Available online at www.ijournalse.org

Emerging Science Journal

(ISSN: 2610-9182)

Vol. 7, No. 1, February, 2023

A Framework to Create a Deep Learning Detector from a Small Dataset: A Case of Parawood Pith Estimation

Wattanapong Kurdthongmee [1*], Korakot Suwannarat [1], Jeremy Kiplagat [2]

[1] *School of Engineering and Technology, Walailak University 222 Thaiburi, Thasala, Nakhon Si Thammarat 80160, Thailand.*

[2] *International College, Walailak University 222 Thaiburi, Thasala, Nakhon Si Thammarat 80160, Thailand.*

Abstract

A deep learning-based object detector has been successfully applied to all application areas. It has high immunity to variations in illumination and deviations among objects. One weakness of the detector is that it requires a huge, undefinable dataset for training the detector to avoid overtraining and make it deployable. This research proposes a framework to create a deep learning-based object detector with a limited-sized dataset. The framework is based on training the detector with the regions surrounding an object that typically contain various features over a more extensive area than the object itself. Our proposed algorithm further post-processes the detection results to identify the object. The framework is applied to the problem of wood pith approximation. The YOLO v3 framework was employed to create the detector with all default hyperparameters based on the transfer learning approach. A wood pith dataset with only 150 images is used to create the detector with a ratio between training to testing of 90:10. Several experiments were performed to compare the detection results from different approaches to preparing the regions surrounding a pith, i.e., all regions, only close neighbors, and only diagonal neighbors around a pith. The best experiment result shows that the framework outperforms the typical approach to create the detector with approximately twice the detection precision at a relative average error.

Keywords:
Object Detection;
Wood Pith Detection;
YOLO Object Detection;
Small Dataset Training.

Article History:

Received:	17	April	2022
Revised:	10	October	2022
Accepted:	19	October	2022
Available online:	07	November	2022

1- Introduction

Deep learning-based object detection has proven to be superior to the standard approaches, which are based on the algorithms of Viola-Jones [1], scale-invariant Feature Transform (SIFT) [2], or Histogram of Oriented Gradients (HOG) [3]. In terms of speed and accuracy, two popular DL object detection frameworks are the Single Shot MultiBox Detector (SSD) [4], and You Only Look Once (YOLO) [5]. Both have been applied to many areas, from tumor detection in the medical field to defect detection in the manufacturing and agricultural fields. The flourishing of DL-based object detections results from the fact that there exist: (1) enormous datasets for training a detector; (2) high-performance computer platforms, i.e., with GPUs (Graphic Processing Units), for training a detector in addition to acceptable performance and affordable price edge processing platforms for deployment; (3) open-source infrastructures for training and deploying a detector; and (4) common competition platforms for researchers. Concerning the domain of object detection in an image, the standard and publicly available datasets (with their sizes in parenthesis) are COCO (2,500,000), ImageNet (14,197,122), and CIFAR (60,000). These datasets are large. The large dataset size is believed to provide various features for each object within the dataset inherently. The detector can extract as many features as possible and leverage its effectiveness for deployment. Unfortunately, the definition of "huge" is not clearly defined. It

* **CONTACT**: kurdthongmee.wattanapong@gmail.com

DOI: http://dx.doi.org/10.28991/ESJ-2023-07-01-017

is difficult for researchers to prepare a suitable dataset for their specific application. Due to the lack of such information, it is unavoidable that researchers rely on a trial-and-error approach to train their detector by varying the hyperparameters until the detector reaches an acceptable error. It consumes considerable time, even on a computer platform with GPU acceleration.

Small datasets for training DL-based classifiers and object detectors have gained the attention of many researchers in the field. It has been chiefly applied to medical problems for classification because creating a massive dataset in this application domain is challenging [6]. A comparison between shallow and deep architecture classifiers on small datasets was reported by Pasupa et al. [7]. Wang and Perez [8] proposed a novel form of data augmentation called "neural augmentation" to improve the performance of a DL classifier trained with small datasets. The augmentation allows the classifier to learn augmentations that best improve it. Due to the limited size of the dental dataset, Yang et al. [9] proposed using an automatic method with medical knowledge to crop out the ROIs (regions of interest) within the images for clinical evaluation. Then, pairs of ROIs are used to train the DL classifier. Bae et al. [10] reported that the accuracy of FusionNet with data augmentation using Perlin noise was significantly higher than that with conventional data augmentation for the DL classifier trained with small data samples of high-resolution computed tomography images. Furthermore, many techniques were employed to increase the prediction accuracy [11], i.e., overlap pooling, flipped-image augmentation, and dropout. It is claimed that the classification performance is comparable to that of expert dentists and radiologists. Later, the approach to pre-train the DL classifier with another dataset before training with the real dataset of CT scans was presented by Ausawalaithong et al. [12]. It is claimed that the approach solves the problem of a small dataset and produces a promising result for further investigation.

On the material defect problem, Feng et al. [13] relied on using a stacked auto-encoder to pre-train a deep neural network with a small dataset to optimize initial weights. Barz & Denzler [14] claimed that the integration of the cosine loss function into the DL classifier provides substantially better performance than cross-entropy on datasets with only a handful of samples per class. They also demonstrated that integrating prior knowledge in the form of class hierarchies is straightforward with the cosine loss and further improves classification performance. Rajpurkar et al. [15] reported that pretraining the DL classifier for CT (Computed Tomography) exams significantly improves the classifier's performance when the dataset size is limited. Related to the DL architecture, Brigato & Iocchi [16] showed that model complexity is a critical factor when only a few samples per class are available. Huang et al. [17] presented a novel DL framework for fabric defect segmentation based on two networks: the segmentation and decision networks. They reported that the framework demands only 50 defect samples to get accurate segmentation results and can achieve the requirement of real-time detection with a speed of 25 frames per second (FPS). Within the same application domain, Jing et al. [18] proposed a highly efficient convolutional neural network, Mobile-Unet, which achieves end-to-end defect segmentation. The median frequency balancing loss function is integrated within the network to overcome the sample imbalance. Depth-wise separable convolution is also introduced. It dramatically reduces the network's complexity, cost, and model size. Finally, the softmax layer is used to generate the segmentation mask. On the small dataset of the weld X-ray image, Ajmi et al. [19] reported that a 3% increase in accuracy of detection is achieved by replacing channels B (blue) and G (green) of the images with the Canny edge map and a binary image provided by an adaptive Gaussian threshold. Recently, Ju et al. [20] demonstrated the improvement of the original CNN model for jujube defect detection by embedding the SE module and replacing the softmax loss function with the triplet and center loss functions. The experimental results show that the SE-ResNet50-CL model optimizes the fine-grained classification problem of jujube defect recognition, and the test accuracy reaches 94.15%. The model claims to have good stability and high recognition accuracy in complex environments, although it was trained from a limited dataset of jujube images.

Low-complexity DL architectures have been proven to perform comparably well or better than state-of-the-art ones for problems with small datasets and without data augmentation. From the reviews, it can be summarized that most publications in the field of DL classifiers or detectors with small datasets have attempted to: (1) modify the internal DL architecture; (2) change the loss functions; (3) create an appropriate pre-train model; and (4) rely on using different image augmentations. Primarily, it has been targeted for the classification problem. It is in contrast to our proposed framework in this paper, which targets an object detection problem.

Indeed, a variety of features of objects within the dataset relies heavily on the size of the dataset. Before training the detector, they are annotating all objects within an image by drawing bounding boxes around the objects and assigning them with a class label. In this way, bounding boxes provide the specific features of their enclosed objects. For some application domains, i.e., a defect within manufacturing parts, a tumor within the brain, or a pith within a cross-section of timber, apart from the difficulty of preparing a large dataset, the objects themself are relatively small in size. This prevents the applicability of the DL detectors to these domains. In this research, we hypothesize that a DL detector could successfully be trained from the dataset of regions around an object within the dataset. Compared to the detector trained from the ordinary routine, its detection accuracy is expected to be leveraged based on the limited dataset size. This comes from the fact that the variety of features within the regions around an object is relatively more significant than the ones within the object. By somehow embedding the geometric information to these regions around an object, once they are detected, by a detector, they can be processed by an algorithm to locate the object. In this paper, the hypothesis is tested on the problem of approximating pith location within a cross-sectional image of parawood.

2- Materials and Methods

2-1- Dataset

This research employs the parawood cross-sectional dataset to confirm that the framework works effectively. All images were captured by the built-in camera of the iPhone 11 and saved in a high-resolution HEIC format. Each image contains only a single cross-section with a single pith with some deviations from the center of the image. All images were resized from their HEIC format to have a resolution of 416×312 to conform with the requirement of an input image format for training the YOLO v3 detector. We randomly selected only 150 images from the whole dataset of 1,000 images for training the detector. These images were randomly assigned to the training and testing datasets with a ratio of 90:10. The rest 850 images were explicitly reserved to be used as a validation dataset. Each image of the training and testing datasets is manually annotated by drawing a bounding box around the pith and the wood cross-section using the LabelImg tool, freely available on the Internet.

Pith is the region within the image covering ten annular rings. An area surrounding a pith is defined as the area out of the pith region but within the bounding box around the wood cross-section. Figure 1 illustrates two sample images of the cross-section of the wood with the outer annotated bounding box covering the cross-section of the wood and the inner one surrounding its pith. It is evident that the object, the wood piths - to be specific, are relatively small and only enclose a tiny region within the image. From this annotation information, the pith, and the cross-section of wood, a Python script was coded by our team member to create a set of non-overlapping bounding boxes whose classes are: *UL*, *UM*, *UR*, *ML*, *MR*, *LL*, *LM*, and *LR*. From these classes, the left alphabets indicate the vertical locations of the bounding box relative to a pith; i.e., *U*, *M*, and *L* are for upper, middle, and lower, respectively. Similarly, the right alphabets indicate the horizontal locations of the bounding box relative to a pith; i.e., *L*, *M*, and *R* are for left, middle, and right, respectively. For example, the bounding box whose class is *UM* is in the upper central region of the pith.

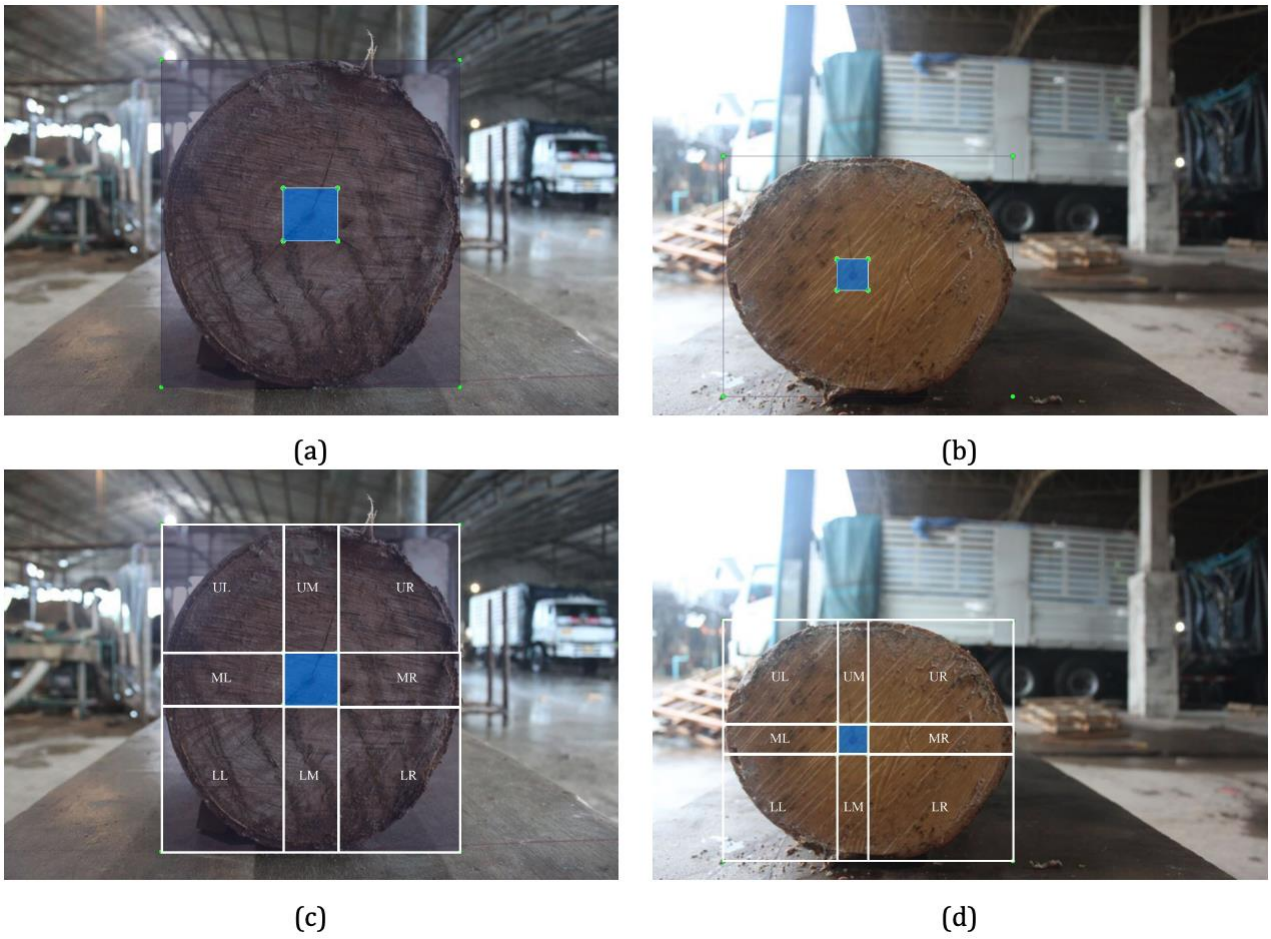

Figure 1. Sample wood cross-sectional images with annotated bounding boxes covering the wood cross-section and its pith in (a) and (b) and with automatic generation of non-overlapping bounding boxes surrounding piths whose classes are: *UL*, *UM*, *UR*, *ML*, *MR*, *LL*, *LM*, and *LR* in (c) and (d).

2-2- Method

2-2-1- Training the Detector

The datasets previously prepared were used to train the detectors to recognize the features within the regions around the piths. Four different detectors with the following details have been experimented with:

- Diagonal neighbors: the dataset contains only the bounding boxes with *UL*, *UR*, *LL*, and *LR* classes,
- Close neighbors: the dataset contains only the bounding boxes with *UM*, *ML*, *MR*, and *LM* classes,
- All neighbors with class annotations: the dataset includes all classes of the bounding boxes,
- All neighbors without class annotation: the dataset contains all bounding boxes assigned with a single class of *NP*.

YOLO v3 with 106 layers was used as a framework of choice. The darknet53.conv.74 was used for training the detectors on a transfer learning basis. The default open source darknet software was used, and no explicit approach to increase the training dataset size through image augmentation was performed. However, we know that the darknet has a built-in facility to perform image augmentation. The darknet facility was enabled in all experiments. For the training configurations, we based on using all the default settings for training the detector with the exceptions of the number of classes (n), the sizes of the filter at the layer preceding all YOLO layers (defined by $(n + 5) \times 3$), and the number of training iterations (defined by $2000 \times n$) which were modified for specific experiments. The detectors were trained on a Google Colab platform with GPU acceleration. The average training time was around 12 hours per experiment. We monitored the loss at each iteration during training to check if the detector was trained to converge to an acceptable loss. Apart from training these 4 detectors to compare the benefit of our framework, the ordinary routine to train YOLO v3 was used to create a single class detector.

2-2-2- Identifying a Pith

In contrast to an ordinary detector that produces the bounding box around a pith, if it is detectable, after processing the image, our detectors were expected to produce a set of bounding boxes around the region of a pith. It required us to develop a post-processing algorithm to identify the pith region using the information from the detectors' output. At this point, let us define the horizontal extension of the bounding box of *B*, or *B'*, to be the bounding box with equal height to *B*, but its width is extended to equal the width of an image. Additionally, the vertical extension of the bounding box of *B* is the bounding box with equal width to *B*, but its height is extended to be equal to the height of an image. Also, we define functions $MIN(a,b)$ and $MAX(a,b)$ that produce as their output the minimum and maximum values between a and b, respectively. Suppose the coordinate of a bounding box X is defined by (LX, TX, RX, BX). The following rules can be drawn to make use of a set of detected bounding boxes around the pith to locate the region of a pith:

- For a pair of *UM* and *LM* bounding boxes, the region of a pith is approximated by: $P^* = (MIN(L_{UM},L_{LM}),B_{UM},MAX(R_{UM},R_{LM}),T_{LM})$ (see Figure 2-a)),
- For a pair of *ML* and *MR* bounding boxes, the region of a pith is approximated by: $P^* = (R_{ML},MIN(T_{ML},T_{MR}),L_{MR},MAX(B_{ML},B_{MR}))$ (see Figure 2-b)),
- For a pair of *UL* and *LR* bounding boxes, the region of a pith is approximated by: $P^* = (R_{UL}, B_{UL}, L_{LR}, T_{LR})$ (see Figure 2-c)),
- For a pair of *UR* and *LL* bounding boxes, the region of a pith is approximated by: $P^* = (R_{LL}, B_{UR}, L_{UR}, T_{LL})$ (see Figure 2-d)),
- For a pair of *UL* and *MR* bounding boxes, one possible bounding box is: $P^* = (R_{UL}, B_{UL}, L_{MR}, B_{MR})$ (see Figure 2-e)),
- For a pair of *UL* and *LM* bounding boxes, the region of a pith is approximated by: $P^* = (R_{UL}, B_{UL}, R_{LM}, T_{LM})$ (see Figure 2-f)),
- For a pair of *UR* and *LM* bounding boxes, the region of a pith is approximated by: $P^* = (L_{LM}, T_{UR}, R_{LM}, T_{LM})$ (see Figure 2-g)),
- For a pair of *UR* and *ML* bounding boxes, the region of a pith is approximated by: $P^* = (R_{ML}, T_{ML}, L_{UR}, B_{ML})$ (see Figure 2-h)),
- If a pair of close neighbour bounding boxes is either *UM* - *ML* or *UM* - *MR* or *LM* - *ML* or *LM* - *MR*, the region of a pith is approximated by the intersection region between the appropriate horizontal and vertical extensions of these bounding boxes. The examples for the cases of *UM* - *ML* and *LM* - *MR* are illustrated in Figure 2-i and 2-j,
- Otherwise, no region of a pith can be located.

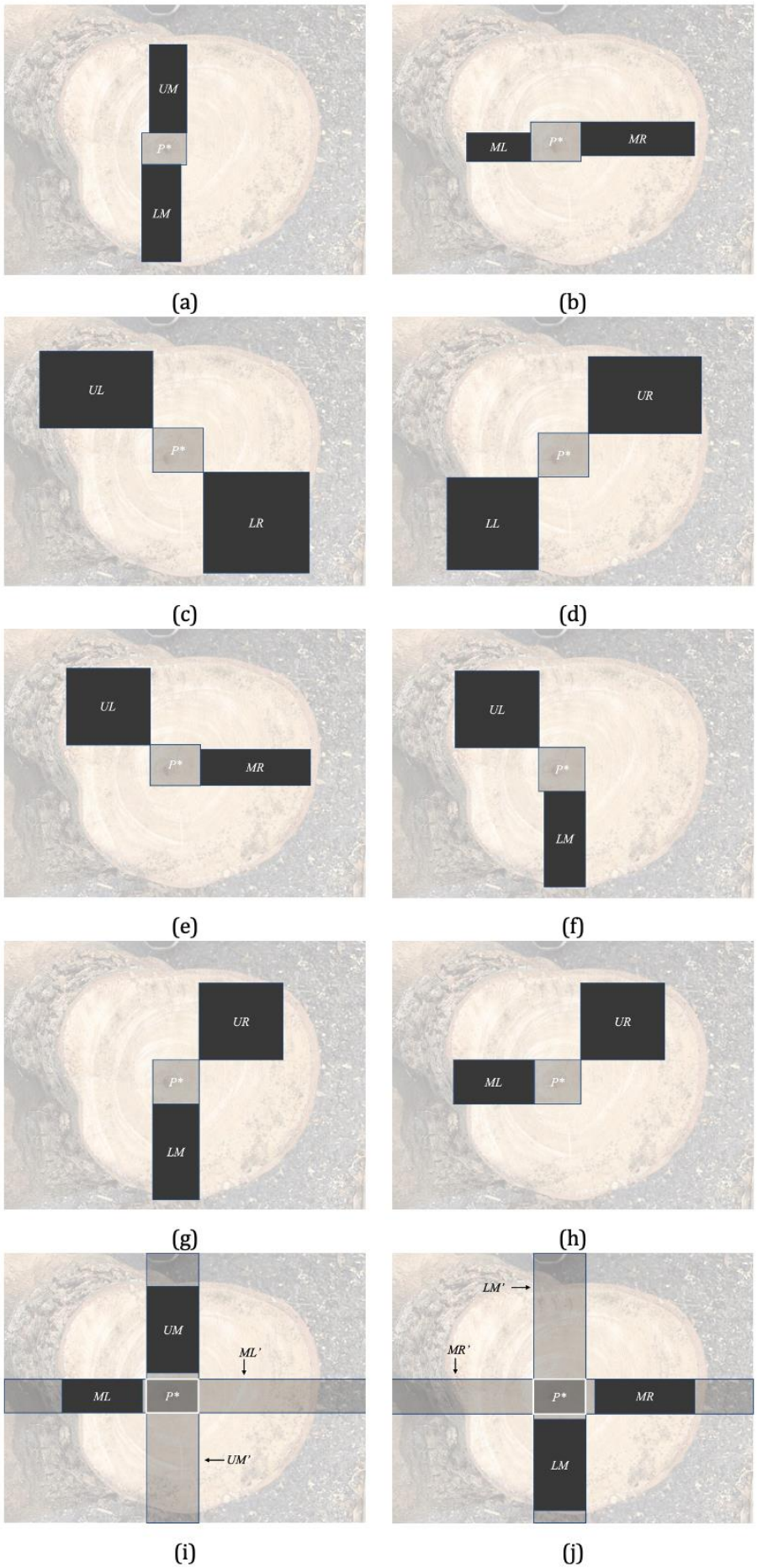

Figure 2. Different cases to approximate the region of a pith based on using the detected bounding boxes around a pith

In practice, a set of detected bounding boxes around the region of a pith gives rise to multiple $P*$. Our algorithm finds the intersection between all members of $P*$. The result is finally used to represent a single region of a pith. The detector trained from all neighbors with a single class can only produce a set of bounding boxes around a pith with *NP*-class no matter where they are. Let us name a set of bounding boxes around a pith with *NP*-class as *N*. They are illustrated in Figure 3-a. These bounding boxes require further processing to come up with the approximate region of a pith, i.e., use their relative position to map any bounding box class, which are: *UL*, *UM*, *UR*, *ML*, *MR*, *LL*, *LM*, and *LR*. The following are the details of the algorithm designed to serve this purpose:

- Create a bounding box *M* that covers all members of *N* (see Figure 3-b). It is expected that if all classes within *N* are detected, *M* will cover the cross-section of wood. This is accomplished by visiting all members of *N* to find the minimum *(L, T)* and the maximum *(R, B)* coordinates. Therefore, these two coordinates are assigned to be the coordinates of *M*. *M* is considered valid if one of the ratios between its width and height is between 0.75 and 1.25. This means that it approximately covers the cross-section of wood; i.e. the detector could detect at least a pair of non-in-line diagonal neighbors such as *UL* and *LR*. It is noted that the provided correct ratio comes from our analysis of the possible ratios of bounding boxes around the cross-section of wood within the training dataset.
- Create a set of 8 ideal bounding boxes *I* inside *M* and assign their coordinates, and label them as *UL*, *UM*, *UR*, *ML*, *MR*, *LL*, *LM*, *and LR*. These ideal bounding boxes attempt to imitate the bounding boxes around a pith with respect to *M*. The scales of all middle bounding boxes of I are defined as half of the other bounding boxes. Figure 3-c) illustrates these ideal bounding boxes represented by a set of green rectangles. *I* provides a way to classify the relative position to a pith of the member of *M*.
- Visit each member *m* of *M*; i.e. M_m. Find the intersection between Mm and all members of *I*. The class of M_m is the class *i* of *I*; i.e. I_i, whose intersection between M_m and I_i is maximum. The result from this step is illustrated in Figure 3-d.

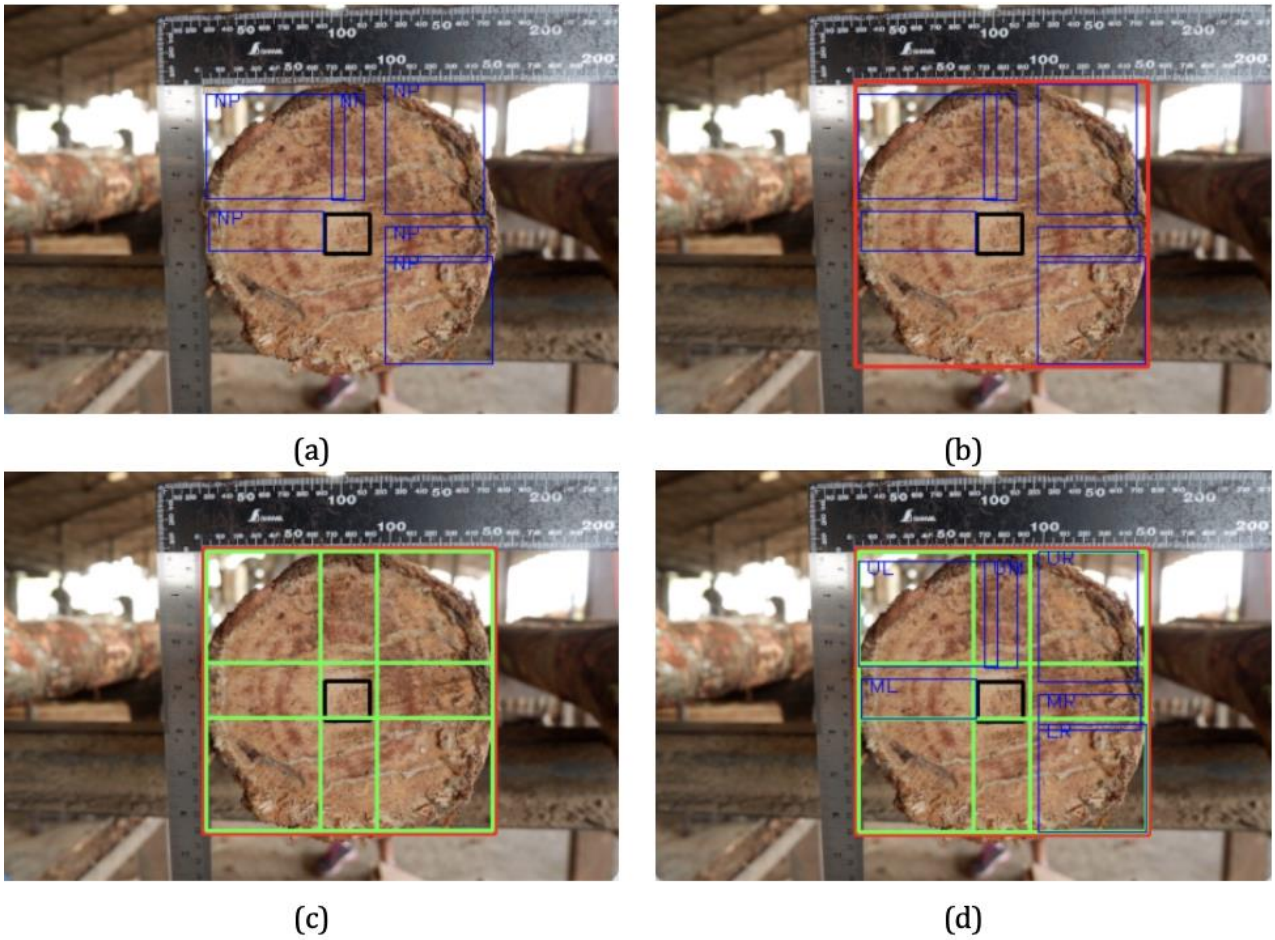

Figure 3. Sample step-by-step demonstration of assigning class to the detected bounding boxes of the single-class detector

After all members of *N* are assigned with their exact class with respect to the pith, the algorithm presented earlier can be used to find the region of the pith. It is noted that both algorithms take a negligible amount of time during processing. This comes from the fact that the total size of detected bounding boxes around the pith region is relatively small.

2-2-3- Summary of the Research Methodology

To summarise the proposed research methodology, the flowcharts in Figure 4 illustrate all the processes that we based on using for training all the detectors (Figure 3-a) and inferencing them in order to find the best detector (Figure 3-b).

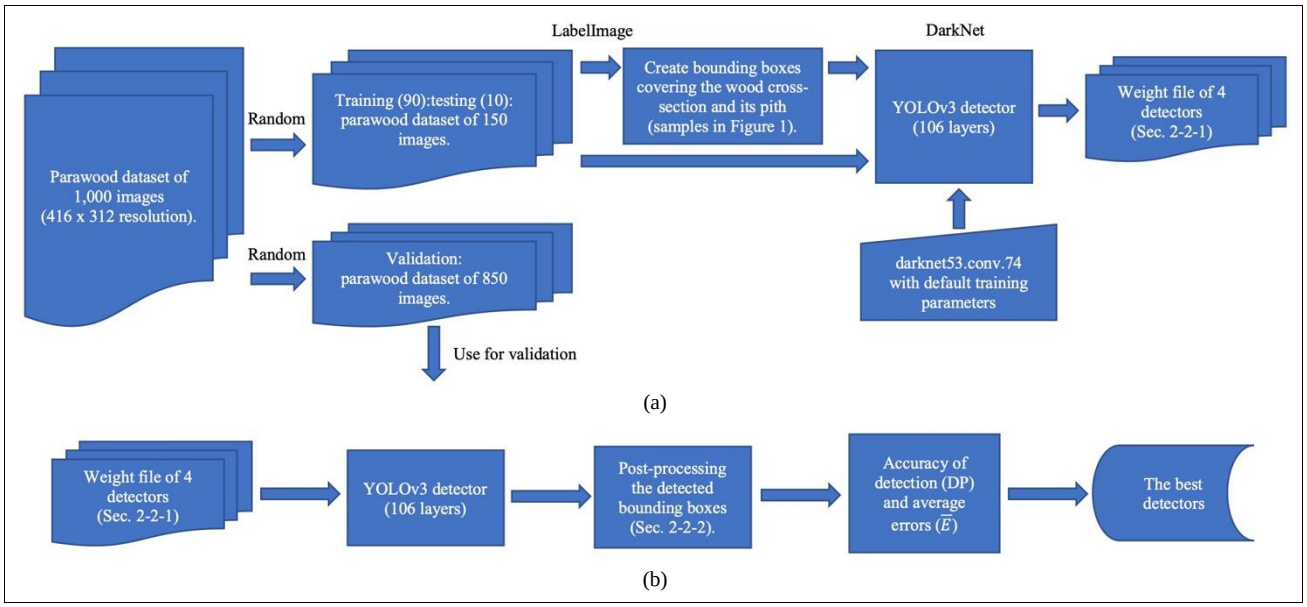

Figure 4. Summary of the proposed research methodology during: (a) training and (b) inferencing stages

3- Results

The captured losses of all detectors during the training stage are comparatively illustrated in Figure 5. All detectors seem to produce an acceptable loss after training them for 1,500 iterations. The weights from these detectors were then used to validate with the validation dataset, whose total size is 850 images. Some sample detection results randomly selected from validating the implementation of the proposed framework against the validation dataset on the training dataset basis are shown in Figures 6 to 9. Within the figures, the blue rectangles represent the detected set of bounding boxes around a pith, and the black and white rectangles surround the ground truths and the approximated regions of a pith processed by our proposed algorithm, respectively. In Figure 9, the ideal bounding boxes and the bounding boxes covering all the detected bounding boxes are represented by a set of green and red rectangles, respectively. For these figures, in most cases, the set of bounding boxes around the piths is fully detectable by the detectors. After processing the detected bounding boxes around the piths, the results fit quite well with the ground truths.

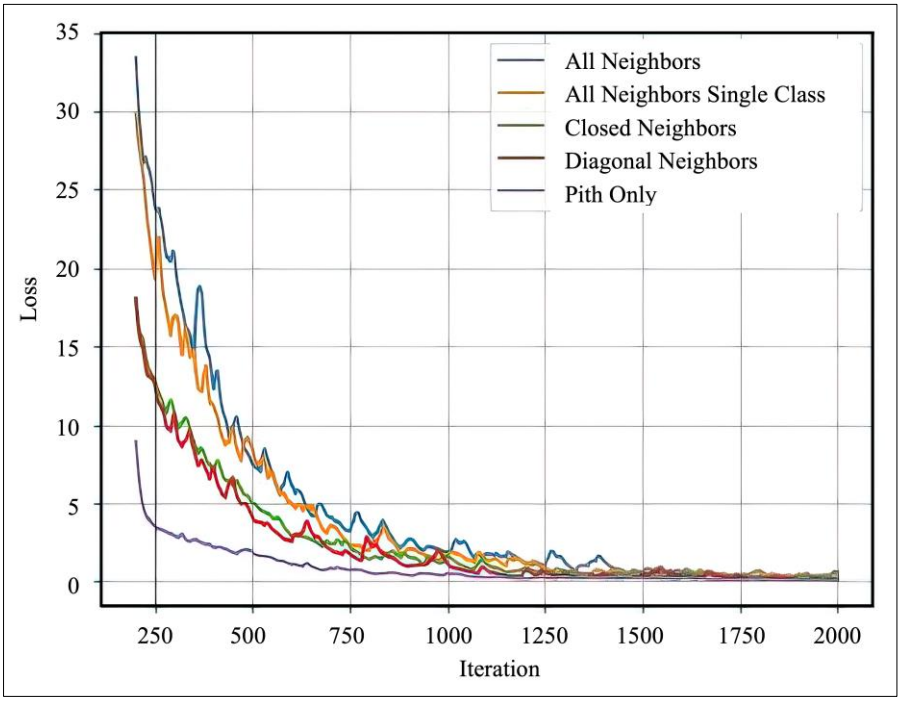

Figure 5. Plot of all detectors' losses versus iteration number during their training stage

Figure 6. Sample detection results from the detector trained from a dataset of diagonal neighbors around a pith

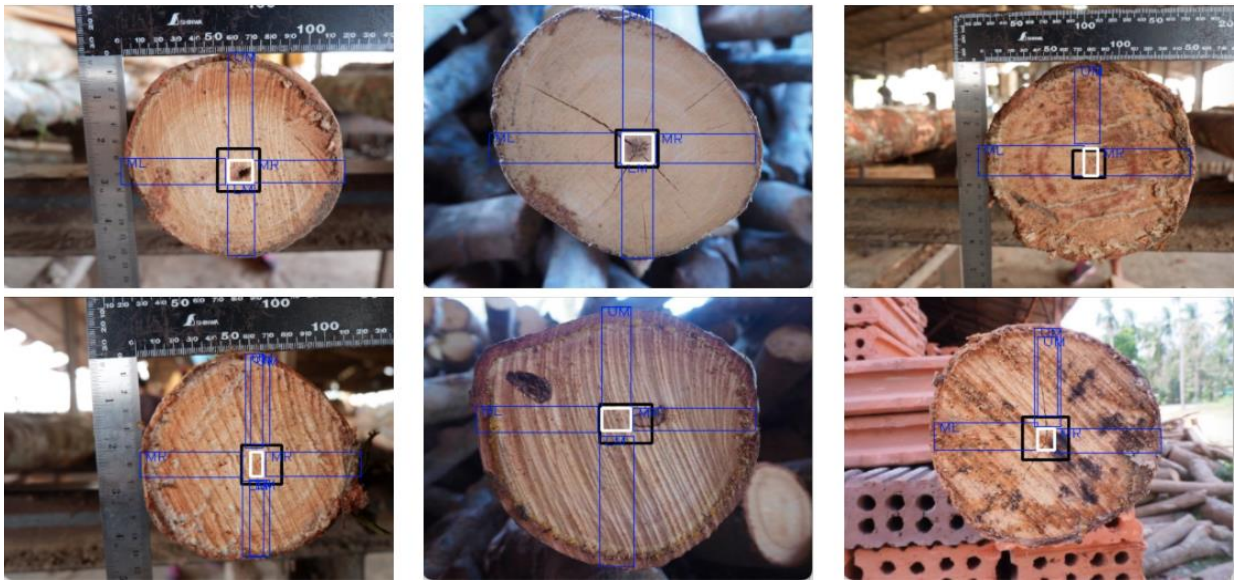

Figure 7. Sample detection results from the detector trained from a dataset of close neighbors around a pith

Figure 8. Sample detection results from the detector trained from a dataset of all neighbors around a pith

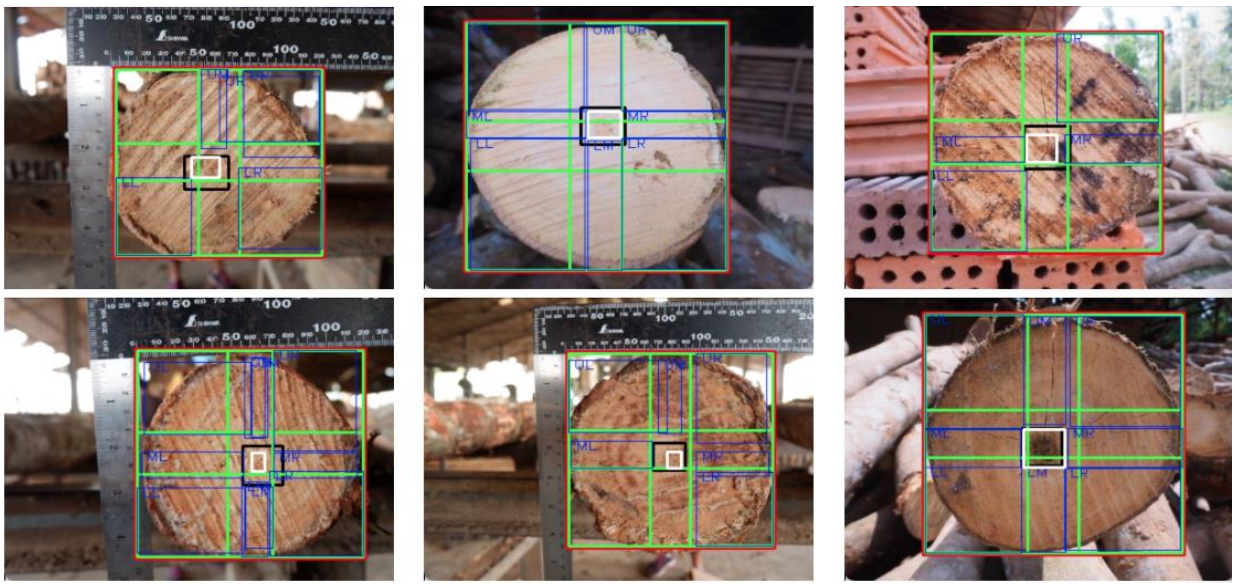

Figure 9. Sample detection results from the detector trained from a dataset of all neighbors around a pith with a single class

Table 1 comparatively shows the experiment results between the ordinary detector and all the detectors trained by our proposed framework. Apart from the detection precision, which is defined by Equation 1:

$$DP = \frac{TP}{TP+FP} \times 100\% \quad (1)$$

where TP and FP are the True Positive and False Positive, respectively, the average error between the approximated pith locations and the ground truths, $\bar{E}$, are also shown. $\bar{E}$ is defined by:

$$\bar{E} = \frac{\sum_{i=0}^{n-1}(|D_i - GT_i|)}{n} \quad (2)$$

where for an image i whose pith is detectable, D_i is the approximated pith location, GT_i is the ground truth and n is the total number of images whose pith are detectable. It is noted that the approximated pith location and the ground truth are defined to be at the center of their bounding box.

Table 1. Comparison of experiment results between the ordinary detector and the detectors trained by our proposed framework

Datasets	Average error (pixels)	Detection precision (%)
Pith only	22.20	43.60
Diagonal neighbours	21.54	57.35
Close neighbours	21.27	69.67
All neighbours	24.28	90.15
All neighbours single class	25.89	92.42

From Table 1, it is evident that the top detectors trained from our small datasets, the one trained from all neighbors with a single class, outperform the ordinary detector by roughly a factor of 2. All detectors produce a comparable average error. If the losses during the training stage are considered (see Figure 5), it can be seen that the ordinary detector seems to be overtrained. This means it behaves very well with the training dataset but is undeployable in practice. It can be explained that the small size of the dataset and the small region covered by a pith in each image within the dataset, make it difficult for the ordinary detector to extract the necessary features during the training stage. This, in turn, reduces its effectiveness for deployment. This contrasts with our proposed framework, which is not based on using the regions of the pith itself. The areas around a pith whose size is larger than the pith clearly provide a richer set of features. In addition, the regions around a pith are separated into different classes. This leverages the capability of the detector to detect any of these regions. This is in contrast to the ordinary detector, which only has two distinct choices, which are: pith detectable or undetectable.

When it comes to comparison results for the dataset within our proposed framework, it can be seen that the detector can be classified into two groups: the ones that are based on training with all neighbors and those that are based on training with some neighbors. The former group proved to produce considerably better results. For the latter group of detectors, it can be explained that only the regions around a pith can provide a richer set of features, while the total

number of regions is equal to the size of the training dataset, i.e., there are only 135 (90% of 150 images) regions in each class. This is in contrast to the detector trained from the all neighbors single class dataset, which consists of 135 × 8 regions, i.e., the total size of the training dataset multiplied by the total number of classes. Additionally, increasing the number of classes leverages the success of the detector.

4- Conclusion

A deep learning (DL) based object detector has been successfully applied to all application areas. It requires a vast, undefinable dataset for training the detector to make it deployable. In this paper, we present the framework for creating a DL-based object detector using a small-size dataset. The framework is based on using the regions around the object. It increases the size of the dataset and the regions of an image where the detector could extract necessary features compared to the standard approach based on using an object of limited size and covering areas. The framework has proven effective for the problem of pith detection on the wood cross-section. That is to say, the detector trained from a dataset of only 135 images in the default configuration of the YOLO v3 framework gives rise to a maximum detection precision of 92.42 percent, which is twice that of the ordinary detector. The average error is about 26 pixels, comparable to the ordinary detector. The dataset that produces the best detection accuracy has the size 8-fold of the original dataset, with more significant covering regions than a pith. In addition to providing the details of training dataset preparation, the post-processing algorithm that uses the detection result to approximate the region of an object is also described. The framework can be applied to a problem with similar characteristics to the parawood's pith.

5- Declarations

5-1- Author Contributions

Conceptualization, W.K. and K.S.; methodology, W.K. and K.S.; software, W.K. and K.S.; validation, W.K. and J.K.; writing—original draft preparation, W.K. and J.K.; writing—review and editing, W.K. and J.K. All authors have read and agreed to the published version of the manuscript.

5-2- Data Availability Statement

The data presented in this study are available on request from the corresponding author.

5-3- Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

5-4- Institutional Review Board Statement

Not applicable.

5-5- Informed Consent Statement

Not applicable.

5-6- Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this manuscript. In addition, the ethical issues, including plagiarism, informed consent, misconduct, data fabrication and/or falsification, double publication and/or submission, and redundancies have been completely observed by the authors.

6- References

[1] Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2001. doi:10.1109/cvpr.2001.990517.

[2] Lowe, D. G. (1999). Object recognition from local scale-invariant features. Proceedings of the Seventh IEEE International Conference on Computer Vision. doi:10.1109/iccv.1999.790410.

[3] Dalal, N., & Triggs, B. (n.d.). Histograms of Oriented Gradients for Human Detection. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). doi:10.1109/cvpr.2005.177.

[4] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. Lecture Notes in Computer Science, 21–37. doi:10.1007/978-3-319-46448-0_2.

[5] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). doi:10.1109/cvpr.2016.91.

[6] Hwang, J. J., Jung, Y. H., Cho, B. H., & Heo, M. S. (2019). An overview of deep learning in the field of dentistry. Imaging Science in Dentistry, 49(1), 1–7. doi:10.5624/isd.2019.49.1.1.

[7] Pasupa, K., & Sunhem, W. (2016). A comparison between shallow and deep architecture classifiers on small dataset. 2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE). doi:10.1109/iciteed.2016.7863293.

[8] Wang, J., & Perez, L. (2017). The effectiveness of data augmentation in image classification using deep learning. Convolutional Neural Networks Vis. Recognit, 11, 1-8.

[9] Yang, J., Xie, Y., Liu, L., Xia, B., Cao, Z., & Guo, C. (2018). Automated Dental Image Analysis by Deep Learning on Small Dataset. 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), Tokyo, Japan, 492-497. doi:10.1109/compsac.2018.00076.

[10] Bae, H. J., Kim, C. W., Kim, N., Park, B. H., Kim, N., Seo, J. B., & Lee, S. M. (2018). A Perlin Noise-Based Augmentation Strategy for Deep Learning with Small Data Samples of HRCT Images. Scientific Reports, 8(1), 1–7. doi:10.1038/s41598-018-36047-2.

[11] Oh, Y., Park, S., & Ye, J. C. (2020). Deep Learning COVID-19 Features on CXR Using Limited Training Data Sets. IEEE Transactions on Medical Imaging, 39(8), 2688–2700. doi:10.1109/TMI.2020.2993291.

[12] Ausawalaithong, W., Thirach, A., Marukatat, S., & Wilaiprasitporn, T. (2018). Automatic Lung Cancer Prediction from Chest X-ray Images Using the Deep Learning Approach. 2018 11th Biomedical Engineering International Conference (BMEiCON). doi:10.1109/bmeicon.2018.8609997.

[13] Feng, S., Zhou, H., & Dong, H. (2019). Using deep neural network with small dataset to predict material defects. Materials & Design, 162, 300–310. doi:10.1016/j.matdes.2018.11.060.

[14] Barz, B., & Denzler, J. (2020). Deep Learning on Small Datasets without Pre-Training using Cosine Loss. 2020 IEEE Winter Conference on Applications of Computer Vision (WACV). doi:10.1109/wacv45572.2020.9093286.

[15] Rajpurkar, P., Park, A., Irvin, J., Chute, C., Bereket, M., Mastrodicasa, D., Langlotz, C. P., Lungren, M. P., Ng, A. Y., & Patel, B. N. (2020). AppendiXNet: Deep Learning for Diagnosis of Appendicitis from a Small Dataset of CT Exams Using Video Pretraining. Scientific Reports, 10(1). doi:10.1038/s41598-020-61055-6.

[16] Brigato, L., & Iocchi, L. (2021). A Close Look at Deep Learning with Small Data. 2020 25th International Conference on Pattern Recognition (ICPR), Milan, Italy, 2490-2497. doi:10.1109/icpr48806.2021.9412492.

[17] Huang, Y., Jing, J., & Wang, Z. (2021). Fabric Defect Segmentation Method Based on Deep Learning. IEEE Transactions on Instrumentation and Measurement, 70, 1–15. doi:10.1109/tim.2020.3047190.

[18] Jing, J., Wang, Z., Rätsch, M., & Zhang, H. (2022). Mobile-Unet: An efficient convolutional neural network for fabric defect detection. Textile Research Journal, 92(1–2), 30–42. doi:10.1177/0040517520928604.

[19] Ajmi, C., Zapata, J., Martínez-Álvarez, J. J., Doménech, G., & Ruiz, R. (2020). Using Deep Learning for Defect Classification on a Small Weld X-ray Image Dataset. Journal of Nondestructive Evaluation, 39(3), 1–13. doi:10.1007/s10921-020-00719-9.

[20] Ju, J., Zheng, H., Xu, X., Guo, Z., Zheng, Z., & Lin, M. (2022). Classification of jujube defects in small data sets based on transfer learning. Neural Computing and Applications, 34(5), 3385–3398. doi:10.1007/s00521-021-05715-2.