PARAMETER
JURNAL MATEMATIKA, STATISTIKA DAN TERAPANNYA

e-ISSN. 2829-6303
Vol. 05, No. 2, September 2026
Pages 467-486

Steel Defect Monitoring Through Convolutional Neural Network Image Segmentation and Neural Network Classification Integrated with U-Control Chart

Inas Wulan Ramadhani[1], Wibawati[1*], Muhammad Ahsan[1]

[123]Department of Statistics, Faculty of Science and Data Analytics, Institut Teknologi Sepuluh Nopember
Kampus ITS Sukolilo, Surabaya, 60111, East Java, Indonesia

E-mail Correspondence Author: ******wibawati@its.ac.id*

Abstract

Automated visual inspection can detect steel surface defects, but image-level detection alone does not indicate whether the production process is statistically stable. This study proposes an image-based monitoring procedure that integrates deep-learning-based defect detection with a U-control chart. Using the Severstal steel defect dataset, run-length-encoded (RLE) annotations were reconstructed into binary masks, and the steel surface images were resized and normalized. U-Net segmentation was used to localize defective regions, while a MobileNetV2-based model supported multi-label defect classification. The predicted masks were thresholded, and external contour detection was applied to count distinct defect regions in each image. The image-level counts were aggregated into subgroups of 50 images and monitored using initial Phase I, revised Phase I, and Phase II U-chart analyses. The initial Phase I chart signaled one out-of-control subgroup; after this subgroup was screened, the revised Phase I subgroups fell within the control limits, and the Phase II subgroups generally remained within the revised limits. The results show that the proposed workflow can transform image segmentation outputs into defect-per-unit statistics and interpretable monitoring signals, linking automated visual inspection with statistical process control.

Keywords: *MobileNetV2, statistical process control; steel defect detection; U-control chart; U-Net segmentation.*

: https://doi.org/10.30598/parameterv5i3pp467-486

This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution-ShareAlike 4.0 International License.

Copyright © 2026 The Author(s).
Research Ariticle ***Open Acces***
Email: jurnalparameter@gmail.com
Homepage: https://ojs3.unpatti.ac.id/index.php/parameter

Submitted: June 2026
Accepted: September 2026

1. INTRODUCTION

Image-based process monitoring has become increasingly important in modern manufacturing because visual information can be used to detect product defects and evaluate process behavior. Industrial vision systems automatically capture product surfaces and transform them into measurable quality information, reducing the dependence on manual inspection and supporting objective quality assessment [1], [2]. In smart manufacturing, this capability is particularly useful because image-based inspection can support automated and data-driven decision-making [3].

Steel surface inspection is a manufacturing application that strongly benefits from automated image analysis. Surface defects, such as scratches, inclusions, holes, patches, and other irregularities, may reduce product quality and affect downstream production processes [4]. Deep learning methods, particularly convolutional neural networks (CNNs), have been widely used for steel surface defect detection because they can learn complex visual patterns directly from image data [5], [6], [7], [8]. A recent review by Chen [9] also emphasizes that deep learning-based approaches have become dominant in steel surface defect detection due to their ability to improve feature extraction, classification, and localization performance compared with traditional visual inspection methods. In this context, image segmentation is useful because it separates defective regions from normal surface areas and provides information about defect location, shape, and area [10], [11]. U-Net is a widely used segmentation architecture because its encoder–decoder structure combines contextual information with pixel-level localization [12]. Convolutional neural networks are also commonly used for image classification because they can learn hierarchical image features through convolutional and nonlinear transformation layers [13].

However, image-level defect detection alone does not directly indicate whether the production process is statistically stable. Previous studies have shown that image data can be incorporated into statistical process control by converting visual information into monitoring statistics and by developing image-monitoring frameworks based on control charts for detecting surface-related process shifts in manufacturing systems [14], [15], [16], [17]. Beyond image-based applications, control-chart-based monitoring has also been applied to anomaly detection in complex data environments, such as intrusion detection using a PCA-based Hotelling's T^2 control chart [18]. Attribute control charts can be used when inspection outputs are expressed as counts, categories, or defect-related measures [19], [20], [21], [22]. Therefore, when the number of defects is obtained from each steel surface image or subgroup, the information can be expressed as defects per inspected unit and monitored using a U-control chart.

This method was developed to bridge the gap between deep learning-based steel surface defect detection and statistical process monitoring. The proposed workflow reconstructs RLE annotations into binary masks, uses U-Net segmentation to localize defect regions, applies neural network classification to support defect-type interpretation, converts segmented regions into defect counts through contour detection, and monitors the resulting defect-per-unit statistics using Phase I and Phase II U-chart procedures. The contributions of this study are threefold. First, it provides a reproducible workflow for converting steel surface images and RLE annotations into image-derived defect-count data. Second, it integrates segmentation, classification, and U-chart procedures for monitoring steel surface defects. Third, it applies a Phase I–Phase II monitoring framework for identifying unusual changes in defect-per-unit behavior.

2. METHOD

The method customizes convolutional neural network-based image classification [13], U-Net image segmentation [12], and U-control chart for defect-per-unit monitoring [21], [23]. The overall workflow of the method, from steel surface images and RLE annotations to out-of-control signals, is summarized in Figure 1.

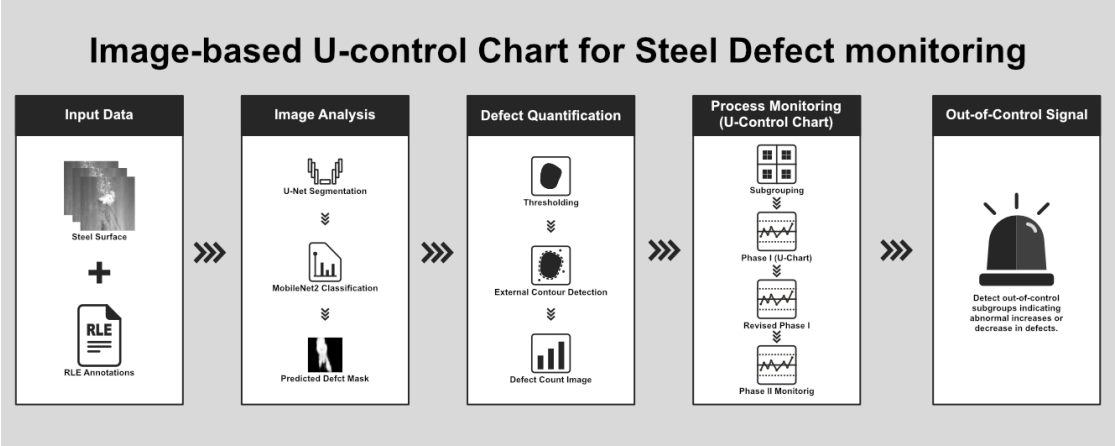


Figure 1. Workflow of the image-based U-control chart for steel surface defect monitoring

2.1. Data sources and inspection structure

The demonstrated application uses the Severstal: Steel Defect Detection dataset, which contains steel surface images and RLE annotations for four defect classes [24]. Each image represents an inspected steel surface, and each annotation contains an image identity, a defect class identity, and an RLE-encoded defect region. One image may contain no defect, one defect class, or more than one defect class.

The image identity links each annotation to the corresponding steel surface image. The class identity is used to construct a four-dimensional multi-hot target vector for the classification task, while the RLE annotation is decoded into a binary mask for segmentation and defect counting. This structure allows the same data source to support both pixel-level segmentation and image-level multi-label classification. For monitoring, each steel surface image is treated as one inspected unit. Image-level defect counts are arranged in a reproducible sequence and divided into Phase I and Phase II data. In the implementation, 80% of the ordered images are assigned to Phase I and 20% to Phase II. Each U-control chart subgroup contains 50 inspected images. Because the Severstal dataset does not include actual production timestamps, the sequence should be interpreted as a reproducible demonstration sequence rather than the true chronological manufacturing order.

2.2. Annotation reconstruction from Run-Length-Encoding (RLE) to binary masks

The original Severstal annotations are provided in RLE format. RLE represents the locations of defective pixels by storing the starting pixel position and the length of consecutive defect pixels. Before model training and defect counting, each RLE annotation is decoded into a binary mask at the original annotation size of 256 × 1600 pixels. In the reconstructed mask, pixels belonging to defective regions are assigned a value of 1, while background pixels are assigned a value of 0. The decoding follows the column-major ordering rule used in the Severstal annotation format. If an image contains more than one defect class, class-specific masks are reconstructed separately and then combined using a logical OR operation to form one union defect mask.

The union mask is used as the target mask for binary segmentation and as the spatial basis for defect counting. This stage is essential because the U-control chart requires count data, not encoded pixel strings.

2.3. Image preprocessing

Each steel surface image is read in RGB format, resized to 128 × 128 pixels, and normalized to the range [0,1] by dividing pixel intensity values by 255. The reconstructed binary masks are resized to the same 128 × 128 dimension using nearest-neighbor interpolation so that binary labels are preserved. The same preprocessing procedure is applied to the training, validation, Phase I, and Phase II images. This consistency is necessary because changing the preprocessing rule may change the predicted mask and therefore alter the defect count used in the U-control chart. Additional image enhancement, such as histogram equalization, contrast stretching, median filtering, or Gaussian filtering, may be added in industrial applications when illumination variation or sensor noise is severe **[25], [26]**.

2.4. Defect segmentation using U-Net

The preprocessed images and reconstructed union masks are used to train a U-Net segmentation model. U-Net is an encoder-decoder segmentation architecture. The encoder captures contextual information through successive convolution and pooling operations, while the decoder reconstructs spatial resolution through upsampling or transposed convolution. Skip connections transfer features from the encoder to the decoder so that localization details are preserved **[12]**. In this method, the input image shape is 128 × 128 × 3 and the target mask shape is 128 × 128 × 1. The encoder contains three convolutional blocks with 16, 32, and 64 filters. Each block contains convolution, batch normalization, ReLU activation, and max-pooling. The bottleneck uses 128 filters. The decoder uses transposed convolution, skip connections, convolution, batch normalization, and ReLU activation to reconstruct the mask.

The output layer is a 1 × 1 convolution with a sigmoid activation that produces a pixel-level defect probability map. The probability map is converted into a binary mask using a segmentation threshold of 0.50. Pixels with predicted probabilities greater than or equal to 0.50 are treated as defect pixels. Segmentation is required because classification alone does not provide the location, area, or shape of the defect. The binary segmentation mask produced by U-Net can be converted into quantitative information such as defect area, bounding-box location, and the number of separated defect regions **[10], [27], [28]**. These quantities are necessary before image outputs can be used in a control chart.

2.5. Defect classification using MobileNetV2

The classification task is formulated as a multi-label problem. A four-dimensional target vector is constructed for each image, where each element indicates the presence or absence of one defect class. MobileNetV2 pretrained on ImageNet is used as the feature extraction backbone because it provides an efficient CNN architecture based on inverted residuals and linear bottlenecks **[29]**. Convolutional neural networks learn feature representations by applying trainable kernels to images. A typical CNN contains convolutional layers for local feature extraction, activation layers for nonlinear transformation, pooling layers for spatial reduction, and fully connected or dense layers for classification **[6], [7], [13]**. This structure enables the network to represent low-level features, such as edges and textures, and higher-level patterns related to defect

morphology [27]. The general architecture of a CNN for image classification is illustrated in Figure 2.

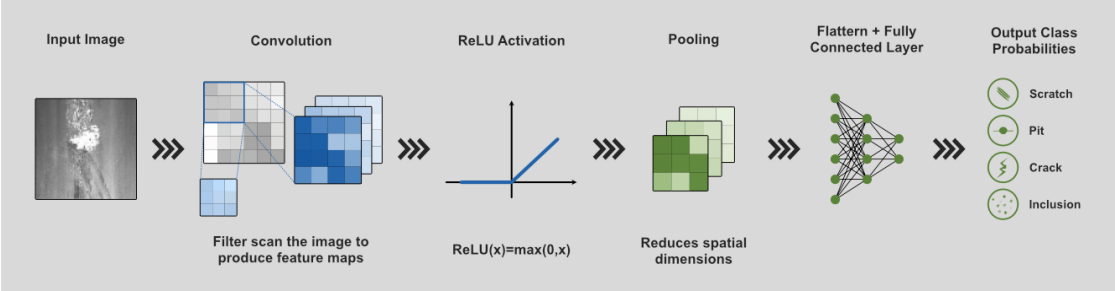


Figure 2. Convolutional Neural Network Architecture

For an input feature map x and a convolution kernel k, the convolutional output at a location can be written as

$$y = sum(x \times k) + b, \quad (1)$$

where b is the bias term. After convolution, the Rectified Linear Unit is commonly applied as

$$ReLU(x) = max(0, x). \quad (2)$$

ReLU supports efficient neural-network training by preserving positive activations and suppressing negative responses [30]. Pooling layers reduce feature-map size, while dense layers integrate learned features into class-level predictions [13]. In this method, the original top layer of MobileNetV2 is removed. The classification head consists of global average pooling, a dense layer with 256 neurons and ReLU activation, a dropout layer with a rate of 0.30, and a four-neuron sigmoid output layer. The sigmoid output allows more than one defect class to be predicted for a single image. A classification threshold of 0.50 converts probabilities into binary class labels. The classification output supports defect-type interpretation, but it is not used directly to calculate the U-control chart statistic. The U-control chart statistic is calculated from segmentation-derived defect counts. Therefore, classification errors may affect defect-type interpretation, but they do not directly determine whether a subgroup is classified as out of control.

2.6. Model training configuration

The segmentation and classification models are trained using image-level training and validation sets split by ImageId. This prevents the same image from appearing in both training and validation data. The split is 80% training and 20% validation. The U-Net model is compiled using binary cross-entropy loss, the Adam optimizer, and metrics consisting of accuracy, Dice coefficient, and Intersection over Union. It is trained for a maximum of 30 epochs with a batch size of 16. ReduceLROnPlateau and EarlyStopping callbacks monitor validation loss, reduce the learning rate when improvement stops, and restore the best model weights. Adam is used because it is an adaptive stochastic optimization method [31]. The MobileNetV2 classification model is trained in two stages. During the first stage, the backbone is frozen and only the classification head is trained for a maximum of 20 epochs with a batch size of 64. Data augmentation includes rotation, width shift, height shift, zooming, horizontal flipping, and nearest-neighbor filling. During the second stage, the last 30 MobileNetV2 layers are unfrozen and fine-tuned for a maximum of 10 epochs using Adam with a learning rate of 1×10^{-5}. Dropout and data augmentation are included to reduce overfitting and improve generalization [32], [33]. The architecture and hyperparameter settings of both models are summarized in Table 1.

Table 1. Architecture and hyper-parameters of the proposed image-based U-control chart method

Component	Hyperparameter/Setting
Input image	$128 \times 128 \times 3$; normalized to [0,1]
Original mask	RLE mask size = 256×1600
Data split	80% training; 20% validation; split by ImageId
Segmentation model	U-Net
U-Net encoder filters	16, 32, 64
U-Net bottleneck	128 filters
U-Net output	1×1 Conv2D; sigmoid activation
Segmentation training	Binary cross-entropy; Adam; batch size 16; max 30 epochs
Classification model	MobileNetV2 with ImageNet weights
Classification head	GlobalAveragePooling2D; Dense 256; Dropout 0.30; Dense 4 sigmoid
Classification training	Binary cross-entropy; Adam; batch size 64
Fine-tuning	Last 30 layers unfrozen; max 10 epochs
Data augmentation	Rotation, shift, zoom, and horizontal flip
Thresholds	Segmentation = 0.50; classification = 0.50
Defect counting	OpenCV external contour detection; minimum area = 50 pixels

2.7. Defect counting using connected-component labelling

After the U-Net model generates a predicted probability mask, the mask is thresholded at 0.50 to obtain a binary defect mask. External contour detection is then applied to identify distinct defect regions. The implementation uses OpenCV findContours with RETR_EXTERNAL and CHAIN_APPROX_SIMPLE. Each detected external contour is treated as a candidate defect region. Small fragments with area smaller than 50 pixels are removed to reduce false counts from isolated pixels or small false-positive regions. Each remaining contour is counted as one detected defect region. For each detected defect region, the implementation records the defect identification number, confidence score, defect area in pixels, bounding-box coordinates, bounding-box area, and number of polygon points. The confidence score is the mean predicted probability inside the bounding-box region. The main variable used for monitoring is the total number of detected defect regions per image.

2.8. U-control chart construction and phase-based monitoring

Image-level defect counts are converted into subgroup-level defect-per-unit statistics. The U-control chart is used when the monitored quality characteristic is the number of nonconformities per inspected unit. For subgroup i, c_i is the total number of detected defects and n_i is the number of inspected images. The defect-per-unit statistic is calculated as

$$u_i = \frac{c_i}{n_i}. \tag{3}$$

The center line is estimated from Phase I data as

$$\bar{u} = \frac{\sum c_i}{\sum n_i}. \tag{4}$$

Assuming a Poisson count structure, the subgroup-specific control limits are

$$UCL_i = \bar{u} + 3\sqrt{\frac{\bar{u}}{n_i}}, \tag{5}$$

$$LCL_i = max\left(0, \bar{u} - 3\sqrt{\frac{\bar{u}}{n_i}}\right). \tag{6}$$

A subgroup is classified as out of control when $u_i > UCL_i$ or $u_i < LCL_i$. Points above the UCL indicate an unusual increase in detected defects per unit, while points below the LCL indicate process behavior that differs from the established baseline even if the defect count is lower [21], [23]. Phase I subgroups are used to estimate the initial center line and control limits. If Phase I contains out-of-control subgroups, these subgroups are investigated before revision. Removal is justified only when an assignable cause or an unsuitable baseline subgroup is identified. After removal, the revised Phase I center line and control limits are recalculated using the retained Phase I subgroups. Phase II observations are monitored using the revised Phase I limits. The Phase II data are not used to estimate new limits. When a Phase II point exceeds the revised UCL or falls below the revised LCL, the corresponding subgroup should be reviewed by examining the original images, predicted masks, detected components, confidence scores, and classification outputs.

2.9. Evaluation metrics

The classification model is evaluated using class-wise confusion matrices and metrics derived from true positive, true negative, false positive, and false negative outcomes. The main metrics are precision, recall, F1-score, and accuracy, defined as

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{7}$$

$$\text{Recall} = \frac{TP}{TP + FN}, \tag{8}$$

$$\text{F1-score} = \frac{2(\text{precision} \times \text{recall})}{\text{precision} + \text{recall}}, \tag{9}$$

and

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{10}$$

Area under the receiver operating characteristic curve is used to summarize discrimination performance across thresholds [34]. For segmentation, validation loss, pixel accuracy, Dice coefficient, and Intersection over Union are used because defect pixels may occupy a small proportion of the steel surface image. Dice and IoU evaluate overlap between the true and predicted masks and are therefore more informative than pixel accuracy alone for sparse defect regions [27].

2.10. Software environment

The implementation was developed using the Python programming language. TensorFlow/Keras was used to construct and train the U-Net segmentation and MobileNetV2-based classification models. OpenCV was used for image reading, resizing, and the detection of external contours. NumPy and Pandas were used for processing numerical and tabular data. Scikit-learn was used for training-validation splitting and classification evaluation, and Matplotlib was used for visualization.

The reproducible implementation requires train.csv, train_images, and test_images, when available. The generated outputs included reconstructed masks, predicted masks, classification summaries, detected component tables, image-level defect-count tables, subgroup U-chart tables, trained model files, and pseudocode documentation.

2.11. Analysis steps

The analysis procedure consists of the following steps, and the complete procedure is summarized as pseudocode in **Algorithm 1**.

1. Data preparation: The Severstal steel surface images and RLE annotation file are loaded and matched using ImageId.
2. Annotation reconstruction: The RLE annotations are decoded into binary defect masks and combined into a union mask when one image contains multiple defect classes.
3. Image preprocessing: The steel surface images and reconstructed masks are resized to 128 × 128 pixels, and the image pixel values are normalized to the range [0,1].
4. Classification target construction: The defect class labels are converted into four-dimensional multi-hot vectors for multi-label classification.
5. Training-validation split: The image data are divided into training and validation sets using an 80%–20% split by ImageId.
6. U-Net segmentation modeling: The U-Net model is trained to predict pixel-level defect probability masks from preprocessed steel surface images.
7. MobileNetV2 classification modeling: The MobileNetV2-based model is trained to predict the presence or absence of each defect class in a multi-label setting.
8. Defect region extraction: The predicted segmentation masks are thresholded and processed using external contour detection to identify distinct defect regions.
9. Defect counting: Each detected contour that satisfies the minimum area threshold is counted as one defect region for the corresponding image.
10. Subgroup formation: Image-level defect counts are arranged into sequential subgroups with 50 inspected images per subgroup.
11. Initial Phase I U-control chart: Phase I subgroups are used to estimate the initial center line and control limits for defect-per-unit monitoring.
12. Revised Phase I U-control chart: Out-of-control Phase I subgroups are investigated and, when justified, removed before recalculating the revised control limits.
13. Phase II monitoring: Phase II subgroups are monitored using the revised Phase I control limits to identify unusual defect-per-unit behavior.
14. Method validation: The workflow is validated using segmentation outputs, classification metrics, detected-component results, and U-control chart signals.

Algorithm 1. Pseudocode of the image-based U-control chart procedure for steel surface defect monitoring

```
Input:
I_j ← steel surface image for j = 1, 2, ..., N
R_j ← run-length encoded defect annotation for image j
C_j ← defect class label for image j
θ_s ← segmentation threshold, θ_s = 0.50
A_min ← minimum component area threshold, A_min = 50 pixels
m ← subgroup size for U-control chart monitoring, m = 50 images
Phase_I_range ← 80% of the ordered image sequence
Phase_II_range ← 20% of the ordered image sequence
```

Output:
Predicted defect masks
Predicted defect class labels
Image-level defect counts
Subgroup defect-per-unit statistics
Out-of-control signals from the U-control chart

Procedure:
Step 1: Load Dataset
Load steel surface images and the annotation file.
Extract ImageId, ClassId, and run-length encoded annotation from the dataset.
Step 2: Reconstruct RLE Annotation
For each image j:
    Decode R_j into a binary defect mask with the original size of 256 × 1600 pixels.
    If image j contains more than one defect class:
        Reconstruct each class-specific mask separately.
        Combine all class-specific masks using a logical OR operation.
    End If
End For
Step 3: Preprocess Images and Masks
For each image j:
    Read the steel surface image in RGB format.
    Resize the image to 128 × 128 pixels.
    Normalize pixel intensity values into the range [0,1].
    Resize the reconstructed binary mask to 128 × 128 pixels using nearest-neighbor interpolation.
End For
Step 4: Construct Multi-label Classification Target
For each image j:
    Convert the presence or absence of each defect class into a four-dimensional multi-hot vector.
End For
Step 5: Split Training and Validation Data
Split the image data into training and validation sets by ImageId.
Use 80% of the images for training and 20% for validation.
Step 6: Train U-Net Segmentation Model
Use preprocessed images as inputs.
Use reconstructed union binary masks as target outputs.
Train the U-Net model using binary cross-entropy loss and the Adam optimizer.
Evaluate the segmentation model using validation loss, pixel accuracy, Dice coefficient, and Intersection over Union.
Step 7: Train MobileNetV2 Classification Model
Use preprocessed images as inputs.
Use multi-hot defect class vectors as target outputs.
Train the MobileNetV2-based classification model in two stages:
    Stage 1: Freeze the backbone and train the classification head.
    Stage 2: Unfreeze the last 30 layers and fine-tune the model.
Evaluate the classification model using validation loss, accuracy, AUC, classification report, and class-wise confusion matrices.
Step 8: Generate Predicted Defect Masks
For each image j:
    Use the trained U-Net model to generate a predicted defect probability mask.
    Convert the probability mask into a binary defect mask using $\theta_s = 0.50$.
End For
Step 9: Detect and Count Defect Regions
For each predicted binary mask:
    Apply external contour detection using OpenCV findContours.
    Remove detected contours with area smaller than $A_{min} = 50$ pixels.
    Count each remaining external contour as one detected defect region.
    Record the image-level defect count c_j.
End For
Step 10: Generate Predicted Defect Class Labels
For each image j:
    Use the trained MobileNetV2 model to generate class probabilities.
    Convert class probabilities into binary class labels using a classification threshold of 0.50.
End For

```
Step 11: Construct Phase I and Phase II Monitoring Data
Arrange the image-level defect counts into a reproducible monitoring sequence.
Assign 80% of the ordered sequence to Phase I.
Assign 20% of the ordered sequence to Phase II.
Step 12: Form U-control Chart Subgroups
Arrange image-level defect counts into sequential subgroups with subgroup size m =
50.
For each subgroup i:
    Calculate c_i as the total number of detected defect regions.
    Calculate n_i as the number of inspected images.
    Calculate the defect-per-unit statistic:
        u_i = c_i / n_i
End For
Step 13: Construct Initial Phase I U-control Chart
Use Phase I subgroups to estimate the center line:
    ū = Σc_i / Σn_i
For each Phase I subgroup i:
    Calculate the control limits:
        UCL_i = ū + 3√(ū / n_i)
        LCL_i = max(0, ū - 3√(ū / n_i))
    If u_i > UCL_i or u_i < LCL_i:
        Flag subgroup i as out of control.
    End If
End For
Step 14: Revise Phase I U-control Chart
Investigate out-of-control Phase I subgroups.
If assignable causes or unsuitable baseline subgroups are identified:
    Remove the corresponding Phase I subgroups.
    Recalculate the revised center line and revised control limits using the
retained Phase I subgroups.
End If
Step 15: Monitor Phase II Subgroups
For each Phase II subgroup i:
    Apply the revised Phase I center line and control limits.
    If u_i > UCL_i,rev or u_i < LCL_i,rev:
        Signal subgroup i as out of control.
        Review the corresponding original images, predicted masks, detected
components, confidence scores, and classification outputs.
    End If
End For
End Procedure.
```

3. RESULTS AND DISCUSSION

The method was validated by checking whether each workflow stage produced outputs consistent with its function: annotation reconstruction, segmentation, classification, defect-count extraction, and U-chart monitoring. This staged validation is necessary because the final monitoring statistic depends on the consistency of all previous image-processing and model-prediction steps.

3.1. RLE mask reconstruction validation

The first validation step was performed by reconstructing the run-length encoded annotations into binary masks and comparing them with the corresponding steel surface images. This step confirmed that the encoded defect locations were correctly converted into spatial defect regions. The reconstructed masks were then resized to match the input dimension of the segmentation model and used as target outputs for U-Net training. This validation is important because any error in RLE decoding would shift the defect location or distort the defect shape. Such an error would affect segmentation training and, subsequently, the defect-count statistics used in the U-control chart.

3.2. Segmentation validation

The segmentation model was validated by comparing the original image, true mask, predicted probability mask, and binary predicted mask. This visual comparison was used to examine whether the U-Net model could localize defective regions on the steel surface. **Figure 3** shows that the predicted mask generally follows the main defective region in the true mask, indicating that the model can identify the spatial location of the defect.

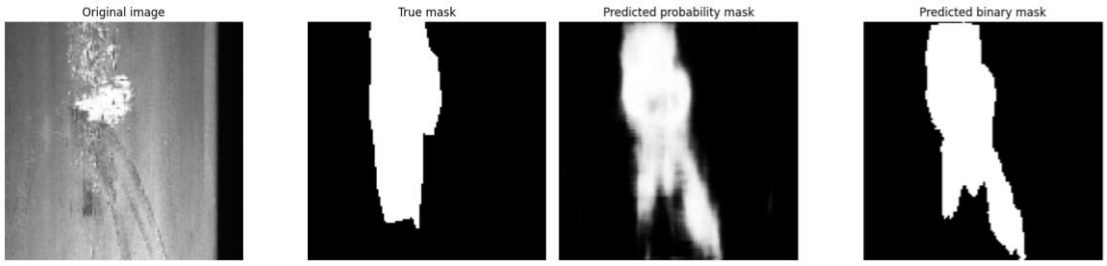


Figure 3. Original image, True Mask, Predicted Mask, and Predicted Binary Mask

The segmentation training curves were also examined using validation loss, pixel accuracy, Dice coefficient, and Intersection over Union. These metrics were used because defect pixels occupy only a small part of the steel surface image. Therefore, overlap-based metrics such as Dice coefficient and IoU are more informative than pixel accuracy alone. These training curves are presented in **Figure 4**.

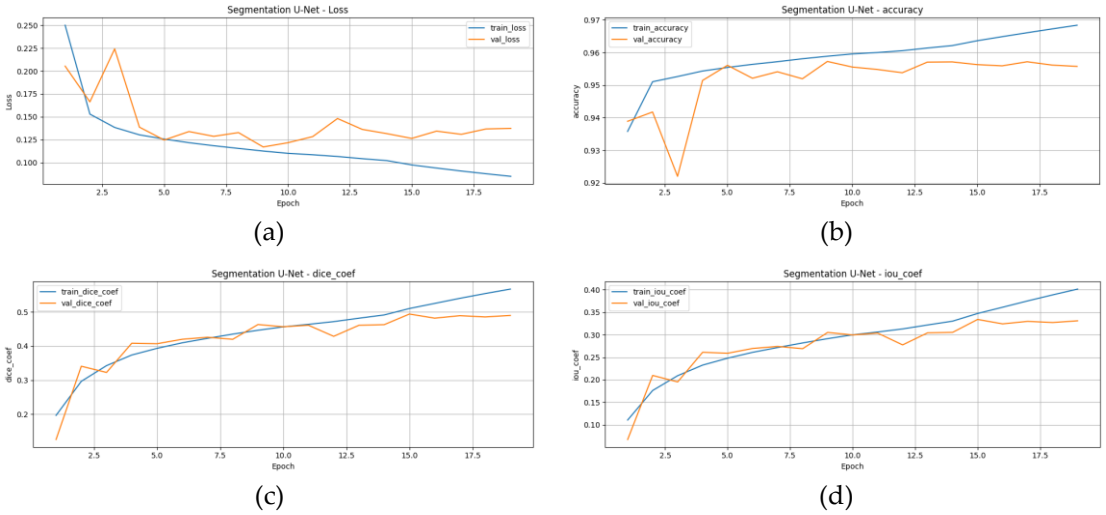


Figure 4. Segmentation training curves: (a) loss, (b) accuracy, (c) Dice coefficient, and (d) IoU

Although the model was able to detect the main defect area, several practical segmentation issues were observed. Under-segmentation may cause some defective pixels to be missed, while small false-positive fragments may appear as additional disconnected regions. These errors are important for this method because they can directly reduce or increase the number of detected defect regions and consequently affect the U-chart statistic.

3.3. Classification validation

The MobileNetV2-based classification model was validated using training curves, accuracy, AUC, classification reports, and class-wise confusion matrices. The classification model produced four output probabilities corresponding to the four defect classes. The class-wise confusion matrices were used to evaluate whether the model

could support defect-type interpretation for each class. The training curves for the frozen-backbone and fine-tuning stages are shown in **Figure 5**.

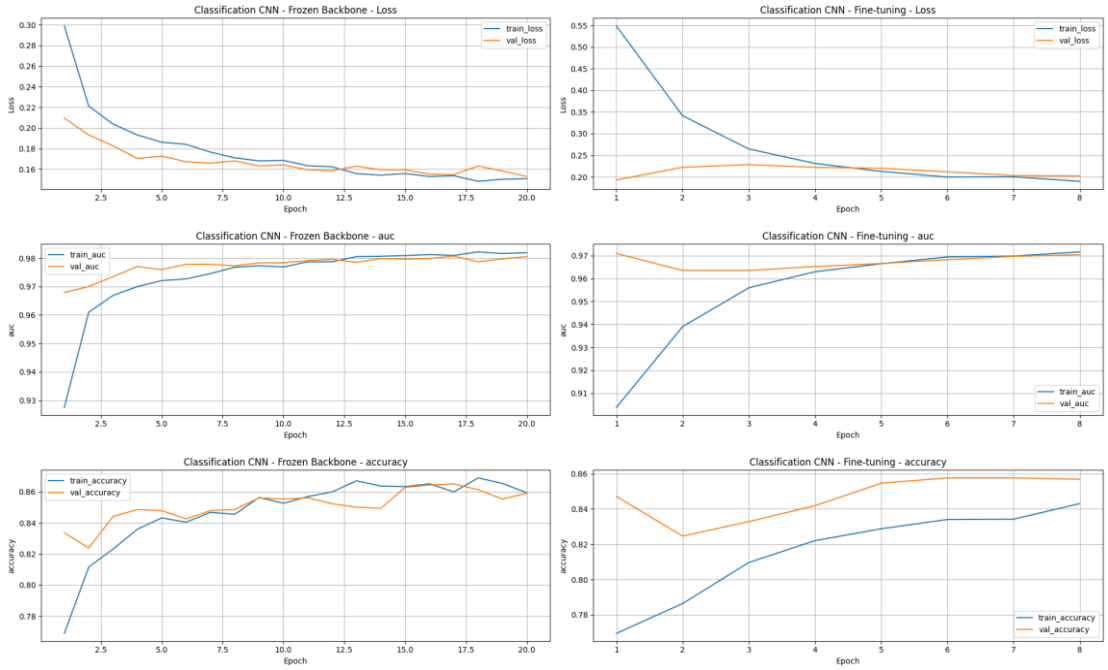


Figure 5. Classification training curves for frozen-backbone and fine-tuning stages

The validation results showed that classification performance varied across defect classes. This variation is expected in a multi-label steel defect dataset because the defect classes are not equally represented. Some classes are easier to identify because they appear more frequently or have clearer visual patterns, while minority classes tend to produce more false negatives. The class-wise confusion matrices are shown in **Figure 6**, while **Figure 7** shows that Class 3 accounts for the largest proportion of images in both Phase I and Phase II.

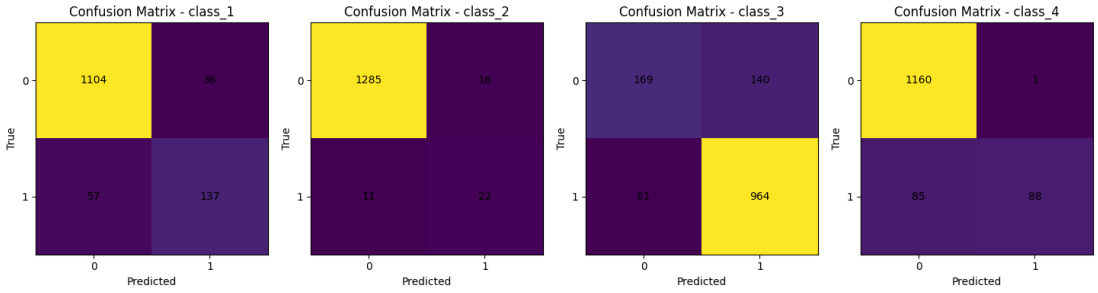


Figure 6. Class-wise confusion matrices for the multi-label classification model

In this method, the classification output is used to support interpretation of defect type, not to calculate the U-chart statistic directly. The U-chart statistic is calculated from segmentation-derived defect counts. Therefore, classification errors may affect defect-type interpretation, but they do not directly determine whether a subgroup is classified as out of control.

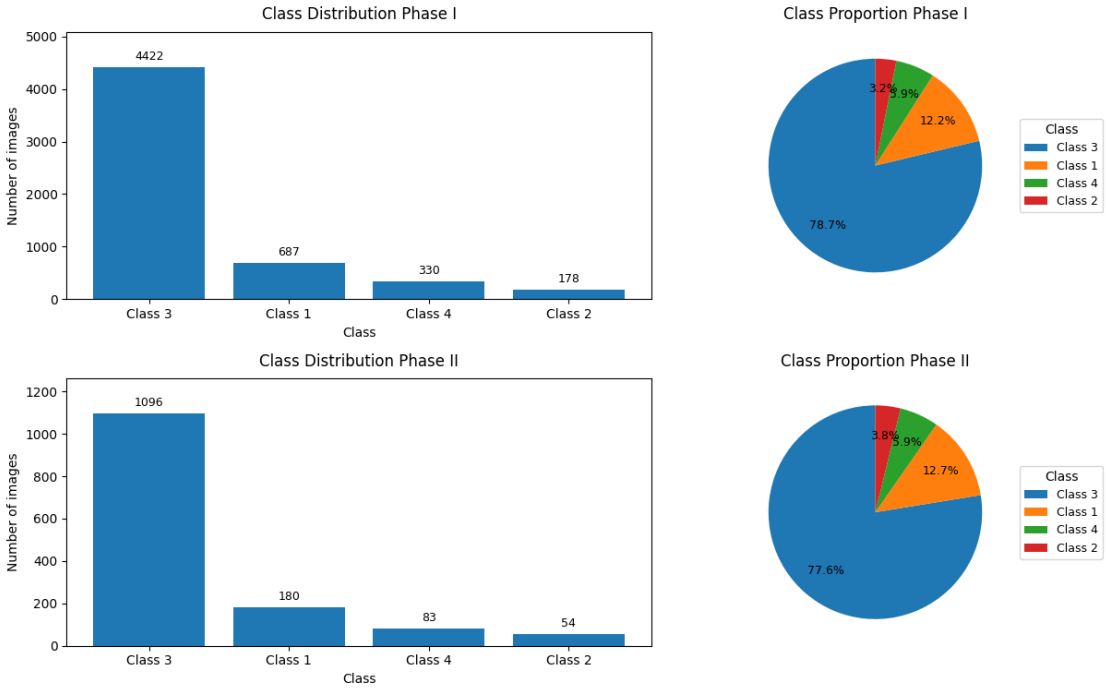


Figure 7. Frequency Distribution and Class Proportions for Phase I and Phase II

3.4. Defect count extraction validation

The defect-counting stage was validated by converting predicted segmentation masks into detected-component tables. After the U-Net model produced a predicted probability mask, the mask was thresholded at 0.50 to obtain a binary defect mask. External contour detection was then applied to identify distinct defect regions. Components with an area smaller than 50 pixels were removed to reduce false counts from isolated pixels or small fragmented predictions.

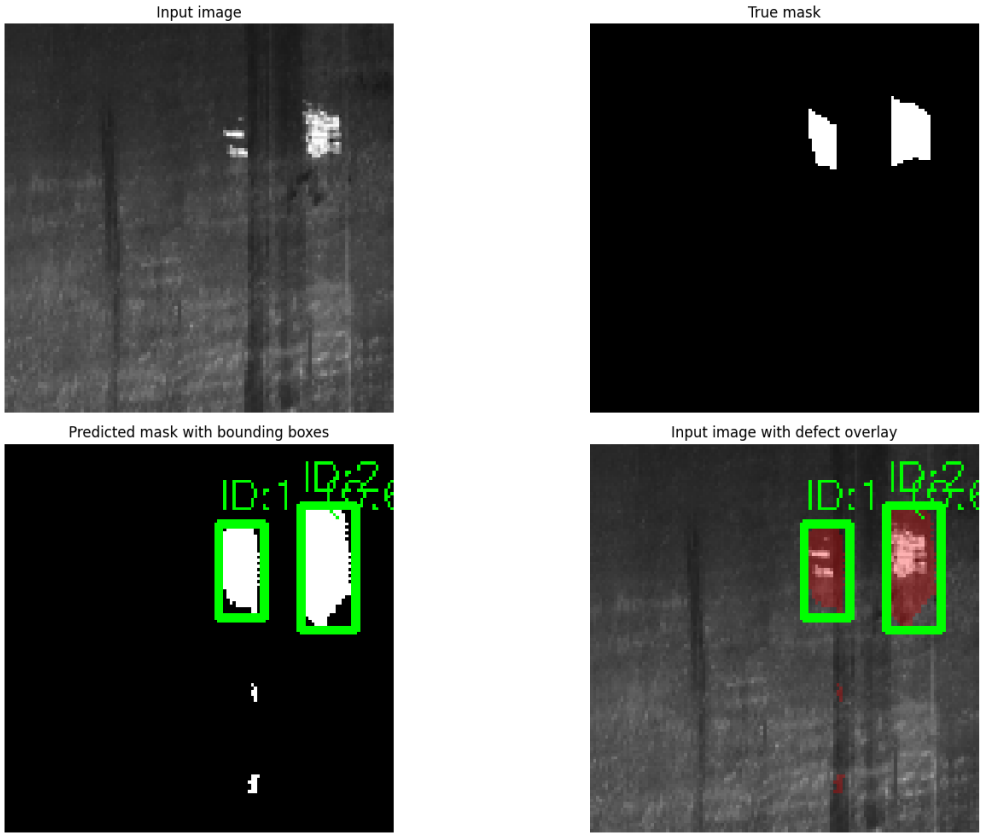


Figure 8. Detected defect components and bounding-box visualization

For each detected component, the method recorded the defect identification number, confidence score, defect area, bounding-box coordinates, bounding-box area, and number of polygon points. **Figure 8** presents an example of detected defect components and bounding-box visualization. This output confirms that the predicted segmentation mask can be transformed into quantitative defect-count data. In one validation example, two detected defect regions were obtained, with confidence scores of 0.558 and 0.884 and a total detected defect area of 255 pixels. This example demonstrates that the image-level defect count can be extracted from the segmentation result and used as the input for subgroup-level U-chart monitoring.

3.5. U-control chart based on U-Net-derived defect counts

The U-control chart was used to validate whether the image-derived defect counts could be transformed into statistical process monitoring signals. The monitoring statistic was calculated from the number of detected defect regions obtained after U-Net segmentation, binary thresholding, minimum-area filtering, and external contour detection. Each subgroup consisted of 50 inspected steel surface images, and the defect-per-unit value was calculated as the total number of detected defect regions divided by the number of inspected images in the subgroup.

3.5.1. Initial Phase I U-control chart

The initial Phase I U-control chart was constructed to evaluate the stability of the baseline monitoring data. Phase I subgroups were used to estimate the preliminary center line and control limits. As shown in **Figure 9**, most Phase I subgroups were located around the center line, indicating that the average number of detected defects per inspected unit was relatively stable. However, an out-of-control subgroup was detected in the initial Phase I chart.

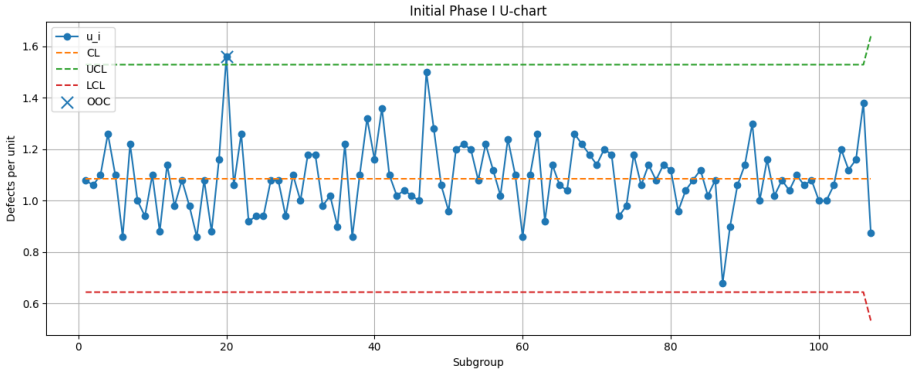


Figure 9. Initial Phase I U-chart

The presence of an out-of-control point in Phase I indicates that the initial baseline should not be accepted directly as the final in-control condition. In statistical process monitoring, Phase I is used to screen historical or reference data before prospective monitoring is performed. Therefore, the detected out-of-control subgroup should be investigated before recalculating the control limits.

3.5.2. Revised Phase I U-control chart

After the out-of-control subgroup in the initial Phase I chart was screened, the U-control chart was recalculated using the retained Phase I subgroups. **Figure 10** shows the revised Phase I U-control chart. The revised chart shows that the retained Phase I subgroups fall within the control limits, indicating that the revised Phase I data can be used as the stable baseline for subsequent monitoring.

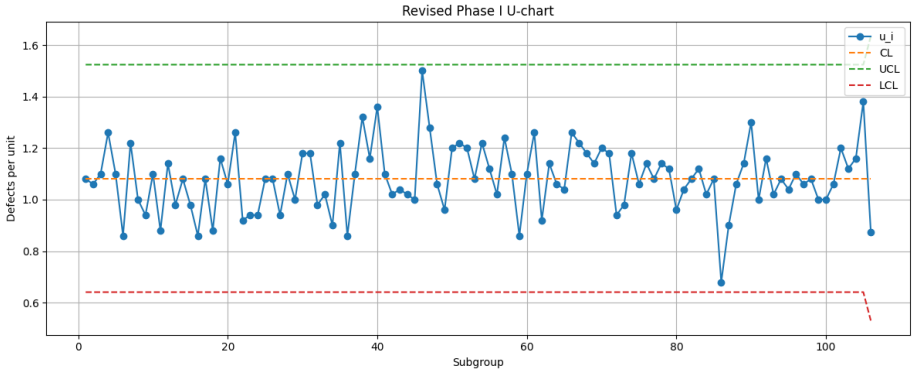


Figure 10. Revised Phase I U-chart after Phase I screening

The revised Phase I chart is important because Phase II monitoring must be based on a stable reference condition. If unstable Phase I observations are retained, the resulting control limits may be misleading. They may become too wide, reducing sensitivity to process changes, or too narrow, increasing the risk of false alarms. Therefore, the revised Phase I center line and control limits were used as the final reference limits for Phase II monitoring.

3.5.3. Phase II U-control chart using revised Phase I limits

The Phase II U-control chart was constructed by applying the revised Phase I limits to the subsequent monitoring subgroups. The Phase II data were not used to estimate new control limits. Instead, each Phase II defect-per-unit value was compared with the revised Phase I center line, UCL, and LCL. This follows the prospective monitoring principle in statistical process control, where Phase I establishes the baseline and Phase II evaluates future process behavior.

As shown in **Figure 11**, the Phase II subgroups are generally located within the revised Phase I control limits. This indicates that, under the demonstrated monitoring sequence, the Phase II defect-per-unit behavior does not show strong evidence of out-of-control variation. Therefore, the process can be interpreted as statistically stable during Phase II based on the revised Phase I baseline.

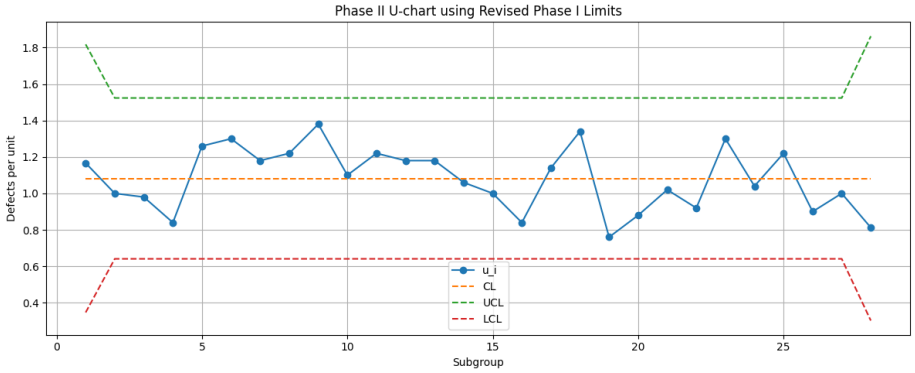


Figure 11. Phase II U-chart using revised Phase I limits

If a future Phase II subgroup exceeds the UCL, it indicates an unusual increase in the number of detected defects per inspected unit. This may reflect special-cause variation, such as changes in raw material, surface condition, machine setting, or production environment. If a Phase II subgroup falls below the LCL, it may indicate an unusual reduction in detected defects. Although this may appear favorable, it still represents a departure from the established baseline and should be examined. In both cases, the corresponding steel surface images, predicted masks, detected components, confidence scores, and classification outputs should be reviewed to determine whether the signal is caused by an actual process change, a segmentation error, or another assignable cause. Overall, the U-control chart validation confirms that the proposed workflow can transform image segmentation outputs into statistically interpretable monitoring signals. The segmentation model localizes defective regions, the contour-based procedure converts predicted masks into defect-count data, and the U-control chart evaluates whether the defect-per-unit behavior remains consistent with the revised Phase I baseline.

3.6. Limitations

1. The demonstrated application uses the Severstal dataset, which does not provide actual production timestamps or inspection order. The Phase I and Phase II sequence should therefore be interpreted as a reproducible demonstration sequence rather than the true chronological manufacturing sequence.
2. The U-control chart output depends on segmentation quality. Under-segmentation may reduce the detected defect count, while over-segmentation or small false-positive fragments may inflate the count and cause false alarms.
3. The segmentation threshold of 0.50 and minimum component area of 50 pixels may need adjustment for different image resolutions, illumination conditions, noise levels, or defect morphologies.
4. Resizing images to 128 x 128 pixels reduces computational cost but may remove fine defect details from the original high-resolution images. Applications requiring precise boundary localization may require larger image sizes or additional post-processing.

5. The classification output supports defect-type interpretation but does not directly determine the U-chart statistic. Additional validation is needed if defect-class predictions are used as primary decision outputs.
6. Phase I revision should not be performed automatically in industrial applications. Subgroups should be removed only after assignable-cause investigation or engineering justification.
7. The method was demonstrated on an offline dataset. Real-time deployment would require integration with production timestamps, image acquisition systems, automated storage, computing infrastructure, and operator decision rules.

4. CONCLUSION

This study integrated U-Net segmentation, MobileNetV2-based multi-label classification, and the U-control chart into a single workflow for monitoring steel surface defects. The RLE annotations of the Severstal dataset were reconstructed into binary masks, the predicted segmentation masks were converted into image-level defect counts through thresholding, minimum-area filtering, and external contour detection, and the counts were monitored as defects per inspected unit in subgroups of 50 images. The validation showed that the U-Net model localized the main defective regions, whereas classification performance varied across defect classes because the classes were not equally represented. Because the U-chart statistic was calculated from segmentation-derived defect counts, classification errors affected defect-type interpretation but did not directly determine the out-of-control decision. The initial Phase I U-chart signaled one out-of-control subgroup. After this subgroup was screened, the revised Phase I chart provided a stable baseline, and the Phase II subgroups generally remained within the revised limits, indicating no strong evidence of out-of-control variation in the demonstrated monitoring sequence. These results confirm that image segmentation outputs can be transformed into statistically interpretable monitoring signals. Future studies should apply the method to production data with actual inspection timestamps, adjust the segmentation threshold and minimum component area for different imaging conditions, evaluate larger input image sizes for fine defect details, and integrate the workflow with real-time image acquisition and operator decision rules.

Acknowledgments

The authors acknowledge the availability of the Severstal: Steel Defect Detection dataset through Kaggle.

Author Contributions Statement

Inas Wulan Ramadhani: Software, Formal analysis, Investigation, Data curation, Writing - original draft, Visualization. Muhammad Ahsan: Conceptualization, Methodology, Validation, Resources, Writing - review and editing, Supervision, Project administration, Funding acquisition. Wibawati: Validation, Resources, Writing - review and editing.

Conflict of Interest Statement

Authors state no conflict of interest.

Data Availability

The data that support the findings of this study are openly available in the Severstal: Steel Defect Detection dataset on Kaggle at https://kaggle.com/competitions/severstal-steel-defect-detection. Users who reproduce or extend this study should comply with the access conditions and usage terms of the original dataset provider.

REFERENCES

[1] E. N. Malamas, E. G. M. Petrakis, M. Zervakis, L. Petit, and J.-D. Legat, "A survey on industrial vision systems, applications and tools," *Image Vis. Comput.*, vol. 21, no. 2, pp. 171–188, Feb. 2003, doi: 10.1016/S0262-8856(02)00152-X.

[2] A. Sioma, "Vision System in Product Quality Control Systems," *Applied Sciences*, vol. 13, no. 2, p. 751, Jan. 2023, doi: 10.3390/app13020751.

[3] M. Babic, M. A. Farahani, and T. Wuest, "Image Based Quality Inspection in Smart Manufacturing Systems: A Literature Review," *Procedia CIRP*, vol. 103, pp. 262–267, 2021, doi: 10.1016/j.procir.2021.10.042.

[4] A. A. P. Chazhoor, E. S. L. Ho, B. Gao, and W. L. Woo, "A Review and Benchmark on State-of-the-Art Steel Defects Detection," *SN Comput. Sci.*, vol. 5, no. 1, p. 114, Dec. 2023, doi: 10.1007/s42979-023-02436-2.

[5] T. S. Adeyemi, "Defect Detection in Manufacturing: An Integrated Deep Learning Approach," *Journal of Computer and Communications*, vol. 12, no. 10, pp. 153–176, 2024, doi: 10.4236/jcc.2024.1210011.

[6] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artif. Intell. Rev.*, vol. 53, no. 8, pp. 5455–5516, Dec. 2020, doi: 10.1007/s10462-020-09825-6.

[7] J. Wang, T. Zhang, Y. Cheng, and N. Al-Nabhan, "Deep Learning for Object Detection: A Survey," *Computer Systems Science and Engineering*, vol. 38, no. 2, pp. 165–182, 2021, doi: 10.32604/csse.2021.017016.

[8] W. Zhao, F. Chen, H. Huang, D. Li, and W. Cheng, "A New Steel Defect Detection Algorithm Based on Deep Learning," *Comput. Intell. Neurosci.*, vol. 2021, no. 1, Jan. 2021, doi: 10.1155/2021/5592878.

[9] F. Chen, L. Fu, Y. Zhang, J. Li, Q. Zhang, and S. Bi, "A Review of Deep Learning-Based Steel Surface Defect Detection," *Academic Journal of Science and Technology*, vol. 15, no. 1, pp. 198–202, Apr. 2025, doi: 10.54097/g36nm962.

[10] N. Siddique, S. Paheding, C. P. Elkin, and V. Devabhaktuni, "U-Net and Its Variants for Medical Image Segmentation: A Review of Theory and Applications," *IEEE Access*, vol. 9, pp. 82031–82057, 2021, doi: 10.1109/ACCESS.2021.3086020.

[11] C. Williams, F. Falck, G. Deligiannidis, C. C. Holmes, A. Doucet, and S. Syed, "A Unified Framework for U-Net Design and Analysis," in *Advances in Neural Information Processing Systems 36*, San Diego, California, USA: Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2023, pp. 27745–27782. doi: 10.52202/075280-1206.

[12] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," 2015, pp. 234–241. doi: 10.1007/978-3-319-24574-4_28.

[13] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, IEEE, Aug. 2017, pp. 1–6. doi: 10.1109/ICEngTechnol.2017.8308186.

[14] S. Chen and L. Wells, "A multi-image monitoring framework for statistical process control to improve manufacturing systems," *Qual. Reliab. Eng. Int.*, vol. 38, no. 2, pp. 679–702, Mar. 2022, doi: 10.1002/qre.3008.

[15] F. M. Megahed, W. H. Woodall, and J. A. Camelio, "A Review and Perspective on Control Charting with Image Data," *Journal of Quality Technology*, vol. 43, no. 2, pp. 83–98, Apr. 2011, doi: 10.1080/00224065.2011.11917848.

[16] J. M. Prats-Montalbán and A. Ferrer, "Statistical process control based on Multivariate Image Analysis: A new proposal for monitoring and defect detection," *Comput. Chem. Eng.*, vol. 71, pp. 501–511, Dec. 2014, doi: 10.1016/j.compchemeng.2014.09.014.

[17] Y. Okhrin, W. Schmid, and I. Semeniuk, "A control chart for monitoring image processes based on convolutional neural networks," *Stat. Neerl.*, vol. 79, no. 1, Feb. 2025, doi: 10.1111/stan.12366.

[18] M. Ahsan, M. Mashuri, H. Kuswanto, and D. D. Prastyo, "Intrusion Detection System Using Multivariate Control Chart Hotelling's T2 Based on PCA," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 8, no. 5, pp. 1905–1911, Oct. 2018, doi: 10.18517/ijaseit.8.5.3421.

[19] S. Bersimis, A. Sachlas, and P. Economou, "Monitoring defects on products' surface by incorporating scan statistics into process monitoring procedures," *Qual. Reliab. Eng. Int.*, vol. 41, no. 1, pp. 293–317, Feb. 2025, doi: 10.1002/qre.3652.

[20] S. Dahari, M. A. Talib, and A. A. Ghapor, "Robust Control Chart Application in Semiconductor Manufacturing Process," *Journal of Advanced Research in Applied Sciences and Engineering Technology*, vol. 43, no. 2, pp. 203–219, 2024, doi: 10.37934/araset.43.2.203219.

[21] Amitava. Mitra, *Fundamentals of quality control and improvement*. John Wiley & Sons, Inc., 2016.

[22] A. Pipatprapa, "Intelligent Attribute Control Chart System Establishing Fuzzy Image Processing Technique," *Kasem Bundit Engineering Journal*, vol. 8, no. Special Issue, pp. 1221–132, May 2018, Accessed: Jun. 18, 2026. [Online]. Available: https://ph01.tci-thaijo.org/index.php/kbej/article/view/120382/102523

[23] D. C. Montgomery, *Introduction to Statistical Quality Control*, 8th ed. John Wiley & Sons, 2020.

[24] A. Grishin, BorisV, iBardintsev, inversion, and Oleg, "Severstal: Steel Defect Detection," Kaggle. Accessed: Jul. 02, 2026. [Online]. Available: https://kaggle.com/competitions/severstal-steel-defect-detection

[25] A. Bano and M. Salamat, "Image Processing Methods: A Systematic Literature Review of Classical and Modern Approaches," *The Asian Bulletin of Big Data Management*, vol. 5, no. 4, pp. 91–123, Nov. 2025, doi: 10.62019/s6z7we84.

[26] R. C. . Gonzalez and R. E. . Woods, *Digital image processing*. Pearson, 2018.

[27] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image Segmentation Using Deep Learning: A Survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 1–1, 2021, doi: 10.1109/TPAMI.2021.3059968.

[28] R. Yulianto *et al.*, "Innovative UNET-Based Steel Defect Detection Using 5 Pretrained Models," *Evergreen*, vol. 10, no. 4, pp. 2365–2378, Dec. 2023, doi: 10.5109/7160923.

[29] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2018, pp. 4510–4520. doi: 10.1109/CVPR.2018.00474.

[30] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, 2010, pp. 807–804.

[31] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Jan. 2017, [Online]. Available: http://arxiv.org/abs/1412.6980

[32] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *J. Big Data*, vol. 6, no. 1, p. 60, Dec. 2019, doi: 10.1186/s40537-019-0197-0.

[33] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.

[34] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, Jun. 2006, doi: 10.1016/j.patrec.2005.10.010.