

Deep Learning for Smart Industry: Efficient Manufacture Inspection System With Fog Computing

Liangzhi Li¹, Student Member, IEEE, Kaoru Ota², Member, IEEE, and Mianxiong Dong³, Member, IEEE

Abstract—With the rapid development of Internet of things devices and network infrastructure, there have been a lot of sensors adopted in the industrial productions, resulting in a large size of data. One of the most popular examples is the manufacture inspection, which is to detect the defects of the products. In order to implement a robust inspection system with higher accuracy, we propose a deep learning based classification model in this paper, which can find the possible defective products. As there may be many assembly lines in one factory, one huge problem in this scenario is how to process such big data in real time. Therefore, we design our system with the concept of fog computing. By offloading the computation burden from the central server to the fog nodes, the system obtains the ability to deal with extremely large data. There are two obvious advantages in our system. The first one is that we adapt the convolutional neural network model to the fog computing environment, which significantly improves its computing efficiency. The other one is that we work out an inspection model, which can simultaneously indicate the defect type and its degree. The experiments well prove that the proposed method is robust and efficient.

Index Terms—Deep learning, fog computing, manufacture inspection, smart industry.

I. INTRODUCTION

INDUSTRY 4.0, or in other words, the smart industry, is a word representing the current trend of manufacturing revolution [1]. Two most important concepts in this smart industry era are automation and data. The former one is one of our main objectives and the latter one is one of our most useful tools. Data can be analyzed and learned by some artificial intelligence (AI) methods, deep learning as an example, and empower the computers and manipulators with humanlike abilities. In order to collect more data, which is essential for the AI approaches, more and more Internet of things (IoT) [2] enabled devices are

deployed in smart factories [3]. Using these data, people have come up with lots of new ways to make the manufacturing process more automated and efficient. In this paper, we focus on the autonomous manufacturing inspection, which is an area with a long history but still has many problems currently, especially in the big data era.

One serious problem existing in this task is that current inspection system cannot guarantee good performances while keeping the processing efficiency. The traditional methods, such as filters or some feature-based classification approaches, are simple but not so effective in all the scenarios. Then, the deep learning based methods [4] turned up, bringing greatly improved analysis and recognition abilities, however, at the same time, slowing down the running speed, as these methods usually require a lot of computation [5], [6].

It may not be a serious problem when the data size is not so large. But with the rapid development of smart industry, people invest their maximum effort on the manpower reduction by deploying vision sensors [7], [8] in each manufacturing line and empowering them with the ability of autonomous defect detection, which results in a hugely increased data size. Considering the production capacity, the computing efficiency has become the bottleneck to implement a real-time inspection system for smart industries.

Computation offloading is important to improve the computing efficiency and build a real-time system [9], and the newly developed fog computing [10] can serve as a good solution in this scenario. In this paper, we design our manufacture inspection system, named DeepIns, as three modules, i.e., the fog-side computing module, the backend communication module, and the server side computing module. The two computing modules are to calculate the deployed deep models, and the communication module is responsible for the data exchanges and command transfer.

We focus on the combination of these three modules. In order to further decrease the response latency and network traffic, we design the fog-side computing module with an early-exit feature, which can stop the inference process and obtain the classification results in advance.

The application scenario is shown in Fig. 1; in some places of the production line, which usually move along one side, several sensors or cameras are deployed to capture the visual information regarding the products. Then, the information will

Manuscript received April 16, 2018; accepted May 9, 2018. Date of publication June 1, 2018; date of current version October 3, 2018. This work was supported in part by the JSPS KAKENHI under Grant JP16K00117 and in part by the KDDI Foundation. Paper no. TII-18-0932. (Corresponding author: Mianxiong Dong.)

The authors are with the Department of Information and Electronic Engineering, Muroran Institute of Technology, Muroran 050-8585, Japan (e-mail: 16096502@mmm.muroran-it.ac.jp; ota@mmm.muroran-it.ac.jp; mxdong@mmm.muroran-it.ac.jp).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TII.2018.2842821

1551-3203 © 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Specifically, instead of one fixed pose, multiple poses of an image are analyzed, which can capture more details of the product and obtain the required visual information. Normally, the feature training is hard to conduct since there exists a large number of defect samples of products, which is one of the limitations of inspection approaches as well. Aiming to avoid this difficulty, Wu *et al.* [16] proposed a method, which fuses three image features for achieving automatic surface inspection. Specifically, features are learned to analyze the qualified image, which consists of pixel regularity, subimage gray level difference, and color histogram. Then, images are tested based on the detection fusion of the three features. Moreover, the simulation results illustrated in [16] prove that the proposed method is efficient for detecting industrial product and can ensure the product quality.

In [14], a modern method using pulse coupled neural networks [17] is proposed for specific defects of integrated circuits (IC) without the reference image. Due to the characters of IC, the method presents a six-step process for identifying the position of IC defects, which can be illustrated briefly as an estimation of image noise parameter, bilateral filter, edge detection, Hough transform, classification, and defect detection. Since the main goal of the method is to achieve the effective segmentation and accurate detection, the image segmentation is the most significant step in the process. The detection process is performed in a self-adaptive way, and the simulation results present that the defect detection can be operated automatically with high effectiveness.

In building material inspection, modern methods can also play an important role to guarantee the stability and safety of building construction. As the key factor in a building, pile quality is studied in [18]. A wavelet packet transform method is proposed, which is adopted to analyze the stress wave reflected signals. The wavelet packet energy ratio can reflect much information of pile defects, such as the energy distribution of signals. Weixin *et al.* [18] use multilayer support vector machine to classify the pile defects and present a novel approach, i.e., feature extraction to detect the defects. As shown in the simulation, the proposed approach can achieve high accurate classification and efficient pile defect detection.

B. Computation Offloading and Fog Computing

As a promising technology, fog computing aims to help mobile devices to operate tasks at the network edge directly [19], and is supposed to solve the ever-increasing computation in various scenarios of mobile applications [20]. For the IoT, fog computing is adopted to improve the quality of service [21]. And it is shown that fog computing can be useful to move the resources, such as storage and processing, closer to the data of IoT, instead of delivering the data to the cloud. The analytical model proves that the fog-capable devices can reduce the service delay effectively. On the other hand, fog computing has been applied into Internet of vehicle scenarios [22] to guarantee the low latency by delivering computational resources for vehicle-based fog nodes, so as to achieve a real-time traffic management. Besides the low latency, researchers also pay attention to other performance of the fog computing, such as energy

consumption. However, reducing energy consumption may lead to a larger execution delay [23]. Therefore, there exist lots of methods aiming to balance the latency and energy consumption. Aiming to achieve the balance, a multiobjective optimization model is built by queue theory, which describes the tradeoff among execution delay, energy consumption, and payment cost of offloading processes [23]. Moreover, Ahn *et al.* [24] propose a two-tier offloading method, which supports the device to use resources from either the fog or the cloud based on offloading decision. And the optimized decision is made according to the tradeoff between the low latency and energy savings. The simulation results illustrate how the performance varies along with parameters, and prove that the proposed method achieves a better performance than the cloud-only offloading method in terms of latency and energy consumption. Similarly, some studies adopt the advanced technique to improve the performance of the fog computing. For example, the social relationships of the energy harvesting techniques and various queue models are applied into the computational offloading system in fog computing, which can reduce the latency and energy consumption as well [20]. Considering the fog node is usually battery operated, Bozorgchenani *et al.* [25] present a suboptimal partial offloading method, which can generate a great impact on the network lifetime according to the simulation results. Moreover, since additional data communication caused by offloading from the cloud to the network edge may incur more overhead, Zhu *et al.* [19] design a task offloading policy for the fog computing model, which takes execution time, energy consumption, and other costs into account. Furthermore, based on the challenges of cloud computing, fog computing is designed to deal with the situation of unreliable connectivity to the cloud [26]. By studying a language independent remote procedure call middleware between the fog node and the droplet, which is the fog-aware small application, Saqib and Hamid [26] propose a reliable and lightweight architecture, called as FogR, for computation offloading, and illustrate that the fog network can be used for responding to an emergency in a smart traffic system with high reliability.

III. DEEPINS SYSTEM

In this paper, we propose a system named DeepIns to address the problem of industrial manufacture inspection. We mainly focus on two objectives, i.e., a deep model, which has a state-of-the-art classification performance while suiting for the fog computing environment. In the following two sections, we, respectively, introduce our solutions for these two tasks. In addition, notations are listed in Table I.

A. Deep Model

The system structure is shown in Fig. 2(a). As we can see, the system mainly consists of three components. First, we need visual sensors in each assembly line to watch the production status. The adopted cameras should be equipped with network interfaces, which usually are wired network adapter, for image uploading and some command transfer. The data are sent to local fog nodes to extract some lower level features and, if possible, output fast inference results for efficiency improvement and

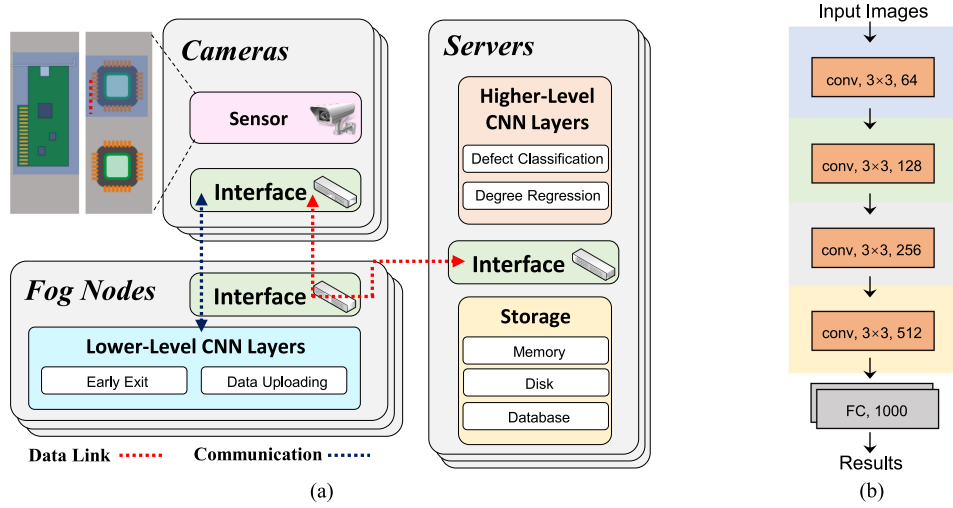


Fig. 2. Overview of the defect detection system. (a) System framework. (b) Model structure of the proposed deep learning network.

TABLE I
MAIN NOTATIONS

Notation	Description
\mathcal{X}	Set of input data
\mathcal{Y}	Set of ground-truth defect category labels
\mathcal{R}	Set of ground-truth defect degree labels
$h_{\theta}(x_i)$	Output of layer θ
\mathcal{W}	Weights of proposed network
\mathbf{W}_{θ}	Weight matrix of layer θ
\mathcal{B}	Biases of proposed network
\mathbf{B}_{θ}	Bias matrix of layer θ
L	Loss function of the deep model
L^e	The loss function for the exit point e
λ	Term weights for loss function units
γ	Term weights for the loss functions in exit points
TP	True positive
FP	False positive
TN	True negative
FN	False negative

traffic saving. This early-exit feature will be detailed in the next section. If the fog nodes cannot give accurate judgments by themselves, they will continue to upload the intermediate values of convolutional neural network (CNN) model inference to the remote cloud servers for further processing. The central servers have the ability to finish the entire CNN inference, conduct the classification and regression, and output the final results.

Fig. 2(b) presents the deep model we used in our system. There are four convolutional layers and two fully connected layers. This part is common and frequently seen in the other deep models, and we put the emphasis on the decision-making part, which is also the major difference. We connect two end modules in the top of the deep model, i.e., one classifier and one regressor. The classifier is to judge whether there are any defects in a production and to find the specific defect category, whereas the regressor is to infer the approximate degree of the possible defects. Then, the problem comes to how to design a loss function to jointly train the classifier and the regressor.

Similar to the works [27] and [28], a multitask loss, which can achieve two goals simultaneously, is presented as the loss function for our proposed deep model

$$L = \lambda_1 L_{\text{defect}} + \lambda_2 L_{\text{degree}} + \lambda_3 L_{\text{dec}} \quad (1)$$

where L_{defect} and L_{degree} are loss functions focused on different goals, and are introduced detailedly in the following. $J_{\text{dec}}(\mathcal{W})$ is a commonly used weight decay term and $L_{\text{dec}} = \|\mathcal{W}\|_2^2$. On the other hand, λ_1 , λ_2 , and λ_3 are predefined weights to describe the relative proportion of corresponding loss functions in the final loss value.

Specifically, L_{defect} indicates the loss value to judge the defect categories of the selected data samples, that is, to check whether or not there is a specific defect in current product. In addition, L_{defect} is built based on the softmax loss function [29].

Given n input data $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$, the set $\mathcal{Y} = \{y_{x_1}, y_{x_2}, \dots, y_{x_n}\}$ and $\mathcal{D} = \{r_{x_1}, r_{x_2}, \dots, r_{x_n}\}$ represented ground-truth labels and degrees, respectively, related to each input data x_i . In particular, $y \in \{y_1^*, y_2^*, \dots, y_m^*\}$ and d are the real values for the possible results

$$L_{\text{defect}} = -\frac{1}{n} \left[\sum_{i=1}^n \sum_{j=1}^m \mathbf{1}\{y_{x_i} = y_j^*\} \log \frac{e^{h_{\text{defect}}^{(j)}(x_i)}}{\sum_{\phi=1}^m e^{h_{\text{defect}}^{(\phi)}(x_i)}} \right]. \quad (2)$$

In the formula of L_{defect} , n is the number of input samples, and m indicates the total category number of the predefined defects, whereas i and j represent the ID of the input sample and specific defect, respectively. With the range from 0 to 1, $h_{\text{defect}}^{(j)}(x_i)$ varies along with the possibilities that defects in the image are involved in a specific defect category. Besides, $h_{\text{defect}}(x_i)$ represents the output of softmax classification layer, and $h_{\text{degree}}(x_i)$ represents the output of smooth L_1 localization layer.

L_{degree} can be used to predict the degree of the detected defect. Essentially, it is a regression function. In this paper, the minimum rank is set to 1, representing the slightest defect degree. According to the calculation of smooth L_1 loss in fast

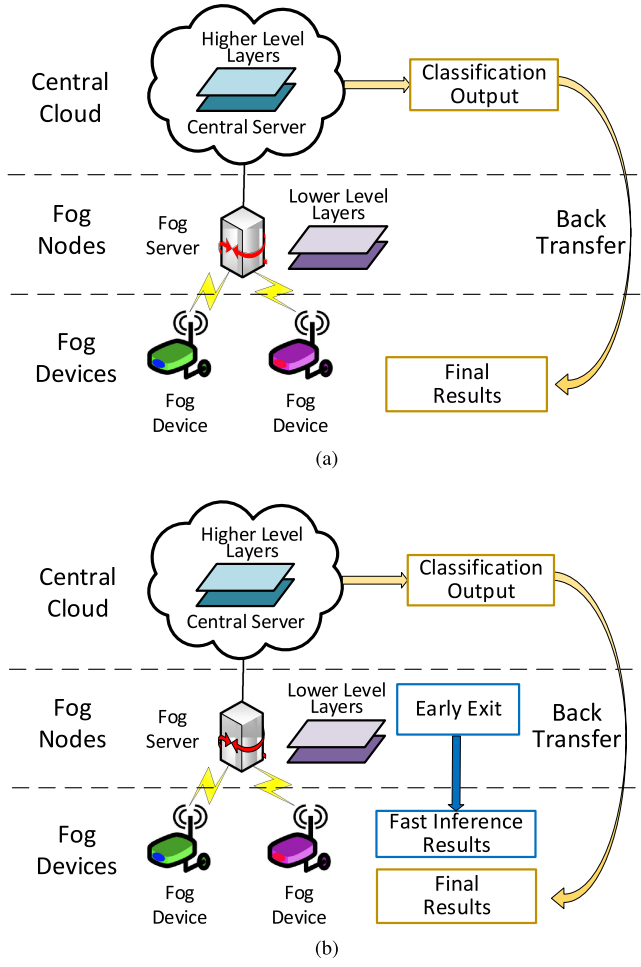


Fig. 3. Deep learning structure in fog computing scenarios. (a) Our past work, which allocates different parts of the deep models to different devices for higher efficiency. (b) Newly proposed structure with the ability of early exit, which can significantly decrease the communication needs.

R-CNN [30], L_{degree} is defined as follows:

$$L_{\text{degree}} = \frac{1}{n} \sum_{i=1}^n \text{smooth}_{L_1} \left(h_{\text{degree}}^{(d)}(x_i) - d_{x_i} \right) \quad (3)$$

and

$$\text{smooth}_{L_1}(z) = \begin{cases} 0.5z^2, & |z| < 1 \\ |z| - 0.5, & |z| \geq 1 \end{cases} \quad (4)$$

B. Offloading Strategy

There are several different offloading strategies for the fog computing scenarios. In our past work [31], we have implemented an offloading method to transfer the deep learning computation from the end nodes to the central cloud. The main concept is to divide the models by layers, which are allocated to different devices to average the computing burden. This design is shown in Fig. 3(a). As we can see, the lower layers of the deep model are processed by the fog nodes and fog servers, as a whole, and the intermediate values will be uploaded to the cloud servers through network communications for further

computing. The central cloud servers will perform the classification and output the final results, which will be transferred back to the fog nodes. After this, the fog nodes, the users in other words, can finally get the classification results.

This strategy is simple yet effective. With this method, we significantly improve the running efficiency of the inference process. However, we find several problems in this strategy. First, the users cannot get the classification results until the final output is transferred back to these edge devices. This will cause severe performance problem when the network latency is nonignorable. Second, the intermediate values should be uploaded to the central cloud servers for further computation. This process results in huge network traffic because the extracted features are still in a big size.

To address these problems, we improve our strategy and design a new deep learning framework in Fig. 3(b) for the fog computing environments. The difference is that we adopt an early-exit feature [32] of the deep models as a way to allocate the lower parts to the fog devices while ensuring they can obtain early results without waiting for the results from the central servers. If the fog devices have obtained some results that are able to meet their requirements, they can just stop uploading intermediate values and adopt the fast inference as the final results.

In order to empower the parallel deep model with the ability of early exit, we should modify its structure in several aspects. The first problem is how to train the network in the fog computing environment. In essence, this problem is how to adapt the existing loss functions to multiple classifiers (or regressors).

With (1), we can get the loss function for an early-exit branch e , as shown in the following:

$$L^e = \lambda_1 L_{\text{defect}}^e + \lambda_2 L_{\text{degree}}^e + \lambda_3 L_{\text{dec}}^e \quad (5)$$

where L^e is the loss function for the exit point e , and L_{defect}^e and L_{degree}^e are the sub-loss-functions of exit e , respectively, for category classification and degree regression.

According to (2) and (3), we can obtain modified functions for L_{defect}^e and L_{degree}^e as follows:

$$L_{\text{defect}}^e = -\frac{1}{n} \left[\sum_{i=1}^n \sum_{j=1}^m \mathbf{1}\{y_{x_i} = y_j^*\} \log \frac{e^{h_{\text{defect},e}^{(j)}(x_i)}}{\sum_{\phi=1}^m e^{h_{\text{defect},e}^{(\phi)}(x_i)}} \right] \quad (6)$$

$$L_{\text{degree}}^e = \frac{1}{n} \sum_{i=1}^n \text{smooth}_{L_1} \left(h_{\text{degree},e}^{(d)}(x_i) - d_{x_i} \right) \quad (7)$$

where $h_{\text{defect},e}^{(\phi)}(x_i)$ varies from 0 to 1, according to the possibilities that the defects in the image belong to one specific defect category and $h_{\text{degree},e}^{(d)}(x_i)$ represents the inferred value of the possible defect degree. $h_{\text{defect},e}^{(\phi)}(x_i)$ and $h_{\text{degree},e}^{(d)}(x_i)$ are, respectively, computed by the classifier and regressor of early exit e .

Assume that there are E early exits in the adopted network; then, we can get the following equation:

$$L_{\text{all}} = \sum_{e=1}^E \gamma_e L^e \quad (8)$$

where L^e is the loss function for exit e and γ represents their weights.

Weight γ is defined according to the inclination and experiences. Giving the early exit a higher weight will make the network try to learn some discriminate features in their lower level layers, which can effectively increase the fast-inference accuracy and, in this case, the fog nodes need not to upload the intermediate values, resulting in faster response and lower network traffic. However, the absence of the higher level layers does have some inference on the performance. Setting the final exit a higher weight will move the emphasis of the network to the improvement of the final results again. But, at the same time, the transferring of the intermediate values also bring larger communication consumption, which of course has some effects on the system throughput. In industrial application, the weight γ should be decided according to the actual situation. A higher value can be assigned to the final classifiers/regressors, if both the fog nodes and network conditions are capable of handling heavy burdens; otherwise, leaving the higher values to the lower exits for higher efficiency.

At the training process, we adopt the stochastic gradient descent (SGD) algorithm, which is traditional yet effective, rather than the recently risen adaptive optimization methods, such as RMSProp, Adam, etc. The reason is that, although SGD algorithm may be outperformed by the adaptive optimization methods in the training speed, SGD shows a significant advantage in their generalization ability [33]. In other words, SGD can lead to a well-trained model with better accuracy for actual applications.

Another problem in enabling the deep model with fog computing support is how to exchange data between fog nodes and central servers, in what format, and how to decrease the network traffic. In our past work [31], we simply upload the intermediate values of the lower level layers, which are in the local fog nodes, to the higher level layers, which are in the cloud servers. This strategy works well with a regular data size that is not very large. However, it may encounter several problems if the processing queue increases quickly and each fog node is full of products waiting to be processed, which is a common scene in actual industrial scenarios. In this paper, we use an extra max pooling layer to concatenate the feature, as shown in Fig. 4, which can also effectively decrease the data size of the intermediate values.

One remaining question is in what situations the early exit can happen and in what situations cannot. One obvious solution is to set a threshold T_{loss} for the maximum acceptable loss value. If the loss value of one exit point is less than T_{loss} , then the inference process can be terminated and its inference results will be regarded as the final output. Otherwise, the inference should continue and the intermediate values will be uploaded to central servers for further computation.

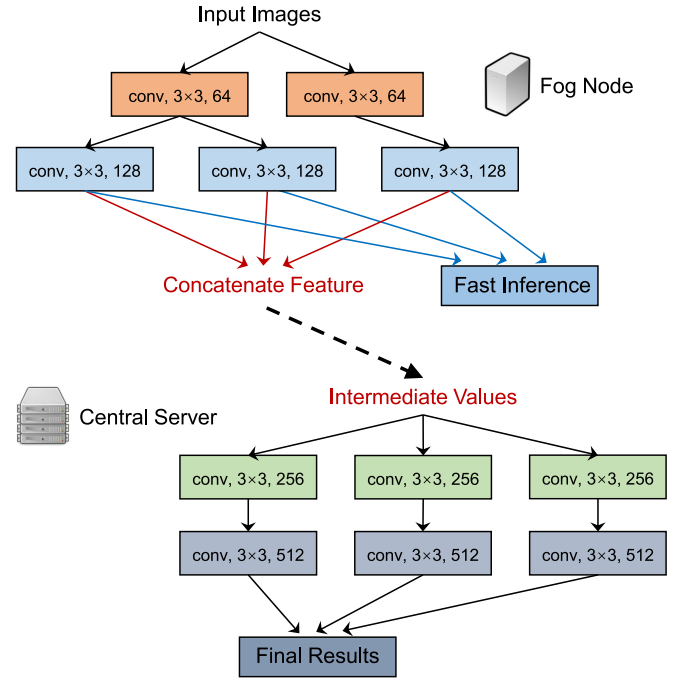


Fig. 4. Data exchange between submodels, which are, respectively, deployed in the fog node and central servers.

With the above features, the proposed DeepIns system can be directly used in fog environments with well-trained parameters. Although the training process is time consuming, the model inference is not. With the support of fog computing in DeepIns system, the running efficiency can be further improved and can meet the requirement of real-time manufacture lines with very high production outputs. In Section IV, we will test both of the recognition accuracy and system efficiency.

IV. PERFORMANCE EVALUATION

Several experiments are conducted in this section to evaluate the performance of the proposed defect detection system. At first, we perform a comparison experiment of the defect detection.

We manually capture a dataset using some tile production as the data source for network training and testing. We label these data with several different categories for different defects. We also classify these defects into conforming or nonconforming products with specific values for their defect degrees. The dataset consists of ten categories, each of which has 200 images for training process and 50 testing images for network testing.

As the convention in defect detection area, we use the receiver operating characteristic (ROC) curve for the comparison between our method and the existing approaches. The ROC curve, which is also known as a relative operating characteristic curve, is plotted by a comparison of the true positive rate (TPR) and the false positive rate (FPR) with various detection methods. As functions of the classifier parameter, TPR against FPR can illustrate the diagnostic ability of a binary classifier system, where TPR explains how many correct positive results

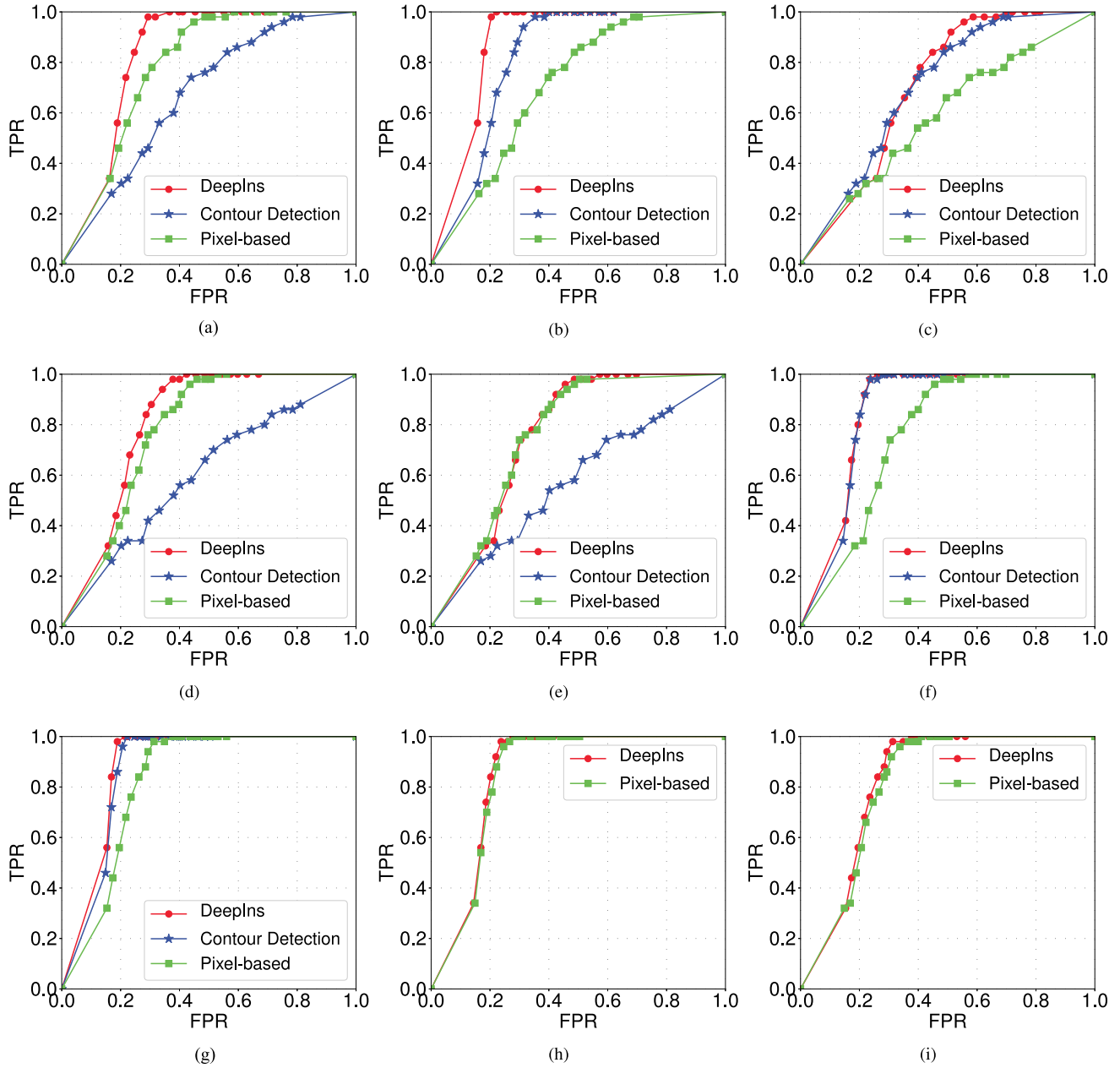


Fig. 5. Comparison results of the defect detection experiments. The defects are as follows: (a) Blob. (b) Corner. (c) Edge. (d) Spot. (e) Pinhole. (f) Crack. (g) Line. (h) Saturation. (i) Brightness.

occur among the positive sample space and FPR explains how many incorrect positive results occur among the negative sample space. Specifically, TPR and FPR are expressed as follows:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (9)$$

$$\text{FPR} = \frac{\text{FP}}{\text{TN} + \text{FP}}. \quad (10)$$

We compare the proposed DeepIns method with two famous methods in the defect detection area, namely, the contour detection approach and the pixel-based method. The former one first extracts the contour information from the raw input images, and then judges their categories with some contour classifiers.

The latter one defines some pixel-based features to represent the input data while decreasing its size, and then trains a classifier using the combination of these pixel-based features.

The ROC curves of these methods on the product image data are shown in Fig. 5, where both TPR and FPR vary from 0 to 1. Among all these scenarios, it can be seen that the ROC curves increase rapidly at the beginning, that is, when $\text{FPR} \in (0, 0.3)$. Then, the ROC curves come into a period of relatively steady growth. And the values of the ROC curves become 1 eventually. Consistent with our above-mentioned theoretical analysis, the proposed method shows the efficiency in various scenarios, and outperforms both the contour detection method and the pixel-based method.

TABLE II
RUNNING EFFICIENCY

Number of Tiles	With DeepIns	Local Computation
100	62.11	136.91
200	138.46	265.87
300	197.91	397.12
400	265.37	523.76
500	321.18	651.61
600	398.15	782.18

In addition, we conduct another simulation on the running efficiency, as shown in Table II. We can see that with fog support, the running efficiency has been significantly improved, which makes a real-time manufacture inspection system possible.

V. CONCLUSION

In this paper, we propose a manufacture inspection system for the smart industry. This system adopts the state-of-the-art deep models to analyze the input images captured by the sensors and find the defective products with particular values to indicate the degree of the defects. In order to implement a real-time system capable of dealing with a large size of data, we design the proposed system on the basis of fog computing, which can offload the computation burden to the fog nodes and significantly decrease the overload of the central servers. The simulations prove that our system is robust and efficient, and can outperform some existing approaches in the tests.

In the future, we plan to perform the classification and regression experiments on some other kinds of productions. Also, we will continue to improve the offloading strategy to bring multiple fog devices into deep model computation, simultaneously, which can further enhance the running efficiency of fog deep leaning applications.

REFERENCES

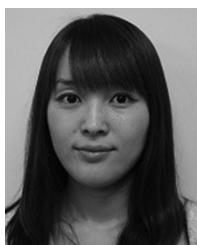
- [1] J. Wan *et al.*, "A manufacturing big data solution for active preventive maintenance," *IEEE Trans. Ind. Informat.*, vol. 13, no. 4, pp. 2039–2047, Aug. 2017.
- [2] Z. Bi, L. D. Xu, and C. Wang, "Internet of things for enterprise systems of modern manufacturing," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1537–1546, May 2014.
- [3] L. D. Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Trans. Ind. Informat.*, vol. 10, no. 4, pp. 2233–2243, Nov. 2014.
- [4] Q. Zhang, L. T. Yang, Z. Yan, Z. Chen, and P. Li, "An efficient deep learning model to predict cloud workload for industry informatics," *IEEE Trans. Ind. Informat.*, to be published, doi: [10.1109/TII.2018.2808910](https://doi.org/10.1109/TII.2018.2808910).
- [5] K. Ota, M. S. Dao, V. Mezaris, and F. G. B. D. Natale, "Deep learning for mobile multimedia: A survey," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 13, no. 3s, pp. 34:1–34:22, Jun. 2017. [Online]. Available: <http://doi.acm.org/10.1145/3092831>
- [6] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "A tensor-train deep computation model for industry informatics big data feature learning," *IEEE Trans. Ind. Informat.*, to be published, doi: [10.1109/TII.2018.2791423](https://doi.org/10.1109/TII.2018.2791423).
- [7] H. Harb and A. Makhoul, "Energy-efficient sensor data collection approach for industrial process monitoring," *IEEE Trans. Ind. Informat.*, vol. 14, no. 2, pp. 661–672, Feb. 2018.
- [8] K. Ota, M. Dong, J. Gui, and A. Liu, "QUON: Incentive mechanisms for crowd sensing networks," *IEEE Netw.*, vol. 32, no. 2, pp. 114–119, Mar. 2018.
- [9] X. Tao, K. Ota, M. Dong, H. Qi, and K. Li, "Performance guaranteed computation offloading for mobile-edge cloud computing," *IEEE Wireless Commun. Lett.*, vol. 6, no. 6, pp. 774–777, Dec. 2017.
- [10] J. Fu, Y. Liu, H. C. Chao, B. Bhargava, and Z. Zhang, "Secure data storage and searching for industrial IoT by integrating fog computing and cloud computing," *IEEE Trans. Ind. Informat.*, to be published, doi: [10.1109/TII.2018.2793350](https://doi.org/10.1109/TII.2018.2793350).
- [11] A. Nurhadiyatna *et al.*, "Development of visual inspection system for detecting surface defects on sensor chip," in *Proc. Int. Conf. Comput., Control, Informat. Appl.*, Oct. 2017, pp. 100–105.
- [12] E. Prakasa, E. Kurniawan, A. Nurhadiyatna, L. Musa, and P. Reidler, "Implementation on 3D surface algorithm for measuring thickness parameter of sensor chip," in *Proc. Int. Symp. Intell. Signal Process. Commun. Syst.*, Nov. 2015, pp. 199–203.
- [13] X. Gibert, V. M. Patel, and R. Chellappa, "Deep multitask learning for railway track inspection," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 1, pp. 153–164, Jan. 2017.
- [14] A. Abedini and M. Ehsanian, "Defect detection on IC wafers based on neural network," in *Proc. 29th Int. Conf. Microelectron.*, Dec. 2017, pp. 1–4.
- [15] O. Semeniuta, S. Dransfeld, and P. Falkman, "Vision-based robotic system for picking and inspection of small automotive components," in *Proc. IEEE Int. Conf. Autom. Sci. Eng.*, Aug. 2016, pp. 549–554.
- [16] X. Wu, H. Xiong, Z. Yu, and P. Wen, "A surface defect detection method based on multi-feature fusion," in *Proc. 9th Int. Conf. Digit. Image Process.*, 2017, vol. 10420, Art. no. 104200S.
- [17] A. Abedini, A. Miri, and A. Maleki, "Parallel improved pulse coupled neural network application for edge detection in image processing," *J. Comput. Eng. Inf. Technol.*, vol. 6, no. 2, 2017, doi: [10.4172/2324-9307.1000170](https://doi.org/10.4172/2324-9307.1000170).
- [18] K. Weixin, Z. Yuchen, and L. Lingxi, "Pile defect detection based on wavelet packet energy ratio and support vector machine," in *Proc. 13th IEEE Int. Conf. Electron. Meas. Instrum.*, Oct. 2017, pp. 92–97.
- [19] Q. Zhu, B. Si, F. Yang, and Y. Ma, "Task offloading decision in fog computing system," *China Commun.*, vol. 14, no. 11, pp. 59–68, Nov. 2017.
- [20] L. Liu, Z. Chang, and X. Guo, "Socially-aware dynamic computation offloading scheme for fog computing system with energy harvesting devices," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1869–1879, Jun. 2018.
- [21] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, "On reducing IoT service delay via fog offloading," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 998–1010, Apr. 2018.
- [22] X. Wang, Z. Ning, and L. Wang, "Offloading in internet of vehicles: A fog-enabled real-time traffic management system," *IEEE Trans. Ind. Informat.*, to be published, doi: [10.1109/TII.2018.2816590](https://doi.org/10.1109/TII.2018.2816590).
- [23] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, "Multiobjective optimization for computation offloading in fog computing," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 283–294, Feb. 2018.
- [24] S. Ahn, M. Gorlatova, and M. Chiang, "Leveraging fog and cloud computing for efficient computational offloading," in *Proc. IEEE MIT Undergraduate Res. Technol. Conf.*, Nov. 2017, pp. 1–4.
- [25] A. Bozorgchenani, D. Tarchi, and G. E. Corazza, "An energy and delay-efficient partial offloading technique for fog computing architectures," in *Proc. IEEE Global Commun. Conf.*, Dec. 2017, pp. 1–6.
- [26] M. T. Saqib and M. A. Hamid, "FogR: A highly reliable and intelligent computation offloading on the internet of things," in *Proc. IEEE Region 10 Conf.*, Nov. 2016, pp. 1039–1042.
- [27] S. Song and J. Xiao, "Deep sliding shapes for amodal 3D object detection in RGB-D images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 808–816.
- [28] L. Li, K. Ota, M. Dong, and W. Borjigin, "Eyes in the dark: Distributed scene understanding for disaster management," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 12, pp. 3458–3471, Dec. 2017, doi: [10.1109/TPDS.2017.2740294](https://doi.org/10.1109/TPDS.2017.2740294).
- [29] W. Liu, Y. Wen, Z. Yu, and M. Yang, "Large-margin softmax loss for convolutional neural networks," in *Proc. 33rd Int. Conf. Int. Conf. Mach. Learn.*, 2016, vol. 48, pp. 507–516. [Online]. Available: <http://dl.acm.org/citation.cfm?id=3045390.3045445>
- [30] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, Washington, DC, USA, 2015, pp. 1440–1448. [Online]. Available: <http://dx.doi.org/10.1109/ICCV.2015.169>
- [31] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the internet of things with edge computing," *IEEE Netw.*, vol. 32, no. 1, pp. 96–101, Jan. 2018.
- [32] S. Teerapittayanon, B. McDanel, and H. T. Kung, "BranchyNet: Fast inference via early exiting from deep neural networks," in *Proc. 23rd Int. Conf. Pattern Recognit.*, Dec. 2016, pp. 2464–2469.
- [33] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht, "The marginal value of adaptive gradient methods in machine learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4151–4161.



Liangzhi Li (S'17) received the B.Sc. and M.Eng. degrees in computer science from South China University of Technology, Guangzhou, China, in 2012 and 2016, respectively. He is currently working toward the Ph.D. degree in electrical engineering at Muroran Institute of Technology, Muroran, Japan.

His research interests include machine learning, big data, robotics, and smart grid.

Mr. Li is the recipient of the Best Paper Award from FCST 2017.



Kaoru Ota (M'12) was born in Aizu-Wakamatsu, Japan. She received the M.S. degree in computer science from Oklahoma State University, Stillwater, OK, USA, in 2008, and the B.S. and Ph.D. degrees in computer science and engineering from The University of Aizu, Aizuwakamatsu, Japan, in 2006 and 2012, respectively.

She is currently an Assistant Professor with the Department of Information and Electronic Engineering, Muroran Institute of Technology, Muroran, Japan. Her research interests include

wireless networks, cloud computing, and cyber-physical systems.

Dr. Ota is the recipient of the Best Paper Awards from ICA3PP 2014, GPC 2015, IEEE DASC 2015, IEEE VTC 2016-Fall, FCST 2017, and 2017 IET Communications Premium Award. She is an Editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, the IEEE COMMUNICATIONS LETTERS, *Peer-to-Peer Networking and Applications* (Springer), *Ad Hoc and Sensor Wireless Networks*, *International Journal of Embedded Systems* (Inderscience), and *Smart Technologies for Emergency Response and Disaster Management* (IGI Global), as well as a Guest Editor for *ACM Transactions on Multimedia Computing, Communications and Applications* (leading), the IEEE INTERNET OF THINGS JOURNAL, the IEEE ACCESS, the IEEE COMMUNICATIONS MAGAZINE, the IEEE NETWORK, the IEEE WIRELESS COMMUNICATIONS (2015), *IEICE Transactions on Information and Systems* (2014), and *Ad Hoc and Sensor Wireless Networks* (Old City Publishing) (2014). She is also the recipient of the IEEE TCSC Early Career Award 2017.



Mianxiong Dong (M'13) received the B.S., M.S., and Ph.D. degrees in computer science and engineering from The University of Aizu, Aizuwakamatsu, Japan, in 2006, 2008, and 2013, respectively.

He is currently an Associate Professor with the Department of Information and Electronic Engineering, Muroran Institute of Technology, Muroran, Japan. His research interests include wireless networks, cloud computing, and cyber-physical systems.

Dr. Dong is the recipient of the Best Paper Awards from IEEE HPCC 2008, IEEE ICSS 2008, ICA3PP 2014, GPC 2015, IEEE DASC 2015, IEEE VTC 2016-Fall, FCST 2017, and 2017 IET Communications Premium Award. He is an Editor for the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING, the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, the IEEE NETWORK, the IEEE WIRELESS COMMUNICATIONS LETTERS, the IEEE CLOUD COMPUTING, and the IEEE ACCESS. He has been serving as the Vice Chair of the IEEE Communications Society Asia/Pacific Region Meetings and Conference Committee, Leading Symposium Chair of the IEEE ICC 2019, Student Travel Grants Chair of the IEEE GLOBECOM 2019, and Symposium Chair of the IEEE GLOBECOM 2016 and 2017. He is also the recipient of the IEEE TCSC Early Career Award 2016, IEEE SCSTC Outstanding Young Researcher Award 2017, and the 12th IEEE ComSoc Asia-Pacific Young Researcher Award 2017.