JOURNAL OF ENGINEERED FIBERS AND FABRICS

Original Article

Journal of Engineered Fibers and Fabrics
Volume 16: 1–13

DOI: 10.1177/15589250211008346
journals.sagepub.com/home/jef
SAGE

EfficientDet for fabric defect detection based on edge computing

Shaojun Song, Junfeng Jing, Yanqing Huang and Mingyang Shi

Abstract

The productivity of textile industry is positively correlated with the efficiency of fabric defect detection. Traditional manual detection methods have gradually been replaced by deep learning algorithms based on cloud computing due to the low accuracy and high cost of manual methods. Nonetheless, these cloud computing-based methods are still suboptimal due to the data transmission latency between the end devices and the cloud. To facilitate defect detection with more efficiency, a low-latency, low power consumption, easy upgrade, and automatical visual inspection system with the help of edge computing are proposed in this work. Firstly, the method uses EfficientDet-D0 as the detection algorithm, integrating the advantages of lightweight and scalable and can suit the resource-constrained edge device. Secondly, we performed data augmentations on five fabric datasets and verified the adaptability of the model in different types of fabrics. Finally, we transplanted the trained model to the edge device NVIDIA Jetson TX2 and optimized the model with TensorRT to make it detection faster. The performance of the proposed method is evaluated in five fabric datasets. The detection speed is up to 22.7 frame per second (FPS) on the edge device Jetson TX2. Compared with the cloud-based method, the response time is reduced by 2.5 times, with the capability of real-time industrial defect detection.

Keywords

Edge computing, fabric defect detection, deep learning, convolutional neural network

Date received: 7 March 2021; accepted: 19 March 2021

Introduction

TEXTILES are widely used in our lives, such as clothes, towels, filter cloth, and also in some select fields, such as aerospace, medical hygiene.[1,2] However, in the process of textile production, it is inevitable to produce defective fabrics. Most of the defects are caused by machine failures, defective yarns, and oil stains on sewing gadgets.[2] These unqualified textiles will not only damage the reputation of the company but even have a vital impact on the safety of certain products. According to market research, the price of second-class textile fabrics is reduced by 45%–65% compared with first-class textile fabrics.[3] Therefore, to boost profits and product competitiveness, fabric defect inspection has become an essential step. In the light of feedback from textile factories, the accuracy of manual inspection method can only reach 70% and the high labor cost is a significant burden on the enterprise. Accordingly, the automated fabric defect inspection system is desirable for quality control of the textile industry. At present,

College of Electronics and Information, Xi'an Polytechnic University, Xi'an, China

Corresponding author:
Junfeng Jing, College of Electronics and Information, Xi'an Polytechnic University, No.19 Jinhua South Road, Xi'an, Shaanxi 710048, China.
Email: Jingjunfeng0718@sina.com

automated fabric inspection system based on computer vision (CV) is mainly divided into two categories: traditional image recognition method, deep learning based method.[4]

From the perspective of image analysis, traditional image recognition inspection has three classes[5]: statistical analysis,[6,7] frequency domain analysis[2,8–10] and model analysis[5,11,12] used the Gabor filter band and lattice segmentation to extract image patch features, and then use it for defect detection. Jing et al.[10] proposed a fabric defect detection based on CIE L × a × b color space using 2-D Gabor filter. Xiaobo et al.[11] apply the Gaussian-Markov random field (GMRF) model to extract fabric features and detection. All of these methods required manually designed feature extraction schemes. Nevertheless, with the wide variety of textiles today, the generality of these methods has been greatly restricted. In recent years, with the substantial improvement of Graphics Processing Units (GPU) computing capabilities, deep learning method has made significant progress. Different from traditional image processing algorithms, deep learning methods use the convolutional neural network (CNN) to automatically extract image features, which integrates feature learning into the whole process of model building and dramatically improves the generality of the model. Cha et al.[13] apply CNN to wall crack detection, and the proposed CNN method showed very robust performance compared to the traditional well-known edge detection algorithms (i.e. Canny[14] and Sobel). Jing et al.[15] proposed a fabric defect detection method based on YOLO-V3,[16] which improves the detection effect by clustering datasets and adding network detection layers and finally applied it to gray cloth and lattice.

In the textile industry scenario, an automatic fabric defect detection system needs to meet three requirements.[5] First, the high-speed running product line needs the detection system satisfy the requirements of real-time and low response latency. Second, the power consumption of the product line should be as low as possible. Finally, in order to upgrade the product line conveniently, the detection system must scale well. In response to the above requirements, a fabric defect detection method based on edge computing is proposed in this paper. In summary, the main contributions of this work are listed as follows:

1. This paper proposes an automatic fabric defect detection system that combines EfficientDet and edge computing, and edge device is applied to fabric defect detection. Deploying a lightweight network on the edge device NVIDIA Jetson TX2 reduces the overall response time of the system and leverages model features to make the production line easier to upgrade.
2. In order to improve the robustness of the model, we augmented the training datasets and compared EfficientDet with mainstream one-stage detection networks on five different fabric datasets.
3. To adapt to the limited computing resources of edge devices, we optimized the model with TensorRT to make it satisfy the requirements of industrial detection speed. Besides, the corresponding latency of cloud computing and edge computing is compared.

The rest of the paper is organized as follows: Section 2 introduces the related work of this paper, including traditional defect detection methods and detection methods based on deep learning, as well as the current status of edge computing. Section 3 describes the details of our proposed method. Section 4 briefly introduces five different fabric datasets, and subsequently discusses the experimental results, including the test results of the model, TensorRT optimization after model transplantation, and the comparison between cloud computing and edge computing. Section 5 concludes this paper.

Related work

In this section, the computer vision based detection method is briefly reviewed from the following three perspectives: traditional defect detection methods, detection methods based on deep learning, and the status quo and applications of edge computing.

Traditional defect detection methods

According to the different analysis directions of the image, the detection model can be divided into three categories: statistics-based methods, spectrum-based methods, and model-based methods. Latif-Amet et al.[6] proposed the sub-band co-occurrence matrix (SBCM) to detect defects encountered in textile images. It combines wavelet theory and co-occurrence matrices, decomposes the gray level images into sub-bands, and then divides the image into non-overlapping windows to extract co-occurrence features. Nevertheless, the reliability of the method and how it can be applied to other fabrics is not clear. Tsai and Molina[7] proposed the morphological method of arc SE for machined surface inspection. This morphology method can effectively remove tool-marks and highlights local defects, but it has greater limitations and only targets for the surfaces with circular tool-marks. Given the strong texture characteristics of some fabric images, some scholars consider adopting the method of spectrum analysis to detect fabric defects. Anandan and Sabeenian[2] utilize the immediate duplication of curvelet change information at adjoining scales to recognize critical edges from the clamor. And use Curvelet Transform (CT) and Gray-Level Co-event Matrices (GLCM) to find potential texture defects. Nevertheless, the real-time performance of the algorithm is not mentioned. Jing et al.[8] combines the genetic algorithm with the Gabor filter to match the fabric defect-free image texture information,

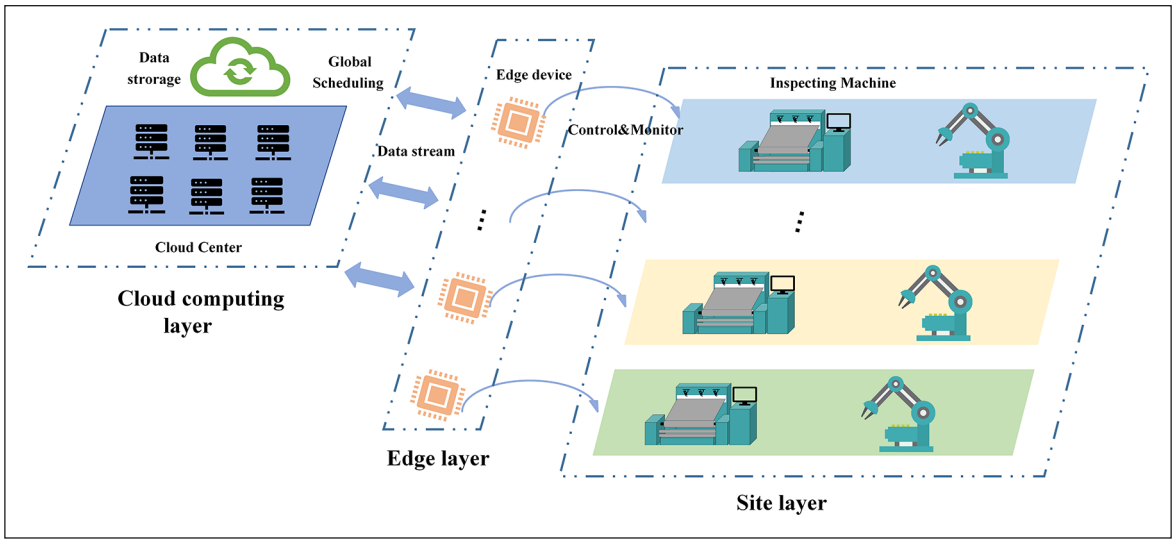

Figure 1. Edge computing deployment view.

and then use the adjusted Gabor filter to detect defects on the fabric. But the parameters selected in the Gabor filter is a challenging task in defect detection problem. When dealing with fabric with intricate textures, the model-based method is more suitable. The complex textures can be modeled as a stochastic process and treat the defect detection problem as a statistical hypothesis-testing problem on the statistics derived from the model. Xiaobo[11] and Allili et al.[12] respectively apply the Gaussian-Markov random field (GMRF) model and Gaussian mixture model to extract fabric features and detection. However, these methods are computationally expensive and have low accuracy for small defects.

Deep learning methods for defect detection

On account of the limitations of traditional methods, more and more scholars have begun to study detection methods based on deep learning in recent years. For example, Quyang et al.[17] proposed a deep learning-based defect detection method on on-loom by combining the techniques of pre-processing, fabric motif determination, candidate defect map generation, and CNNs. Li et al.[18] proposed a compact CNN architecture and applied it to some common fabric defects. Liu et al.[19] proposed a multistage GAN network, which generates defect samples by training multistage GAN and detect them through a semantic segmentation network. It performs well on the accuracy metric of various fabric datasets. However, the paper does not mention the effect of the model in practical application, such as whether the detection speed meets the requirements. Zhao et al.[20] proposed a multi-defect detection pipeline and applied it to Fuxing Electric Multiple Units. It improves the anchor and feature fusion mechanism of Region Proposal Network (RPN) and combines the super-resolution strategy with CNN in the classification stage to improve classification performance. Although deep learning methods are superior to traditional methods in generality, the deployment of their systems often requires substantial computing resources. When computing resources are insufficient, the detection performance of the system will be seriously reduced.

Edge computing and its application

Most of the current deep learning methods are combined with cloud computing. In order to meet excessive computing needs, cloud computing has adopted a powerful data center for intensive processing. However, in this cloud-centric approach, data jams caused by the transmission of masses data (i.e. images and videos) will greatly influence production efficiency.[21] To alleviate this problem, the combination of edge computing and deep learning came into being. Edge computing is an open platform that sinks the computing capability and storage facilities to the side close to users or data sources and integrates core functions of network, computing, storage, and applications.[22] The deployment diagram of edge computing is shown in Figure 1. Edge computing uses the advantages of distributed deployment and closer data sources, which can effectively solve problems such as high broadband cost, transmission latency, and data congestion.[23] This novel pattern enables computation-intensive and latency-critical applications. With the accelerated application of 5 G, the application of edge computing will continue to expand, such as industrial Internet, smart city, and smart transportation.[24] Lin et al.[25] combines edge computing framework with the deep Q network (DQN) to solve complex job shop scheduling problems (JSP), which performs better than the other methods that only use one dispatching rule. Liu et al.[26] proposed a video recognition system for

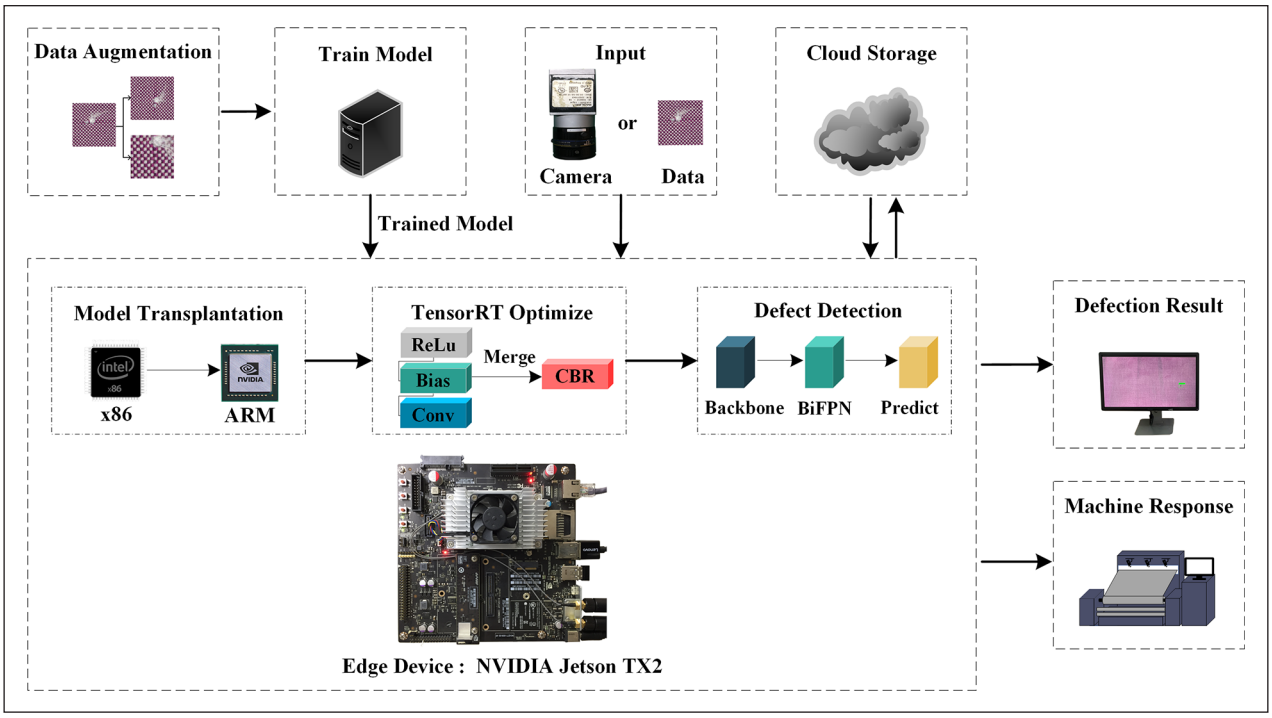

Figure 2. Defect detection method.

dietary assessment based on edge devices. Liu et al.[27] proposed the application of edge computing to public intelligent monitoring and tracking.

Proposed method

To satisfy the requirements of the industrial automatic fabric defect detection system, we proposed an automated fabric defect detection method based on edge computing.

The overall framework of our method is shown in Figure 2. In 3.1, we first elaborate the feasibility of the proposed method from the three requirements of industrial defect detection. Then introduce the detection algorithm used in this paper, as described in 3.2. To empower the model has the capability of real-time industrial defect detection, we optimize the trained model with NVIDIA TensorRT and introduce it 3.3.

Design basis

In the industrial scenario, automatic fabric defect detection needs three necessary attributes. First, the automatic fabric defect detection system needs to match the high-speed production line and respond to the defective fabric promptly. This requires not only fast detection algorithm but also fast system response. Second, the production line should be characterized by low power consumption to reduce the production cost. Third, in order to make the production line can be upgraded conveniently in the future, the detection system must have the characteristic of scale. In response to the above requirements, we adopt the detection system based on edge computing and using EfficientDet-D0[28] algorithm. As described, edge computing can offload computing tasks from the cloud to the edge device, which not only improves the detection speed but also protects the privacy of data in the production process of enterprises. Likewise, the reduction of network bandwidth also reduces the production cost of enterprises. EfficientDet uses the lightweight, scalable backbone and feature extract network, the params and FLOPs of the Efficientdet-D0 are only 3.9 M and 2.54 B, respectively, which gives it an absolute advantage in detection speed. The addition of edge equipment can effectively reduce the power consumption of the production line and system response time, as well as the production investment of enterprises. At the same time, the multi-scale characteristic of EfficientDet model accords with the requirement that the industrial production line needs time and convenient upgrade.

In order to enhance the robustness of the model in the real industrial scenario, we augmented the training datasets, including random cropping, flipping, blurring, brightness gain, and noise. We applied them to five different fabric datasets to verify the detection effect. The random crop strategy is used to simulate the sample that collected from various camera perspectives. Random flip is used to expand the data to improve the quality of the training model.[29] A random brightness gain can mimic different lighting conditions. The noise and blur can enhance the robustness of the model.

After the model is trained on the local workstation, we transplanted it to the edge device NVIDIA Jetson TX2. To further improve the detection speed, we use TensorRT to

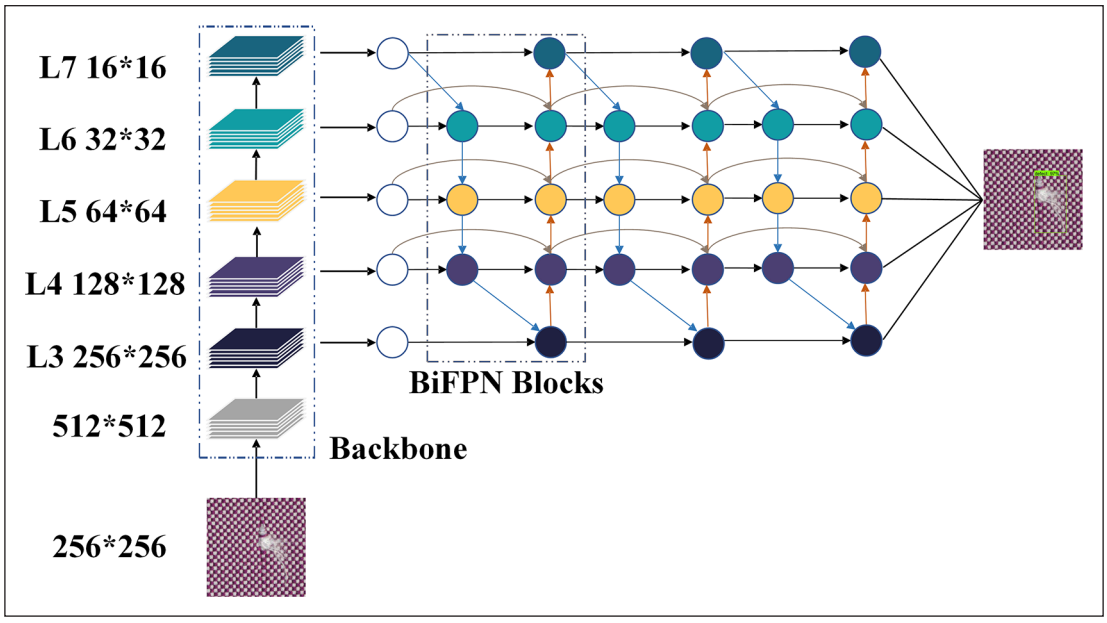

Figure 3. EfficientDet-D0 architecture.

optimize and accelerate the model. The response latency of the detection system in cloud computing and edge computing were compared to verify the feasibility of the proposed approach.

EfficientDet architecture

EfficientDet is a lightweight, scalable detection Network, and it contains a total of eight models, D0–D7. From D0 to D7, the accuracy and time complexity of the model increases with the model size. The eight models can meet a broad spectrum of resource constraints. The backbone of the EfficientDet employs EfficientNet,[30] which uses a mass of deep separable convolution[31] to make the model more lightweight. Moreover, the neck part of the network uses the bidirectional weighted feature pyramid network (BiFPN). Starting from the path aggregation network (PANet),[32] BiFPN removes the nodes that only have one input edge, then add a shortcut between the input and output node if they are the same level. The purpose is to fusing more features without increase the computation cost. Since the resolution of different input features, additional weight was added on each input features during features fusion. This weighted fusion make the network learn the importance of different features. We use the feature in level 5 as a concrete example to describe the fusion mechanism:

$$F_5^{mid} = Conv\left\{\frac{\delta_1 \cdot F_5^{in} + \delta_2 \cdot Resize(F_6^{mid})}{\delta_1 + \delta_1 + \theta}\right\} \quad (1)$$

$$F_5^{out} = Conv\left\{\frac{\delta_1' \cdot F_5^{in} + \delta_2' \cdot F_5^{mid} + \delta_3' \cdot Resize(F_4^{out})}{\delta_1' + \delta_2' + \theta}\right\} \quad (2)$$

where F_i^{in} is the input feature at L_i, L_i is the feature level. F_i^{mid} represents the intermediate feature on the top-down pathway at L_i. And F_i^{out} is the output feature on the bottom-up pathway at L_i. θ is used to avoid numerical instability. δ is the weighted for feature fusion. After the BiFPN, all the fused features are processed by a class and box prediction network to predict and locate defects. Figure 3 shows the structure of the network. The BiFPN, box, and class prediction network will be repeated times according to the size of different model. Different size of the backbone, BiFPN, and prediction network constitute the eight EfficientDet model which can adjust to different computing resources.

TensorRT optimization

TensorRT is an optimizer provided by NVIDIA specifically for the neural network inference stage. It mainly optimized from two aspects: Layer & Tensor fusion and Weight & Activation Precision Calibration. When the neural network performs inference calculations, the calculations of each layer must be completed by Compute Unified Device Architecture (CUDA) in the GPU, which makes much time wasted on the startup of CUDA and the read and write operations on each layer. TensorRT greatly reduces the number of layers by merging horizontally or vertically between layers (the merged structure is called CBR, which represents convolution, bias, and ReLU, and all these are merged into one layer). Vertical merging can merge convolution, bias, ReLu into a CBR structure, and only occupy one CUDA core. Horizontal merging can merge CBRs with the same structure but different weights into a wider layer, also

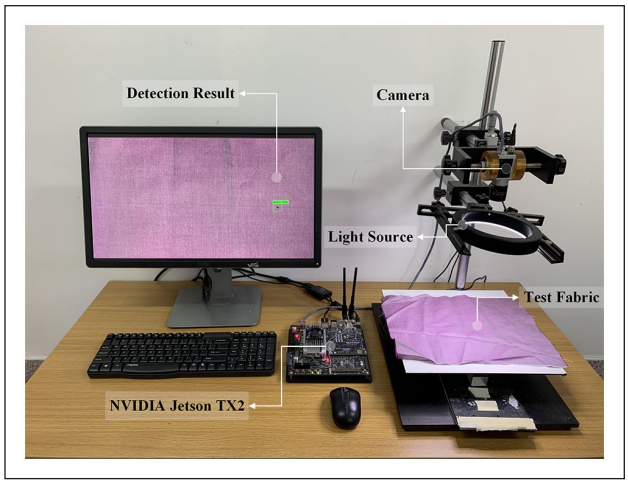

Figure 4. Detection device based on Jetson TX2.

occupy one CUDA core. As a result of the merge, there are fewer levels of the calculation graph and fewer CUDA cores to use so that the overall model structure will be smaller, faster, and more efficient.

Experiments and discussion

In this section, we will evaluate the performance of the proposed method through a series of experiments. All the local experiments were performed on a local workstation configured with Intel i7-5930K processor (3500 MHz), 64 GB memory, and a NVIDIA GeForce GTX TITAN X GPU. The software part used the Windows 10 operating system and Tensorflow 2.1 deep learning framework. All the experiments on the edge were performed on NVIDIA Jetson TX2 edge device. NVIDIA Jetson TX2 is an efficient and fast embedded Artificial intelligence (AI) computing device. The GPU adopts NVIDIA Pascal architecture with 256 CUDA cores and 8GB memory. The processor cluster consists of dual-core Denver2 processor and 4-Core ARM architecture cortex-A57, power consumption is only 7.5 W. It is suitable for edge computing scenarios. Figure 4 shows the Jetson TX2 and its detection device.

Datasets and augmentation

In this experiment, we used five different fabric datasets, in which all images are derived from the real-word fabric production line and manually annotated. Furthermore, the camera used to capture textile image data is Basler acA2500-14gm, with GigE interface. And the lens adopts Basler Lens C125-0618-5M-P with a resolution of 5 megapixels and a fixed focal length of 6 mm. Meanwhile, according to the characteristics of textiles, we choose the Opt white ring LED light source, the specific model is OPI-RI5000-W. When collecting images, the distance between the textile and the lens is 320mm, and the distance between the light source and the textile is 270 mm. The fabric types of the five datasets are dark red fabric (DRF), grid fabric (GF), camouflage fabric (CF), digital printed fabric (DPF), and light blue fabric (LBF), as shown in Figure 5.

In fabric defect detection, the number of datasets is not as sufficient as we think, and the lack of samples is a challenge to the detection methods based on deep learning. Therefore, we augmented the five datasets to improve the training effect and industrial robustness of the model. Take the DRF dataset as an example, In the case of the dark red fabric data set, we employed random cropping, blurring, brightness gain, flip, and noise for each original image, as shown in Figure 6.

We divided all images into two parts: 90% as the training set and 10% as the test set. The sample numbers of training sets and test sets in the five fabric datasets are shown in Table 1. The five datasets include fabrics with different types of textures, including both regular-textured fabrics and random-patterned fabrics, and the types of defects span an extensive range (such as large-area defects in DRF and tiny defects in CF), which can verify the proposed method in the performance of different types of fabrics and defects. Please note that before training, we normalize all images size to 256×256.

Evaluation metrics

The evaluation metrics in this experiment includes two parts, detection speed and detection accuracy. We utilized frame per second (FPS) to evaluate detection speed and mean Accuracy Precision (mAP) to evaluate the detection accuracy of the proposed method. The evaluation metrics FPS and mAP are defined as follows:

$$mAP = \frac{\sum AP_C}{N} \tag{3}$$

where AP_C represents the average accuracy of each type of defect, C represents which type of defect; N is the total number of categories.

$$FPS = \frac{F_{TotalFrame}}{T_{TotalTime}} \tag{4}$$

where $F_{TotalFrame}$, $T_{TotalTime}$ are the total number of detection frames and the total time spent in detection, respectively.

Experiment on the workstation

Given the limited computing capability of the edge device, we train the model on the local workstation and then transplant the trained model to the edge device to reduce the

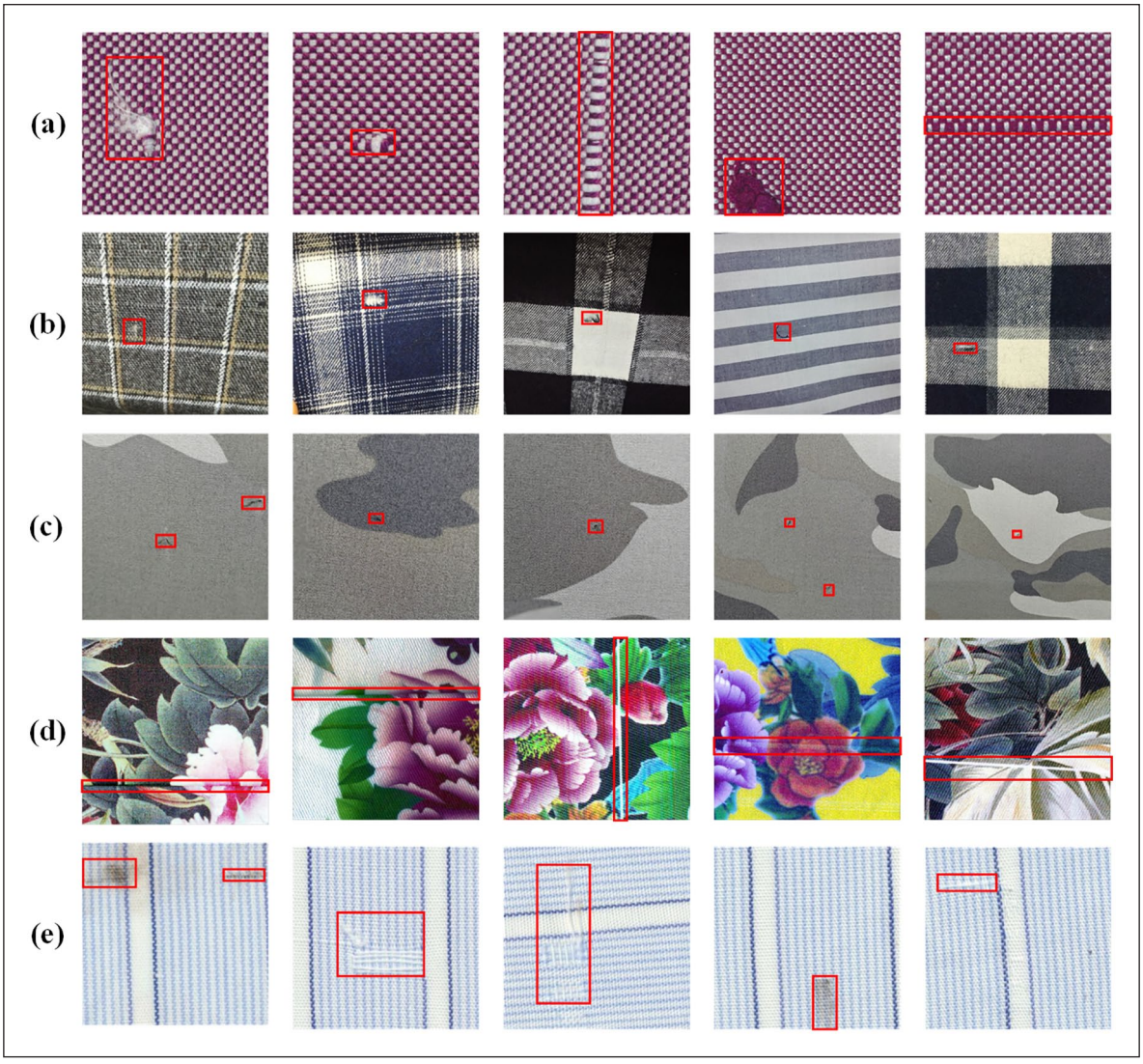

Figure 5. Five fabric datasets: (a) DRF, (b) GF, (c) CF, (d) DPF, and (e) LBF.

training cycle of the model. The Box prediction network of EfficientDet is based on anchors, so we reference the YOLO-V2[33] anchors clustering method to achieve better training results. According to the network settings, we perform k-means clustering on the training data, and the final anchor box ratio is shown in Table 2. At the same time, we also compared the current mainstream one-stage object detection methods, including YOLO-V3, SSD,[34] RetinaNet.[35]

Qualitative analysis. All models are compared on the five fabric datasets listed in Table 1. The test results of different models are shown in Figure 7, where Figure 7(a) and (b), (c) and (d), (e) and (f), (g) and (h), (i) and (j) are the test results of LBP, CF, DRF, GF, DPF datasets, respectively. For the LBF dataset, the four models can effectively detect defects, but the location accuracy is different. From the detection results in column (a), there are more non-target areas in the detection results of all models, except for EfficientDet-D0. For the tiny defects in the CF dataset, RetinaNet and YOLO-V3 have missed and falsely detected, respectively, as shown in the columns of Figure 7(c) and (d). For defects across the entire sample, the detection results of all models are not optimal. Intuitive displays such as the DRF and DPF datasets. In both datasets, there are defects that across the entire sample, but neither RetinaNet nor SSD can detect this type of defect, although YOLO-V3 has detected defects, there are too many defect-free areas detected. In the column e of the DRF dataset, RetinaNet only bounding half of the defect area. In the GF dataset, the shape of the defect is small, and there are multiple defects in a single sample. Except for RetinaNet, other models have detected multiple defects, but the bounding box predicted by the model is quite different from the ground

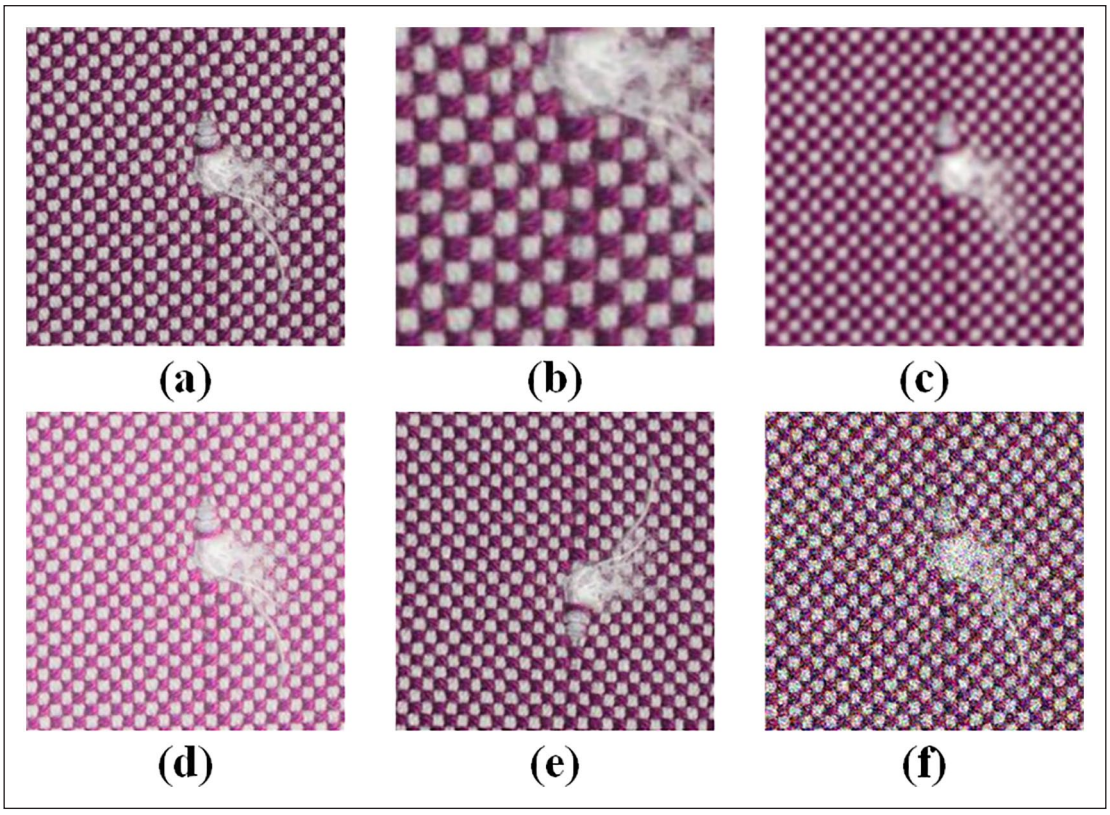

Figure 6. Data augmentation: (a) original, (b) crop, (c) blur, (d) brightness, (e) flip, and (f) noise.

Table 1. Distribution of augmented datasets.

Dataset	Training set	Test set	Total	Image size
DRF	523	59	582	256 × 256
GF	461	52	513	256 × 192
CF	540	60	600	256 × 256
DPF	463	51	514	1024 × 1024
LBP	625	69	694	256 × 256

Table 2. Anchor box ratio for five datasets.

Dataset	Anchor 1	Anchor 2	Anchor 3
DRF	60:53	30:200	255:25
GF	17:30	22:15	28:38
CF	15:14	21:21	21:36
DPF	52:1002	91:1022	1021:53
LBP	34:95	52:45	88:77

truth, except EFficientDet-D0. Judging from the overall visual contrast, the Efficientdet-D0 model achieves better detection effect in five different fabric datasets, and it can detect the sample defect as well as the predicted bounding box is relatively accurate.

Quantitative analysis. FPS and mAP were respectively used to quantitatively analyze the accuracy and speed of the model. The detection accuracy of the five datasets is shown in Figure 8 and Table 3. By comparison, the Efficientdet-D0 model achieves better performance in all five datasets. In the longitudinal comparison, in the GF data set, the sample background is changeable, and there are many small defects. The detection accuracy of EfficientDet-D0 reaches 98%, which is higher than the other three models and indicates its strong ability to detect tiny defects. In the DPF data set, the aspect ratio of defects is large, and most of the defects across the entire sample. In the detection of such defects, EfficientDet-D0 is better than other models. In the horizontal comparison, EfficientDet-D0 can obtain high mAP on all five datasets, showing its high robustness. At the same time, we compare the mean precision and standard deviation of the model on five datasets. Efficientdet-D0 has the highest mean precision with the lower standard deviation, which shows its excellent generalization.

Figure 9 and Table 4 shows the detection speed of the compared models on the same hardware platform, respectively. The fastest model is EfficientDet-D0, which has a detection speed of 42 FPS and a single detection time of 23.8 ms. The detection speed of YOLO-V3 is close to that of SSD, reaching 23 FPS and 24 FPS, respectively, and the

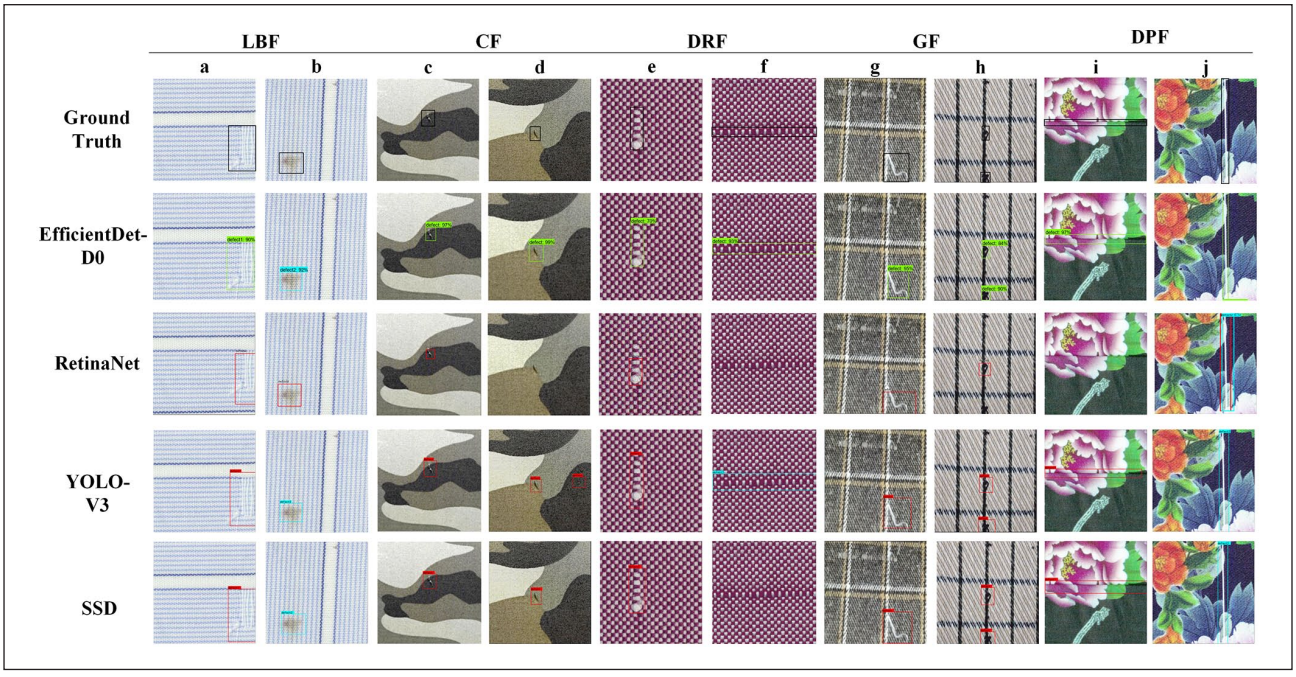

Figure 7. Visual comparison of detection results by different models in five datasets. Where a-j is the representative of the dataset.

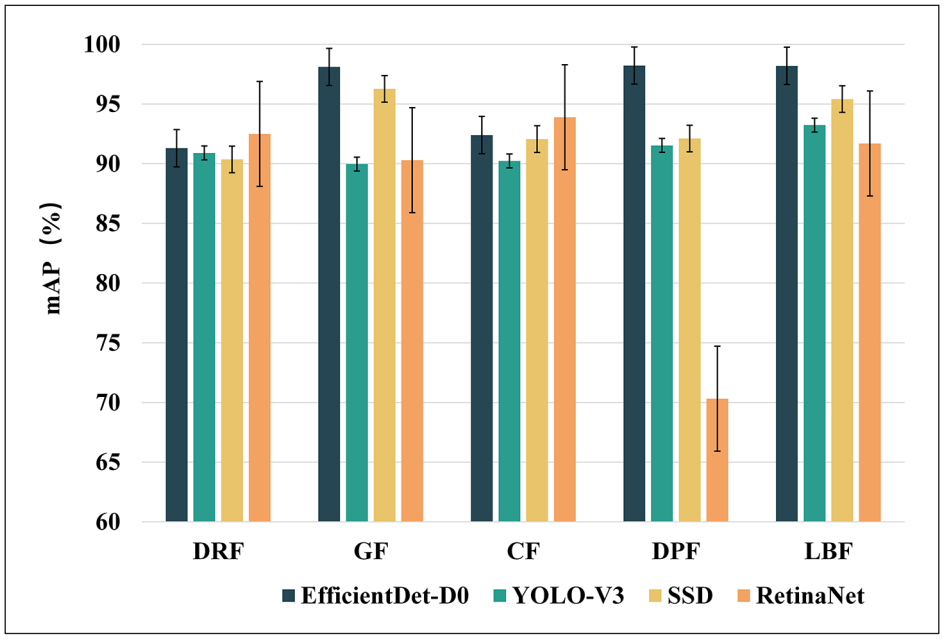

Figure 8. Comparison of detection accuracy of different datasets.

single detection time is about 41 ms. RetinaNet has the slowest detection speed, only 12.8 FPS.

Experiment on the edge device

Taking into account the real-time and low-power requirements of industrial defect detection, we transplanted the model trained on the local workstation to our edge computing device Jetson TX2. We compared the inference time of the models mentioned above on the edge device, as shown in Table 4. Note that the size of the image we sent to the network for inference is 256×256, and the final output size is 512×512. As shown in Figure 10 and Table 5, due to the limited computing resources of

Table 3. Comparison of detection accuracy (mAP).

	DRF (%)	GF (%)	CF (%)	DPF (%)	LBP (%)	Mean precision (%)	Standard deviations
EfficientDet-D0	91.30	98.11	92.40	98.22	98.20	95.65	3.12
YOLO-V3	90.90	89.97	90.23	91.53	93.23	91.17	1.16
SSD	90.36	96.27	92.06	92.11	95.41	93.24	2.23
RetinaNet	92.50	90.30	93.90	70.32	91.70	87.74	8.79

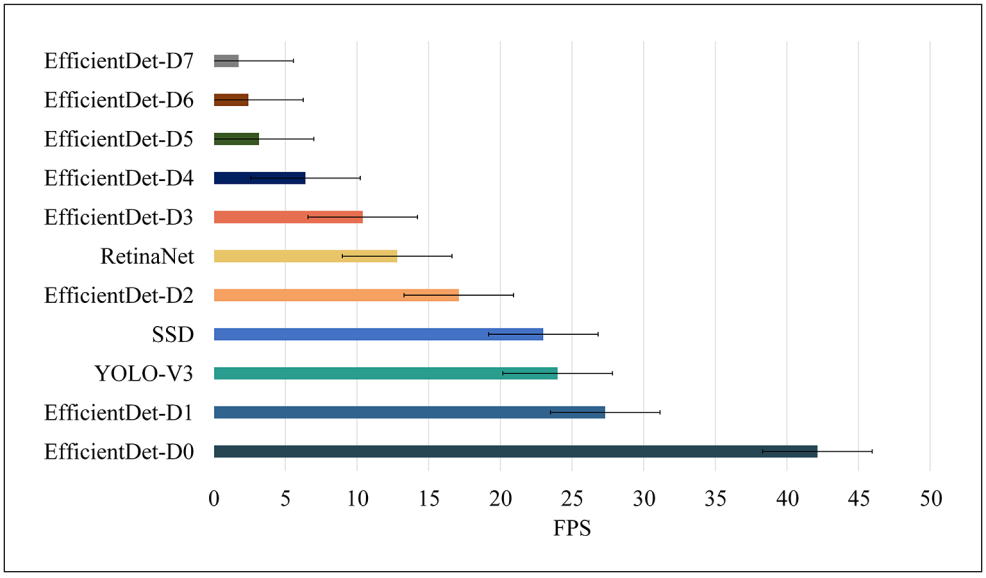

Figure 9. Comparison of inference time for different models on workstation.

Table 4. Inference time on workstation.

Dataset	FPS	Inference time (ms)
EfficientDet-D0	42.00	23.81
EfficientDet-D1	27.00	37.03
YOLO-V3	24.00	41.67
EfficientDet-D2	17.11	58.45
RetinaNet	12.80	78.13
SSD	23.00	43.48
EfficientDet-D3	10.41	96.15
EfficientDet-D4	6.40	156.13
EfficientDet-D5	3.16	315.99
EfficientDet-D6	2.42	412.97
EfficientDet-D7	1.73	578.51

edge devices, the detection speed of all models transplanted to Jetson TX2 has decreased. In order to improve the speed of the model, we adopted TensorRT optimization for the EfficientDet-D0. The optimized model has a shorter inference time, which is more in line with the requirements of industrial detection speed. In this experiment, we adopted a single-precision optimization strategy for the model. The comparison of the model before and after acceleration is shown in Figure 11.

After the model was optimized by TensorRT, the inference time of the model was shortened from 72.8 to 43.9 ms, and the FPS increased from 13.7 to 22.7. The detection time of the model is reduced by nearly half, and the speed is improved by nearly twice, with the capability of industrial real-time detection.

We used the optimized model to test the five datasets, and the detection results were shown in Figure 12. The accelerated model performs well in the detection of five datasets.

In order to verify the advantages of edge computing in industrial defect detection, we compared the corresponding latency of edge computing and cloud computing. In this experiment, we test the data transmission latency from the cloud server to the local. Although cloud computing takes less time to compute, data transmission consumes much time, which slows down the overall detection speed. In edge computing, edge devices can directly process received images without uploading data to the cloud and downloading detection results, which reduces most of the time and improves the overall response speed. We use the Ali cloud server as the test platform to measure the data

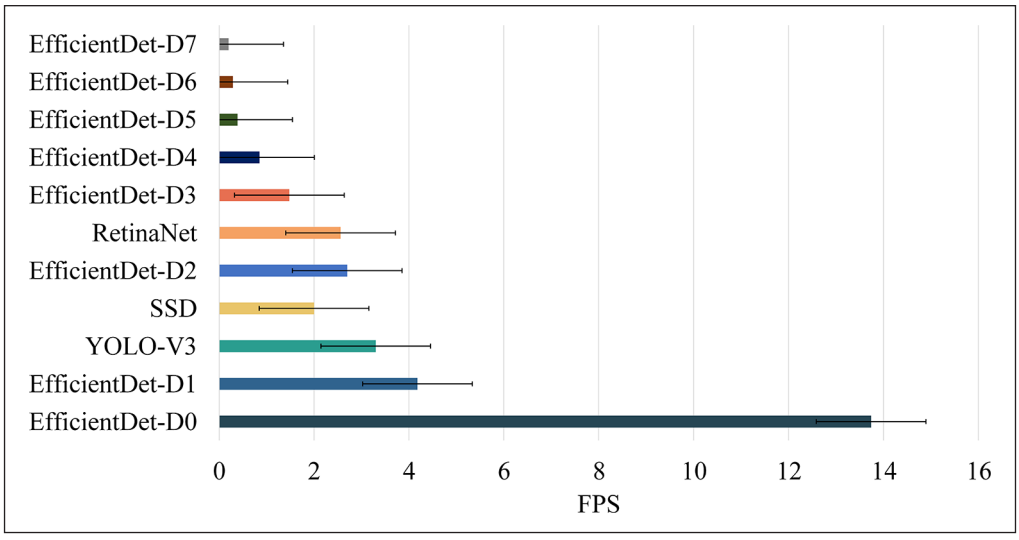

Figure 10. Comparison of inference time for different models on Jetson TX2.

Table 5. Inference time on Jetson TX2.

Dataset	FPS	Inference time (ms)
EfficientDet-D0	13.74	72.80
EfficientDet-D1	4.18	239
YOLO-V3	3.30	300
EfficientDet-D2	2.70	369
RetinaNet	2.56	390
SSD	2.00	500
EfficientDet-D3	1.48	674
EfficientDet-D4	0.85	1182
EfficientDet-D5	0.39	2550
EfficientDet-D6	0.29	3422
EfficientDet-D7	0.20	4960

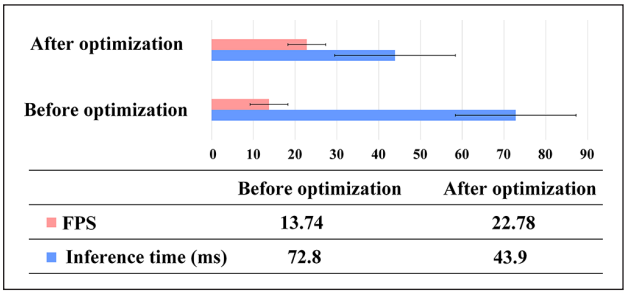

Figure 11. Comparison before and after EfficientDet-D0 model optimization.

transmission latency from the local to the cloud server through the HTTP protocol. The data transmission latency from cloud to local is shown in Table 6.

The local bandwidth used in this experiment is 300M, and the Ali cloud server has 1M. There are seven hops from local to the cloud. In Table 6, the latency in original images upload and result return include resource scheduling, establishing connections, and all the time it takes to complete data transfer. Except for resource scheduling and connection establishment latency, the exact latency for uploading data is 48.39 ms, and the data transfer latency for returning results from the cloud is 35.70 ms. If we remove the connection establishment time (after the first connection is established), from original images upload to results return, compared with cloud computing, the time consumed by edge computing is reduced by 2.5 times. After comparison, it can be seen that edge computing occupies a significant advantage in detection speed, which is a feasible detection solution for industrial applications that require high detection speed.

Conclusions

In this paper, we propose a fabric defect detection method based on edge computing which satisfies the requirements of industrial defect detection production line for a low response, low power consumption, and convenient upgrade. Combining the lightweight and scalable EfficientDet algorithm with NVIDIA Jetson TX2 can better address the latency and power consumption issues. Meanwhile, data augmentation is used to improve the detection performance of the model. Furthermore, TensorRT optimize is utilized to accelerate the detection speed of the model for fast defect detection. The comparisons experimental indicate that the response time of our method is 2.5x faster than the cloud computing-based detection method. Therefore, the proposed edge computing-based detection approach shows significant potential and advantages in real-time industrial defect detection.

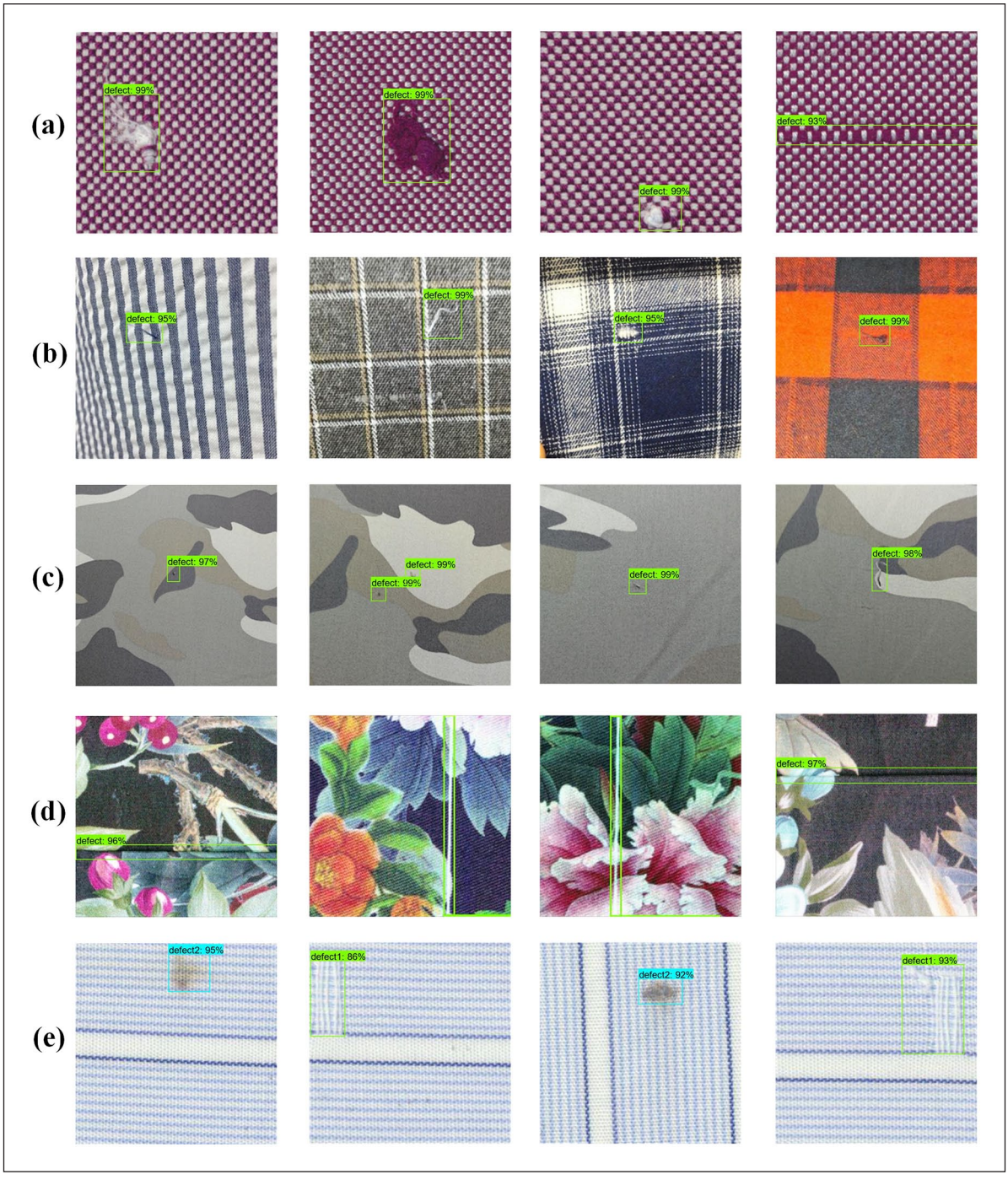

Figure 12. Detection results: (a) DRF, (b) GF, (c) CF, (d) DPF, and (e) LBF.

Table 6. Latency comparison.

Steps	Cloud computing (ms)	Edge computing (ms)
Original images upload	99.80	/
Inference	23.80	43.90
Result return	85.30	/

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article:

This work was supported in part by the Shaanxi Provincial Education Department under Grant 19JC018.

ORCID iD

Junfeng Jing https://orcid.org/0000-0001-6646-3698

References

1. Yapi D, Allili MS and Baaziz N. Automatic fabric defect detection using learning-based local textural distributions in the contourlet domain. *IEEE Trans Autom Sci Eng* 2018; 15(3): 1014–1026.
2. Anandan P and Sabeenian RS. Fabric defect detection using discrete curvelet transform. *Procedia Comput Sci* 2018; 133: 1056–1065.
3. Kang X and Zhang E. A universal defect detection approach for various types of fabrics based on the elo-rating algorithm of the integral image. *Text Res J* 2019; 89(21–22): 4766–4793.
4. Rongsheng L, Ang W, Tengda Z, et al. Review on automated optical (visual) inspection and its applications in defect detection. *Acta Opt Sin* 2018; 38(8): 0815002.
5. Zhu Z, Han G, Jia G, et al. Modified DenseNet for automatic fabric defect detection with edge computing for minimizing latency. *IEEE Internet Things J* 2020; 7(10): 9623–9636.
6. Latif-Amet A, Ertüzün A and Erçil A. An efficient method for texture defect detection: sub-band domain co-occurrence matrices. *Image Vis Comput* 2000; 18(6–7): 543–553.
7. Tsai DM and Molina DER. Morphology-based defect detection in machined surfaces with circular tool-mark patterns. *Measurement* 2014; 44(1): 40–57.
8. Jing J, Yang P, Li P, et al. Supervised defect detection on textile fabrics via optimal Gabor filter. *J Ind Text* 2014; 44: 40–57.
9. Jia L, Chen C, Liang J, et al. Fabric defect inspection based on lattice segmentation and Gabor filtering. *Neurocomputing* 2017; 238: 84–102.
10. Jing J, Liu S, Li P, et al. The fabric defect detection based on CIE L*a*b* color space using 2-D Gabor filter. *J Text Inst* 2016; 107(10): 1305–1313.
11. Xiaobo Y. Fabric defect detection of statistic aberration feature based on GMRF model. *J Text Res* 2013; 4: 26.
12. Allili MS, Baaziz N and Mejri M. Texture modeling using contourlets and finite mixtures of generalized Gaussian distributions and applications. *IEEE Trans Multimedia* 2014; 16(3): 772–784.
13. Cha YJ, Choi W and Büyüköztürk O. Deep learning based crack damage detection using convolutional neural networks. *Comput Aided Civ Infrastruct Eng* 2017; 32(5): 361–378.
14. Canny J. A computational approach to edge detection. *IEEE Trans Pattern Anal Mach Intell* 1986; PAMI-8(6): 679–698.
15. Jing J, Zhuo D, Zhang H, et al. Fabric defect detection using the improved YOLOv3 model. *J Eng Fiber Fabr* 2020; 15: 1558925020908268.
16. Redmon J and Farhadi A. Yolov3: an incremental improvement. *arXiv preprint* arXiv:1804.02767, 2018.
17. Ouyang W, Xu B, Hou J, et al. Fabric defect detection using activation layer embedded convolutional neural network. *IEEE Access* 2019; 7: 70130–70140.
18. Li Y, Zhang D and Lee DJ. Automatic fabric defect detection with a wide-and-compact network. *Neurocomputing* 2019; 329: 329–338.
19. Liu J, Wang C, Su H, et al. Multistage gan for fabric defect detection. *IEEE Trans Image Process* 2019; 29: 3388–3400.
20. Zhao B, Dai M, Li P, et al. Defect detection method for electric multiple units key components based on deep learning. *IEEE Access* 2020; 8: 136808–136818.
21. Li E, Zeng L, Zhou Z, et al. Edge AI: on-demand accelerating deep neural network inference via edge computing. *IEEE Trans Wirel Commun* 2019; 19(1): 447–457.
22. Shi W, Sun H, Cao J, et al. Edge computing-an emerging computing model for the internet of everything era. *J Comput Res Dev* 2017; 54(5): 907–924.
23. Khan WZ, Ahmed E, Hakak S, et al. Edge computing: a survey. *Future Gener Comput Syst* 2019; 97: 219–235.
24. Zhang K, Zhu Y, Maharjan S, et al. Edge intelligence and blockchain empowered 5G beyond for the industrial internet of things. *IEEE Netw* 2019; 33(5): 12–19.
25. Lin CC, Deng DJ, Chih YL, et al. Smart manufacturing scheduling with edge computing using multiclass deep Q network. *IEEE Trans Ind Inform* 2019; 15(7): 4276–4284.
26. Liu C, Cao Y, Luo Y, et al. A new deep learning based food recognition system for dietary assessment on an edge computing service infrastructure. *IEEE Trans Serv Comput* 2017; 11(2): 249–261.
27. Liu G, Liu S, Muhammad K, et al. Object tracking in vary lighting conditions for fog based intelligent surveillance of public spaces. *IEEE Access* 2018; 6: 29283–29296.
28. Tan M, Pang R and Le QV. EfficientDet: scalable and efficient object detection. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, Seattle, WA, 13–19 June 2020, pp.10781–10790. IEEE.
29. Shorten C and Khoshgoftaar TM. A survey on image data augmentation for deep learning. *J Big Data* 2019; 6(1): 60.
30. Tan M and Le QV. EfficientNet: rethinkingmodelscaling for convolutional neural networks. *arXiv preprint* arXiv:1905.11946, 2019.
31. Chollet F. Xception: deep learning with depthwise separable convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, Honolulu, HI, 21–26 July 2017, pp.1251–1258. IEEE.
32. Liu S, Qi L, Qin H, et al. Path aggregation network for instance segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, Salt Lake City, UT, 18–23 June 2018, pp.8759–8768. IEEE.
33. Redmon J and Farhadi A. YOLO9000: better, faster, stronger. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, Honolulu, HI, 21–26 July 2017, pp.7263–7271. IEEE.
34. Liu W, Anguelov D, Erhan D, et al. Ssd: single shot multibox detector. In: *European conference on computer vision*, Amsterdam, The Netherlands, 11–14 October 2016, pp.21–37. Cham: Springer.
35. Goyal P and Kaiming H. Focal loss for dense object detection. *IEEE Trans Pattern Anal Mach Intell* 2018; 39: 2999–3007.