Scoring Bayesian Neural Networks for Learning from Inconsistent Labels in Surface Defect Segmentation

Abstract

This paper focus on surface defect segmentation uncertainty challenges that arise due to human errors and biases in the data annotation process, particularly in ambiguous transition and weak feature areas. Firstly, uncertain areas are defined, where it could not be unambiguously identified as defect or defect-free. Then a scoring Bayesian neural network is proposed, using Bayesian neural computation to solve the segmentation probability and provide an expression for uncertain areas. The variance of segmentation probability is utilized to assess the quality of labels, thereby improving model performance. The approach is validated against prevailing state-of-the-art methods on five datasets, demonstrating superior performance. This study provides a crucial pathway for addressing human errors and biases in defect detection.

Keywords:
Surface defect segmentation, Bayesian neural networks, Semantic segmentation probability, Inconsistent label

1. Introduction

Surface defect detection is a crucial component of the manufacturing process, with widespread applications across numerous industries, including workpiece [1], flat steel [2], additive manufacturing [3], and 3C glass [4], among others. In recent years, Convolutional Neural Networks (CNNs) have demonstrated remarkable advancements in the field of surface defect detection, significantly improving performance standards and establishing new benchmarks.

While deep learning models hold considerable potential, these data-driven approaches have been found to reproduce, and in certain instances, even amplify human errors and biases that are intrinsic to the training dataset,

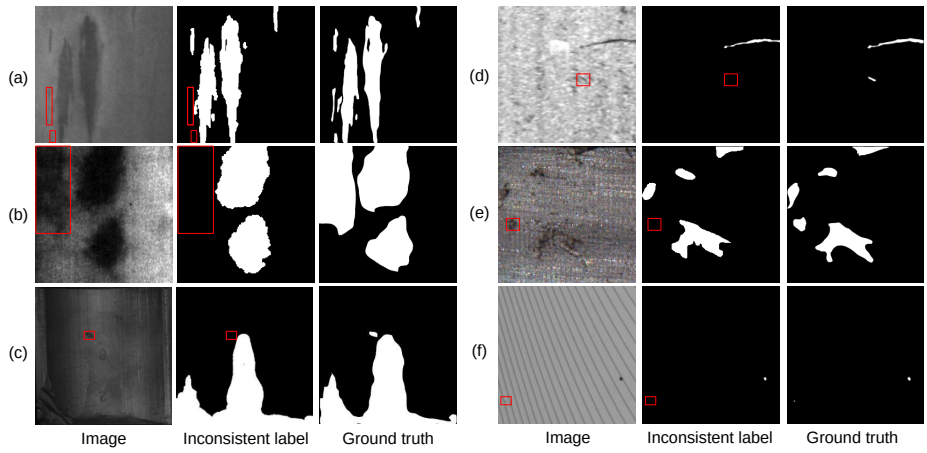

Figure 1: The inconsistent labels

particularly those introduced during the labeling process [5, 6]. This issue is pervasive, affecting not only natural and medical imagery [7, 8], but is notably severe in the sphere of surface defect detection.

The task of annotation of surface defect segmentation confronts two primary challenges: 1) transition areas: in defect detection images, considering the causes of defects like scratches or pollution, there is inevitably an extensive transition area between the normal background and the anomalous foreground; 2) weak features areas: the issue of inter-class similarity is prevalent in defect detection, making it challenging to distinguish the background from the foreground. Moreover, given the irregular contours of defects and the possibility of some being minute enough to evade detection during the labeling process, these weak features areas are particularly prone to being overlooked in the labeling stage.

To compound this, the annotation of transition and weak features areas are subject to a host of variables such as the fluctuating emotional states of workers, differing judgement standards, and varied technical skill levels, thereby causing inconsistent labeling outcomes. As illustrated in Fig. 1, the boundaries of certain areas are marked within a limited range, others within an extensive range, while some areas are disregarded altogether.

A variety of strategies have been proposed to learn from inconsistent labels, incorporating methods based on Bayesian probabilities [9], proxy labels [10], and 'divide-and-rule' approaches [7]. However, these methods predominantly depend on multi-labels, which can prove to be costly for defect detec-

tion. Furthermore, our focus should shift towards labels at the pixel-level, as opposed to those at the image level.

In the realm of pixel-level labels, both SEAL [11] and STEAL [12] enhance the delineation of semantic regions by explicitly accounting for annotation inconsistencies during training, thereby leveraging prior knowledge. SEAL reimagines the optimization of latent edge labels as a problem of minimum-cost assignment in a bipartite graph, subsequently adjusting labels during training guided by a biased Gaussian kernel and a Markov prior. On the other hand, STEAL introduces a novel layer and loss function that mandate the edge detector to predict a peak response along the normal direction of the edge, while concurrently regularizing its direction. However, all these approaches operate under the presumption of the existence of precise edges. In industrial inspections, owing to the origins of defects, transitional and weak features areas are widespread. Therefore, the assumption of precise edges often proves fallacious. Even some areas with weak features lack annotations. Thus, simply focusing on label edges is not adequate.

The challenges posed by manual labeling are virtually unavoidable due to the inevitable inconsistencies arising from the annotation process. However, if these challenges are appropriately addressed, they can often aid in resolving practical issues. In this paper, we argue that areas characterized by transition and weak features cannot be unambiguously identified as defect or defect-free, which means that these areas are still uncertain. These uncertain areas embody all pixels that cannot be definitively classified using existing features.

Nonetheless, traditional deep learning models designed for segmentation fail to account for this inherent uncertainty [13]. While Bayesian probability theory provides a mathematical tool to address uncertainty areas, it typically imposes prohibitive computational costs. Consequently, without modifying the model or its optimization, we leverage Monte-Carlo dropout [14] as an approximation of Bayesian inference over the network's weights. This approach allows us to approximate the posterior distribution by drawing samples from the Bernoulli distribution. We weigh and aggregate the results of multiple samplings to generate a segmentation probability map. In this map, areas with a probability of 0 are designated as normal, areas with a probability of 1 as abnormal, and areas with a probability within $(0, 1)$ as uncertain areas.

The chief aim of this paper is to achieve defect detection without the requirement for supplementary data and additional labels. Considering the high costs related to image acquisition and annotation, labels of mediocre consistency and even those tainted by noise should be fully leveraged. Lower

label quality equates to higher uncertainty in Bayesian inference, which manifests as heightened variance. Therefore, we propose Scoring Networks (SN) to assess label quality, using the variance from multiple solutions of the Bayesian Neural Networks (BNN) as the labels for the SN.

In summary, the principal contributions of this paper can be delineated as follows:

1) We introduce an innovative uncertainty-centric framework that generates segmentation probabilities, thereby providing a representation of uncertainty areas driven by inconsistent labeling.

2) To fully exploit limited data and labels, We developed the Score-BNN, which utilize the variance from multiple Bayesian Neural Network solutions as an indicator of label quality, thereby enhancing model performance.

3) Our method was rigorously tested on five publicly accessible datasets, exhibiting comparable or superior performance against established state-of-the-art models, thereby confirming its effectiveness and efficiency.

2. Related Works

2.1. Bayesian Neural Networks

In the realm of deep learning, Bayesian Neural Networks (BNNs) [15] proffer a probabilistic interpretation of deep learning models. They achieve this by making inferences on distributions over the weights of the models, gravitating towards Gaussian processes. However, the copious quantity of parameters in these networks presents a significant hurdle when attempting to model a distribution over the kernels, inevitably leading to additional computational burdens.

Interestingly, Yarin [13] established that the use of dropout during the training phase of neural networks can be seen as a Bayesian approximation. This approach circumvents any increase in computational complexity or deterioration of test accuracy. Concurrently, Yarin proposed a practical architecture for dropout CNNs [14]. In this context, the training of such networks can be reinterpreted as an approximate Bernoulli variational inference in BNNs. Furthermore, the evaluation of these models can be performed by approximating the predictive posterior - a strategy termed Monte Carlo dropout during the testing phase.

Bolstering this probabilistic approach to deep learning, Alex Kendall and his team introduced a framework named Bayesian SegNet [16]. This frame-

work facilitates probabilistic pixel-wise semantic segmentation, enhancing the capabilities of traditional deep learning models.

Inspired by these innovative methods, we propose a surface defect detection network based on BNNs for learning a robust model from inconsistent labels.

2.2. Background of surface defect detection

Surface defect detection is integral to upholding product quality and process control across an array of manufacturing sectors, including but not limited to automotive parts production and semiconductor fabrication. As such, methods anchored in deep learning have garnered considerable attention and instigated a breadth of research [1, 2, 3, 4].

However, these deep learning-based approaches are not impervious to challenges. The training of such models generally necessitates large, labeled datasets, acquisition of which can prove arduous and costly in the sphere of surface defect detection. Consequently, considerable research has been dedicated to data augmentation [17], anomaly detection [18], and domain generalization [19] methods in recent times.

Data augmentation techniques, such as those drawing on Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and Diffusion Models, aim to manufacture a wider and richer data distribution from a confined dataset. By amplifying data diversity, these techniques augment the overall model's capability, granting it a more holistic comprehension of potential data representations. Anomaly detection methods typically take advantage of the relatively effortless collection of positive samples (non-defective). They construct a distribution model of positive sample features utilizing reconstructive or feature embedding models, thereby enabling effective defect detection. By capturing the 'normal' representation of the data, anomalies or defects become conspicuous, facilitating their identification. Domain adaptation methods combat the challenge of adapting to different types, batches, and data distributions without necessitating exhaustive data collection or labeling for each novel task. These techniques permit models, trained on a specific domain or dataset, to be applied to a different albeit related domain, hence reducing the reliance on manual data collection and labeling for each new task.

Moreover, deep learning models frequently mirror or even amplify human errors and biases instilled during the data labeling process, thereby

complicating the detection of subtle defects. This paper introduces a pioneering framework that confronts the aforementioned challenges, leveraging uncertainty quantification to bolster the reliability and robustness of surface defect detection.

3. Proposed approach

3.1. Overview

The architecture of the proposed Scoring Bayesian Neural Networks for Surface Defect Detection (Score-BNN) is characterized by two distinct phases: training and testing, as illustrated in Fig. 2.

In the training phase, we feed the input to the Bayesian Neural Network (BNN) N times to generate N unique segmentation results (Seg). These results are subsequently compared with the label to derive the segmentation errors (Seg error). Concurrently, we employ the Scoring Networks (SN) to evaluate the label's score (Score), which requires the concatenation of the input and label as its input. The scoring loss ($Loss_{Score}$) is calculated as the variance of the Seg error (Var) minus the Score, and is utilized to back-propagate the SN. The segmentation loss ($Loss_{seg}$) is computed as the product of the Score and the mean of the Seg error (Mean), and serves to back-propagate both the SN and BNN.

In the testing phase, the BNN is run N times with the same input, yielding multiple distinct segmentation results (Seg). The SN is then utilized to compute the Seg score from the concatenation of Seg and the input. Each Seg is subsequently weighted by its corresponding Seg score, and the results are aggregated to yield the final segmentation probability.

3.2. Bayesian Neural Networks for Surface Defects Detection

3.2.1. Probabilistic modelling

Given the training inputs $\{x_1, ..., x_N\}$ and corresponding labels $\{y_1, ..., y_N\}$, our goal is to estimate a function $y = f(x)$. In the Bayesian approach, we impose a prior distribution over the function space, denoted as $p(f)$. Subsequently, we seek the posterior distribution over the same space conditioned on our dataset, i.e., $p(f \mid X, Y)$.

In the context of Bayesian neural networks, our primary interest lies in discovering the posterior distribution over the convolutional weights.

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

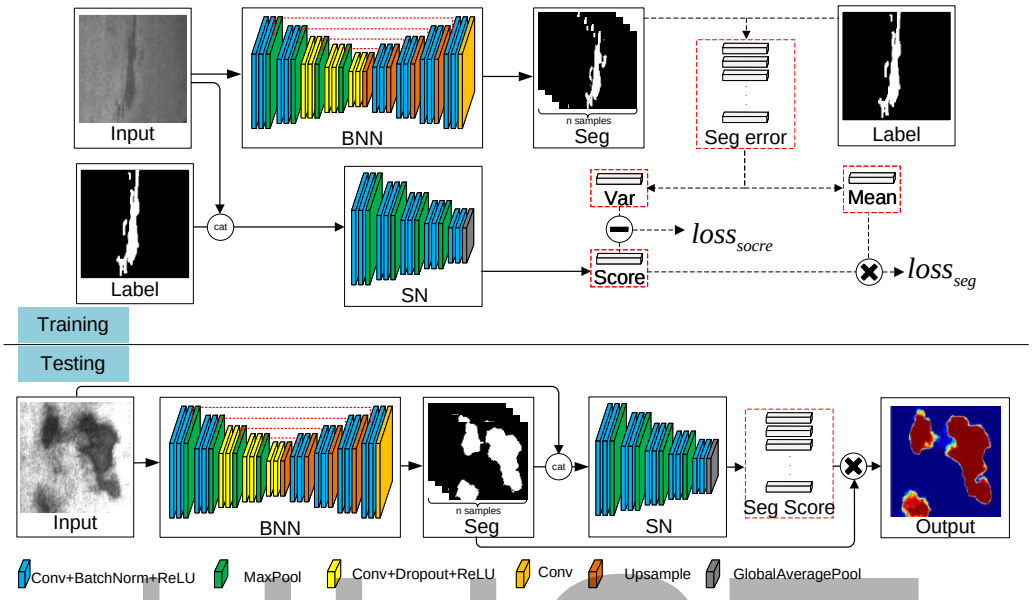

Figure 2: The architecture of our proposed Scoring Bayesian Neural Networks for Surface Defects Segmentation.

where W_i represents the weights of the i^{th} layer in the convolutional network. However, the distribution $p(w \mid X, Y)$ is not analytically tractable. Consequently, we define an approximation variational distribution $q(w)$ to approximate $p(w)$. Inspired by [14], we employ Gaussian prior distributions to approximate $q(w)$. Subsequently, the Gaussian process can be approximated using Bernoulli-distributed random variables with dropout probabilities $b_{i,j}$ and variational parameters of the CNN's kernels, K_i. Here, $b_{i,j}$ is the dropout probability of the j^{th} neuron in the i^{th} layer of the network, while K_i is the convolutional kernel of the i^{th} layer of the network. Consequently, $q(W_i)$ is defined for every layer i as follows:

$$\begin{aligned} W_i &= K_i \cdot diag([b_{i,j}]_{j=1}^{J}) \\ b_{i,j} &\sim Bernoulli(p_i) for\ i = 1, ..., I, j = 1, ...J \end{aligned} \tag{2}$$

The operator $diag(\cdot)$ transforms vectors into diagonal matrices whose elements originate from the vector. p_i signifies a fixed Bernoulli distribution probability, generally set at $p_i = 0.5$. The network comprises I layers in total, each containing J neurons.

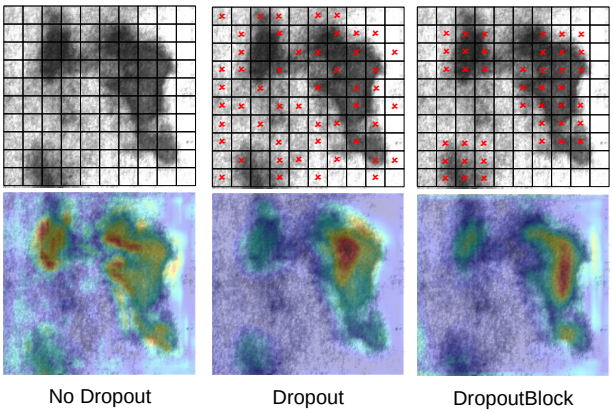

Figure 3: Activation map of varying dropout results in the NEU-Seg dataset.

In networks inference, we approximate the integral with Monte Carlo integrations:

$$p(y^*|x^*, X, Y) \approx \frac{1}{T}\sum_{t=1}^{T} \hat{f}(x^*, \hat{w}_t) \tag{3}$$

where x^* and y^* are the input and output in test set, and $\hat{w}_t \sim q(w)$. T is a hyperparameter used to balance the accuracy of calculation results and computational overhead.

3.3. Spatially correlated Bayesian CNN block

Dropout is implemented following convolution layers in numerous approaches to create Bayesian neural networks [13, 20, 16]. However, due to the spatial correlation of convolutional layers, the efficacy of dropout tends to diminish. Motivated by DropBlock [21], we propose a spatially correlated Bayesian CNN block.

Initially, we replace batch normalization with dropout in the CNN block. A typical CNN block comprises a convolution, batch normalization, and an activation function. As per equation 2, dropout must be applied after the convolution layers to facilitate a Bayesian CNN block. However, combining the two powerful techniques of Dropout and Batch Normalization often results in a performance decline [22]. Hence, we substitute batch normalization with dropout, forming the fundamental structure of the Bayesian CNN block, which includes a convolution, dropout, and an activation function, as demonstrated in Fig. 3.

Subsequently, we explore the impact of various dropout structures on CNN. Standard dropout selectively disables features. When these features are correlated, the input information cannot be completely discarded even with dropout, which leads to a prior loss of variance. In this paper, we introduce kernel-likely dropout, which disables pixels in entire blocks, as depicted in Fig. 3. We employ an activation map to visualize the convolutional layer activations prior to the first up-sampling of the BNN. Generally, models trained with a spatially correlated Bayesian CNN block learn spatially distributed representations, which induce high activations in accurate regions. In contrast, models devoid of regularization tend to focus on larger and less accurate regions.

3.4. Networks design

The overall structure of our network, as demonstrated in Fig. 2, is designed based on Unet [23]. This model performs optimally on the majority of defect detection datasets, as evidenced in Table 3 4 5 6 7. This performance can potentially be attributed to the relatively fixed structure of industrial images and the simplicity of their semantics, wherein low-level features hold comparable importance to high-level ones.

In practice, excessive regularization may inhibit the speed of network learning. With partial Bayesian CNN blocks, however, we can realize a Bayesian CNN. Thus, we explore the optimal quantity and placement for the Bayesian CNN block. As shown in Table 2, Intersection over Union (IoU) and Pixel Accuracy (PA) are maximized when the Bayesian CNN block is positioned in the last two layers of the encoder and in the central layer. These findings indicate that applying the Bayesian CNN block to the lower layers does not yield superior performance. Low-level features such as edges and corners are consistent across the distribution of models, whereas high-level features like shape and context are more effectively modeled with the Bayesian CNN block. Furthermore, the decoder maps the latent representation to a semantic segmentation result, taking into account both high-level and low-level information.

3.5. Scoring Networks

In this study, we aim to identify conflicting information among inconsistent labels within a mini-batch by computing the relative quality score. During the training process, different levels of inconsistency among annotations may exhibit varying rates of loss descent. Upon training completion,

the predicted segmentations for the inconsistent labels could present a higher uncertainty loss in comparison with well-annotated ones. In this context, we postulate that a larger variance in segmentation error corresponds to greater label inconsistency.

As depicted in Fig. 2, during the training phase, the mean of segmentation errors is reweighted by the Scores, computed by the Scoring Network (SN) using the concatenation of inputs and labels. The Bayesian Neural Network (BNN) then backpropagates based on the reweighted loss $Loss_{seg}$, prioritizing those with higher uncertainty, while the SN backpropagates based on both $Loss_{score}$ (the difference between the Score and the variance of segmentation errors) and $Loss_{seg}$.

The design of the SN is based on the encoder of the BNN, excluding the Bayesian NN block. In the final layers, we incorporate a global average pooling operation to obtain the score for each input in a mini-batch. To circumvent potential overfitting, which could result in an excessively large or small score, we introduce the activation function tanh after the final layer, constraining the score within the range of $[-1, 1]$. Following processing by the softmax layer, the maximum possible ratio of two scores within the same mini-batch is confined to the range $[e^2, \infty]$.

3.6. Training Mechanism

Let us denote the training set samples as $\{x_1, \ldots, x_N\}$, with their corresponding annotations represented by $\{y_1, \ldots, y_N\}$. The segmentations of the i^{th} sample in the training set, computed M times by the Bayesian Neural Network (BNN), are designated as $\{\hat{y}_1, \ldots, \hat{y}_M\}$. Additionally, the scores of the annotations, calculated by the Scoring Network (SN), are represented by $\{s_1, \ldots, s_N\}$.

In the realm of defect detection, the background area typically greatly surpasses the defect area in size. As a remedy to this imbalance, we utilize a combination of Binary Cross Entropy (BCE) loss and Dice coefficient (Dice) loss functions:

$$BceLoss_i^j = -y_i \cdot log\left(\hat{y}_i^j\right) - (1 - y_i) \cdot log\left(1 - \hat{y}_i^j\right) \tag{4}$$

$$DiceLoss_i^j = 1 - 2 \cdot \left|V_{y_i} \bigcap V_{\hat{y}_i^j}\right| / \left(|V_{y_i}| + \left|V_{\hat{y}_i^j}\right|\right) \tag{5}$$

$$Loss_i^j = \lambda BceLoss_i^j + DiceLoss_i^j \tag{6}$$

In the aforementioned equation, V represents the segmented area of the samples, while $\lambda > 0$ acts as a hyperparameter balancing the relative significance of the two components, namely the BCE and Dice losses.

Subsequently, the mean and variance of the loss ($Loss$) are computed as follows:

$$Mean_i = \frac{1}{M} \sum_{j=1}^{M} Loss_i^j \tag{7}$$

$$Var_i = \frac{1}{M} \sum_{j=1}^{M} \left(Loss_i^j - Mean_i \right) \tag{8}$$

Hence, the loss associated with segmentation is computed as:

$$Loss_{seg} = \sum_{i=1}^{N} s_i \cdot Mean_i \tag{9}$$

Extending this, we establish the label for the score as follows:

$$ls_i = softmax\left(1 - \frac{e^{-\hat{\lambda} \cdot Var_i}}{1 + e^{-\hat{\lambda} \cdot Var_i}}\right), i = 1, \ldots, N \tag{10}$$

where $\hat{\lambda}$ serves as a hyperparameter designed to adjust for variance.

Finally, the score loss is defined as:

$$Loss_{score} = \frac{1}{N} \sum_{i=1}^{N} -ls_i \cdot log(s_i) - (1 - ls_i) \cdot log(1 - s_i) \tag{11}$$

3.7. Sematic Segmentation Probability

Finally, we approximate the semantic segmentation probability utilizing Monte Carlo integrations. As illustrated in Fig. 2, a sample from the testing set is denoted by x^*. $\{y_i^*, \ldots y_K^*, \}$ represents the segmentation of x^* calculated K times by the BNN, while s_k^* is computed by the SN via the concatenation of y_k^* and x^*. Following eq. 3, the probability is determined as:

$$p(x^*) \approx \sum_{k=1}^{K} s_k^* \cdot y_k^* \tag{12}$$

4. Experiments and Results

4.1. Implementation details

4.1.1. Parameters setting

The mini-batch size is steadfastly maintained at 4, with an initial learning rate of 0.003, subsequently decreasing at a rate of 0.0001. Furthermore, the Dropout probability is set at 50%.

4.1.2. Computation platform

Our implementation of Score-BNN is constructed in the PyCharm environment, leveraging the open-source Pytorch toolkit. The model's training is carried out on a high-performance server, outfitted with an NVIDIA Tesla A100 GPU (boasting 40GB memory) and operating on a CentOS 8 Linux system. Moreover, to ensure the practical applicability of our method in industrial settings, we have evaluated the Score-BNN on an edge computing device setup, employing an GTX 1070 GPU, endowed with 8GB of memory.

4.2. Dataset

In this study, we have selected five datasets to examine the applicability and generalizability of our proposed method. These datasets encompass three benchmarks: NEU-Seg [24], MT [25] defect dataset, and KSDD [26] defect dataset. Additionally, we included two datasets derived from real industrial production lines: MCSD-Seg [18], which focuses on motor commutators, and WDD, which is dedicated to wafers.

4.3. Evaluation Metrics

To assess the segmentation capability, we employ intersection-over-union (IoU) and pixel-accuracy (PA) as performance metrics, using them to compare with other segmentation methods. To manifest the network's capacity for modeling uncertainty, we utilize a lower approximation for accuracy computation and an upper approximation for recall rate calculation. Ultimately, we adopt F1-Score, a measure that combines both accuracy and recall rate.

4.4. Ablative Study

4.4.1. An ablation study of various modules

In order to validate the effectiveness of each module in our proposed method, we conduct a series of ablation studies. Initially, we utilize CNN

Table 1: Performance results of various module on the NEU-Seg Dataset

	CNN	Bayesian CNN block	Score Net	IoU	PA
s1	√			0.7590	0.9676
s2	√	√		0.7693	0.9693
s3	√		√	0.7570	0.9670
s4	√	√	√	0.7771	0.9696

Table 2: Performance results of diverse architectural variants on the NEU-Seg Dataset

		IoU	PA
s1	No Dropout	0.7570	0.9670
s2	Dropout Encoder	0.7559	0.9674
s3	Dropout Decoder	0.5373	0.9343
s4	Dropout Center	0.7617	0.9676
s5	Dropout Classifier	0.6538	0.9511
s6	Dropout Central Enc-Dec	0.7733	0.9691
s7	Dropout Central Enc	0.7771	0.9696

networks devoid of the Bayesian CNN block and SN as a control group. Following this, we deploy the combination of CNN and SN to ascertain the efficacy of the SN. The combination of CNN and Bayesian CNN block is utilized to verify the potency of the Bayesian CNN block. Finally, the amalgamation of all three elements constitutes our network.

As demonstrated in Table 1, the IoU of the Bayesian CNN block improves by 1.03% in comparison to the control group, and our proposed method further enhances it by 0.78% on this basis. Conversely, it can be observed that the standalone SN incurs almost no improvement. Based on our analysis, the SN operates in concert with the Bayesian neural network, functioning under the precondition of variance in multiple solutions.

4.4.2. Various architectural variants

In an effort to substantiate the assertions made in section 3.4, we assessed the performance of the Bayesian CNN Block in various positions while maintaining constant network structure and dataset. As shown in Tabel 2 Scenarios included no Dropout (s1), presence of Dropout only in the Encoder section (s2), Decoder section (s3), center section (s4), final classification operation

Table 3: Comparison results on NEU-Seg datasets

Methods	IoU	PA	F1
UNet	0.7590	**0.9696**	0.8659
SegNet	0.7241	0.9613	0.8421
FCN	0.7372	0.9646	0.8490
PSPNet	0.6453	0.9524	0.7799
RefineNet	0.5600	0.9124	0.7355
DeepLabv3+	0.7350	0.9643	0.8906
PGANet	0.7517	0.9668	0.8620
Bayes-SegNet	0.7477	0.9650	0.9168
Ours	**0.7771**	**0.9696**	**0.9329**

(s5), center and the two layers of Encoder and Decoder near the center (s6), and finally, center and two layers of Encoder proximal to the center (s7).

The results revealed a detrimental effect of Dropout on the Decoder part and classification operation. However, positive influences were evident in the Encoder and center sections, with the latter demonstrating a stronger effect. This corroborates our stance that the Bayesian operation exhibits superior performance in the high-level feature layer. Ultimately, the most optimal effect was observed when the Bayesian operation was applied at the center layer and the Encoder layer adjacent to the center, achieving an IoU of 77.71%.

4.5. Comparison results with state-of-the-Art methods on five datasets

Initially, we juxtapose our method with Bayes-SegNet [16], a semantic segmentation model premised on Bayesian neural networks. Following that, we compare our methodology with conventional semantic segmentation models, including Unet [23], SegNet[27], FCN [28], PSPNet[29], RefineNet[30], Deeplab[31], and PGANet [32]. For a clearer visual comparison, the final semantic segmentation results are presented as heat maps.

4.5.1. Results on NEU-Seg dataset

The visual comparison between our approach and other methods for NEU-Seg images is illustrated in Fig. 4. Due to the complexities in hot-rolled conditions, some of the inclusion defects (row 1-2) and the patches defects (row 6-7) exhibit discernible gradual regions at the boundaries, and the scratches

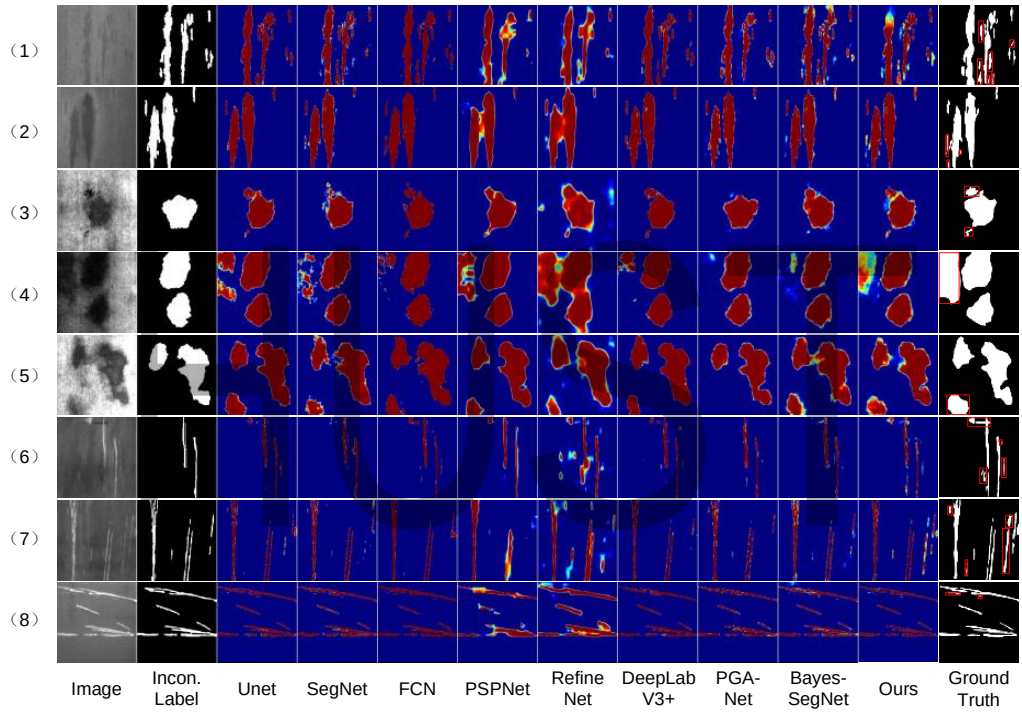

Figure 4: Visualization results on the NEU-Seg dataset

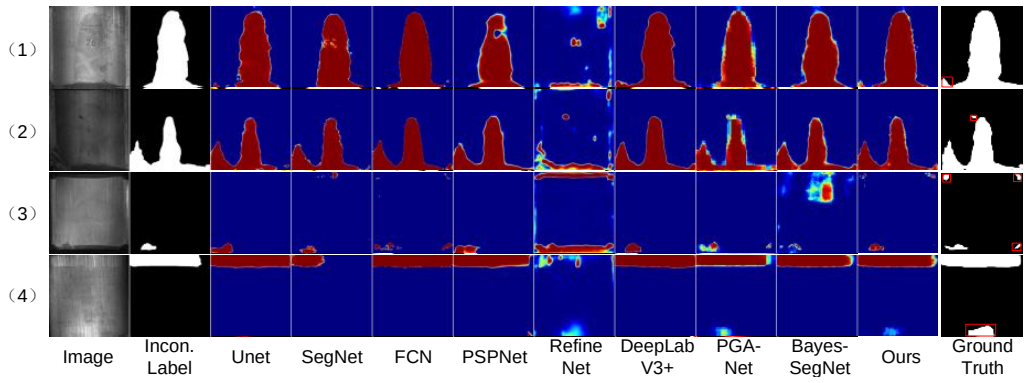

Figure 5: Visualization results on the MTDD dataset

Table 4: Comparison results on MTDD datasets

Methods	IoU	PA	F1
UNet	0.7358	0.9748	0.8548
SegNet	0.6614	0.9739	0.8028
FCN	0.7144	0.9821	0.8363
PSPNet	0.5463	0.9745	0.7002
RefineNet	0.0982	0.8832	0.3723
DeepLabv3+	0.7286	0.9802	0.8468
PGANet	0.7165	**0.9818**	0.8182
Bayes-SegNet	0.7211	0.9770	**0.9112**
Ours	**0.7573**	0.9796	0.8965

defects (row 3-5) contrast poorly with the background. Consequently, in the inconsistent labels, some defects are not annotated, while some boundaries are inaccurate. Evidently, our method outperforms the others. Score-BNN not only segments out the unlabeled parts of the image but also illustrates the transitional regions probabilistically. As demonstrated by the quantitative comparisons in table 3, our approach exceeds existing methods in all three metrics, improving the IoU value to 77.71%.

4.5.2. Results on MTDD

As depicted in Fig. 5, due to the variability of defect shapes and the randomness of lighting conditions, defects in the corners (row 1, 3), very small defects (row 2), and suspected defects (row 4) are not annotated. It can

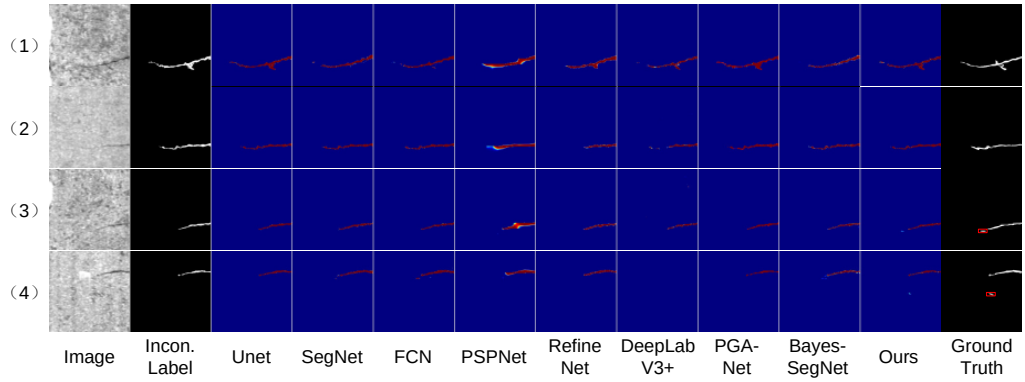

Figure 6: Visualization results on the KSDD dataset

Table 5: Comparison results on KSDD datasets

Methods	IoU	PA	F1
UNet	0.8970	0.9988	0.9463
SegNet	0.8756	0.9985	0.9334
FCN	0.8350	0.9980	0.9116
PSPNet	0.4946	0.9910	0.6661
RefineNet	0.6655	0.9956	0.8024
DeepLabv3+	0.4668	0.9937	0.6531
PGANet	0.7859	0.9969	0.8780
Bayes-SegNet	0.8818	0.9986	0.9620
Ours	**0.9197**	**0.9991**	**0.9621**

be observed that the performance of our method more closely approximates the real scenario. As outlined in table 4, our method enhances the IoU value to 75.73%.

4.5.3. Results on KSDD

Fig. 6 displays a visual comparison of partial KSDD defect image detection results. The primary challenge with this dataset lies in the very small size of the defective samples, albeit the annotations are almost consistent. Nevertheless, it can be observed that our results bear a strong resemblance to the labels. Also, certain suspected areas have been marked in rows 3-4. As detailed in table 5, we have managed to elevate the IoU value to 91.97%.

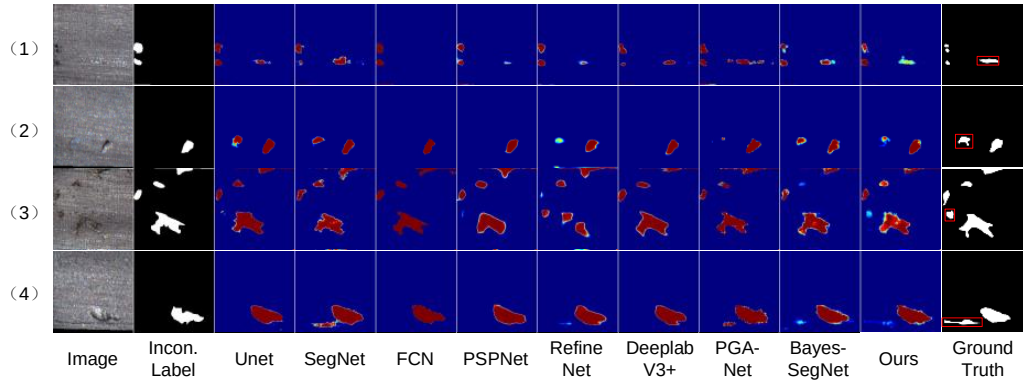

Figure 7: Visualization results on the MCSD-Seg dataset

Table 6: Comparison results on MCSD-Seg datasets

Methods	IoU	PA	F1
UNet	0.8336	0.9923	0.9058
SegNet	0.7347	0.9911	0.8512
FCN	0.8194	0.9918	0.8889
PSPNet	0.7948	0.9895	0.8807
RefineNet	0.6179	0.9867	0.7810
DeepLabv3+	0.8537	**0.9932**	0.9141
PGANet	0.7611	0.9882	0.8471
Bayes-SegNet	0.8361	0.9922	0.9449
Ours	**0.8558**	0.9931	**0.9456**

4.5.4. Results on MCSD-Seg

Fig. 7 showcases a portion of the MCSD-Seg results and the corresponding predictions. As the surface of the motor commutator images is curved, it leads to uneven illumination, making some defects appear with low contrast against the background, and brightening some areas. It can be discerned that BNN-SDD sensitively detects the suspicious sections of the image, and its contours align more closely with the actual situation. As indicated in Table 6, the IoU value has been elevated to 85.58%, with PA and F1-Score also delivering superior performance.

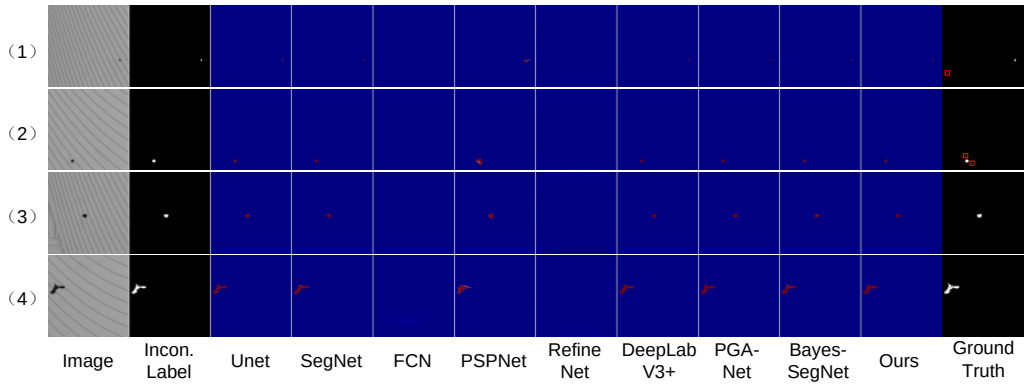

Figure 8: Visualization results on the WDD dataset

Table 7: Comparison results on WDD datasets

Methods	IoU	PA	F1
UNet	0.8213	0.9999	0.9049
SegNet	0.8604	0.9999	0.9272
FCN	0.0698	0.9995	0.1305
PSPNet	0.2929	0.9993	0.5720
RefineNet	0.0698	0.9995	0.1305
DeepLabv3+	0.7211	0.9999	0.8365
PGANet	0.6733	0.9999	0.7894
Bayes-SegNet	0.7572	0.9999	0.9084
Ours	**0.9109**	**1.0000**	**0.9558**

Table 8: Real-time analysis

Methods	Times(ms)	Model Size (MB)
UNet	7.47	29.96
SegNet	9.29	112.32
FCN	7.62	85.27
PSPNet	14.21	177.70
RefineNet	33.14	450.16
DeepLabV3+	24.23	226.37
PGANet	19.07	198.36
Bayes-SegNet	11.89	112.32
Ours	14.90	47.93

4.5.5. Results on WDD

Fig. 8 depicts the WDD, wherein the background exhibits dense intersecting lines of varying shapes. These lines mingle with defects, creating noise that hinders inspection. Concurrently, these defects are small in size, possess a diversity of shapes, and some have indistinct features with low contrast. Nevertheless, the WDD is annotated by professional wafer inspection engineers, providing high-quality labels. Hence, this serves as a control group to underscore that our method demonstrates potent semantic segmentation in high-quality datasets. Consequently, as shown in Table 7, the IoU is amplified to 91.09%.

4.6. Real-time Study

To the best of our knowledge, automated optical inspection necessitates high real-time performance. As illustrated in Table 8, we evaluate computational efficiency on a standard personal computer configuration. The number of calculations by BNN is set at $M = 16$. As can be observed, our proposed method not only offers a more compact model size but also achieves defect detection speed in real-time.

5. Conclusion

In this paper, we have introduced the Scoring Bayesian Neural Networks for Surface Defect Detection (Score-BNN), a novel framework that explicitly addresses the challenge of inconsistent labeling in surface defect detection.

Our approach leverages the concept of uncertainty derived from Bayesian Neural Networks to generate probabilistic, pixel-wise segmentation, effectively overcoming the biases and errors often embedded within the labeling process.

Our method capitalizes on inconsistencies in the labeling process, viewing them as indicators of inherent uncertainties rather than as mere noise. The proposed Scoring Networks (SN) use variance from multiple solutions of the Bayesian Neural Networks as an innovative measure of label quality, facilitating better utilization of available data and improving overall detection performance.

Our extensive experiments on five publicly available datasets validated the superiority of our approach over state-of-the-art methods. The results demonstrated robustness and adaptability to real-world industrial surface defect detection tasks.

In conclusion, this research highlights the potential of embracing uncertainty and label quality considerations in the development of more robust and reliable deep learning models for surface defect detection. The methodologies we have proposed provide significant insights that will contribute to advancements in the field.

References

[1] J. Xing, M. Jia, A convolutional neural network-based method for workpiece surface defect detection, Measurement 176 (2021) 109185.

[2] Q. Luo, X. Fang, L. Liu, C. Yang, Y. Sun, Automated visual defect detection for flat steel surface: A survey, IEEE Transactions on Instrumentation and Measurement 69 (3) (2020) 626–644.

[3] C. Huang, G. Wang, H. Song, R. Li, H. Zhang, Rapid surface defects detection in wire and arc additive manufacturing based on laser profilometer, Measurement 189 (2022) 110503.

[4] W. Ming, F. Shen, X. Li, Z. Zhang, J. Du, Z. Chen, Y. Cao, A comprehensive review of defect detection in 3c glass components, Measurement 158 (2020) 107722.

[5] A. S. Rich, T. M. Gureckis, Lessons for artificial intelligence from the study of natural stupidity, Nature Machine Intelligence 1 (4) (2019) 174–180.

[6] H. Song, M. Kim, D. Park, Y. Shin, J.-G. Lee, Learning from noisy labels with deep neural networks: A survey, IEEE Transactions on Neural Networks and Learning Systems (2022).

[7] Z. Liao, Y. Xie, S. Hu, Y. Xia, Learning from ambiguous labels for lung nodule malignancy prediction, IEEE Transactions on Medical Imaging 41 (7) (2022) 1874–1884.

[8] D. Karimi, H. Dou, S. K. Warfield, A. Gholipour, Deep learning with noisy labels: Exploring techniques and remedies in medical image analysis, Medical image analysis 65 (2020) 101759.

[9] E. Simpson, I. Gurevych, A bayesian approach for sequence tagging with crowds, arXiv preprint arXiv:1811.00780 (2018).

[10] Y. Yao, J. Deng, X. Chen, C. Gong, J. Wu, J. Yang, Deep discriminative cnn with temporal ensembling for ambiguously-labeled image classification, in: Proceedings of the aaai conference on artificial intelligence, 2020, pp. 12669–12676.

[11] Z. Yu, W. Liu, Y. Zou, C. Feng, S. Ramalingam, B. Kumar, J. Kautz, Simultaneous edge alignment and learning, in: Proceedings of the European conference on computer vision (ECCV), 2018, pp. 388–404.

[12] D. Acuna, A. Kar, S. Fidler, Devil is in the edges: Learning semantic boundaries from noisy annotations, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 11075–11083.

[13] 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.

[14] Y. Gal, Z. Ghahramani, Bayesian convolutional neural networks with bernoulli approximate variational inference, arXiv preprint arXiv:1506.02158 (2015).

[15] R. M. Neal, Bayesian learning for neural networks, Vol. 118, Springer Science & Business Media, 2012.

[16] A. Kendall, V. Badrinarayanan, R. Cipolla, Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding, arXiv preprint arXiv:1511.02680 (2015).

[17] S. Niu, Y. Peng, B. Li, Y. Qiu, T. Niu, W. Li, A novel deep learning motivated data augmentation system based on defect segmentation requirements, Journal of Intelligent Manufacturing (2023) 1–15.

[18] T. Niu, B. Li, W. Li, Y. Qiu, S. Niu, Positive-sample-based surface defect detection using memory-augmented adversarial autoencoders, IEEE/ASME Transactions on Mechatronics 27 (1) (2021) 46–57.

[19] S. Ma, K. Song, M. Niu, H. Tian, Y. Wang, Y. Yan, Shape consistent one-shot unsupervised domain adaptation for rail surface defect segmentation, IEEE Transactions on Industrial Informatics (2023).

[20] Y. Gal, Z. Ghahramani, Dropout as a bayesian approximation: Insights and applications, in: Deep Learning Workshop, ICML, Vol. 1, 2015, p. 2.

[21] G. Ghiasi, T.-Y. Lin, Q. V. Le, Dropblock: A regularization method for convolutional networks, Advances in neural information processing systems 31 (2018).

[22] X. Li, S. Chen, X. Hu, J. Yang, Understanding the disharmony between dropout and batch normalization by variance shift, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 2682–2690.

[23] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18, Springer, 2015, pp. 234–241.

[24] K. Song, Y. Yan, A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects, Applied Surface Science 285 (2013) 858–864.

[25] Y. Huang, C. Qiu, K. Yuan, Surface defect saliency of magnetic tile, The Visual Computer 36 (2020) 85–96.

[26] J. Gan, Q. Li, J. Wang, H. Yu, A hierarchical extractor-based visual rail surface inspection system, IEEE Sensors Journal 17 (23) (2017) 7935–7944.

[27] V. Badrinarayanan, A. Kendall, R. Cipolla, Segnet: A deep convolutional encoder-decoder architecture for image segmentation, IEEE transactions on pattern analysis and machine intelligence 39 (12) (2017) 2481–2495.

[28] 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.

[29] H. Zhao, J. Shi, X. Qi, X. Wang, J. Jia, Pyramid scene parsing network, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 2881–2890.

[30] G. Lin, A. Milan, C. Shen, I. Reid, Refinenet: Multi-path refinement networks for high-resolution semantic segmentation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1925–1934.

[31] 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 transactions on pattern analysis and machine intelligence 40 (4) (2017) 834–848.

[32] 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 Transactions on Industrial Informatics 16 (12) (2019) 7448–7458.