Unsupervised surface defect detection using deep autoencoders and data augmentation

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

2018

Abdul Mujeeb, Dai, W., Erdt, M., & Sourin, A. (2018). Unsupervised surface defect detection using deep autoencoders and data augmentation. Proceedings of the 2018 International Conference on Cyberworlds (CW), 391-398. doi:10.1109/cw.2018.00076

https://hdl.handle.net/10356/137972

https://doi.org/10.1109/cw.2018.00076

Unsupervised Surface Defect Detection Using Deep Autoencoders and Data Augmentation

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 Univeristy, Singapore
daiw0004@e.ntu.edu.sg

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

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

***Abstract*—Surface level defect detection, such as detecting missing components, misalignments and physical damages, is an important step in any manufacturing process. In this paper, similarity matching techniques for manufacturing defect detection are discussed. We are proposing an algorithm which detects surface level defects without relying on the availability of defect samples for training. Furthermore, we are also proposing a method which works when only one or a few reference images are available. It implements a deep autoencoder network and trains input reference image(s) along with various copies automatically generated by data augmentation. The trained network is then able to generate a descriptor—a unique signature of the reference image. After training, a test image of the same product is sent to the trained network to generate a test image descriptor. By matching the reference and test descriptors, a similarity score is generated which indicates if a defect is found. Our experiments show that this approach is more generic than traditional hand-engineered feature extraction methods and it can be applied to detect multiple type of defects.**

Keywords—Defect Detection, AOI, Deep Learning, Autoencoders, Unsupervised Learning, Data Augmentation, Similarity Matching

I. INTRODUCTION

During various stages of a manufacturing process, surface level defects may occur, such as missing and misaligned components, physical damages like scratches, cracks, holes, etc. To find these defects, either visual inspection is done by human experts, or Automatic Optical Inspection (AOI) is conducted using computer algorithms. AOI is becoming increasingly popular, and it is the focus of this paper. From the image processing point of view, the surface level defects can be defined as deviations in the test image compared to a given reference image. In this work we aim at detecting significant differences between a test and a reference images, i.e. at similarity matching of two images. Currently, most manufacturing defect detection systems use conventional image processing techniques. These techniques rely on hand-engineered features which are chosen by skilled people with good technical and domain knowledge. An alternative approach to conventional image processing is machine learning algorithms, which is gaining popularity very rapidly and have replaced many traditional techniques in various areas of applications.

The conventional techniques of surface inspection rely on hand-designed features which have a problem of weak adaptability, i.e. for every new type of defect and product the feature extraction process needs to be redesigned. In contrast, machine learning algorithms have the capability to automatically extract powerful features with less prior domain knowledge. Supervised learning is by far the most common type of machine learning algorithm, and it requires a large number of training samples. Most machine learning-based defect detection algorithms use supervised learning method, and thus they need a large number of training images. This may not be however practical in many cases where defect samples may not be available at all during training. In this paper, we are proposing an unsupervised learning algorithm which does not require any defect image during training.

The structure of this 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 conventional and machine learning techniques. In Section III, we propose an approach to solve the problem of surface level defect detection. The implementation details are discussed in Section IV. In Section V, we analyze and compare the experimental results. We outline further integration plans in Section VI and conclude the paper in Section VII.

II. RELEVANT STUDIES AND OUR WORK

Surface analysis technique is an algorithmic attempt to inspect a surface for possible defects. Defects could be of various types and many different methods are used for inspection. We can broadly categorize the feature extraction techniques as *traditional image processing based* and *machine learning based* techniques.

A. Traditional Image Processing Based Techniques

A very simple similarity matching technique is to directly compare the pixel values of the two images by using some correlation measure. This method is rarely used since it is

extremely sensitive to minor transformations due to imaging conditions, e.g., intensities. Therefore, more robust methods have been devised, e.g., one method is to compute a feature descriptor of each image and then to apply a distance measure on the two descriptors. Many techniques have been developed for various applications, and in the following paragraph we perform a survey of such traditional image processing methods found in literature.

Work [1] classified the inspection techniques into *statistical*, *structural*, *filter-based*, and *model-based*. *Statistical methods* evaluate the spatial distribution of pixel intensities. The most popular statistical methods use first order statistics such as mean, variance, range, and other histogram-based computations together with the second order statistics. For example, [2] used Weibull distribution on local edges to detect texture anomaly in images. Based on *structural similarity*, the authors of [3] proposed an image comparison technique using statistical parameters instead of pixel-based similarity. *Filter-based approaches* often imply a bank of filters, such as gradient filters, Gabor filters, Fourier transform that are applied to the image, where the energy of the filter responses is used as a feature. In [4], an image matching technique was developed using Phase-Only Correlation technique which uses the phase components in Discrete Fourier Transforms of given images. If prior knowledge is available about the underlying process that generates the texture pattern, and if this pattern can be defined by a stochastic model, then the *model-based approaches* naturally yield very accurate results in texture analysis. Texture classification techniques are used in various disciplines including biomedical [5, 6, 7], textile [8, 9], metal surfaces [10], wood [11], food product inspection [12], etc. Two very popular feature extraction techniques are Scale-Invariant-Feature-Transform [13, 14] and Speed Up Robust Features [15], which have shown very good results in many pattern recognition applications.

B. Machine Learning Based Techniques

Selecting a feature extraction technique for a given application is a challenging task. Machine learning is an alternative technique that extracts the best features automatically from the given data, and it has already started finding its applications in the field of manufacturing defect detection. In [16], a feature extraction technique was proposed based on neural networks that works on any arbitrary textured images. Neural networks for supervised steel defect classification were applied in [17]. Work [18] used a neural network-based defect detection on DAGM dataset outperforming all the existing techniques applied so far.

To our best knowledge, the machine learning techniques used in defect detections are basically supervised learning algorithms relying on large number of training samples. This poses a practical limitation in the defect detection applications where large number training samples may not be available. In some cases, even defect free samples are limited to one or very few samples. Algorithms have been developed that can learn from only one training sample, and are called one-shot learning algorithms. To our knowledge one-shot learning methods have been successfully used in object/pattern recognition applications. For example, [19] implemented an offline signature verification using Convolutional Siamese Network in which a neural network is trained using very large number of training samples, and after training it is able to extract a unique descriptor from any single signature image. Work [20] introduced a face recognition technique where only one image per person is available. The network is trained using a large face database, and after the training the network becomes capable of generating a descriptor for any given face image. This approach of generating descriptor has outperformed conventional hand-engineered descriptor algorithm. Transfer learning [21] was used to extract features from low volume training data, however, this required that the transfer domain had to be similar to the new domain.

C. Problem Definition

It can be concluded that since similarity matching techniques can be of various types, human expertise is needed to choose the best technique for a given application. Even though the best results have been obtained by machine learning techniques, they depend on availability of large training data. For example, in [22] it is shown how to directly learn features from image data using Convolutional Networks without resorting to manually designed features. Thus, we have to answer a question "*can we learn best features automatically where no defect sample is available, and only one or very limited normal samples are available for training?*" In this regard, we aim to implement a machine learning-based feature extractor which: 1) does not require domain knowledge and is portable to wide variety of products and defects, and 2) does not depend on the availability of defect samples for training. To that end, and also inspired by the recent advances in deep learning, we choose to represent such a feature extractor in terms of a deep neural network

A deep learning network is able to generate a unique descriptor from a given input image by learning various geometric patterns. However, before a network is able to extract patterns, it must be trained (i.e. parameters have to be tuned). This process of parameter tuning needs a large number of training samples. We generate a large number of copies of the input image by applying data augmentation techniques, such as flipping and rotation. Furthermore, we consider autoencoders, which are neural networks used to find a compact representation of an input image. In [23] deep learning and autoenoders are used to generate a compressed representation for image features, and the last layer of the encoder is used as a compact descriptor of the input image representing its features. After the training, the network is able to generate a descriptor for a given test image with or without a defect. By comparing the reference and test image descriptors, we may decide about the presence of any possible defect.

III. Methodology and Approach

A. Motivation and Background

This work is a part of an industrial project involving the development of AOI techniques in a manufacturing environment as shown in Fig. 1. The set-up contains a high quality camera which sends images of components to a computer running defect detection algorithm. We aim at devising an algorithm which is flexible enough to be used with a wide range of products and defects, and is able to operate within the relatively limited resources of a typical embedded system. Our focus is on defects which can be visually checked by human experts (Fig. 2).

B. Feature Extractor

Many AOI problems are essentially similarity matching problems, which require comparing two images to find a quantitative measure of similarity. Feature selection is probably the most important step in similarity matching. In order to compare two images, features vectors are extracted from each image and compared using some algorithm as shown in Fig. 3. This process consists of two steps, viz., a feature extractor (FE) algorithm and a matching algorithm. A FE algorithm extracts features from two images which are then compared using the matching algorithm to generate a quantitative measure of similarity between the two images.

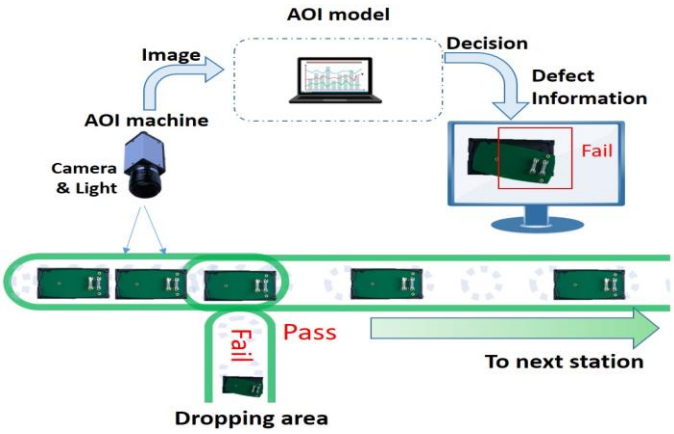


Figure 1. Target AOI system (actual electronic components have been replaced with place-holder images).

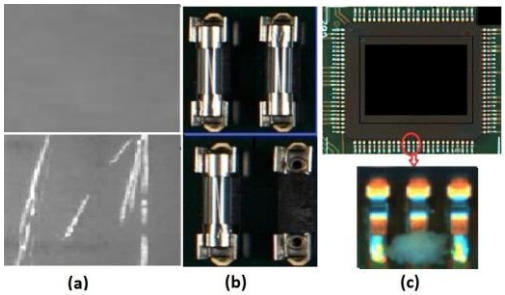


Figure 2. Defects examples: (a) scratches, (b) misplaced components, and (c) soldering defects.

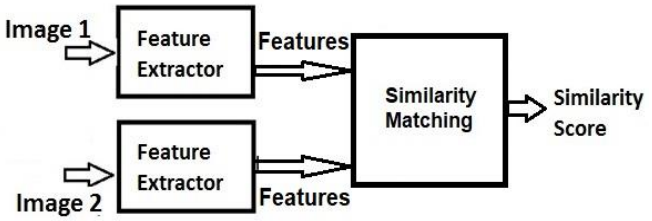


Figure 3. Similarity matching algorithm.

If we had many images of defect free and defected samples, then highly discriminative features can be extracted automatically via supervised learning, which is by far the most common method of machine learning. However, for small datasets or even cases where only a single reference image is available, it is challenging to learn generic features, which is why so far matching algorithms have mostly relied on supervised training methods. If sufficient training samples are not available, then traditional hand-crafted feature selection methods are employed.

C. Research Hypothesis and Choice of Feature Extractor

Our hypothesis is that *in contrast to conventional manual feature extraction methods, we can extract more robust features by using only defect free images, deep learning algorithms, and by creating data augmented copies of the image.*

We propose to use a deep Autoencoder network to extract features. An Autoencoder algorithm, also known as data compression algorithm, consists of two neural networks called as Encoder and Decoder which are connected in series as shown in Fig. 4. The encoder part extracts patterns from a given input image, and its last layer consists of fewer nodes than the input size. The reduced size output becomes a compressed representation of the input image. During training, the decoder takes this compressed representation as the input and tries to reproduce the original image as its output. Each node of the encoder output represents a certain pattern which is learnt from the input dataset. The process of feature extraction at various layers in a Neural Network is explained in [24] through a visualization technique. The lowest layer extracts more generic features like lines, curves, etc., whereas the nodes in the deeper layers may extract more specific shapes like semicircles, squares, etc., which are combination of the previous layer features as shown in Fig. 5. Each node in the last layer of the encoder represents a certain geometric pattern. So an encoder with N number of nodes in the last layer generates an N-size descriptor of an input image. A problem with autoencoder-based feature extractor is that it performs very poorly if used with only very few training images. Alternatively, the techniques like Siamese network have been used where training is performed by comparing a reference image with a similar image called positive image and a different image called negative image.

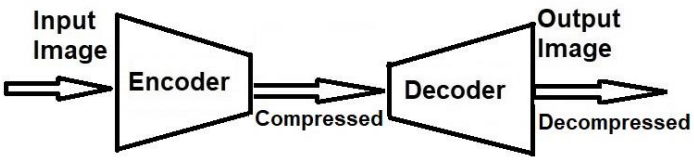


Figure 4. An Autoencoder block diagram.

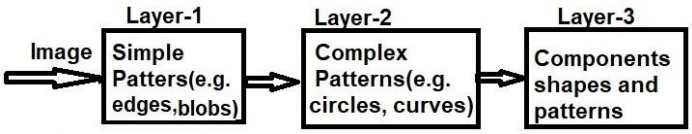


Figure 5. Feature extractions at various layers.

In Siamese networks, the optimization is performed using a loss function utilizing similarity of positive and difference of negative images. This optimization function is called triplet loss function [25]. However, during training this approach still requires a large number of images (with positive and negative pairs) from the same domain, and it has been used successfully in face recognition, digit recognition and signature verification. Once a network has been tuned, we can then capitalize on powerful discriminative features to generalize the predictive power of the network to new data.

In our case, the challenge remains in the training phase so that the network should be able to generate features from samples never seen before. Therefore, we perform data augmentation of the input image by horizontal and vertical flipping followed by rotation in multiple steps. After training, the encoder is able to generate a descriptor for a given image of the same product. The reference and test images are passed through the same trained network to generate reference descriptor and test descriptor respectively. The feature vectors are normalized to unity magnitude and then are compared for similarity using L2-norm, which generates a similarity score between 0 and 1 representing identical or maximally different images respectively.

D. Effect of Image Patch Size

For defect localization, a large image is divided into smaller sub-images, and each sub-image has to be trained independently of one another. We presume that the ratio of the defect region and the total image region is important. If the ratio is too small, then the region of the defect will contribute less to the feature descriptor as shown in Fig. 6, and a generated score will remain close to the normal image score.

E. Validation Plan

We have chosen three different types of defects to ensure the generality of the proposed method. Firstly, we used scratch images from NEU dataset [26] (Fig. 2(a)). Good images were created by removing of the scratched areas using an inpainting algorithm [27]. Secondly, we created our own database of misplaced components from a research AOI platform (Fig. 2(b)). Thirdly, a dataset of soldering images was obtained from a professional PCB manufacturer (Fig. 2(c)).

IV. IMPLEMENTATION DETAILS

Background of the normal image could be smooth or texture-based. Two different methods have been used to extract features using an autoencoder depending on the type of background. The number of required training samples is different for smooth and texture surfaces, as shown in Table I.

TABLE I. TRAINING SAMPLES NEEDED.

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

If the surface is smooth, then only one reference image may be used for feature extraction by creating various copies using data augmentation. For surfaces with textures, we need large number of good images for training that include most common texture patterns which are likely to appear on the surface. If components are placed on smooth surfaces, we may rely on data augmentation to generate large training dataset. In both cases, however, we do not need any defect image for training.

A. Data Augmentation

To ensure sufficient training iterations to extract features and avoid overfitting, we performed data augmentation on the reference input image. This was achieved by horizontal and vertical flipping followed by rotations. The original image, as well as its flipped copies, were rotated by various angles between –180 to +180 degrees in steps of 3 degrees. This generated sufficiently large number of images to perform training of the encoder.

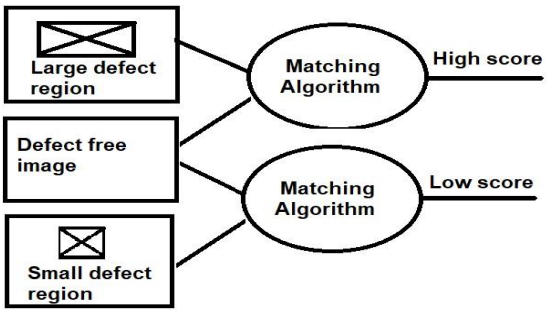


Figure 6. Effect of defect region selection on detection.

B. Image Preprocessing

Image preprocessing includes graying of input images (if needed), and data normalization. In applications where color information is important, such as solder images, graying is skipped. Data normalization is performed on each training sample and is described in Fig. 7.

```
max = Maximum pixel value in the image
min = Minimum pixel value in the image
REPEAT for each pixel
BEGIN
  PixelValue= (PixelValue – min) / (max-min)
END
```

Figure 7. Data normalization algorithm.

C. Creating Smaller Blocks from Input Image

For better accuracy of defect detection and narrowing down the defect location, large input images are converted into small images by dividing them using horizontal and vertical grids, as shown in Fig. 8. For this experiment, we chose grid size of 64×64 pixels. Hence, if an input image is of size 640×640 pixels, it is divided into 100 patches of 64×64 pixels. Each sub-image is processed independently of others, and the training process is applied on each of the patches.

D. Deep Autoencoders

We created a deep autoencoder for training with three fully connected layers. Multiple hidden layers are able to learn complex geometric patterns from the training data. The last layer of the encoder is used as a descriptor (feature vector). The decoder part was used only during training, and it is a mirrored image of the encoder part. The last layer in the autoencoder uses Sigmoid activation function, whereas all other layers use Rectifier Linear Unit (Relu) activation function.

A loss function measures how close the reconstructed output vector *y* is to the original input vector *x*. We are choosing binary cross-entropy of reconstruction loss function, which calculates how many bits of information is preserved in the reconstruction as compared to the original. The loss function is defined as

$$J(x, y) = -\sum_{k}[x_k \log y_k + (1 - x_k)\log(1 - z_k)]$$

ADAM optimizer [28] is used to optimize the loss function which is converging between 500 to 1,000 iterations.

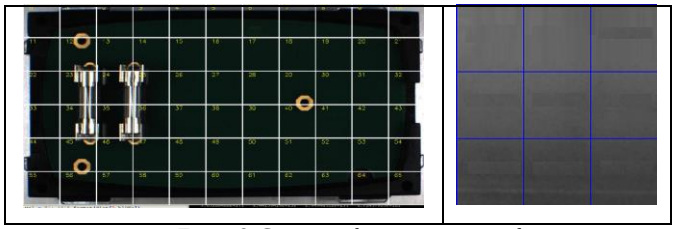


Figure 8. Creating sub-images using grids.

E. Similarity Matching Algorithm

The last layer of the encoder is a 1D array, and it represents the feature vector. If **X** is a feature vector with **N** components as follows.

$$\mathbf{X} = \{x_1, x_2, x_3, ...x_N\}$$

Then, a unit vector of **X** is obtained by dividing each of its components by its Euclidean length as follows.

$$x_i' = \frac{x_i}{\|\mathbf{X}\|} \qquad \text{[for every component } i\text{]}$$

where $\|\mathbf{X}\| = \sqrt{x_1^2 + x_2^2 + x_3^2 + ...x_N^2}$.

Two vectors generated from two images are matched for similarity using L2-norm. For any two given vectors **X** and **Y** of length N, the distance is computed as follows.

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

which generates a similarity score between 0 (identical images) and 1 for two feature vectors **X** and **Y**.

F. Training Algorithm

Two slightly different methods have been used for texture and smooth backgrounds, which are shown in Fig. 9(a) and 9(b), respectively. Each training sample is flattened from 2D to 1D, and training is performed in batch mode using all the training images as shown in Fig. 10.

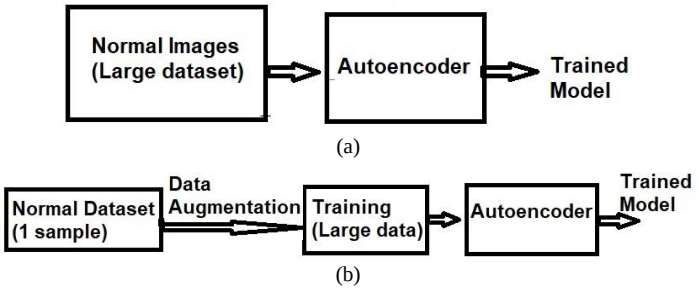


Figure 9. Training process: (a) for texture background and (b) for smooth background.

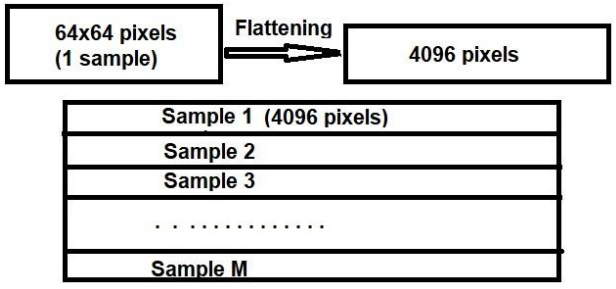


Figure 10. Training matrix (using M samples).

G. Test Algorithm

Test algorithm involves extracting features from a test image using the trained model and matching the test and reference (i.e. training) images feature vectors. Figures 11(a) and 11(b) show the algorithm used for matching the descriptors for texture and smooth backgrounds, respectively. Additionally, a test image also goes through the same image preprocessing steps as the training images, such as graying, normalization, and flattening.

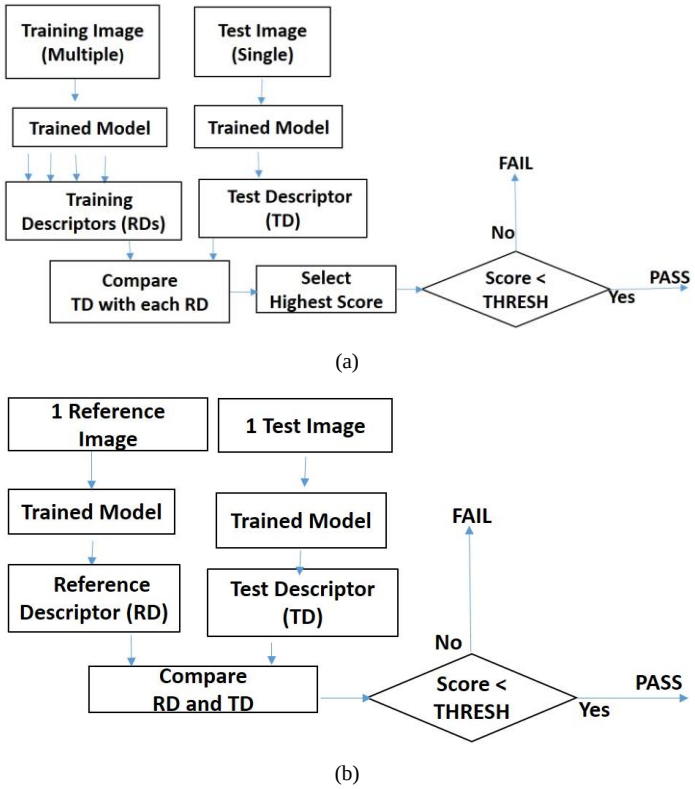


Figure 11. Test algorithms: (a) for texture background and (b) for smooth background.

V. Experimental Results

A. Evaluation Datasets

For evaluation we have chosen three datasets with three different types of surface defects. Dataset 1 includes the images captured from an AOI research platform (Fig. 12). Dataset 2 includes soldering images obtained from the manufacturing facility used in the project (Fig. 13). Dataset 3 includes NEU surface scratch images (Fig. 14). For training we used only defect-free images, and they were not used at the later stages. At the testing stage, we used both normal and defected images. Table II shows the number and types of sample images used for evaluation. Datasets 1 and 3 contain images of larger size which were divided into patches of 64×64 pixels. Dataset 2 contains images of smaller size of 35×140 pixels, hence they were directly used in the experiment without creating any small blocks.

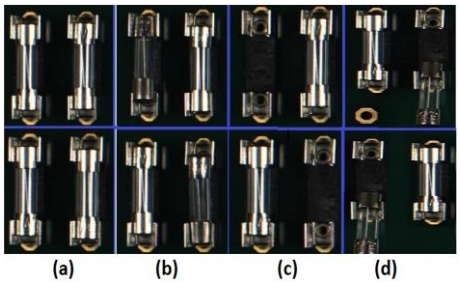


Figure 12. Dataset 1:
(a) normal, (b) misaligned, (c) missing, (d) misplaced.

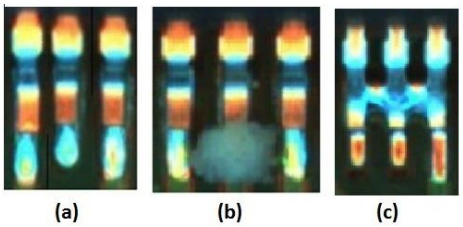


Figure 13. Dataset 2:
(a) normal, (b) defected(foreign element) and (c) defected(bridge).

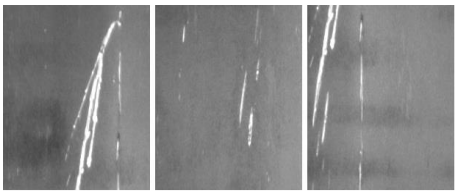


Figure 14. Dataset 3:
scratch samples from NEU dataset.

TABLE II. IMAGES USED FOR EVALUATION.

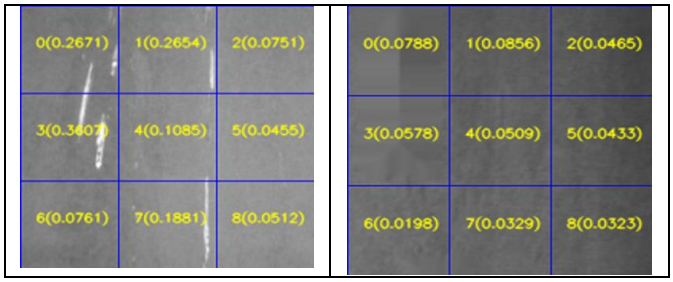
Dataset	Training (Normal)	Test (Normal)	Test (Defected)	Size & Type
Internal	1 (×480 augmented copies)	330 patches	330 patches	64×64 (Gray)
Solder Defects	200 images	100 images	79 images	35×140 (RGB)
NEU Scratches	65×9 patches	30×9 patches	300×9 patches	64×64 (Gray)

B. Detection Accuracy

Defect detection accuracy depends on the chosen value of the threshold. Figs. 15 and 16 show a marked increase in the matching threshold in defected blocks as compared to normal blocks for datasets 1 and 3, respectively. In Fig. 15, the defected region score is >0.3 as compared to normal region score which is <0.05. In Fig. 16, the defected region score is >0.25 as compared to normal region score which is <0.05. We generate ROC curves

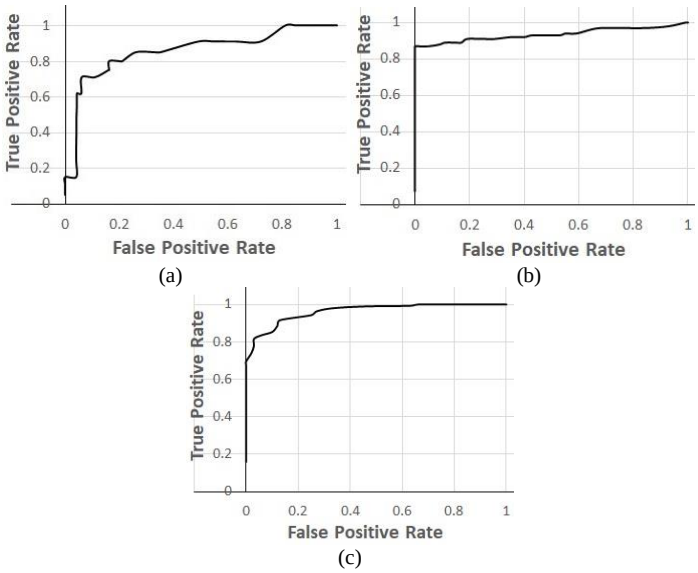
by thresholding the distance between the test and the reference image descriptors, as shown in Fig. 17. For dataset 1, we are able to use the defects with True Positive Rate (TPR) of over 80% while the False Positive Rate (FPR) is 16%. For dataset 2, the TPR and FPR are 89% and 5%, respectively, and for dataset 3, the TPR and FPR are 83% and 7%, respectively.

Figure 15. Defected and normal blocks.



(a) (b)

Figure 16. (a) defected sample and (b) normal sample.



(a) (b) (c)

Figure 17. ROC curves. (a) Dataset 1. Missing/Misaligned Components. (Thresh = 0.15, TPR = 0.81, FPR = 0.16); (b) Dataset 2. Solder Defects. (Thresh = 0.24, TPR = 0.89, FPR = 0.05); (c) Dataset 3. Surface Scratches. (Thresh = 0.15, TPR = 0.83, FPR = 0.07).

It should be noted that our objective is to find an automatic and generic feature extraction technique. It is possible that a manually crafted technique may be able to detect a certain defect (e.g., scratches) with the same or even better accuracy, but that technique may not be suitable for other type of defects (e.g., misplaced components). The ROC curves indicate that the technique is generic and can be applied to detect various different types of defects with good accuracy.

VI. Integration Plan

In the next stages, this work is planned to be integrated in the manufacturing pipeline, as shown in Fig. 18. The target AOI system is an embedded system with memory and speed constraints. The training is to be performed offline on a more powerful system preferably containing a GPU. The trained model contains a few thousand parameters which can be easily fit into most embedded systems used for AOI. Some additional image processing steps, such as image segmentation and alignment, will also be included at the deployment stage.

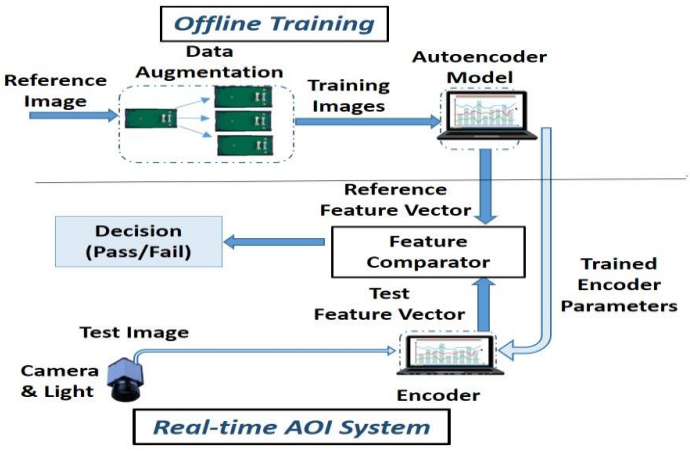


Figure 18. Integration setup.

VII. Summary and Future Outlook

An autoencoder-based feature extractor was proposed in this paper which is used as a defect detection algorithm. This technique has the advantage of automatic feature extraction, and it can be applied to detect various different types of defects without much domain expertise. Furthermore, it does not depend on the availability of defect images during training. We, however, must acknowledge that this method has certain limitations. For images with smooth background, data augmentation works well, but if the background is texture-based, then we still need a large number of defect-free training samples, and data augmentation is not a viable option. Large size images need to be divided into blocks of a proper size, hence the user

needs to have some understanding about the size of the defect region.

In our future work, we will divide a larger image into non-uniform blocks instead of fixed size blocks, which may allow us to detect various different types of defects in one product by choosing proper block sizes where they are most likely to occur.

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] 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

[6] 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

[7] A. Papadopoulosa, D. Fotiadisb, and A. Likasb, "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

[8] 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

[9] 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

[10] K.Y. Song, M. Petrou, and J. Kittler, "Texture crack detection," Machine Vision Applications, vol. 8, no. 1, Jan. 1995, pp. 63-76

[11] 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

[12] 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

[13] 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

[14] D. Lowe, "Distinctive image features from scale-invariant keypoints." International Journal of Computer Vision, vol. 60, no. 2, pp. 91– 110, Nov. 2004

[15] 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

[16] 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

[17] J. Masci, U. Meier, and D. Ciresan "Steel defect classification with max-pooling convolutional neural networks," International Joint Conference on Neural Networks, 2012

[18] 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

[19] 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

[20] Y. Guo and L. Zhang, "One-shot face recognition by promoting underrepresented classes," arXiv preprint arXiv:1707.05574, 2017

[21] 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

[22] S. Zagoruyko and N. Komodakis, "Learning to Compare Image Patches via Convolutional Neural Networks," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015

[23] 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

[24] M.D. Zeiler and R. Fergus, "Visualization and understanding convolutional networks," European Conference on Computer Vision (ECCV), pp 818-833, 2014

[25] G. Koch, R. Zemel and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," ICML Deep Learning Workshop, 2015

[26] NEU Metal Surface dataset (date of citation: 7th May 2018) http://faculty.neu.edu.cn/yunhyan/NEU_surface_defect_database.html

[27] T. Alexandru, "An image inpainting technique based on the fast marching method," Journal of Graphics Tools, vol. 8, no. 1, pp. 23-34, 2004

[28] K. P. Diederik and B. Jimmy, "ADAM: A method for stochastic optimization," arXiv:1412.6980, Dec. 2014