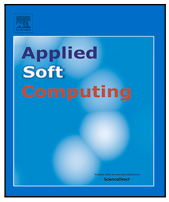
Applied Soft Computing Journal 165 (2024) 112138

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc



Learning trustworthy model from noisy labels based on rough set for surface defect detection

Tongzhi Niu [a], Zhenrong Wang [a], Weifeng Li [a], Kai Li [b], Yuwei Li [a], Guiyin Xu [c], Bin Li [a,*]

[a] School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Luoyu Road 1037, Wuhan, 430074, Hubei, China
[b] College of Mechanical and Electrical Engineering, Central South University, Changsha, 410083, Hunan, China
[c] Department of Mechanical Engineering, National University of Singapore, 117575, Singapore

ARTICLE INFO

Keywords:
Trustworthy model
Surface defect detection
Noisy label
Rough set
Bayesian neural networks

ABSTRACT

In surface defect detection, some regions remain ambiguous and cannot be distinctly classified as abnormal or normal. This challenge is exacerbated by subjective factors, including workers' emotional fluctuations and judgment variability, resulting in noisy labels that lead to false positives and missed detections. Current methods depend on additional labels, such as clean and multi-labels, which are both time-consuming and labor-intensive. To address this, we utilize Rough Set theory and Bayesian neural networks to learn a trustworthy model from noisy labels for Surface Defect Detection. Our approach features a novel pixel-level representation of suspicious areas using lower and upper approximations, and a novel loss function that emphasizes both precision and recall. The Pluggable Spatially Bayesian Module (PSBM) we developed enhances probabilistic segmentation, effectively capturing uncertainty without requiring extra labels or architectural modifications. Additionally, we have devised a 'defect discrimination confidence' metric to better quantify uncertainty and assist in product quality grading. Without the need for extra labeling, our method significantly outperforms state-of-the-art techniques across three types of datasets and enhances seven types of classic networks as a pluggable module, without compromising real-time computing performance. For further details and implementation, our code is accessible at https://github.com/ntongzhi/RoughSet-BNNs.

1. Introduction

Surface defect inspection is vital in manufacturing, widely applied in sectors such as semiconductor electronics, automotive, pharmaceuticals, and chemicals. Recently, deep learning have excelled in identifying surface defects [1–4]. However, because deep learning models are data-driven, they are susceptible to replicating or amplifying human errors and biases present in the training datasets during the data labeling process [5–8].

Two primary challenges arise in the annotation task: (1) Objective challenges: Some suspicious regions cannot be uniquely classified as abnormal or normal, such as those with weak features or border regions. In many cases, the defect and background are the same material, with very similar colors and textures. The boundary between the defect and the background is usually not a distinct line but a gradient region. (2) Subjective challenges: The labeling of weak feature and border regions is easily affected by factors including workers' unstable emotions, judgment standards, and technique levels, resulting in noisy labels.

The indistinct nature of these suspicious regions leads to noisy labels, manifested in two ways: over-labeling, where suspicious regions are erroneously annotated as anomalies, and under-labeling, where they are incorrectly marked as normal. It is common for suspicious regions with identical characteristics to be inconsistently labeled across different samples, which is the primary characteristic of noisy labels in surface defect detection.

This inconsistency in noisy labels hinders the learning of robust representations for suspicious regions. As illustrated in Fig. 1, the model trained on noisy labels exhibits two types of errors: (1) False detection: The model overfits to over-labeled suspicious regions in the noisy labels, leading to the erroneous detection of certain regions as anomalies, even though they are annotated as normal, as exemplified in rows (a) and (b). (2) Missed detection: Conversely, the model underfits and fails to recognize some suspicious regions as anomalies, despite their annotation as such in the labels, as indicated in rows (c) and (d). Accurate measurement of the geometric dimensions of abnormal regions, such as length or diameter, is crucial for determining defectiveness. However, both false and missed detections result in the imprecise measurement of these dimensions, leading to inconsistent evaluations of product quality.

* Corresponding author.
E-mail address: libin999@hust.edu.cn (B. Li).

https://doi.org/10.1016/j.asoc.2024.112138
Received 7 January 2024; Received in revised form 9 August 2024; Accepted 12 August 2024
Available online 24 August 2024

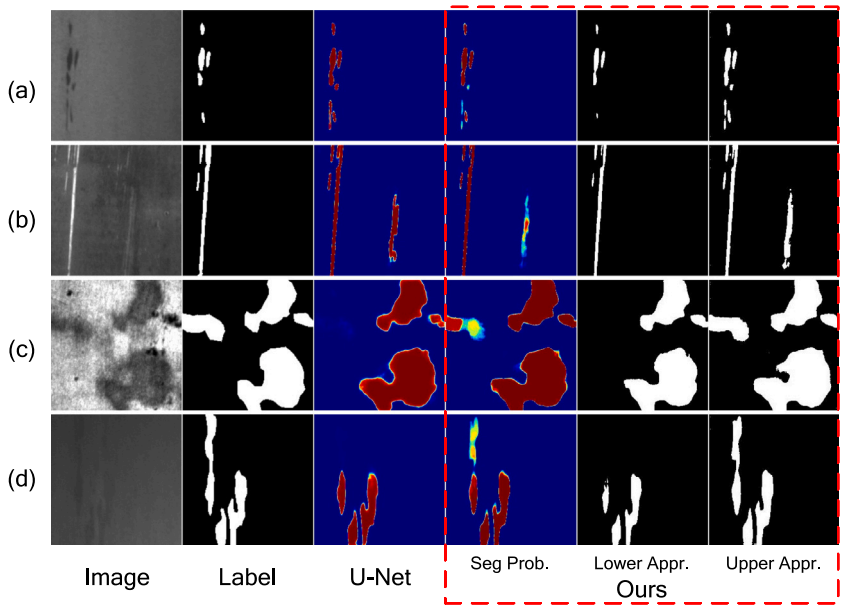


Fig. 1. Results of our model learning from noisy labels. As can be seen from row (a) and (b), the suspicious regions are annotated as normal in the labels, but are falsely detected as abnormal in the U-net. In contrast, as shown in the row (c) and (d), the suspicious regions are marked as abnormal but detected as normal. In our methods, U-net based BNNs, suspicious regions are represented by segmentation probabilities. The lower approximations are same as the labels in false detections, and the upper approximations are same as the labels in missing detections.

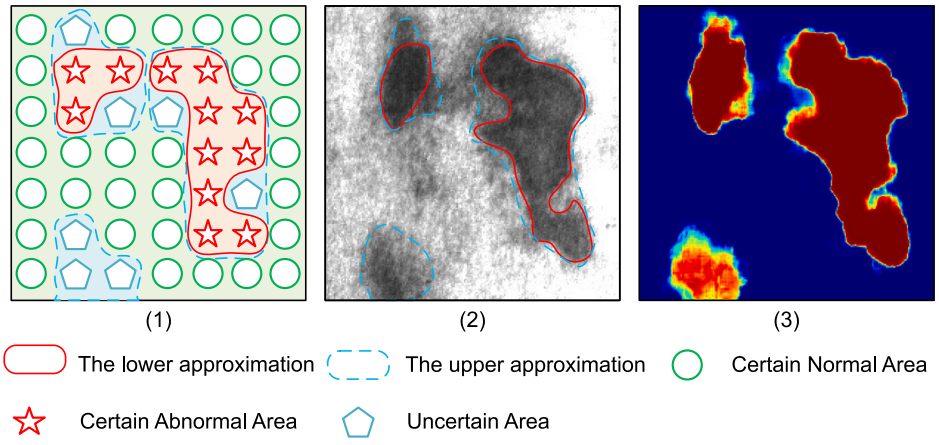


Fig. 2. Uncertain model of noisy label based on Rough Set. The suspicious regions are described by lower and upper approximations in (1) and (2). The segmentation probability that is solvable for neural networks is used to characterize uncertain regions in (3).

To learn from noise labels, several robust architectures have been developed, including regularization method [9,10], robust loss functions [11,12], and sample selection methods [13,14]. However, most of these approaches focus primarily on image-level labels. In contrast, for pixel-level noisy labels, current methods aim to mitigate their negative effects using evaluation-based and correction-based techniques. Evaluation-based strategies such as ADL [15] and Pick-and-Learn [16] adjust the training loss based on the quality of image-level labels but lack the granularity offered by a pixel-level approach. Additionally, the design of effective evaluation strategies is complicated by the typical inconsistencies, rather than outright errors, associated with noisy labels in defect detection. Correction-based methods like GAT [17], WSSS [18], MV-DAR [6], and MTCL [7] often depend on supplementary label information, such as a few clean labels, intermediate labeling variables, and multi-labels. These requirements are both time-consuming and labor-intensive, particularly due to the difficulty in distinguishing between suspicious and true labels.

Firstly, the suspicious regions are defined as uncertain areas, distinct from the normal and abnormal regions, and represent them with two precise boundary lines [19]: the lower and upper approximations, as shown in Fig. 2. The lower approximation encompasses all regions that are definitively anomalous, while the upper approximation includes potential anomalies. These uncertain regions comprise pixels that cannot be uniquely classified using available features. Specifically, we employ neural network-derived segmentation probabilities to denote uncertain regions, where a pixel's value indicates its probability of abnormality, as depicted in Fig. 2(3). Ultimately, these approximations are precise and free from uncertainty and inconsistency. Therefore, inspired by Tversky loss [20], we have restructured the loss function so that the lower approximation calculates the precision penalty, and the upper approximation determines the recall penalty.

Secondly, the uncertainty of segmentation probabilities is captured using Bayesian neural networks (BNNs). Inspired by Dropout-based BNNs [21–24] and building on DropBlock [25], we introduce the PSBM,

which drops contiguous regions from a layer's feature map rather than independent random units. We investigate the optimal application of PSBM blocks in constructing BNNs without modifying the existing network architecture. Specifically, we explore the appropriate number and placement of PSBM blocks. During training, the uncertain regions identified by the variance of multiple BNN outputs are utilized for label correction. In testing, the intersection and union of multiple model outputs are used to determine the lower and upper approximations, respectively.

To this end, we introduce the concept of 'defect discrimination confidence' as a metric to gauge the uncertainty in identifying anomalies as defects. Within the framework of segmentation probability, various geometric dimensions are determined based on differing probability levels. These dimensions are then compared against thresholds established by national, industry, or factory standards. When the geometric dimensions fall below the threshold, the confidence is set at 0%. Conversely, if the dimensions exceed the threshold, the confidence level reaches 100%. For cases where the threshold lies within the range of geometric dimensions, the confidence is assigned the corresponding probability value of the threshold. This confidence level can be leveraged in production to classify products. Factories can adjust the confidence threshold in accordance with their specific requirements, thereby achieving consistent and trustworthy judgment outcomes.

In summary, this paper introduces a novel framework for learning a trustworthy model from noisy labels without the need for additional labeling or alterations to the network structure. Our key contributions are as follows:

(1) **Representation of Suspicious Regions**: We define suspicious regions as uncertain areas with precise lower and upper approximations using Rough Set theory, capturing the ambiguity in defect detection.

(2) **Redesigned Loss Function**: Inspired by Tversky loss, we redesign the loss function to incorporate precision and recall penalties based on the lower and upper approximations.

(3) **PSBM**: We introduce the PSBM, which drops contiguous regions from a layer's feature map to capture segmentation uncertainty effectively without modifying the existing network architecture.

(4) **Defect Discrimination Confidence**: This metric quantifies uncertainty in defect identification, allowing for reliable quality control and product classification based on adjustable confidence thresholds.

Our proposed method offers several key advantages:

(1) **Efficient Use of Noisy Labels**: No additional label information is required, saving data labeling costs and fully utilizing limited data.

(2) **Redesigned Loss Function**: Incorporates precision and recall penalties to enhance the model's ability to focus on relevant features while minimizing false positives and false negatives.

(3) **Pluggable Module**: The PSBM can be easily integrated into existing defect detection systems, enhancing their performance without requiring significant changes to the underlying architecture.

(4) **Scalability and Adaptability**: Our method outperforms state-of-the-art techniques across multiple datasets and can enhance various classic networks, demonstrating its scalability and adaptability to different industrial scenarios.

2. Related works

Our proposed method focuses on learning trustworthy models from noisy labels, effectively addressing the uncertainty and inconsistency arising from suspicious regions in surface defect detection. The method incorporates both Rough Set theory and Bayesian Neural Networks (BNNs). This section provides a brief review of the development of Rough Set theory and BNNs. Subsequently, we discuss the latest research on learning from noisy labels.

2.1. Development of rough set and Bayesian neural networks

The fundamental concepts of Rough Sets and approximation spaces were introduced by Pawlak [19]. Since their inception, Rough Sets have been recognized as a potent mathematical tool for analyzing objects with vague descriptions, characterized by inconsistency and ambiguity. Any subset X of a universe U can be defined either precisely or approximately using elementary sets. In the approximate approach, X is characterized by two distinct sets: the lower and upper approximations. The lower approximation of X comprises all elementary sets contained within X, while the upper approximation includes all elementary sets that intersect with X. The boundary region of a Rough Set, delineated by the difference between its upper and lower approximations, contains elements whose membership in X is uncertain. In our research, we similarly express the suspicious region in an approximate manner. The lower approximation encompasses all regions that definitely belong to defects, whereas the upper approximation includes all potential defect regions.

BNNs [26] addresses challenges in Bayesian neural networks by defining prior distributions that converge to Gaussian or non-Gaussian processes as network size increases and demonstrates the effectiveness of hybrid Monte Carlo methods for posterior integration. Nonetheless, due to the immense parameter count, modeling a distribution over these kernels poses a significant challenge and incurs additional computational costs. Dropout-based BNNs [23] demonstrated that applying dropout in neural networks equates to a Bayesian approximation of Gaussian processes, without compromising computational efficiency or test accuracy. Concurrently, Yarin [21] introduces a Bayesian CNN that reduces overfitting on small data by using probability distributions over kernels and Bernoulli variational distributions, significantly improving classification accuracy. Alex [22] developed a deep learning framework for probabilistic semantic segmentation, known as Bayesian SegNet. Building upon these foundations, our paper proposes a surface defect detection network based on BNNs, extending previous work in the field.

2.2. Latest research on learning from noisy labels methods

Noisy labels in real-world datasets are estimated to range from 8.0% to 38.5% [27], making learning from noisy labels an increasingly critical task. To address this, robust architectures [28–31] have been developed, featuring noise adaptation layers atop the softmax layer and bespoke architectures. Robust regularization methods [9,10] enhance resistance to label noise using standard techniques such as data augmentation, weight decay, and dropout. To minimize risk on unseen clean data, robust loss functions [11,12] have been designed, even in the presence of noisy training data. Sample selection methods [13,14] offer strategies to sift true-labeled examples from noisy datasets. However, for defect detection, we concentrate on segmentation tasks, where these image-level noisy label methods are not directly applicable.

As shown in Table 1, at the pixel level, methods for learning from noisy labels can be categorized into four types: correcting noisy labels, filtering noisy labels, redesigning loss functions, and blurring label edges. Correcting noisy labels-based methods, such as GAT [17], use a Graph Attention Network to propagate information from clean labels within a superpixel-based graph, while WSSS [18] employs an L1-optimization based sparse learning model with an intermediate labeling variable. However, these methods require additional label information, which is time-consuming and labor-intensive. Filtering noisy labels-based methods, like ADL [15], propose adaptive denoising strategies to detect and remove high-loss noisy labels during training, and Pick-and-Learn [16] uses a quality awareness module to assess label quality and re-weight samples. Yet, in surface defect detection, noisy labels are common, and filtering them means not fully utilizing the limited data and labels. Redesigning loss functions-based methods, such as the Noise-Robust Dice Loss [32,33], combine MAE loss and Dice loss to address noisy labels but have limited effectiveness in surface defect

Table 1
Current literature summary.

Study	Method	Contribution	Limitations	Our contribution
GAT [17]	Correct noisy labels	Use a Graph Attention Network to propagate information from clean labels within a superpixel-based graph.	Require additional label information, which is time-consuming and labor-intensive.	Identifies inaccurate information as uncertain regions using rough sets instead of correcting noisy labels, which do not require additional label information
WSSS [18]		Employs an L1-optimization based sparse learning model with an intermediate labeling variable.		
ADL [15]	Filter noise labels	Propose adaptive denoising strategies to detect and remove high-loss noisy labels during training.	Noisy labels are common, and filtering them means not fully utilizing the limited data and labels.	Maximizes limited data, using rough sets and redesigned loss functions to leverage both certain and uncertain information in noisy labels.
Pick-and-Learn [16]		Uses a quality awareness module to assess label quality and re-weight samples.		
Noise-Robust Dice Loss [32,33]	Redesign loss function	Combine MAE loss and Dice loss to address noisy labels.	Limited effectiveness in surface defect detection.	Redesigns the loss function, incorporating precision and recall penalties.
Gaussian edge softening [34,35]	Blurring label edges	Reduce noise and improve performance for uncertain ground truth by smoothing labels.	Over-smoothing can result in the loss of valuable label information.	Using upper and lower approximations allows model to take into account both valuable information.

detection. Blurring label edges-based methods, like Gaussian edge softening [34,35], reduce noise and improve performance for uncertain ground truth by smoothing labels. However, this can cause the model to underfit, and over-smoothing can result in the loss of valuable label information.

In this paper, instead of correcting or filtering noisy labels, we define the inaccurate information in noisy labels (suspicious regions in the image) as uncertain areas with precise lower and upper approximations using Rough Set theory. We then extend BNNs and redesign the loss function to capture the uncertainty in noisy labels, enabling the model to learn a confidence metric from the noisy labels.

3. Proposed methodology

3.1. Overall framework

The proposed methodology comprises three major components: (1) the representation of noisy labels and the redesigned loss function; (2) the Pluggable Spatially Correlated Bayesian Module (PSBM) and its application modes; (3) the method for calculating 'defect discrimination confidence'.

Initially, we will explore the fundamental concepts of representing noisy labels based on Rough Set theory and introduce the redesigned loss functions.

Next, we will elucidate the probabilistic model and PSBM. To integrate PSBM into various customized models effectively, we will present adaptable application modes. This section will also detail the training mechanism and the process for inferring segmentation probabilities. To illustrate, we will use U-net as an example to demonstrate the development of a U-net based Bayesian Neural Network (BNN).

Lastly, the method for calculating 'defect discrimination confidence' will be presented, detailing how it contributes to the identification of defects.

3.1.1. Representation of noisy labels based on rough set

Let us examine a straightforward framework for representing knowledge with noisy labels, where a finite collection of regions is characterized by a finite collection of attributes. This concept can be formally defined through an information system denoted as S:

$$S = (U, A) \tag{1}$$

In this context, U represents a finite, nonempty set containing all the regions within the image, while A denotes a finite, nonempty set of attributes, such as texture, grayscale, and others.

Definition 1 (*Indiscernible Relation*)**.** In our system, each region in the set U can be described using attributes from the set A. The indiscernible relation $ind(B)$ for a subset of attributes $B \subseteq A$ groups regions together if they share the same values for these attributes. It is defined as follows:

$$ind(B) = \{(x, y) \in U \times U \mid \text{all attributions in } B \text{ are the same for } x \text{ and } y\} \tag{2}$$

This means that any two regions x and y are in relation $ind(B)$ if they cannot be distinguished by any attributes in B. We denote the group of all regions indistinguishable from a region x as $[x]$.

Definition 2 (*Upper and Lower Approximation Sets*)**.** For an anomaly region $R \subseteq U$, the lower and upper approximation sets are defined as follows:

$$\overline{apr}(R) = \{r \in U \mid [r] \cap R = \emptyset\} \tag{3}$$

$$\underline{apr}(R) = \{r \in U \mid [r] \subseteq R\} \tag{4}$$

where $[r]$ represents the equivalence class of r.

Definition 3 (*Anomaly, Boundary, and Normally Regions*)**.** The collection of all equivalence classes is referred to as the quotient set of U, denoted as $U/A = [r] \mid r \in U$. This allows the universe to be partitioned into three distinct regions: the anomaly region, the boundary region, and the normal region.

$$ANO(R) = \underline{apr}(R) \tag{5}$$

$$NOR(R) = U - \overline{apr}(R) \tag{6}$$

$$BND(R) = \overline{apr}(R) - \underline{apr}(R) \tag{7}$$

If a region $r \in ANO(R)$, it certainly belongs to the anomaly set R. If a region $r \in NOR(R)$, it certainly does not belong to R. If a region $r \in BND(R)$, it is indeterminate whether r belongs to R. Hence, the suspicious regions are represented by $BND(R)$.

Definition 4 (*Segmentation Probability*)**.** To achieve a solvable representation for neural networks, we define the segmentation probability to characterize the suspicious regions, where the pixel value indicates the likelihood of the pixel being anomalous.

$$\underline{apr}(R) = \{v \in V \mid v = 1\} \tag{8}$$

$$\overline{apr}(R) = \{v \in V \mid v > 0\} \tag{9}$$

Here, v represents the pixel value, and V is the set of all pixels in the image

3.1.2. Redesigned loss function

Despite the inconsistency present in the suspicious regions, the lower and upper approximations remain consistent. Drawing inspiration from Tversky loss [20], we have reformulated the loss function as follows:

$$\begin{aligned} Loss = 1 &- \alpha \frac{\left|\underline{apr}(Y) \cap \mathcal{Y}\right|}{\left|\underline{apr}(Y) \cap \mathcal{Y}\right| + \left|\mathcal{Y} - \underline{apr}(Y)\right|} \\ &- \beta \frac{\left|\overline{apr}(Y) \cap \mathcal{Y}\right|}{\left|\overline{apr}(Y) \cap \mathcal{Y}\right| + \left|\overline{apr}(Y) - \mathcal{Y}\right|} \end{aligned} \tag{10}$$

Here, Y denotes the label, $\mathcal{Y}$ represents the output of the neural networks, and $|\cdot|$ means the cardinality of a set, with $\alpha + \beta = 1$. The lower approximation is utilized to compute the precision penalty, while the upper approximation is employed to determine the recall penalty.

3.2. The methods and application of pluggable spatially Bayesian module

3.2.1. Probabilistic modeling

The training data and labels are $X = \{x_1, \ldots, x_N\}$ and $Y = \{y_1, \ldots, y_N\}$, and we try to determine a posterior distribution $p(f \mid X, Y)$ by employing Bayesian approach, where $y = f(x)$ and our goal is to determine a function $y = f(x)$.

In BNNs, the objective is to determine the posterior distribution of the convolutional weights:

$$w = (W_i)_{i=1}^{I} \tag{11}$$

Let W_i denote the weight of the ith layer in the convolutional neural networks. Directly obtaining $p(w \mid X, Y)$ is difficult, so we approximate it with $q(w)$ using Gaussian priors and Bernoulli random variables. Dropout probabilities $b_{i,j}$ and kernel parameters K_i define $q(W_i)$ for each layer.

$$\begin{aligned} &W_i = K_i \cdot diag([b_{i,j}]_{j=1}^{J}) \\ &b_{i,j} \sim Bernoulli(p_i) for\ i = 1, \ldots, I, j = 1, \ldots, J \end{aligned} \tag{12}$$

The $diag(\cdot)$ operator transforms vectors into diagonal matrices with the vector elements placed along the diagonals. The probability p_i is assigned a fixed value from a Bernoulli distribution, usually $p_i = 0.5$. Here, I represents the layer, and J denotes the neurons.

In optimizing the network, we seek to minimize the Kullback–Leibler (KL) divergence between two distributions:

$$KL(q(w) \parallel p(w \mid X, Y)) \tag{13}$$

Minimizing this KL divergence is equivalent to maximizing the *log evidence lower bound* (ELBO):

$$L_{IV} := \sum_{i=1}^{N} E(y_i, \hat{f}(x_i, \hat{w_i})) - KL(q(w) \parallel p(w)) \tag{14}$$

where $E(\cdot)$ is a softmax likelihood loss function, with $\hat{w_i} \sim q(w)$. As suggested by Ref. [23], we apply L_2 regularization to the weights:

$$L_{dropout} := \sum_{i=1}^{N} E(y_i, \hat{f}(x_i, \hat{w_i})) + \sum_{i=1}^{I} (\|W_i\|_2^2 + \|b_i\|_2^2) \tag{15}$$

Here, $\|\cdot\|_2^2$ denotes the square of the L_2 norm.

For network inference, we use Monte Carlo integration:

$$p(y^* | x^*, X, Y) \approx \frac{1}{T} \sum_{t=1}^{T} \hat{f}(x^*, \hat{w_t}) \tag{16}$$

Here, x^* and y^* are test inputs and outputs, and $\hat{w_t} \sim q(w)$. The hyperparameter T balances accuracy and computational cost.

Algorithm 1 PSBM

Require: Feature map: A_{input} of size (C, H, W); Convolution kernel: K of size (L, L); Probability: $p = 0.5$.

Ensure: Output feature map: A_{output}.

1: Bernoulli distribution probability of PSBM γ :
$$\gamma = ((1-p)/L^2) \cdot (W^2/(W - L + 1)^2)$$
2: Dropout probabilities b_i:
$$b_j \sim Bernoulli(\gamma) \text{ for } j = 1, ..., L^2$$
3: Randomly sample Mask M:
$$M = diag([b_j]_{j=1}^{L^2})$$
4: Get Block mask M_{Block} by max pooling, pooling size is (L, L), stride is $(1, 1)$, and padding is $(L/2, L/2)$:
$$M_{Block} = 1 - max\_pool(M)$$
5: Apply the Block Mask:
$$A_{output} = A_{input} \times M_{Block}$$
6: Normalize the features:
$$A_{output} = A_{output} \times cout(M_{Block})/cout\_ones(M_{Block})$$
7: return A_{output}

3.2.2. The PSBM and its application modes

Based on DropBlock [25], we have designed the PSBM. The pseudocode of PSBM is illustrated in Algorithm 1. To enhance computing efficiency, the entire PSBM process performs tensor calculations on the GPU. Specifically, the Mask M is generated using the *torch.Bernoulli* function, and the Block Mask M_{Block} is created through maximum pooling. Moreover, as both the probability p and feature A_{output} are normalized, the latter term in Eq. (15) can be considered as 0.

Our exploration focuses on applying PSBM in the construction of BNNs without altering the network structure. We address the following key considerations:

(1) The optimal number of PSBM applications. While Dropout randomly removes units from neural networks to prevent overfitting, it can also diminish the network's learning capacity. Excessive regularization may slow down the learning process. Therefore, we adhere to two principles: PSBM should not be applied to every network layer as a regularization method, and it should not coexist with other regularization methods within the same layer.

(2) The appropriate placement of PSBM. Given that standard semantic segmentation models typically comprise encoders and upsampling layers, we categorize network layers into low-level and high-level. Low-level layers extract basic features like edges and corners, while high-level layers capture more complex features, such as shapes and contextual relationships. Low-level features are consistent across different models, but applying PSBM to mask high-level features may prove more effective. Consequently, PSBM should ideally be implemented at the higher levels of the encoders.

3.2.3. Training mechanism

During training, the inputs of the training set are denoted as $\{x_1, \ldots, x_N\}$, with corresponding annotations $\{y_1, \ldots, y_N\}$. The segmentations of the ith, $\{\hat{y}_i^1, \ldots, \hat{y}_i^{\mathcal{T}}\}$, are computed $\mathcal{T}$ times using BNNs:

$$\hat{y}_i^t = \hat{f}(x_i, \hat{w}_t) \tag{17}$$

Initially, we compute the mean μ_i and variance σ_i of these segmentations:

$$\mu_i = \frac{1}{\mathcal{T}} \sum_{t=1}^{\mathcal{T}} (\hat{y}_i^t) \tag{18}$$

$$\sigma_i = \frac{1}{\mathcal{T}} \sum_{t=1}^{\mathcal{T}} (\hat{y}_i^t - \mu_i) \tag{19}$$

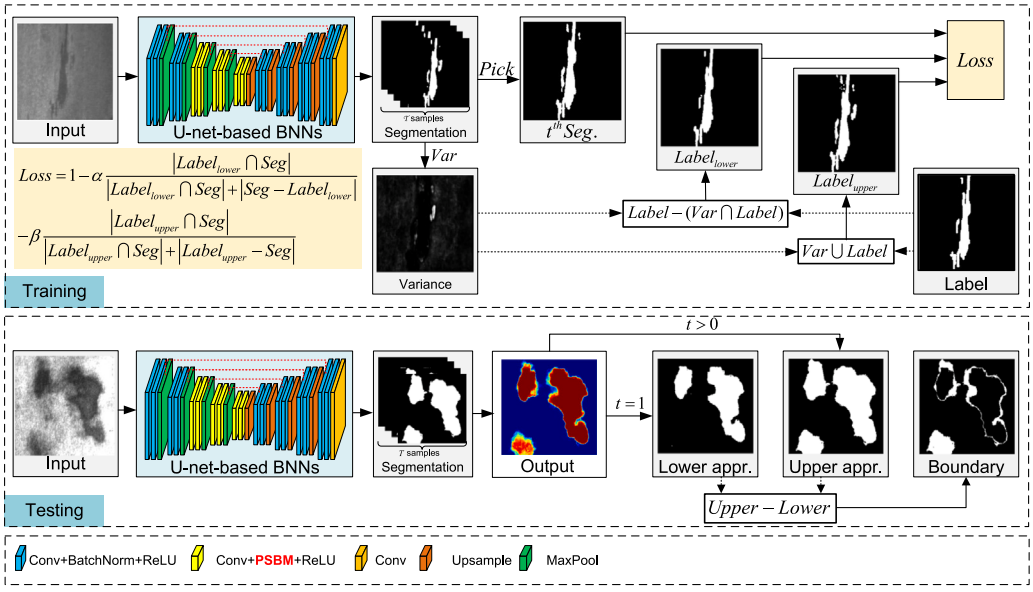


Fig. 3. An overview of optimization and inference of U-Net-based BNNs transformed by pluggable Bayesian modules.

The variance σ_i is then normalized at the pixel level:

$$\hat{\sigma}_i = (\sigma_i - min(\sigma_i))/(max(\sigma_i) - min(\sigma_i)) \tag{20}$$

Typically, the variance represents the uncertain part of the label, corresponding to the suspicious region. We therefore adjust the labels using these normalized variances:

$$\underline{apr}(y_i) = y_i - y_i \times \hat{\sigma}_i \tag{21}$$

$$\overline{apr}(y_i) = y_i + \hat{\sigma}_i \tag{22}$$

Finally, the loss function, based on Eqs. (10) and (15), is formulated as:

$$Loss = \frac{1}{N}\sum_{i=1}^{N}(1 - \alpha\frac{\underline{apr}(y_i) \times \hat{y}_i^t}{\underline{apr}(y_i) \times \hat{y}_i^t + (1 - \underline{apr}(y_i)) \times \hat{y}_i^t} - \beta\frac{\overline{apr}(y_i) \times \hat{y}_i^t}{\overline{apr}(y_i) \times \hat{y}_i^t + \overline{apr}(y_i) \times (1 - \hat{y}_i^t)}) \tag{23}$$

Here, $\alpha + \beta = 1$, and $\hat{y}_i^t$ is randomly selected from $\{\hat{y}_i^1, \ldots, \hat{y}_i^{\mathcal{T}}\}$.

3.2.4. Inference for segmentation probability

We use Monte Carlo integrations to approximate the segmentation probability. The inputs and outputs of the testing set are denoted as x^* and y^*, respectively. Following Eq. (16), the segmentation probability is calculated as:

$$p(y^*) \approx \frac{1}{T}\sum_{t=1}^{T}\hat{f}(x^*, \hat{w}_t) \tag{24}$$

3.2.5. U-net based Bayesian neural networks

To more clearly illustrate the optimization and inference processes of BNNs, we have designed the overall network structure based on U-net [36], as shown in Fig. 3. In line with the approach described in Section 3.2.2, we replace BatchNorm with PSBM in the last three layers of the encoder.

During training, the input is processed multiple times by the U-net-based BNNs to produce N sample segmentations (Seg). The noisy labels are then corrected using the variance (Var) of these multiple Segs, leading to the formation of the lower approximation ($Label_{lower}$) and upper approximation ($Label_{Upper}$). These, along with the ith Seg, are utilized to compute the loss function (Eq. (23)).

For testing, the segmentation probability (Output) is approximated using Monte Carlo integrations, as indicated in Eq. (24). Based on the Output, we determine the lower and upper approximations, as well as the boundary regions

3.3. Defect discrimination confidence

Now, we get the probability of each pixel in the image. Then, it is crucial to calculate the confidence of networks discriminations based on the probability. As we all know, the semantic segmentation results cannot be used directly to distinguish whether the sample is NG or not. The geometric dimensions of the defective region, such as length, diameter, etc., need to be counted. Finally, according to the threshold given by the national-, industry-, or factory-standard, it is determined whether it is defective or not. Therefore, we define the discriminant confidence based on the relationship between the threshold and the probability.

As illustrated in Fig. 4, we take different probabilities $\lambda \in [0, 1]$. According to Eqs. (8) and (9), when $\lambda = 1$, the lower approximation is obtained, while $\lambda > 0$, the upper approximation is given. And v is the value of each pixel of the segmentation. We define that the geometric dimensions (GD) of the defective region are calculated by

$$g(\lambda) = \int_{v=\lambda}^{v=1} GD(v)\, dv \tag{25}$$

And the threshold given by the national-, industry-, or factory-standard is represented by Λ. The confidence is calculated as follows:

$$C_{x^*}(\Lambda) = \begin{cases} 0\%, & g(0^+) < \Lambda \\ g^{-1}(\Lambda), & g(1) < \Lambda < g(0^+) \\ 100\%, & g(1) > \Lambda \end{cases} \tag{26}$$

where $g^{-1}(\Lambda)$ represent the inverse function of $g(\Lambda)$.

The 'defect discrimination confidence' function plays a crucial role in enhancing quality control by providing confidence levels for defect predictions, thereby assessing the reliability of detected defects. Although it is not utilized during the training phase, this metric is derived from the segmentation probability maps produced by the Bayesian Neural Network (BNN) after training. It is essential for evaluating the certainty of defect detections, which aids in quality grading and

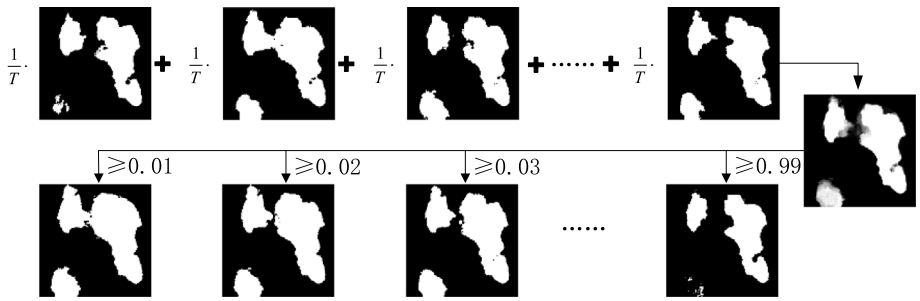


Fig. 4. Confidence evaluation method.

Table 2
Details of three datasets.

Dataset	Image numbers		Noise level	Open-source status
	Train	Test		
NEU-seg [37]	3630	840	Pixel-level	Yes
MCSD-seg [38]	532	134	Pixel-level	No
LC-seg [39]	502	126	Pixel-level	No

Table 3
Result of ablation example in NEU-Seg dataset.

	U-net	PSBM	Bayes	Loss	Recall	Precision	IoU
s1	√				0.8845	0.8455	0.7581
s2	√	√			0.8802	0.8563	0.7644
s3	√	√	√		0.9113	0.9175	0.7643
s4	√	√	√	√	**0.9350**	**0.9390**	**0.7670**

decision-making. Overall, by implementing 'defect discrimination confidence', the reliability of defect detection processes is significantly improved.

The 'defect discrimination confidence' metric provides confidence levels for defect predictions, helping assess the reliability of detected defects and aiding in quality control decisions. This metric is not used during training but is calculated from the segmentation probability maps produced by the Bayesian Neural Network (BNN) after training. It evaluates the certainty of defect detections, which aids in quality grading and decision-making. By using 'defect discrimination confidence', we improve the reliability of defect detection and enhance quality control processes.

4. Experiments

4.1. Implementation details

We set the base learning rate to 0.003 with a decay factor of 0.0001. The mini-batch size is 4, and the dropout probability is 50%. The method is implemented in PyCharm using PyTorch and trained on a server with an NVIDIA Tesla A100 GPU (40 GB memory) running CentOS 8 Linux.

4.2. Datasets

In this study, three datasets have been chosen to support and evaluate the proposed method's applicability and generality. These include one benchmark dataset (Open-source), NEU-seg [37], and two datasets (Not open-source) sourced from actual industrial production lines: MCSD-seg [38] for motor commutators, and LC-seg [39] for light chips. All images have been resized to a uniform dimension of 256×256 pixels. The division between training and testing sets is made randomly in an 8:2 ratio, as detailed in Table 2.

4.3. Evaluation metrics

Noise in labels makes it difficult to get clean data. We use the lower approximation for accuracy and the upper approximation for recall. To evaluate segmentation, we use the intersection-over-union (IoU) metric, comparing our method with other segmentation approaches.

4.4. Ablation experiment

In this study, we have implemented the following key improvements at the network level: (1) the introduction of the PSBM, based on DropBlock; (2) the derivation of upper and lower approximations through Bayesian inference; (3) the redesign of the loss function using Rough Set theory. Consequently, we conducted four sets of ablation experiments using the U-net [36] on the NEU-seg dataset: s1 employing the original U-net as the baseline control group; s2 using a U-net modified with PSBM as the second control group; s3 adding Bayesian inference to the modified U-net (second control group) as the third group; s4 our method as the fourth group, which enhances U-net with PSBM, Bayesian inference, and a Rough Set-based loss function.

The ablation study results are presented in Table 3 and Fig. 5. While PSBM improves U-net's IoU values, it only minimally affects precision rates, and even reduces recall. This suggests that although Dropout can enhance the network's data fitting capability, it is less effective for noisy labels in surface defect detection. Bayesian inference significantly improves both recall and precision rates by providing a posterior distribution. It effectively demarcates false detection regions with lower probabilities and identifies missing detection regions, demonstrating its ability to capture label uncertainty effectively. Further, the Rough Set-based loss function enhances the recall and precision rates. It also delineates anomalies more distinctly and yields more probabilistically accurate results.

4.5. Comparative experiment

In this study, we focus on anomaly segmentation for surface defect detection. Classification model learning methods from noisy labels, as in [9–14,28–31], are not directly comparable to our approach. Pixel-level segmentation methods that rely on additional label information prove challenging to implement in defect detection datasets. Consequently, our method is compared with ADL [15], Pick-and-Learn [16], Label Smoothing [34] and Noise-Robust Dice Loss [32]. Additionally, considering our method's foundation in Dropout, we include the Dropout noise model [28] for comparison. To maintain consistency across comparisons, all methods are adapted from the U-net architecture.

As shown in Table 4, our quantitative analysis reveals that existing methods demonstrate limited improvement in recall rate, precision rate, and IoU. This finding indicates that evaluating noisy labels at the image level is not sufficiently accurate, and addressing the inconsistency in noisy labels is a significant challenge. Our qualitative analysis

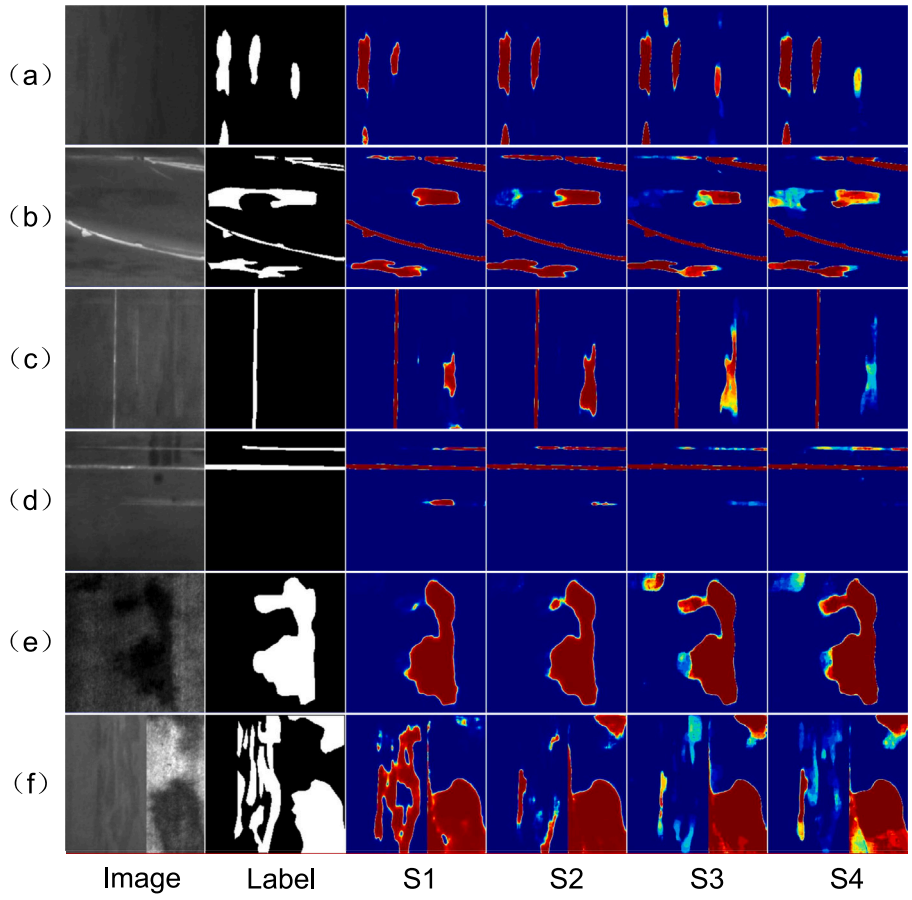


Fig. 5. Results of Ablation experiment. S1, S2, S3, and S4 are the results of original U-net, improved U-net based on PSBM, improved U-net based on PSBM and Bayesian inference, and ours methods (improved by PSBM, Bayesian inference and Rough set based loss function).

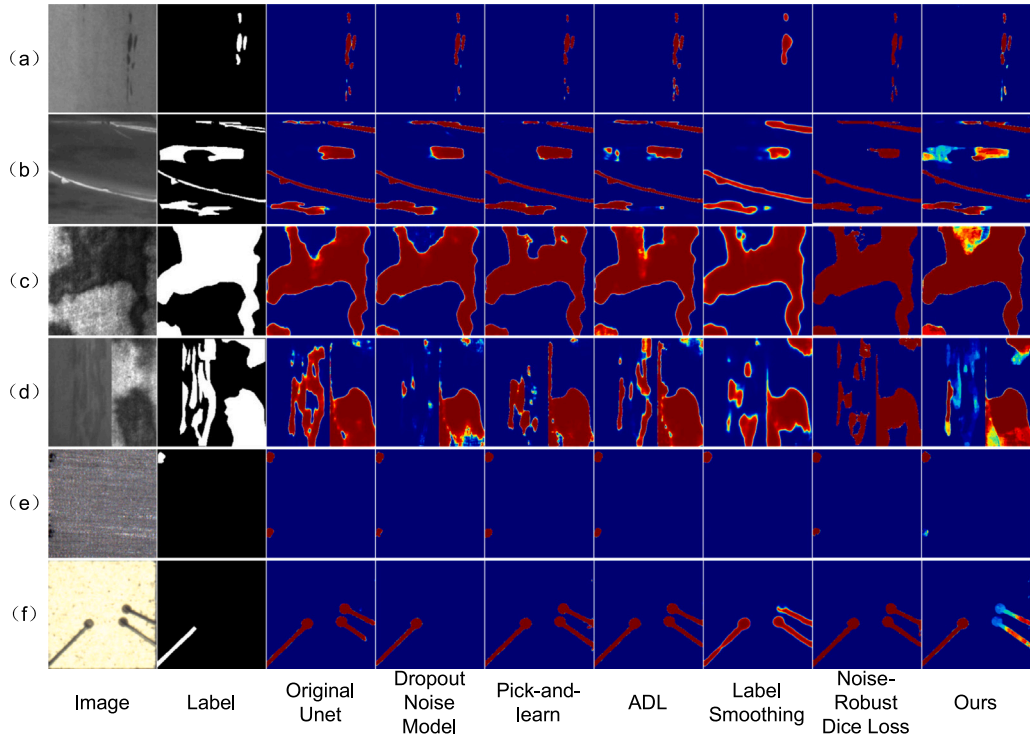


Fig. 6. Results of comparative experiment. Comparative methods are original Unet [36], Dropout noise model [28], Pick-and-learn [16], ADL [15], Label Smoothing [34], Noise-Robust Dice Loss [32] and ours methods.

Table 4
Results of comparative experiments.

	NEU-seg			MCSD-seg			LC-seg		
	Recall	Precision	IoU	Recall	Precision	IoU	Recall	Precision	IoU
U-net [36]	0.8845	0.8455	0.7581	0.8234	0.8590	0.7245	0.8572	0.8894	0.7688
Dropout noise model [28]	0.8755	0.8535	0.7587	0.8131	0.8760	0.7247	0.8051	0.9148	0.7444
Pick-and-learn [16]	0.8639	0.8633	0.7569	0.8427	0.8464	0.7314	0.8401	0.8643	0.7407
ADL [15]	0.8730	0.8556	0.7583	0.8203	0.8663	0.7214	0.8401	0.8888	0.7505
Label Smoothing [34]	0.8807	0.8318	0.7451	0.8416	0.8208	0.7095	0.8551	0.8041	0.7226
Noise-Robust Dice Loss [32]	0.8775	0.7812	0.7041	0.8375	0.8475	0.7216	0.8062	0.8116	0.6919
Ours	**0.9350**	**0.9390**	**0.7670**	**0.8881**	**0.8978**	**0.7386**	**0.8978**	**0.9401**	**0.7825**

Table 5
Results of comparative experiments.

	NEU-seg			MCSD-seg			LC-seg		
	Recall	Precision	IoU	Recall	Precision	IoU	Recall	Precision	IoU
U-net [36]	0.8845	0.8455	0.7581	0.8234	0.8590	0.7245	0.8572	0.8894	0.7688
U-net Based BNNs	**0.9350**	**0.9390**	**0.7670**	**0.8881**	**0.8978**	**0.7386**	**0.8978**	**0.9401**	**0.7825**
PGA-Net [4]	0.8664	0.8710	0.7675	0.8526	0.8580	0.7381	0.9143	0.8458	0.7762
PGA-Net Based BNNs	**0.8960**	**0.9013**	**0.7677**	**0.9019**	**0.8939**	**0.7514**	**0.9330**	**0.8524**	**0.7929**
FCN [40]	0.8623	0.8567	**0.7514**	0.8582	0.8479	0.7393	0.8484	0.8761	0.7676
FCN Based BNNs	**0.8936**	**0.8973**	0.7473	**0.8744**	**0.9169**	**0.7465**	**0.8587**	**0.9043**	**0.7804**
SegNet [41]	0.8353	0.8684	0.7395	0.7744	0.8306	0.6628	0.8336	0.8807	0.7021
SegNet Based BNNs	**0.9107**	**0.8945**	**0.7532**	**0.8552**	**0.8944**	**0.6858**	**0.8384**	**0.9013**	**0.7491**
DeepLabV3+[42]	0.8791	0.8545	0.7651	0.8253	0.8662	0.7391	0.8579	0.8948	0.7300
DeepLabV3+ Based BNNs	**0.9021**	**0.8971**	**0.7696**	**0.8642**	**0.8767**	**0.7358**	**0.8781**	**0.8955**	**0.7781**
Lsa-Net [43]	0.8752	0.8771	0.7790	0.8427	0.8734	0.7514	0.8075	0.8611	0.7310
Lsa-Net Based BNNs	**0.9320**	**0.9190**	**0.7868**	**0.8895**	**0.8926**	**0.7550**	**0.8673**	**0.9062**	**0.7411**
A-Net [44]	0.8846	0.8408	0.7569	0.8481	0.8271	0.7219	0.8007	0.8678	0.7209
A-Net Based BNNs	**0.8917**	**0.8620**	**0.7627**	**0.8644**	**0.8973**	**0.7302**	**0.8643**	**0.8989**	**0.7315**

(Fig. 6) shows that our method achieves high recall and precision rates, effectively minimizing both false positives and missed detections in abnormal regions. Focusing on exploring and exploiting accurate and consistent elements within noisy labels for model training proves more efficient than attempting to evaluate and rectify the labels themselves.

4.6. Application experiments

To demonstrate the robustness and adaptability of our method, we have transformed classic semantic segmentation models (U-net [36], FCN [40], SegNet [41], and DeepLabV3+ [42]), as well as a classic surface defect detection model (PGA-Net [4], Lsa-Net [43], and A-Net [44]), into Bayesian Neural Networks (BNNs). These transformations were tested on three datasets (NEU-seg, MCSD-seg, and LC-seg) to verify their robustness.

As depicted in Table 5 and Figs. 7–9, the BNN-converted models demonstrate an average improvement of 5.36%, 5.24%, and 1.74% in recall rate, precision rate, and IoU, respectively, compared to their original counterparts. These results highlight the significant enhancements our method brings to both the accuracy and recall rate, as well as the overall segmentation capability. Specifically, we observe two main improvements: (1) Reductions in both false and missing detections. As illustrated in Figs. 7–9, the probabilities are used to represent abnormal regions, leading to substantial increases in the recall rate of the upper approximation and the precision rate of the lower approximation. (2) Demonstrated robustness in the pluggability of our method. We have successfully applied our approach to five classic models across three distinct datasets, showcasing its broad applicability to various customized networks in surface defect detection.

4.7. PSBM application modes experiments

To validate the application modes of PSBM, we conducted experiments focusing on its placement and frequency of use within the U-net architecture. Seven comparative experimental groups were designed as follows: (1) the original U-net; (2) PSBM applied to each layer of the encoder; (3) PSBM applied to each layer of the decoder; (4) PSBM applied to the center of U-net; (5) PSBM applied to the last layer of U-net (Classifier layer); (6) PSBM applied to every layer of U-net, including the center, encoder, and decoder; (7) PSBM applied to the center and the last two layers of the encoder.

Table 6
Results of PSBM application experiments.

	Recall	Precision	IoU
U-net	0.8845	0.8455	0.7581
Encoder	0.9217	0.9248	0.7631
Decoder	0.9355	0.9171	0.7562
Center	0.9037	0.9044	0.7609
Classifier	0.6801	0.6725	0.6375
Center-encoder–decoder	**0.9384**	0.9179	0.7546
Center-encoder	0.9350	**0.9390**	**0.7670**

The results, as presented in Table 6, indicate that applying PSBM to the encoder and the center layer significantly enhances the network's representation ability. The recall rate, precision rate, and IoU achieved their best performance when PSBM was positioned in the Center-encoder.

4.8. Real-time analysis

The integration of automated optical inspection in production lines necessitates high real-time performance, necessitating models that are both lightweight and rapid. To mimic a factory computer scenario, the evaluation of BNN-SDD was carried out on a typical personal computer setup, equipped with an NVIDIA GeForce GTX 1070 GPU (8G memory), ensuring the feasibility of deploying our proposed method in an industrial context. The software versions used in the experiments include: Nvidia driver version 552.22, CUDA version 11.8, Cudnn version 9.1, Python version 3.11.7, and Pytorch version 2.3.0.

We evaluated the processing time by continuously testing 180 images from the NEU-seg dataset and calculating the average time per

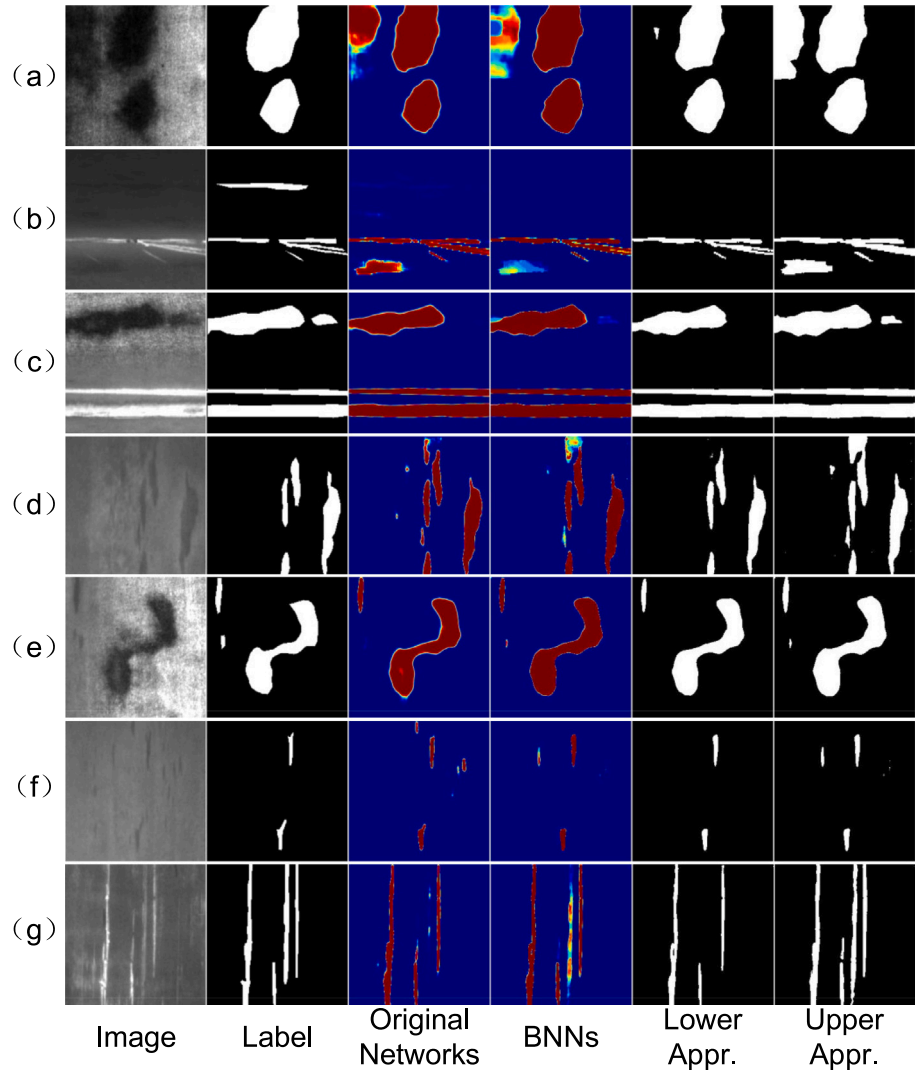


Fig. 7. Results of Application examples for NEU-seg. Rows (a), (b), (c), (d), (e), (f) and (g) are the results of U-net based BNNs, PGA-Net Based BNNs, FCN Based BNNs, SegNet Based BNNs, DeepLabV3+ Based BNNs, Lsa-Net Based BNNs and A-Net Based BNNs. The order of columns are the images, labels, results of original models, segmentation probabilities of BNNs, lower approximations and upper approximations.

Table 7
Result of real-time example.

	Original network		BNNs	
	Parameters (MB)	Times (ms)	Parameters (MB)	Times (ms)
U-net [36]	31.41	4.38	31.40	4.87
PGA-net [4]	205.63	10.81	205.63	9.13
FCN [40]	44.71	4.26	44.71	4.71
SegNet [41]	117.77	5.66	112.32	6.10
DeepLabV3+ [42]	237.85	19.53	237.85	19.32
LSA-Net [43]	86.41	7.48	86.39	7.89
A-Net [44]	1.56	15.43	1.56	16.64

image for comparison. This approach mirrors the real-world operation on a production line, where each image is taken and processed sequentially. Although BNNs typically require 16 computations, we executed these in parallel, achieving remarkable performance.

The results in Table 7 show that our method maintains real-time performance with negligible differences in processing time and parameter size when compared to the original networks. This consistency is crucial for applications requiring real-time defect detection. While Table 7 focuses on real-time performance metrics, our method also improves detection accuracy, as demonstrated in other sections of the paper (please refer to the results in Sections Section 4.6). These improvements do not compromise the efficiency highlighted in Table 7, thereby offering a balanced enhancement in both accuracy and performance.

Table 8
Thresholds given by factory-standard in MCSD-seg.

Dataset	Length (mm)	Width (mm)
Tin color	1.5	1.5
Scratches	2.0	2.0
Indentations	0.80	0.30
Smudge	0.30	0.14

4.9. Application in the production line of motor commutator

Defect detection in motor commutators represents a typical scenario for metal surface defect inspection. The MCSD-seg dataset used in this study includes four types of defects: tin color, scratches, indentations, and dirt. Industry-standard thresholds for defect determination are detailed in Table 8. The images are sized at 256 × 256 pixels, with a pixel equivalent of 0.014 mm/pix.

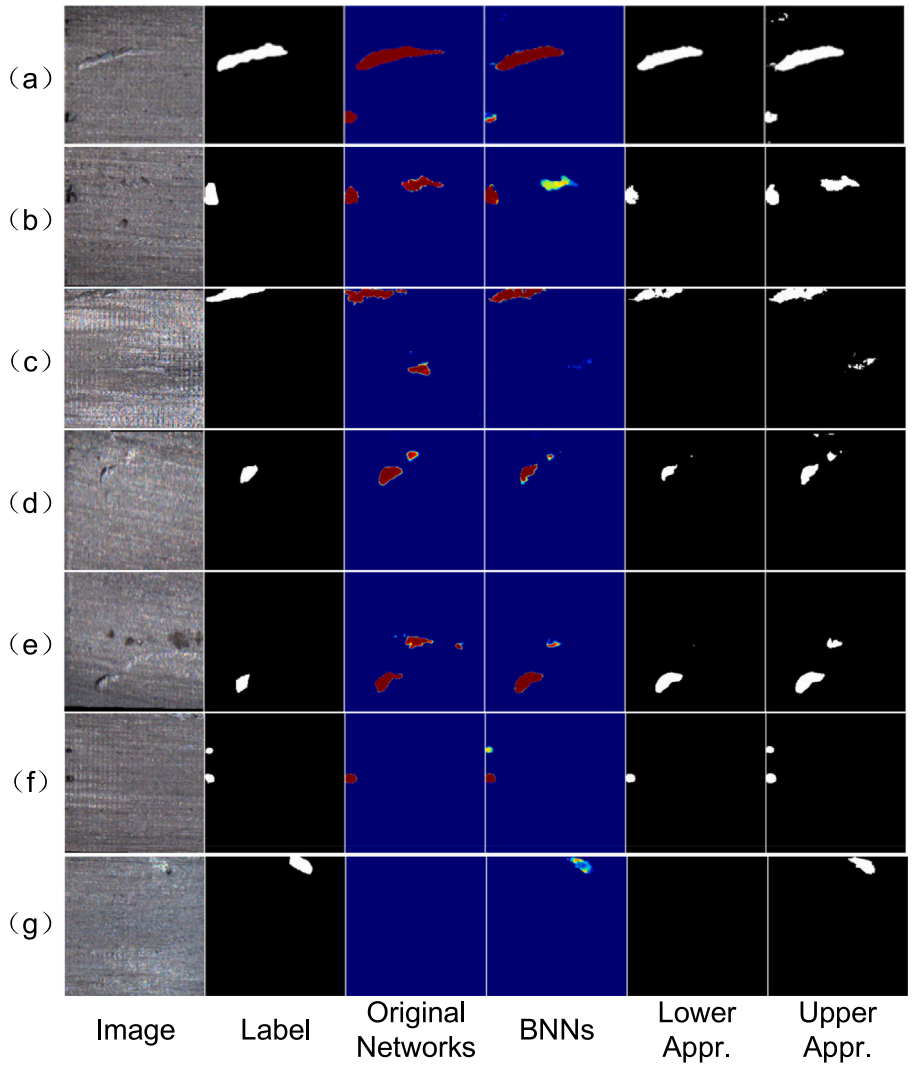


Fig. 8. Results of Application examples for MCSD-seg. Rows (a), (b), (c), (d), (e), (f) and (g) are the results of U-net based BNNs, PGA-Net Based BNNs, FCN Based BNNs, SegNet Based BNNs, DeepLabV3+ Based BNNs, Lsa-Net Based BNNs and A-Net Based BNNs. The order of columns are the images, labels, results of original models, segmentation probabilities of BNNs, lower approximations and upper approximations.

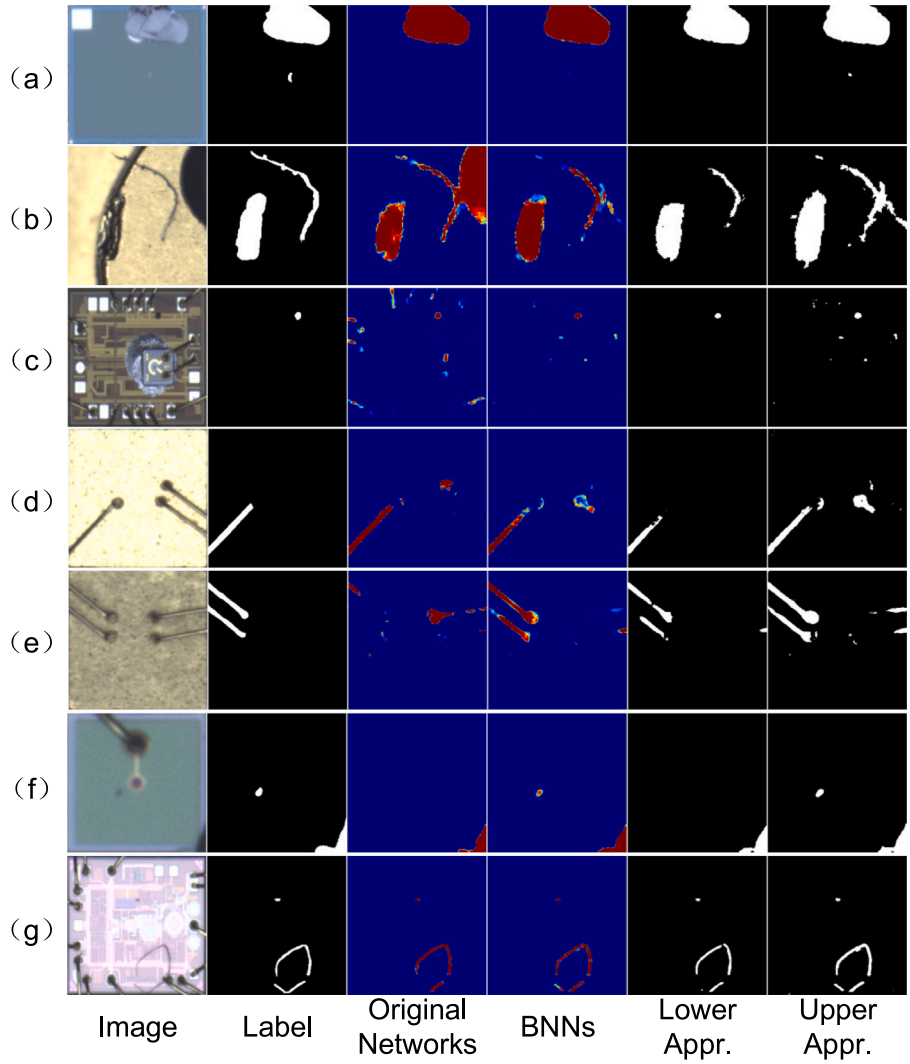


Fig. 9. Results of Application examples for LC-seg. Rows (a), (b), (c), (d), (e), (f) and (g) are the results of U-net based BNNs, PGA-Net Based BNNs, FCN Based BNNs, SegNet Based BNNs, DeepLabV3+ Based BNNs, Lsa-Net Based BNNs and A-Net Based BNNs. The order of columns are the images, labels, results of original models, segmentation probabilities of BNNs, lower approximations and upper approximations.

Initially, using U-Net-based BNNs, we carried out model optimization and inference to determine the segmentation probability for MCSD-seg, depicted in row (b) of Fig. 10. Subsequently, applying formulas (24) and (25), we calculated the confidence levels for each connected region within these probabilities, as illustrated in row (c) of Fig. 10. Red contour lines mark the anomaly regions of confidence that meet the threshold criteria and are thus classified as defective. This confidence score represents the likelihood of a connected region being a defect, proving to be a valuable metric for both trustworthy defect judgment and product classification.

5. Conclusion

In this paper, instead of correcting or filtering noisy labels, we identify inaccurate information in noisy labels (suspicious regions in images) as uncertain areas using Rough Set theory to provide precise lower and upper approximations. Additionally, we redesign the loss function and introduce the PSBM. Our method significantly improves defect detection without requiring extra labeling or architectural changes, outperforming state-of-the-art techniques across multiple datasets and enhancing various classic networks as a pluggable module.

Firstly, while no extra label information is needed, the accuracy of upper and lower approximations in representing suspicious regions depends on the quality of the noisy labels, which must include both over-labeling and under-labeling. Furthermore, the robustness to more complex noise distributions, such as varying noise levels, remains unexplored. Secondly, the redesigned Tversky-based loss function faces challenges, including the complexity of parameter selection and the difficulty in finding the sensitivity balance point across different datasets. Lastly, the Bayesian neural network employs the Monte Carlo integration method during training, which increases both computational cost and time expenditure. Finally, our method is only applied to limited scenarios, such as metal surfaces and optical chip surfaces, where images have clear suspicious regions, like weak features or border areas. The applicability of this approach to fields such as textiles and medicine remains to be explored.

In future research, we plan to explore the applicability of our method across varying levels of noise. First, our initial focus will be on semi-supervised learning techniques, which integrate a small set of labeled data with a larger set of unlabeled data during training. This approach aims to decrease the model's reliance on the quality of noisy labels. Second, we will investigate active learning methods, which aim to improve the quality of labels and models by selectively relabeling certain information. Furthermore, we will study the causes and manifestations of noise labels in more surface defect detection scenarios so that the method can be applied to more cases.

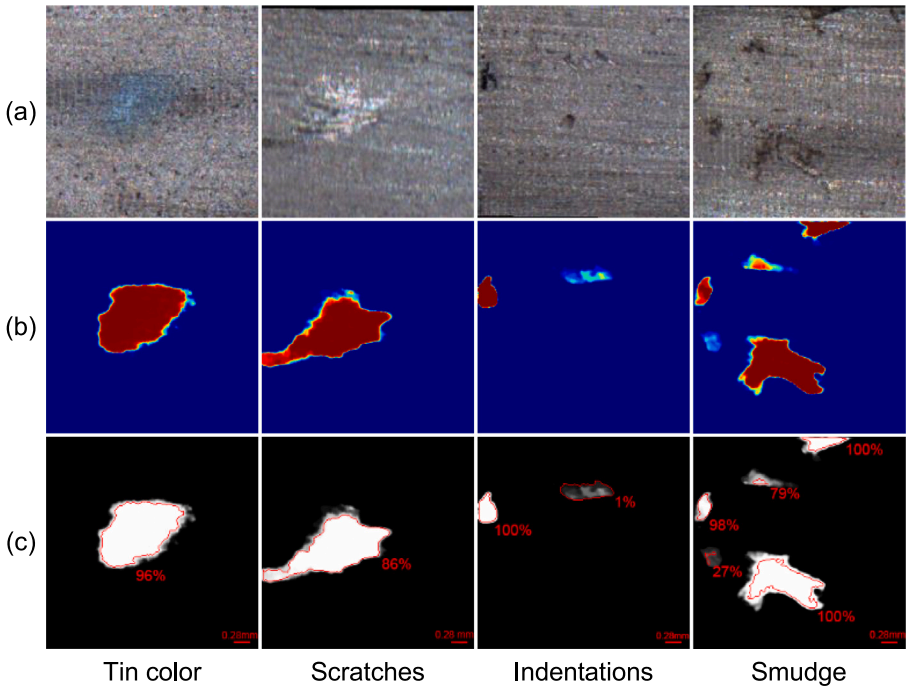


Fig. 10. Results of Application in the production line of motor commutator. Row (a), (b), and (c) are the images, segmentation probabilities, and confidences.

6. Practical application and future usefulness

Our method can be directly integrated into existing defect detection systems used in various industries without requiring significant modifications to the underlying architecture. Practitioners can utilize the proposed method as follows: (1) Handling noisy labels: by identifying and managing noisy labels using Rough Set theory, practitioners can improve the accuracy of defect detection without needing to invest in extensive re-labeling efforts. (2) Integration with existing systems: the proposed PSBM can be added as a module to enhance the defect detection capabilities of current systems. This modularity ensures ease of adoption.

Potential uses of our methods for future studies: (1) Innovative approaches for noisy labels: Different from any existing method, we analyzed the causes of noisy labels in the surface defect detection scenario and proposed a suspected area characterization method based on rough sets. This provides a new approach to dealing with noisy labels. (2) Providing ideas for other methods: For example, the size of the uncertain area can reflect the noise level of the image label and provide a judgment indicator for active learning to screen data and re-label. (3) Comprehensive evaluations: our method can be used as a benchmark to evaluate new noisy label defect detection approaches, fostering the development of more accurate and efficient solutions.

By providing a practical, modular, and adaptable solution, our method serves as a valuable tool for both practitioners dealing with real-life problems and researchers aiming to advance the field of defect detection.

CRediT authorship contribution statement

Tongzhi Niu: Writing – review & editing, Writing – original draft, Validation, Methodology, Data curation, Conceptualization. **Zhenrong Wang:** Writing – review & editing, Visualization. **Weifeng Li:** Writing – review & editing, Validation. **Kai Li:** Writing – review & editing, Formal analysis. **Yuwei Li:** Writing – original draft, Visualization. **Guiyin Xu:** Writing – review & editing, Software. **Bin Li:** Writing – review & editing, Project administration, Conceptualization.

Declaration of competing interest

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

Data availability

Data will be made available on request.

References

[1] X. Hu, J. Yang, F. Jiang, A. Hussain, K. Dashtipour, M. Gogate, Steel surface defect detection based on self-supervised contrastive representation learning with matching metric, Appl. Soft Comput. 145 (2023) 110578, http://dx.doi.org/10.1016/j.asoc.2023.110578.

[2] S.S. Hameed, V. Muralidharan, B.K. Ane, Comparative analysis of fuzzy classifier and ANN with histogram features for defect detection and classification in planetary gearbox, Appl. Soft Comput. 106 (2021) 107306, http://dx.doi.org/10.1016/j.asoc.2021.107306.

[3] W. Li, B. Li, S. Niu, Z. Wang, B. Liu, T. Niu, Selecting informative data for defect segmentation from imbalanced datasets via active learning, Adv. Eng. Inform. 56 (2023) 101933, http://dx.doi.org/10.1016/j.aei.2023.101933.

[4] H. Dong, K. Song, Y. He, J. Xu, Y. Yan, Q. Meng, PGA-Net: Pyramid feature fusion and global context attention network for automated surface defect detection, IEEE Trans. Ind. Inform. 16 (12) (2020) 7448–7458, http://dx.doi.org/10.1109/TII.2019.2958826.

[5] A.S. Rich, T.M. Gureckis, Lessons for artificial intelligence from the study of natural stupidity, Nat. Mach. Intell. 1 (4) (2019) 174–180, http://dx.doi.org/10.1038/s42256-019-0038-z.

[6] Z. Liao, Y. Xie, S. Hu, Y. Xia, Learning from ambiguous labels for lung nodule malignancy prediction, IEEE Trans. Med. Imaging (2022) http://dx.doi.org/10.1109/TMI.2022.3149344.

[7] Z. Xu, D. Lu, J. Luo, Y. Wang, J. Yan, K. Ma, Y. Zheng, R.K.-Y. Tong, Anti-interference from noisy labels: Mean-teacher-assisted confident learning for medical image segmentation, IEEE Trans. Med. Imaging 41 (11) (2022) 3062–3073, http://dx.doi.org/10.1109/TMI.2022.3176915.

[8] T. Niu, B. Chen, Q. Lyu, B. Li, W. Luo, Z. Wang, B. Li, Scoring bayesian neural networks for learning from inconsistent labels in surface defect segmentation, Measurement 225 (2024) 113998, http://dx.doi.org/10.1016/j.measurement.2023.113998.

[9] S. Jenni, P. Favaro, Deep bilevel learning, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 618–633.

[10] M. Lukasik, S. Bhojanapalli, A. Menon, S. Kumar, Does label smoothing mitigate label noise? in: International Conference on Machine Learning, PMLR, 2020, pp. 6448–6458.
[11] Z. Zhang, M. Sabuncu, Generalized cross entropy loss for training deep neural networks with noisy labels, Adv. Neural Inf. Process. Syst. 31 (2018).
[12] R. Wang, T. Liu, D. Tao, Multiclass learning with partially corrupted labels, IEEE Trans. Neural Netw. Learn. Syst. 29 (6) (2017) 2568–2580, http://dx.doi.org/10.1109/TNNLS.2017.2699783.
[13] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. Tsang, M. Sugiyama, Co-teaching: Robust training of deep neural networks with extremely noisy labels, in: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 31, Curran Associates, Inc., 2018.
[14] J. Huang, L. Qu, R. Jia, B. Zhao, O2U-net: A simple noisy label detection approach for deep neural networks, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, ICCV, 2019.
[15] Z. Wang, Z. Zhang, I. Voiculescu, RAR-u-NET: A residual encoder to attention decoder by residual connections framework for spine segmentation under noisy labels, in: 2021 IEEE International Conference on Image Processing, ICIP, 2021, pp. 21–25.
[16] H. Zhu, J. Shi, J. Wu, Pick-and-learn: automatic quality evaluation for noisy-labeled image segmentation, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer, 2019, pp. 576–584.
[17] R. Yi, Y. Huang, Q. Guan, M. Pu, R. Zhang, Learning from pixel-level label noise: A new perspective for semi-supervised semantic segmentation, IEEE Trans. Image Process. 31 (2022) 623–635, http://dx.doi.org/10.1109/TIP.2021.3134142.
[18] Z. Lu, Z. Fu, T. Xiang, P. Han, L. Wang, X. Gao, Learning from weak and noisy labels for semantic segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 39 (3) (2017) 486–500, http://dx.doi.org/10.1109/TPAMI.2016.2552172.
[19] Z. Pawlak, Rough sets, Int. J. Comput. Inf. Sci. 11 (5) (1982) 341–356.
[20] S.S.M. Salehi, D. Erdogmus, A. Gholipour, Tversky loss function for image segmentation using 3D fully convolutional deep networks, in: International Workshop on Machine Learning in Medical Imaging, Springer, 2017, pp. 379–387.
[21] Y. Gal, Z. Ghahramani, Bayesian convolutional neural networks with Bernoulli approximate variational inference, 2015, arXiv preprint arXiv:1506.02158.
[22] A. Kendall, V. Badrinarayanan, R. Cipolla, Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding, 2015, arXiv preprint arXiv:1511.02680.
[23] Y. Gal, Z. Ghahramani, Dropout as a bayesian approximation: Representing model uncertainty in deep learning, in: International Conference on Machine Learning, PMLR, 2016, pp. 1050–1059.
[24] Y. Hiasa, Y. Otake, M. Takao, T. Ogawa, N. Sugano, Y. Sato, Automated muscle segmentation from clinical CT using Bayesian U-net for personalized musculoskeletal modeling, IEEE Trans. Med. Imaging 39 (4) (2020) 1030–1040, http://dx.doi.org/10.1109/TMI.2019.2940555.
[25] G. Ghiasi, T.-Y. Lin, Q.V. Le, Dropblock: A regularization method for convolutional networks, Adv. Neural Inf. Process. Syst. 31 (2018).
[26] R.M. Neal, Bayesian Learning for Neural Networks, Vol. 118, Springer Science & Business Media, 2012.
[27] H. Song, M. Kim, D. Park, Y. Shin, J.-G. Lee, Learning from noisy labels with deep neural networks: A survey, IEEE Trans. Neural Netw. Learn. Syst. (2022) 1–19, http://dx.doi.org/10.1109/TNNLS.2022.3152527.
[28] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[29] T. Xiao, T. Xia, Y. Yang, C. Huang, X. Wang, Learning from massive noisy labeled data for image classification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2015.
[30] J. Yao, J. Wang, I.W. Tsang, Y. Zhang, J. Sun, C. Zhang, R. Zhang, Deep learning from noisy image labels with quality embedding, IEEE Trans. Image Process. 28 (4) (2019) 1909–1922, http://dx.doi.org/10.1109/TIP.2018.2877939.
[31] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: International Conference on Machine Learning, PMLR, 2015, pp. 448–456.
[32] G. Wang, X. Liu, C. Li, Z. Xu, J. Ruan, H. Zhu, T. Meng, K. Li, N. Huang, S. Zhang, A noise-robust framework for automatic segmentation of COVID-19 pneumonia lesions from CT images, IEEE Trans. Med. Imaging 39 (8) (2020) 2653–2663, http://dx.doi.org/10.1109/TMI.2020.3000314.
[33] X. Zhou, X. Liu, D. Zhai, J. Jiang, X. Ji, Asymmetric loss functions for noise-tolerant learning: Theory and applications, IEEE Trans. Pattern Anal. Mach. Intell. (2023) http://dx.doi.org/10.1109/TPAMI.2023.3236459.
[34] P. Pornvoraphat, K. Tiankanon, R. Pittayanon, P. Sunthornwetchapong, P. Vateekul, R. Rerknimitr, Real-time gastric intestinal metaplasia diagnosis tailored for bias and noisy-labeled data with multiple endoscopic imaging, Comput. Biol. Med. 154 (2023) 106582, http://dx.doi.org/10.1016/j.compbiomed.2023.106582.
[35] D. Karimi, C.K. Rollins, C. Velasco-Annis, A. Ouaalam, A. Gholipour, Learning to segment fetal brain tissue from noisy annotations, Med. Image Anal. 85 (2023) 102731, http://dx.doi.org/10.1016/j.media.2022.102731.
[36] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer, 2015, pp. 234–241.
[37] K. Song, Y. Yan, A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects, Appl. Surf. Sci. 285 (2013) 858–864, http://dx.doi.org/10.1016/j.apsusc.2013.09.002.
[38] T. Niu, B. Li, W. Li, Y. Qiu, S. Niu, Positive-sample-based surface defect detection using memory-augmented adversarial autoencoders, IEEE/ASME Trans. Mechatronics 27 (1) (2022) 46–57, http://dx.doi.org/10.1109/TMECH.2021.3058147.
[39] T. Niu, Z. Xie, J. Zhang, L. Tang, B. Li, H. Wang, A generalized well neural network for surface defect segmentation in optical communication devices via template-testing comparison, Comput. Ind. 151 (2023) 103978, http://dx.doi.org/10.1016/j.compind.2023.103978.
[40] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3431–3440.
[41] V. Badrinarayanan, A. Kendall, R. Cipolla, Segnet: A deep convolutional encoder-decoder architecture for image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 39 (12) (2017) 2481–2495, http://dx.doi.org/10.1109/TPAMI.2016.2644615.
[42] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A.L. Yuille, Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs, IEEE Trans. Pattern Anal. Mach. Intell. 40 (4) (2017) 834–848, http://dx.doi.org/10.1109/TPAMI.2017.2699184.
[43] W. Li, B. Li, S. Niu, Z. Wang, M. Wang, T. Niu, LSA-net: Location and shape attention network for automatic surface defect segmentation, J. Manuf. Process. 99 (2023) 65–77, http://dx.doi.org/10.1016/j.jmapro.2023.05.001.
[44] B. Chen, T. Niu, W. Yu, R. Zhang, Z. Wang, B. Li, A-net: An A-shape lightweight neural network for real-time surface defect segmentation, IEEE Trans. Instrum. Meas. (2023) http://dx.doi.org/10.1109/TIM.2023.3341115.