deployment.

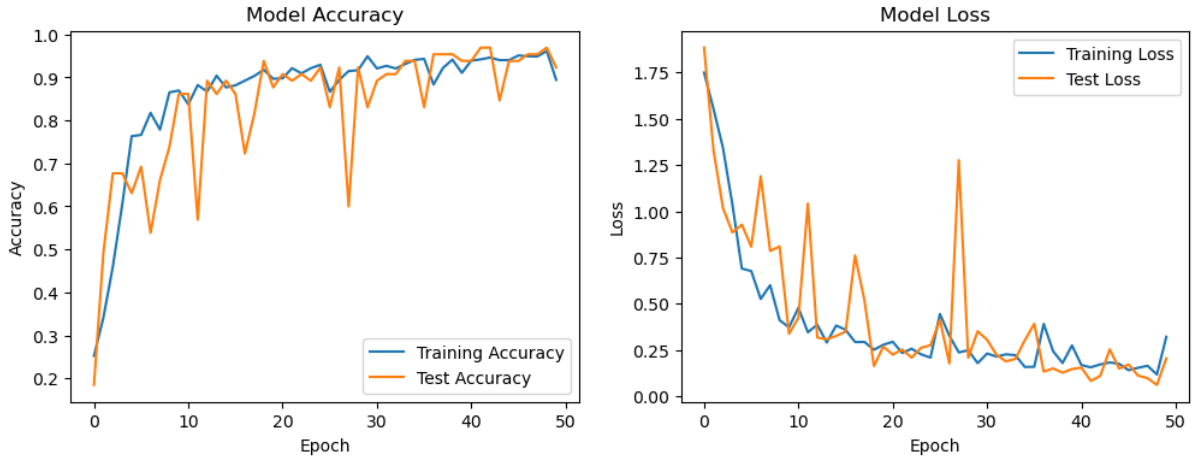


Figure 2: Accuracy over epochs for training and validation sets (left), and Loss over epochs for training and validation sets (right).

3 Results

3.1 Dataset Analysis

3.1.1 Image Statistics

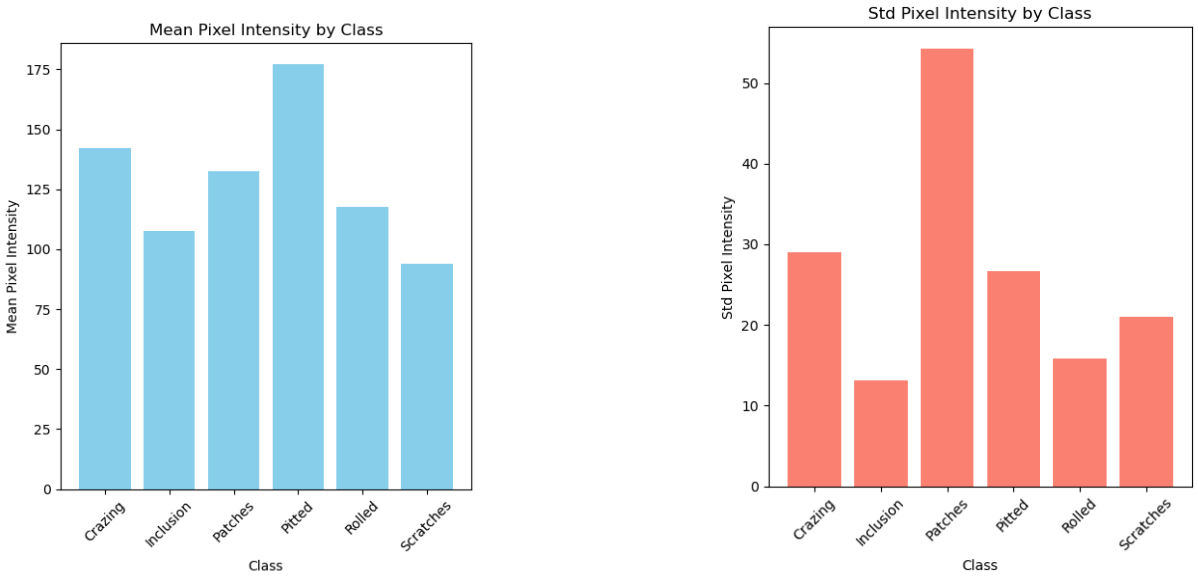
The `compute_image_stats` function produced a DataFrame of mean and standard deviation of pixel intensities for each class. Bar plots visualize, Fig. 3:

- **Mean Pixel Intensity**: Variations across classes suggest distinct brightness profiles, potentially useful for classification.
- **Standard Deviation**: Differences in pixel intensity variability indicate textural differences among defect types.

The image statistics reveal that the mean pixel intensity for the Pitted defect class is the highest among all classes. This can be attributed to the nature of pitted defects, which are characterized by numerous small black pores scattered across the surface. These pores create localized dark regions, but the majority of the surrounding pixels remain relatively bright, resulting in an overall increase in the average pixel intensity compared to other defect types.

Similarly, when examining the standard deviation of pixel intensity, the Patches defect class exhibits a value more than twice that of the other classes. This significant variation arises because patches defects typically cover a large portion of the image—often occupying 60-70% of the surface area. The extensive size of these defects leads to a wide range of pixel intensities within the image, particularly at the boundaries where the defect meets the normal surface. This boundary region causes a sharp contrast in intensity values, thereby increasing the standard deviation. The high variation reflects the complex texture and irregular shape of patch defects, distinguishing them clearly from other defect types with more uniform intensity distributions.

The Fig. 4 depicts a histogram titled "Pixel Intensity Distribution by Class (Training Set, Counts)," which illustrates the distribution of pixel intensities (ranging from 0 to 250) across six defect classes: Crazing, Inclusion, Patches, Pitted, Rolled, and Scratches. Each class is represented by a distinct color, with the y-axis showing the count of pixels



(a) Mean Pixel Intensity by Defect Class.

(b) Standard Deviation of Pixel Intensity by Defect Class.

Figure 3: Image Statistics of Dataset

and the x-axis representing pixel intensity values. The histogram reveals that the pixel intensity distributions vary across classes, with Patches showing a prominent peak around 50–100, suggesting a darker intensity profile, while Scratches exhibit a significant peak near 250, indicating a brighter profile. Other classes like Crazing, Inclusion, Pitted, and Rolled show more centralized distributions around 100–150, with overlapping patterns but varying amplitudes. This suggests that while there are shared intensity characteristics, each defect type has unique textural or brightness features.

This histogram is highly useful for defect detection tasks as it provides insights into the statistical properties of pixel intensities across different defect classes, which can inform preprocessing and feature engineering for machine learning models. For instance, the distinct peaks for Patches and Scratches could guide threshold-based segmentation or serve as input features for a Convolutional Neural Network (CNN) to improve classification accuracy. The overlapping distributions of other classes highlight the need for advanced techniques, such as texture analysis or data augmentation, to enhance model discrimination. Overall, this visualization aids in understanding dataset characteristics, optimizing model design, and identifying potential challenges in distinguishing similar defect types.

3.1.2 Image Size and Format

The `check_image_sizes` function confirmed:

- All images are 200x200 pixels, grayscale, with `uint8` data type, as shown in Fig. 5.
- No loading errors or inconsistencies were found, ensuring dataset uniformity.

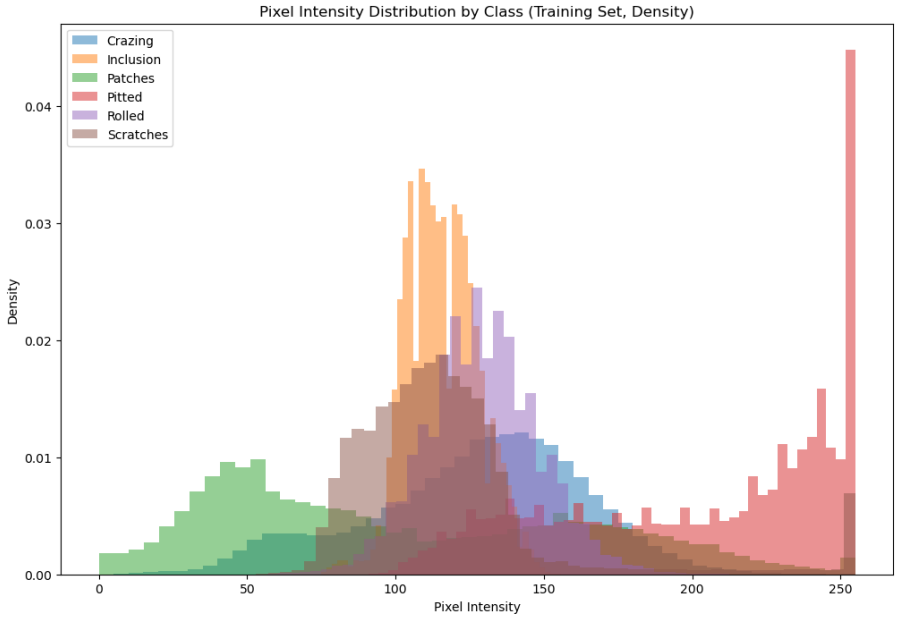


Figure 4: Pixel Intensity Distribution by Class (Training Set, Counts)

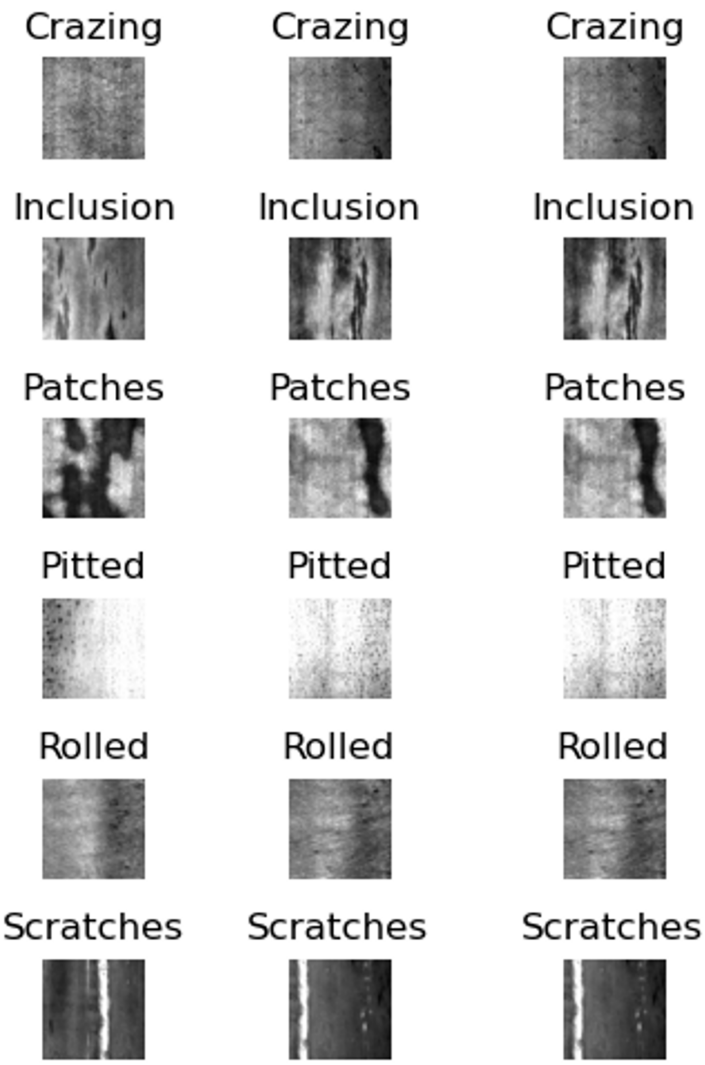


Figure 5: Image samples

3.2 CNN Model Performance

The CNN model was evaluated using a separate test set comprising 65 images (approximately 20% of the total dataset). The classification report provides a detailed breakdown of precision, recall, and F1-score for each defect class.

Table 1: Classification Report for Test Set

Defect Class	Precision	Recall	F1-Score	Support
Crazing	0.71	1.00	0.83	5
Inclusion	0.86	1.00	0.92	12
Patches	1.00	0.92	0.96	12
Pitted	1.00	0.67	0.80	12
Rolled	0.92	1.00	0.96	12
Scratches	1.00	1.00	1.00	12
Accuracy	92.3% (60 out of 65 images correctly classified)			
Macro Avg	0.92	0.93	0.91	65
Weighted Avg	0.94	0.92	0.92	65

The CNN model achieved an overall accuracy of **92.3%** on the test set, correctly classifying 60 out of 65 images (i.e., $\frac{60}{65} \approx 0.923$). The weighted F1-score of **0.92** further demonstrates the model's strong and balanced performance across all defect classes.

This accuracy estimate is based on the following class-wise breakdown of correct predictions, derived from recall values and class support:

- **Crazing**: 5/5 (Recall = 1.00)
- **Inclusion**: 12/12 (Recall = 1.00)
- **Patches**: $0.92 \times 12 \approx 11$
- **Pitted**: $0.67 \times 12 \approx 8$
- **Rolled**: 12/12 (Recall = 1.00)
- **Scratches**: 12/12 (Recall = 1.00)

Total correct predictions: $5 + 12 + 11 + 8 + 12 + 12 = 60$

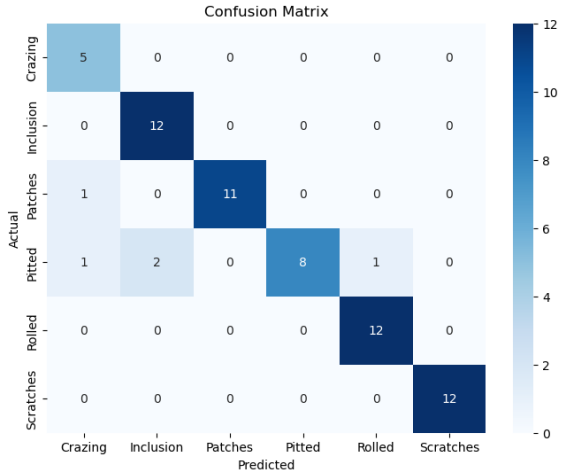


Figure 6: Confusion martix

Notable observations:

- **Scratches** and **Rolled** defects were perfectly classified, achieving an F1-score of 1.00 and 0.96 respectively.
- **Crazing** achieved full recall (1.00) but lower precision (0.71), indicating some false positives—likely due to visual similarity with linear defects like Scratches.
- **Pitted** defects had a relatively lower recall (0.67), suggesting the model occasionally misclassified these samples, possibly due to their small, dispersed nature.

Overall, the CNN model demonstrates high classification accuracy and robustness across all defect categories, with minor room for improvement in distinguishing between visually similar classes.

4 Conclusion

This study presents a comprehensive approach to automated surface defect detection using convolutional neural networks (CNNs), based on the NEU Surface Defect Database. The project began with a detailed analysis of the dataset, confirming consistent image formats (grayscale, 200×200 pixels, uint8) and revealing class-wise statistical differences in pixel intensity distributions. Notably, Pitted defects exhibited the highest mean pixel intensity, while Patches showed the greatest standard deviation—insights that suggest their unique visual characteristics could aid classification.

A CNN model was designed and trained on this dataset, achieving an overall test accuracy of **92.3%** (60 out of 65 samples correctly classified). The model demonstrated strong performance across most defect types, with a weighted F1-score of **0.92**, indicating a balanced trade-off between precision and recall. Perfect recall was achieved in several classes (e.g., Scratches, Rolled), while minor confusion was observed between visually similar classes like Crazing and Scratches.

These results validate the effectiveness of CNNs for visual surface inspection tasks in industrial contexts. However, the model's performance on classes with subtle or overlapping features—such as Pitted and Crazing—suggests that additional feature extraction techniques, like texture analysis or frequency-domain transforms, may further enhance accuracy.

Future work will focus on:

- Improving the CNN architecture with deeper models or transfer learning (e.g., ResNet, EfficientNet).
- Exploring advanced preprocessing techniques such as edge detection and contrast enhancement.
- Expanding the dataset for better generalization and robustness across real-world variability.

Overall, this project provides a solid foundation for developing reliable, automated defect detection systems in manufacturing and quality control pipelines.

Disclaimer

The dataset used in this project is solely for self-learning and educational purposes. It is not intended for any commercial use or product deployment. All experiments and results presented are for academic and non-commercial demonstration only.

Acknowledgment

The dataset used for this project, **NEU Surface Defect Database**, was sourced from the publicly available NEU-DET dataset published by the Northeastern University (NEU), China. The authors gratefully acknowledge the availability of this resource, which enabled the training and evaluation of the surface defect classification model.

References

[1] Matplotlib Development Team. *Matplotlib Documentation*, 2024. `https://matplotlib.org/stable/contents.html`.

[2] NumPy Developers. *NumPy Documentation*, 2024. `https://numpy.org/doc/`.

[3] OpenCV.org. *OpenCV Documentation*, 2024. `https://docs.opencv.org/`.

[4] TensorFlow Developers. *TensorFlow Documentation*, 2024. `https://www.tensorflow.org/`.

[5] The Pandas Development Team. *Pandas Documentation*, 2024. `https://pandas.pydata.org/docs/`.

[6] Qike Wu. Neu-det. `https://ieee-dataport.org/documents/neu-det`, 2024. Accessed 2025-08-11.

Appendix

The trained CNN model was tested on unseen images from the test dataset. Below is an example of a test image along with its predicted label, demonstrating the model's ability to correctly classify surface defects.

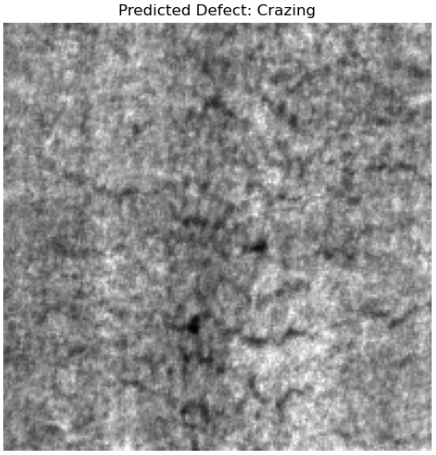


Figure 7: Example prediction made by the CNN model. The image belongs to the *Crazing* class and was correctly classified with high confidence.