Proceedings of the 2020 International Manufacturing Science And Engineering Conferences
MSEC2020
June 22-June 26, 2020, Cincinnati, USA

MSEC2020-6445

EDGE COMPUTING ACCELERATED DEFECT CLASSIFICATION BASED ON DEEP CONVOLUTIONAL NEURAL NETWORK WITH APPLICATION IN ROLLING IMAGE INSPECTION

Jiayu Huang, Nurretin Sergin, Akshay Dua, Erfan Bank Tavakoli, Hao Yan*, Fengbo Ren, Feng Ju,
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University
Tempe, Arizona 85281
Email*: haoyan@asu.edu

ABSTRACT

This paper develops a unified framework for training and deploying deep neural networks on the edge computing framework for image defect detection and classification. In the proposed framework, we combine the transfer learning and data augmentation with the improved accuracy given the small sample size. We further implement the edge computing framework to satisfy the real-time computational requirement. After the implement of the proposed model into a rolling manufacturing system, we conclude that deep learning approaches can perform around 30-40% better than some traditional machine learning algorithms such as random forest, decision tree, and SVM in terms of prediction accuracy. Furthermore, by deploying the CNNs in the edge computing framework, we can significantly reduce the computational time and satisfy the real-time computational requirement in the high-speed rolling and inspection system. Finally, the saliency map and embedding layer visualization techniques are used for a better understanding of proposed deep learning models.

1 INTRODUCTION

In modern manufacturing systems, image-based inspection techniques have gained much popularity in modern industrial systems due to the low cost and high efficiency of the image sensors. Furthermore, both the production speed and sensing rate have increased dramatically. For example, in the current high-speed metal rolling manufacturing system, the production line speed is around 100m/s [1]. To keep up with the high production speed, high-speed image inspection systems are required to help improve the manufacturing process. With tens or even hundreds of these cameras installed in the entire production line, it becomes possible that the amount of data generated in one second can be up to gigabytes for the entire manufacturing systems.

On the other hand, image data present complex spatial correlation with little loss so that it becomes a powerful base for analysis and inspection. Figure 1 shows an illustration sample of different types of anomalies produced by the rolling manufacturing processes. Given images with different labels observed in Figure 1, we would like to classify the image patches to different types of anomalies for better fault diagnosis. However, different types of anomalies have very different and complex spatial correlation, which creates significant challenges for the design of machine learning algorithms.

In recent years, deep learning methods have been proposed and also have achieved initial success in image classification in manufacturing systems. But the development of different deep learning architectures in the manufacturing setting with a limited sample size has not been not fully discussed. Furthermore, deep learning algorithms typically have larger computational complexity than traditional machine learning methods, which has not yet been addressed for real-time implementation. Traditional computing architectures will have to integrate centralized cloud computing for data processing if they are applied in real-time implementation, as mentioned above. However, cloud computing results in sending a massive amount of data to and from the centralized cloud system, which increases the demand for the

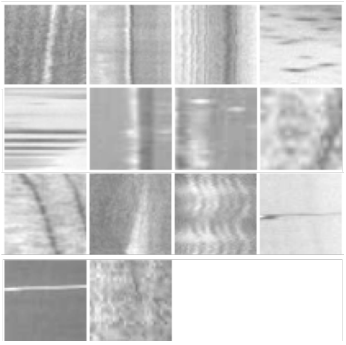

FIGURE 1. ANOMALY SNAPSHOTS IN ROLLING MANUFACTURING PROCESS

network bandwidth. Also, data transmission latency limits the performance of real-time IoT applications. The solution to these limitations of cloud computing is edge computing, which decentralizes the processing of data near the end-user. Edge computing significantly reduces the data transfer latency by processing data locally and lowers the network bandwidth requirements.

In this paper, we propose a complete procedure to develop and apply deep learning architectures into the defect classification framework using real data collected from the rolling manufacturing process. Parameters of pre-trained models except the last few layers are used directly to save training time and improve the performance for limited sample classification. Performance comparison between different methods is given in this paper, which shows the application of part of layers in pre-trained models and the parameters can achieve a significant improvement based on limited sample size in the manufacturing process. Furthermore, through implementing proposed CNNs into the edge computing framework, we are able to alleviate the processing power requirement and network transmission of the central processing server and satisfy the real-time requirement in the manufacturing systems.

2 LITERATURE REVIEW

2.1 Development of image classification

As the capacity of obtaining images with the high quality increases during the production process, image-based techniques have been widely used in manufacturing industry quality control with specific tasks such as surface quality inspection [2]. Yan et al. [3] have divided the traditional approaches for image-based process monitoring and diagnosis into four steps: image acquisition, image pre-processing, feature extraction, and process monitoring & fault diagnostics. Image acquisition and image pre-processing refer to the steps of transferring the image data from sensors to the central server with preliminary data pre-processing. Feature extraction aims to extract useful features from the original images. For feature extraction methods, methods such as bag of features [4], dimension reduction [3], spatial pyramids and kernel-based pyramid matching [5], SIFT features [6] are often used. In numerous manufacturing industries like steel companies, classifying defects plays a significant part in fault diagnostics. To achieve this, supervised learning techniques are often used in the extracted features, which is very common in manufacturing currently. Many other ML algorithms are established based on it, such as neural networks (NNs), support vector machines (SVMs), random forest (RF) and Bayesian modeling that are all widely used in production system [7]. For example, SPM features are used in SVM to realize tasks with high accuracy [8]. Recently, Neural Networks, which were inspired by the functionality of the brain, have become popular in manufacturing [9, 10]. They can be applied in the classification of tool wear, estimation of drift trend for yield enhancement and process control. [11, 12]

Recent progress of deep learning architectures such as the Convolutional Neural Networks (CNN) has enabled machine learning algorithms to be directly applied to the images without manual feature extraction. CNNs [13] has gained some initial attention on supervised image classification. Different architectures including AlexNet [14], VGGNet [15], GoogleNet [16], and ResNet [17] continue to improve the image classification accuracy.

Because of fast and easy access to image data in the manufacturing process, the application of CNNs has also become popular in product defect detection, segmentation, and recognition in the inspection systems [18]. Method comparison between CNNs and other ML methods show us that CNNs provide significant improvement in detecting faults in many different settings [19]. Tao et al. have applied CNNs for the metal surface image defect classification [20]. Besides, some classic CNNs such as LeNet-5 are modified to improve fault diagnosis [21]. One specific reason for the popularity of the deep learning techniques applied to manufacture applications is the transfer learning capacity. Especially, training only the last few layers of any pre-trained CNNs on natural images can often achieve great accuracy for other tasks. However, due to the large gap between natural images used in many pre-trained CNNs and manufacturing inspection images, and to the best of our knowledge, the effects of different architectures of pre-trained CNNs have not been fully studied in the manufacturing defect detection problem.

2.2 Edge-computing framework

Recently, the edge computing framework has attracted the attention of some researchers, given its faster computational speed, closer to the data source, smaller latency, better security, scalability, versatility, and reliability. Moreover, there is a growing trend of deploying deep learning architectures, such as CNNs into the edge computing framework [22], which have shown initial success in many IoT applications.

GPUs and FPGAs are used at the edge servers. Recently, FPGAs are becoming popular for the edge computing [23]. The advantages of using FPGAs for edge computing include offering high energy efficiency as compared to GPUs. Moreover, GPUs achieve high throughput when processing a batch of data in parallel, while FPGAs throughput does not depend on the batch size of data. Thus, FPGAs are more suitable for the processing of streaming data in real-time image defect detection and classification in the manufacturing setting.

In an industrial system design, an FPGA could be installed as a co-processor (*i.e.*, a processor used to supplement functions of the primary processor) or as a System on Chip (SoC) solution. Host processors performing system processing and FPGAs could be integrated on the same board. As another option, some of the processing functions could be moved from processors to an FPGA. It is also possible to offload all processor tasks into an FPGA as an FPGA-based Soc platform [24].

In literature, using FPGAs for computations is by implementing a design at Register Transfer Level (RTL) (*e.g.*, [25]) using Hardware Descriptive Languages (HDLs) such as Verilog and VHDL. Also, FPGA developers can utilize High-level Synthesis (HLS) tools that take C/C++ code as input and output RTL code (*e.g.*, [26]), without knowing the underlying hardware details.

To the best of our knowledge, there is no research about the application and feasibility of FPGA in the manufacturing system, which will be further studied in this paper.

3 MATERIALS AND METHODS

In this paper, we will first describe the procedure of applying deep learning algorithms for image defect classification. After the image acquisition, we divided the follow-up procedures into the training phase and the online deployment phase in the edge computing framework. In the training phases, additional techniques such as image augmentation and transfer learning can be further applied. In the online deployment, we will discuss how to deploy the trained model into the edge computing framework. Finally, the saliency map and embedding layers map can be used for the domain engineers to better visualize and understand these classification results. Figure 2 shows the details of the proposed framework.

3.1 Training Techniques

We will first discuss the training techniques that helped to solve the small sample problem. We will implement the transfer learning techniques on training based on two pre-trained architectures AlexNet and ResNet and compare it with the end-to-end training techniques on the customized Neural Network.

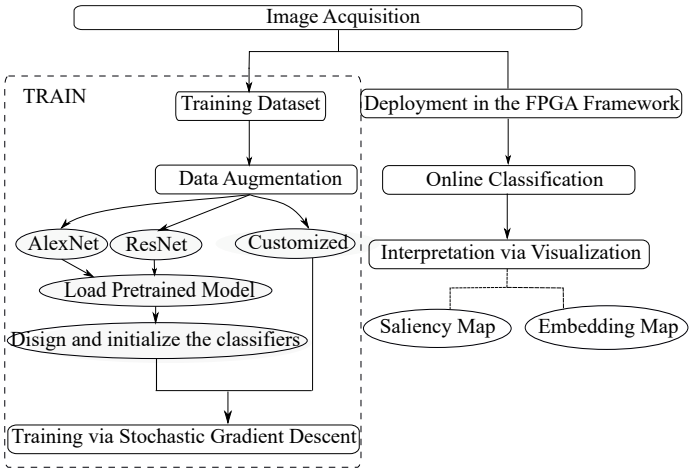

FIGURE 2. FLOWCHART OF PROPOSED METHODS

3.1.1 CNN architectures: Transfer Learning and Customized Architectures A typical CNN normally consists of convolutional layers, activation layers, and fully connected layers. The layers that builds the architectures in this study are summarized as follows:

1. C(O,K,S,P): Convolutional layer with arguments referring to number of output channels O, kernel size K, stride S and size of zero-padding P.
2. MP(K): Maxpooling layer, which compute the largest value for each $K \times K$ window.
3. AP(): Average layer, which computes the average value for each window to get output with 1×1 size.
4. R(): Rectified linear unit (ReLU) defined as $f(x) = \max(0,x)$.
5. SM(): Softmax layer, defined as $f(\mathbf{x})_i = \exp(x_i)/\sum_i \exp(x_i)$.
6. BN(): Batch-normalization layer, which helps reduce the amount by what the hidden unit values shift around.
7. D(): Dropout layer, which can help reduce overfitting by randomly dropping out some nodes in the process of training.

Transfer Learning AlexNet famously won the ILSVRC 2012 competition, which consists of five convolutional layers, and three fully-connected layers with a last 1000-way softmax layer to do classification [27]. Recently, ResNets such as ResNet50 have won the ILSVRC 2015 by introducing the residual module and deeper network. In this paper, we will apply both AlexNet and ResNet50, which have been pre-trained to classify 1000 object categories on the ImageNet database [28]. Furthermore, to modify the AlexNet and ResNet to model the defective images, we design the following architectures and replace the last few layers with fully convolutional layers for classification. Fully convolutional layers indicate that the neural network is composed

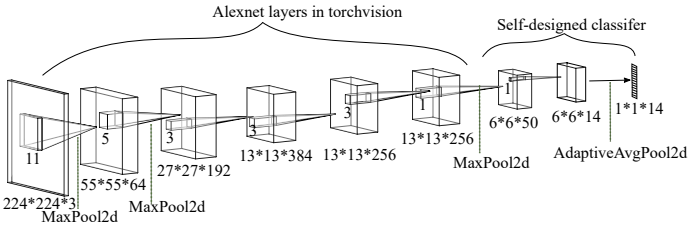

FIGURE 3. ALEXNET ARCHITECTURE

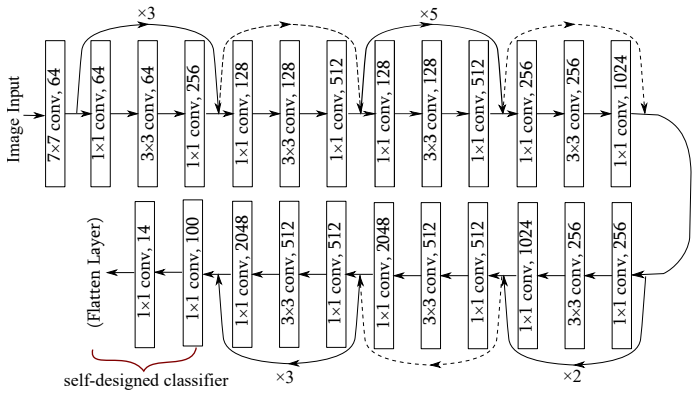

FIGURE 4. RESNET50 ARCHITECTURE

of all convolutional layers without any fully-connected layers commonly found at the bottom of other neural networks. Designing a fully convolutional network is more beneficial for manufacturing defect detection in the following aspects: 1) More spatial information: using fully connected layers may cause the loss of spatial information, whereas a fully convolutional network uses the spatial information from the beginning to the end. 2) Computational efficiency: using fully convolutional layers is typically more computational efficient due to highly efficient convolutional operators. 3) Reduced model complexity: The number of parameters in the deep learning models can be reduced greatly, which lead to the less strict requirement for sample size needed to train such deep learning networks. This is especially imporant for manufacturing systems. Finally, the architectures for AlexNet and ResNet with our self-designed fully convolutional CNNs parts are visualized respectively in Figure 3 and Figure 4.

Customized Architecture To design a customized network for rolling examples, we don't require the images to have such high resolution since the amount of information needed to classify the image is less than the natural imaging. Therefore, we downsample the size of the images to 64×64 and design the architecture based on it. The architecture is visualized in Table 1 and Figure 5.

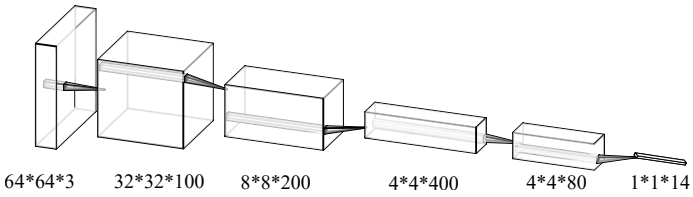

FIGURE 5. SIMPLE CNNS ARCHITECTURE

3.1.2 Image Pre-processing and Augmentation

The input to AlexNet and ResNet is an RGB image of size $224 \times 224 \times 3$, which means that three channels are required, and all images should be cropped to meet the input size requirement. Due to the lack of uniformness of our rolling image dataset, image pre-processing has been done to generate data in a tensor format parallel with RGB images without omitting useful information. Transformations like cropping and resizing are necessary for pre-processing based on the following augmentation layers:

1. Cr(K): Randomly crop the image into $K \times K$ size output with a random ratio from 0.5 to 1.0.
2. Re(K): Resize the image into an expected $K \times K$ size output image.
3. To(): Convert the PIL image into a tensor format.
4. L(): Normalize the image data ranging from 0.000 to 1.000.
5. Nor(): Normalize a tensor image with mean and standard deviation. Here we set $(0.485, 0.456, 0.406)$ as the mean, and $(0.229, 0.224, 0.225)$ as the standard deviation, which is to ensure the images have the same mean and standard deviation of the natural images for pre-training.

3.2 Implementation of Edge-computing Framework

In this paper, we have implemented Intel FPGA SDK for OpenCL to develop the FPGA accelerator for the proposed design. OpenCL (Open Computing Language) is a framework for parallel programming and accelerating algorithms across heterogeneous platforms [29]. In general, OpenCL enables developers to compute kernels using a C-based programming language. OpenCL provides an application programming interface (API) for the host processor to communicate with the hardware accelerator (*e.g.*, the FPGA) or kernels communicating directly.

In order to deploy Neural Networks (NNs) using OpenCL, we have implemented the required kernels for different functionalities of the target NN such as convolutional layers, activation layers, batch normalization layers, and different pooling layers and connect these modules together. We also utilize a multi-level memory hierarchy to facilitate the storage and transfer of data. OpenCL improves the code portability and programmability of FPGAs as well as enabling more efficient and easier design space exploration to find an optimum architecture configuration for a specific FPGA device and CNN architecture.

TABLE 1. ARCHITECTURE DETAILS OF DEEP LEARNING MODELS USED IN THIS STUDY

Model Type	Sub Module	Architecture
AlexNet	Feature	C(64, 11, 4 ,2) - R() - MP() - C(192, 5, 1, 2) - R() - MP() - C(384, 3, 1, 1) - R()- C(256, 3, 1, 1) - R() - C(256, 3, 1, 1)
	Classifier	C(50, 1, 1, 1) - R() -C(14, 1, 1, 1) - AP()
ResNet	ResModule()	Identity + C(64, 1, 1, 0) - BN() - C(64, 3, 1, 1) - BN() - C(256, 1, 1, 0) - BN() - R()
	Feature	C(3, 7, 2, 3) - BN() - R() - MP() - ResModule() - ResModule() - ResModule() - ResModule()
	Classifier	C(100, 1, 1, 1) - R() -C(14, 1, 1, 1) - AP()
Simple CNNs	Main Architecture	C(100, 3, 1, 1) - MP()- R() - C(200, 3, 1, 1) - MP()- R() - C(400, 3, 1, 1) - MP()- R() - D() - C(80, 3, 1, 1) - R() - C(14, 1, 1, 1)- R() - AP()

TABLE 2. DATA AUGMENTATION LAYERS

Model Type	Augmentation Layers
AlexNet and ResNet	Cr(256) - Re(224) - To() - L() - Nor()
Simple CNNs	Cr(80) - Re(64) - To() - L() - Nor()

TABLE 3. IMAGE SAMPLES IN EACH CATEGORY

Label	0	1	2	3	4	5	6	7
Count	43	72	75	134	54	89	78	18
Label	7	8	9	10	11	12	13	-
Count	18	105	72	75	127	115	96	-

In our work, the FPGA is used as a co-processor or an I/O companion alongside a host processor via the PCIe interface. The design includes a host program and a set of kernels for the CNN architecture. The OpenCL framework compiles the host code and the CNN architecture kernels and configures the kernels onto the FPGA. By executing the host program, we run the architecture using input data and commands sent from the host processor and receive results processed by the FPGA on the host processor [30].

3.3 Visualization

3.3.1 Saliency Map Visualization Neural networks are often described as "black box" [31]. The lack of understanding of how neural networks make predictions enables unpredictability, causing distrust of AI-assisted systems for the domain engineers in the manufacturing system. Saliency maps in computer vision provide indications of the most salient regions within images. By creating a saliency map for neural networks, we can gain some intuition on "where the network is paying the most attention to" in an input image. The spatial configuration of targets could be reflected clearly and efficiently by highlighting the regions discriminating target from background [32]. In this paper, we propose to borrow the saliency map techniques based on the guided backpropagation to back-propagate the gradient back to the original input images [33] to highlight the corresponding areas, which is responsible for such classification results.

3.3.2 Embedding layer visualization For the high-dimensional problem, it is challenging to get the map of how well the model can make a decision given unseen samples. In this paper, t-distributed Stochastic Neighbor Embedding (t-SNE) technique is applied to visualize the embedding layer of the CNNs architectures. By reducing the tendency to crowd points together in the center of a single map [34], it can help us visualize the clusters of each label in a 2-D map before the final layer of our CNNs algorithms.

4 CASE STUDY

4.1 Data processing

In this section, we will use real images collected from the quality inspection of the rolling manufacturing to illustrate the performance of the proposed defects classification procedures. The domain engineers helped us labeled the anomalous images as different classes. In total, there are 1173 images, with each one being labeled as one of 14 various known classes by domain engineers. Figure 1 illustrates 14 different defect classes. Table 3 shows the respective number of samples for each type of defect. It is clear that many classes have limited samples due to the sparsity of the defective samples. To avoid overfitting on the small sample size in Table 3, we implement the data augmentation techniques illustrated in Table 2 for deep learning methods.

4.2 Parameters tuning

Comparison among different CNNs and other methods that are not included in deep learning is conducted based on both the training and, more importantly, testing performance in the metal rolling manufacturing system. Learning rates for the train stage for all three different CNNs are set to 0.0001 and the optimizers are Adam(Adaptive Moment Estimation) in our case study. The ratio of training and testing sample size is fixed to 8:2. For the random forest model, the number of trees here is set as 1000, and the maximum of depth of the trees is 9. For SVM, the regularization parameter is 1 and we use linear kernel to fit the model. For decision trees method, the depth maximum is fixed to 9 eventually. All the tuning parameters are selected by the cross validation procedure.

4.3 Evaluation and result comparison

For the evaluation of the proposed methods, we first like to compare the classification accuracy, which is defined as the percentage of the images which are classified correctly. For the benchmark methods, we compared the proposed methods with several image classification techniques in the literature, such as random forest, SVM, and decision tree. The training and test accuracy of the proposed methods and the benchmark methods are shown in Table 4. From Table 4, we can conclude the following: 1) The training accuracy rates for all methods are around 99%, which means all the models have been trained sufficiently. 2) The test accuracy scores generated from deep learning methods outperform all non-deep learning methods dramatically, which achieve at most 56% for random forest. For example, the pre-trained AlexNet, ResNet50, and customized NN can achieve 84%, 83%, and 77%. 3) Among all the deep learning methods, the pre-trained models, namely AlexNet and ResNet50 perform better than the customized simple CNN. The reason for the superior performance of the pre-trained models is that transfer learning can be very beneficial in the case of limited training sample size. Furthermore, given that manufacturing images often contains simpler patterns compared to the natural images, using deeper architecture such as ResNet50 does not help too much. 4) Based on Table 4, we recommend Alexnet, given its less model complexity and greater accuracy. In Figure 6, the confusion matrix generated by testing based on trained AlexNet model and fitted Random forest model, which performs best among listed CNN methods and Non-CNN methods respectively in this paper. Figure 6 indicates that AlexNet model is able to predict more pictures accurately than Random Forest can. One thing in common is that some specific labels are so similar that both methods cannot identify them very clearly, such as label 5 and 6.

To understand more about how the proposed methods are able to classify these images, we will use the saliency map to illustrate how the NNs give such prediction intuitively and how it can be used to locate different types of anomalies. The saliency

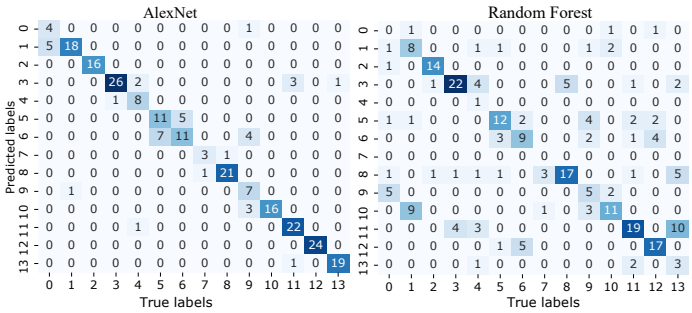

FIGURE 6. CONFUSION MATRIX

TABLE 4. ACCURACY COMPARISON

Methods	Training Accuracy	Testing Accuracy
AlexNet	99%	84%
ResNet50	99%	83%
Simple CNNs	98%	77%
Random Forest	99%	56%
SVM	98%	34%
Decision Trees	99%	39%

maps are plotted in Figure 7. We can clearly see that the pre-trained networks such as Alexnet and ResNet50 are able to focus on the major discriminative patterns, such as different line patterns and curve patterns. Even though the customized simple CNN can detect the majority of the patterns as well, it is much noisier since the limited number of samples introduce larger variance in the trained CNN kernels. Furthermore, the saliency maps also highlight the image defect location for better localization of defects in these inspection images.

Furthermore, we plotted the embedding maps of trained AlexNet to visualize the image representation before the final classification layer. By using the t-SNE to reduce the dimensionality to 2-D, we can conclude that in Figure 8, most defects with the same label are nicely clustered. Therefore, it becomes easy to understand why test accuracy gained by trained Alexnet can reach 84% around. Furthermore, these low-dimensional representations can be used for other machine learning tasks as well. Finally, we would like to evaluate the impact of different hardware designs on the real-time implementation of the proposed CNN architectures for online defect classification. The rolling inspection system in our system has the sampling rate as 50HZ, which requires the processing time lower than 20ms per sample for real-time implementation. We will compare the performance of the cloud computing architecture and the edge computing ar-

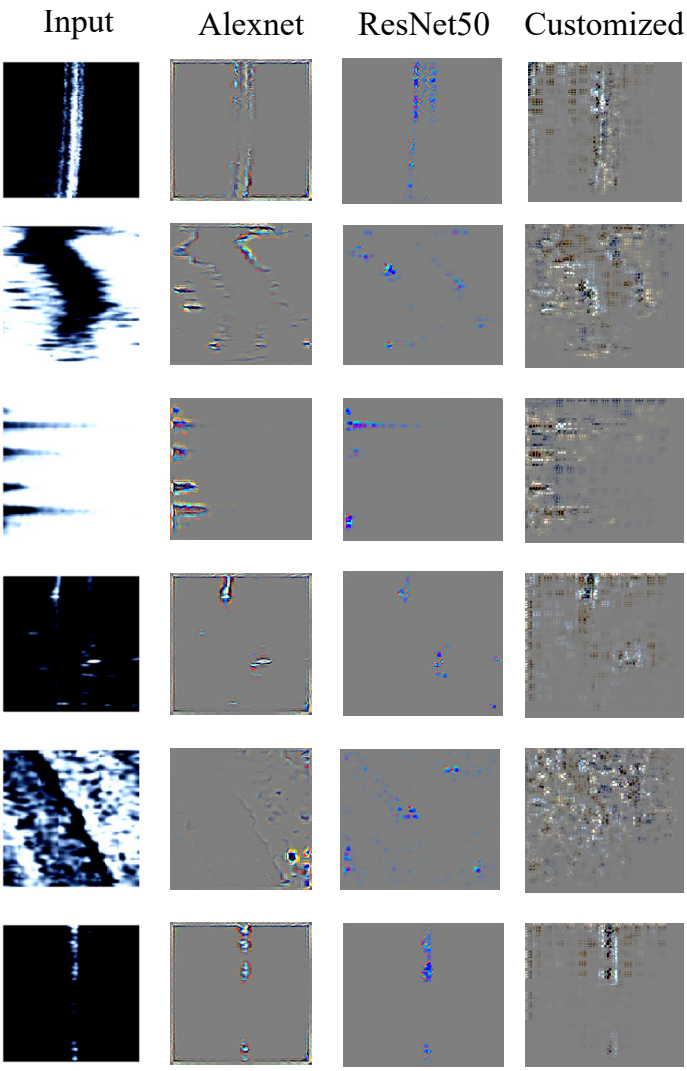

FIGURE 7. SALIENCY MAPS OF TRAINED ALEXNET

chitecture in terms of computational time for processing each single image sample and whether they can satisfy the real-time monitoring requirement. For cloud computing, we compare the performance based on the CPU and GPU computing. For edge computing, we will compute the performance based on FPGA. For modified AlexNet model, we decompose the total computational time (denoted as total) into reading data to memory (denoted as to memory), time to transfer the data from memory to GPU (denoted as to GPU), feature layers in CNN (denoted as feature), classifier layers in CNN (denoted as classifier) in Table 5. The reason for the slow computational time in the CPU architecture (i.e., more than 100ms) is the feature layers, which requires to compute many expensive convolutional layers. In comparison, using the GPU architecture can greatly reduce the computational time in the feature layer due to the efficient implementation of the convolutional operator. However, the latency of transferring the data to GPU takes more than 50ms. In comparison, the FPGA framework has little or no latency to start with and has comparable computational time with GPU, which results in the total computational time around 8.137ms, which satisfies our real-time monitoring performance. In this case, FPGA is much more efficient than direct computing in the architecture based on CPU and GPU computing.

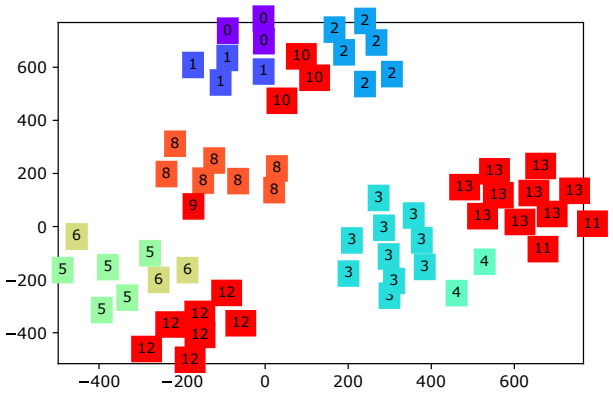

FIGURE 8. EMBEDDING MAP OF TRAINED ALEXNET

TABLE 5. TIME COMPARISON FOR ALEXNET(UNIT:MS)

Steps	CPU	GPU	Edge
To memory	13	13	-
To GPU	-	58.990	-
Feature	109.243	4.020	7.875
Classifier	1.499	0.463	0.262
Total	123.742	76.473	8.137

5 CONCLUSION

In this paper, we proposed a unified deep learning framework for image defect classification or fault diagnosis in the manufacturing industry and deployed in the edge computing framework. With the application of transfer learning, data augmentation, and fully convolutional architectures, we demonstrate that even with limited defective samples, deep learning approaches can achieve 84% an 83% classification accuracy, which performs 30-40% better than traditional machine learning algorithms such as random forest, decision tree, and SVM.

Furthermore, to implement the proposed framework in the high-speed rolling and inspection system, we deployed the CNNs into the edge computing framework and demonstrated how the edge computing framework can greatly reduce the computational time of the CNNs deployed in the CPU and GPU architecture. Application of both pre-trained CNNs and edge-computing make it possible for the small manufacturing companies to perform real-time fault diagnosis in high-speed rolling or inspection system with high accuracy. The comparison of the defect classification speed we have done in this paper may help these companies to decide whether they need to purchase the edge-computing device to eventually improve the efficiency of the entire manufacturing system.

Finally, we implement the saliency map visualization and embedding layer technique to provide a clear understanding of how deep learning models make such decisions in the image defect classification applications, which can be useful for diagnostics.

REFERENCES

[1] Xi-yuan, S., and Chao, W., 2012. "Study on deformation behavior of rolled metal at ultra high rolling speed in wire rod mills". *Advanced Materials Research,* ***581***, pp. 912–918.

[2] Hornberg, A., 2017. *Handbook of Machine and Computer Vision: The Guide for Developers and Users*. John Wiley & Sons, Incorporated, Somerset, Germany.

[3] Yan, H., Paynabar, K., and Shi, J., 2014. "Image-based process monitoring using low-rank tensor decomposition". *IEEE Transactions on Automation Science and Engineering,* ***12***(1), pp. 216–227.

[4] Csurka, G., Dance, C., Fan, L., Willamowski, J., and Bray, C., 2004. "Visual categorization with bags of keypoints". In Workshop on Statistical Learning in Computer Vision, ECCV, Vol. 1, Prague, pp. 1–2.

[5] Lazebnik, S., Schmid, C., and Ponce, J., 2006. "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories". In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Vol. 2, IEEE, pp. 2169–2178.

[6] Lowe, D. G., 2004. "Distinctive image features from scale-invariant keypoints". *International journal of computer vision,* ***60***(2), pp. 91–110.

[7] Brunato, M., and Battiti, R., 2005. "Statistical learning theory for location fingerprinting in wireless LANs". *Computer Networks,* ***47***(6), Apr., pp. 825–845.

[8] Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., and Gong, Y., 2010. "Locality-constrained Linear Coding for image classification". In 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE, pp. 3360–3367.

[9] Wuest, T., Weimer, D., Irgens, C., and Thoben, K.-D., 2016. "Machine learning in manufacturing: Advantages, challenges, and applications". *Production & Manufacturing Research,* ***4***(1), Jan., pp. 23–45.

[10] Yazdchi, M. R., Mahyari, A. G., and Nazeri, A., 2008. "Detection and Classification of Surface Defects of Cold Rolling Mill Steel Using Morphology and Neural Network". In 2008 International Conference on Computational Intelligence for Modelling Control Automation, pp. 1071–1076.

[11] Sick, B., 2002. "On-line and Indirect Tool Wear Monitoring in Turning with Artificial Neural Networks: A Review of More than a Decade of Research ". *Mechanical Systems and Signal Processing,* ***16***(4), July, pp. 487–546.

[12] Yaw-Jen Chang, Kang, Y., Chih-Liang Hsu, Chi-Tim Chang, and Chan, T. Y., 2006. "Virtual Metrology Technique for Semiconductor Manufacturing". In The 2006 IEEE International Joint Conference on Neural Network Proceedings, pp. 5289–5293.

[13] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D., 1989. "Backpropagation applied to handwritten zip code recognition". *Neural computation,* ***1***(4), pp. 541–551.

[14] Krizhevsky, A., Sutskever, I., and Hinton, G. E., 2012. "Imagenet classification with deep convolutional neural networks". In Advances in neural information processing systems, pp. 1097–1105.

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

[16] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A., 2015. "Going deeper with convolutions". In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1–9.

[17] He, K., Zhang, X., Ren, S., and Sun, J., 2016. "Deep residual learning for image recognition". In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778.

[18] Lea, C., Reiter, A., Vidal, R., and Hager, G. D., 2016. "Segmental spatiotemporal cnns for fine-grained action segmentation". In European Conference on Computer Vision, Springer, pp. 36–52.

[19] Lee, K. B., Cheon, S., and Kim, C. O., 2017. "A convolutional neural network for fault classification and diagnosis in semiconductor manufacturing processes". *IEEE Transactions on Semiconductor Manufacturing,* ***30***(2), May, pp. 135–142.

[20] Tao, X., Zhang, D., Ma, W., Liu, X., and Xu, D., 2018. "Automatic Metallic Surface Defect Detection and Recognition with Convolutional Neural Networks". *Applied Sciences,* ***8***(9), Sept., p. 1575.

[21] Wen, L., Li, X., Gao, L., and Zhang, Y., 2017. "A new convolutional neural network-based data-driven fault diagnosis method". *IEEE Transactions on Industrial Electronics,* ***65***(7), pp. 5990–5998.

[22] Han, Y., Wang, X., Leung, V., Niyato, D., Yan, X., and Chen, X., 2019. "Convergence of edge computing and deep learning: A comprehensive survey". *arXiv preprint arXiv:1907.08349*.

[23] Biookaghazadeh, S., Zhao, M., and Ren, F., 2018. "Are fpgas suitable for edge computing?". In USENIX Workshop on Hot Topics in Edge Computing (HotEdge) 2018, USENIX Association.

[24] Chiang, J., and Zammattio, S., 2014. "Wp:"five ways to build flexibility into industrial applications with fpgas". *Altera Corporation, September*.

[25] Bank-Tavakoli, E., Ghasemzadeh, S. A., Kamal, M., Afzali-Kusha, A., and Pedram, M., 2019. "Polar: A pipelined/overlapped fpga-based lstm accelerator". *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*.

[26] Li, Y., Liu, Z., Xu, K., Yu, H., and Ren, F., 2018. "A gpu-outperforming fpga accelerator architecture for binary convolutional neural networks". *ACM Journal on Emerging Technologies in Computing Systems (JETC),* ***14***(2), p. 18.

[27] Krizhevsky, A., Sutskever, I., and Hinton, G. E., 2012. "Imagenet classification with deep convolutional neural networks". In Advances in Neural Information Processing Systems, pp. 1097–1105.

[28] He, K., Zhang, X., Ren, S., and Sun, J., 2015. "Deep Residual Learning for Image Recognition". *arXiv preprint arXiv:1512.03385*, Dec.

[29] Munshi, A., 2009. "The opencl specification". In 2009 IEEE Hot Chips 21 Symposium (HCS), IEEE, pp. 1–314.

[30] Czajkowski, T. S., Aydonat, U., Denisenko, D., Freeman, J., Kinsner, M., Neto, D., Wong, J., Yiannacouras, P., and Singh, D. P., 2012. "From opencl to high-performance hardware on fpgas". In 22nd international conference on field programmable logic and applications (FPL), IEEE, pp. 531–534.

[31] Olden, J. D., and Jackson, D. A., 2002. "Illuminating the "black box": A randomization approach for understanding variable contributions in artificial neural networks". *Ecological Modelling,* ***154***(1), pp. 135–150.

[32] Hong, S., You, T., Kwak, S., and Han, B., 2015. "Online tracking by learning discriminative saliency map with convolutional neural network". In International conference on machine learning, pp. 597–606.

[33] Simonyan, K., Vedaldi, A., and Zisserman, A., 2013. "Deep inside convolutional networks: Visualising image classification models and saliency maps". *arXiv preprint arXiv:1312.6034*.

[34] Maaten, L. v. d., and Hinton, G., 2008. "Visualizing data using t-sne". *Journal of machine learning research,* ***9***(Nov), pp. 2579–2605.