One class based feature learning approach for defect detection using deep autoencoders

Abdul Mujeeb; Dai, Wenting; Erdt, Marius; Sourin, Alexei

2019

Abdul Mujeeb, Dai, W., Erdt, M., & Sourin, A. (2019). One class based feature learning approach for defect detection using deep autoencoders. *Advanced Engineering Informatics*, 42, 100933-. doi:10.1016/j.aei.2019.100933

<https://hdl.handle.net/10356/137979>

<https://doi.org/10.1016/j.aei.2019.100933>

© 2019 Elsevier. All rights reserved. This paper was published in *Advanced Engineering Informatics* and is made available with permission of Elsevier.

Downloaded on 26 Apr 2023 23:27:43 SGT

One Class Based Feature Learning Approach for Defect Detection Using Deep Autoencoders

Abdul Mujeeb
School of Electrical and
Electronic Engineering, Nanyang
Technological University,
Singapore
amujeeb@ntu.edu.sg

Wenting Dai
School of Computer Science and
Engineering, Nanyang
Technological University,
Singapore
daiw0004@e.ntu.edu.sg

Marius Erdt
Fraunhofer Research Center,
Nanyang Technological
University,
Singapore
marius.erdt@fraunhofer.sg

Alexei Sourin
School of Computer Science and
Engineering, Nanyang
Technological University,
Singapore
assourin@ntu.edu.sg

Abstract—Detecting defects is an integral part of any manufacturing process. Most works still utilize traditional image processing algorithms to detect defects owing to the complexity and variety of products and manufacturing environments. In this paper, we propose an approach based on deep learning which uses autoencoders for extraction of discriminative features. It can detect different defects without using any defect samples during training. This method, where samples of only one class (i.e. defect-free samples) are available for training, is called One Class Classification (OCC). This OCC method can also be used for training a neural network when only one golden sample is available by generating many copies of the reference image by data augmentation. The trained model is then able to generate a descriptor—a unique feature vector of an input image. A test image captured by an Automatic Optical Inspection (AOI) camera is sent to the trained model to generate a test descriptor, which is compared with a reference descriptor to obtain a similarity score. After comparing the results of this method with a popular traditional similarity matching method SIFT, we find that in the most cases this approach is more effective and more flexible than the traditional image processing-based methods, and it can be used to detect different types of defects with minimum customization.

Keywords—Automatic Optical Inspection; Deep Learning; Unsupervised Learning; One Class Classification; Autoencoders

I. INTRODUCTION

In a manufacturing environment, different defects may occur at various stages, such as misaligned or missing components, broken edges, cracks, holes, scratches, etc. To find these defects in various products, either human experts perform visual inspection, or Automatic Optical Inspection (AOI) may be performed using a computer vision algorithm. Since manual defect detection is very subjective, slow, and inconsistent, it does not satisfy the current manufacturing requirements. The focus of this paper is at AOI which is becoming more popular with the advances in hardware and software. From the image processing perspective, manufacturing defects can be viewed as deviations in a test image from a reference image. Presently, most defect detection systems rely on traditional image processing techniques, which make use of manually crafted features

chosen by experts with reasonable technical skills and domain knowledge. Machine learning methods are alternatives to traditional image processing, and they are gaining popularity very rapidly due to their superior performance in various domains. A machine learning approach is essentially different from a traditional image processing approach, because in the former the extracted features are data specific, but in the latter features are extracted algorithmically and are not data specific.

Most machine learning methods used for defect detection are based on supervised learning technique, which typically requires a large number of labelled samples for training. Supervised learning is not feasible in many AOI applications if labelled defect samples are not available for training. Unsupervised learning is another class of machine learning algorithms which does not require large number of labelled samples for training. We propose such unsupervised algorithm in this paper where defect detection relies on the dissimilarity between a test image from the set of reference images used in the training, hence any type of defects (i.e. dissimilarity from the reference) could be detected. In order to prove that the method is generic enough to detect various types of defects on different products, we have used three different types of defects, viz., solder joint defects, scratches on metal surface, and misplaced components on a Printed Circuit Board (PCB).

The paper is organized as follows: Section II provides an overview of the existing work done in the area of defect detection using conventional image processing and machine learning techniques. In Section III, the proposed unsupervised method for defect detection is explained. Section IV contains the implementation details, while in Section V, the analysis of the experimental results is presented. Section VI highlights the integration plans, and the paper is concluded in Section VII.

II. BACKGROUND

AOI is an algorithmic method for defect detection in manufacturing products, e.g., PCBs, metal surface, etc. A wide range of algorithms exist due to varied nature of

products and defects. There are several steps performed on a test image for AOI based defect detection, as shown in Fig. 1. First, using a high quality camera an image is captured, and then it undergoes a pre-processing stage to become more suitable for defect detection algorithm. If needed, Region of Interest (ROI) extraction is performed before applying defect detection algorithm, e.g., solder joints are extracted from a given PCB image. Finally, the defect detection algorithm is applied, which is usually the most complex stage in the flowchart.

Figure 1. AOI based defect detection flowchart.

In the defect detection stage, pixels are converted into feature vectors, which are the heart of a defect detection algorithm. In the following sections, an overview of the *traditional image processing* algorithms and *machine learning* based algorithms is presented.

A. Traditional Image Processing Techniques

A primitive image matching technique is to compare the pixel intensities of the two images directly by using some correlation measure. This can be a good method for the images generated using software, but it is extremely sensitive to even slightest noise, alignment, intensity and other transformations due to image capturing conditions, and therefore it is rarely used. A more common approach is to compare feature vectors of two images, which are generated by applying some algorithm on the images. Many feature extraction techniques have been developed, and we are performing a survey of such methods in the following paragraphs.

In [1], the AOI methods are classified as *statistical*, *structural*, *filter-based*, and *model-based*. *Statistical methods* are used to compute the spatial distribution of pixels of the image, and most of the methods are based on first order statistical computations such as mean, variance, range, and other histogram-based measurements, along with the higher order statistics. In [2], Weibull distribution is used on local edges for the detection of texture anomalies. The authors of [3] presented an algorithm for image matching based on *structural similarity* using statistical parameters. *Filter-based methods* employ filter banks, such as Fourier transform, Gabor filters, and gradient filters, to generate feature vectors which consist of the filter responses. In [4], a Phase-Only Correlation based image matching technique was developed, which uses the phase information in the Discrete Fourier Transforms for computing similarity. If the knowledge of the underlying process that generates the texture pattern is available, and it can be defined by a stochastic model, then the *model-based approaches* can yield very accurate results in

texture defect detection. Texture analysis techniques have been proposed in various applications such as, textile [5, 6], metal surfaces [7], biomedical [8, 9, 10], food product inspection [11], wood [12], etc. Speed Up Robust Features [13], and Scale-Invariant-Feature-Transform (SIFT) [14, 15] are two widely used techniques for feature extraction, which have demonstrated good performance in many pattern matching applications. Soldering defects in PCB is a highly specialized domain, which has its own set of defect detection techniques, and most of the research works for finding such defects employ traditional image processing techniques [16, 17, 18]. Work [19] used the HAAR wavelet transform to detect defects in bare PCBs (with no parts mounted on them). The reference and the test PCBs features are extracted by applying the HAAR transform, and the difference operation was done in the wavelet domain. Paper [20] used several image processing operations, such as logical XOR, image subtraction, to classify 14 most common defects in bare PCBs.

Work [21] proposed a two-step method which does not require alignment of test and template images. In the first step, a Phase Only Fourier Transform is used for saliency detection (i.e. segmentation of regions), and in the second step, discrepancies are evaluated between salient regions in the test image with corresponding regions in the template images. This method is useful when selected regions on a board (e.g., dice on semiconductor) are to be inspected.

B. Machine Learning (ML) Techniques

Selecting appropriate image features for an application is a non-trivial task. Since machine learning methods can extract feature automatically from data, it has become an attractive alternative to traditional algorithms. Machine learning techniques are broadly categorized as Supervised and Unsupervised learning methods. In the former, large number of labeled samples for all the classes are needed to learn the patterns from the data, whereas, the latter method does not depend on labeled data for training. Many studies have been conducted on the effectiveness of machine learning methods for defect detection. In [22], a neural network-based feature extraction technique was proposed, which works on any textured images. A neural network was used in [23] for steel defect classification using supervised learning. A neural network-based inspection algorithm was applied on DAGM dataset in [24], which outperformed the defect detection techniques used till then. In [25], a railway track fastener defect detection algorithm was implemented using CNNs for positioning and recognition of defects simultaneously, and it outperformed traditional methods in speed as well as accuracy.

A practical limitation in the use of machine learning methods for AOI is unavailability of training samples of defects. The applications where only one class (i.e. normal)

samples are available, and only few to none defect samples are available for training, are classified as *One Class Classification* (OCC) problems. In some cases, even the availability of defect free samples is limited to one or very few samples, and such problems are called *One-shot* learning algorithms. Various methods have been used for OCC and One-shot learning. Transfer learning is a method where a network is trained using samples from a different domain, and then the network is able to extract good features from the target domain. The success of transfer learning largely depends on how similar the training domain is to the application domain. For example, a transfer learning method was used by the authors of [26], which shows how image representation learned with CNN on a large scale annotated dataset can be transferred to another task of visual recognition with limited amount of training data. The method reused layers trained on the ImageNet dataset to compute mid-level image representation for images in the PASCAL VOC 2007 dataset, where 18 out of 20 classes showed much improved accuracy. This option of using a pre-trained network is an attractive option in many applications where number of training samples is low, but a typical pre-trained network (such as VGG16, Resnet50, etc.) have huge number of parameters, typically in the range of tens of millions, which is a big constraint on memory and speed. Furthermore, these systems work best when the last layers are retrained with the small amount of supervised training data. In our case, the defect data is not available, which makes this approach not feasible.

A signature verification system was developed by authors of [27] using transfer learning and Convolutional Siamese Network. The system was first trained with very large number of training samples (i.e. database of signature images), to extract a unique descriptor from only one signature image of a user. In [28] a similar concept of transfer learning and Convolutional Neural Network (CNN) was used for face recognition where only one image per person can be used for face identification. The network is trained using large number of face images, and after the training the model becomes capable of generating a descriptor for any face image. In [29] a transfer learning-based bare-PCB defect detection was introduced, which utilizes a pre-trained model for feature extraction. The experiment was conducted using artificial defect data to learn feature extraction using supervised method, with the assumption that a system trained with artificial data would work thereby reducing the requirement of a large dataset. Our alternative consideration for the neural network architecture was Siamese network, which is a method used when the number of training data is small. It is trained by using two pairs of images called positive and negative pairs. A positive pair consists of two positive (normal) images, and a negative pair contains a positive (normal) and a negative (defected) images. A different optimization function is used in Siamese networks than the common optimization functions used in supervised learning. This optimization function, called

as triplet loss function [30], utilizes the similarity of positive image pair and the difference of negative pair simultaneously. After the Siamese network has been trained, its powerful discriminative features can be used to generalize the predictive power of the network to new data. As we do not have defect images (i.e. negative) at the time of training, we found this method not practical for our application.

Semi-supervised learning is an attractive research direction where unlabeled data is used for training. In [31], the author proposed a method based on semi-supervised learning (SSL) for auto-labelling of PCB solder images using an Active learning algorithm. In [32], an SSL based image classification technique was presented, which can be used with millions of unlabeled data for more accurate training of the model, and it is very good to use when little labelled data is available. Such SSL techniques are not suited to our task because the defect samples are not available at the training stage. We also explored the applications of deep learning in medical image analysis [33], and found that the medical application domain is more similar to defect detection as compared to more common ML based vision algorithms applied on public datasets, such as ImageNet, PASCAL VOC, etc. ML techniques are used in various stages of medical image processing such as segmentation, localization, detection and classification.

C. Problem Definition

The purpose of this research is to explore a possibility of machine learning algorithms in AOI when very little or no defect data is available for training. Thus, we solve a problem *whether it is possible to extract features automatically for the purpose of defect detection where only normal samples are available for training*. Our goal is to find a machine learning-based feature extractor which: 1) is generic enough to be ported across wide variety of defects and products, 2) can be trained by just using normal images, and 3) possibly be able to work when only a single reference image is available as a golden sample.

Inspired by the recent success stories of deep learning in many domains, we implemented such a feature extractor using a deep network. For the case where only one standard reference image (golden sample) is available, multiple copies of the reference image are generated by applying data augmentation techniques, such as rotation and flipping. We are using deep autoencoders, which are neural networks comprising of many hidden layers. A deep learning based autoencoders can be used to generate a compressed representation of an image as described in [34]. After training, the network can generate a feature vector for an input image (of the same class), and, by comparing the feature vectors of a test and a reference image, we can find if there is any defect in the latter one.

III. METHODOLOGY

A. Background and Motivation

This method is developed as a part of an industrial research project aiming at developing techniques for manufacturing defect detection as illustrated in Fig. 2. We are focusing at defects which can be visually inspected (Fig. 3). The target system contains a high-quality camera and a lighting system to captures images which are sent to a computer running AOI algorithm. Our goal is to devise a method, which can work with a wide variety of defects with minimum customization, and is able to run on a typical embedded system with relatively limited resources and processing power. The system should be cost effective, especially for small businesses which are not able to afford expensive AOI systems.

B. Feature Extractor

A typical Image-matching process consists of two steps: a feature extractor algorithm and a feature matching algorithm. The features are extracted from two images to be compared and then are compared using a matching algorithm to generate a similarity score as shown in Fig. 4.

Figure 2. Target AOI system (actual electronic components have been replaced with place-holder images).

Figure 3. Defect examples: (a) scratches, (b) misplaced components, and (c) solder defects (picture modified to protect confidential information).

Figure 4. Image matching algorithm

C. Deep Feature Learning

Selection of features is the most important step in any machine vision applications. As mentioned before, in a supervised solution we need samples in large quantities. The lack of defected samples, as well as need of a generic solution, motivated us to explore the possibility of using a deep learning-based feature extractor. A deep autoencoder trained using only normal images can learn to extract features which are different from features of an image which is not normal (i.e. defected). Furthermore, in One-shot classification cases, multiple copies of the single normal image can be generated using data augmentation techniques which allows for proper training of autoencoder.

To prove this idea, we are proposing to use a deep autoencoder for feature extraction. An autoencoder consists of two serially connected neural networks, called as Encoder and Decoder, as shown in Fig. 5. The encoder learns patterns from an image, and its output layer contains fewer number of nodes than the number of input nodes, which makes the encoder output as the compressed representation of the input image. The decoder aims at reconstructing the input image as its output using this compressed representation. Each output node represents a certain geometric pattern. Work [35] explains the process of feature extraction at various layers through visualizations, which is an empirical method rather than mathematical to provide an insight into the workings of a deep network. It was demonstrated that the process of feature extraction goes from more generic to more specific as the layers become deeper. The initial layer extracts generic features like lines, curves, etc., and each subsequent layer extracts more specific patterns, which are combination of the features from previous layers as shown in Fig 6. The size of feature vector can be controlled by choosing the number of nodes in the encoder output, therefore, an encoder with N number of output nodes generates an N -size feature vector.

Figure 5. Block diagram of an autoencoder.

Figure 6. Feature extractions at various layers.

An autoencoder does not performs well if only few training images are used. Therefore, if number of normal images is not large (e.g., in the case of a single golden image), we have created many copies of the input data through data augmentation.

In the case of only one standard reference image, data augmentation of the reference image is performed by flipping operation (horizontal and vertical) followed by rotations at small angles. After training, the reference and test images are passed through the model to generate a reference descriptor and test descriptor respectively, which are normalized to unity magnitude. Using L2-norm for comparing both vectors, a similarity score between 0 and 1 is generated, which represents identical or maximally different images respectively, and by selecting a proper threshold (between 0 and 1), the presence of a defect can be identified.

D. Effect of Block Size

A larger image is divided into smaller images, and each patch is trained independently of others. This is done to detect defects which are confined to very small regions. The block size is important, because if it is too large, then the defect area will have less contribution in the feature vector, as shown in Fig. 7, and the generated score will not be high enough to cross the detection threshold.

IV. IMPLEMENTATION

An image can be of a smooth (uniform) or a textured background, and depending on the type of background, two slightly separate strategies have been used for feature extraction. as shown in Table I.

TABLE I. TRAINING SAMPLES NEEDED.

Surface Type	Training (Normal)	Training (Defected)	Data Augmentation
Texture	Large	0	No
Uniform	1 or Few	0	Yes

For the uniform background, only one reference image can be used for training, and multiple copies for training are created by data augmentation. For textured background, large number of normal images must be used, so that the training set is diverse enough to include most common texture

patterns. In both cases, we do not require any defect images for training.

A. Data Augmentation

We are performing data augmentation on the reference input image as mentioned above, which is achieved by flipping the reference image(s) horizontally and vertically followed by small angular rotations. In this experiment, the original image and its flipped copies, were rotated by various angles between -180 to $+180$ degrees with step size of three degrees. This way, over four hundred copies were generated to perform training of the encoder.

Figure 7. Effect of defect region selection on detection.

B. Image Preprocessing

Before training, image preprocessing was performed which includes graying of images (if needed). In applications where color information is important, such as solder images, color-to-grayscale conversion was skipped. Data normalization was performed on all images, where range of pixels in each image was converted between 0 and 1.

C. Generating Patches

If the defect regions are very small as compared to the overall image size, it is more effective to work on smaller blocks of an input image. Therefore, large images are converted into smaller image patches by horizontal and vertical grids, as shown in Fig. 8. We are using grid size of 64×64 pixels in this experiment. For example, an image of 640×640 pixels is divided into hundred sub-images of 64×64 pixels. Training for each patch is performed separately, independently of others.

D. Deep Autoencoders

We have implemented a deep autoencoder. The encoder consists of three fully connected layers, each containing 64 nodes. The last layer of the encoder becomes the feature vector. The decoder is a mirrored image of the encoder, and has no part in the feature extraction after the training is complete. All layers of the autoencoder use Rectifier Linear

Unit (Relu) activation function, except the last one which uses Sigmoid activation function. A loss function, called as reconstruction loss function, is used to measure how closely the out image is constructed to the input image. We have chosen binary cross-entropy for reconstruction loss function, which calculates how many bits of information are preserved in the reconstruction data Y as compared to the original data X , and is defined in equation 1.

$$J(x, y) = -\sum_k [x_k \log y_k + (1 - x_k) \log(1 - y_k)] \quad (1)$$

Where x_i is the i th input sample, y_i is the corresponding output sample, and $(i = 0, 1, 2, \dots, N-1)$.

Figure 8. Creating image patches using grids.

For optimization of loss function, ADAM optimizer [36] has been used, which is converging within 1000 iterations.

E. Similarity Matching

The encoder's output layer is a one-dimensional array representing the feature vector. An N -sized feature vector $\mathbf{X}$ can be represented as an array shown in equation 2, where each component is a floating point number representing a certain structure in the image.

$$\mathbf{X} = \{x_1, x_2, x_3, \dots, x_N\} \quad (2)$$

The feature vector is converted to a unit vector by dividing each component i , by the Euclidean length as shown in equation 3.

$$x_i = \frac{x_i}{\|\mathbf{X}\|} \quad (3)$$

where,

$$\|\mathbf{X}\| = \sqrt{x_1^2 + x_2^2 + x_3^2 + \dots + x_N^2} \quad (4)$$

The similarity score is found by using L2-distance between the two feature vectors. For example, the L2 distance between the two feature vectors $\mathbf{X}$ and $\mathbf{Y}$, each of length N , represents similarity, and is computed as shown in equation 5,

$$D = \sqrt{\sum_{i=1}^N (x_i - y_i)^2} \quad (5)$$

The generated score is between 0 (identical) and 1 (maximally different).

F. Training Algorithm

Two slightly different strategies have been used for textured and uniform backgrounds as shown in Fig. 9(a) and 9(b), respectively. Each image is flattened to create a 1-dimensional array, and the training is performed in batch mode using the flattened images.

Figure 9. Training for textured background and smooth background images

G. Test Algorithm

Testing involves two steps. Firstly, feature vectors are generated for test and reference images using the trained model, and secondly, both feature vectors are tested for similarity using L2 distance. Figures 10(a) and 10(b) illustrate the algorithm flowchart for matching the descriptors for textured and uniform backgrounds, respectively. It should be noted that a test image goes through the same preprocessing steps as the training images, such as graying, flattening, and normalization.

(a) With textured background

(b) With uniform background.

Figure 10. Test algorithms: (a) Textured, (b) Smooth background

H. Evaluation

In this experiment, we are using six different types of datasets to ensure the flexibility of the method. Dataset 1 comprises images from our local AOI research setup (Fig. 11). Datasets 2 consists of solder images provided by an IC manufacturer (Fig. 12), and are named as solder dataset-1 (SDS-1), solder dataset-2 (SDS-2), solder dataset-3 (SDS-3) and solder dataset-4 (SDS-4), respectively. They contain two different types of normal shapes and two different defect shapes in various combinations to ensure that the method is flexible. All the solder images were extracted manually from the IC images. Dataset 3 contains surface scratch images from NEU (Fig. 13). The dataset contains only defected images, hence normal images were created by redrawing the scratched areas using an inpainting algorithm [37]. Only normal images were used for training, which were not included in the testing later on. We are using both normal and defected images for testing. Table II shows how the datasets were used for training and evaluation. Datasets 1 and 3 contain larger images which were divided into smaller images of 64×64 pixels. Dataset 2 contains images of smaller size of 30×120 pixels, and therefore were used in the experiment without creating patches.

Figure 11. Dataset 1:
(a) normal, (b) misaligned, (c) missing, (d) misplaced.

Figure 12. Dataset 2 (consists of 4 sub datasets):
(a) SDS-1, (b) SDS-2, (c) SDS-3. (d) SDS-4

Figure 13. Dataset 3: scratch samples from NEU dataset.

TABLE II. IMAGES USED FOR EVALUATION.

Dataset	Training (Normal)	Test (Normal)	Test (Defected)	Size & Type
Internal	1 (×480 augmented copies)	330 patches	330 patches	64×64 (Grayscale)
Solder DS-1	550 images	300 images	300 images	30×120 (RGB)
Solder DS-2	680 images	300 images	300 images	30×120 (RGB)
Solder DS-3	690 images	300 images	300 images	30×120 (RGB)
Solder DS-4	522 images	300 images	300 images	30×120 (RGB)
NEU Scratches	65×9 patches	30×9 patches	300×9 patches	64×64 (Grayscale)

I. Detection Accuracy

True Positive Rate (TPR) is a measure that indicates how many defects are successfully detected, whereas, False Positive Rate (FPR) is a measure of how many normal cases are detected successfully. Correct defect detection is called Sensitivity, while Specificity is the correct detection of defect-free samples, both measuring between 0 and 1:

$$\text{Sensitivity} = \text{TPR}$$

$$\text{Specificity} = 1 - \text{FPR}$$

For a good system, both Sensitivity and Specificity should be high (ideally 1). Both quantities cannot improve at

the same time, so when Sensitivity increases, the Specificity decreases and vice-versa.

As mentioned before, a defect is identified by the matching score against a threshold, hence the accuracy of defect detection depends on the value of the threshold. As shown in Figs. 14 and 15, the matching value is increased significantly in defected blocks, and is lower in normal blocks for datasets 1 and 3, respectively.

Receiver Operating Characteristic (ROC) curves are generated by thresholding the distance between the reference and the test images feature vectors, as shown in Fig. 16 and Fig 17 for autoencoder and SIFT, respectively. For dataset 1, the True Positive Rate (TPR) is over 80%, while the False Positive Rate (FPR) is 16%. TPR and FPR for dataset 2 are 89% and 5%, respectively, and for dataset 3, the TPR and FPR are 83% and 7%, respectively.

Figure 14. Image with a missing component

Figure 15. Images with and without scratches

Additionally, a popular image processing algorithm, SIFT, has been used to compare the performance of our method. When applied to an image, SIFT generates a set of values called key-points. Two images are compared by matching the key-points of both, and the decision is made if a certain number of key-points are matched in both images. For this experiment, we compared a test image with reference image based on the ratio of the matching key-point to the

total key-points. For example, if a test image has 50 key-points, and 25 of them are matched with the key-points in a reference vector, the matching score is considered as 0.5. Figure 18 shows ROC curves based on SIFT matching, where the threshold represents the ratio of matching key-points.

Out of the six datasets used in this study, two solder datasets (SDS-2 and SDS-4) show better performance with SIFT. For other four datasets, our method shows better detection accuracy and the SIFT is practically not usable because a good balance between sensitivity and specificity is not achievable. One important observation is that the threshold selection for each dataset is different, and selecting an optimal threshold was done empirically. In practical situation, since the defect samples are, presumably, not available at training, hence optimal threshold selection would be a gradual process, which will be refined once the system is in operation.

It is worth noting that our objective in this work is to find an automatic and flexible feature extractor. It is possible (as mentioned above in the case of two solder datasets where SIFT performance is marginally better) that a traditional image processing technique may yield the same or even better accuracy, but that particular algorithm may not be suitable for other defects (e.g., misplaced components). The ROC curves of our method show that the technique is flexible, and can be applied for detection of varied types of defects.

V. INTEGRATION PLAN

In the near future, we are planning to incorporate this work in the manufacturing setup, as schematically illustrated in Fig. 18. An embedded system with memory and speed constraints is the target AOI system. The algorithm is fast enough to run on a common CPU and does not require GPU. The training, however, needs to be conducted offline on a more powerful computer preferably containing a GPU. The trained model contains a less than few hundred thousand parameters, which can fit into modern embedded systems. Additional image preprocessing steps, as mentioned in Fig. 1, will also be implemented at the deployment stage.

(a) Dataset-1 (At Thresh = 0.15, TPR = 0.81, FPR = 0.16)

(a) Dataset-1 (At threshold=0.4, TPR=0.80, FPR=0.54)

(b) Dataset-2 (Solder defects, SDS1 to SDS4)

(b) Dataset-2 (Solder defects, SDS1 to SDS4)

(c) Dataset-3 (At Thresh = 0.18, TPR = 0.81, FPR = 0.03).

(c) Dataset-3 (At threshold=0.7, TPR=0.75, FPR=0.19)

Figure 16. ROC curves using autoencoder. (a) Dataset 1, (b) Dataset 2. Solder Defects, (c) Dataset 3. Surface Scratches.

Figure 17. ROC curves using SIFT. (a) Dataset 1, (b) Dataset 2. Solder Defects, (c) Dataset 3. Surface Scratches.

Figure 18. Integration setup.

VI. SUMMARY AND FUTURE OUTLOOK

This paper proposes a deep autoencoder-based feature extractor, which can be used as a flexible defect detection algorithm. The technique does not require any defect image during the training, and can be used to detect varied types of defects without much domain expertise. We, however, must acknowledge that at this stage, this method has a few limitations as well. One important parameter is threshold selection, which varies significantly from dataset to dataset, and was chosen empirically to find best accuracies. The method can be applied in situations when there is only one golden sample as a reference image, by using data augmentation technique, but if the background is texture-based, then we still need a large number of normal samples during training, and simple data augmentation does not work. Block size is another parameter; whose optimal value depends on the type/size of the defect. A large image is divided into blocks of sub images of size suitable for inspection, hence the user still needs to have some understanding about the size of the defect region.

In our future work, we plan to divide a larger image into non-uniform blocks. By choosing custom block sizes and locations where defects are more likely to occur, the accuracy of defect detection can be increased further. Different configurations of autoencoder is another area where the defect detection performance can possibly be improved.

ACKNOWLEDGEMENT

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

- [1] C. Tikhe and J. S. Chitode, "Metal surface inspection for defect detection and classification using Gabor Filter," *International Journal of Innovative Research in Science, Engineering and Technology*, vol. 3, no. 6, 2014
- [2] F. Timm and E. Bartha, "Non-parametric texture defect detection using Weibull features," *Proceedings of SPIE. SPIE-IS&T*, vol. 7877, 2011, pp. 78,770j–78,770j
- [3] Z. Wang and A. Bovik, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, 2004
- [4] M. Miura, S. Sakai, S. Aoyama, J. Ishii, K. Ito, and T. Aoki, "High-accuracy image matching using phase-only correlation and its application," *SICE Annual Conference*, pp. 307–312, 2012.
- [5] A. Bodnarova, M. Bennamoun, and S. Latham, "Optimal Gabor filters for textile flaw detection," *Pattern Recognition-Elsevier*, vol. 35, no. 12, pp. 2973–2991, Dec. 2002
- [6] M. A. Garcia and D. Puig, "Pixel classification by divergence-based integration of multiple texture methods and its application to fabric defect detection," *Joint Pattern Recognition Symposium Springer*, vol. 2781, pp. 132–139, 2003
- [7] K.Y. Song, M. Petrou, and J. Kittler, "Texture crack detection," *Machine Vision Applications*, vol. 8, no. 1, Jan. 1995, pp. 63–76
- [8] T. Wang and N. Karayiannis, "Detection of micro-calcifications in digital mammograms using wavelets," *IEEE Transactions On Medical Imaging*, vol. 17, no. 4, pp. 498–509, Aug. 1998
- [9] M. Melloul and L. Joskowicz, "Segmentation of micro-calcification in X-ray mammograms using entropy thresholding," In *Proceedings of the 16th International Congress on Computer-Assisted Radiology and Surgery*, pp. 490–495, 2002
- [10] A. Papadopoulos, D. Fotiadis, and A. Likas, "An automatic micro-calcification detection system based on a hybrid neural network classifier," *Elsevier Artificial Intelligence in Medicine*, vol. 25, no. 2, pp. 149–167, Jun. 2002
- [11] V. Leemans and M. Destain, "A real-time grading method of apples based on features extracted from defects," *Journal of Food Engineering*, vol. 6, no. 1, pp. 83–89, Jan. 2004
- [12] I. Silven, M. Niskanen, and H. Kauppinen, "Wood inspection with non-supervised clustering," *Machine Vision and Applications*, vol. 13, no. 56, pp. 275–285, Mar. 2003
- [13] R. Carro, J. Ahuactzi and E. Huerta, "Face recognition using SURF," *International Conference on Intelligent Computing Theories and Methodologies, Springer*, vol. 9225, pp. 316–326, ICIC 2015
- [14] B. Suvdaa1, J. Ahn, and J. Ko, "Steel surface defects detection and classification using SIFT and voting strategy," *International Journal of Software Engineering and Its Applications*, vol. 6, no. 2, Apr. 2012
- [15] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004
- [16] Ibrahim, I., Ibrahim, Z., Khalil, K., et al.: 'An algorithm for classification of five types of defect on bare printed circuit

- board', *Int. J. Comput. Sci. Eng. Syst.*, 2011, 5, (3), pp. 201–208
- [17] Putera, S.H.I., Ibrahim, Z.: 'Printed circuit board defect detection using mathematical morphology and MATLAB image processing tools'. *Proc. Int. Conf. on Education Technology and Computer*, Shanghai, China, June 2010, pp. V5-359–V5-363
- [18] Putera, S.H.I., Dzafaruddin, S.F., Mohamad, M.: 'Matlab based defect detection and classification of printed circuit board'. *Proc. Int. Conf. on Digital Image Computing Techniques and Applications*, Bangkok, Thailand, May 2012, pp. 115–119
- [19] Ibrahim, Z., Al-Attas, S.A.R.: 'Wavelet-based printed circuit board inspection system', *Int. J. Signal Process.*, 2004, 1, (1), pp. 73–79
- [20] Khalid, N.K., Ibrahim, Z., Abidin, M.S.Z., et al.: 'An algorithm to group defects on printed circuit board for automated visual inspection', *Int. J. Simul. Syst. Sci. Technol.*, 2008, 9, (2), pp. 1–10
- [21] X. Bai, Y. Fang, W. Lin, L. Wang and B. Ju, "Saliency-Based Defect Detection in Industrial Images by Using Phase Spectrum," in *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2135-2145, Nov. 2014.
- [22] X. Wu, K. Cao, and X. Gu, "A surface defect detection based on convolutional neural networks," *International Conference on Computer Vision Systems*, pp. 185-194, ICVS 2017
- [23] J. Masci, U. Meier, and D. Ciresan "Steel defect classification with max-pooling convolutional neural networks," *International Joint Conference on Neural Networks*, 2012
- [24] T. Wang, Y. Chen, M. Qiao and H. Snoussi, "A fast and robust convolutional neural network based defect detection model in product quality control," *The International Journal of Advanced Manufacturing Technology*, vol. 94, pp. 3465-3471, Feb. 2018
- [25] X. Wei, Z. Yang, Y. Liu, D. Wei, L. Jia, y. Li, "Railway track fastener defect detection based on image processing and deep learning techniques: A comparative study," *Engineering Applications of Artificial Intelligence* 80 (2019): 66-81.
- [26] M. Oquab, L. Bottou, I. Laptev and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," *IEEE International Conference on Computer Vision and Pattern Recognition*, 2014
- [27] S. Dey, A. Dutta, J. Ignacio, S. Ghosh, J. Liados, and U. Pal, "Signet: convolutional siamese network for writer independent offline signature verification," *arXiv preprint arXiv:1707.02131*, 2017
- [28] Y. Guo and L. Zhang, "One-shot face recognition by promoting underrepresented classes," *arXiv preprint arXiv:1707.05574*, 2017
- [29] 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. 2018, no. 16, pp. 1415-1420, 11 2018.
- [30] G. Koch, R. Zemel and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," *ICML Deep Learning Workshop*, 2015
- [31] Wenting Dai, Abdul Mujeeb, Marius Erdt and Alexei Sourin, "Towards Automatic Optical Inspection of Soldering Defects," In *Proc. 2018 Int. Conf on Cyberworlds, IEEE CPS*, Singapore, 3-5 Oct. 2018, pp. 375-372.
- [32] F. Ratle, G. Camps-Valls, J. Weston, "Semisupervised neural networks for efficient hyperspectral image classification," *IEEE Trans. Geosci. Remote Sensing*, vol. 48, no. 5, pp. 2271-2282, May 2010.
- [33] J. Ker, L. Wang, J. Rao and T. Lim, "Deep Learning Applications in Medical Image Analysis," in *IEEE Access*, vol. 6, pp. 9375-9389, 2018.
- [34] S. Petscharnig, M. Lux and S. Chatzichristofis, "Dimensionality reduction for image features using deep learning and Autoencoders," *Proceedings of the 15th International Workshop on Content-Based Multimedia Indexing*, Florence, Italy, pp. 23:1-23:6, 2017
- [35] M.D. Zeiler and R. Fergus, "Visualization and understanding convolutional networks," *European Conference on Computer Vision (ECCV)*, pp 818-833, 2014
- [36] K. P. Diederik and B. Jimmy, "ADAM: A method for stochastic optimization," *arXiv:1412.6980*, Dec. 2014
- [37] T. Alexandru, "An image inpainting technique based on the fast marching method," *Journal of Graphics Tools*, vol. 8, no. 1, pp. 23-34, 2004