



OPEN

Attentive context and semantic enhancement mechanism for printed circuit board defect detection with two-stage and multi-stage object detectors

Twahir Kiobya¹, Junfeng Zhou¹✉, Baraka Maiseli² & Maqbool Khan^{3,4}

Printed Circuit Boards (PCBs) are key devices for the modern-day electronic technologies. During the production of these boards, defects may occur. Several methods have been proposed to detect PCB defects. However, detecting significantly smaller and visually unrecognizable defects has been a long-standing challenge. The existing two-stage and multi-stage object detectors that use only one layer of the backbone, such as Resnet's third layer (C_4) or fourth layer (C_5), suffer from low accuracy, and those that use multi-layer feature maps extractors, such as Feature Pyramid Network (FPN), incur higher computational cost. Faced by these challenges, we propose a robust, less computationally intensive, and plug-and-play Attentive Context and Semantic Enhancement Module (ACASEM) for two-stage and multi-stage detectors to enhance PCB defects detection. This module consists of two main parts, namely adaptable feature fusion and attention sub-modules. The proposed model, ACASEM, takes in feature maps from different layers of the backbone and fuses them in a way that enriches the resulting feature maps with more context and semantic information. We test our module with state-of-the-art two-stage object detectors, Faster R-CNN and Double-Head R-CNN, and with multi-stage Cascade R-CNN detector on DeepPCB and Augmented PCB Defect datasets. Empirical results demonstrate improvement in the accuracy of defect detection.

Keywords Semantic information, Context information, Squeeze and excitation, Feature fusion

Printed Circuit Boards (PCBs) are boards where electronic components such as transistors, resistors, capacitors, and Integrated Circuits are mounted to create electronic devices. They have copper painted paths to enable electrical connectivity between components. One advantage of printed circuit boards is the easiness of assembling different electronic components to form electronic systems¹.

During production of PCBs, defects in some of the produced boards may occur. Several types of defects are commonly identified with PCBs, such as, short circuit, mousebite, missing hole, spurious copper and open circuit².

The main industrial method for PCB defects detection is the Automated Optical Inspection (AOI) method. This is a type of visual inspection where a camera zooms through boards to see if there are any defects³. Moganti et al.⁴ categorized AOI into three major groups, namely; Referential, Non-referential, and Hybrid AOI types. In the referential inspection type, PCB images are compared with a defect-free reference image. The difference between the two images can help localize the error. A good example of this inspection was discussed in the work by Ibrahim and Al-Attas⁵.

Non referential AOI utilizes pre-defined rules to detect defects. However, the advent of deep learning has rendered this method not widely applicable as was the case in the past. Deep learning methods such as Single Shot Multibox Detector (SSD)⁶, FPN⁷, and Faster R-CNN⁸ offer better results than the rule-based methods.

¹School of Computer Science and Technology, Donghua University, Shanghai 201620, People's Republic of China. ²College of Information and Communication Technologies, University of Dar es Salaam, P. O. Box 33335, Dar es Salaam, Tanzania. ³Pak-Austria Fachhochschule-Institute of Applied Sciences and Technology, Mang, Haripur 22621, Pakistan. ⁴Software Competence Center Hagenberg GmbH, Softwarepark, Hagenberg, 4232 Linz, Austria. ✉email: zhoujf@dhu.edu.cn

Hybrid approaches for AOI defects inspection combine referential and non-referential techniques. Examples of methods under this category were proposed by Radev and Shirvaikal⁹ and Fang et al.¹⁰. These methods however, have some shortcomings such as high costs due to the fact that they combine multiple hardware or software, also due to their high complexities, some problems related to performance may occur, such as low processing speed compared to other methods. The main reason for this is because they have to gather information from multiple sources before giving an answer.

Even though the existing deep learning methods have achieved great success in the detection of PCB defects there are shortcomings which need to be addressed. For example, most of the two stage PCB defects detectors have their success primarily due to parameter tuning. This affects the consistency and predictability in new parameters settings. Our plug and play ACASEM module has shown a consistency in improving the accuracy of PCB defects detection across a wider range of parameters initialization.

As we have pointed out, current improvements of the two-stage and multi-stage detectors to the task of PCB defects detection owes their success to the adoption of FPN as a strategy for enriching feature maps with semantic information. However, FPN adds more computational overhead as it contributes many parameters arising from several 1×1 convolutions in the lateral connections, and the 3×3 convolutions of the smooth layer connections. In addition, most of the current works lack a comprehensive performance evaluation and analysis across a wider range of evaluation metrics. These works evaluate the detection accuracy on only one precision metric at Intersection over union (IoU) of 0.5 but lack insights on the recall metrics across defects spatial dimensions. Motivated with an urge to fill this gap, our work evaluates detection accuracy using average precision metric based on precision values calculated across 10 *IoU* thresholds (from 0.5 up to 0.95 with the increment of 0.05). It also evaluates the detection accuracy using average precision and average recall metrics based on objects' spatial scales including small and medium sizes.

In this paper, we propose the Attentive Context and Semantic Enhancement Module (ACASEM) to enhance detection accuracy of PCB defects. This module enriches the feature map going to the Regional Proposal Network (RPN)⁸ and the Region of Interest (RoI) subnetwork with a feature map rich in the semantic and contextual information. We apply this module to the two-stages Faster R-CNN and Double-Head R-CNN¹¹ and a multi-stage Cascade R-CNN¹² state-of-the-art methods to test and validate its efficiency. Results show that, incorporating the Attentive Context and Semantic Enhancement Module improves detection accuracy of small and medium-sized defects in terms of both Average Precision and Average Recall.

The contributions of this paper are two folds:

1. Firstly, we propose a novel Attentive Context and Semantic Enhancement Module (ACASEM), which harnesses the power of semantic and contextual information from different layers of the backbone and fuse them. The attention mechanism in the module helps to rank the resulting feature maps channels based on their importance, thus assisting the model to pick the most relevant ones.
2. Secondly, we provide a comprehensive performance analysis on the PCB defects detection using a wide range of evaluation metrics. Currently, most of the existing works only offer an analysis on one evaluation metric, the mean Average Precision at the *IoU* = 0.5. This makes it harder to understand how these models perform on defects detection based on other metrics including defects spatial sizes, namely; small or medium size. Our work offers an analysis using more evaluation metrics to provide a wider perspective on the performance of models. We use the mean Average Precision (*mAP*) at *IoU* (starting from 0.5 up to 0.95). We also use the mean Average Recall (*mAR*), Average Precision (*AP*) and Average Recall (*AR*) at various spatial sizes of defects.

Related works

General PCB defect detection

Defects from PCB surfaces (Fig. 1) can broadly be put into five major categories: statistical, spectral, model-based, combination of methods, and artificial intelligence (AI) based. These categories have received wide attention within the scholarly community, and there have been significant developments in each category:

Statistical techniques explore the variability in pixels distribution to identify patterns that can lead to the isolation and identification of defects^{13,14}. The statistical technique organizes deviant pixels (from a PCB image) into a defective region, separating them from the normal pixels.

Spectral techniques utilize special transforms, such as Fourier transform, to convert time-varying signals into the frequency domain¹⁵. For example, Da et al.¹⁶ used a guided ultrasonic wave and a quantitative detection with Fourier transform to detect defects in pipes. Chan and Pang¹⁷ used Fourier transform to detect fabric defects. The authors used a central spatial frequency to reduce defect detection computational time and to provide more parameters needed for classification.

Model-based techniques simulate a process that generates specific image features, such as textures. For example, Baykut et al.¹⁸ used Markov random fields to detect faults in digital signal processing systems. Their technique offers a real-time property for detecting faults, and hence it may be applicable in time-sensitive applications, especially those from automation and monitoring industries.

The AI category¹⁹, which this work focuses on, provides deep learning techniques for defects detection. Ran et al.²⁰ used SSD algorithm to detect defects in PCB, giving an accuracy of 94.69%. The authors used a dataset with 16,000 PCB training images and 1600 test images having four types of defects: copper exposure, nodules, scratch, and coating.

Combination of methods techniques involve the process of investigating and integrating different competing methods by harnessing their potential benefits and de-emphasizing their limitations. Li et al.²¹ used polarization information and infrared imaging for PCB defects detection. Their work considers the fact that PCB images are

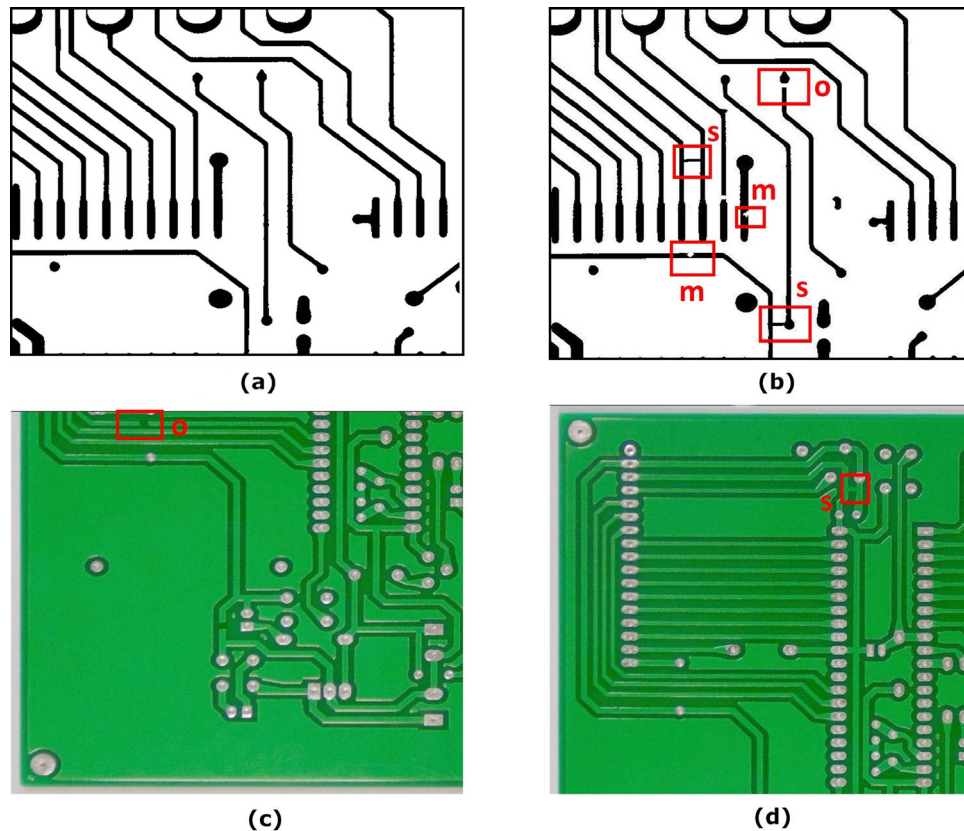


Figure 1. (a) PCB without defects (b) PCB with defects from DeepPCB dataset. (c) and (d) are PCBs with defects from an Augmented PCB Defect dataset. Letter “o” stands for open circuit, “s” is spur and “m” is mousebite defects.

taken at different brightness levels that may affect the detection accuracy. Mitigating this problem, they used thermal infrared imaging that was insensitive against background contrast and illumination. Kang et al.²² proposed a Label Robust and Patch Correlation Enhanced Vision Transformer (LPViT) for detecting PCB defects. In their technique, images are initially sliced into different chunks and randomly masked to improve inter-regional understanding. Then, a label smooth approach is applied to robustify the technique for working with different imaging conditions. The work by Kang et al. achieves 99% F1-Score on the Micro-PCB Dataset.

Feature fusion and attention

Feature Fusion is a prevalent method in deep learning. It enables networks to train with many features of different size and type from different parts of the network. Through the variations introduced by these features networks become more robust and generalize better to the unseen data. In deep learning strategies for feature fusion can be put into three categories namely; early feature fusion, hierarchical feature fusion, attention based feature fusion and graph based feature fusion.

Early feature fusion encompasses two subcategories namely; direct feature fusion and weighted linear combination. Direct fusion involves basic operations such as addition, and concatenation and is characterized by the absence of connections until the fusion stage. Examples of the deep learning models using direct fusion are Fast-SCNN²³ and FasterSeg²⁴ whereas the good examples of models employing the weighted linear combination feature fusion strategy are the bilateral segmentation network (BiSeNet)²⁵ and DeepLabV3²⁶.

Hierarchical feature fusion strategies involves the combination of features from different scales or layers of the neural network in an organized hierarchical style. Examples of methods employing this strategy are the networks such as FPN and fully convolutional network (FCN)²⁷. Also the widely used Resnet based backbones such as Resnet50 and Resnet101 also utilize this fusion.

Attention based feature fusion strategy employ the attention technique to select the best features for fusion. Attention focuses on the most relevant parts of the feature maps thus features from these parts are subsequently used for the intended deep learning activity such as object detection or segmentation. Some of the widely used attention methods are; channel attention, spatial attention and channel-spatial attention.

Channel attention prioritizes channels with the most relevant information while suppressing those with less relevant information. Examples of works utilizing this method are the squeeze-and-excitation networks (SENet), efficient channel attention network (ECA-Net)²⁸ and style-based recalibration module (SRM)²⁹.

Spatial attention focuses on certain areas of the image. It involves feature extraction by convolutional layers followed by the generation of attention maps which assign higher weights to more relevant regions. Examples of works employing this method are the self-attention based methods including visual transformers such as ViT³⁰ and DETR³¹.

Channel-spatial attention leverages the strength of both the channel and spatial attention. It involves the selection of the most relevant channels and regions. Examples of methods in this category are the convolutional attention block module (CBAM)³² and the bottleneck attention module (BAM)³³.

Feature fusion and attention based deep learning methods for PCB defect detection

Leveraging the power of fusion and attention strategies, several works have been proposed to utilize those strategies to enhance the PCB defect detection. Hu et al.³⁴ adopted the hierarchical multi-level and multi-scale fusion with RPN to enhance detection. GARN was used to improve the anchor prediction accuracy whereas ShuffleNet blocks were adopted in the residual parts of the network to cut down the computational cost. The model was trained on the custom dataset obtained from a LED electronics factory in China. The model obtained 94.2% in *mAP*. The strength of this work is that it obtained richer features by combining features of different spatial sizes and the accuracy was improved. The weakness of this work is that it achieved a poor detection accuracy on the “open” defect type and the work only offers an overall detection accuracy, it does not evaluate on a wider range of metrics including spatial sizes such as small and small size, hence it is difficult to know how the model perform on defects of those sizes.

Li et al.³⁵ introduced the SE attention block in the Resnet101 backbone and incorporated the bottom-up path in the FPN to preserve the information from the shallow layers. They utilized a custom dataset spawned by the AOI device on the PCB manufacturing line with images having a resolution of 985×825 . Their work achieved the *mAP* of 96.3%. Even though the work achieved improvements in the detection accuracy however it had some limitations including training to recognise only few defects, and also only one dataset was used in the entire study. Hence the generalizability of the work with a new dataset is unclear.

Chen et al.³⁶ altered the original clustering algorithm to suit the PCB defect detection. They adopted the Swin transformer as a backbone and introduced attention via the SE block that was inserted between the feature extraction and detection networks. YOLO local features and the and the global transformer features are fused together to capture more context. The work achieved a good result but only one dataset was used in the study hence the robustness of the model to different scenarios is ambiguous.

Xin et al.³⁷ approached the PCB defect detection by improving the YOLOv4 model. They replaced the Leaky-ReLU activation function in the backbone and enhanced the probability of an anchor to contain an object by automatically subdividing the image based on the bounding box average size in the image. The work exhibited improvement in the detection accuracy however its drawbacks are only one dataset was used for training hence the effectiveness of the work across different datasets is unsubstantiated.

Jiang et al.³⁸ improved the PCB defect detection using SSD by adding the coordiante attention module. The CA captures the information of features that are impactful and simultaneously focuses on the location of those features. The module was attached behind the backbone layers Conv4_3, FC7 and Conv8_2. This way shallow layers are enhanced to detect defects. The work obtained promising results however its performance was not compared with with sufficient advanced methods to ascertain its merits. Also the work utilized only one dataset thus it unclear regarding the works’s capabilities to extend to broader scenarios. Furthermore the evaluation was done with only one IoU scale (IoU = 0.5) hence it is not clear how the work generalizes across different IoU scales and spatial dimensions such as small and medium sizes.

Li et al.³⁸ introduced the ERF based anchor allocation method to enhance the classification and localization of the PCB defects. The ERF mitigated the anchor frame size issues while the atrous spatial pooling method and the channel attention module enhanced the hard-to-detect and small defects. Their work improved the detection accuracy but had a limitation of using only one dataset hence its effectiveness across new data is not well established.

Zhang et al.³⁹ proposed an improved YOLOv5 algorithm. They introduced a self attention mechanism in the yolohead to facilitate the learning of the global information. Experiments were conducted on the Peking university PCB defect dataset. The detection accuracy of their work surpassed the original YOLOv5 by 4.68% in AP. The pros of this work is that the detection accuracy was enhanced. The drawbacks is that computational complexity in terms of the number of parameters increased by 5M compared to the original YOLOv5.

Gao et al.⁴⁰ proposed a module, feature collection and compression network, to enhance PCB defects detection. This module serves as an assistive block to merge multi-scale feature information. Gao et al., in addition, introduced a novel Gaussian pooling method as an alternative to the RoI pooling. Their work achieved *mAP* of 74.4% on the DeepPCB dataset.

Kamapreet et al.⁴¹ proposed an autoencoder-based skip-connected convolutional neural network. Both encoder and decoder were skip-connected, and feature maps were element-wise summed, and the non-defect images were decoded from the original images. This strategy helps to recover images more efficiently and effectively.

Tang et al.⁴² proposed a model to deal with hyperparameter over-sensitivity in the detection of PCB defects. They applied a novel group pyramid model to extract large-range resolution feature maps, which were eventually merged to predict defects at different scales.

In this study we propose an ACASEM module which is highly adaptable in the context of the two-stage and multi-stage object detectors. To ensure its adaptability, we test it with diverse detectors namely Faster R-CNN, Cascade R-CNN and Double-Head-RCNN using multiple backbones (Resnet50 and Resnet101). To ensure its generalization capabilities we train and test it with multiple datasets namely DeepPCB and the Augmented PCB

Defect dataset and also perform training and testing with inputs of multiple sizes (640 × 640) and (800 × 800). This way we able to address the limitations of many works which do not address the issue of generalization across different datasets and input resolution. Table 1 summarizes the strength and weaknesses of some fusion and attention based deep learning models.

Methodology

Overview

In this section, we introduce key components, architecture and techniques related to our model. We also introduce means to evaluate performance of the model. Figure 2 represents a methodology overview for the adoption of ACASEM to enhance PCB defects by the two-stage and multi-stage object detectors.

| Work | FusionStrategy | Dataset | Evaluation | Strength | Limitations |
|----------------------------|-------------------------------------------------------|--------------------------------------------------------|-------------------------------|---------------------------------------------------------------------------------------|-------------------------------------------------------|
| Hu et al. ³⁴ | Hierarchical, multi-scale, multi-level | PKU PCB defect | 94.2 | Improved defect detection | Only one dataset was used, generalizability not clear |
| Li et al. ³⁵ | Hierarchical multi-scale, multi-level + attention | Micro-PCB | 96.3 <i>mAP@0.5</i> | Improved defect detection | Few defects researched, only one dataset was used |
| Chen et al. ³⁶ | Attention (SE) | Augmented PCB Defect | 97.04 <i>mAP@0.5</i> | Improved defect detection | Generalizability not clear |
| Xin et al. ³⁷ | Attention | Micro-PCB | 96.88 <i>mAP@0.5</i> | Improved defect detection | Only one dataset was used |
| Jiang et al. ³⁸ | Attention (CA) | Custom dataset with 5 classes and 420 annotations each | 96.46 <i>mAP@0.5</i> | Improved defect detection | Only one dataset was used compared with YOLOv3 only |
| Li et al. ³⁸ | Attention | Custom unpublished dataset 420 annotations each | 89.86 <i>mAP@0.5</i> | Improved defect detection | Only one dataset was used, generalizability not clear |
| Zhang et al. ³⁹ | Attention SENet + CBAM | PKU PCB defect | 96.14 <i>mAP@0.5</i> | Improved defect detection Good interpretability High accuracy Enhanced speed | Only one dataset was used |
| Ours | Hierarchical multi-scale, multilayer + attention (SE) | DeepPCB | 79.5 <i>mAP@0.5 : 0.95</i> | Improved defect detection | Detectors not lightweight |
| | | Augmented PCB Defect | 98.6 <i>mAP@0.5</i> | Generalizable Higher accuracy Highly adaptable | Detection not real-time |

Table 1. Comparison of some feature fusion and attention based works with our proposed methods.

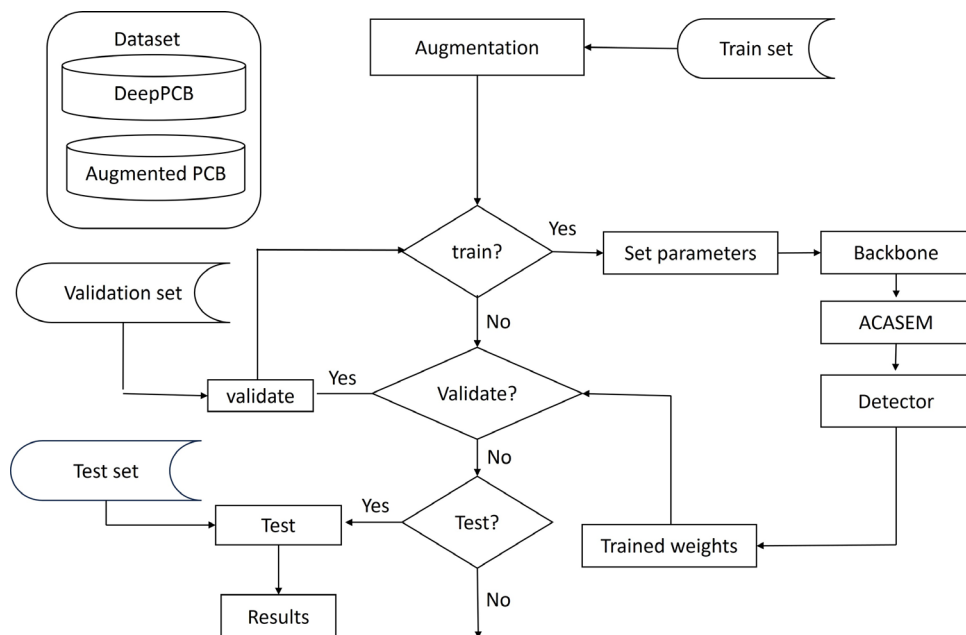


Figure 2. A methodology overview for enhancing defects detection with ACASEM.

Attentive context and semantic enhancement module (ACASEM) architecture

ACASEM, depicted in Fig. 3, uses the idea of multi-scale and multi-layer fusion. The key purpose is to fuse semantic information from different layers of backbones in order to get feature maps richer in the semantic information to enhance defects detection.

The module consists of two main sub-modules namely; the adaptable feature fusion, and the attention sub-modules. In the adaptable feature fusion sub-module, feature maps from different layers of the backbone are concatenated to form new feature maps with the same spatial dimensions as the original feature maps but with more channels. In our case, the down-sampled Resnet50 or Resnet101 second and third layer feature maps are concatenated to form intermediate feature maps, then channels of these feature maps are shuffled to enhance information flow between channels in order to improve inter channel learning. After that, global context is added through a broadcast mechanism.

The attention sub-module is composed of the squeeze and exitation mechanism, this helps to rank channels based on their importance to the task of object detection. It works in three main steps: firstly it squeezes feature maps into a vector, then it is followed with the excitation operation which models channels dependencies. In this phase, the *Sigmoid* function is used as an excitation operator, its main purpose is to induce non-linearity in the flow. The last step involves calculating the weighted feature map. This is obtained by multiplying the extracted feature map with weights⁴³. The SE operation can be represented with Eq. (1) for the squeeze stage, Eqs. (2) and (3) for rescaling. The ACASEM mode of functioning is summarized in aAlgorithm 1.

$$z_c = F_{sq}(u_c) = \sum_{i=1}^H \sum_{j=1}^H u_c(i, j). \tag{1}$$

u_c is a feature map after a convolution on feature map x with filter c

$$s = F_{ex} = \sigma(g(z, W)) \tag{2}$$

$$F_{scale}(u_c, s_c) = s_c \cdot u_c \tag{3}$$

whereby, z is an input spcific operator, F_{scale} denotes the channel-wise product of scalar s_c and feature map u_c , σ represents the sigmoid activation function and W denotes weights.

-
1. $T_2 \leftarrow \text{downsample}(T_2, T_1)$
 2. $T_{concat} \leftarrow \text{concat}(T_2, T_1)$
 3. $T_{concat} \leftarrow \text{reshape}(T_{concat}, \text{beta})$
 4. $T_{concat} \leftarrow \text{transpose}(T_{concat}, 1, 2)$
 5. $T_{concat} \leftarrow \text{reshape}(T_{concat}, N, -1, h, w)$
 6. $T_{concat} \leftarrow \text{conv}_{1 \times 1}(T_{concat})$
 7. $C_0 \leftarrow \text{AvgPool}(C_0)$
 8. $T_{concat} \leftarrow C_0 + T_{concat}$
 9. $T_{concat} \leftarrow SE(T_{concat})$
 10. $\text{return } T_{concat}$
-

Algorithm 1. Attentive context and semantic enhancement operation.

C_0, T_1 and T_2 represent the input feature maps, in our case it is a layer1, layer2 and layer3 feature maps of Resnet50 and Resnet101 backbones. T_{concat} represents the concatenated feature maps, h and w are heights and

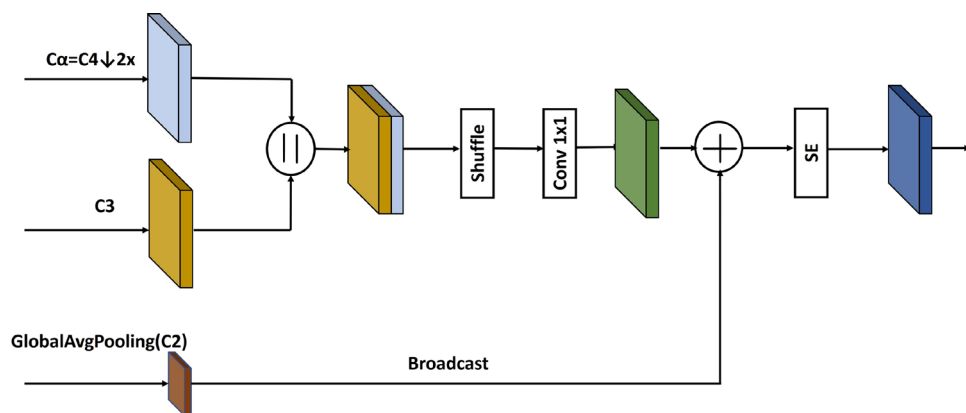


Figure 3. The attentive context and semantic enhancement module. C2,C3 and C4 are Layer1(conv2_x), Layer2 (conv3_x) and Layer3 (conv4_x) respectively.

width of the feature maps. Line 4 up to line 6 represent the shuffling operation while line 9 represents the squeeze and excitation operation of the model.

Backbones

Two backbones, Resnet50, and Resnet101 are used to test the efficiency of our method. Resnet50 has 48 convolution layers, one max pooling layer and one average pooling layer; Resnet101 has 99 convolutional layers, one max pooling layer and one average pooling layer. These layers are sometimes referred to as depth. The two backbones are built with residual blocks which use the idea of skip connections⁴⁴. These connections help to minimize the vanishing gradient problem by enabling a smooth propagation of weights, even when the depth of the network is large. In this way, deeper networks can be trained efficiently.

In the experiments, we use weights of Resnet50 and Resnet101 pre-trained on the caffe using ImageNet dataset. Table 2 shows an overall architecture of Resnet50 and Resnet101^{45,46}.

Detectors

We use the two-stage Faster R-CNN and Double-Head R-CNN and the Multi-stage Cascade R-CNN detectors to verify the effectiveness of the ACASEM. Every detector is trained separately. In their architecture, all three detectors use RPN for generating proposals which are further refined to detect objects. The difference between these detectors primarily lies on the number and arrangement of networks heads. Faster R-CNN network head has two fully connected layers while Double-Head R-CNN has two network heads, one head consisting of multiple fully connected layers and the other consisting of convolutional layers.

Cascade R-CNN consists of multiple stages which are sequentially trained. Each stage takes inputs from previous stage, refines it and passes it to the next stage for further refinement until the final stage. In Cascade R-CNN bounding box regression takes place in every stage.

Training loss

In the RPN, the classification loss used is a binary cross entropy loss (BCE) as shown in Eq. (4) while the loss used for bounding boxes regression in Faster R-CNN and Double-Head R-CNN is an L1 loss as shown in Eq. (5). On the other hand, smoothL1 regression Loss is used for Cascade-R-CNN in both RPN and the detection heads. Cross entropy loss as shown in Eq. (6) is used in all three detectors (Faster R-CNN, Double-Head R-CNN and Cascade R-CNN) for multi-class classification.

$$BCE = -(y \log(p) + (1 - y) \log(1 - p)) \quad (4)$$

$$L1Loss = \sum_{i=1}^n |y_{pred} - y_{target}| \quad (5)$$

$$CE = - \sum_{c=1}^M y_{i,c} \log(p_{i,c}) \quad (6)$$

In Eq. (4), y denotes the binary classes (0 or 1), while p is the probability of category class 1, and $1 - p$ represents the probability of binary class 0. In Eq. (5), y_{pred} denotes the predicted value while y_{target} denotes the ground truth value. In Eq. (6), M denotes the number of classes while $y_{i,c}$ is a binary indicator to specify whether c is the exact class and $p_{i,c}$ denotes the predicted probability of class c .

General model design

Figures 4, 5, and 6 show the general architectures of Faster R-CNN, Cascade R-CNN, and Double-Head R-CNN, all with ACASEM. In these models, the backbone is for extracting information from the input image, ACASEM for enriching the model with context and semantic information, RPN for objects proposals, Region of Interest (RoI) head is for pooling objects from feature maps based on the RPN proposals and the detector for eventual object detection and recognition.

| Resnet layer | Output size | Resnet 50 | Resnet 101 |
|-------------------|-------------|--------------------------------------------------------------------------|--------------------------------------------------------------------------|
| Conv_1x | 112 x 112 | 7 x 7, 64 stride 2 | 7 x 7, 64 stride 2 |
| Conv_2x | 56 x 56 | 3 x 3 max pool, stride 2 [(1 x 1, 64), (3 x 3, 64), (1 x 1, 256)] x 3 | 3 x 3 max pool, stride 2 [(1 x 1, 64), (3 x 3, 64), (1 x 1, 256)] x 3 |
| Conv_3x | 28 x 28 | [(1 x 1, 128), (3 x 3, 128), (1 x 1, 512)] x 4 | [(1 x 1, 128), (3 x 3, 128), (1 x 1, 512)] x 4 |
| Conv_4x | 14 x 14 | [(1 x 1, 256), (3 x 3, 256), (1 x 1, 1024)] x 3 | [(1 x 1, 256), (3 x 3, 256), (1 x 1, 1024)] x 23 |
| Conv_5x | 7 x 7 | [(1 x 1, 512), (3 x 3, 512), (1 x 1, 2048)] x 3 | [(1 x 1, 512), (3 x 3, 512), (1 x 1, 2048)] x 3 |
| | 1 x 1 | Average pool, 1000-d fc | |
| FLOPs, parameters | | 3.8×10^9 , 25.6 M | 7.6×10^9 , 44.5 M |

Table 2. Resnet50 and Resnet101 backbone architecture.

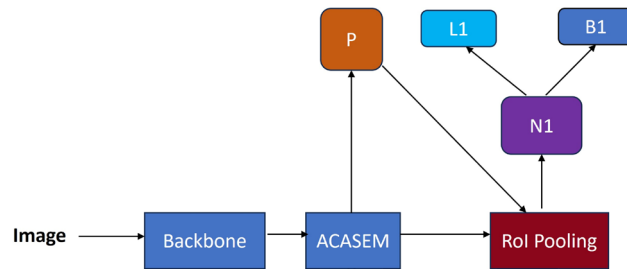


Figure 4. The architecture of Faster R-CNN w/ACASEM. P represents object proposals from the RPN, $N1$ is the network body, $L1$ is the class labels prediction and $B1$ is the bounding box predictions.

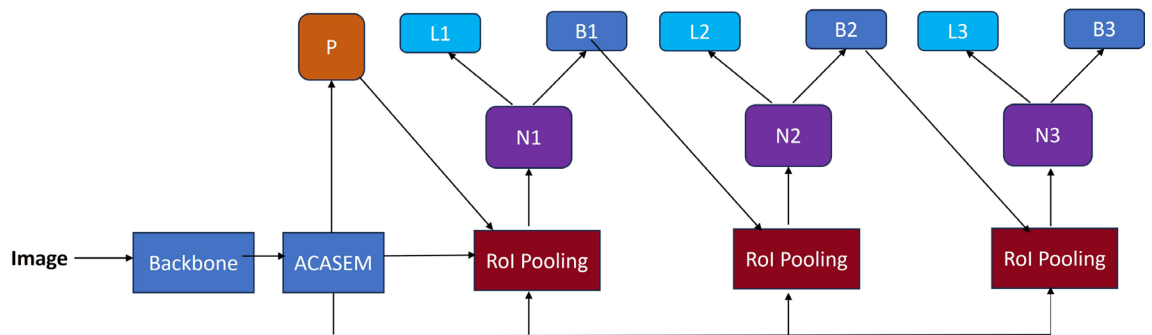


Figure 5. The architecture of Cascade R-CNN w/ACASEM. P represents object proposals from the RPN, N_j is the network body, L_j are class labels predictions and B_j are the bounding box predictions.

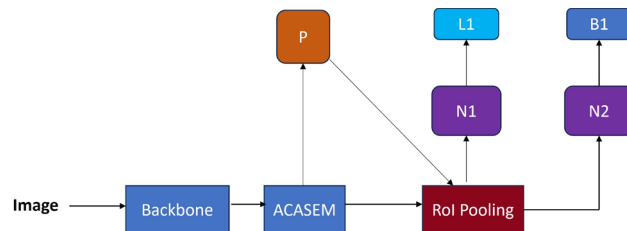


Figure 6. The architecture of Double-Head R-CNN w/ACASEM. P represents object proposals from the RPN, $N1$ and $N2$ are network bodies, $L1$ is the class labels prediction and $B1$ is the bounding box predictions.

During training, weights of the backbone's first layer are kept frozen while updates are conducted on the second and third layer. Feature maps from these layers are fused together using ACASEM, enriching them with context and semantic information to enhance defects' detection. Furthermore, in order to pick the best bounding boxes around the detected objects, NMS is used, whereby, Intersection over Union (IoU) threshold is set to 0.7. This means that all bounding boxes whose ratio of overlap with the ground truth is below 0.7 will be discarded^{47,48}.

Rolalign pooling strategy is adopted and the pooled feature map is set to the size of 14×14 . In the RPN, the binary cross-entropy and L1 losses are used for objectness prediction and for initial bounding boxes regression respectively. On the detector, cross-entropy loss is used for classification while L1 loss is used for bounding box regression in Faster R-CNN and Double-Head R-CNN while SmoothL1 Loss is used in Cascade R-CNN.

Evaluation metrics

mAP , AP , mAR and AR metrics are used to evaluate the detection accuracy of the models. In the case of experiments on the DeepPCB dataset, we go further to evaluate the precision accuracy of small and medium sized defects using the Average Precision for small objects (AP_s) and the Average Precision for medium sized object (AP_m). The recall for small and medium sized defects are measured with Average Recall for small (AR_s) and medium sized objects (AR_m). For the Augmented PCB Defect dataset, we use the Average Precision at $IoU = 0.5$ to compare the performance of models with other models. Equations (7)–(9) summarizes the calculations of the AP , mAP , and mAR respectively.

$$AP = \int_0^1 p(r) dr \quad (7)$$

$$mAP = \frac{1}{K} \sum_{i=1}^k AP_i \quad (8)$$

$$mAR = \frac{1}{K} \sum_{i=1}^k AR_i \quad (9)$$

In Eq. (7), $p(r)$ represents the precision function of recall while dr represents the recall steps for which AP is computed. In Eqs. (8) and (9), K represents the total number of categories, while i represents the i^{th} category.

Experiments

Dataset and parameter setting

Two datasets, namely DeepPCB⁴² and Augmented PCB Defect, are used. DeepPCB has 1,500 images, among them, 1,000 are in the training set and 500 are in the test set. Images in this dataset have been obtained through a linear scan Charge-Coupled Device (CCD) in the 48 pixels per 1 millimeter resolution. In this dataset, the image size is 640×640 and there are six defect categories, namely; open circuit (open), short-circuit (short), mousebite, spur, pinhole, and spurious copper (copper) categories⁴⁹.

The Augmented PCB Defect dataset has a total of 10,668 image crops with a resolution of 600×600 each. This is the augmented version of the PCB Defect dataset which originally had only 693 images each having a resolution of 2777×2138 . The Augmented PCB Defect dataset has six categories too. These include; open circuit, missing hole, spurious copper, spur, short and mousebite.

With regard to parameter settings, 1,024 input and output channels are used for the RPN. The number of proposals per image is set to 1,000, and anchor ratios used are (0.5, 1.0, 2.0). Non Maximum Suppression (NMS) has also been used as a strategy to obtain promising bounding boxes around objects. The IoU threshold used for NMS is 0.7, this means that all bounding boxes with the IoU less than that will be discarded.

For optimization, the Stochastic Gradient Descent (SGD) is used, the learning rate is set to 0.02, momentum is set to 0.9, weight decay is set to 0.0001, The maximum number of epochs used is 35 for the experiments conducted on the DeepPCB dataset and 18 for the experiments conducted on the Augmented PCB Defect dataset. Similarly, the linear warm up policy is adopted with the warm up iterations of 500 and the warm up ratio of 1.0/3 for the Cascade R-CNN based experiments while the warm up ratio of 0.001 is used for the Faster R-CNN and the Double-Head R-CNN based experiments.

All experiments involving Faster R-CNN detector were conducted on the A100 GPU. Experiments involving Cascade R-CNN were conducted on the V100 GPU while the Double-Head R-CNN experiments on the DeepPCB and Augmented PCB Defect dataset were conducted on V100 GPU and RTX 3080 GPUs respectively. The choice for the GPU was based on the availability of the GPU and the code compatibility.

Augmentation

Data augmentation was used to enhance the models' generalization capabilities. An online augmentation strategy based on random flips was used to increase the number of samples for efficient training and prevent overfitting. The online augmentation strategy maintains dataset consistency, it doesn't add extra images to the training set rather it adds specified variations to the image according to the defined augmentation strategy every time the image is process in each epoch. This way, the number of training examples increases while the number of images in the training set remain the same.

Furthermore, we applied a padding of 32 to images that goes into the training loop. This operation determines the minimum number of pixels required to make the width and height of images divisible by 32, this is crucial in case the aspect ratio of the intended image size and the original image size don't match. In the Convolutional Neural Networks (CNN) operations such as convolution and pooling require images whose dimensions are multiples of numbers such as 32, 64 or 128 to operate smoothly.

Experimental results

Results on the DeepPCB dataset

Results based on the faster R-CNN

Table 3 shows the results overview of experiments on the DeepPCB Dataset. In this table, mAP , AP_s , AP_m , AR and AR_s and AR_m metrics are used to summarize the performance of models.

Results based on the Double-Head R-CNN

Table 4 represents performance results of Double-Head R-CNN and variants on the DeepPCB dataset. In these results, Double-Head R-CNN w/Resnet50 and Double-Head R-CNN with Resnet101 are baseline3 and baseline4 respectively.

| Method | Input | mAP | AP _s | AP _m | mAR | AR _s | AR _m |
|----------------------------------------|-----------|-------------|-----------------|-----------------|-------------|-----------------|-----------------|
| Faster R-CNN w/ Resnet50 (baseline1a) | 640 × 640 | 74.4 | 74.4 | 74.8 | 79.9 | 79.4 | 80 |
| Faster R-CNN w/ Resnet50 (baseline1b) | 800 × 800 | 76.3 | 75.8 | 76.9 | 82.0 | 82.1 | 80.9 |
| Faster R-CNN w/ Resnet101 (baseline2a) | 640 × 640 | 73.8 | 72.8 | 74.8 | 79.9 | 79.2 | 80.0 |
| Faster R-CNN w/ Resnet101 (baseline2b) | 800 × 800 | 76.2 | 76.0 | 77.0 | 82.4 | 83.0 | 82.2 |
| Faster R-CNN w/Resnet50+ACASEM | 640 × 640 | 76.7 | 76.9 | 77.2 | 82.7 | 83.5 | 82.2 |
| Faster R-CNN w/Resnet50+ACASEM | 800 × 800 | 78.1 | 78.9 | 78.0 | 83.7 | 84.4 | 83.5 |
| Faster R-CNN w/Resnet101+ACASEM | 640 × 640 | 75.8 | 75.4 | 76.8 | 81.9 | 81.6 | 82 |
| Faster R-CNN w/Resnet101+ACASEM | 800 × 800 | 77.1 | 78.1 | 77.4 | 83.5 | 84.1 | 83.2 |

Table 3. Faster R-CNN performance on the DeepPCB dataset. Significant values are in bold.

| Method | Input | mAP | AP _s | AP _m | mAR | AR _s | AR _m |
|---------------------------------------------|-----------|-------------|-----------------|-----------------|-------------|-----------------|-----------------|
| Double-Head R-CNN w/ Resnet50 (baseline3a) | 640 × 640 | 74.7 | 75.1 | 74.8 | 79.2 | 79.4 | 79.1 |
| Double-Head R-CNN w/ Resnet50 (baseline3b) | 800 × 800 | 76.7 | 77.0 | 76.7 | 81.2 | 81.2 | 81.0 |
| Double-Head R-CNN w/ Resnet101 (baseline4a) | 640 × 640 | 74.5 | 74.7 | 74.7 | 79.2 | 79.0 | 79.3 |
| Double-Head R-CNN w/ Resnet101 (baseline4b) | 800 × 800 | 76.1 | 75.9 | 76.2 | 80.9 | 81.2 | 80.9 |
| Double-Head R-CNN w/Resnet50+ACASEM | 640 × 640 | 76.7 | 76.7 | 76.9 | 81.2 | 81.6 | 81.6 |
| Double-Head R-CNN w/Resnet50+ACASEM | 800 × 800 | 78.5 | 78.9 | 78.5 | 82.8 | 83.3 | 82.6 |
| Double-Head R-CNN w/Resnet101+ACASEM | 640 × 640 | 76.4 | 76.4 | 76.6 | 81.0 | 81.3 | 80.8 |
| Double-Head R-CNN w/Resnet101+ACASEM | 800 × 800 | 78.0 | 77.9 | 78.2 | 82.5 | 82.7 | 82.3 |

Table 4. Double-Head R-CNN performance on the DeepPCB dataset. Significant values are in bold.

Results based on the Cascade R-CNN

Table 5 highlights performance results of Cascade R-CNN and Cascade R-CNN w/ ACASEM at different image input resolutions and Resnet backbones on the DeepPCB dataset. In this configuration, Cascade R-CNN w/ Resnet50 and Cascade R-CNN w/Resnet 101 are baseline5 and baseline6 respectively.

Results on the Augmented PCB Defect dataset

Results based on the Faster R-CNN

Table 6 summarizes the performance of the baselines (Faster R-CNN) and Faster R-CNN w/ACASEM on the Augmented PCB Defect dataset. In this table, the results of the detection accuracy are compared at AP₅₀. The comparison is based on the performance of models on the Resnet50 and Resnet101.

Results based on the Cascade R-CNN

Table 7 presents the performance of the Cascade R-CNN baselines and Cascade R-CNN + ACASEM methods using resnet50 and Resnet101 backbones on the Augmented PCB Defect dataset.

Results based on the Double-Head R-CNN

Table 8 highlights the performance of Double-Head R-CNN + ACASEM and the Double-Head R-CNN baselines on the Augmented PCB Defect dataset using Resnet50 and Resnet101 backbones.

| Method | Input | mAP | AP _s | AP _m | mAR | AR _s | AR _m |
|-----------------------------------------|-----------|-------------|-----------------|-----------------|-------------|-----------------|-----------------|
| Cascade R-CNN w/ Resnet50 (baseline5a) | 640 × 640 | 75.3 | 75.0 | 75.6 | 80.2 | 80.2 | 80.3 |
| Cascade R-CNN w/ Resnet50 (baseline5b) | 800 × 800 | 77.1 | 76.8 | 77.2 | 81.7 | 81.5 | 81.6 |
| Cascade R-CNN w/ Resnet101 (baseline6a) | 640 × 640 | 75.2 | 74.7 | 75.6 | 80.3 | 80.4 | 80.2 |
| Cascade R-CNN w/ Resnet101 (baseline6b) | 800 × 800 | 76.4 | 76.0 | 76.7 | 81.0 | 80.2 | 81.3 |
| Cascade R-CNN w/Resnet50+ACASEM | 640 × 640 | 78.4 | 77.3 | 78.8 | 82.9 | 82.7 | 83.1 |
| Cascade R-CNN w/Resnet50+ACASEM | 800 × 800 | 79.5 | 78.9 | 79.8 | 83.7 | 83.1 | 83.9 |
| Cascade R-CNN w/Resnet101+ACASEM | 640 × 640 | 77.8 | 76.4 | 78.3 | 82.5 | 81.7 | 82.7 |
| Cascade R-CNN w/Resnet101+ACASEM | 800 × 800 | 78.9 | 77.7 | 79.4 | 83.2 | 82.6 | 83.4 |

Table 5. Cascade R-CNN performance on the DeepPCB dataset. Significant values are in bold.

| Method | <i>mAP</i> | <i>D</i> ₁ | <i>D</i> ₂ | <i>D</i> ₃ | <i>D</i> ₄ | <i>D</i> ₅ | <i>D</i> ₆ |
|---------------------------------------|-------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| Faster R-CNN w/Resnet50 (baseline7a) | 97.6 | 96.4 | 97.3 | 97.4 | 97.1 | 98 | 99.3 |
| Faster R-CNN w/Resnet101 (baseline7b) | 97.9 | 98 | 98 | 97.6 | 96.5 | 97.8 | 99.2 |
| Faster R-CNN w/Resnet50+ ACASEM | 98.6 | 98.5 | 98.2 | 98.2 | 98 | 99.2 | 99.2 |
| Faster R-CNN w/Resnet 101+ ACASEM | 98.5 | 98.9 | 97.6 | 98.3 | 97.8 | 99 | 99.3 |

Table 6. Faster R-CNN performance on the Augmented PCB Defect dataset. *D*₁ stands for open-circuit, *D*₂ : short, *D*₃ : mousebite, *D*₄ : spur, *D*₅ : spurious_copper and *D*₆ : missing_hole defects. Significant values are in bold.

| Method | <i>mAP</i> | <i>D</i> ₁ | <i>D</i> ₂ | <i>D</i> ₃ | <i>D</i> ₄ | <i>D</i> ₅ | <i>D</i> ₆ |
|----------------------------------------|-------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| Cascade R-CNN w/Resnet50 (baseline8a) | 98.0 | 98.4 | 96.7 | 98.1 | 97.5 | 98.2 | 99.5 |
| Cascade R-CNN w/Resnet101 (baseline8b) | 98.3 | 98.1 | 97.5 | 98.2 | 97.7 | 98.8 | 99.5 |
| Cascade R-CNN w/Resnet50+ ACASEM | 98.6 | 98.9 | 97.6 | 98.6 | 98.0 | 99.4 | 99.2 |
| Cascade R-CNN w/Resnet 101+ ACASEM | 98.5 | 99.0 | 97.8 | 98.9 | 97.4 | 99.1 | 99.1 |

Table 7. Cascade R-CNN performance on the Augmented PCB Defect dataset. *D*₁ stands for open-circuit, *D*₂ : short, *D*₃ : mousebite, *D*₄ : spur, *D*₅ : spurious_copper and *D*₆ : missing_hole defects. Significant values are in bold.

| Method | <i>mAP</i> | <i>D</i> ₁ | <i>D</i> ₂ | <i>D</i> ₃ | <i>D</i> ₄ | <i>D</i> ₅ | <i>D</i> ₆ |
|--------------------------------------------|-------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| Double-Head R-CNN w/Resnet50 (baseline9a) | 97.0 | 95.4 | 97.3 | 96.1 | 96.5 | 97.3 | 99.5 |
| Double-Head R-CNN w/Resnet101 (baseline9b) | 98.0 | 97.8 | 97.5 | 96.1 | 98.1 | 99.2 | 98.0 |
| Double-Head R-CNN w/Resnet50+ ACASEM | 98.6 | 98.0 | 98.3 | 98.8 | 98.7 | 98.6 | 99.4 |
| Double-Head R-CNN w/Resnet101+ ACASEM | 98.2 | 98.5 | 97.8 | 97.9 | 97.9 | 98.2 | 99.1 |

Table 8. Double-Head R-CNN performance on the Augmented PCB Defect dataset. *D*₁ stands for open-circuit, *D*₂ : short, *D*₃ : mousebite, *D*₄ : spur, *D*₅ : spurious_copper and *D*₆ : missing_hole defects. Significant values are in bold.

Results analysis

Analysis of Faster R-CNN w/ACASEM performance on the DeepPCB dataset

Based on the experimental results presented in Table 3, When the backbone is Resnet50 and the input resolution is 640×640 , Faster R-CNN w/ACASEM outperforms the baseline method (baseline1a) by 2.3% in the overall *mAP*, by 2.4% in *AP_m* and by 2.5% in *AP_s*. It also surpasses the baseline method by 2.8% in *mAR*, by 4% in *AR_s* and by 2.2% in *AR_m*. Similarly with the same backbone but the input resolution of 800×800 , Faster R-CNN w/ACASEM surpasses the corresponding baseline method (baseline1b) by 1.8% in *mAP*, by 1.1% in *AP_m* and by 3.1% in *AP_s*. It also surpasses that baseline on the *mAR*, *AR_m* and *AR_s*, by 1.7%, 2.6% and 2.3% respectively. The best performance of Faster R-CNN w/ACASEM on the DeepPCB dataset using Resnet50 backbone and the input resolution of 800×800 is summarized with the precision-recall curves shown in Fig. 7.

When the backbone is Resnet101 and the input resolution is 640×640 , Faster R-CNN w/ACASEM outperforms the baseline method (baseline2a) by 2% in the overall *mAP*, by 2% in the *AP_m* and by 1.6% in the *AP_s*. In the case of recalls, it surpasses the baseline method by 2% in *mAR*, by 2% in *AR_m* and by 2.4% in *AR_s*. With the same backbone and input resolution of 800×800 , Faster R-CNN w/ACASEM outperforms the baseline (baseline2b) by 0.9% in the overall *mAP*, by 0.4% in the *AP_m*, and by 2.1% in the *AP_s*. It also surpasses the baseline in the *mAR*, *AR_m* and *AR_s* by 1.1%, 1.0% and 1.1% respectively.

Analysis of Double-Head R-CNN w/ACASEM performance on the DeepPCB dataset

Based on the experimental results presented in Table 4, when the input image resolution is 640×640 and the backbone is Resnet50, Double-Head R-CNN w/ACASEM outperforms the baseline method (baseline3a) by 2.0% in *mAP*, by 1.6% in *AP_s*, and by 2.1% in *AP_m*. Also, there is an increase in *mAR* by 2.0%, *AR_s* by 2.2% and *AR_m* by 2.5% respectively. Similarly when the input resolution is 800×800 , it outperforms the baseline (baseline3b) by 1.8% in *mAP*, by 1.9% in *AP_s*, and 1.8% in *AP_m*. Additionally, it surpasses the baseline by 1.6% in *mAR*, 2.1% in *AR_s*, and 1.6% in *AR_m*. The precision-recall curve in Fig. 8 summarizes the best performance of Double-Head R-CNN on the DeepPCB dataset at the input size of 800×800 and Resnet50 backbone.

When the backbone is Resnet101 and the resolution of the input image is 640×640 , Double-Head R-CNN w/ACASEM obtains superior performance to the baseline (baseline4a) by 1.7% in *mAP*. *AP_s* increases by 1.3%

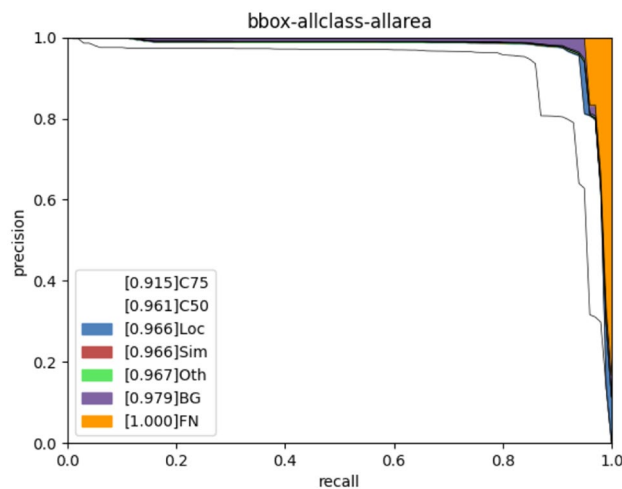


Figure 7. Precision-Recall curves of Faster R-CNN w/ACASEM performance on DeepPCB dataset with Resnet50 backbone and input resolution of 800×800 .

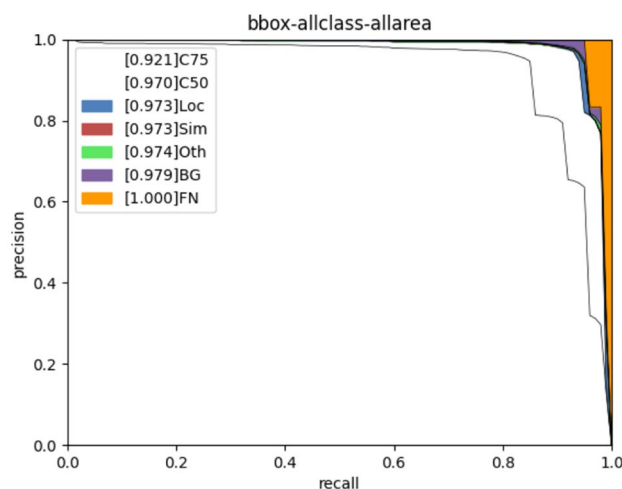


Figure 8. Precision-recall curves of Double-Head R-CNN w/ACASEM performance on DeepPCB dataset with Resnet50 backbone and input resolution of 800×800 .

and AP_m increases by 1.8%. Similarly, mAR increases by 1.8%, AR_s by 1.9%, and AR_m by 1.7%. Likewise, when the input is 800×800 , Double-Head R-CNN w/ACASEM outperforms the baseline (baseline4b) by 1.9% in mAP , AP_s improves by 4.0%, and AP_m improves by 2.0%. Also, mAR improves by 1.6%, AR_s and AR_m improve by 1.5% and 1.5% respectively.

Analysis of Cascade R-CNN w/ACASEM performance on the DeepPCB dataset

Results presented in Table 5 has shown that, with the input resolution of 640×640 and Resnet50 backbone, Cascade R-CNN w/ Resnet50 + ACASEM excels the counterpart baseline (baseline5a) by 3.1% in mAP , 2.3% in AP_s and 2.3% in AP_m . Furthermore, it excels the baseline by 2.7% in mAR , by 2.5% and 2.8% in AR_s and AR_m respectively. At the same time when the input resolution is 800×800 , Cascade R-CNN w/ Resnet50 + ACASEM surpasses the baseline (baseline5b) by 2.4% in mAP , by 2.1% in AP_s , and by 2.6% in AP_m . Furthermore, It surpasses the baseline by 2.0% in mAR , by 1.6% and 2.3% in AR_s and AR_m respectively. Figure 9 presents the precision-recall curves for the best performance of Cascade R-CNN w/ACASEM on the DeepPCB dataset at the input resolution of 800×800 and Resnet50 backbone.

When the input resolution is 640×640 and the backbone is Resnet101, Cascade R-CNN w/ Resnet101 + ACASEM model outperforms the baseline method (baseline6a) in mAP , AP_s and AP_m by 2.6%, 1.7%, and 2.7% respectively. At the same time, it surpasses the baseline in the mAR , AR_s , and AR_m by 2.2%, 1.3% and 2.5% respectively. Similarly, when the input resolution is 800×800 , Cascade R-CNN w/ Resnet101 + ACASEM achieves superior performance to the baseline method (baseline6b) by 2.5%, 1.7% and 1.7% in mAP , AP_s and AP_m respectively, and also showcases its prowess over the baseline as it eclipses it by 2.2%, 2.4%, and 2.1% in mAR , AR_s , and AR_m respectively.

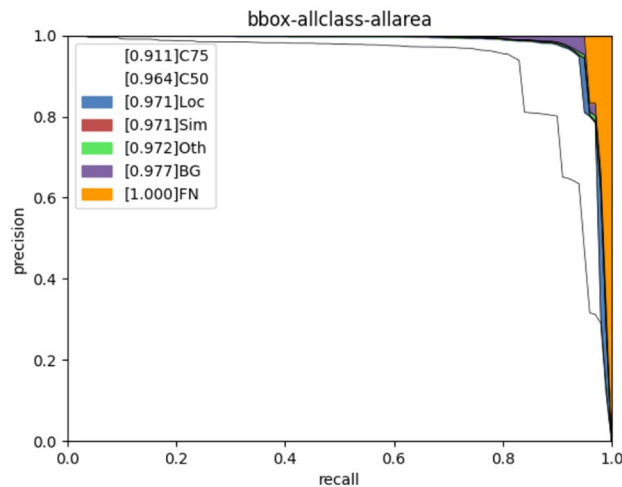


Figure 9. Precision-recall curves of cascade R-CNN w/ACASEM performance on DeepPCB dataset with Resnet50 backbone and input resolution of 800×800 .

Analysis of Faster R-CNN w/ACASEM performance on the Augmented PCB Defect dataset

Based on Table 6, it is observed that when the ACASEM is incorporated in the Faster R-CNN, new models have higher detection accuracies than baseline methods. When the backbone is Resnet50, Faster R-CNN w/ACASEM outperforms the baseline method (baseline7a) in the overall *mAP* by 1% and in the *AP* values of the defective categories: open circuit by 0.9%, short by 0.9%, mousebite by 0.8%, spur by 0.9%, spurious copper by 1%, and missing hole by 0.9%.

Likewise, when the backbone is Resnet101, Faster R-CNN w/ACASEM outperforms its counterpart baseline (baseline7b) in the overall *mAP* by 0.6% and in the *AP* values category by category: open circuit by 0.9%, mousebite by 0.7%, spur by 1.3%, spurious copper by 1.2%, and missing hole by 0.1%.

Analysis of Cascade R-CNN w/ACASEM performance on the Augmented PCB Defect dataset

From Table 7, we deduce that there is an increase in *mAP* by 0.6% when ACASEM is adopted in the Cascade R-CNN w/Resnet50 as compared to the baseline method (baseline8a). Also, the *AP* in detecting open-circuit, mousebite, short(circuit), spur, spurious-copper defects improves by 0.5%, 0.9%, 0.5%, and 1.2% respectively. Similarly when the backbone is Resnet101, Cascade R-CNN w/ACASEM outperforms the baseline method (baseline8b) by 0.2% in the *mAP* and also achieves a 0.9%, 0.2%, 0.7%, and 0.3% *AP* improvement in detecting open circuit defects, short circuit, mousebite and spurious copper defects respectively.

Analysis of Double-Head R-CNN w/ACASEM performance on the Augmented PCB Defect dataset

Based on Table 8, the inclusion of the ACASEM into Double-Head R-CNN w/Resnet50 increased the *mAP* by 1.6% above the baseline method (baseline9a). It is also observed that the *AP* for detecting individual defect categories improved by 2.6%, 2.7%, 1%, 2.2%, 1.3% for open-circuit, mousebite, short(circuit), spur and spurious-copper defects respectively. Similarly when the ACASEM is incorporated into Double-Head R-CNN w/Resnet101, it surpassed the baseline method (baseline9b) in *mAP* by 0.2% while the detection accuracy of the open-circuit, short, mousebite, and missing hole defects improved by 0.7%, 0.3%, 1.8%, and 1.1% respectively.

Performance comparison with other methods on the DeepPCB dataset

Based on Table 9, we observe that models which contain ACASEM, outperform YOLOv3, SSD, ID-YOLO and Lightnet networks in terms of *mAP*, and *mAR*. Furthermore it is observed that Cascade R-CNN w/Resnet50 + ACASEM performs competitively with DDTR w/ReswinT-T and DDTR w/ReswinT-S when the image resolution is 640×640 . However, increasing the input resolution to 800×800 results in Cascade w/Resnet50 + ACASEM outperforming the current DDTR w/ReswinT-T and DDTR w/ReswinT-S *mAP* at 640×640 input resolution.

Performance comparison with other methods on the Augmented PCB Defect dataset

From the experimental results presented in Table 10, our models (Faster R-CNN w/ACASEM, Cascade R-CNN w/ACASEM and Double-Head R-CNN w/ACASEM) each utilizing Resnet50 backbone and attaining the *mAP* of 98.6% on the Augmented PCB Defect dataset, outperformed FPN and Faster R-CNN (finetuned) each employing Resnet101 backbone by 6.37% and 2.16% in *mAP* respectively. They also outperformed Faster R-CNN which utilized VGG16 backbone by a significant margin of 40.03% in *mAP*.

| Method | Resolution | mAP | mAR |
|----------------------------------------------|------------|-------|-------|
| YOLOv3 ⁵⁰ | 640 × 640 | 70.71 | 78.6 |
| SSD ⁶ | 640 × 640 | 72.51 | 77.7 |
| ID-YOLO ⁵¹ | 640 × 640 | 71.48 | 77.62 |
| Lightnet ⁵² | 640 × 640 | 76.22 | 81.01 |
| Cascade R-CNN w/SwinT-T | 640 × 640 | 77.24 | 82.74 |
| Cascade R-CNN w/SwinT-S | 640 × 640 | 77.41 | 83.28 |
| DDTR w/ResSwinT-T ⁵³ | 640 × 640 | 78.75 | 83.53 |
| DDTR w/ResSwinT-S ⁵³ | 640 × 640 | 78.62 | 83.9 |
| Faster R-CNN w/Resnet50 + ACASEM (Ours) | 640 × 640 | 76.7 | 82.7 |
| Double-Head R-CNN w/Resnet50 + ACASEM (Ours) | 640 × 640 | 76.7 | 81.2 |
| Cascade R-CNN w/Resnet50 + ACASEM (Ours) | 640 × 640 | 78.4 | 82.9 |
| Faster R-CNN w/Resnet50 + ACASEM (Ours) | 800 × 800 | 78.1 | 83.7 |
| Double-Head R-CNN w/Resnet50 + ACASEM (Ours) | 800 × 800 | 78.5 | 82.8 |
| Cascade R-CNN w/Resnet50 + ACASEM (Ours) | 800 × 800 | 79.5 | 83.7 |

Table 9. Comparison of performance on the DeepPCB dataset.

| Method | Backbone/feature maps | Head | mAP |
|-----------------------------------------------------|-------------------------|------|-------|
| Faster R-CNN w/ VGG16 | <i>LastLayer</i> | 2fc | 58.57 |
| FPN w/ Resnet101 | (P_k) | 2fc | 92.23 |
| Faster R-CNN (Finetuned) ⁴⁹ w/ Resnet101 | C_5 | 2fc | 96.44 |
| YOLOv7 ⁵⁴ | – | – | 96.1 |
| YOLOv7 ⁵⁵ | ($S-E_1, S-E_3, E-E$) | – | 97.3 |
| CDI-YOLO ⁵⁶ | ($E_2, E-CA, S$) | – | 98.3 |
| Faster R-CNN w/Resnet50 + ACASEM (Ours) | (C_2, C_3, C_4) | 2fc | 98.6 |
| Double-Head R-CNN w/Resnet50 + ACASEM (Ours) | (C_2, C_3, C_4) | 2fc | 98.6 |
| Cascade R-CNN w/Resnet50 + ACASEM (Ours) | (C_2, C_3, C_4) | 2fc | 98.6 |

Table 10. Comparison of performance on the Augmented PCB Defect dataset. C_2, C_3, C_4 and C_5 refer to layer1, layer2, layer3 and layer4 of both resnet50 and resnet101 backbones. (P_k) represent the prediction layers of FPN, ($S-E_2, S-E_3, E-E_3$) represents feature maps in the Yolov7 backbone emanating from the first S-ELAN, third S-ELAN and the E-ELAN blocks while (E_2, E_3, S) are features from the second ELAN block, the ELAN-CA block and the SPPCSPC-DS block and 2fc represents the two fully connected layers in the detection heads.

Trade-off between training speed and mAP: selecting the most suitable backbone

In these experiments, backbones are used separately. Experiments are performed to prove the contribution of ACASEM under different backbones. However, from the experimental results, we can also deduce between Resnet50 and Resnet101 the most suitable backbone to use for our model in real-world scenario for PCB defects detection based on the nature of data available and other considerations such as computational cost and training speed.

From Table 11, it is observed that better mAP is obtained when the ACASEM is used with Resnet50 backbone, and the image input resolution is 800 × 800. However, faster training speed is obtained when the input resolution is 640 × 640, even though the mAP at this resolution is lower.

It can also be observed that when the backbone is Resnet101, the number of parameters are approximately 51% higher than when the backbone is Resnet50. Hence, for these reasons, the most suitable backbone to use with our model in a real-world scenario is Resnet50.

However Resnet50 and Resnet101 are both heavyweight backbones with higher computation cost, hence lightweight two-stage detectors may be considered for the real-time detection. In this scenario the ACASEM can easily be integrated into the two-stage lightweight detectors for higher detection accuracy while maintaining faster inference speed.

Visual evaluation

Figures 10 and 11 illustrate the performance of the models visually. All models gave similar visual results. Figure 10 shows defect detection on DeepPCB test image, while Fig. 11 shows defect detection on four Augmented PCB Defect dataset test images. Based on the visual results, all models detected defects with a very high confidence. No defect was undetected and no defect was misclassified. The success of these models on localization is largely due to spatial information preservation during feature fusion whereas the success in attaining high

| Model | Backbone + Neck | InputSize | Params(M) | Speed(s/iter) | mAP |
|---------------|--------------------|-----------|-----------|---------------|------|
| Faster R-CNN | Resnet50 /w ACASEM | 640 × 640 | 37.14 | 0.1624 | 76.7 |
| | | 800 × 800 | 37.14 | 0.1792 | 78.1 |
| Faster R-CNN | Resnet101 / ACASEM | 640 × 640 | 56.08 | 0.1745 | 75.8 |
| | | 800 × 800 | 56.08 | 0.1944 | 77.1 |
| Cascade R-CNN | Resnet50 /w ACASEM | 640 × 640 | 37.15 | 0.7183 | 78.4 |
| | | 800 × 800 | 37.15 | 0.7305 | 79.5 |
| Cascade R-CNN | Resnet101 / ACASEM | 640 × 640 | 56.09 | 0.7316 | 77.8 |
| | | 800 × 800 | 56.09 | 0.7574 | 78.9 |

Table 11. Trade-off between training speed and mAP.

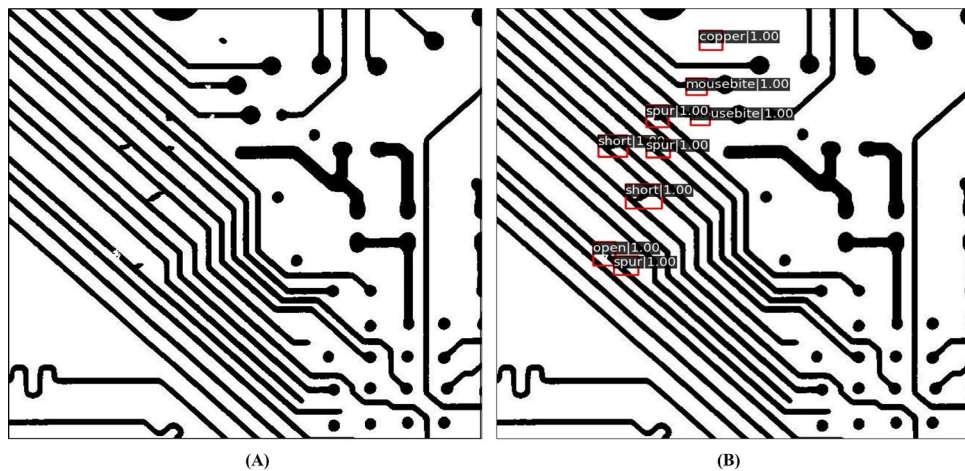


Figure 10. Detection results on DeepPCB dataset (zoom in for clearer bounding boxes).

detection confidence is largely due the use of feature maps with the most relevant features, rich in contextual and semantic information as a result of the attention mechanism employed, global and local context leverage, and an effective semantic information harnessing mechanism that avoids semantic confusion during feature fusion in the ACASEM.

Discussion

From the experimental results, models which contain the ACASEM achieved higher detection accuracy in terms of AP at IoU of 0.5, mAP ($IoU = 0.5:0.95$) and mAR than their baseline methods. Similarly, ACASEM improved the detection accuracy of small defects. This is evident from the increasing of AP_s and AR_s values in models which contain ACASEM in contrast to their baselines. The achievements of ACASEM could be attributed to the fact that it widens the receptive field by aggregating more semantic and context information from multiple feature maps of the backbone. The global context from the shallower layer 2 added via the broadcast operation increases the model context awareness, improving its capacity to identify and differentiate defects from the surroundings. In addition, the shuffling operation enhances the information flow across the channels and creates diverse feature maps that improve the model's ability to generalize across variations within the data. The multilayer-fusion that involves concatenation instead of addition preserves feature maps spatial and semantic information, reducing the problem of feature confusion. Similarly, its inbuilt attention mechanism made of the squeeze and excitation sub-module helps to prioritize the most important information. These factors may have led to ACASEM enhancing detection accuracy.

Also, we observed that baseline methods which use Resnet50 as a backbone perform better on the DeepPCB dataset than baselines which use Resnet101. The reason for this phenomenon is that, despite applying augmentation, DeepPCB data is still insufficient to optimally train deeper backbone like Resnet101. However, it is enough for backbones with less layers like Resnet50. On the other hand, we observed that on the Augmented PCB Defect dataset which contains more data, the power of Resnet101 manifests clearly as the detection accuracy of baseline methods which use it as a backbone is higher than that of baselines which use Resnet50 as a backbone. The very same phenomenon was also observed by Silva et al.⁵⁷ who pointed out that shallower models perform well on DeepPCB because images in this dataset do not represent complex texture and color information. However, deeper backbones have higher computational costs than shallow ones because they have more parameters. Hence, a trade-off between detection accuracy and computational cost may be taken into consideration when dealing with huge PCB defects datasets.

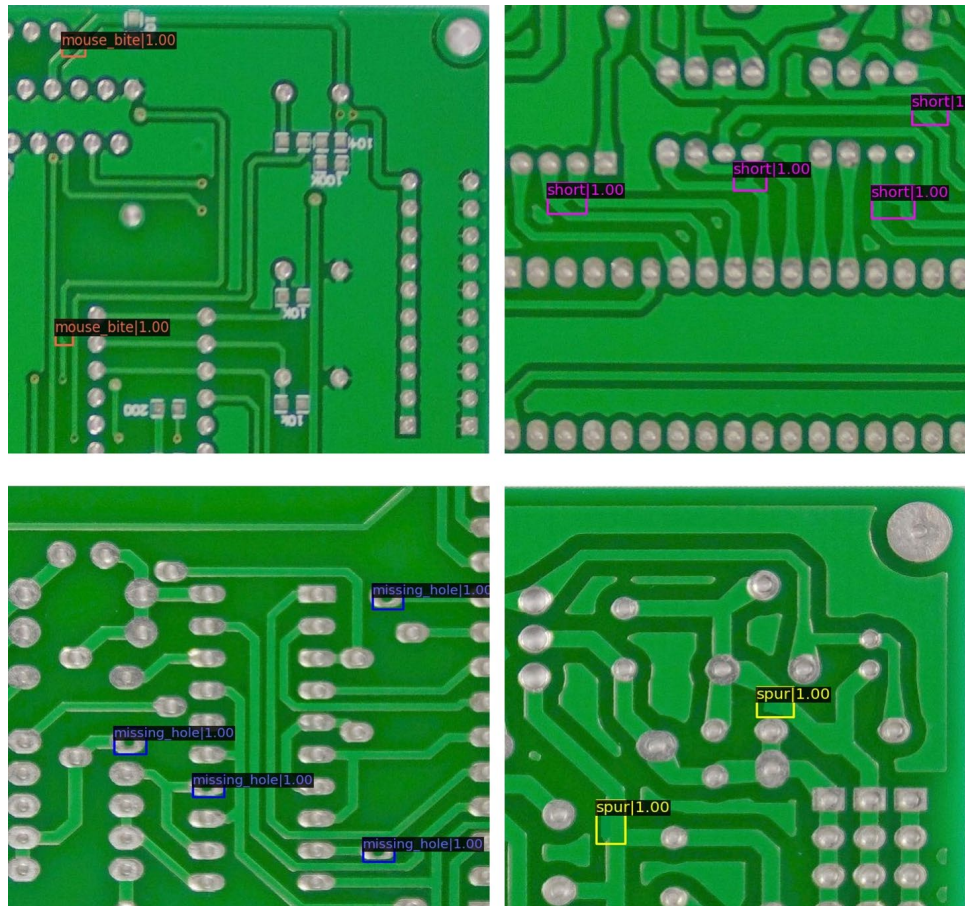


Figure 11. Detection result on Augmented PCB Defect.

In addition, the experimental results reveal that when the input resolution was increased from 640×640 to 800×800 , the defect detection accuracy based on both the average precision and average recall metrics increased. This increment may be attributed to the availability of more context information due to a larger field of view and as a consequence of handling object scale variations much better. Hence as objects in images appear at different scales, the network is able to learn the variations efficiently.

Furthermore, the positive increment in mAP , AP_s and AP_m imply that the quality of the overall detection, the detection of small sized, and medium sized defects has improved whereas the positive increment in mAR , AR_s and AR_m imply that the number of defects successfully captured as true defects has increased. Since defects in PCB is a very critical issue then the more defects a model can capture, the better it can be considered. Thus average recall information is extremely important to highlight the capability of the model to capture the true defects. Our works having achieved improvements in precision and recall based metrics not only prove that they can enhance the quality of the defect detection but can also identify and capture more defects present in the boards.

The good performance achieved by our works in diverse experimental set up involving multiple detectors, backbones and input resolution showcases our works' adaptability and generalizability. This imply that the ACASEM can be incorporated into other detectors such as the two-stage lightweight detectors to enhance the detection accuracy of those detectors with the addition of just a little computation cost. This way, high accuracy, less computational cost, faster inference speed, and real-time two-stage PCB defect detectors may be obtained.

Conclusions

In this work, the plug and play ACASEM has been proposed to enhance the detection accuracy of the PCB defects. This module uses the idea of multiscale feature fusion, global context, and attention. It captures both the context and semantic information of feature maps from different layers of the backbone and fuses them to obtain enriched feature maps used for object detection. Squeeze and excitation attention sub-module within the ACASEM allows a network to learn and focus on the most relevant information and subdue the less relevant ones.

The ACASEM was tested on the state-of-the-art Faster R-CNN, Double-Head R-CNN and Cascade R-CNN models, and the results have shown that adopting the ACASEM outperforms the baseline methods in the overall precision in terms of mAP , AP_m , and AP_s . It also outperforms the baseline methods in mAR , AR_s and AR_m , hence it is evident that ACASEM not only improves the detection accuracy of small PCB defects, but the overall accuracy as well.

In addition, results have shown improvement in defect detection accuracy compared to the existing methods. This implies that ACASEM can provide a means of achieving more robust models with better detection accuracy, and with little addition of computational cost.

Despite the achievements of this work in terms of the enhanced accuracy, adaptability, and generalization the next step in the direction of this research is to further reduce computational costs and the inference time. We believe that leveraging the ACASEM's good adaptability, it can be easily integrated into the two-stage lightweight object detectors to create more powerful lightweight models. By incorporating the ACASEM, compressing the RPN via the depthwise separable convolution strategy to reduce the number of FLOPs and parameters, and by employing the Light-Head RCNN detector, a good performing two-stage lightweight model can be obtained.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Received: 19 February 2024; Accepted: 1 August 2024

Published online: 05 August 2024

References

- Ong, N. Manufacturing cost estimation for PCB assembly: An activity-based approach. *Int. J. Prod. Econ.* **38**, 159–172 (1995).
- Zhang, Q. & Liu, H. Multi-scale defect detection of printed circuit board based on feature pyramid network. In *2021 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, 911–914 (IEEE, 2021).
- Rau, H. & Wu, C.-H. Automatic optical inspection for detecting defects on printed circuit board inner layers. *Int. J. Adv. Manuf. Technol.* **25**, 940–946 (2005).
- Moganti, M. & Ercal, F. Automatic PCB inspection systems. *IEEE Potentials* **14**, 6–10 (1995).
- Ibrahim, Z. & Al-Attas, S. A. R. Wavelet-based printed circuit board inspection system. *Int. J. Signal Process.* **1**, 73–79 (2004).
- Liu, W. et al. SSD: Single shot multibox detector. In *Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, 21–37 (Springer, 2016).
- Lin, T.-Y. et al. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2117–2125 (2017).
- Ren, S., He, K., Girshick, R. & Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Adv. Neural Inf. Process. Syst.* **28** (2015).
- Radev, P. & Shirvaikar, M. Enhancement of flying probe tester systems with automated optical inspection. In *2006 Proceeding of the Thirty-Eighth Southeastern Symposium on System Theory*, 367–371 (IEEE, 2006).
- Fang, J., Xiong, K., Zhang, C., Shang, L. & Gao, G. A hybrid optical detection algorithm for plug-in capacitor. In *International Conference on Frontiers of Electronics, Information and Computation Technologies*, 1–4 (2021).
- Wu, Y. et al. Rethinking classification and localization for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10186–10195 (2020).
- Cai, Z. & Vasconcelos, N. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6154–6162 (2018).
- Gan, Y. S., Chee, S.-S., Huang, Y.-C., Liong, S.-T. & Yau, W.-C. Automated leather defect inspection using statistical approach on image intensity. *J. Ambient. Intell. Humaniz. Comput.* **12**, 9269–9285 (2021).
- Djukic, D. & Spuzic, S. Statistical discriminator of surface defects on hot rolled steel. *Image Vis. Comput.* **158**–163 (2007).
- Zheng, X., Zheng, S., Kong, Y. & Chen, J. Recent advances in surface defect inspection of industrial products using deep learning techniques. *Int. J. Adv. Manuf. Technol.* **113**, 35–58 (2021).
- Da, Y., Dong, G., Wang, B., Liu, D. & Qian, Z. A novel approach to surface defect detection. *Int. J. Eng. Sci.* **133**, 181–195 (2018).
- Chan, C.-H. & Pang, G. K. Fabric defect detection by Fourier analysis. *IEEE Trans. Ind. Appl.* **36**, 1267–1276 (2000).
- Baykut, A., Atalay, A., Erçil, A. & Güler, M. Real-time defect inspection of textured surfaces. *Real-Time Imaging* **6**, 17–27 (2000).
- Karimi, M. H. & Asemani, D. Surface defect detection in tiling industries using digital image processing methods: Analysis and evaluation. *ISA Trans.* **53**, 834–844 (2014).
- Ran, G., Lei, X., Li, D. & Guo, Z. Research on PCB defect detection using deep convolutional neural network. In *2020 5th International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*, 1310–1314 (IEEE, 2020).
- Li, M. et al. Multisensor image fusion for automated detection of defects in printed circuit boards. *IEEE Sens. J.* **21**, 23390–23399 (2021).
- An, K. & Zhang, Y. Lpvit: A transformer based model for PCB image classification and defect detection. *IEEE Access* **10**, 42542–42553 (2022).
- Poudel, R. P., Liwicki, S. & Cipolla, R. Fast-scnn: Fast semantic segmentation network. *arXiv preprint arXiv:1902.04502* (2019).
- Chen, W. et al. Fasterseg: Searching for faster real-time semantic segmentation. *arXiv preprint arXiv:1912.10917* (2019).
- Yu, C. et al. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 325–341 (2018).
- Chen, L.-C., Papandreou, G., Schroff, F. & Adam, H. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587* (2017).
- Long, J., Shelhamer, E. & Darrell, T. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3431–3440 (2015).
- Wang, Q. et al. Eca-net: Efficient channel attention for deep convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11534–11542 (2020).
- Lee, H., Kim, H.-E. & Nam, H. Srm: A style-based recalibration module for convolutional neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1854–1862 (2019).
- Dosovitskiy, A. et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).
- Carion, N. et al. End-to-end object detection with transformers. In *European Conference on Computer Vision*, 213–229 (Springer, 2020).
- Woo, S., Park, J., Lee, J.-Y. & Kweon, I. S. Cbam: Convolutional block attention module. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 3–19 (2018).
- Park, J., Woo, S., Lee, J.-Y. & Kweon, I. S. Bam: Bottleneck attention module. *arXiv preprint arXiv:1807.06514* (2018).
- Hu, B. & Wang, J. Detection of PCB surface defects with improved faster-rcnn and feature pyramid network. *IEEE Access* **8**, 108335–108345 (2020).

35. Li, D. *et al.* An improved PCB defect detector based on feature pyramid networks. In *Proceedings of the 2020 4th International Conference on Computer Science and Artificial Intelligence*, 233–239 (2020).
36. Chen, W., Huang, Z., Mu, Q. & Sun, Y. PCB defect detection method based on transformer-yolo. *IEEE Access* **10**, 129480–129489 (2022).
37. Xin, H., Chen, Z. & Wang, B. PCB electronic component defect detection method based on improved yolov4 algorithm. In *Journal of Physics: Conference Series*, Vol. 1827, 012167 (IOP Publishing, 2021).
38. Li, J., Li, W., Chen, Y. & Gu, J. Research on object detection of PCB assembly scene based on effective receptive field anchor allocation. *Comput. Intell. Neurosci.* **2022**, 7536711 (2022).
39. Zhang, Y., Xu, M., Zhu, Q., Liu, S. & Chen, G. Improved yolov5s combining enhanced backbone network and optimized self-attention for PCB defect detection. *J. Supercomput.* 1–29 (2024).
40. Gao, Y., Lin, J., Xie, J. & Ning, Z. A real-time defect detection method for digital signal processing of industrial inspection applications. *IEEE Trans. Ind. Inf.* **17**, 3450–3459 (2020).
41. Kamalpreet, K. & Beant, K. PCB defect detection and classification using image processing. *Int. J. Emerg. Res. Manag. Technol.* **3**, 1–10 (2014).
42. Tang, S., He, F., Huang, X. & Yang, J. Online pcb defect detector on a new PCB defect dataset. *arXiv preprint arXiv:1902.06197* (2019).
43. Hu, J., Shen, L. & Sun, G. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7132–7141 (2018).
44. Orhan, A. E. & Pitkow, X. Skip connections eliminate singularities. *arXiv preprint arXiv:1701.09175* (2017).
45. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778 (2016).
46. Wu, H., Xin, M., Fang, W., Hu, H.-M. & Hu, Z. Multi-level feature network with multi-loss for person re-identification. *IEEE Access* **7**, 91052–91062 (2019).
47. Girshick, R., Donahue, J., Darrell, T. & Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 580–587 (2014).
48. Felzenszwalb, P. F., Girshick, R. B., McAllester, D. & Ramanan, D. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.* **32**, 1627–1645 (2009).
49. Ding, R., Dai, L., Li, G. & Liu, H. Tdd-net: A tiny defect detection network for printed circuit boards. *CAAI Trans. Intell. Technol.* **4**, 110–116 (2019).
50. Redmon, J. & Farhadi, A. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (2018).
51. Qin, L. *et al.* Id-yolo: Real-time salient object detection based on the driver's fixation region. *IEEE Trans. Intell. Transp. Syst.* **23**, 15898–15908 (2022).
52. Liu, J., Li, H., Zuo, F., Zhao, Z. & Lu, S. Kd-lightnet: A lightweight network based on knowledge distillation for industrial defect detection. *IEEE Trans. Instrum. Meas.* (2023).
53. Feng, B. & Cai, J. PCB defect detection via local detail and global dependency information. *Sensors* **23**, 7755 (2023).
54. Wang, C.-Y., Yeh, I.-H. & Liao, H.-Y. M. You only learn one representation: Unified network for multiple tasks. *arXiv preprint arXiv:2105.04206* (2021).
55. Wang, C.-Y., Bochkovskiy, A. & Liao, H.-Y. M. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7464–7475 (2023).
56. Xiao, G., Hou, S. & Zhou, H. PCB defect detection algorithm based on cdi-yolo. *Sci. Rep.* **14**, 7351 (2024).
57. Silva, L. H. d. S. *et al.* Automatic optical inspection for defective pcb detection using transfer learning. In *2019 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, 1–6 (IEEE, 2019).

Acknowledgements

This work was partly supported by the grants from the Natural Science Foundation of China (No.: 62372101, 61873337, 62272097).

Author contributions

T.K. contributed to methodology, Data curation, formal analysis and writing of the original draft. Supervision, resources and funds acquisition were performed by J.Z. Validation, Review and editing were done by all authors. Visualization was done by B.M. and M.K. Conceptualization were done by T.K, J.Z and M.K. All authors reviewed and approved the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to J.Z.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

© The Author(s) 2024