**PARAMETER**
JURNAL MATEMATIKA, STATISTIKA DAN TERAPANNYA

e-ISSN. 2829-6303
Vol. 05, No. 3, December 2026
Pages 453-470

Check for updates

# Performance Evaluation of U-Net for Semantic Segmentation of Surface Defects in the Casting Surface Defect Dataset

**Akhmad Imam Haromain [1], Wibawati [1*], Muhammad Ahsan [1]**

[1]Department of Statistics, Faculty of Science and Data Analytics,
Institut Teknologi Sepuluh Nopember
Jl. Raya ITS, Surabaya, 60111, East Java, Indonesia

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

***Abstract***

*Accurate segmentation of casting surface defects is challenging because high-resolution images contain small and highly imbalanced defect regions. This study evaluates U-Net with a ResNet34 encoder for semantic segmentation of scratch, spot, and rust defects using the Casting Surface Defect Dataset (CSDD). The dataset comprises 2,100 RGB images (3,648 × 3,648 pixels), partitioned into 1,343 training, 335 validation, and 422 test images. Training used 768 × 768-pixel patches at the original pixel scale, defect-aware sampling, and combined cross-entropy and soft Dice loss. A validation-based learning-rate search preceded three independent training runs (seeds 2027, 2028, and 2029). Full-resolution predictions were reconstructed using overlapping sliding windows and uniform logit averaging. On the official test set, mean defect-mIoU was 60.004 ± 0.264%, four-class mIoU 69.978 ± 0.198%, and macro Dice 74.539 ± 0.205%. Class-wise IoU was 72.857% for scratch, 58.084% for spot, and 49.072% for rust. Image-wise IoU distributions revealed heterogeneous performance, while confusion-matrix analysis showed that 33.61% of ground-truth rust pixels were classified as background. Qualitative examples illustrated accurate localization, false positives, over-segmentation, and missed rust. Findings characterize the evaluated configuration without establishing individual strategy effects or generalization beyond official CSDD partitions.*

***Keywords:*** *Casting surface defect dataset, class imbalance, defect-aware sampling, semantic segmentation, U-Net.*

: https://doi.org/10.30598/parameterv5i3pp453-470

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: October 2026*
*Accepted: December 2026*

## 1. INTRODUCTION

Surface defect inspection is an essential component of quality control in the metal casting industry, as defects can affect product appearance, functionality, and conformity to specifications. Manual inspection has several limitations, particularly operator fatigue, subjective assessments, and inconsistent inspection results. Advances in machine vision and deep learning have enabled automated inspection through the analysis of material surface characteristics. Ren et al. [1] reviewed the development of machine vision-based defect detection methods across various industrial applications, while Chen et al. [2] identified challenges related to limited training samples, small defect sizes, data imbalance, and computational efficiency. Therefore, automated inspection requires methods capable of accurately identifying both defect types and their spatial extent.

One promising approach is semantic segmentation, which assigns a class label to each image pixel to obtain detailed information about defect location, shape, and spatial extent. U-Net is an encoder–decoder segmentation architecture that employs skip connections to integrate spatial and semantic information [3]. A ResNet34 encoder based on residual learning [4] can be incorporated into U-Net to provide pretrained feature representations for semantic segmentation. In industrial inspection, Tabernik et al. [5] developed a deep learning-based surface defect segmentation method using a limited number of defective samples. Neven and Goedemé [6] employed a ResNet34-based U-Net for segmenting steel surface defect types and severity levels, while Ashrafi et al. [7] evaluated several deep learning architectures for steel surface defect detection and segmentation. These studies demonstrate the relevance of U-Net-based architectures to industrial surface inspection across different dataset characteristics and evaluation procedures.

Developing segmentation methods requires datasets with adequate defect region annotations. Bergmann et al. [8] introduced MVTec AD for industrial anomaly detection with pixel-level annotations, while Božič et al. [9] developed a mixed-supervision approach and introduced KolektorSDD2 for surface defect inspection. Specifically for metal casting applications, Mao et al. [10] introduced the Casting Surface Defect Dataset (CSDD), comprising 2,100 RGB images with a resolution of 3,648 × 3,648 pixels and 56,356 annotated defect instances across three classes: scratch, spot, and rust. In their experimental setup, Mao et al. [10] resized the original images to 1,024 × 1,024 pixels. They also reported that most defects were smaller than 100 pixels in the original images, with some smaller than 50 pixels. In the CSDD benchmark, U-Net achieved a mean Intersection over Union (mIoU) of 54.29%, with class-wise IoU values of 65.73%, 50.64%, and 46.49% for scratch, spot, and rust, respectively. These differences indicate varying segmentation performance across defect types, particularly for spot and rust, which achieved lower IoU values than scratch. The combination of small defect sizes and image downsampling highlights the importance of investigating segmentation approaches that preserve the original pixel scale, although their potential performance benefits require evaluation under controlled experimental conditions.

CSDD segmentation presents challenges associated with high image resolution and severe pixel-level class imbalance, as defect regions occupy only a small proportion of each image compared with the background. Buda et al. [11] demonstrated that class imbalance can affect convolutional neural network performance. For surface defect segmentation, Xie et al. [12] developed a balanced loss approach, while Sudre et al. [13] investigated generalized Dice loss for highly imbalanced segmentation tasks. Beyond loss-function adjustments, foreground-oriented sampling has been used to increase the

representation of annotated regions in training patches, as illustrated by Isensee et al. [14]. In this study, image–mask pairs are cropped into patches at their original pixel scale, while defect-aware sampling is applied to increase the frequency with which patches containing scratch, spot, or rust regions are selected during training. Although patch extraction preserves local spatial resolution, it restricts the image context available to each model input, and aggregating predictions from overlapping patches may affect segmentation results [15]. These considerations motivate a processing strategy that accommodates high-resolution images while increasing the exposure of the model to annotated defect regions during training.

Although Mao et al. [10] established segmentation benchmarks for CSDD, their experiments resized the original 3,648 × 3,648-pixel images to 1,024 × 1,024 pixels. Given the small size of many annotated defects relative to the image dimensions, such downsampling reduces the spatial detail available to represent fine defect structures and boundaries. The reported benchmark therefore does not directly characterize the performance of a U-Net–ResNet34 configuration that processes image–mask patches at the original pixel scale, incorporates defect-aware sampling, and reconstructs full-resolution predictions from overlapping patches. Furthermore, evaluating this configuration across independent training runs is important for characterizing performance variability, while aggregate metrics alone are insufficient to describe class-specific segmentation performance and prediction errors. Taha and Hanbury [16] discussed segmentation evaluation metrics, while Reinke et al. [17] emphasized the importance of selecting and interpreting metrics according to task characteristics. Therefore, evaluation should incorporate defect-focused metrics, class-wise performance analysis, confusion matrices, and segmentation visualizations to provide a more comprehensive understanding of model performance.

Based on these considerations, this study aims to evaluate U-Net with a ResNet34 encoder for semantic segmentation of surface defects on CSDD using patches extracted at the original pixel scale and defect-aware sampling to increase the representation of annotated defect regions during training. The experiments involved validation-based learning rate selection followed by three independent training runs with different random seeds using the official dataset partitions. Inference was performed using a sliding window approach with logit aggregation to reconstruct segmentation masks at the original image resolution. The primary evaluation metric was defect mean Intersection over Union (defect-mIoU), supplemented by four-class mIoU, Dice/F1-score, precision, recall, confusion matrix analysis, and qualitative visualizations. Specifically, this study addresses three research questions: (1) how does the evaluated U-Net–ResNet34 configuration perform across the three defect classes on the official CSDD test set; (2) how does its segmentation performance vary across independent training runs; and (3) what class-specific error patterns and limitations are revealed by quantitative and qualitative evaluations? The main contribution is an empirical characterization of an integrated U-Net–ResNet34 segmentation configuration that combines native-scale patch processing, defect-aware sampling, and full-resolution prediction reconstruction, with particular emphasis on performance variability and class-specific segmentation errors.

## 2. METHOD

### 2.1. Dataset Description and Experimental Design

This study employed the Casting Surface Defect Dataset (CSDD), developed by Mao et al. [10], comprising 2,100 RGB images with a resolution of 3,648 × 3,648 pixels and 56,356 annotated defect instances. The dataset includes three defect classes: scratch, spot, and rust. Semantic segmentation was performed using four pixel labels: 0 for background, 1 for scratch, 2 for spot, and 3 for rust. The operational class-ID mapping was supported by visual inspection of 25 training-image examples with clearly identifiable defect characteristics, retained from a documented review of corresponding RGB images and segmentation masks. These examples covered scratch, spot, and rust, and their annotated regions were visually consistent with the assigned categories. However, the available dataset documentation did not explicitly specify the numerical class-ID ordering, and direct confirmation from the dataset developers was unavailable. The official CSDD split [10] was preserved, consisting of 1,343 training images, 335 validation images, and 422 test images, without reshuffling or transferring images between subsets. The training set was used for model optimization, the validation set for learning-rate and checkpoint selection, and the test set exclusively for final evaluation.

The experiments employed U-Net [3] with a ResNet34 encoder [4] initialized using ImageNet-pretrained weights. Training comprised two stages: learning rate selection using random seed 2026 and final training through three independent runs using seeds 2027, 2028, and 2029. The best checkpoint from each run was selected based on validation defect-mIoU. All selected checkpoints were subsequently evaluated independently on the official test set without using test results for model selection. The original image dimensions were preserved, and evaluation was performed at the full resolution of 3,648 × 3,648 pixels. Although the official image-level partitions were retained, independence between physical casting specimens across partitions could not be established from the available information. Therefore, the findings are restricted to the official CSDD benchmark and do not establish generalization to independent physical specimens or different industrial production conditions.

### 2.2. Data Preprocessing and Patch Sampling

Preprocessing began by verifying the correspondence between RGB images and ground-truth masks, image dimensions, and annotation labels. Given the high image resolution, training was performed using 768 × 768-pixel patches extracted from images and masks at identical spatial coordinates without resizing the original images. This approach preserved the original pixel scale while allowing high-resolution images to be processed in smaller regions. To increase the likelihood of selecting training patches containing annotated defects, a defect-aware sampling strategy was adopted, considering the effects of class imbalance on convolutional neural network learning [11] and the principles of foreground-oriented sampling [14].

For each patch extraction, defect-aware sampling was selected with a probability of 0.70, while random sampling was selected with a probability of 0.30. In the defect-aware branch, one of the three defect classes was selected with equal probability, followed by random selection of an image containing the target class and a target pixel with cropping coordinates chosen to include that pixel. The random branch did not require the presence of a particular defect class. Sampling was performed with replacement, generating 10,744 patches per epoch. This budget corresponded to eight times the number of training images (1,343 × 8), without requiring an equal number of

patches per image. Consequently, an image could be selected multiple times within an epoch, potentially with different crop locations or augmentations, while uniform image-level coverage was not guaranteed. The sampling probability represented branch selection rather than the exact proportion of defect-containing patches, and was not optimized through ablation experiments.

Geometric augmentation included rotations of 0°, 90°, 180°, and 270°, as well as horizontal flipping, applied identically to images and masks. These transformations preserved image–mask alignment without interpolating categorical mask labels. Brightness and contrast adjustments were applied only to RGB images with a probability of 0.50 and randomly sampled factors ranging from 0.90 to 1.10, following the image augmentation principles discussed by Shorten and Khoshgoftaar [18]. After augmentation, RGB pixel intensities were scaled to the range of 0–1 and standardized using ImageNet channel-wise means and standard deviations. ImageNet-based normalization was also reported by Jumaboev et al. [19] in their deep learning-based photovoltaic thermal image segmentation study. The normalization procedure used in this study is expressed in **Equation (1)**.

$$\tilde{I}_q(x,y) = \frac{I_q(x,y) \,/\, 255 - \mu_q}{\sigma_q} \tag{1}$$

Here, $I_q(x,y)$ denotes the pixel intensity at coordinates $(x,y)$ for channel $q \in \{R,G,B\}$, and $\tilde{I}_q(x,y)$ represents the normalized pixel value. The channel-wise means and standard deviations were $\mu_q$ = (0.485, 0.456, 0.406) and $\sigma_q$ = (0.229, 0.224, 0.225), respectively. Normalization was applied only to RGB images, while the masks retained their categorical labels. Validation and test images were processed deterministically using 768 × 768-pixel patches with a stride of 576 pixels, producing 36 patches per image without random augmentation, padding, or resizing of the original images. Defect-aware sampling was not applied during validation or testing, and full-resolution prediction reconstruction is described in Section 2.5.

### 2.3. U-Net–ResNet34 Architecture

The segmentation model employed the U-Net architecture [3] with a ResNet34 encoder incorporating residual learning [4]. This established encoder–decoder configuration was adopted to evaluate the patch-based segmentation approach rather than to introduce a new network architecture. The encoder was initialized with ImageNet-pretrained weights [20] and configured with a depth of five stages to extract features at multiple spatial resolutions. These features were transferred to the decoder through skip connections, enabling the integration of spatial and semantic information during segmentation reconstruction. The decoder consisted of five stages with successive output channel dimensions of 256, 128, 64, 32, and 16. Feature upsampling was performed using nearest-neighbor interpolation, with batch normalization applied within the decoder blocks and no additional attention modules. The model received 768 × 768-pixel RGB patches and generated four logit maps corresponding to the background, scratch, spot, and rust classes. The model was implemented using segmentation-models-pytorch version 0.5.0, comprising 24,436,804 trainable parameters. All parameters, including those of the pretrained encoder, were optimized during training. The same architectural configuration was maintained across all experiments.

### 2.4. Training Procedure and Checkpoint Selection

Training was performed using the AdamW optimizer [21] with a weight decay of 0.0001, $\beta_1$ = 0.9, $\beta_2$ = 0.999, and $\varepsilon = 10^{-8}$. The batch size was set to 4, resulting in 2,686 optimization steps per epoch from 10,744 patches, with gradient clipping applied at a maximum gradient norm of 1.0. The loss function combined categorical cross-entropy and soft Dice loss with equal weights of 0.5. Cross-entropy measured prediction errors across all classes, whereas soft Dice loss focused on the overlap of defect regions, following the application of Dice-based loss functions in segmentation [22]. The combined loss function is expressed in **Equation (2)**.

$$\mathcal{L}_{total} = 0.5\mathcal{L}_{CE} + 0.5\mathcal{L}_{Dice} \tag{2}$$

The categorical cross-entropy component was calculated using **Equation (3)**.

$$\mathcal{L}_{CE} = -\frac{1}{|\Omega_B|}\sum_{i\in\Omega_B}\sum_{c=0}^{3} y_{ic}\, ln(p_{ic}) \tag{3}$$

The soft Dice loss component, calculated for defect classes present in the ground truth of a given batch, is expressed in **Equation (4)**.

$$\mathcal{L}_{Dice} = 1 - \frac{1}{|\mathcal{C}_B|}\sum_{c\in\mathcal{C}_B}\left(\frac{2\sum_{i\in\Omega_B} p_{ic}y_{ic} + \varepsilon_D}{\sum_{i\in\Omega_B} p_{ic} + \sum_{i\in\Omega_B} y_{ic} + \varepsilon_D}\right) \tag{4}$$

Here, $\Omega_B$ denotes the set of valid pixels in a batch, $y_{ic}$ is the ground-truth indicator for pixel $i$ and class $c$, and $p_{ic}$ represents the predicted probability obtained using the softmax function. The set $\mathcal{C}_B$ contains the defect classes present in the ground truth of a batch and is therefore a subset of {1,2,3}. The constant $\varepsilon_D = 10^{-7}$ was used to ensure numerical stability and is distinct from the AdamW epsilon of $10^{-8}$. **Equation (4)** applies when $|\mathcal{C}_B| > 0$; otherwise, the soft Dice loss was set to zero. The soft Dice loss was averaged only over defect classes present in the batch ground truth. Pixels assigned the padding label 255 were excluded from both loss components.

The learning rate schedule combined linear warm-up during the first epoch, increasing from 10% to the reference learning rate, followed by cosine decay to 1% of the reference value over a 40-epoch scheduling horizon without warm restarts, based on the cosine annealing principle [23]. The learning rate was updated at every optimization step. During the selection stage, three candidate learning rates, namely $3 \times 10^{-5}$, $1 \times 10^{-4}$, and $3 \times 10^{-4}$, were each evaluated for 20 epochs using seed 2026. The selected learning rate was determined based on the highest validation defect-mIoU and subsequently used for three final training runs with seeds 2027, 2028, and 2029, each with a maximum of 40 epochs. The 20-epoch selection runs retained the 40-epoch scheduling horizon, and the search was limited to the predefined candidates and training budget. Validation was conducted every two epochs using all 335 validation images at their original resolution. During final training, early stopping was applied with a patience of five validation evaluations, a minimum improvement threshold (min_delta) of 0.001 on a 0–1 scale, and a minimum of 10 epochs before early stopping could be triggered. Early stopping was not applied during learning rate selection. The best checkpoint for each run was selected according to the highest validation defect-mIoU, without using test data. The min_delta threshold governed early stopping only

and did not alter the best-checkpoint selection criterion. Training was performed on an NVIDIA GeForce RTX 5070 GPU using bfloat16-based mixed precision, while loss calculations were performed in float32 precision. The software environment comprised Windows 11, Python 3.12.10, PyTorch 2.10.0+cu128 (CUDA 12.8), segmentation-models-pytorch 0.5.0, NumPy 2.5.2, and Pillow 12.3.0. Random states were initialized in Python, NumPy, PyTorch, and CUDA using the run-specific seeds; patch sampling and augmentation used generators indexed by seed, epoch, sample index, and stream. cuDNN benchmarking was disabled and deterministic cuDNN behavior was enabled, although strict PyTorch deterministic-algorithm enforcement was not activated.

### 2.5. Sliding Window Inference and Prediction Reconstruction

Inference on the validation and test sets was performed using a sliding window approach to generate segmentation masks at the original image resolution, following patch-based inference principles applied by Isensee et al. **[14]**. Each 3,648 × 3,648-pixel image was processed using 36 patches of 768 × 768 pixels with a stride of 576 pixels, resulting in an overlap of 192 pixels in both spatial directions. The resulting 6 × 6 patch grid covered the entire image without padding. Each pixel was covered by one to four patches, and reconstruction included a coverage check to ensure that no pixel remained unprocessed. Prediction aggregation was required to reconstruct the segmentation output because some image regions received predictions from multiple overlapping patches **[15]**. Each patch generated four logit maps, which were mapped back to their original image coordinates. Logits in overlapping regions were aggregated using an arithmetic mean with uniform weighting, rather than averaging softmax probabilities, after which the final class label was determined by selecting the class with the highest aggregated logit value. This procedure is expressed in **Equation (5)**.

$$\bar{z}_c(x,y) = \frac{1}{|\mathcal{W}(x,y)|} \sum_{k \in \mathcal{W}(x,y)} z_c^{(k)}(x,y),$$
$$\hat{y}(x,y) = \underset{c \in \{0,1,2,3\}}{arg\,max}\, \bar{z}_c(x,y). \tag{5}$$

Here, $\bar{z}_c(x,y)$ denotes the aggregated logit for class $c$ at coordinates $(x,y)$, $\mathcal{W}(x,y)$ represents the set of patch indices covering those coordinates, and $z_c^{(k)}(x,y)$ is the logit predicted by the $k$-th patch. The symbol $\hat{y}(x,y)$ denotes the final predicted class label. Aggregation was performed directly on logits using float32 precision, without Gaussian weighting, test-time augmentation, or additional postprocessing. The uniform-weighting scheme was fixed, and alternative overlap-aggregation strategies were not evaluated. Potential segmentation artifacts at patch boundaries were not separately quantified. The resulting full-resolution segmentation masks were used for model evaluation.

### 2.6. Segmentation Performance Evaluation Metrics

Evaluation was conducted by comparing predicted segmentation masks with ground-truth masks across all 422 images in the official test set. Metrics were calculated from a 4 × 4 global confusion matrix, with ground-truth classes as rows and predicted classes as columns, accumulated over the reconstructed full-resolution pixels for each training run, rather than averaging image-wise or patch-wise metrics. The evaluation metrics included Intersection over Union (IoU), Dice/F1-score, precision, and recall to

assess segmentation overlap and classification errors [16], [17]. For each class $c$, these metrics were calculated using true positives ($TP_c$), false positives ($FP_c$), and false negatives ($FN_c$), as defined in **Equation (6)**.

$$IoU_c = \frac{TP_c}{TP_c + FP_c + FN_c}, \quad Dice_c = \frac{2TP_c}{2TP_c + FP_c + FN_c},$$
$$Precision_c = \frac{TP_c}{TP_c + FP_c}, \quad Recall_c = \frac{TP_c}{TP_c + FN_c}. \tag{6}$$

The primary evaluation metric was defect mean Intersection over Union (defect-mIoU), defined as the average IoU across the three defect classes while excluding the background class. As a supplementary metric, four-class mIoU, which includes the background class, was also calculated. Both metrics are formulated in **Equation (7)**.

$$defect - mIoU = \frac{1}{3}\sum_{c=1}^{3} IoU_c\,, \qquad mIoU_4 = \frac{1}{4}\sum_{c=0}^{3} IoU_c. \tag{7}$$

Here, $c$ = 0 denotes the background class, whereas $c$ = 1, 2, 3 represent scratch, spot, and rust, respectively. Dice/F1-score, precision, and recall were also summarized using macro-averaging across the three defect classes, assigning equal weight to each class. As a complementary analysis, class-wise IoU was calculated per image and seed from individual confusion matrices, excluding images in which the corresponding class was absent from the ground truth. The three seed-specific IoUs were averaged for each image, and distributions across images were summarized using the median and interquartile range (IQR). These image-wise summaries did not replace the primary global metrics. To analyze interclass classification errors, the confusion matrix was normalized row-wise by the number of ground-truth pixels in each class. Normalization was performed separately for each run, and the resulting matrices were averaged across runs to visualize overall misclassification patterns. This averaged matrix was used for visualization rather than for calculating the reported performance metrics. All evaluation metrics were calculated independently for seeds 2027, 2028, and 2029, then summarized using the mean and sample standard deviation across seeds, reported as mean ± SD. The standard deviation was used descriptively and was not intended as a measure of statistical significance. In addition to quantitative evaluation, qualitative analysis was conducted by comparing the original images, ground-truth masks, and predictions from seed 2027 at corresponding spatial locations to examine differences in defect coverage and segmentation errors. These examples illustrated selected error patterns rather than their prevalence across the test set. The evaluation results are presented in Section 3.

## 3. RESULTS AND DISCUSSION

### 3.1. Dataset Characteristics and Class Imbalance

This study employed 2,100 images from the Casting Surface Defect Dataset (CSDD), comprising 1,343 training images, 335 validation images, and 422 test images. All images have a resolution of 3,648 × 3,648 pixels and contain four segmentation classes: background, scratch, spot, and rust. The pixel distributions across the three dataset partitions are presented in **Table 1**.

**Table 1.** Pixel distribution of each class in the CSDD dataset

| Class | Training (%) | Validation (%) | Test (%) |
|---|---|---|---|
| Background | 99.7832 | 99.7869 | 99.7762 |
| Scratch | 0.1103 | 0.1091 | 0.1151 |
| Spot | 0.0151 | 0.0149 | 0.0153 |
| Rust | 0.0915 | 0.0891 | 0.0934 |

As shown in **Table 1**, the background class accounts for more than 99.7% of pixels across all dataset partitions, whereas the three defect classes collectively represent only 0.2168% of training pixels, 0.2131% of validation pixels, and 0.2238% of test pixels. Spot has the smallest pixel proportion, followed by rust and scratch. This distribution indicates severe class imbalance, which can obscure defect-specific performance when evaluation metrics include a background class with high segmentation accuracy. Therefore, this study employed defect-mIoU as the primary evaluation metric, assigning equal weight to the three defect classes. The training-set class distribution motivated the predefined defect-aware sampling strategy described in Section 2, whereas validation and test class distributions were examined descriptively and were not used to tune the sampling strategy.

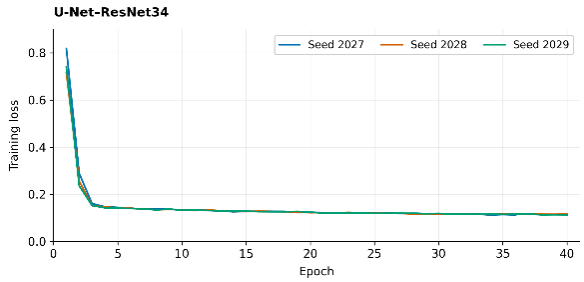
### 3.2. U-Net–ResNet34 Training and Validation Results

U-Net–ResNet34 training consisted of two stages: learning rate selection and final model training. In the first stage, three candidate learning rates were evaluated over 20 epochs using random seed 2026. The learning rate was selected based on the highest defect-mIoU achieved on the validation set, as presented in **Table 2**.

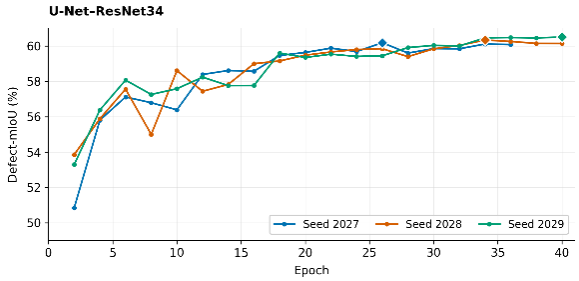
**Table 2.** Learning rate search results for U-Net–ResNet34

| Learning rate | Best epoch | Validation defect-mIoU (%) |
|---|---|---|
| $3 \times 10^{-5}$ | 20 | 59.535 |
| $1 \times 10^{-4}$ | 20 | 59.187 |
| $3 \times 10^{-4}$ | 20 | 57.791 |

As shown in **Table 2**, the learning rate of $3 \times 10^{-5}$ achieved the highest validation defect-mIoU of 59.535%, outperforming the other two candidates by 0.348 and 1.744 percentage points, respectively. Accordingly, this learning rate was selected for final training. However, all three candidates achieved their highest observed validation scores at epoch 20, the predefined screening limit. Thus, the selection was specific to the evaluated learning rates and training budget, without establishing convergence or optimality beyond this limit. Final training was conducted using random seeds 2027, 2028, and 2029, each with a maximum of 40 epochs. The progression of training loss and validation performance is illustrated in **Figure 1**.




(a)




(b)

**Figure 1.** Progression of (a) training loss and (b) validation defect-mIoU for U-Net–ResNet34 across three training runs

As illustrated in **Figure 1(a)**, training loss decreased sharply during the initial epochs, followed by a more gradual reduction toward approximately 0.12. The relatively similar patterns across the three seeds suggest broadly comparable training-loss trends under the applied training configuration. Meanwhile, **Figure 1(b)** shows a rapid improvement in validation defect-mIoU during the early training stages, followed by fluctuations near 60% toward the end of training. These trends indicate that decreasing training loss did not necessarily correspond to continuous improvements in validation performance. Therefore, checkpoints were selected based on the highest validation defect-mIoU rather than simply using the final training epoch. The selected checkpoint for each training run is presented in **Table 3**.

**Table 3.** Best checkpoints selected based on validation defect-mIoU

| Seed | Best epoch | Final epoch | Defect-mIoU (%) |
|---|---|---|---|
| 2027 | 26 | 36 | 60.189 |
| 2028 | 34 | 40 | 60.339 |
| 2029 | 40 | 40 | 60.513 |

As shown in **Table 3**, the three training runs achieved a mean validation defect-mIoU of $60.347 \pm 0.162\%$. The best checkpoints were obtained at different epochs, namely epochs 26, 34, and 40. Training with seed 2027 was terminated at epoch 36 through the early stopping mechanism, whereas the other two runs reached the maximum of 40 epochs. These differences indicate variability in optimization trajectories across runs despite identical training configurations. Because seed 2029 achieved its highest observed validation score at the final epoch, potential improvements beyond the 40-epoch limit remain unexamined. All selected checkpoints were subsequently used for final testing without adjustments based on test-set performance.

### 3.3. Segmentation Performance on the Official Test Set

The final evaluation of U-Net–ResNet34 was conducted on all 422 images in the official CSDD test set using the best checkpoint from each training run. Predictions were generated through sliding window inference and logit aggregation at the original image resolution. Evaluation metrics were computed from the global confusion matrix accumulated for each run and subsequently summarized using the mean and sample standard deviation across seeds. The results are presented in **Table 4**.

**Table 4.** U-Net–ResNet34 performance on the official CSDD test set

| Seed | Defect-mIoU (%) | Four-class mIoU (%) | Macro Dice (%) |
|---|---|---|---|
| 2027 | 59.700 | 69.750 | 74.303 |
| 2028 | 60.136 | 70.078 | 74.639 |
| 2029 | 60.176 | 70.108 | 74.676 |
| Mean ± SD | 60.004 ± 0.264 | 69.978 ± 0.198 | 74.539 ± 0.205 |

As presented in **Table 4**, U-Net–ResNet34 achieved a mean defect-mIoU of 60.004 ± 0.264%, a four-class mIoU of 69.978 ± 0.198%, and a macro Dice coefficient of 74.539 ± 0.205%. Furthermore, the mean macro precision and macro recall were 74.144 ± 0.316% and 74.961 ± 0.257%, respectively. The higher four-class mIoU was primarily attributable to the inclusion of the background class, which achieved an IoU of approximately 99.90%. Therefore, defect-mIoU was considered the primary performance indicator because it more directly reflects segmentation performance across the three defect categories. The highest defect-mIoU was obtained with seed 2029 at 60.176%, whereas the lowest was observed with seed 2027 at 59.700%, corresponding to a difference of 0.476 percentage points. All training runs were included in the performance summary to avoid selectively reporting only the best result. Compared with the mean validation defect-mIoU of 60.347 ± 0.162%, the mean test performance decreased by 0.343 percentage points. This difference indicates that aggregate test performance remained close to validation performance within the official CSDD partitions, but does not establish generalization to independent physical specimens or different industrial production conditions. For descriptive context, Mao et al. **[10]** reported a mean IoU of 54.29% across the three defect classes for their U-Net benchmark on CSDD. However, differences in image resolution, model configuration, and training and evaluation protocols prevent a direct performance comparison. Consequently, the numerical difference cannot be attributed to native-scale patch processing, defect-aware sampling, or any other individual component without controlled experiments. The benchmark results are therefore used for descriptive contextualization rather than as evidence of a controlled performance improvement.

### 3.4. Class-Wise Performance and Segmentation Error Analysis

Class-wise evaluation was conducted to identify differences in the ability of U-Net–ResNet34 to recognize individual defect categories. **Table 5** presents the mean IoU and its standard deviation across seeds, together with the mean precision, recall, and Dice/F1-score obtained from the three test runs.

**Table 5.** Class-wise defect segmentation performance on the test set

| Class | IoU (%) | Precision (%) | Recall (%) | Dice (%) |
|---|---|---|---|---|
| Scratch | 72.857 ± 0.204 | 84.922 | 83.686 | 84.297 |
| Spot | 58.084 ± 0.534 | 72.092 | 74.938 | 73.484 |
| Rust | 49.072 ± 0.070 | 65.420 | 66.261 | 65.836 |

As shown in **Table 5**, scratch achieved the highest IoU of 72.857%, followed by spot at 58.084% and rust at 49.072%. Scratch also exhibited relatively balanced precision and recall, whereas spot achieved higher recall than precision. In contrast, rust obtained the lowest values across all reported metrics. These class-wise performance differences are visualized in **Figure 2**.

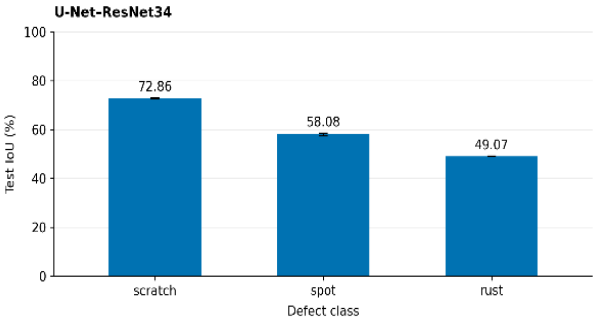



**Figure 2.** Mean IoU of the three defect classes on the U-Net–ResNet34 test set

**Figure 2** also displays the standard deviations across the three training runs. The difference between scratch and rust reached 23.785 percentage points. Although spot accounted for the smallest proportion of pixels in the dataset, its IoU remained higher than that of rust. This suggests that pixel frequency alone does not account for the observed class-wise IoU ranking. Defect appearance, similarity to surface structures, and boundary ambiguity may contribute, but their individual effects were not isolated experimentally. To examine image-level heterogeneity, class-wise IoU was averaged across the three seeds for each test image and summarized only for images containing the corresponding ground-truth class. Scratch was present in all 422 images and had a median IoU of 75.50% (Q1–Q3: 67.59–80.63%). Spot was present in 393 images and had a median of 64.59% (47.03–78.91%), whereas rust was present in 233 images and had a median of 52.16% (37.72–61.53%). These distributions reveal image-to-image variability not captured by the global IoU values, although the summaries remain descriptive and do not establish its underlying causes. To further investigate classification errors, a confusion matrix was normalized row-wise according to the number of ground-truth pixels belonging to each class. The normalized confusion matrices from the three training runs were then averaged, as presented in **Figure 3**.

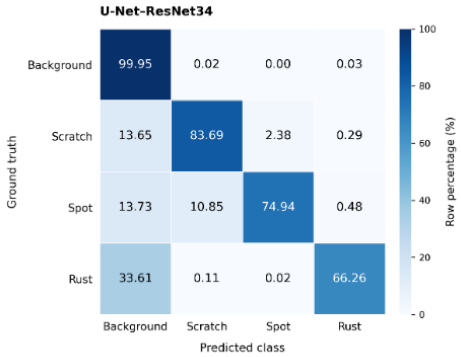



**Figure 3.** Mean normalized confusion matrix of U-Net–ResNet34 on the CSDD test set

As shown in **Figure 3**, the diagonal values of the row-normalized confusion matrix indicate correct classification of approximately 99.95% of background pixels, 83.69% of scratch pixels, 74.94% of spot pixels, and 66.26% of rust pixels. The most prominent error for rust occurred when 33.61% of its pixels were classified as background. Meanwhile, spot misclassification included 13.73% of pixels predicted as background and 10.85% predicted as scratch. Misclassification as background was the dominant observed error for ground-truth rust pixels, while spot also showed confusion with scratch; these patterns describe pixel-level errors rather than their underlying causes. Although the background class achieved a very high recognition rate, its overwhelming pixel

frequency means that even a small proportion of errors may correspond to a considerable number of false-positive defect predictions. Therefore, segmentation performance should not be interpreted solely from aggregate metrics but should also incorporate class-wise evaluation and error analysis. For rust, recall of 66.261% reflects missed ground-truth pixels, whereas precision of 65.420% indicates false-positive predictions. Such errors could lead to overlooked defects or unnecessary inspections, although their operational consequences were not evaluated in this study. These findings further indicate that improving rust recognition is an important direction for future research.

### 3.5. Qualitative Segmentation Results

Qualitative evaluation was conducted to complement the quantitative findings by comparing original images, ground-truth masks, and U-Net–ResNet34 predictions at corresponding spatial locations. Visualizations were obtained using the checkpoint from seed 2027, which was designated in advance rather than selected based on the highest test-set performance. Four images were examined to illustrate contrasting segmentation outcomes: images 1249 and 2164 for rust-related errors, and images 0007 and 0848 for scratch and spot segmentation. These examples were purposively selected from six previously archived visualization previews based on their defect classes and observed error characteristics.

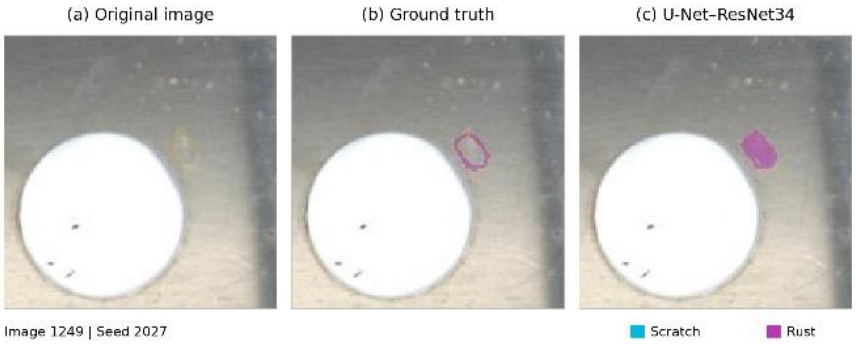



**Figure 4.** Comparison of (a) the original image, (b) ground truth, and (c) U-Net–ResNet34 prediction for image 1249 using seed 2027

As illustrated in **Figure 4**, the rust region in the ground truth exhibits a ring-like structure, whereas the U-Net–ResNet34 prediction covers a larger area and tends to fill the interior of the annotated region. This difference indicates over-segmentation, in which the predicted defect region extends into pixels that are not labeled as rust in the reference annotation. Although the defect location was identified, the discrepancy in spatial coverage still resulted in pixel-level segmentation errors. This observation demonstrates that correctly locating a defect does not necessarily imply accurate delineation of its shape and boundaries.

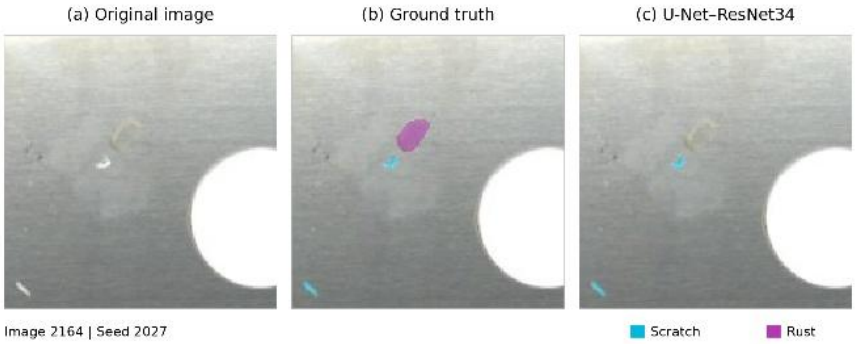



**Figure 5.** Comparison of (a) the original image, (b) ground truth, and (c) U-Net–ResNet34 prediction for image 2164 using seed 2027

In contrast to **Figure 4**, **Figure 5** shows a rust region clearly identified in the ground truth but not predicted as rust by U-Net–ResNet34, resulting in an image-level rust IoU of 0.00%. Meanwhile, several scratch regions remain identifiable in the prediction. This example illustrates a missed rust region, consistent with the confusion matrix finding that 33.61% of ground-truth rust pixels were classified as background across the test set (**Figure 3**).

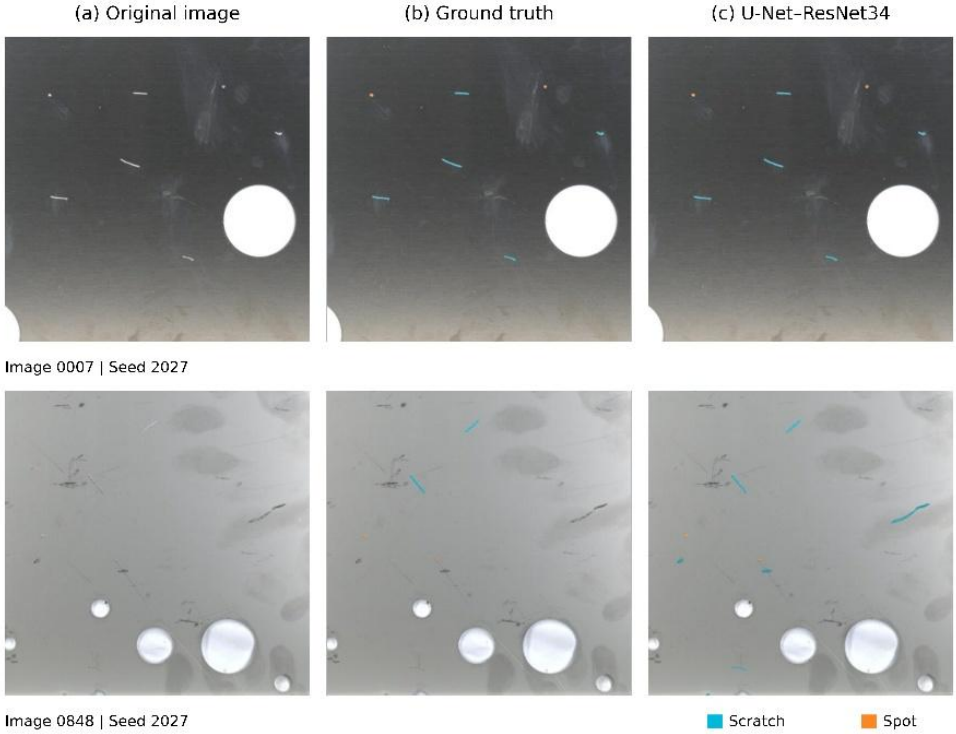



**Figure 6.** Comparison of (a) the original image, (b) ground truth, and (c) U-Net–ResNet34 prediction for images 0007 (top) and 0848 (bottom) using seed 2027.

As illustrated in **Figure 6**, the predicted scratch regions in image 0007 largely coincide with the cyan ground-truth annotations, while the orange spot predictions also appear at corresponding locations. Minor spatial differences remain, but the image achieved IoUs of 86.34% for scratch and 84.38% for spot. In contrast, image 0848 shows additional cyan scratch predictions, particularly along a surface streak on the right side and in lower regions that are not marked as scratch in the displayed ground truth. Although several orange spot predictions appear near their annotated locations in the displayed crop, the full-resolution confusion matrix reveals additional false-positive spot pixels. False positives exceeded false negatives for both scratch and spot, contributing to their

lower IoUs of 31.66% and 44.06%, respectively. Overall, the selected examples illustrate relatively accurate defect localization, over-segmentation, false-positive predictions, and missed defects. However, these visualizations are illustrative and do not establish the prevalence or underlying causes of these errors across the test set. The quantitative evaluation of all 422 full-resolution test images remains the primary basis for the reported findings.

### 3.6. Computational Requirements and Limitations

Computational performance was evaluated on an NVIDIA GeForce RTX 5070 GPU using bfloat16 mixed precision and an inference batch size of one. U-Net–ResNet34 comprised 24,436,804 trainable parameters and processed each 3,648 × 3,648-pixel image using 36 overlapping patches. Across three independent runs on the 422-image test set, the mean inference time was 1.1732 $\pm$ 0.0038 seconds per image. The mean peak allocated and reserved CUDA memory were 317.22 MiB and 438.00 MiB, respectively. Inference time included patch extraction, normalization, data transfers, model execution, logit aggregation, and full-resolution mask reconstruction, but excluded image decoding, model loading, warm-up, metric computation, and file output. The reported memory values represent PyTorch CUDA allocator measurements rather than total GPU memory consumption. Therefore, these results characterize computational requirements under the tested configuration and do not establish end-to-end deployment latency or real-time performance in industrial environments.

## 4. CONCLUSION

This study evaluated the performance of U-Net–ResNet34 for semantic segmentation of surface defects using the Casting Surface Defect Dataset (CSDD) through patch-based training, defect-aware sampling, and sliding window inference at the original image resolution. Based on three independent training runs with different random seeds, the model achieved a mean defect-mIoU of 60.004 $\pm$ 0.264% on the official test set of 422 images, with a four-class mIoU of 69.978 $\pm$ 0.198% and a macro Dice coefficient of 74.539 $\pm$ 0.205%. Class-wise evaluation showed that scratch achieved the highest IoU of 72.857%, followed by spot at 58.084% and rust at 49.072%. Quantitative analysis identified rust as the lowest-performing defect class, with an average of 33.61% of its ground-truth pixels classified as background. Image-wise IoU distributions revealed heterogeneous segmentation performance, particularly for spot and rust, while selected qualitative examples illustrated accurate defect localization, false-positive predictions, missed rust regions, and over-segmentation. These findings highlight the importance of defect-focused evaluation in the presence of severe pixel-level class imbalance. The findings are restricted to the official CSDD partitions; physical-specimen independence across splits was not established, controlled ablations and evaluations under different industrial production conditions were not performed, and matched-protocol comparisons with alternative architectures were not reported in this article. Future research could focus on improving rust segmentation performance, conducting controlled ablations and dedicated model comparisons, and assessing model generalization using data from more diverse industrial environments.

**Funding Information**

The authors declare that this research received no specific funding from any public, commercial, or non-profit funding agency.

## Author Contributions Statement

First author: Conceptualization, methodology, data curation, formal analysis, investigation, visualization, and writing – original draft. Second author and Third author: Supervision, methodology, validation, and writing – review & editing. All authors reviewed and approved the final manuscript.

## Conflict of Interest Statement

The authors declare that they have no financial or non-financial conflicts of interest that could have influenced the conduct of this research or the reporting of its findings.

## Data Availability

The CSDD dataset used in this study was obtained directly from its developers. Dataset documentation and instructions for requesting access are available at https://github.com/Kerio99/CSDD . The repository does not provide a direct download of the original image and annotation files, and dataset access remains subject to the provider's terms. The study-specific implementation and experimental configurations were maintained locally; no public repository or archived release of these materials is provided with this manuscript. Supporting analytical results are reported in the article.

## REFERENCES


[1] Z. Ren, F. Fang, N. Yan, and Y. Wu, "State of the Art in Defect Detection Based on Machine Vision," *Int. J. Precis. Eng. Manuf. - Green Technol.*, vol. 9, no. 2, pp. 661–691, 2022, doi: 10.1007/s40684-021-00343-6.

[2] Y. Chen, Y. Ding, F. Zhao, E. Zhang, Z. Wu, and L. Shao, "Surface Defect Detection Methods for Industrial Products: A Review," *Appl. Sci.*, vol. 11, no. 16, 2021, doi: 10.3390/app11167657.

[3] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Springer, 2015, pp. 234–241. doi: 10.1007/978-3-319-24574-4_28.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[5] D. Tabernik, S. Šela, J. Skvarč, and D. Skočaj, "Segmentation-based deep-learning approach for surface-defect detection," *J. Intell. Manuf.*, vol. 31, no. 3, pp. 759–776, 2020, doi: 10.1007/s10845-019-01476-x.

[6] R. Neven and T. Goedemé, "A Multi-Branch U-Net for Steel Surface Defect Type and Severity Segmentation," *Metals (Basel).*, vol. 11, no. 6, 2021, doi: 10.3390/met11060870.

[7] S. Ashrafi, S. Teymouri, S. Etaati, J. Khoramdel, Y. Borhani, and E. Najafi, "Steel surface defect detection and segmentation using deep neural networks," *Results Eng.*, vol. 25, 2025, doi: 10.1016/j.rineng.2025.103972.

[8] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, "MVTec AD — A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9592–9600. doi: 10.1109/CVPR.2019.00982.

[9] J. Božič, D. Tabernik, and D. Skočaj, "Mixed supervision for surface-defect detection: From weakly to fully supervised learning," *Comput. Ind.*, vol. 129, 2021, doi: 10.1016/j.compind.2021.103459.

[10] K. Mao, P. Wei, Y. Wang, M. Liu, S. Wang, and N. Zheng, "CSDD: A Benchmark Dataset for Casting Surface Defect Detection and Segmentation," *IEEE/CAA J. Autom. Sin.*, vol. 12, no. 5, pp. 947–960, 2025, doi: 10.1109/JAS.2025.125228.

[11] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Networks*, vol. 106, pp. 249–259, 2018, doi: 10.1016/j.neunet.2018.07.011.

[12] Z. Xie, C. Shu, Y. Fu, J. Zhou, and D. Chen, "Balanced Loss Function for Accurate Surface Defect Segmentation," *Appl. Sci.*, vol. 13, no. 2, 2023, doi: 10.3390/app13020826.

[13] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. J. Cardoso, "Generalised Dice Overlap as a Deep Learning Loss Function for Highly Unbalanced Segmentations," in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, 2017. doi: 10.1007/978-3-319-67558-9_28.

[14] F. Isensee, P. F. Jaeger, S. A. A. Kohl, J. Petersen, and K. H. Maier-Hein, "nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation," *Nat. Methods*, vol. 18, no. 2, pp. 203–211, 2021, doi: 10.1038/s41592-020-01008-z.

[15] G. A. Reina, R. Panchumarthy, S. P. Thakur, A. Bastidas, and S. Bakas, "Systematic Evaluation of Image Tiling Adverse Effects on Deep Learning Semantic Segmentation," *Front. Neurosci.*, vol. 14, 2020, doi: 10.3389/fnins.2020.00065.

[16] A. A. Taha and A. Hanbury, "Metrics for evaluating 3D medical image segmentation: Analysis, selection, and tool," *BMC Med. Imaging*, vol. 15, no. 1, 2015, doi: 10.1186/s12880-015-0068-x.

[17] A. Reinke *et al.*, "Understanding metric-related pitfalls in image analysis validation," *Nat. Methods*, vol. 21, no. 2, pp. 182–194, 2024, doi: 10.1038/s41592-023-02150-0.

[18] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *J. Big Data*, vol. 6, 2019, doi: 10.1186/s40537-019-0197-0.

[19] S. Jumaboev, D. Jurakuziev, and M. Lee, "Photovoltaics Plant Fault Detection Using Deep Learning Techniques," *Remote Sens.*, vol. 14, no. 15, 2022, doi: 10.3390/rs14153728.

[20] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, 2015, doi: 10.1007/s11263-015-0816-y.

[21] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," in *International Conference on Learning Representations (ICLR)*, 2019. [Online]. Available: https://openreview.net/forum?id=Bkg6RiCqY7

[22] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation," in *2016 Fourth International Conference on 3D Vision (3DV)*, 2016. doi: 10.1109/3DV.2016.79.

[23] I. Loshchilov and F. Hutter, "SGDR: Stochastic Gradient Descent with Warm Restarts," in *International Conference on Learning Representations (ICLR)*, 2017. [Online]. Available: https://openreview.net/forum?id=Skq89Scxx