S & M 2800

Implementation of a Fruit Quality Classification Application Using an Artificial Intelligence Algorithm

Ming-Chih Chen, Yin-Ting Cheng,* and Chun-Yu Liu

Department of Electronic Engineering, National Kaohsiung University of Science and Technology (First Campus), No. 1, University Rd., Yanchao Dist., Kaohsiung City 82445, Taiwan

(Received May 16, 2021; accepted November 24, 2021)

Keywords: real-time object detection, object track, fruit classification, machine learning, neural network model

Fruit quality classification in the consumer market has become a considerable burden following the decrease in the young adult population engaged in agriculture in Taiwan owing to its labor-intensiveness. We propose a system to identify the external quality of fruit, which utilizes a camera as an image sensor and an artificial intelligence algorithm as a classifier. This application is suitable for real operating environments. Fruits are mainly detected by the "you only look once" (YOLO)-V3 algorithm, with the designated fruit continuously tracked using the characteristics of the image, such as size, height, width, etc., and the quality of fruit is detected during the tracking process. Finally, the switching gap of the application distinguishes fruits of different quality. The proposed application detects round fruit such as apples, oranges, and lemons using our newly developed process. We also provide a graphical user interface to control and collect data, evaluate models, and monitor the entire system operation to improve the efficiency of the proposed application. The experimental results show that the proposed application achieves an accuracy rate of up to 88% after testing on 6000 fruit images.

1. Introduction

In recent years, the miniaturization of chips and the technological advancement of image sensors have proceeded. Artificial intelligence algorithms have also been rapidly developed and applied in various fields. Among them, machine learning and deep learning have been extensively used in image processing as their pattern recognition capabilities are compatible with human vision. In machine learning and deep learning, features can be captured from data through training. We can apply feature operations in fruit detection and recognition to implement automatically controlled gateways for fruit quality classification with supervised learning.

We have been investigating deep learning applications in machine learning and their application to fruit quality classification, which is an essential but labor-intensive process in agriculture because human vision and manual screening are often required. Hence, we have designed a classification application for automatically sorting fruits according to their quality to help farmers. Furthermore, we aim to implement the system with a reduced hardware cost.

*Corresponding author: e-mail: i107109118@nkust.edu.tw
https://doi.org/10.18494/SAM3553

ISSN 0914-4935 © MYU K.K.
https://myukk.org/

2. Literature Review

Krizhevsky *et al.* proposed the training of a large, deep convolutional neural network (CNN) with high-resolution images from the ImageNet image database.(1) In 2017, Howard *et al.* proposed a highly effective convolution computation, depthwise (DW) convolution, which reduces the number of parameters and computation.(2) When the input consists of three channels, each channel is computed using a single convolution kernel. The main difference from the convolution computation of Krizhevsky *et al.* is that the DW convolution is performed on a 2D plane. In addition, the size of the filter must be the same as the input channel of the previous layer. Finally, the size of the output tensor from the figure must be consistent with the number of filters.

Chollet also proposed the use of a type of convolution computation, referred to as the DW separable convolution, which is composed of two architectures: the first architecture is the same as the DW convolution mentioned above, and the second architecture is called the pointwise (PW) convolution.(3) In 2018, Sandler *et al.* proposed a network architecture that is a continuation of Chollet's research.(4) In this architecture, there are two modes to be set. In the Stride=1 mode, the pace of each movement is one pixel and convolution is performed with the convolution kernel. The input features and the completed features are summed to solve the gradient value disappearance problem. In the Stride=2 mode, no summation is performed. Instead, a linear activation function is used when a 1×1 kernel is used for convolution. Chen *et al.* proposed the octave convolution, in which, unlike the CNN, a new alpha hyperparameter is added to set the size of its value.(5)

Apte *et al.* applied the "you only look once" (YOLO) algorithm to a mobile app in 2017, where the algorithm was used to classify images taken by the phone.(6) The authors modified the YOLO feature extraction network model to enhance the inference speed of the model. Their work built on that of Iandola *et al.*, who applied a fire module layer to significantly reduce the number of parameters and increase the image detection speed.(7) In 2018, Redmon and Farhadi proposed the YOLO-v3 algorithm, which is an improvement of the YOLO-v1 algorithm proposed by Apte *et al.*(8) The YOLO-v3 algorithm was able to process 608×608 images on an NVIDIA Titan X graphics card at a speed of up to 20 frames per second. Santad *et al.* proposed the use of camera images to enhance security monitoring systems and achieve tracking through object movements.(9) The proposed system used the YOLO algorithm to detect luggage and analyze its association with its owner.

Yogesh and Dubey established a feature database containing the external properties of fruit (such as color, size, shape, and texture), wherein images were first segmented and a speeded-up robust features algorithm was applied to the segmented images. The feature database was then used to detect fruit defects.(10) In 2018, Khaing *et al.* proposed the use of the backpropagation algorithm to train a CNN for the identification and classification of fruit using an automatic classification system based on vision.(11) Since object detection using artificial intelligence algorithms requires numerous images for training, Jeong *et al.* proposed the use of the YOLO algorithm to construct an image preprocessing system and demonstrated its capability of highly efficient training.(12) In 2019, Hendry and Chen proposed the use of the YOLO-Darknet deep learning framework for car number plate detection. The authors used the YOLO algorithm with seven convolution layers to detect a single class.(13) The detection method is a sliding-window process. In the same year, Sachin *et al.* proposed a method to detect and classify three green vegetables of different types and sizes. The method used the YOLO algorithm with TensorFlow and OpenCV for image recognition. OpenCV was primarily used to preprocess the images before training, and the YOLO algorithm was used for detection and classification.(14) In 2020, Liu *et al.* proposed the use of the YOLOv3 to improve a tomato detection model. The model replaced the traditional rectangular box (R-Bbox) with a circular box (C-Bbox) for tomato localization.(15) Chen *et al.* proposed the combination of an artificial intelligence algorithm and microprocessors to control the gateway switch of a conveyor.(16) The platform was used to build a small fruit/vegetable quality classification system.

3. Structure of System

3.1 System overview

The proposed system combines appropriate equipment and technologies, such as the YOLO object detection algorithm, an STM32 microprocessor, a graphical user interface, and an MG996R servomotor, to recognize and classify the quality of fruit. The system's functions include fruit detection, quality recognition, and fruit tracking and control.

The system is coupled with a graphical user interface to display the current location of the fruit and the classification status. Cameras are integrated to perform fruit image detection and image preprocessing. By using the YOLO algorithm for fruit detection, overlapping objects can be separated, and the detected fruit image can be tracked continuously. A fruit recognition model is designed to recognize the quality of the fruit and perform classification. Another model is then used to determine whether the quality of the fruit is satisfactory.

Figure 1 shows the structure diagram of the fruit quality classification system, which uses a neural network model trained by two supervised learning methods. Model-A uses the YOLO algorithm to perform object detection model training; Model-B uses the CNN architecture to design the recognition model. The model is trained to search for functions fitting the features in the data. After evaluating the completed model, the model is integrated into the system architecture.

3.2 Software architecture

The fruit quality classification system is divided into four parts. The first part consists of the software control of the front and back gateways. The second part is the network model built by the CNN method, where Model-A uses Tiny-YOLO to detect fruit on the conveyor platform and Model-B recognizes the type of fruit and checks for defects.

The third part consists of the comparison and elimination algorithm used to eliminate the repeated bounding boxes. The fourth part is the tracking algorithm used to track the fruit on the conveyor platform. Finally, TensorRT is applied to compress the model. Figure 2 shows a software diagram of the fruit quality classification system.

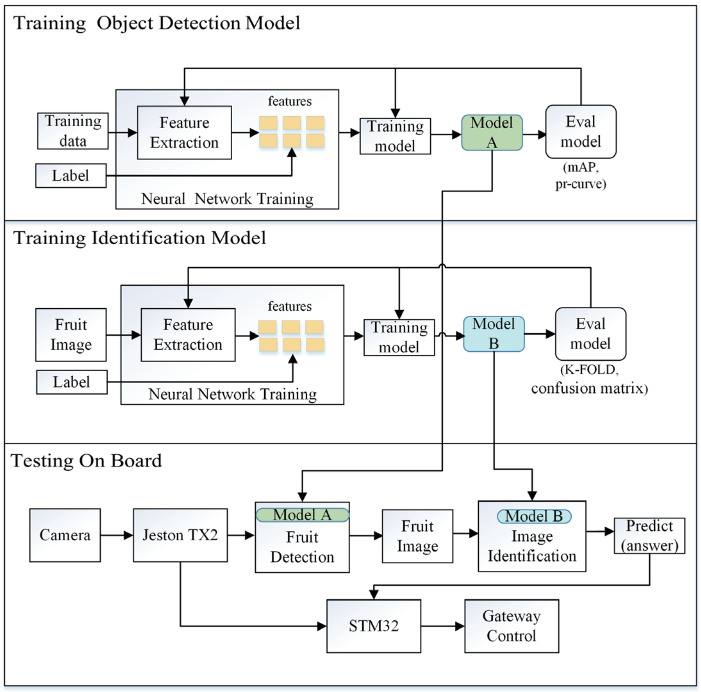


Fig. 1. (Color online) Structure diagram of fruit quality classification system.

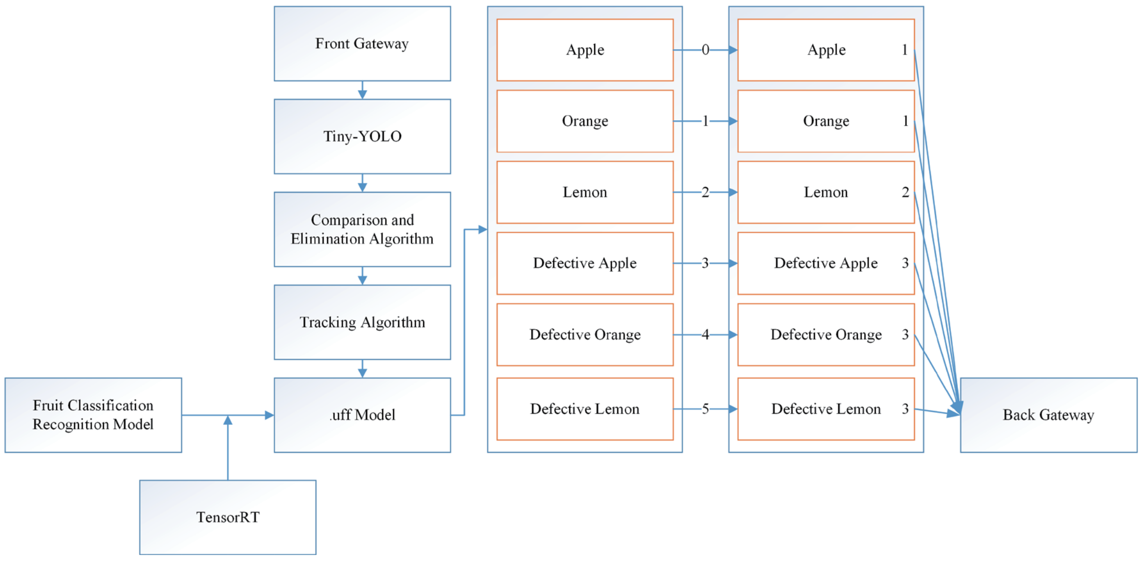


Fig. 2. (Color online) Software diagram of fruit quality classification system.

4. Research Methods

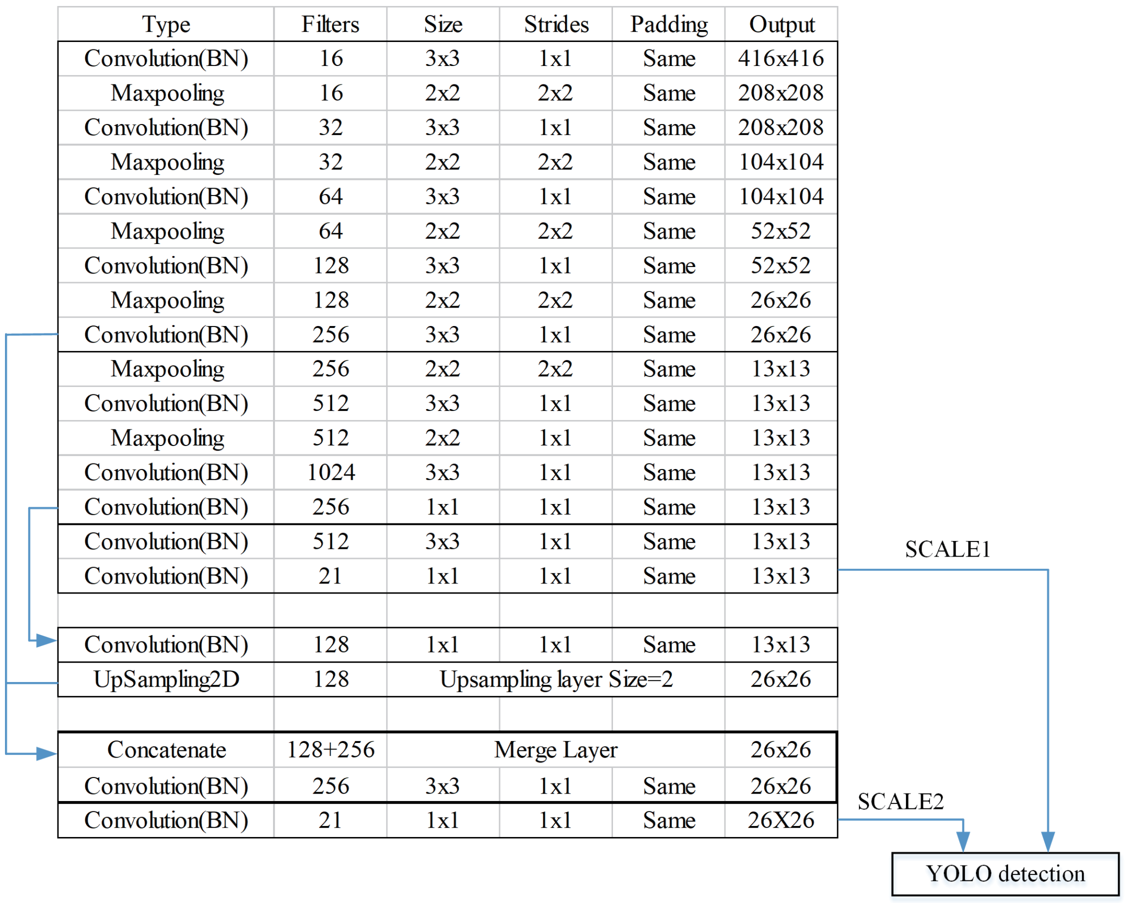
4.1 Object detection model

In this work, the system is designed with the Tiny-YOLO algorithm as the baseline, and different feature acquisition networks are designed. In the proposed system, we classify the good fruit in one category and the defective fruit in another category. The time required for classification can be reduced using the designed classification method, which helps the model to converge and generate the fruit dataset faster. Fruit detection is then performed on the trained model.

The Tiny-YOLO architecture is composed of the CNN and Maxpooling. Figure 3 shows the complete Tiny-YOLO architecture. A batch normalization (BN) layer is added after each convolution layer to normalize the parameters, facilitate model learning, and accelerate training.

4.2 Comparison and elimination algorithm

We have implemented an algorithm to compare the distance of each box and the nearest bounding box one by one, which eliminates repeated boxes and reduces errors in object tracking due to fruit defects being occluded by the good parts. The elimination principle is to eliminate the object bounding box without defects when both an object bounding box with defects and an object bounding box without defects appear simultaneously, as shown in Fig. 4.



Type	Filters	Size	Strides	Padding	Output
Convolution(BN)	16	3x3	1x1	Same	416x416
Maxpooling	16	2x2	2x2	Same	208x208
Convolution(BN)	32	3x3	1x1	Same	208x208
Maxpooling	32	2x2	2x2	Same	104x104
Convolution(BN)	64	3x3	1x1	Same	104x104
Maxpooling	64	2x2	2x2	Same	52x52
Convolution(BN)	128	3x3	1x1	Same	52x52
Maxpooling	128	2x2	2x2	Same	26x26
Convolution(BN)	256	3x3	1x1	Same	26x26
Maxpooling	256	2x2	2x2	Same	13x13
Convolution(BN)	512	3x3	1x1	Same	13x13
Maxpooling	512	2x2	1x1	Same	13x13
Convolution(BN)	1024	3x3	1x1	Same	13x13
Convolution(BN)	256	1x1	1x1	Same	13x13
Convolution(BN)	512	3x3	1x1	Same	13x13
Convolution(BN)	21	1x1	1x1	Same	13x13
Convolution(BN)	128	1x1	1x1	Same	13x13
UpSampling2D	128	Upsampling layer Size=2			26x26
Concatenate	128+256	Merge Layer			26x26
Convolution(BN)	256	3x3	1x1	Same	26x26
Convolution(BN)	21	1x1	1x1	Same	26X26

Fig. 3. (Color online) Complete Tiny-YOLO architecture.

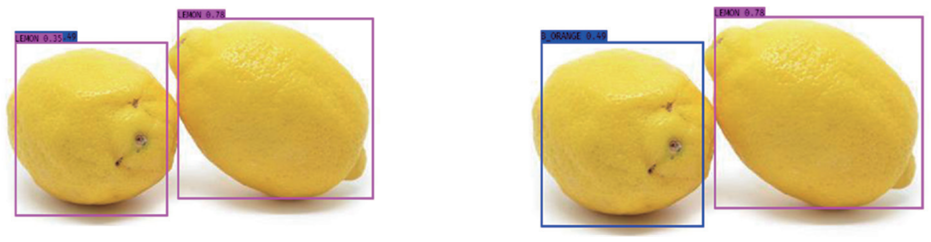


Fig. 4. (Color online) Example of eliminating repeated boxes.

4.3 Object tracking algorithm

The proposed object tracking algorithm achieved a short computation time by using the size, width, and height of the received image and the confidence level of the detected object to perform a numerical comparison and perform tracking without image information. In addition, we track a fruit to improve the classification accuracy since the fruit is rolled during the delivery process. By detecting each fruit surface, the damaged parts can be detected for recognition. Statistics are calculated before the fruit leaves the platform to find the highest number for channel separation. Figure 5 shows the fruit tracking process.

4.4 Classification recognition model

Using an object detection model to output the images required by the architecture reduces the time and labor costs. In addition, two architectures, a fully connected architecture and an autoencoder, are used to build this model, which is followed by a damage recognition process. An interpolation method is used to demagnify the image and reduce the time for model training and detection. In the future, this method can be adapted to expand its application to different types of fruit.

4.4.1 Convolution layer and fully connected architecture

The architecture uses four convolution layers, as shown in the red box on the left of Fig. 6. The green box is the fully connected architecture used for classification. All parameters and weights can be trained when using such an architecture. Finally, there are six nodes in the output, which coincide with the six categories of fruit used for testing in this study.

4.4.2 Convolution layer and deconvolution layer architecture (autoencoder)

Figure 7 shows the convolution and deconvolution architecture, also referred to as the autoencoder architecture. The red part consisting of an asymmetric architecture is the encoder and the green part is the decoder. This architecture first compares the similarity between the result obtained by automatic feature extraction and the result entered initially. If the similarity requirement is satisfied, the encoder output features are input to the classification layer.

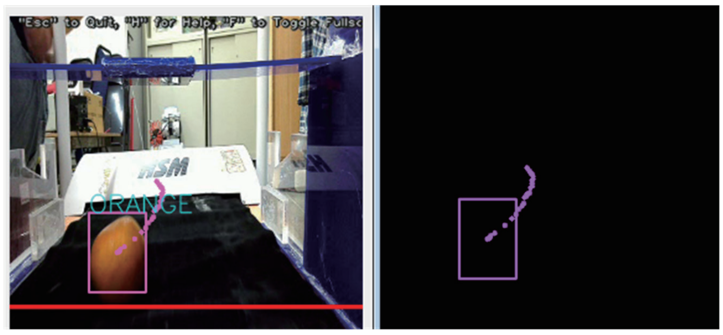


Fig. 5. (Color online) Object tracking using the image.

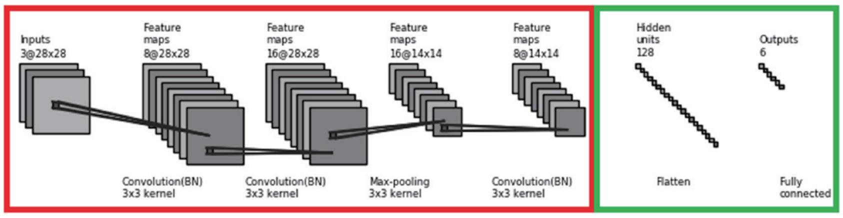


Fig. 6. (Color online) Convolution layers and fully connected layer architecture.

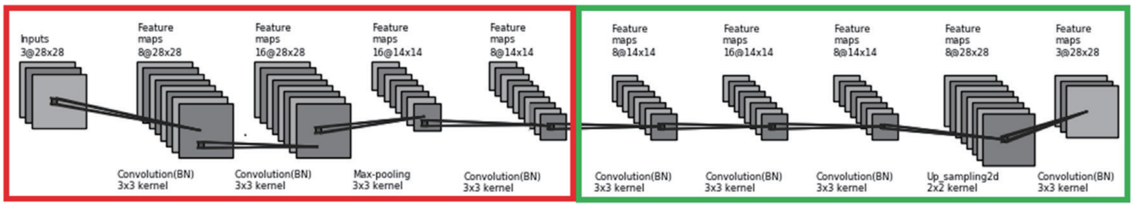


Fig. 7. (Color online) Autoencoder architecture.

5. Experimental Results

5.1 Recognition model evaluation and training

The fruit identification model uses the YOLO algorithm to process the dataset. The YOLO object detection algorithm circles the fruit and crops out the part containing the fruit from an image or video, similar to the region of interest (ROI) method. The photos are classified into six categories. The training and verification picture ratios are both set to 0.2. The classification method assigns labels of 0 to 5 for apples, oranges, lemons, defective apples, defective oranges, and defective lemons, respectively, as shown in Table 1.

The two trained recognition model architectures are configured with a batch size of 256 and 50 epochs. The first architecture is the fully connected architecture and the second is the autoencoder architecture. Categorical cross-entropy is used as the loss function, and Adam is used as the learning optimizer. The initial established default learning rate α is 0.001. Training is stopped when the new loss value is not lower than the original loss value more than 10 consecutive times.

5.2 Training model

5.2.1 Convolution layer and fully connected architecture

In this architecture, all network layers are configured to be trainable from start to finish. All network layers change with each training session due to the influences of the forward and backward propagation algorithms.

5.2.2 Convolution layer and deconvolution layer architecture (autoencoder)

This framework uses a two-stage training model. The first stage encodes the eigenvalues in an automatic encoder and decodes the encoded eigenvalue. The loss function uses the mean squared error (MSE), and the learning optimizer is RMSprop, which is an adaptive learning rate method. The MSE is used to calculate the coder input and decoder output. As shown in Fig. 8,

Table 1
Number of pictures in the dataset.

	Apples	Oranges	Lemons	Defective apples	Defective oranges	Defective lemons
Training and verification	280	405	267	348	291	324
Testing	745	878	638	736	720	749

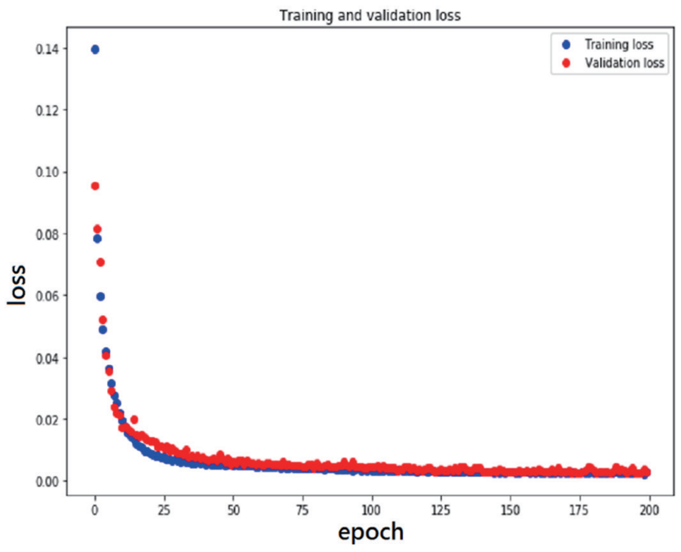


Fig. 8. (Color online) Loss value of autoencoder.

both the training loss and validation loss decrease towards zero, which means that the model is learning.

5.3 *k*-fold cross-validation

With Fig. 9 as an example, the data are randomized and divided into k (=3) equal parts. One part of the data is treated as the validation data and the remaining $k - 1$ equal parts are treated as the training data. The procedure is repeated until the training data are used as validation data. Finally, the cross-validation is repeated k times, and the sum and average of each result are calculated. The benefits of this method are that it learns from the data in multiple directions to avoid becoming trapped in a localized suboptimal solution and avoids overfitting.

Figure 10 shows the accuracy of k-fold evaluation obtained by the fully connected layer architecture with all parameters and weights of the model trained directly from scratch. Figure 11 shows the accuracy obtained by the autoencoder. The accuracy of the fully connected

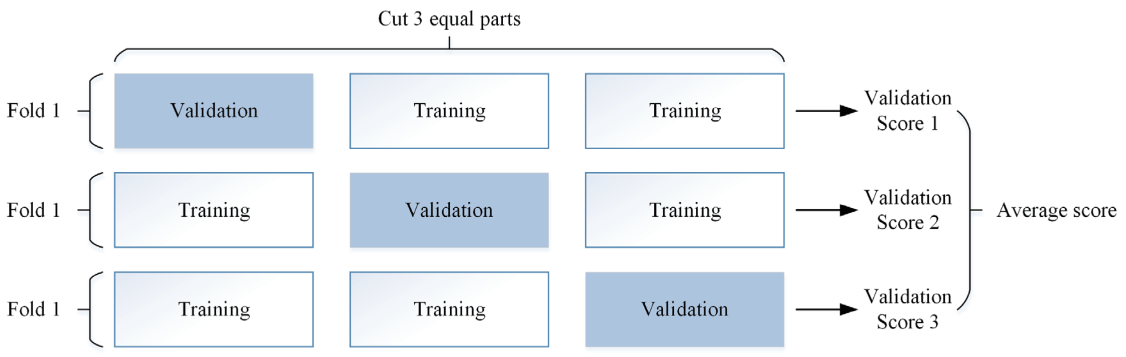


Fig. 9. (Color online) Segmentation in *k*-fold cross-validation.

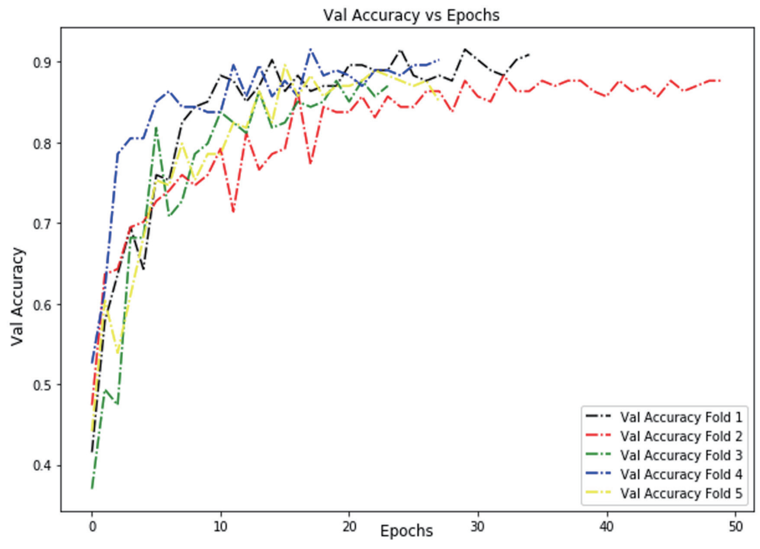


Fig. 10. (Color online) Cross-validation accuracy of fully connected architecture.

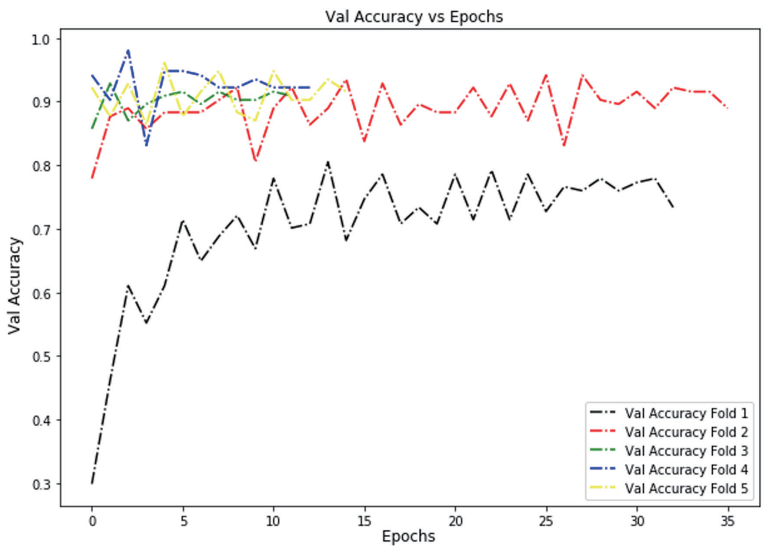


Fig. 11. (Color online) Cross-validation accuracy of autoencoder architecture.

layer architecture increases from zero, whereas that of the autoencoder starts at around 80%. Moreover, in exceptional cases, the accuracy starts at zero and increases, as shown by the black line in Fig. 11. This means that these test data are very different from the other test data. The distribution of the randomly selected data is significantly different from that of the training data. However, the average accuracy approached 87% with increasing number of epochs for both architectures. Nevertheless, the autoencoder shortens the model training time.

5.4 Confusion matrix

The confusion matrix is used to evaluate the accuracy of supervised learning algorithms. The matrix compares test data with the data after model prediction and uses various indices to measure the classification effect. There are four possible situations when using binary classification as an example, as shown in Table 2.

Tables 3 and 4 provide the numerical evaluations for the confusion matrix for the fully connected and autoencoder architectures, respectively. Both architectures achieved 88% precision and recall. However, for Class 4 (defective oranges), the recall rate was relatively low, with an assessed value between 65% and 68%, about 15% lower than those of the other categories. The low recall rate was because the images of Class 3 (defective apples) and Class 4 (defective oranges) in Dataset 1 were very similar, causing defective oranges to be identified as defective apples and leading to misjudgment.

Figures 12 and 13 show the resultant graphs for the correctly identified number of pictures using the confusion matrix for both architectures. The numbers of misjudgments were higher in both architectures for the classification of Class 4 (defective oranges), but overall, there are very few differences between the two architectures.

Table 2
Description of the confusion matrix.

		Predictive value	
		0	1
Actual value	0	True negative (TN)	False positive (FP)
	1	False positive (FP)	True negative (TN)

Table 3
Numerical evaluation for the confusion matrix of the fully connected architecture.

	Precision	Recall	F1-score	Support
Class 0	0.84	0.92	0.88	754
Class 1	0.89	0.97	0.93	878
Class 2	0.94	0.97	0.95	638
Class 3	0.88	0.87	0.87	736
Class 4	0.84	0.65	0.74	720
Class 5	0.92	0.93	0.93	749
Avg./Total	0.89	0.89	0.88	4466

Table 4
Numerical evaluation for the confusion matrix of the autoencoder architecture.

	Precision	Recall	F1-score	Support
Class 0	0.85	0.88	0.86	754
Class 1	0.92	0.95	0.94	878
Class 2	0.94	0.95	0.94	638
Class 3	0.83	0.93	0.88	736
Class 4	0.85	0.68	0.76	720
Class 5	0.89	0.89	0.89	749
Avg./Total	0.88	0.88	0.88	4466

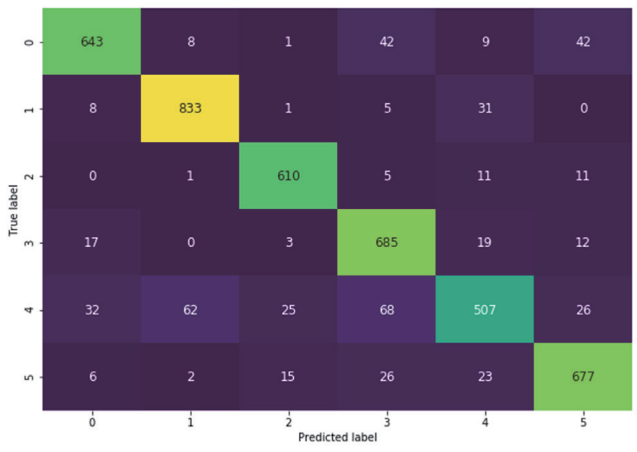


Fig. 12. (Color online) Number of pictures for the confusion matrix of the fully connected architecture.

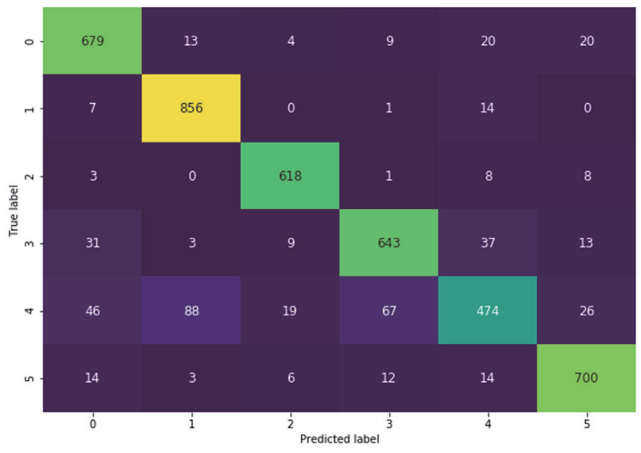


Fig. 13. (Color online) Number of pictures for the confusion matrix of the autoencoder architecture.

6. Conclusion

In this research, we applied the Tiny-YOLO neural network model to perform object detection and compared several other models in terms of structural performance. We developed a system by building the model architecture with the TensorFlow and Keras frameworks. In addition, we acquired the images used in the dataset by using a camera as an image sensor.

With a four-layer convolution network for object detection, the fruit recognition model for classification of the fruit quality achieved an accuracy of 88%. In an experiment, the complete system was implemented with a graphical user interface for user-friendly operation. In this work, a relatively large model was used to classify the three types of fruit. If the model is applied to a more compact development board, then a single classification model can be used for each type of fruit.

References

1 A. Krizhevsky, S. Ilya, and E. H. Geoffrey: Commun. ACM **60** (2017) 84. https://doi.org/10.1145/3065386
2 A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam: Comput. Sci. (2017). https://arxiv.org/abs/1704.04861
3 F. Chollet: Proc. Conf. Computer Vision and Pattern Recognition (IEEE, 2017) 1251. http://doi.org/10.1109/CVPR.2017.195
4 M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen: Proc. Conf. Computer Vision and Pattern Recognition (IEEE, 2018) 4510. http://doi.org/10.1109/CVPR.2018.00474
5 Y. Chen, H. Fan, B. Xu, Z. Yan, Y. Kalantidis, M. Rohrbach, Y. Shuicheng, and J. Feng: Proc. Int. Conf. Computer Vision (IEEE, 2019) 3435. http://doi.org/10.1109/ICCV.2019.00353
6 M. Apte, S. Mangat, and P. Sekhar: Stanford University (2017). http://cs231n.stanford.edu/reports/2017/pdfs/135.pdf
7 F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer: Computer Science (2016). https://arxiv.org/abs/1602.07360
8 J. Redmon and A. Farhadi: Computer Science (2018). https://arxiv.org/abs/1804.02767
9 T. Santad, P. Silapasupphakornwong, W. Choensawat, and K. Sookhanaphibarn: Proc. IEEE Global Conf. Consumer Electronics Conf. (IEEE, 2018) 157. https://doi.org/10.1109/GCCE.2018.8574819
10 Yogesh and A. K. Dubey: Proc. Int. Conf. Reliability, Infocom Technologies and Optimization (IEEE, 2016) 590. https://doi.org/10.1109/ICRITO.2016.7785023
11 Z. M. Khaing, Y. Naung, and P. H. Htut: Proc. IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conf. (IEEE, 2018) 1805. https://doi.org/10.1109/EIConRus.2018.8317456
12 H. J. Jeong, K. S. Park, and Y. G. Ha: Proc. Int. Conf. Big Data and Smart Computing (IEEE, 2018) 635. https://doi.org/10.1109/BigComp.2018.00113
13 Hendry and R. C. Chen: Image Vision Comput. **87** (2019) 47. https://doi.org/10.1016/j.imavis.2019.04.007
14 C. Sachin, N. Manasa, V. Sharma, and A. A. Nippun Kumaar: Proc. Int. Conf. Cutting-edge Technologies in Engineering (IEEE, 2019) 101. https://doi.org/10.1109/ICon-CuTE47290.2019.8991457
15 G. Liu, J. C. Nouaze, P. L. T. Mbouembe, and J. H. Kim: Sensors **20** (2020) 2145. https://doi.org/10.3390/s20072145
16 M.-C. Chen, Y.-T. Cheng, and C.-Y. Liu: Sens. Mater. **34** (2022) 135. https://doi.org/10.18494/SAM3552