Deep Learning for Smart Industry:Efficient Manufacture Inspection Systemwith Fog Computing

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

Liangzhi Li, *Student Member, IEEE,* Kaoru Ota, *Member, IEEE,* Mianxiong Dong, *Member, IEEE*

***Abstract*—With the rapid development of Internet of Things (IoT) 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 the 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 (CNN) 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*—Fog computing, manufacture inspection, smart industry, deep learning.**

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 human-like 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 work, 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 the 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 be uploaded to the fog nodes for further analysis, and the fog node will use the installed inspection model to find the possible defects and give the results to the central server. Ultimately, the central server can collect all the necessary data in order to give recommendations and feedback on current production status.

The main contributions of our work include:

- We propose a multi-branch deep model for the manufacture inspection application. The proposed model can effectively find the potential defects and measure their degrees.

Authors are with the Department of Information and Electronic Engineering, Muroran Institute of Technology, Japan.
E-mail:{16096502, ota, mxdong}@mmm.muroran-it.ac.jp

Manuscript received xx xx, 2017; revised xx xx, 2017.

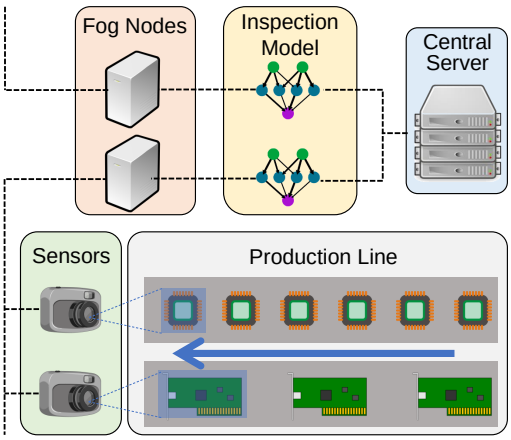

Fig. 1. Fog computing based manufacture inspection for the smart industry. Each fog node is able to check the status of the productions, using the deployed deep learning model, and upload the results or intermediate values to the central server.

- We make some modifications to the common deep model structure and adapt it to the fog computing environments. Compared with existing works, the proposed structure has the advantages of faster response and lower network traffic.
- We implement an actual fog system as the experimental environment and conduct several comparison experiments with a local deep learning system. We also compare the recognition performance with traditional approaches.

Section II introduces the existing research in related area. Section III presents some notations and the problem definition, and gives an overview of the proposed system. Also, in this section, we give a detailed introduction regarding the inspection method, and proposes the implementation of the fog system for the smart industry. Experiments are presented in Section IV. Finally, Section V gives the conclusions.

II. Related Works

A. Manufacture Inspection

In order to accurately inspect and assess the quality of products, some studies present various visual inspection approaches to detect product defects in a large scale production. In [11], a contour finding method based on image processing is proposed, which uses image filters and can detect surface defects by pixel chip pads. As a tiny sensor, the pixel chip is a significant part for detecting defects. In the image processing, three steps are used to achieve the goal. In the first step, the surface pad is segmented into areas by K-Means clustering. Then the areas are extracted by Gabor filter, and Canny Edge filter detects defects of the surface in the last step. Similarly, the feature of sensor chips is measured with a 3D surface algorithm proposed in [12]. In the algorithm, 3D scanner applies various methods such as triangulation and interferometry, and is aimed to scan the chip surfaces. Moreover, thousands of sensors are adopted in the system to collect depth information of chip surface, thereby measuring flatness and thickness of the chip surface, which are the main indications of chip quality. According to the experimental results, the proposed algorithm can improve production efficiency and avoid critical damage. However, the surface defects cannot be detected completely in the research.

Modern methods use deep learning mechanism to realize more reliable and more efficient inspection in more complicated scenarios than traditional methods mentioned above. And whether the method has the learning ability is the crucial distinction between modern methods and traditional methods. Till now, challenges, such as full automation and special products, are still the obstacle for product inspection. However, it has been verified that modern methods can deal with challenges better than traditional methods while reducing human errors. For instance, modern methods can provide more frequent inspections by computer vision and pattern recognition, based on which, [13] proposes a concept to combine multiple detectors within a multitask learning framework. The method is used to detect railroad tracks, which need to be monitored periodically to guarantee safe transportation. In this scenario, railway material needs to be identified in inspection, and only a small number of training samples is available for learning an anomaly detector. Therefore, authors in [13] adopt multiple detectors to improve accuracy for detecting defects on components of the railway. Moreover, a task can be beneficial from the knowledge, which has been learned by others. Therefore, it is shown that the combined system, i.e. the multi-task learning framework, can obtain better performance than conducting each task separately.

Generally, it has to analyze shape features at the first place in the detection process. The shape defects can be classified into regular shape defects and irregular shape defects, in which irregular shape defects are the complicated problem since they are not the regular pattern [14]. As for special products, whose surface is partially reflective or components are small, it may cause much overhead and reduce the reliability of inspection in traditional methods. Yet modern methods have been applied widely for detecting irregular shape defects. For example, [15] introduces a method of multi-camera/multi-pose inspection station to tackle noise and support advanced image analysis, and proposes the first functional prototype implementation of the method. The method presents the effective automated visual inspection by detecting star washers. Due to the small size of components and the inherent reflectivity of products, challenges need to be addressed in the vision-based inspection. [15] describes the strategy for dealing with the challenges. 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, [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, sub-image 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. Authors in [18] use multi-layer Support Vector Machine (SVM) 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 Internet of Things (IoT), fog computing is adopted to improve the quality of service (QoS) [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 (IoV) 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 multi-objective optimization model is built by queue theory, which describes the trade-off among execution delay, energy consumption and payment cost of offloading processes [23]. Moreover, Surin 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 trade-off 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, Arash 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, Qiliang 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 could [26]. By studying a language independent Remote Procedure Call middleware between the fog node and the droplet, which is the fog-aware small application, Md. Tanzim Saqib *et. al* [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.

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
$\boldsymbol{h_\theta(x_i)}$	Output of layer θ
$\mathcal{W}$	Weights of proposed network
$\boldsymbol{W_\theta}$	Weight matrix of layer θ
$\mathcal{B}$	Biases of proposed network
$\boldsymbol{B_\theta}$	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

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.

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

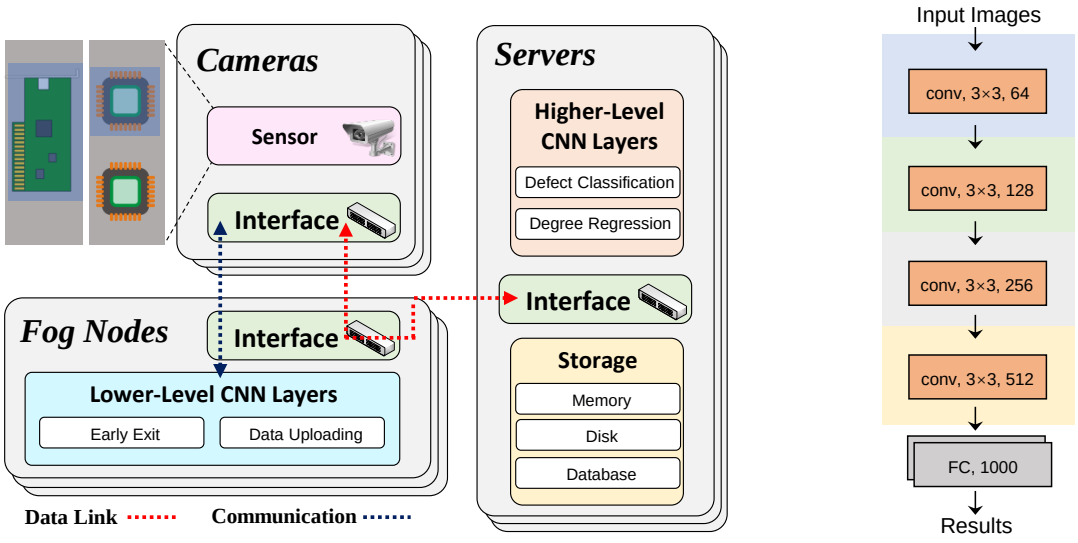

Fig. 2. Overview of the defect detection system. (a) System framework. (b) The model structure of the proposed deep learning network.

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 traffic saving. This early exit feature will be detailed in next section. If the fog nodes cannot give accurate judgments by themselves, they will continue to upload the intermediate values of 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, while 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], [28], a multi-task loss, which can achieve two goals simultaneously, is presented as the loss function for our proposed deep model,

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

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

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

In the formula of L_{defect}, n is the number of input samples, and m indicates the total category number of the pre-defined defects, while 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, $\boldsymbol{h_{\text{defect}}(x_i)}$ represents the output of Softmax classification layer, and $\boldsymbol{h_{\text{degree}}(x_i)}$ 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 our paper, the minimum rank is set to 1, representing the slightest defect degree. According to the calculation of smooth L_1 loss in Fast R-CNN [30], L_{degree} is defined as follows.

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

and

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

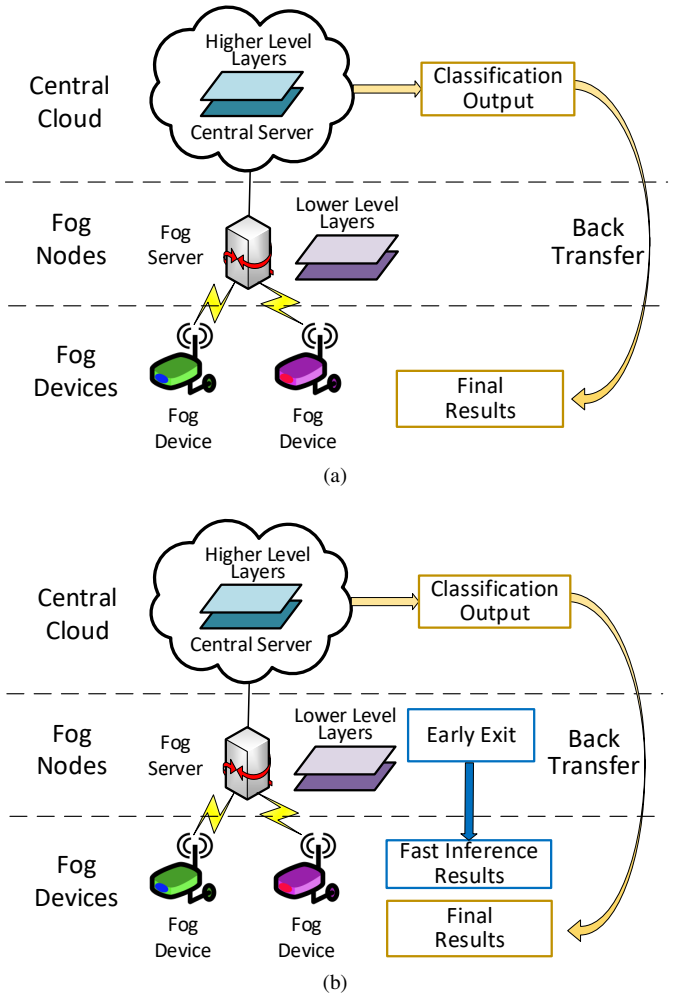

Fig. 3. The deep learning structure in fog computing scenarios. (a) Our past work which allocates different parts of the deep models to different devices for higher efficiency. (b) The newly-proposed structure with the ability of early exit, which can significantly decrease the communication needs.

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 non-ignorable. 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 which 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 Eq. (1), we can get the loss function for an early-exit branch e, as shown below.

$$L^e = \lambda_1 L^e_{\text{defect}} + \lambda_2 L^e_{\text{degree}} + \lambda_3 L^e_{\text{dec}} \tag{5}$$

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

According to Eq. (2)(3), we can obtain modified functions for L^e_{defect} and L^e_{degree}, as follows

$$L^e_{\text{defect}} = -\frac{1}{n}\left[\sum_{i=1}^{n}\sum_{j=1}^{m}\mathbf{1}\{y_{x_i} = y^*_j\}\log\frac{e^{h^{(j)}_{\text{defect},e}(x_i)}}{\sum_{\phi=1}^{m} e^{h^{(\phi)}_{\text{defect},e}(x_i)}}\right] \tag{6}$$

$$L^e_{\text{degree}} = \frac{1}{n}\sum_{i=1}^{n}\text{smooth}_{L_1}(\boldsymbol{h^{(d)}_{\text{degree},e}(x_i)} - \boldsymbol{d_{x_i}}) \tag{7}$$

where $h^{(\phi)}_{\text{defect},e}(x_i)$ varies from 0 to 1, according to the possibilities that the defects in the image belong to one specific defect category and $h^{(d)}_{\text{degree},e}(x_i)$ represents the inferred value of the possible defect degree. $h^{(\phi)}_{\text{defect},e}(x_i)$ and $h^{(d)}_{\text{degree},e}(x_i)$ 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

$$L_{all} = \sum_{e=1}^{E}\gamma_e L^e \tag{8}$$

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

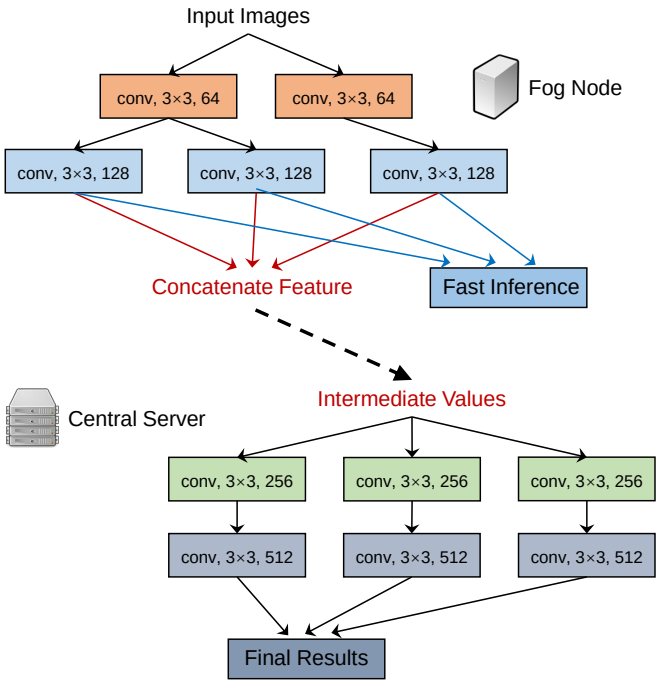

Fig. 4. The data exchange between sub-models, which are respectively deployed in the fog node and central servers.

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 which 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 work, 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 can not. 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.

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 the next section, 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 non-conforming 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 TRP explains how many correct positive results 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:

$$TPR = \frac{TP}{TP + FN} \tag{9}$$

$$FPR = \frac{FP}{TN + FP} \tag{10}$$

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

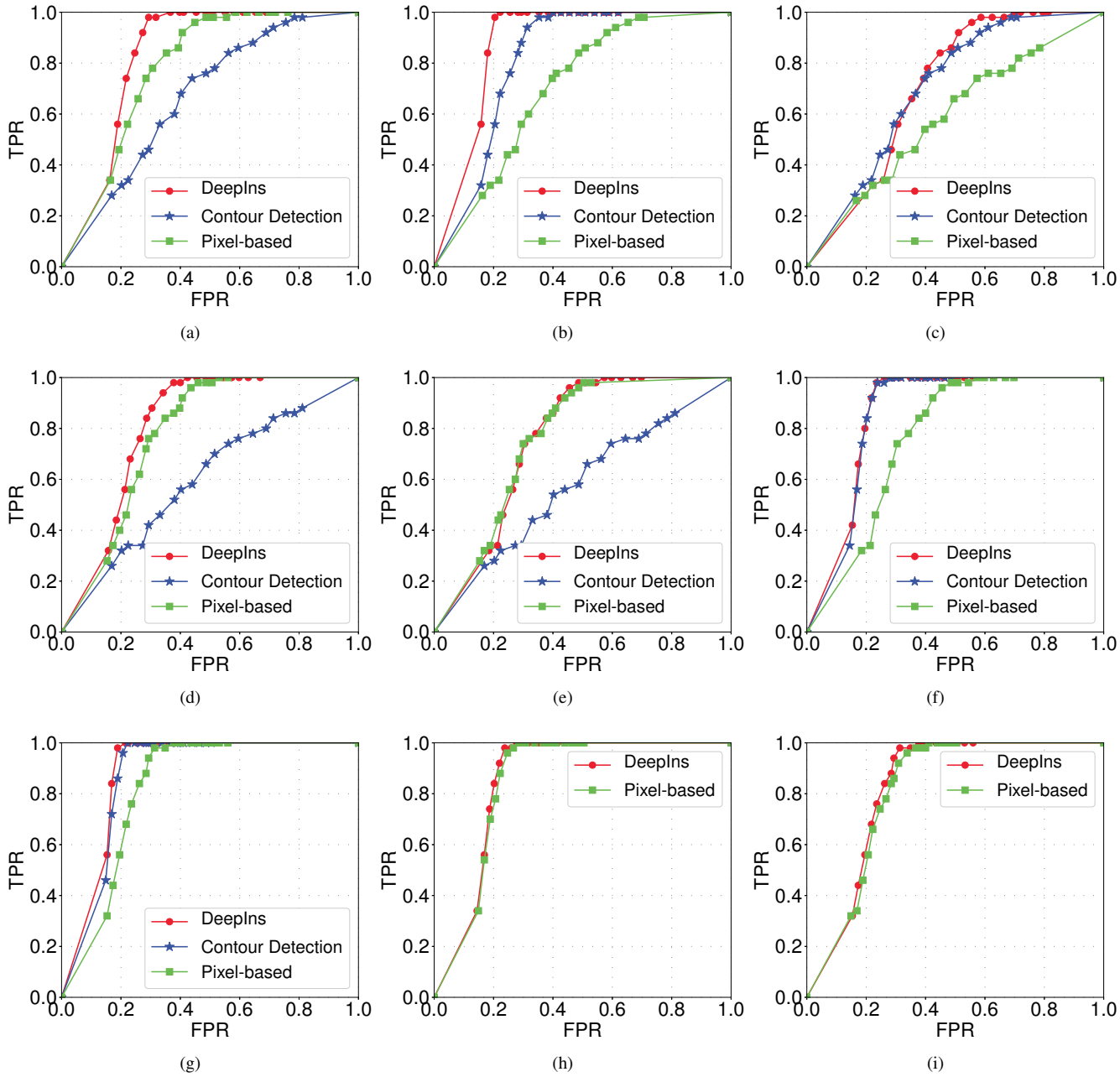

Fig. 5. The comparison results of the defect detection experiments. The defects are as follows: (a)Blob. (b)Corner. (c)Edge. (d)Spot. (e)Pinhole. (f)Crack. (g)Line. (h)Saturation. (i)Brightness.

represent the input data while decreasing its size, 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 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 theoretical analysis above, the proposed method shows the efficiency in various scenarios, and outperforms both the contour detection method and pixel-based method.

In addition, we conduct another simulation on the running efficiency. 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

TABLE II
THE 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

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&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.

ACKNOWLEDGMENT

This work is partially supported by JSPS KAKENHI Grant Number JP16K00117 and KDDI Foundation. Mianxiong Dong is the corresponding author.

REFERENCES

[1] J. Wan, S. Tang, D. Li, S. Wang, C. Liu, H. Abbas, and A. V. Vasilakos, "A manufacturing big data solution for active preventive maintenance," *IEEE Transactions on Industrial Informatics*, 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 Transactions on Industrial Informatics*, 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 Transactions on Industrial Informatics*, 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 Transactions on Industrial Informatics*, pp. 1–1, 2018.

[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 Transactions on Industrial Informatics*, pp. 1–1, 2018.

[7] H. Harb and A. Makhoul, "Energy-efficient sensor data collection approach for industrial process monitoring," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 2, pp. 661–672, Feb 2018.

[8] K. Ota, M. Dong, J. Gui, and A. Liu, "Quoin: Incentive mechanisms for crowd sensing networks," *IEEE Network*, vol. 32, no. 2, pp. 114–119, March 2018.

[9] X. Tao, K. Ota, M. Dong, H. Qi, and K. Li, "Performance guaranteed computation offloading for mobile-edge cloud computing," *IEEE Wireless Communications Letters*, 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 Transactions on Industrial Informatics*, pp. 1–1, 2018.

[11] A. Nurhadiyatna, S. Loncaric, E. Prakasa, E. Kurniawan, A. A. Khoirudin, L. Musa, and P. Reidler, "Development of visual inspection system for detecting surface defects on sensor chip," in *2017 International Conference on Computer, Control, Informatics and its Applications (IC3INA)*, 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 *2015 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, Nov 2015, pp. 199–203.

[13] X. Gibert, V. M. Patel, and R. Chellappa, "Deep multitask learning for railway track inspection," *IEEE Transactions on Intelligent Transportation Systems*, 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 *2017 29th International Conference on Microelectronics (ICM)*, 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 *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, 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 *Ninth International Conference on Digital Image Processing (ICDIP 2017)*, vol. 10420. International Society for Optics and Photonics, 2017, p. 104200S.

[17] A. Abedini, A. Miri, and A. Maleki, "Parallel improved pulse coupled neural network application for edge detection in image processing," *Computer Engineering and Information Technology*, vol. 06, no. 2, 2017.

[18] K. Weixin, Z. Yuchen, and L. Lingxi, "Pile defect detection based on wavelet packet energy ratio and support vector machine," in *2017 13th IEEE International Conference on Electronic Measurement Instruments (ICEMI)*, Oct 2017, pp. 92–97.

[19] Q. Zhu, B. Si, F. Yang, and Y. Ma, "Task offloading decision in fog computing system," *China Communications*, 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 of Things Journal*, vol. PP, no. 99, pp. 1–1, 2018.

[21] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, "On reducing iot service delay via fog offloading," *IEEE Internet of Things Journal*, vol. PP, no. 99, pp. 1–1, 2018.

[22] X. Wang, Z. Ning, and L. Wang, "Offloading in internet of vehicles: A fog-enabled real-time traffic management system," *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, pp. 1–1, 2018.

[23] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, "Multiobjective optimization for computation offloading in fog computing," *IEEE Internet of Things Journal*, 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 *2017 IEEE MIT Undergraduate Research Technology Conference (URTC)*, 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 *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, 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 *2016 IEEE Region 10 Conference (TENCON)*, Nov 2016, pp. 1039–1042.

[27] S. Song and J. Xiao, "Deep sliding shapes for amodal 3d object detection in rgb-d images," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 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 Transactions on Parallel and Distributed Systems*, 2017, doi: 10.1109/TPDS.2017.2740294.

[29] W. Liu, Y. Wen, Z. Yu, and M. Yang, "Large-margin softmax loss for convolutional neural networks," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16. JMLR.org, 2016, pp. 507–516. [Online]. Available: http://dl.acm.org/citation.cfm?id=3045390.3045445

[30] R. Girshick, "Fast r-cnn," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ser. ICCV '15. Washington, DC, USA: IEEE Computer Society, 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 Network*, 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 *2016 23rd International Conference on Pattern Recognition (ICPR)*, 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 *Advances in Neural Information Processing Systems*, 2017, pp. 4151–4161.