Detection Defect in Printed Circuit Boards using Unsupervised Feature Extraction upon Transfer Learning

Volkau Ihar
School of Electrical and Electronic Engineering Nanyang Technological University
Singapore
e-mail: volkau.ihar@ntu.edu.sg

Abdul Mujeeb
School of Electrical and Electronic Engineering Nanyang Technological University
Singapore
e-mail: amujeeb@ntu.edu.sg

Dai Wenting
School of Electrical and Electronic Engineering, Nanyang Technological University
Singapore
e-mail: daiw0004@ e.ntu.edu.sg

Marius Erdt
Fraunhofer Research Center
Nanyang Technological University
Singapore
e-mail: marius.erdt@fraunhofer.sg

Alexei Sourin
School of Computer Science and Engineering
Nanyang Technological University
Singapore
e-mail: assourin@ntu.edu.g

***Abstract*— Automatic optical inspection for manufacturing traditionally was based on computer vision. However, there are emerging attempts to do it using deep learning approach. Deep convolutional neural network allows to learn semantic image features which could be used for defect detection in products. In contrast to the existing approaches where supervised or semi-supervised training is done on thousands of images of defects, we investigate whether unsupervised deep learning model for defect detection could be trained with orders of magnitude smaller amount of representative defect-free samples (tenths rather than thousands). This research is motivated by the fact that collection of large amounts of defective samples is difficult and expensive.**

Our model undergoes only one-class training and aims to extract distinctive semantic features from the normal samples in an unsupervised manner. We propose a variant of transfer learning, that consists of combination of unsupervised learning used upon VGG16 with pre-trained on *ImageNet* weight coefficients. To demonstrate a defect detection, we used a set of Printed Circuit Boards (PCBs) with different types of defects - scratch, missing washer/extra hole, abrasion, broken PCB edge. The trained model allows us to make clusters of normal internal representations of features of PCB in high-dimensional feature space, and to localize defective patches in PCB image based on distance from normal clusters. Initial results show that more than 90% of defects were detected.

Keywords-Cyber manufacturing, Industry 4.0, Automatic Optical Inspection, Transfer Learning, Unsupervised learning, Defect Detection, ConvNet

I. INTRODUCTION

Defect detection in automatic manufacturing, in particular, in PCBs, is an essential step in quality control. This topic brought a lot of attention of different researchers for years. A detailed survey for the automated inspection of PCBs (based on 87 references) that covers in detail various applications and algorithms till 1996 could be found in [1]. An updated survey [2] containing 43 references for newly introduced techniques was published in 2014. The most recent survey of PCB defect detection published in 2017 is in [3]. It contains an updated classification of defect detection methods based on 50 references.

In summary, there are several ways to perform the inspection. Manual detection has a lot of shortcomings: it is subjective, tedious and inefficient, and hardly could be quantified. Automatic optical inspection (AOI) based on computer vision is widely used instead, and the main directions in AOI for PCB, according to [4], are (a) reference approach (comparison with a template), (b) non-reference-approach (extraction of constituent elements without comparison to template), and (c) hybrid methods that combine both aforementioned. Later in [2], other methods were added to the hybrid approach including machine learning. Learning-based models, e.g., [5, 6, 7], were utilized for defect detection: for example, method from [5] uses speeded-up robust features (SURF), then it learns the fault pattern and calculates probabilities and random forest. Densities of the features are estimated by using weighted kernel density estimation, that gives a localization of the defect.

The paper is organized as follows: Section II is the overview of the relevant works and studies done in the area of AOI-based defect detection using machine learning techniques. In Section III, we pose a problem, describe the used dataset and propose an architecture for a model that would detect and localize defects in PCB. The implementation details and results are provided in Section IV. In Section V, we discuss and analyze the experimental results. We outline further plans and conclude the paper in Sections VI and VII.

II. Relevant Studies and our work

One of the earliest publications that suggested to use neural network for defect detection in PCB is, likely, [8]. They used binary images and binary morphological image processing along with Learning Vector Quantization neural network. To train the network, thousands of defective patterns were used. On the test dataset, the neural network showed overall correct classification result of 97.55%.

Deep Learning brought new opportunities to conduct AOI in manufacturing using different from computer vision methods approach. A priority was given to *supervised* learning methods which however require large amounts of labeled defective samples for model training. E.g., the supervised learning method was used in [9] to analyze PCBs for defects. The number of defects to be identified in PCB can vary. For example, 7 types of defects were listed in [4], and in 14 types in [10]. In [9], 6 classes were used: 5 defects (copper, short, open spur, mouse bite) plus normal class. As the number of real defects in their dataset was rather small, the dataset was augmented by adding artificial defects on the bare PCB. It is reported almost 100% of accuracy, however this model was mostly trained on artificial rendered images and it would be problematic to generalize it to real images. Among industrial players, VIDI Systems AG (bought by Cognex in 2017) provides AOI and offers to detect defects by supervised learning. They also used unsupervised model based on autoencoder [11]. This model for automated fault detection was based on Deep Belief Networks, and trained only with good samples.

In some industries, it is difficult to get faulty samples of all different types of defects for purposes of supervised learning, as the defect samples are extremely rare. Even if some of them are obtained, there additionally appears a serious problem of class representative imbalance since, firstly, there are no samples of every possible fault, and, secondly, the faulty case could not have a lot of manifestations, and it is not enough for training.

One of the ways to overcome a problem of insufficient presence of defective samples is to consider a problem as a one-class classification, when the whole set of samples is divided into 2 parts: normal and all the types of abnormal ones. In work [12], authors used a set of labeled samples (400 non-defective and 300 defect images) to train in a supervised manner a deep convolutional neural network with a goal to map the non-defective products into a hypersphere in a high dimensional feature space, while the defect images had to be mapped outside this hypersphere.

The approach where really only one class of normal samples (without defective ones at all) is used can be found in [13]. The authors proposed an unsupervised learning-based end-to-end approach for detection and localization of fabric defects. Their model could be trained with a small amount of defect-free samples and could address various types of textile fabrics and detect various unseen before defects. A convolutional denoising autoencoder is used for modeling the distribution of defect-free fabric patches in the patch domain. The autoencoder (AE) tries to reconstruct the image as close as possible by building an intermediate internal representation (feature vector with a relatively small number of parameters, also called a signature). It is assumed that (a) small discrepancies between input and reconstructed images are the tolerable variations, (b) if AE has not seen the anomalies during training, it would reconstruct them with a high reconstruction error, hence by measuring the residual between the response and the original input the defect decision making could be done, and the location of the defect (where a large reconstruction error occurs) could be predicted.

In [14], it was proposed not to use the reconstruction error of the AE but to consider a set of signatures. Normal class samples form a cluster of normal signatures. A new sample is considered as a defect if the distance to the cluster is relatively high or a normal one otherwise. The built deep AE model had a small number of parameters (around 25,000), 6 convolutional layers, and was custom-based for the case of solder defect detection. The accuracy of the model was around 90%.

In our work, we follow the concept from [14], motivated that it uses all the information from the small amount of training normal data twice: for training and to make a cluster of normal signatures for better decision making. However, in our approach, instead of building the autoencoder we employ combined unsupervised and transfer learning approaches to create signatures. This is done to generalize the approach to be used in different domain areas.

III. Problem posing and methodology

The goal of our work is to investigate whether unsupervised deep learning model for defect detection could be trained with a relatively small amount of representative defect-free samples (tenths rather than thousands). The usage of a small number of normal samples for training could be considered as a challenge to extract as much information as possible from a limited volume of input data.

We assume that collection of large amounts of defective samples is difficult and impractical (say, due to rare occurrence of defects). As it is unfeasible to acquire a comprehensive set of the possible defects, so the supervised learning approach is not an option. Even if a small number of defective samples is available, the quality of a supervised learning classification will be heavily affected by an unbalanced representation of normal and abnormal data.

The model proposed in our work undergoes only one-class training and aims to extract distinctive semantic features from the normal samples in an unsupervised manner. The desirable requirements to the architecture of a neural network are: (a) it should be powerful enough to

extract a comprehensive description (as multidimensional feature vector/signature) for normal samples; it is assumed, that for unseen defective samples the model will generate signatures which are distinguishable from normal ones, so normal and defective cases could be separable in this feature space, and (b) the proposed architecture should provide flexibility to be adapted later to different domain areas.

We would like to emphasize that for PCB defect analysis AOI decides the compliance with design of PCB. The principal difference between image classification and technical defect analysis is in two different question to answer: "Does it look like a PCB?" or (our case) "Does it comply to the defect-less PCB design? If not, provide the location of the discrepancy".

A. *Dataset Description*

For the purpose of validation of our approach on a non-artificially generated bare PCB dataset with different defects (and taking into account the absence of such publicly available dataset), we created our own sets. The first one (which we name PCB-G), is a set of greyscale images of PCBs (all have the same design) acquired with Cognex In-Sight 7801 camera setup (resolution 1280x1024) on industrial manufacturing line. Dataset contains 48 normal images and 200 images with 4 types of defect (scratch, abrasion, missing washer /missing hole/extra hole, broken edge of PCB). The second set, PCB-1, with color PCB images captured by Point Grey FlyCapture camera setup on the same manufacturing line. Resolution is 1288x964, and dataset contains 78 normal color images of PCBs (with equal number of green, blue and black color cases of background), and 155 with the same types of defects as for PCB-G. All images were aligned by SIFT algorithm and external parts of PCB images were cut off. Fig. 1 contains an example of defect-free bare PCB from PCB-G dataset. Each PCB image was divided by 11×6 patches, and each patch has size 100 by 100 pixels. We will assign each patch a class number K (K = 1:66) corresponding to its geometric location on the PCB. All the patches of class K have the same design, but these images could have variations and may not be the precise copy of each other. Variations could include (and not be limited to) lines displacement, locations of the element placing and their alignment, etc. The analysis of a new PCB will consist of analysis of each of 66 patches for compliance with normal design.

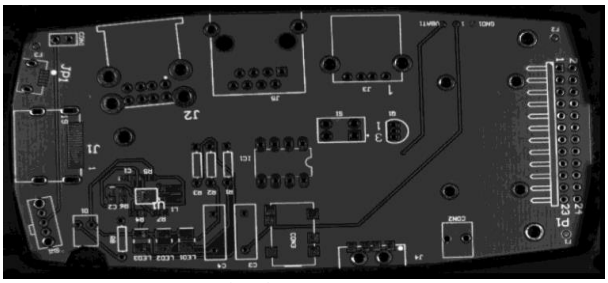

Figure 1. Defect free Printed Circuit Board.

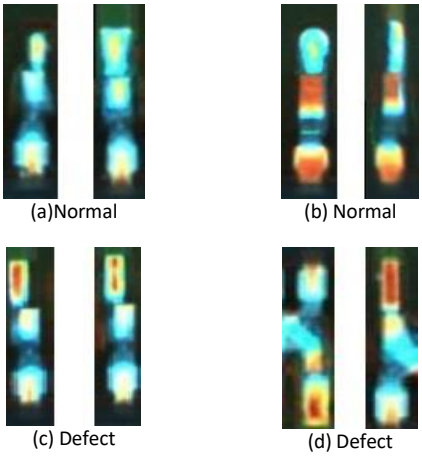

Figure 2. Normal and Defective solders from DS dataset

To prove the flexibility of the approach proposed, we test the algorithm at a dataset of solder images (hereafter named dataset DS) provided by IC manufacturer. This dataset contains two different types of normal shapes and two different defect shapes (Fig. 2); image size is 30×120 pixels. Training set contains 1240 normal samples (682 samples of the first normal shape, and 558 of the second); testing set contains 1814 defective and 2251 normal samples.

B. *The Model Architecture*

We pose the problem as anomaly detection problem without collection of large amounts of failure data, hence the model should be able to extract a representative set of features from diverse images. Transfer learning is a commonly used way to extract features and receive a customized data representation. We consider to use transfer learning based on VGG16 [15]. This deep neural network model has 16 layers (13 convolutional layers and three fully-connected, see Fig. 3) and was trained to classify images into 1000 categories. Training was done on more than 1 million images, and the model along with network weights pre-trained on *ImageNet* are available to be used for transfer learning. There are different approaches to adopt this feature extractor for a custom image dataset: sometimes only the last fully-connected layer (FC) is removed, sometimes the beginning layers (the corresponding coefficients) of VGG16 are frozen and other layers are re-trained using the custom dataset. Usually, it is used for a supervised transfer learning, and the number of neurons in the last FC corresponds to the number of classes to train the model to classify. The original feature vector at fc2 layer of VGG16 (Fig. 3) has 4,096 dimensions, and the output of convolutional part (at flatten layer) has 25,088 elements.

Our model architecture consists of two parts. We use frozen convolutional layers of the pre-trained VGG16 as the first part for initial feature extraction. The second part of architecture is based on approach presented in [16] that provides unsupervised representation learning based on

simple geometric transformations. We expect that it will help to use a rich data representation by transfer learning approach, and to fine-tune this representation to the training data without building a special custom-based model.

The model *RotNet* from [16] allows for extracting complex semantic features from the images without manual labelling data. The authors suggested to use convolution network (*ConvNet*) to recognize 2D-rotations applied to the input data. For example, the input image could be rotated by, 90, 180 and 270 degrees, all in all 4 classes of rotations, and the *ConvNet* learns how to extract features from the rotated input image to recognize geometric transformations and to match the extracted features to the number of the class of rotation. Authors also offered to use as an option a set of rotations with a 45° step. It is the way to convert an unsupervised learning process into a (semi-)supervised, when the image representation is learnt by a supervised way through introduction the classes of rotations [16]. The image understanding incorporated into this rotational *ConvNet* helps to extract the features essential for the image dataset under analysis by using "surrogate supervision signals" [16].

After several experiments, we came up with the following modification of VGG16 (Fig. 4) which proved to work successfully in our task. We froze all the layers in VGG16 starting from the input layer until the flatten layer, and then removed the last fc1 and fc2 layers and added two dense layers—FC1 with 1000 neurons and FC2—with 66 neurons, followed by SoftMax layer.

The frozen layers output parameters are flattened and serve as the input to FC1. We arbitrarily decided to use 1,000 elements in FC1 as our PCB images are not so varied by its nature in comparison with *ImageNet* images of 1,000 classes. The FC2 length corresponds to the number of classes of patches on PCB. The training of this model is done on patches of normal PCBs (66 patches per PCB). We tried several ways to organize the training process and to force our model to learn semantic image features from patches. We will further discuss these attempts in the Discussion section.

Firstly (Experiment A), we fed the patches without any rotation to the described configuration of the network, and trained the network to match the patch to its location on PCB (to match to the correct number of class *K*, that is the output of the SoftMax layer). The loss function used is categorical cross-entropy.

Secondly (Experiment B), we fed the patches randomly rotated by angles 0, 90, 180 and 270 degrees to the network and again trained the model to match the patch (potentially rotated) to its class *K*.

Thirdly (Experiment C), using Experiment A setup we froze all the layers and weights of the original VGG16 up to block4_pool (see Fig. 3), and retrained the last convolutional block of VGG16 (block_5) along with FC1 and FC2 using the *RotNet* approach. And lastly (Experiment D), we used the feature vector (the output of fc1 layer of the original VGG16) without any training by the *RotNet*.

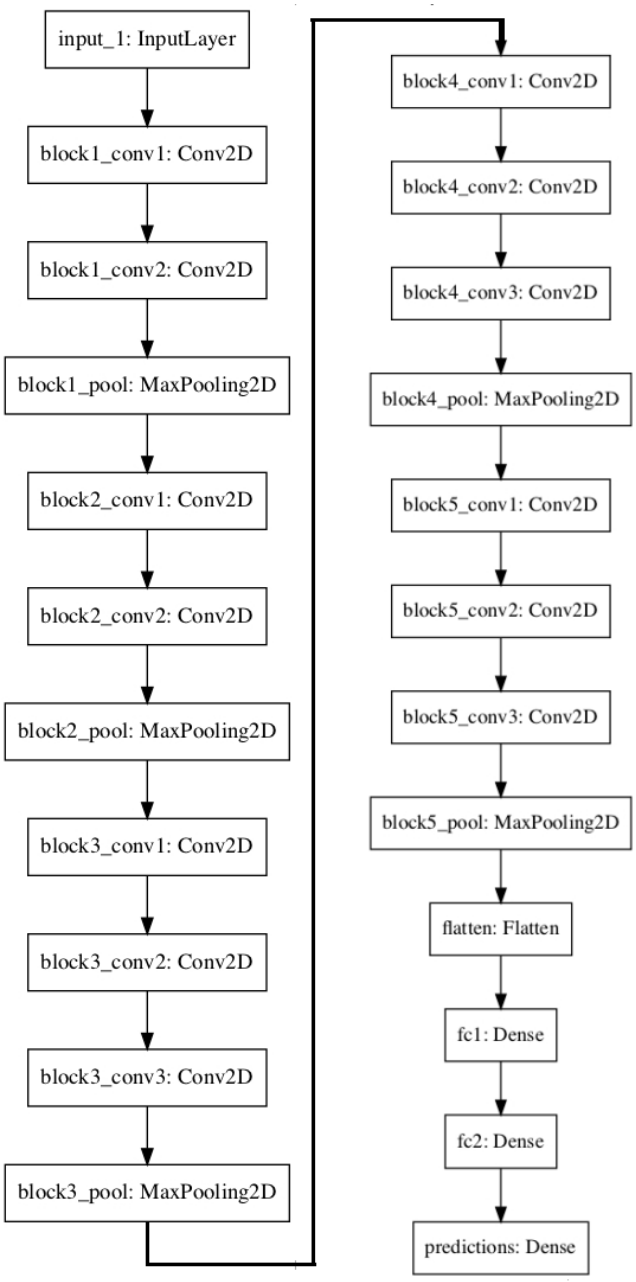

Figure 3. VGG16 Model.

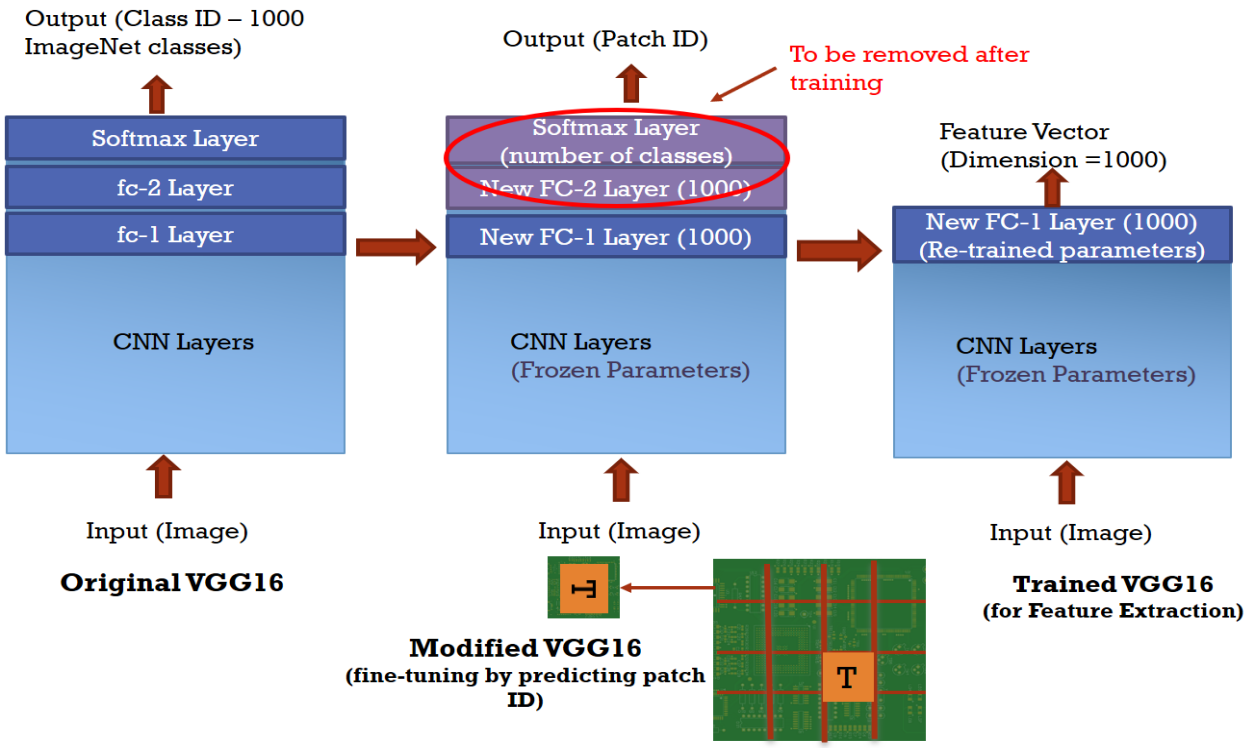

Figure 4. Modified VGG16 deep network.

After training on the normal patches, we cut out the last layers, FC2 and the SoftMax, and used as the output feature vector of our model the output of the layer FC1 (or fc1 for Experiment D). We did not need the output which finally matches the input image to the output class number. We needed only the data representation before the last dense classification layer, i.e. the feature vector that contains the extracted information about the objects and their semantic parts in the analyzed image. We assumed that this feature vector contains essential information about input image structures which could help to distinguish a patch with a normal design from a defective one.

For a normal training patches of the class K (K=1:66 in our case), we calculate a set of normal signatures for this class K, containing multidimensional feature vectors. When a patch of class K of the testing image is analyzed, first the feature vector of the patch is calculated, and then the distance to the cluster of normal representation for this class is found. If this distance is less than some threshold (currently this threshold is global and is set manually), the patch is considered as a normal, otherwise is a defective. This check is done for all the patches of the analyzed PCB image. The location of a defective patch is marked on PCB as a detected defect.

For dataset DS, we used architecture of Experiments A, C, D with K=1:2, corresponding to the number of normal classes of solders. Same as before, only normal images were used for training.

To summarize, Fig. 5 illustrates the overall pipeline for the defect detection process. During the initial stage of AOI, the image is captured using a high quality camera, then it undergoes a pre-processing stage (e.g. noise filtering, background removal, alignment, etc.) to improve the quality for further processing. The defect detection method is based on usage of deep learning model. This model first trained to extracts specific features, and later these features are used for making a decision about the presence of a defect.

IV. RESULTS

The code was implemented in Python using TensorFlow and Keras frameworks. We found that Experiment A setup provides the best accuracy for defect detection for our dataset. The example of resulting output after PCB processing could be found in Fig. 6.

The accuracy of defect detection is provided in Table 1. Testing was done on 60 randomly chosen PCBs. Record "X/Y" in Table 1 means that out of Y defects presented in the test dataset, X were identified. "False Alarm" denotes that not a defective patch was marked as a defect, and the value means the average number of incorrectly marked patches per PCB, e.g. value 0.5 means that 2 PCBs (132 patches together) have 1 incorrectly marked patch in average.

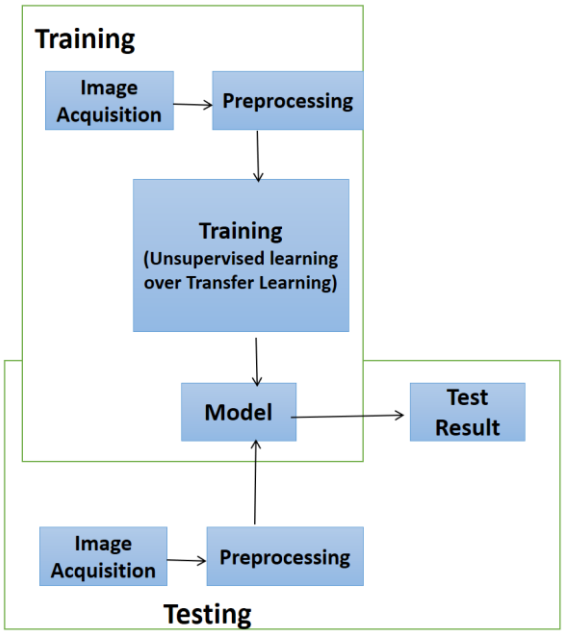

Figure 5. Pipeline for model.

The share of PCBs with black solder mask in false alarms is 70% in PCB-1 and 60% in PCB-G. It shows that images with hardly distinguishable features are prone to be evaluated incorrect way.

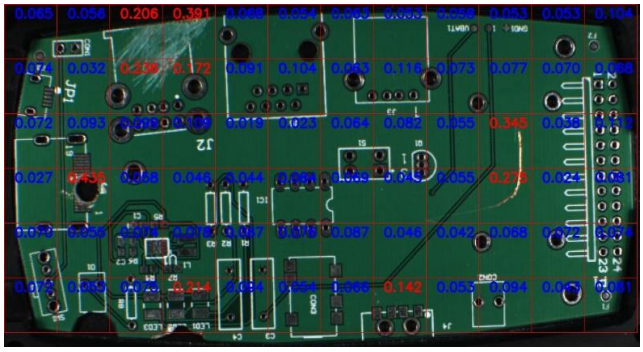

Figure 6. Red number areas are identified as defective. Blue marks are for normal patches. The numerical value is a "distance" from a normal dataset. If it's above the threshold (0.13), the patch is considered as a defect. All 4 types of defects were identified.

TABLE I. ACCURACY OF DEFECT DETECTION ON PCBS.

Defect	**Accuracy (PCB-G)**	**Accuracy (PCB-1)**
Scratch	26/27	34/36
Abrasion	39/39	30/30
Missing washer/hole	7/20	10/43
Broken edge	32/33	32/34
False Alarm	0.5	0.85

The lowest defect detection rate was obtained for category "Missing washer/hole". In this category all incorrectly located holes were identified. Situations with washer presenting and washer missing sometimes hardly distinguishable even by human eye, and training of the model using tens of cases is not enough to extract information to identify such a defect. Other reason also could be considered as an explanation for this low rate. We used a global threshold for all 66 classes. We could assume that if the cluster of normal signatures for some class *K* has smaller diameter (like for classes containing washer elements), the threshold to decide defectiveness also should be adaptive and smaller.

For dataset DS: Experiment A gave sensitivity 93% (correctly identified defects) and specificity 91% (correctly identified normal cases), Experiment C training failed, and Experiment D gave 91% and 89% respectively.

We conducted the same Experiments A and B using ResNet50 network for transfer learning. The results are similar to those with usage of VGG16. Current findings on a different model architecture are the preliminary ones and the analysis will be continued in future.

V. ANALYSIS

Technical system has a pre-defined design and this design information is essential for analysis of defect detections. Unsupervised learning should use some information to extract knowledge from unlabeled data. Transfer learning could be utilized to make use of knowledge learnt in some different but related area. We decided to use VGG16 for feature extraction, and tried several ways (Experiments A-D) to use it alone with the *RotNet* approach to make network to learn about salient features from our data.

The assumptions are:

- Such training will extract generic image features from the domain area elements/patches without any domain-specific knowledge by using features learnt from *ImageNet*. As an analogy, we could consider a space of features extracted from *ImageNet*, but we try to find a subspace of these features that represents or approximates the features from our image space;
- Two similar images will be mapped to close-by feature vectors.
- Image containing defect will have a signature farther away from the cluster of normal representations than a defect-free sample.

Experiment A (no rotation) provided the same result as Experiment B (rotation).

Data augmentation should force the model to extract the features we consider useful and important for object description and analysis, and the images generated should have something in common with those used during the training/testing stage phase, i.e. be meaningful. In Experiment B, when the data augmentation by rotation was done, the network didn't learn more about the features from these rotated images, likely due to the fact that network would never see the rotated images at defect detection stage.

Experiment C, where a part of VGG16 (last convolutional block_5) was retrained using a small dataset of normal samples, showed the worst result. Taking into account the number of parameters of block 5 to train (around 7 million) and number of training samples (less than 100), the back-propagation stage could not find a global optimum. The network with thousands/millions of trainable parameters could not reach optimal state by being trained at only tens of training samples. It does not learn anything as a small number of input parameters cannot be used for optimization of huge number of network parameters.

Experiment D was conducted without any retraining of the pre-trained VGG16 model, so the output feature vector does not contain the features specific for the current dataset. It explains the setup's lower potential for defect detection.

Usually the concern of learning process is not to overfit the model, and model should generalize from data, not "memorize" it in order to provide a good predictive performance on unseen data. For technical data with a predefined design the situation might be somehow different, it could be beneficial to make the model to remember the technical patterns and for defect detection to compare the seen patterns with a new unseen sample. It makes possible to train the network by using a small amount of training samples. The training should cause the model to capture adequately the underlying relevant structures in technical data. Ideally, everything not designed should be irrelevant.

Due to usage of transfer learning with a huge data representative power, we escape an underfitting case, when a machine learning model cannot adequately capture the underlying structure of the data and as a result the model does not fit the data enough and miss important features.

We could assume that adding more normal training samples would facilitate building of more accurate model with better separation of previously seen normal samples from never seen defects.

If a very limited number of normal images (e.g., 10 only) is used for the training stage, the accuracy of the model will be highly dependable on how representative (i.e. how typical of a class or group) the used training images are. The results of defect inspection on a new image would highly rely on similarity/dissimilarity its features to those ones previously seen by the model.

VI. CONCLUSION AND FUTURE WORK

We convert task to self-supervised learning, where labels for un-annotated data are obtained by some heuristic approach using properties of images under consideration. The labels are assigned using the nature of the data. We cut image into geometric patches and each geometric location of a patch was associated with a separate class number. From training data, the model learns some salient features of spatial layout for each patch. For each patch class, the learnt data representations form a cluster of normal signatures of this class samples. This clustering is used later to make a decision on defectiveness of a test patch.

It was mentioned, we used a small set of defect-free images only. The proposed architecture forces the model to learn the underlying structure of the presented data. One of the dangers of the small dataset is, if input data offer too less information for recognition, then the features we would have in mind as important might be not distinguishable enough, and the model might not "see" the features it has to extract the unsupervised way according to our (human) domain knowledge.

It is a very interesting open question: how much information is enough to train a network to make it learn semantic features to detect and distinguish objects.

We proposed our way to use transfer learning to extract features and to use the *RotNet* to fine-tune the model to find a subspace of feature space that is relevant and descriptive for the dataset provided.

The developed neuro-network architecture could be adjusted to analyze other technical images. The requirements to these images are: they should have the same structure or composition. This allows distinguishing normal from defects. The neuro-network model would compare a new sample/input with a consistent structure already known and check the agreement.

In future, we are going to conduct more experiments with other pre-trained network, more extensively analyze ResNet50, and compare the results with VGG16. Also, we would like to use an adaptive threshold for defect detection, and to calculate it an automatic way based on distribution of the scores for normal patches.

VII. SUMMARY

The proposed deep learning network could be trained using a relatively small dataset of representative normal

data, and to detect and to locate previously unseen defects under the condition that the defects look different from the normal elements used for training. The architecture used could also be applied to conduct defect detection in other domain areas, e.g. metal surface analysis. However, it might not work as well in application areas where tiny texture variations are the defects, because small number of training samples does not allow the network to learn about the valid texture patterns variability.

The method presented possesses novelty, as to the best of our knowledge there is no mentioning of attempts to use transfer learning approach for unsupervised learning in defect detection.

VIII. ACKNOWLEDGMENTS

This work was conducted within the Delta-NTU Corporate Lab for Cyber-Physical Systems with funding support from Delta Electronics Inc. and the National Research Foundation (NRF) Singapore under the Corp Lab@University Scheme.

REFERENCES

List and number all bibliographical references in 9-point Times, single-

[1] F. Moganti, F. Ercal, C. H. Dagli and S. Tsunekawa, "Automatic PCB inspection algorithms: A survey," Computer Vision and Image Understanding, vol. 63, no. 2, pp. 287-313 , 1996.

[2] E. M. Taha, E. Emary and K. Moustafa, "Automatic Optical Inspection for PCB Manufacturing: A survey," International Journal of Scientific & Engineering Research, vol. 5, no. 7, pp. 1095-1102, 2014.

[3] D. B. Anitha and M. Rao, "A survey on defect detection in bare PCB and assembled PCB using image processing techniques," in International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), 2017.

[4] W.-Y. Wu, M.-J. J. Wang and C.-M. Liu, "Automated inspection of printed circuit boards through machine vision," Comput. Ind., vol. 28, no. 2 (May 1996), pp. 103-111, 1996.

[5] H. Y. Eun, H. P. Seung, P. Cheong-Sool and B. Jun-Geol, "Feature-Learning-Based Printed Circuit Board Inspection via Speeded-Up Robust Features and Random Forest," Applied Sciences, vol. 8, no. 6, 2018.

[6] S. Huang and Y. Pan, "Automated visual inspection in the semiconductor industry: A survey.," Comput. Ind., vol. 66, p. 1–10, 2015.

[7] H. Kim and S. Yoo, "Defect detection using feature point matching for non-repetitive patterned images," Pattern Anal. Appl., vol. 17, p. 415–429, 2014 .

[8] R. Heriansyah, S. A. R. Al-attas and M. Munim, "Neural Network Paradigm for Classification of Defects," Jurnal Teknologi, vol. 39, no. 1, p. 87–103, 2003.

[9] C. Zhang, W. Shi, X. Li, H. Zhang and H. Liu, "Improved bare PCB defect detection approach based on deep feature learning," The Journal of Engineering, vol. 16, pp. 1415-1420, 2018.

[10] V. Chaudhary, I. R. Dave and K. P. Upla, "Automatic visual inspection of printed circuit board for defect detection and classification," in Proceedings of the 2017 International Conference on Wireless Communications, Signal Processing and Networking, WiSPNET, 2018.

[11] J. Sun, R. Wyss, A. Steinecker and P. Glocker, "Automated fault detection using deep belief networks for the quality inspection of electromotors," Technisches Messen, vol. 81, no. 5, pp. 255 - 263, 2014.

[12] M. Zhang, J. Wu, H. Lin, P. Yuan and Y. Song, "The Application of One-Class Classifier Based on CNN in Image Defect Detection," Procedia Computer Science, vol. 114, pp. 341-348, 2017.

[13] S. Mei, Y. Wang and G. Wen, "Automatic Fabric Defect Detection with a Multi-Scale Convolutional Denoising Autoencoder Network Model," Sensors, vol. 18, no. 4, p. 1064, 2018.

[14] A. Mujeeb, W. Dai, M. Erdt and A. Sourin, "Unsupervised surface defect detection using deep autoencoders and data augmentation," in International Conference on Cyberworlds, Singapore, 2018.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv 1409.1556, 2014.

[16] S. Gidaris, P. Singh and N. Komodakis, "Unsupervised Representation Learning by Predict-ing Image Rotations.," in ICLR 2018, Vancouver, 2018.