Anomaly Sewing Pattern Detection for AIoT System using Deep Learning and Decision Tree

Nguyen Quoc Toan, Seongwon Cho

Abstract

Artificial Intelligence of Things (AIoT), which combines AI and the Internet of Things (IoT), has recently gained popularity. Deep neural networks (DNNs) have achieved great success in many applications. Deploying complex AI models on embedded boards, nevertheless, may be challenging due to computational limitations or intelligent model complexity. This paper focuses on an AIoT-based system for smart sewing automation using edge devices. Our technique included developing a detection model and a decision tree for a sufficient testing scenario. YOLOv5 set the stage for our defective sewing stitches detection model, to detect anomalies and classify the sewing patterns. According to the experimental testing, the proposed approach achieved a perfect score with accuracy and F1score of 1.0, False Positive Rate (FPR), False Negative Rate (FNR) of 0, and a speed of 0.07 seconds with file size 2.43MB.

Keywords : Illumination compensation | Illumination invariance | Face tracking | YCbCr | Skin locus

I. INTRODUCTION

Industry 4.0 refers to the integration of digital technology into industrial processes to create highly automated and connected factories. This is achieved through the use of machine learning and the Internet of Things. However, sending large amounts of data to the cloud can create challenges in terms of speed, bandwidth, security, and cost. One solution to this is edge computing, which allows for local tasks to be performed and controlled directly on embedded devices, rather than relying on the cloud. Computer vision plays a vital role in various industries, often utilizing smart cameras to carry out tasks such as inspecting and measuring parts or guiding robots [1]. These cameras are equipped with a built-in sensor and a small computer that can execute the manufacturer's image library. However, current options available in the market have limited deep learning capabilities, making it difficult to customize using closed-source software or limited APIs, and are also costly. Several hardware companies are developing alternative edge devices as a substitute for smart cameras to make it possible for engineers to create more cost-effective and cutting-edge vision solutions. Google's Coral Edge TPU, for example, offers a favorable combination of attributes for inference neural networks.

* This work was supported by hongik University and Ministry of SMEs and Startups

Manuscript : 2023.12.19
Revised : 2024.02.13
Confirmation of Publication: 2024.02.18

Corresponding Author : Seongwon Cho, e-mail : swcho@hongik.ac.kr

II. RELATED WORK

Several research papers [3-5] have examined the most significant models and achievements in detection in the past few years. The success of AlexNet at the ImageNet Large Scale Visual Recognition Challenge in 2012 [6] was a game changer in deep learning-based detection. This event marked the start of the development of two-stage detectors, which work by generating proposals first and then classifying them as potential objects. With a single detection pipeline, more recent one-stage detectors classify each region of interest and label it as either an object or a background. In recent years, there has been a renewed emphasis on developing smaller networks for mobile applications with fast inference times and high accuracy.

Here are some famous models for mobile object detectors: First, in 2018, MobileNetV2 + SSDLite was released as an improved version of the previous MobileNet classification net- work, along with a new detection framework known as SSDLite [7]. The inverted residual structure is the network's defining feature. The residual connection is used as an expansion layer between the bottleneck layer to introduce nonlinearity via depthwise convolution. Tiny-YOLOv4 is then added to the Darknet framework. It was developed as a fast variant of YOLOv4 [8], which was released in April 2020. YOLOv4 is well-known for its bag of freebies (BoF) and bag of specials (BoS) (BoS). BoF enhances the accuracy while maintaining the same inference time, whereas BoS improves accuracy whereas incurring a minor inference cost. MobileDet is a TensorFlow based detection model that was developed in April 2020 [9]. It improves the performance of non-GPU devices such as CPU, DSP, and Edge TPU. The models were created by using regular convolutions in the search space and then effectively placing them into the network as a result of neural architecture search. This approach resulted in a family of models that outperformed most existing solutions that could be deployed on the three hardware platforms mentioned above. Another notable model is YOLOv5 [2], which was released in June 2020 by Glenn Jocher. Though it has no official research paper, Jocher is acknowledged as the creator of mosaic augmentation in the YOLOv4 [8]. This PyTorch-based model is based on YOLOv3 [10] and features improved augmentation and auto-learning bounding box anchors. The motivation behind using YOLOv5 is to compare its performance with other published models.

Several studies have been published in recent years that either used the Coral TPU as hardware or represent the state-of-the-art (SOTA) in terms of detection networks deployed on other embedded devices. Table 1 contains a summary of these studies.

The Coral TPU was used in [11,12] to implement neural networks as a component of specific applications such as face mask detection. These works, however, either uses older detection networks or focus on classification, and none of them run the Coral TPU with SOTA models. In contrast, studies [13] [14] present broader benchmarks using a variety of networks, but they do not deploy them on the Coral device. There is clearly a lack of research in applications for checking anomalies in the textile industry, specifically for defective sewing patterns.

Table 1. Summary of related work

Paper	Hardware	Neural Network
[11]	Coral TPU + RPi4	-
[12]	IPC, Xavier AGX, Xilinx Zynq	YOLOv3, Center Net, MobilenetV2 + SSDLite
[13]	Coral Dev Board	MobileNetV2 + SQLite
[14]	Drive PX2	YOLOv2-v5, Efficient 0-7
[15]	Jetson TX2	Efficient Lite, YOLOv3
[16]	Xavier AGX, NCS2, Coral TPU	GoogleNet, AlexNet
[17]	Coral TPU	Classification network
Proposed method	Coral TPU + Orange Pi 3	MobileNetV2 + SSDLite, MobileDet, YOLOv4-v5

III. PROPOSED METHOD

3.1 YOLOv5

The YOLO series has undergone several updates over the years, culminating in the release of YOLOv5 in 2020. This latest version boasts an improvement in accuracy and speed when compared to other detection methods and has strong real- time performance capabilities. Hence, YOLOv5 will be used as the baseline experimental model in this study for anomaly sewing detection. It has three components, including the input layer, backbone network, neck network, and output detection layer. An illustration of this structure can be seen in Fig. 1.

3.1.1 Backbone

The main goal of the backbone is to identify crucial features within the input image as referenced in [15]. YOLOv5 utilizes Cross Stage Partial Networks (CSPNet) and Focus as its backbone to effectively identify important aspects of the input image. CSPNet addresses the issue of repeating network optimization gradient information within the backbone network and minimizes the gradient information while enhancing the learning ability of CNNs as referenced in [16] . The backbone network employs the feature map from the base layer and then uses a dense block to transmit the replicated feature map to the next level, thereby separating the feature map from the base layer.

3.1.2 Neck

To enhance the ability of network characteristic fusion, this study adopts the CSP2 structure. The Neck is frequently utilized to construct feature pyramids as referenced in [17], which aid models in achieving good object scaling generalization. It enables the recognition of the same object in various sizes and scales. The Neck is designed to optimize the characteristics of the backbone. It often comprises several bottom-up and top- down pathways. The utilization of an up-sampling and down- sampling block is the earliest form of Neck. It reprocesses and utilizes the feature maps extracted by the backbone at different stages in an efficient way. The Neck plays a crucial role in the target detection architecture, different from SSD (single-shot detector) [18] which does not involve a feature layer aggregation process.

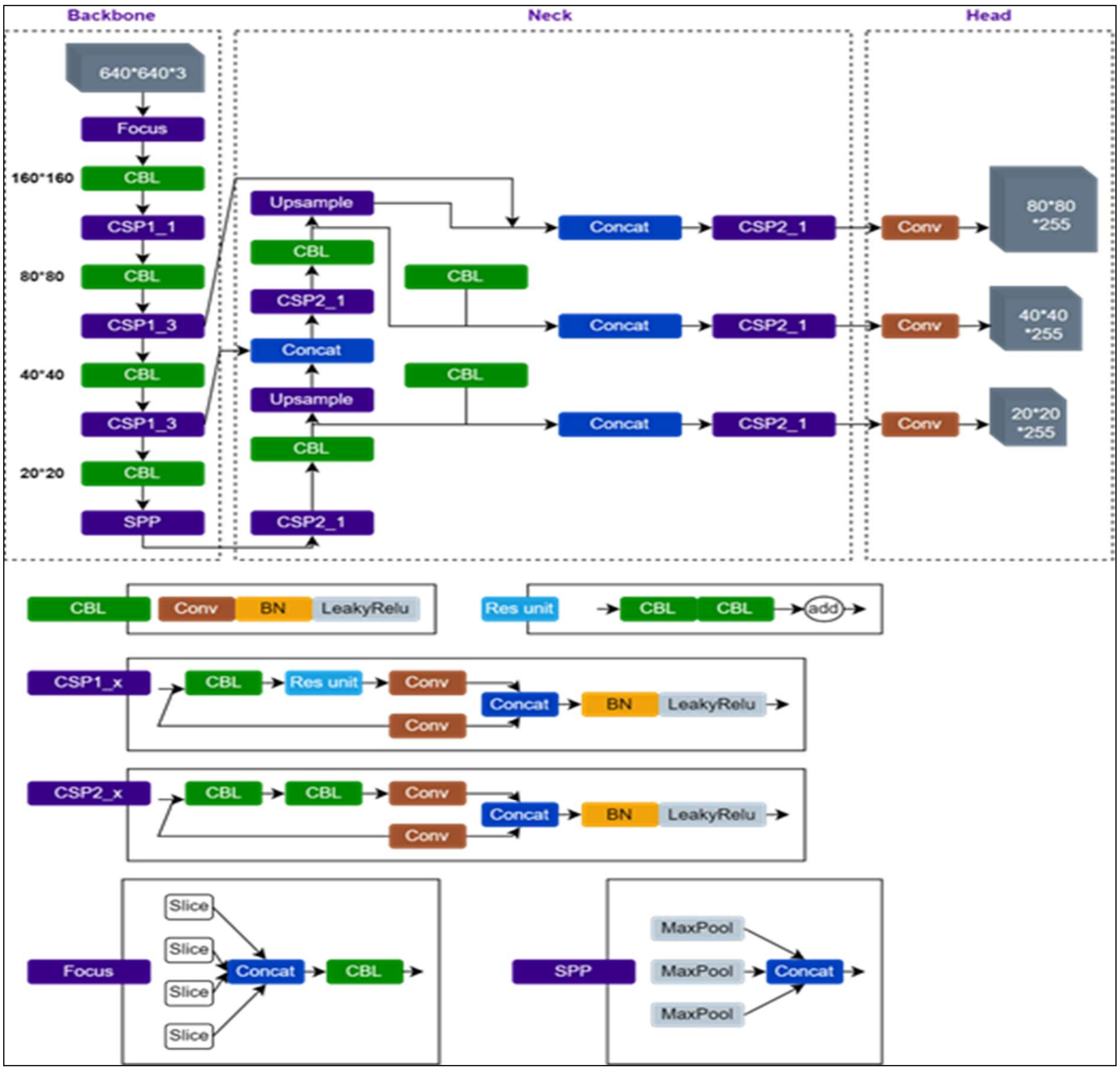

Fig. 1. Structural architecture network of YOLOv5

3.1.3 Head

The Head is responsible for the final detection process. After anchor boxes are applied to features, it generates the final output vectors with class probabilities and bounding boxes as referenced in [19]. The Head is accountable for determining the location and category of the object using the feature maps collected from the backbone. There are two types of heads: one-stage object detectors and two-stage object detectors. The RCNN (Region-based Convolutional Neural Network) series is the most representative of two-stage detectors, which have long been the dominant method in the field of object detection. The Head in the YOLOv5 model is similar to that of the Yolov3 [10] and Yolov4 [8] models, and it is mainly used in the final inspection stage as referenced in [20]. After applying anchor boxes to the feature map, it generates a final output vector comprising class probabilities, object scores, and bounding boxes.

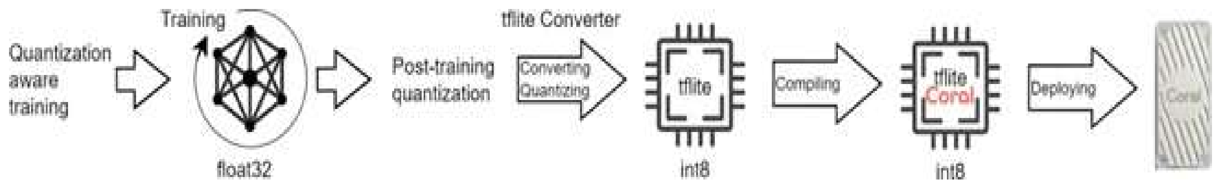

Fig. 2. Deployment of neural networks to the Edge TPU

3.2 Quantization

Edge and embedded technologies often face limitations in terms of memory and computational capabilities. To minimize the strain on these limited resources, various optimization techniques for TensorFlow models have been employed. One of the commonly adopted methods, particularly with the Edge TPU Accelerator Module, is model quantization. Quantization is a valuable approach in AI modeling as it effectively lowers latency, power consumption, and model size while retaining relatively high accuracy levels. The deployment of DNNs to the Edge TPU is a multistep process (Fig. 2). Initially, a deep learning model is transformed into the tflite format. The model with float32 is then quantized to int8 or uint8 format by [21]. Subsequently, the tflite file is processed by the Edge TPU compiler, resulting in a specialized Edge TPU tflite format for inference purposes.

3.3 Decision Tree

The complete decision tree algorithm is shown in Fig. 3. To demonstrate, the process of testing one pattern begins by placing the pattern and capturing it with a camera. Simultaneously, the YOLOv5n deep learning algorithm is run to detect any anomalies in the pattern, frame by frame. If an anomaly is detected, the confidence score is checked against a set confidence threshold of 0.90. If the detection confidence score is greater than 0.90, it is added to the anomaly list. If the score is less than 0.90, it is added to the normal list. Each pattern is evaluated over 100 frames. If the number of elements in the anomaly list is greater than 90 after 100 frames, the pattern is deemed to be an anomaly, otherwise it is classified a norm.

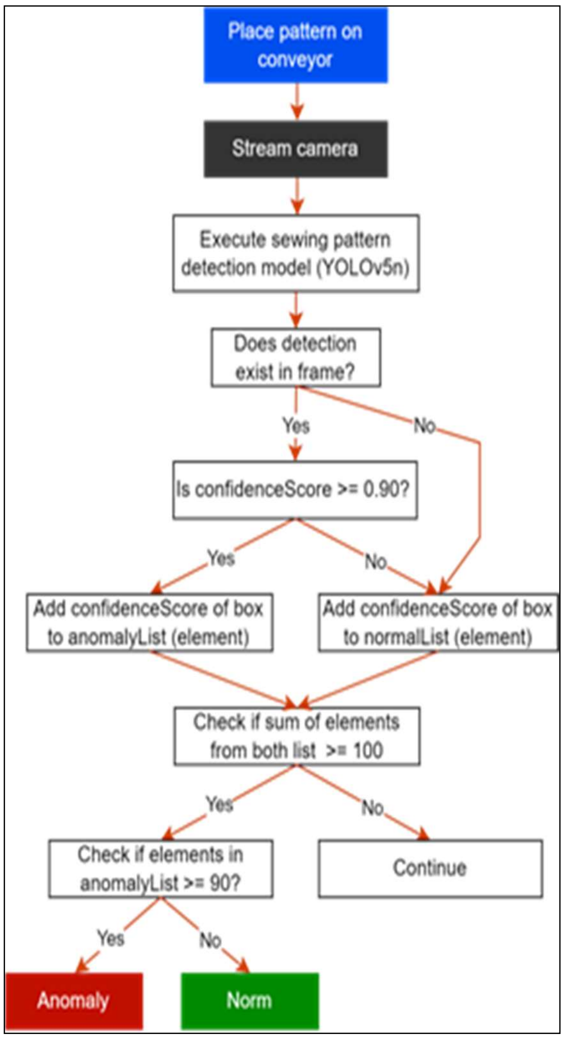
Place pattern on conveyor
Stream camera
Execute sewing pattern detection model (YOLOv5n)
Does detection exist in frame?
Yes
No
Is confidenceScore >= 0.90?
Yes
No
Add confidenceScore of box to anomalyList (element)
Add confidenceScore of box to normalList (element)
Check if sum of elements from both list >= 100
Yes
No
Check if elements in anomalyList >= 90?
Continue
Yes
No
Anomaly
Norm

Fig. 3. Anomaly detection decision tree

3.4 Deployment Pipeline

An AI application can be thought of as a sequence of tasks for processing data. Initially, the data must be loaded and prepared to meet the requirements of the model. In the case of detection, this includes loading and adjusting the size of a JPEG image. The next steps are making predictions and finalizing the output. The final step transforms the raw output from the model into a usable format, which can include thresholding, non−maximum sup− pression, and coordinate transformation. Since this pipeline is run for each prediction, it is important for all the steps to be as efficient as possible. Often, the focus is mainly on optimizing the model, while overlooking other parts of the process. This section introduces a streamlined and optimized deployment solution for the detection model, which also allows for the use of widely−used high−level frameworks.

The software architecture shown in Fig. 4 presents a basic layered approach for a compact deep vision deployment. The section about the TPU was previously discussed. OpenCV [22] is commonly used for loading and adjusting images. It uses shared libraries to perform these tasks. Using an optimized image loader such as libjpeg−turbo [23] can speed up the entire pipeline. The same is true for Numpy [24], which handles mathematical tensor operations on the CPU. A specialized math library like OpenBLAS [25], which utilizes Single Instruction Multiple Data (SIMD), can perform vector operations more quickly and efficiently. This type of software stack is just as fast as a solution written in a compiled language, but it is much more flexible. Additionally, it would be practical to package the application in a lightweight container for easy deployment using virtualization technologies.

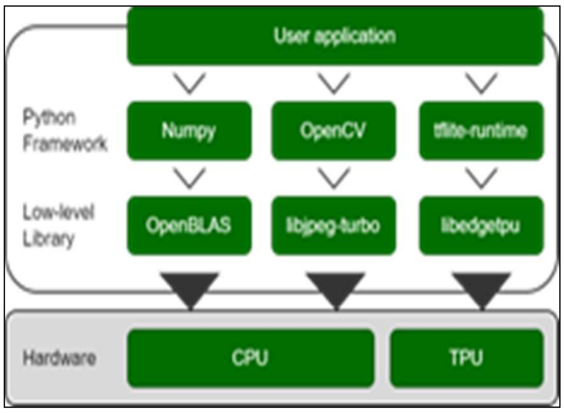

Fig. 4. Micro software stack for fast and lightweight edge deployment

IV. Experiments

4.1 Dataset

The dataset was collected at Hongik University AI Laboratory. Initially, there were 1200 images in the raw dataset labeling by bounding box, then after data augmentation including flip, rotate, adding noise, mosaic [8], color modifications, brightness, and contrast, the final dataset for training including 20000 images, resulting a sufficient dataset for this task.

4.2 Hardware

The models were trained on a computer with the configuration: CPU AMD Ryzen Threadripper 2950X @ 4.40 GHz (16 threads x 32 core), 64GB DDR4 2666MHZ for RAM, GPU NVIDIA GeForce GTX 2080 Ti 12GB x 2, Linux Ubuntu 20.04.4 LTS and Python 3.9.12. In terms of the embedded board, Orange Pi 3 5(CPU:

Cortex−A53, RAM: 2GB DDR3, GPU: Mali−T720 MP2 @ 600Mhz) was used for the experimental process, with the assistance of the accelerator Google Coral Edge TPU 6. Fig. 5 and Fig. 6 show Orange Pi 3 mainboard and TPU, respectively.

Fig. 5. Orange Pi 3 mainboard

Fig. 6. Accelerator Google Coral Edge TPU

4.3 Evaluation Metrics

In our experiments, F1score, False Positive Rate (FPR), False Negative Rate (FNR), and Accuracy have been applied to evaluate the models. Before reading these metrics' formulas, it is important to understand the following terms: True Positives (TP) denote the number of positive samples (Anomalies) that have been accurately identified by the model; True Negatives (TN) represent the number of negative samples (Norms) correctly classified; False Positives (FP) is the number of positive samples (Anomalies) that were wrongly classified; False Negatives (FN) means the number of negative samples (Norms) that were incorrectly identified.

$$F1 = \frac{2TP}{2TP + FN = FN}$$

$$FPR = \frac{FN}{TP + FN}$$

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

$$FNR = \frac{FP}{TP + FP}$$

4.4 Experimental Results

The model has been evaluated with 20 tested samples (each sample has 1 pattern, the ratio between anomaly and norm is 50/50 or there are 10 anomalies and 10 norms). Fig. 7 depicts the setup for testing at the sewing machine company where the testing phase was conducted.

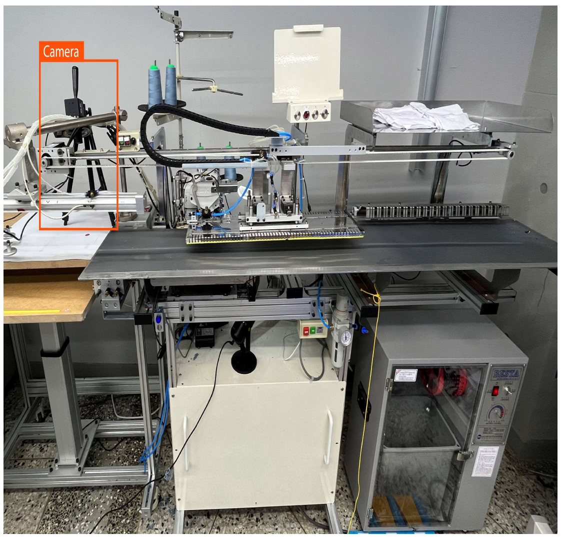

Fig. 7. Setup system for testing

The confusion matrix for the performance of each model is illustrated in Fig. 8. YOLOv5n model has outperformed the other techniques by achieving the perfect evaluation metrics. The fastest execution speed was 0.070 ms with the smallest model size file 2.43 MB. Fig. 9 displays some testing patterns for the anomalies at the testing venue with YOLOv5n based on the proposed decision tree.

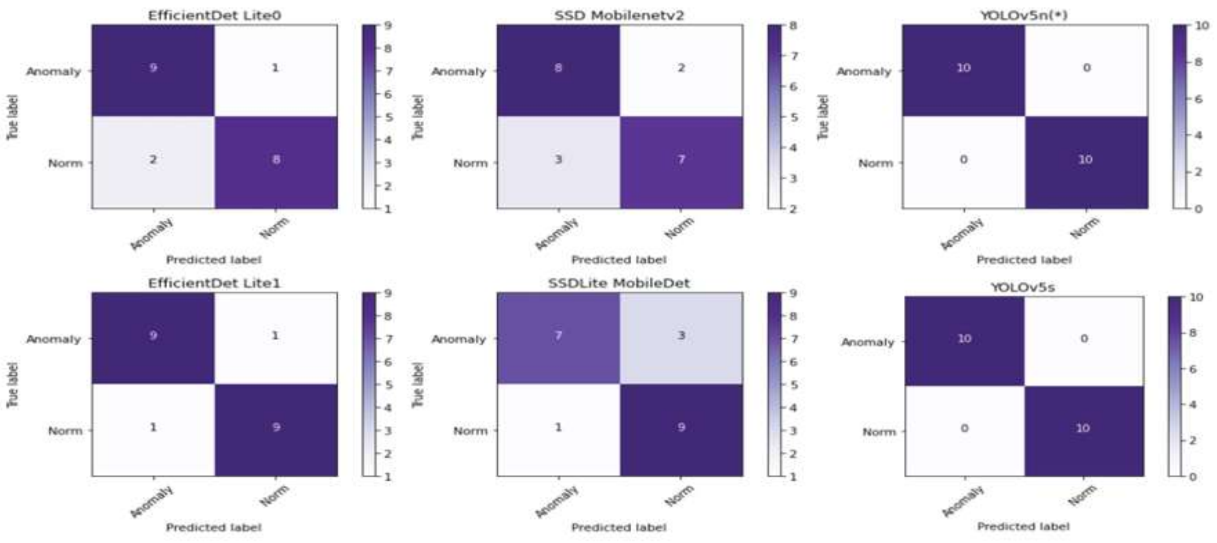

Fig. 8. Confusion matrix of conducted models performed on 20 testing patterns

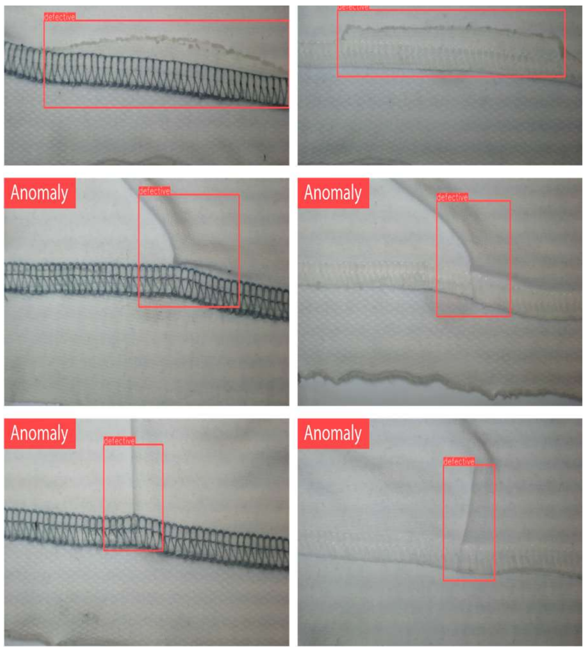

Fig. 9. Results of anomaly sewing detection model (YOLOv5n) on test patterns

V. Conclusion

In this study, the YOLOv5n deep learning algorithm with the proposed decision tree has been successfully applied to the textile industry for locating anomalies and classifying sewing patterns. The algorithm was successfully quantized and run on an Orange Pi 3 embedded mainboard with ARM architecture and a Google Coral Edge TPU accelerator. A decision tree was proposed to make the final conclusion of anomalies, providing a sufficient scenario for anomaly sewing pattern detection. The results of comparison with other techniques indicated that the proposed algorithm achieved robustness, speed, and smaller file size compared to the others. Future research will focus on finding more optimized methods to further enhance the speed and reduce the size of the model while maintaining its performance.

REFERENCES

[1] M. R. Pedersen, L. Nalpantidis, R. S. Andersen, C. Schou, S. Bøgh, V. Kru¨ger, and O. Madsen, "Robot skills for manufacturing: From concept to industrial deployment," *Robotics and Computer-Integrated Manufacturing*, vol. 37, pp. 282–291, 2016.

[2] G. Jocher, "YOLOv5 by Ultralytics," 52020. Available at https://github.com/ultralytics/yolov5.(accessed Dec., 15, 2023).

[3] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, et al., "Deep learning for generic object detection," *A Survey [J].,* 2018.
[4] C. B. Murthy, M. F. Hashmi, N. D. Bokde, and Z. W. Geem, "Investigations of object detection in im− ages/videos using various deep learning techniques and embedded platforms a comprehensive review," *Applied sciences*, vol. 10, no. 9, pp. 3280, 2020.
[5] X. Wu, D. Sahoo, and S. C. Hoi, "Recent advances in deep learning for object detection," *Neurocomputing*, vol. 396, pp. 39–64, 2020.
[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.
[7] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," *in Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.
[8] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," arXiv preprint arXiv:2004.10934, 2020.
[9] Y. Xiong, H. Liu, S. Gupta, B. Akin, G. Bender, Y. Wang, P. J. Kindermans, M. Tan, V. Singh, and B. Chen, "Mobiledets: Searching for object detection architectures for mobile accelerators," *in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3825–3834, 2021.
[10] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018.
[11] A. Ghosh, S. A. Al Mahmud, T. I. R. Uday, and D. M. Farid, "Assistive technology for visually impaired using tensor flow object detection in raspberry pi and coral usb accelerator," *in 2020 IEEE Region 10 Symposium (TEN− SYMP),* pp. 186–189, 2020.
[12] M. Verucchi, G. Brilli, D. Sapienza, M. Verasani, M. Arena, F. Gatti, A. Capotondi, R. Cavicchioli, M. Bertogna, and M. Solieri, "A systematic assessment of embedded neural networks for object detection," *in 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA),* vol. 1, pp. 937– 944, IEEE, 2020.
[13] D. N. N. Tran, H. H. Nguyen, L. H. Pham, and J. W. Jeon, "Object detection with deep learning on drive px2," *in 2020 IEEE International Conference on Consumer Electronics− Asia (ICCE−Asia), pp. 1–4, IEEE, 2020.*
[14] H. H. Nguyen, D. N. N. Tran, and J. W. Jeon, "Towards real−time vehicle detection on edge devices with nvidia jetson tx2," *in 2020 IEEE International Conference on Consumer Electronics − Asia (ICCE−Asia),* pp. 1–4, 2020.
[15] A. Goncalves, P. Ray, B. Soper, D. Widemann, M. Nygard, J. F. Nygard, and A. P. Sales, "Bayesian multitask learning regression for heterogeneous patient cohorts," *Journal of Biomedical Informatics,* vol. 100, pp. 100059, 2019.
[16] X. Zhang, J. Zhou, W. Sun, and S. K. Jha, "A lightweight cnn based on transfer learning for covid−19 diagnosis," *Computers, Materials and Continua*, pp. 1123–1137, 2022.
[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," *in European conference on computer vision,* pp. 21–37, Springer, 2016.
[18] H.K. Jung and G.S. Choi, "Improved yolov5: Efficient object detection using drone images under various conditions," *Applied Sciences*, vol. 12, no. 14, p. 7255, 2022.

[19] M. A. Rahaman, M. M. Ali, K. Ahmed, F. M. Bui, and S. H. Mahmud, "Performance analysis between yolov5s and yolov5m model to detect and count blood cells: deep learning approach," *in Proceedings of the 2nd International Conference on Computing Advancements*, pp. 316–322, 2022.
[20] B. Jiang, R. Luo, J. Mao, T. Xiao, and Y. Jiang, "Acquisition of localization confidence for accurate object detection," *in Proceedings of the European conference on computer vision (ECCV),* pp. 784–799, 2018.
[21] "Post training quantization," https://www.tensorflow.org/lite/performance/post training quantization.(accessed Dec., 15, 2023).

[22] G. Bradski, "The opencv library.," *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, vol. 25, no. 11, pp. 120–123, 2000.
[23] "Libjpeg-turbo github," 07 2015.
[24] C. R. Harris, K. J. Millman, S. J. Van Der Walt, R. Gom- mers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, et al., "Array programming with numpy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.
[25] Q. Wang, X. Zhang, Y. Zhang, and Q. Yi, "Augem: automatically generate high performance dense linear algebra kernels on x86 cpus," *in SC'13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pp. 1–12, IEEE, 2013.

Authors

Nguyen Quoc Toan

He received B.S degree from Can Tho University, Vietnam and M.S degree from Hongik University, Korea.

Seongwon Cho

He received his B.S. degree from Seoul National University, Korea in 1982. He also received his MS and Ph.D degrees from Purdue University, West Lafayette, Indiana, USA in 1987 and 1992, respectively. He has been a professor of Hongik University, Seoul, Korea.