RESEARCH ARTICLE

ADVANCED
INTELLIGENT
SYSTEMS
Open Access

www.advintellsyst.com

PCB Soldering Defect Inspection Using Multitask Learning under Low Data Regimes

Sik-Ho Tsang, Zhaoqing Suo, Tom Tak-Lam Chan, Huu-Thanh Nguyen, and Daniel Pak-Kong Lun*

To increase the reliability of the printed circuit board (PCB) manufacturing process, automated optical inspection is often employed for soldering defect detection. However, traditional approaches built on handcrafted features, predefined rules, or thresholds are often susceptible to the variation of the acquired images' quality and give unstable performances. To solve this problem, a deep learning-based soldering defect detection method is developed in this article. Like many real-life deep learning applications, the number of available training samples is often limited. This creates a challenging low-data scenario, as deep learning typically requires massive data to perform well. To address this issue, a multitask learning model is proposed, namely, PCBMTL, that can simultaneously learn the classification and segmentation tasks under low-data regimes. By acquiring the segmentation knowledge, classification performance is substantially improved with few samples. To facilitate the study, a soldering defect image dataset, namely, PCBSPDefect, is built. It focuses on the dual in-line packages (DIP) at the PCB back side, DIP at the PCB front side, and flat flexible cables. Experimental results show that the proposed PCBMTL outperforms the best existing approaches by over 5–17% of average accuracy for different datasets.

1. Introduction

Quality assurance of the printed circuit boards (PCBs) and PCB assemblies (PCBAs) is vital for electronic product manufacturing. Instead of relying on laborious, costly, and subjective manual inspection, an intelligent automatic optical inspection (AOI) system can be employed to detect defects and aid human operators in decision-making. By utilizing such a system, the time required for inspecting soldering defects can be reduced, resulting in reduced human and time costs. However, traditional AOI systems for PCB soldering defect detection rely heavily on the quality of the acquired PCB images. When a lower-cost imaging system is used, the variation in the image quality will introduce much difficulty to the traditional detection approaches which are built on handcrafted features, predefined rules, or thresholds.[1–7] Besides, these traditional approaches are often computationally intensive which affects their real-time performance. The rise of deep learning has spurred the advancement and implementation of AOI systems.[8] Current approaches often treat PCB soldering defect detection as a supervised image classification problem.[9–13] Vanilla deep learning models are used and trained with thousands of PCB defect images. In fact, for most industrial-grade PCB manufacturing processes, soldering defects are not common. It will take a long time and huge manpower to collect sufficient PCBs with soldering defects to construct a dataset for deep neural network (DNN) training. It is particularly difficult for new production lines where the AOI system needs to be in place before production starts. The PCB samples that can be used for model training are rather limited.

In this article, we propose to adopt the multitask learning (MTL) method for PCB soldering defect detection under the low-data regime. Specifically, we propose to add another head to the model to segment the soldering points. Segmentation tasks are generally more difficult than classifications as they involve predicting the class of every pixel. We hypothesize that if a model is capable of achieving good segmentation of the solder regions in a PCB, it should have learned rich semantic features, leading to an improvement in the feature representation learning for PCB images, and ultimately enhancing the classification performance of soldering defects. Thereby, our strategy to address the low-data problem is to utilize the semantic feature knowledge obtained from the segmentation mask to assist the low-data training. While the segmentation task requires extra labels of soldering point positions, it may also be considered that the proposed approach tackles the low-data problem by using more labels in lieu of training data.

S.-H. Tsang, Z. Suo, T. T.-L. Chan, H.-T. Nguyen, D. P.-K. Lun
Unit 1212-1213
Hong Kong Science Park
12/F, Building 19W, Pak Shek Kok, NT, Hong Kong, China
E-mail: pak.kong.lun@polyu.edu.hk

Z. Suo, D. P.-K. Lun
Department of Electrical and Electronic Engineering
The Hong Kong Polytechnic University
Hung Hom, Hong Kong, China

The ORCID identification number(s) for the author(s) of this article can be found under https://doi.org/10.1002/aisy.202300364.

To verify the above idea, a new PCB soldering defect detection model called PCBMTL is proposed. The new model has a U-Net-like structure but has two heads for segmentation and defect classification, respectively.[14] For training the model, an image dataset with PCB soldering defects is needed. There are however not many open-source datasets related to PCB images, and all of them are not about PCB soldering defects.[15–23] For this reason, a new PCB soldering defect dataset, namely, PCBSPDefect is constructed.[24] The dataset contains images of PCB soldering points of three component parts, namely, dual in-line packages (DIP) located on the PCB back side (BDIP), DIP on the PCB front side (FDIP), and the flat flexible cables (FFC), as depicted in **Figure 1**. These components are typical in conventional electronic circuit boards. Ten classes of soldering defects can be found in the images. They are all typical soldering defects commonly found in PCBs. These images are captured from real defective PCBs. The dataset will be made publicly available for research purposes.

2. Background

2.1. Related Works

AOI for soldering defect detection has been widely studied for decades. The early studies focused on traditional rule-based image processing techniques for detecting soldering defects.[1,2] While they might use statistical or probabilistic techniques, they were all rule-based approaches. Even in recent years, there were still some rule-based methods used for detecting the defects of PCB through-holes and water pump PCB soldering points.[3,4] These rule-based methods relied on handcrafted features, predefined rules, or thresholds, which were difficult to generalize and led to inferior performance in new applications.

In recent years, machine learning (ML) techniques such as K-means clustering, artificial neural networks (ANN), and multilayer perceptron (MLP) have emerged to become the main tools for classification tasks.[25,26] They also gained popularity for soldering quality inspection.[5–7] However, these approaches still use handcrafted features. In addition, ANN or MLP models are relatively shallow and may not be able to aggregate image representation features, which can limit the overall performance.[6,7]

To solve the problem, deep learning techniques were developed and demonstrated to outperform traditional ML methods.[8] They were applied to different defect detection tasks, such as detecting the defects of contemporary artworks and carbon fiber composites.[27,28] DNN techniques were also used in PCB soldering defect detection. For instance, object detection models were used for simultaneously localizing and classifying soldering defects.[29,30] A rule-based approach was used to crop out the soldering points, followed by a convolutional neural network (CNN) to classify whether the soldering point was normal or not.[9] CNN was also adopted for USB soldering point classification after localizing the USB connections.[10] VGGNet, a variant of CNN, was modified to classify abnormal solder joints.[11,12] For soldering defect classification in X-ray modality, 2D CNN and 3D CNN were designed for detecting defects in 2D and 3D X-ray images, respectively.[13] While these approaches achieve some success in defect detection, they require a large number of images (at least 4000) for training, which is hard to achieve in many practice situations.

2.2. Learning Approaches for Low-Data Scenarios

One of the main challenges of the abovementioned deep learning approaches is the need for large amounts of high-quality labeled data. In many real-world scenarios, obtaining such data can be difficult, time-consuming, and expensive. While it is a common
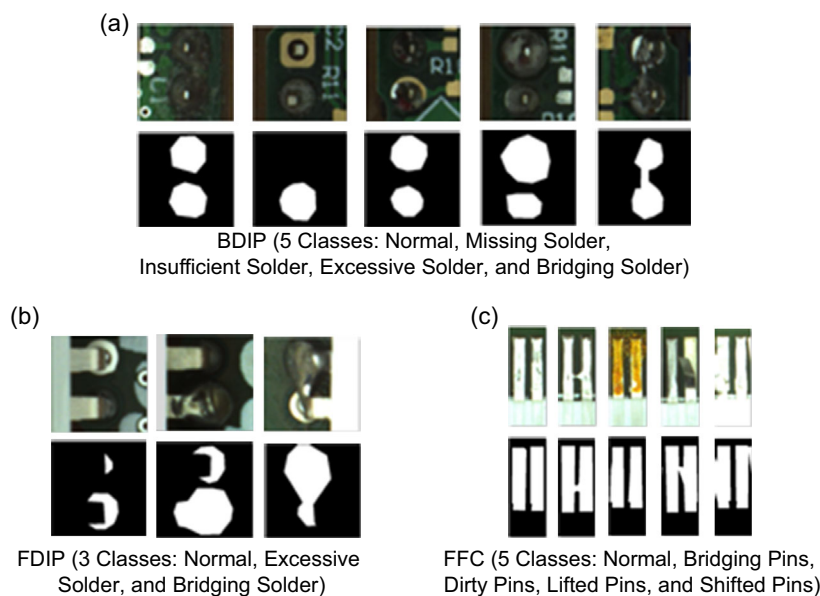

Figure 1. Image samples and the corresponding binary segmentation masks in the proposed PCBSPDefect dataset for soldering defect detection: a) DIP at PCB back side (BDIP), b) DIP at PCB front side (FDIP), and c) flat flexible cables (FFC).

**ADVANCED
SCIENCE NEWS**
www.advancedsciencenews.com

**ADVANCED
INTELLIGENT
SYSTEMS**
www.advintellsyst.com

problem for deep learning research, various approaches under low-data regimes have been developed to address this issue.

One approach is pretraining or transfer learning. The model is first pretrained using a supervised or self-supervised approach based on a massive external dataset such as ImageNet.[31–33] Rich feature representations of the data are learned and then adapted to a smaller dataset through fine-tuning. Although self-supervised learning (SSL) has shown promise in learning representations for downstream applications without domain data, the incurred high cost of the required computational resources as well as the long training time mean the solution will be expensive to the end users. While it is possible to use SSL pretrained models to avoid the training process, they are only available for a few popular ones. They may not be suitable for the requirement of PCB soldering defect detection.

Data synthesizing is another common approach for dealing with the low-data problem. Negative samples can be synthesized from the positive samples (which are assumed to be obtained more easily) through image processing or deep learning techniques.[34,35] However, the accuracy of the model will largely depend on the accuracy of the synthesizer, which is not available for general PCB soldering defects. Another approach is data augmentation to generate samples for training. This can be done by simple operations such as random cropping, rotating, and flipping image samples, or more sophisticated approaches such as Copy–Paste and using generative adversarial networks (GAN).[34,35] Data augmentation is commonly practiced in deep learning applications for reducing overfitting. When using it to solve the low-data problem, the data variability or the quality of the augmented images will not be high enough to replace the true soldering defect samples.

Multitask learning (MTL) is another approach that can be used under low-data regimes. In MTL, a single model is trained to perform multiple related tasks simultaneously, sharing and learning from common representations. This approach can leverage task-specific information to improve the model's performance on each task, even with limited data. The shared representations learned by the model can capture common features across tasks and help regularize the model, reducing overfitting to the limited data. In this article, the MTL technique is applied to PCB soldering defect detection under low-data regimes. It is the first work that addresses the low-data problem in training PCB soldering defect detection models. When the available training data is only 10% of a normal dataset, the proposed MTL approach can still achieve 80% or higher detection accuracy, which significantly outperforms the conventional approaches.

2.3. Contributions

The contribution of this work has three folds: 1) We propose a novel PCB soldering defect detection DNN model namely PCBMTL. It adopts the multitask learning (MTL) approach that allows for improved classification accuracy using a limited number of PCB samples, even under low-data regimes. This is achieved by leveraging the acquired segmentation knowledge, which helps the model better understand the features and characteristics of the different components on the PCB. By training the model to perform both segmentation and classification tasks

simultaneously, we can improve the model's ability to accurately classify defects, even if only a limited number of training samples are available. 2) We developed a new PCB image dataset namely PCBSPDefect for soldering defect detection.[24] The dataset contains images of PCB soldering points of DIP and FFC components with 10 classes of soldering defects. This dataset provides not only image-level labels but also pixel-wise binary segmentation masks that indicate the regions where the PCB has solder. The dataset will be released for public download. 3) We have evaluated the performance of our PCBMTL model using the PCBSPDefect dataset. The results show that the proposed PCBMTL outperforms state-of-the-art methods under different amounts of training data. This indicates the potential of PCBMTL for effective PCB soldering defect classification using limited data. Additionally, the segmentation results can provide valuable insights into the model's decision-making process and can help identify areas for further improvement.

3. Proposed Multitask Learning Approach

3.1. New PCB Soldering Defect Image Dataset

As mentioned in Section 1, there is no open-sourced PCB soldering defect dataset publicly available. To facilitate the study, a new PCB soldering defect image dataset was developed. The new dataset contains soldering point images of DIP and FFC components. The images were captured using a camera placed under a white color ring light, as illustrated in **Figure 2**a. A jig was used to fix the position of the PCBs, enabling easy cropping of BDIP, FDIP, and FFC soldering point images, as depicted in Figure 1 and 2b.

The soldering defects in BDIP are divided into four classes, namely, missing solder, insufficient solder, excessive solder, and bridging solder. In FDIP, there are two classes, including excessive solder and bridging solder. Whereas, for FFC, the defect classes are bridging pins, dirty pins, lifted pins, and shifted pins. Normal PCB images are also added to each category. Every image contains two soldering points or pins so that the problems of bridging solder, bridging pins, and shifted pins can be clearly seen. It also enables us to generalize the dataset to PCBs with different numbers of DIP and FFC soldering points or pins. The images were captured under three brightness levels, as displayed in Figure 2b, to simulate various lighting conditions
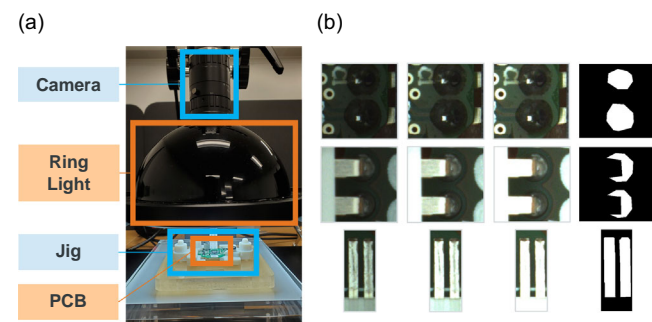

Figure 2. a) Image acquisition system and b) BDIP, FDIP, and FFC image samples captured under different lighting levels and their corresponding segmentation masks.

**ADVANCED
SCIENCE NEWS**

www.advancedsciencenews.com

**ADVANCED
INTELLIGENT
SYSTEMS**
Open Access

www.advintellsyst.com

in real environments. To generate the ground-truth binary segmentation mask, we utilized LabelMe annotation software to label the pixels with solder as foreground (white) and the remaining pixels as background (black).

Using the above setup, three datasets were created for BDIP, FDIP, and FFC, respectively, with around 6,000 images in total. Note that different data augmentation methods were applied to generate the images in the datasets. The details of each dataset are summarized in **Table 1**. To simulate the low-data scenario, a 20:20:60 split ratio is employed for the training, validation, and test sets, respectively. The exact numbers of the training images are shown in the last row of Table 1. To study the effect of training data size, the training set is further reduced by 2% incrementally down to 10% while keeping the validation and test sets unchanged. Each smaller training set is a proper subset of a larger training set, i.e.

$$D_p^{(\text{Train})} \subset D_q^{(\text{Train})} \quad \text{for} \quad p < q \tag{1}$$

where $D_p^{(\text{Train})}$ and $D_q^{(\text{Train})}$ are the training set $D^{(\text{Train})}$ with p and q percent of the total training samples, respectively (with p smaller than q). The sets are also balanced to ensure an equal proportion of different classes. To avoid overlapping, the same sample captured with different lighting levels is only included in one of the three sets.

Table 1. Summary of the BDIP, FDIP, and FFC datasets. (Numbers within brackets are the number of PCBs used).

Train [%]	BDIP (five classes)			FDIP (three classes)			FFC (five classes)		
	Train	Val	Test	Train	Val	Test	Train	Val	Test
10	246 (21)	489	1446	126 (11)	252	744	222 (19)	438	1329
12	294 (25)			153 (13)			264 (22)		
14	342 (29)			180 (15)			306 (26)		
16	390 (33)			198 (17)			348 (29)		
18	441 (37)			225 (19)			393 (33)		
20	489 (41)			252 (21)			438 (37)		

3.2. Proposed Multitask Learning Framework

The main concept behind MTL is to extract and utilize the shared knowledge and information among multiple tasks so that the MTL model can learn multiple tasks concurrently, thereby enhancing the learning efficiency by exploiting the common features across tasks. It is adopted in the design of the proposed model PCBMTL to deal with the low-data problem due to the difficulties in collecting training samples for PCB soldering defect detection, as mentioned in Section 1. The proposed PCBMTL is an end-to-end CNN model that integrates classification and segmentation. This model is capable of taking an input soldering point (pin) image of BDIP, FDIP, or FFC, and generating two outputs: a binary pixel-wise segmentation map of the regions with solder and an image-level soldering defect type prediction. The model architecture depicted in **Figure 3** is an encoder–decoder structure which is similar to some popular structures for image segmentation such as U-Net.[14] The proposed architecture comprises three parts: 1) an encoder for high-level feature representation learning, 2) a decoder or a segmentation branch for pixel-wise segmentation, and 3) an additional branch for classification. Specifically, given an image $I \in \mathbb{R}^{H \times W \times C}$, where $H \times W$ is the corresponding pixel-wise spatial resolution and C is the number of color channels, the proposed model predicts the segmentation map $\hat{M} \in \mathbb{R}^{H \times W}$ and the corresponding image-level soldering class $\hat{y}$.

3.2.1. Encoder

The input images are gradually encoded into low-resolution high-level feature representation by the encoder, as depicted in the left portion of Figure 3. Each level in the encoding branch contains two convolutional blocks, as described by a 3×3 convolution layer (Conv$_{3 \times 3}$) with a stride of 1, followed by batch normalization (BN) and rectifier linear unit (ReLU) activation function in sequential order.[36,37] Unlike U-Net, our model employs BN, which is useful to stabilize the model training. Downsampling (Down) is achieved through the application of a 2×2 max pooling layer with a stride of 2 between two levels, which reduces the
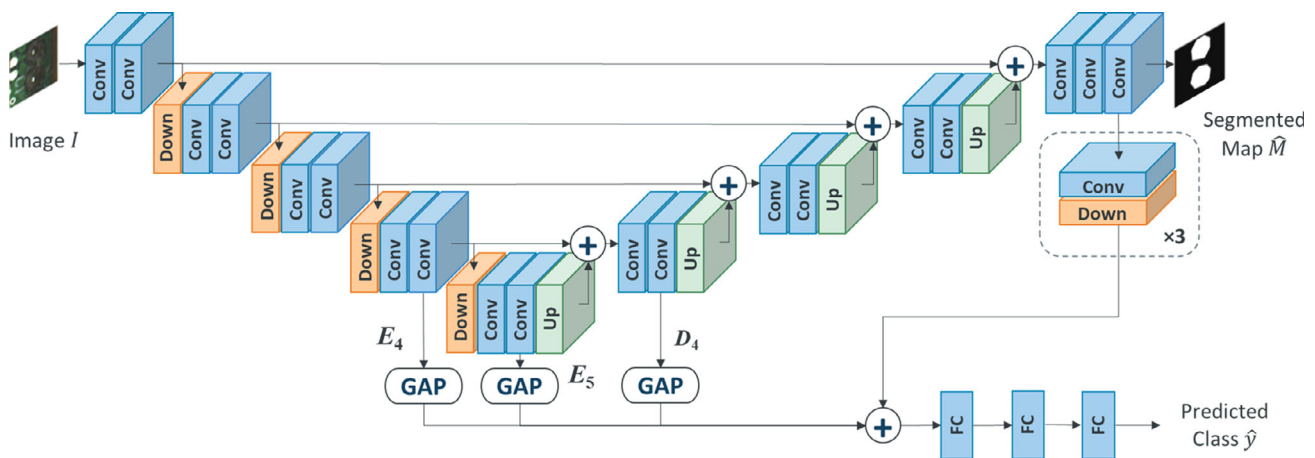

Figure 3. Overview of the proposed MTL framework, PCBMTL. The convolutional layer, fully connected layer, downsampling, upsampling, and global average pooling are denoted as Conv, FC, Down, Up, and GAP, respectively.

spatial dimension by 2 and increases the number of channels by 2. Max pooling is used because it is a simple operation that picks the maximum value of the pooling area from the previous layer, meaning that the most representative feature is selected.[38] The feature maps generated at the end of each level prior to down-sampling can be expressed formally as

$$E_l = \begin{cases} \text{Conv}_{3\times3}(\text{Conv}_{3\times3}(I)) & \text{for } l = 1 \\ \text{Conv}_{3\times3}(\text{Conv}_{3\times3}(\text{Down}(E_{l-1}))) & \text{for } l \in [2, L] \end{cases} \quad (2)$$

where E_l is the encoder features obtained at level l. BN and ReLU are omitted for the sake of simplicity. This procedure is repeated for L times in the encoder.

After passing the input image through the encoder, a total of L levels of image feature maps $\{E_l\}_{l=1}^{L}$ are extracted, where L is set to 5. To perform soldering defect classification, a subset of E_l levels is fed into the classification branch. When using MTL, the extracted features are also passed into the decoder to provide spatial information for predicting a segmentation mask of the regions with solder.

3.2.2. Decoder

In the decoder, the feature maps are upsampled to obtain the predicted segmentation mask $\hat{M}$ at the input resolution. As depicted in the right part of Figure 3, skip connections are used to concatenate the corresponding encoder features with the decoder features to obtain finer segmentation masks with more spatial details.[31] The fused features are then processed by two convolutional blocks to further enhance the feature representation

$$D_l = \begin{cases} \text{Conv}_{3\times3}(\text{Conv}_{3\times3}(\text{Up}(E_{l+1})\oplus E_l)) & \text{for } l = L - 1 \\ \text{Conv}_{3\times3}(\text{Conv}_{3\times3}(\text{Up}(D_{l+1})\oplus E_l)) & \text{for } l \in [1, L-2] \end{cases} \quad (3)$$

where D_l is the encoder features obtained at level l, and $\oplus$ is the concatenation operation. Bilinear interpolation is used for upsampling. The last layer of the decoder consists of a 1×1 convolution layer, which reduces the number of channels and produces the predicted binary segmentation map $\hat{M}$. The sigmoid function is applied to the output to ensure that the values lie between 0 and 1, representing the probability of each pixel that has solder. The operation can be described as follows

$$\hat{M} = \sigma(\text{Conv}_{1\times1}(D_1)) \quad (4)$$

where σ is the sigmoid activation function. Let $\hat{M}_i$ be the probability of having solder for the i-th pixel in $\hat{M}$. $\hat{M}_i$ approaches 1 when the pixel has a high probability of having solder, and vice versa. We hypothesize that the segmentation knowledge of the pixels with solder can improve the performance of the classification task. Notably, when MTL is not employed, the decoder branch is not used for segmentation prediction.

3.2.3. Classification Branch

To classify the soldering defects, a classification branch is added to the end of the network, as shown at the bottom of Figure 3.

The semantic features learned from the segmentation task are extracted and fed as inputs of the classification branch. Specifically, the classification network receives feature maps of E_{L-1}, E_L, and D_{L-1}, which are globally average pooled (GAP) as $\overline{E_{L-1}}$, $\overline{E_L}$, and $\overline{D_{L-1}}$. Additionally, the predicted binary segmentation map $\hat{M}$ can provide hints for the classification task as it indicates the predicted pixels in the image with solder. $\hat{M}$ is further processed using K consecutive 3×3 convolutional and downsampling layers to obtain enhanced features with a smaller spatial size.

$$M_k = \text{Down}(\text{Conv}_{3\times3}(M_{k-1})) \quad \text{for } k \in [1, K] \quad (5)$$

where K is set to 3, M_0 is the predicted segmentation map $\hat{M}$, and the downsampling is implemented by a 4×4 max pooling layer with a stride of 4. The resulting enhanced feature map M_K is then flattened.

Finally, E_{L-1}, E_L, D_{L-1}, and the flattened M_K are fused by concatenation and fed into the classification branch, which has three fully connected (FC) layers to predict the soldering defect type. ReLU is used for first the two layers while softmax is used for the last layer.[37] A dropout of 0.5 is used for each FC layer.[39] The whole operation is described as follows

$$\hat{y} = \text{Softmax}\left(\text{FC}\left(\text{FC}\left(\text{FC}\left(\overline{E_{L-1}}\oplus\overline{E_L}\oplus\overline{D_{L-1}}\oplus M_K\right)\right)\right)\right) \quad (6)$$

where the first two FC layers reduce the feature size by half.

3.3. Multitask Loss Functions

To train the segmentation branch, two losses are used. The first one is the binary cross-entropy loss L_{BCE} estimated between the predicted segmentation map $\hat{M}$ and the ground-truth segmentation map M

$$L_{\text{BCE}}(\hat{M}, M) = \frac{1}{|\Omega|}\sum_i^{\Omega} -M_i\log\hat{M}_i - (1 - M_i)\log(1 - \hat{M}_i) \quad (7)$$

where Ω is the set of pixels and $|\Omega|$ is the cardinality of Ω, i.e., the number of elements within Ω. Another loss is the Dice loss. Given two countable sets A and B, the Dice coefficient is defined as

$$\text{Dice}(A, B) = \frac{2|A \cap B|}{|A \cap B| + |A \cup B|} \quad (8)$$

which is useful to measure the overlapping between A and B similar to Intersection over Union (IoU). It can be observed that $\text{Dice}(A, B)$ is maximized at 1 when $A = B$ and minimized at 0 when $A \setminus B = \varnothing$. Based on the Dice coefficient, the Dice loss is defined as

$$L_{\text{Dice}}(\hat{M}, M) = 1 - 2\frac{\sum_i^{\Omega} M_i\hat{M}_i + \varepsilon}{\sum_i^{\Omega} M_i + \sum_i^{\Omega} \hat{M}_i + \varepsilon} \quad (9)$$

where ε is a very small value for avoiding numerical issues. Consequently, the segmentation loss is the sum of the binary cross-entropy loss and the Dice loss

$$L_{\text{Seg}} = L_{\text{BCE}} + \lambda_D L_{\text{Dice}} \quad (10)$$

ADVANCED
SCIENCE NEWS

www.advancedsciencenews.com

ADVANCED
INTELLIGENT
SYSTEMS

www.advintellsyst.com

where λ_D is a hyperparameter to balance L_{BCE} and L_{Dice}. For classification, the cross-entropy loss is used

$$L_{CE}(\hat{y}, y) = \sum_i^{\gamma} -\gamma \log \hat{y} \tag{11}$$

In the end, when MTL is used, the total loss is the sum of classification loss and segmentation loss, i.e.

$$L_{Total} = L_{CE} + \lambda_{Seg} L_{Seg} \tag{12}$$

where λ_{Seg} is used to balance L_{CE} and L_{Seg}. The two hyperparameters λ_D and λ_{Seg} were selected empirically based on the validation sets of PCBSPDefect. We found that the network already worked well by selecting $\lambda_D = \lambda_{Seg} = 1$ although we believe that further finetuning them may lead to even better results. When MTL is enabled, the network is trained in an end-to-end manner by minimizing (12) so that for each input image, the network requires to predict the segmentation mask well and classify the defect correctly at the same time. The training details are described in the next section. Note that when MTL is disabled, only the cross-entropy loss L_{CE} is used for model training, i.e., $\lambda_{Seg} = 0$.

We can see from (12) that, to minimize the total loss, the network needs to learn well for both classification and segmentation tasks. However, our focus is on classifying soldering defects, and therefore, we aim to utilize the insights gained from the segmentation task to enhance the performance of the classification task. Additionally, the second term in (12) can be viewed as a regularizer that enables the classification task to be constrained by the segmentation task, which avoids the learning of redundant features.

4. Experimental Results

To evaluate the performance of the proposed multitask learning framework, we used our newly developed datasets BDIP, FDIP, and FFC. The implementation of the proposed model, PCBMTL, was carried out using PyTorch and trained from scratch for 50 epochs with a batch size of 16.[40] It is noted that the number of iterations equals the number of training samples times the number of epochs then divided by the batch size. The number of iterations is different for different training percentages. Adam optimizer was used with an initial learning rate of 0.001.[41] Random horizontal flipping was applied for data augmentation during training. As we formulate the problem as a classification task, classification accuracy was used as the evaluation metric. To ensure the accuracy and reliability of our findings, the average results from five independent runs are presented in all cases, which is a common practice under low data regimes. This is necessary because when models are trained using limited samples, with random initialization, the accuracy can fluctuate. Training and testing the model five times and taking the average accuracy across these runs allows us to obtain a more reliable and representative measure of the model performance. Furthermore, as shown in Table 1, the performances of the models at different training percentages, ranging from 10% to 20%, need to be evaluated to study the impact of training data size. Thus, the 5-run

average accuracy of each model was measured for each training percentage.

4.1. Ablation Study of PCBMTL

For selecting different design parameters of PCBMTL, a comprehensive ablation study was conducted. Specifically, we performed ablation experiments on the BDIP dataset by varying the training sample percentage from 10% to 20% of the total. The model was trained using different sizes of training sets, and the corresponding validation accuracy, $\text{Acc}\left(D^{(Val)}|D_p^{(Train)}\right)$, was measured. We also calculated the average accuracy, $\text{Acc}_{Avg}^{(Val)}$, on the validation set $D^{(Val)}$ as follows

$$\text{Acc}_{Avg}^{(Val)} = \frac{1}{|P|} \sum_p^P \text{Acc}\left(D^{(Val)}|D_p^{(Train)}\right) \tag{13}$$

where P is the set of training samples with varying sizes. In this case, $p \in P = \{10, 12, 16, 20\}$.

Table 2 presents the results with different configuration settings. First, by comparing the configuration settings S1 and S2, we can see that S2, using the features of E_5 and MTL for classification, achieves an average accuracy of 76.42%, which is higher than the 69.65% accuracy obtained by S1. This demonstrates that leveraging the knowledge acquired from the segmentation task improves the classification accuracy and confirms the effectiveness of the proposed MTL approach.

In our second analysis, we examined the performance of different combinations of encoder layers E_4, E_5, and the decoder layer D_4 from settings S2 to S7. Settings S2 to S4 use single-level features extracted from E_4, E_5, and D_4, respectively. We observe that using the features from E_5 and D_4 alone for PCBMTL results in accuracies of 76.42% and 74.57%, respectively, which are much higher than using those of E_4 with an accuracy of 51.40%. This suggests that deeper layers, such as E_5 or D_4,

Table 2. Ablation study of the proposed PCBMTL on the BDIP validation set. (Here, and in following tables, the best result is bolded while the second best is underlined).

Settings	E_4	E_5	D_4	M_3	MTL	$\text{Acc}\left(D^{(Val)}|D_p^{(Train)}\right)$ [%]				$\text{Acc}_{Avg}^{(Val)}$ [%]
						$p=10$	$p=12$	$p=16$	$p=20$	
S1		✓				58.12	64.09	77.42	78.98	69.65
S2		✓			✓	61.27	72.31	83.48	88.63	76.42
S3	✓				✓	42.54	44.21	53.62	65.24	51.40
S4			✓		✓	62.13	65.36	84.58	86.22	74.57
S5	✓	✓			✓	62.99	75.58	84.79	87.53	77.72
S6		✓	✓		✓	70.92	82.86	87.20	89.20	82.55
S7	✓	✓	✓		✓	72.80	81.72	88.26	88.96	82.93
S8				✓	✓	28.63	35.66	41.19	55.50	40.25
S9		✓		✓	✓	75.13	<u>84.87</u>	**89.98**	<u>90.43</u>	<u>85.10</u>
S10	✓	✓		✓	✓	73.91	82.58	85.19	89.57	82.81
S11		✓	✓	✓	✓	**79.59**	**85.52**	<u>89.16</u>	**90.88**	**86.29**
S12	✓	✓	✓	✓	✓	<u>76.73</u>	84.13	87.89	89.78	84.63

contain higher semantic features, allowing for better knowledge acquisition. We also studied the utilization of multilevel features in settings S5 to S7. We found that S6 ($E_5 + D_4$) and S7 ($E_4 + E_5 + D_4$) achieve much higher accuracies of 82.55% and 82.93% compared to S5 ($E_4 + E_5$) with an accuracy of 77.72%. This again confirms that E_5 and D_4 contain more useful high-level features for soldering defect classification. In addition, for S5–S7, features at different levels complement each other providing higher accuracies compared to those using settings S2–S4.

Third, we also conducted experiments with the inclusion of the soldering segmentation mask feature M_3 from S8 to S12. Using M_3 in S8 for PCBMTL results in even lower accuracy than S1 which does not use MTL. This suggests that solely using the soldering segmentation mask does not help with soldering defect classification. However, when used in combination with other features, M_3 can be useful. S10 ($E_4 + E_5 + M_3$) is the setting of S5 ($E_4 + E_5$) with the addition of M_3. S10 obtained an accuracy of 82.81% which is higher than the accuracy of 77.72% by S5. Yet, the performance of S10 is on par with the ones in S6 and S7. In contrast, S9, with only $E_5 + M_3$, obtained an 85.10% accuracy, which is higher than that of S10. With the use of the features of D_4, S11 ($E_5 + D_4 + M_3$) and S12 ($E_4 + E_5 + D_4 + M_3$) obtain average validation accuracies of 86.29% and 84.63% respectively, which are among the highest among all configuration settings.

Hence, S9, S11, and S12, achieving the top-3 highest accuracies, were selected for further experiments based on their high accuracy in the ablation study. They are named, PCBMTL-2F, PCBMTL-3F, and PCBMTL-4F, respectively, for the rest of this article.

4.2. Comparisons with State-of-The-Art Approaches

Table 3–5 summarize the comparisons of the proposed PCBMTL with state-of-the-art PCB classification approaches and ResNet-50 by evaluating the corresponding test accuracy, $\text{Acc}\left(D^{(\text{Test})}|D_p^{(\text{Train})}\right)$, where $D^{(\text{Test})}$ is the test set, with $p \in P = \{10, 12, 14, 16, 18, 20\}$.[9,10,12,13,31] We also include a classification and segmentation MTL approach for comparison although it was originally used for biomedical images.[42] Similar to (13), the average accuracy on the test set $D^{(\text{Test})}$, $\text{Acc}_{\text{Avg}}^{(\text{Test})}$, is also provided in Table 3–5.

On the BDIP dataset, as shown in Table 3, PCBMTL-2 F, PCBMTL-3 F, and PCBMTL-4 F achieve the highest average accuracies of 85.31%, 85.92%, and 85.14%, respectively, which outperform state-of-the-art approaches of accuracies ranging from 30.64% to 72.37% by large margins. At $p = 10\%$, the proposed PCBMTL achieves accuracies of 75.19% to 78.59%, which already

Table 3. Comparisons of the proposed PCBMTL with state-of-the-art approaches on the BDIP test set.

Approaches	$\text{Acc}\left(D^{(\text{Test})}|D_p^{(\text{Train})}\right)$ [%]						$\text{Acc}_{\text{Avg}}^{(\text{Test})}$ [%]	$R_{\text{Avg}}^{(\text{Test})}$ [%]
	$p = 10$	$p = 12$	$p = 14$	$p = 16$	$p = 18$	$p = 20$		
PCBMTL-2F	75.19	82.74	86.61	88.06	89.78	89.50	85.31	84.89
PCBMTL-3F	**78.59**	**83.31**	**87.55**	**87.44**	88.98	**89.67**	**85.92**	**85.71**
PCBMTL-4F	75.62	82.99	86.43	86.98	**89.82**	89.00	85.14	84.92
Gao's[9]	53.15	60.44	62.32	62.53	64.23	64.85	61.26	61.08
Ma's[10]	64.08	70.58	73.06	72.97	76.64	76.86	72.37	72.28
Metzner's[12]	61.66	63.31	61.85	64.26	64.62	67.08	63.80	64.07
Zhang's[13]	61.87	68.15	66.53	70.94	69.14	72.60	68.20	68.10
ResNet-50[31]	60.18	65.10	67.10	70.03	71.20	72.27	67.65	68.05
Ciga's[42]	27.91	31.69	31.41	28.04	29.31	35.50	30.64	30.25

Table 4. Comparisons of the proposed PCBMTL with state-of-the-art approaches on the FDIP test set.

Approaches	$\text{Acc}\left(D^{(\text{Test})}|D_p^{(\text{Train})}\right)$ [%]						$\text{Acc}_{\text{Avg}}^{(\text{Test})}$ [%]	$R_{\text{Avg}}^{(\text{Test})}$ [%]
	$p = 10$	$p = 12$	$p = 14$	$p = 16$	$p = 18$	$p = 20$		
PCBMTL-2F	**95.40**	96.37	97.26	97.96	97.77	98.36	97.19	97.20
PCBMTL-3F	95.11	96.80	97.34	98.04	98.15	**98.63**	97.34	97.36
PCBMTL-4F	94.44	**97.10**	**97.98**	**98.41**	**98.49**	98.23	**97.44**	**97.46**
Gao's[9]	75.16	77.55	79.81	80.89	82.42	82.10	79.66	79.53
Ma's[10]	71.45	72.90	75.75	77.93	81.08	83.25	77.06	76.89
Metzner's[12]	68.20	76.91	78.28	81.26	87.45	87.18	79.88	79.73
Zhang's[13]	89.87	91.16	91.94	91.61	92.90	94.09	91.93	91.92
ResNet-50[31]	67.10	70.78	77.82	80.48	83.47	84.46	77.35	79.33
Ciga's[42]	46.02	50.51	45.97	54.78	65.27	51.69	52.37	52.31

ADVANCED
SCIENCE NEWS

www.advancedsciencenews.com

ADVANCED
INTELLIGENT
SYSTEMS

www.advintellsyst.com

Table 5. Comparisons of the proposed PCBMTL with state-of-the-art approaches on the FFC test set.

Approaches	$\mathrm{Acc}\left(D^{(\mathrm{Test})}\|D_p^{(\mathrm{Train})}\right)$ [%]						$\mathrm{Acc}_{\mathrm{Avg}}^{(\mathrm{Test})}$ [%]	$R_{\mathrm{Avg}}^{(\mathrm{Test})}$ [%]
	$p=10$	$p=12$	$p=14$	$p=16$	$p=18$	$p=20$		
PCBMTL-2F	<u>83.49</u>	**84.27**	**83.90**	**87.67**	<u>89.26</u>	**90.17**	**86.46**	**86.23**
PCBMTL-3F	**83.91**	<u>83.58</u>	<u>82.41</u>	<u>86.05</u>	**90.37**	<u>89.93</u>	<u>86.04</u>	<u>85.79</u>
PCBMTL-4F	81.53	81.97	80.20	84.73	88.80	88.46	84.28	83.93
Gao's[9]	59.86	65.60	65.00	66.59	67.15	66.73	65.15	65.33
Ma's[10]	66.02	66.62	62.95	72.37	70.47	74.90	68.89	68.93
Metzner's[12]	60.12	66.31	64.42	61.25	71.51	71.81	65.90	66.09
Zhang's[13]	62.30	65.13	72.26	62.72	70.11	71.66	67.37	67.54
ResNet-50[31]	60.47	60.23	64.21	67.92	71.42	73.26	66.25	66.04
Ciga's[42]	25.76	24.18	28.04	37.41	39.42	35.82	31.73	31.85

outperforms state-of-the-art approaches of accuracies ranging from 35.50% to 72.60%, and are on par with Ma's accuracy of 76.86%, at $p=20$. This verifies the effectiveness of MTL, where the segmentation task helps improve classification accuracy.

On the FDIP dataset, as shown in Table 4, PCBMTL-2F, PCBMTL-3F, and PCBMTL-4F achieve the highest average accuracies of 97.19 %, 97.34%, and 97.44% respectively, with significant improvement over other approaches of accuracies ranging from 52.37% to 91.93%. At $p=10$, the proposed PCBMTL achieves accuracies of 94.44% to 95.11%, which outperforms approaches of accuracies ranging from 51.69% to 94.09%, at $p=20$.

Regarding FFC, as shown in Table 5, PCBMTL-2F, PCBMTL-3F, and PCBMTL-4F achieve the highest average accuracies of 86.46%, 86.04%, and 84.28%, respectively, which outperforms other approaches ranging from 31.73% to 68.89%, with significant margins. Similar to the trend in FDIP, at $p=10$, the proposed PCBMTL obtains accuracies of 81.53% to 83.91%, respectively, which already outperforms with accuracies ranging from 35.82% to 74.90% at $p=20$. The large improvement demonstrates the superiority of the proposed PCBMTL model.

In addition, the average sensitivity or equivalently average recall rate, $R_{\mathrm{Avg}}^{(\mathrm{Test})}$, of each method was also measured. On BDIP, as shown in Table 3, our PCBMTL-2F, PCBMTL-3F, and PCBMTL-4F achieve the highest average recall rates of 84.98%, 85.71%, and 84.92%, respectively, which outperform state-of-the-art approaches of average recall rates ranging from 30.25% to 72.28% by large margins. Similar trends are observed for the FDIP and FFC datasets. On FDIP, as shown in Table 4, our PCBMTL-2F, PCBMTL-3F, and PCBMTL-4F achieve the highest average recall rates of 97.20%, 97.36%, and 97.46%, respectively, which outperform state-of-the-art approaches of average recall rates ranging from 52.31% to 91.92%. On FFC, as shown in Table 5, our PCBMTL-2F, PCBMTL-3F, and PCBMTL-4F achieve the highest average recall rates of 86.23%, 85.79%, and 83.93%, respectively, which outperform state-of-the-art approaches of average recall rates ranging from 31.85% to 68.93%. As our dataset is a balanced dataset, the average recall rate is close to the accuracy. In cases where the dataset is imbalanced, the recall rate is often more informative and relevant than accuracy.

In summary, PCBMTL-2F, PCBMTL-3F, and PCBMTL-4F consistently outperform other approaches on all three datasets for all percentages of training samples. When the number of training samples is only 10% of the original dataset, the proposed PCBMTL can still achieve 80% or higher accuracy, which significantly outperforms the conventional approaches. Overall, PCBMTL is a data-efficient PCB soldering defect detection model that is particularly useful under low data regimes when it is challenging or expensive to collect the required training data.

4.3. Visualizations of Soldering Region Segmentation

Even though the segmentation task is not necessary during inference, we include a visualization of the segmentation results on the test set for a better understanding of what the PCBMTL-3F network has learned. The visualizations, as shown in **Figure 4**, are generated using the models trained with 10%, 16%, and 20% of the training data.

In the case of BDIP, we can observe from Figure 4a that for the normal class, the proposed PCBMTL-3 F can achieve a good segmentation of the soldering region even when the training sample percentage is 10%. As we increase the training sample percentage to 20%, the segmentation becomes much closer to the ground truth. In the case of FDIP, as shown in Figure 4f, the proposed model successfully segments the solder without including the DIP pins. Even though a rough ground-truth segmentation mask is manually annotated in Figure 4h, the proposed model can still accurately segment the solder without misjudging the pins as solder. Similarly, in FFC, for example, in Figure 4j, the proposed model accurately differentiates between silkscreen and soldering areas.

The visualization of segmentation results provides insights into the model's learning and helps identify areas where the model performs well or poorly. By leveraging the segmentation results, we can be confident that the proposed model accurately classifies the soldering defects.

4.4. Further Enhancement with Pretraining

ImageNet pretraining is a popular technique for initializing the weights of a model using the massive ImageNet dataset,
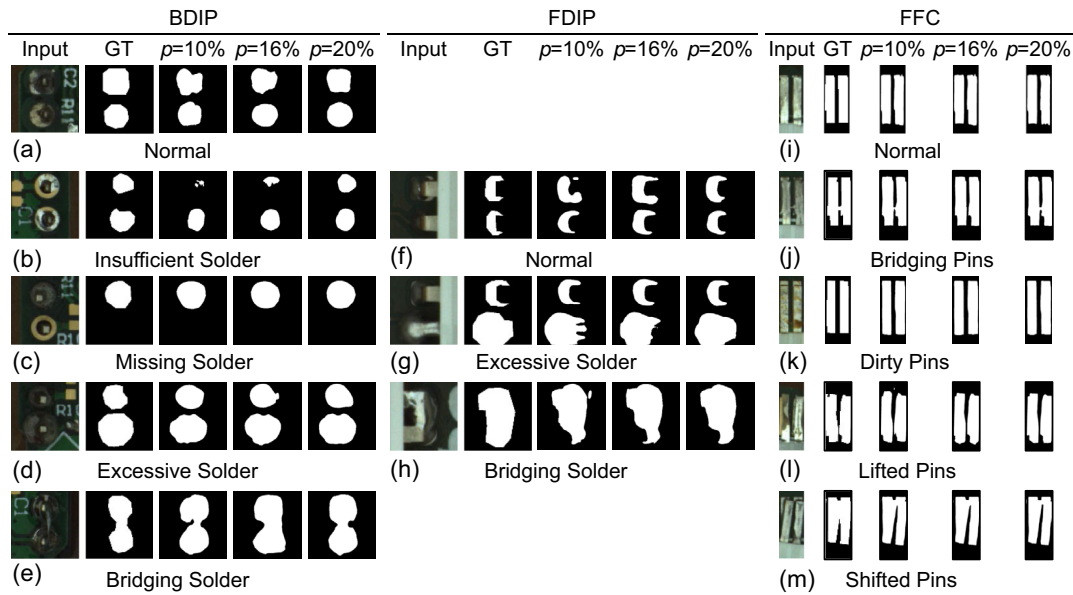
Figure 4. Visualization of soldering segmentation results using PCBMTL-3F trained under different percentages of training samples: a–e) BDIP, f–h) FDIP, and i–m) FFC. (GT: Ground-Truth).

which consists of approximately 1.2 million training images.[33] The pretrained model is subsequently fine-tuned for a specific downstream task. It can be a way to help in the low-data regime since the task-related training data can be reduced. In this section, we investigate the effectiveness of our proposed MTL approach for ImageNet pretrained models on our PCBSPDefect dataset. We first pretrained the encoder and the classification branch of the proposed PCBMIL on ImageNet using (11). Then, during the fine-tuning phase, the entire model was further trained on our BDIP dataset using MTL via (12). The results are presented from **Table 6–8**.

On BDIP (Table 6), our model pretrained on ImageNet without MTL (referred to as PT) achieves an average accuracy of 82.87%.

However, when we utilize both pretraining and MTL, our models, PT + PCBMTL-2F, PT + PCBMTL-3F, and PT + PCBMTL-4F, achieve significantly higher average accuracies of 86.90%, 87.02%, and 86.90%, respectively, with a margin of about 4%. This suggests that although ImageNet pretraining can help the model learn rich image representations, our proposed MTL approach can further improve the classification performance.

On FDIP (Table 7), our model pretrained on ImageNet without MTL (referred to as PT) already achieves an average accuracy of 97.96% since this is a relatively easier task. Yet, similar to the BDIP case, when we utilize both pretraining and MTL, our models, PT + PCBMTL-2F, PT + PCBMTL-3F, and PT + PCBMTL-4F, achieve significantly higher average accuracies of 99.26%,

Table 6. Effectiveness of the proposed PCBMTL approach for ImageNet pretrained (PT) models on the BDIP test set.

Approaches	MTL	$\text{Acc}\left(D^{(Test)}\vert D_p^{(Train)}\right)$ [%]						$\text{Acc}_{Avg}^{(Test)}$ [%]
		$p = 10$	$p = 12$	$p = 14$	$p = 16$	$p = 18$	$p = 20$	
PT (w/o MTL)		72.39	81.45	82.92	84.99	86.90	88.55	82.87
PT + PCBMTL-2F	✓	81.47	84.37	87.66	87.14	**90.21**	**90.57**	86.90
PT + PCBMTL-3F	✓	81.23	**84.51**	**88.17**	**88.22**	89.97	90.00	**87.02**
PT + PCBMTL-4F	✓	**82.17**	84.38	86.13	87.22	88.12	89.99	86.33

Table 7. Effectiveness of our proposed PCBMTL approach for ImageNet pretrained (PT) models on the FDIP test set.

Approaches	MTL	$\text{Acc}\left(D^{(Test)}\vert D_p^{(Train)}\right)$ [%]						$\text{Acc}_{Ag}^{(Test)}$ [%]
		$p = 10$	$p = 12$	$p = 14$	$p = 16$	$p = 18$	$p = 20$	
PT (w/o MTL)		96.64	97.69	97.39	98.31	98.79	98.92	97.96
PT + PCBMTL-2F	✓	**98.36**	**99.17**	99.44	**99.70**	99.30	99.60	99.26
PT + PCBMTL-3F	✓	**98.36**	99.09	**99.46**	99.60	**99.57**	**99.76**	**99.31**
PT + PCBMTL-4F	✓	98.06	98.71	98.90	99.62	**99.57**	99.70	99.09

Table 8. Effectiveness of our proposed PCBMTL approach for ImageNet pretrained (PT) models on the FFC test set.

Approaches	MTL	$\text{Acc}\left(D^{(\text{Test})}\vert D_p^{(\text{Train})}\right)$ [%]						$\text{Acc}_{\text{Avg}}^{(\text{Test})}$ [%]
		$p=10$	$p=12$	$p=14$	$p=16$	$p=18$	$p=20$	
PT (w/o MTL)		87.80	89.41	90.19	91.08	92.39	93.62	90.74
PT + PCBMTL-2F	✓	90.40	91.38	91.77	**93.60**	93.91	95.18	92.71
PT + PCBMTL-3F	✓	**90.99**	**91.48**	91.57	92.48	**94.93**	**95.41**	**92.81**
PT + PCBMTL-4F	✓	90.05	91.09	**92.01**	93.03	94.36	94.97	92.59

99.31%, and 99.09%, respectively, with a margin of over 1%. This again shows that our proposed MTL approach can further improve the classification performance even when pretraining is utilized.

On FFC (Table 8), our model pretrained on ImageNet without MTL (referred to as PT) achieves an average accuracy of 90.74%. Likewise, our models, PT + PCBMTL-2F, PT + PCBMTL-3F, and PT + PCBMTL-4F, achieve significantly higher average accuracies of 92.71%, 92.81%, and 92.59%, respectively, with a margin of about 2%.

All these results are way better than the traditional PCB soldering defect detection methods, as shown in Table 3–5.

5. Conclusion

In PCB assembly, identifying soldering defects is crucial for enhancing manufacturing reliability. AOI using deep learning is a promising solution for this task. Yet, due to many practical reasons, there is a lack of training samples, which poses a challenge for training deep learning PCB soldering defect detection models under low-data regimes. To address this, we have proposed a novel MTL deep learning model PCBMTL in this article, which simultaneously learns the segmentation and classification tasks. With the learned segmentation knowledge, the classification performance is improved even when only a small amount of training samples are available. To facilitate MTL, we also built the PCBSPDefect dataset, which covers three components: BDIP, FDIP, and FFC, with corresponding segmentation masks. Experimental results show that the proposed PCBMTL achieves the highest accuracies of 85.92%, 97.44%, and 86.46% when testing on the BDIP, FDIP, and FFC test sets, respectively, outperforming the best prior arts by over 13%, 5%, and 17%, respectively. Besides, further improvements are noted when the models are pretrained on ImageNet. To the best of our knowledge, this is the first work to provide a classification and segmentation soldering defect dataset, as well as apply MTL to PCB soldering AOI. We believe that the proposed framework can improve the performance of soldering defect inspection while reducing the need for large datasets.

Acknowledgements

Conflict of Interest

The authors declare no conflict of interest.

Data Availability Statement

The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

Keywords

[1] M. R. Driels, D. J. Nolan, *IEEE Trans. Compon., Hybrids, Manuf.* **1990**, *13*, 331.

[2] Y. Matsuyama, T. Honda, H. Yamamura, H. Sasazawa, M. Nomoto, T. Ninomiya, A. Schick, L. Listl, P. Kollensperger, D. Spriegel, P. Mengel, R. Schneider, in *IEEE Workshop on Applications of Computer Vision (WACV)*, IEEE, Sarasota, FL **1996**, pp. 116–122.

[3] C. L. S. C. Fonseka, J. A. K. S. Jayasinghe, *IEEE Trans. Compon. Packag. Manuf.* **2019**, *9*, 353.

[4] K. Zhang, T. Huang, Z. Su, T. Guan, in *IEEE Advanced Information Technology, Electronic and Automation Control Conf. (IAEAC)*, IEEE, Chongqing, China **2021**, pp. 2413–2418.

[5] X. Lu, Z. He, L. Su, M. Fan, F. Liu, G. Liao, T. Shi, *IEEE Trans. Ind. Inf.* **2018**, *14*, 5620.

[6] G. Acciani, G. Brunetti, G. Fornarelli, *IEEE Trans. Ind. Inf.* **2006**, *2*, 200.

[7] M. Liukkonen, T. Hiltunen, E. Havia, H. Leinonen, Y. Hiltunen, *IEEE Trans. Electron. Packag. Manuf.* **2009**, *32*, 89.

[8] Y. LeCun, Y. Bengio, G. Hinton, *Nature* **2015**, *521*, 436.

[9] S. Gao, H. Zhang, H. Mi, in *IEEE Int. Conf. Signal, Information and Data Processing (ICSIDP)*, IEEE, Chongqing, China **2019**, pp. 1–6.

[10] D. Ma, X. Lei, H. Zhao, in *Int. Conf. Artificial Intelligence and Advanced Manufacturing (AIAM)*, Dublin, Ireland **2019**, pp. 598–603.

[11] K. Simonyan, A. Zisserman, in *Int. Conf. Learning Representations (ICLR)*, San Diego, CA **2015**, pp. 1–14.

[12] M. Metzner, D. Fiebag, A. Mayr, J. Franke, in *Int. Electric Drives Production Conf. (EDPC)*, Esslingen, Germany **2019**, pp. 1–6.

[13] Q. Zhang, M. Zhang, C. Gamanayake, C. Yuen, Z. Geng, H. Jayasekara, C.-W. Woo, J. Low, X. Liu, Y. L. Guan, *J. Complex Intell. Syst.* **2022**, *8*, 1525.

[14] O. Ronneberger, P. Fischer, T. Brox, in *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, Munich, Germany **2015**, pp. 232–241.

[15] G. Mahalingam, K. M. Gay, K. Ricanek, in *Int. Conf. Machine Vision Applications (MVA)*, Tokyo, Japan **2019**, pp. 1–5.

[16] H. Lu, D. Mehta, O. Paradis, N. Asadizanjani, M. Tehranipoor, D. L. Woodard, *Cryptology ePrint Archive* **2020**, paper no. 2020/366, https://eprint.iacr.org/2020/366.

[17] M. Ojer, I. Serrano, F. Saiz, I. Barandiaran, I. Gil, D. Aguinaga, D. Alejandro, *Int. J. Adv. Manuf. Technol.* **2020**, *107*, 2261.

[18] M. A. Reza, Z. Chen, D. J. Crandall, *J. Hardware Syst. Secur.* **2020**, *4*, 44.

[19] Y. Fridman, M. Rusanovsky, G. Oren, in *IEEE Physical Assurance and Inspection of Electronics (PAINE), Virtual*, IEEE, Piscataway, NJ **2021**, pp. 1–8.

[20] S. Tang, F. He, X. Huang, J. Yang, *arXiv Preprint*, arXiv:1902.06197v1, **2019**.

[21] W. Huang, P. Wei, M. Zhang, H. Liu, *J. Eng.* **2020**, *2020*, 303.

[22] N. Choudhary, H. Clever, R. Ludwigs, M. Rath, A. Gannouni, A. Schmetz, T. Hülsmann, J. Sawodny, L. Fischer, A. Kampker, J. Fleischer, H. S. Stein, *Adv. Intell. Syst.* **2022**, *4*, 2200142.

[23] J. Wang, J. Yang, G. Lu, C. Zhang, Z. Yu, Y. Yang, *Adv. Intell. Syst.* **2023**, *5*, 2200151.

[24] S.-H. Tsang, Z. Suo, T. T.-L. Chan, H.-T. Nguyen, D. P.-K. Lun, https://github.com/cairs-project-5/PCBSPDefect (accessed: August 2023).

[25] K. Khalil, A. Kumar, M. Bayoumi, *IEEE Trans. Circuits Syst. II: Exp. Briefs* **2022**, *69*, 5094.

[26] K. Khalil, O. Eldash, A. Kumar, M. Bayoumi, in *IEEE Int. Conf. Electronics, Circuits and Systems (ICECS)*, IEEE, Bordeaux, France **2018**, pp. 745–748.

[27] Y. Liu, F. Wang, K. Liu, M. Mostacci, Y. Yao, S. Sfarra, *Quant. Infrared Thermogr. J.* **2023**, https://doi.org/10.1080/17686733.2023.2225246.

[28] K. Liu, M. Zheng, Y. Liu, J. Yang, Y. Yao, *IEEE Trans. Ind. Inf.* **2023**, *19*, 6429.

[29] Y. Ge, Y. Zhou, J. Yang, Y. Zhang, F. Xie, X. Tang, in *Int. Conf. Robotics, Automation, Intelligent Control (ICRAIC)*, Wuhan, China **2021**, pp. 1–8.

[30] A. Caliskan, G. Gurkan, in *Int. Conf. Innovations in Intelligent Systems and Applications (INISTA)*, Kocaeli, Turkey **2021**, pp. 1–6.

[31] K. He, X. Zhang, S. Ren, J. Sun, in *IEEE Computer Vision and Pattern Recognition (CVPR)*, IEEE, Las Vegas, NV **2016**, pp. 770–778.

[32] T. Chen, S. Kornblith, M. Norouzi, G. E. Hinton, in *Int. Conf. Machine Learning (ICML)*, Vienna, Austria **2020**, pp. 1597–1607.

[33] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, F.-F. Li, in *IEEE Computer Vision and Pattern Recognition (CVPR)*, IEEE, Miami, FL **2009**, pp. 248–255.

[34] G. Ghiasi, Y. Cui, A. Srinivas, R. Qian, T.-Y. Lin, E. D. Cubuk, Q. V. Le, B. Zoph, in *IEEE/CVF Computer Vision and Pattern Recognition (CVPR)*, IEEE, Nashville, TN **2021**, pp. 2917–2927.

[35] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, in *Int. Conf. Neural Information Processing Systems (NIPS)*, Cambridge, MA **2014**, pp. 2672–2680.

[36] S. Ioffe, C. Szegedy, in *Int. Conf. Machine Learning (ICML)*, Lille, France **2015**, pp. 448–456.

[37] V. Nair, G. E. Hinton, in *Int. Conf. Machine Learning (ICML)*, Haifa, Israel **2010**, pp. 807–814.

[38] K. Khalil, O. Eldash, A. Kumar, M. Bayoumi, *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2022**, *30*, 303.

[39] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, *J. Mach. Learn. Res.* **2014**, *15*, 1929.

[40] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, in *Neural Information Processing Systems (NeurIPS)*, Vancouver, Canada **2019**, pp. 1–12.

[41] D. P. Kingma, J. Ba, in *Int. Conf. Learning Representations (ICLR)*, San Diego, CA **2015**, pp. 1–15.

[42] O. Ciga, A. L. Martel, *J. Med. Image Anal.* **2021**, *68*, 101912.