Hindawi
Mathematical Problems in Engineering
Volume 2022, Article ID 6505372, 14 pages
https://doi.org/10.1155/2022/6505372

Research Article

A Novel Multicategory Defect Detection Method Based on the Convolutional Neural Network Method for TFT-LCD Panels

Yung-Chia Chang,[1] **Kuei-Hu Chang**,[2,3] **Hsien-Mi Meng**,[4] **and Hung-Chih Chiu**[1]

[1]*Department of Industrial Engineering and Management, National Yang Ming Chiao Tung University, Hsinchu 300, Taiwan*
[2]*Department of Management Sciences, R.O.C. Military Academy, Kaohsiung 830, Taiwan*
[3]*Institute of Innovation and Circular Economy, Asia University, Taichung 413, Taiwan*
[4]*Department of Industrial Engineering and Enterprise Information, Tunghai University, Taichung, 407, Taiwan*

Correspondence should be addressed to Kuei-Hu Chang; evenken2002@gmail.com

Received 8 December 2021; Revised 26 January 2022; Accepted 29 January 2022; Published 16 February 2022

Academic Editor: Gengxin Sun

Defects on thin film transistor liquid crystal display (TFT-LCD) panel could be divided into either macro- or microdefects, depending on if they are easy to be detected by the naked eye or not. There have been abundant studies discussing the identification of macrodefects but very few on microones. This study proposed a multicategory classification model using a convolutional neural network model to work with automatic optical inspection (AOI) for identifying defective pixels on the TFT-LCD panel. Since the number of nondefective pixels outnumbered the defective ones, there exists a very serious class-imbalanced problem. To deal with that, this study designed a special training strategy that worked with data augmentation to increase the effectiveness of the proposed model. Actual panel images provided by a mobile manufacturer in Taiwan are used to demonstrate the efficiency and effectiveness of the proposed approach. After validation, the model constructed by this study had 98.9% total prediction accuracy and excellent specificity and sensitivity. The model could finish the detection and classification process automatically to replace the human inspection.

1. Introduction

Defects on thin film transistor-liquid crystal displays (TFT-LCD) can be divided into macrodefects and microdefects [1]. The main difference between them is whether they can be easily detected by the naked eye or not. For example, Mura and light leakage are categorized as macrodefects, while defective pixels, pin holes, and particles are categorized as microdefects due to their minute size. Most studies related to the detection of defects on TFT-LCD panels focus on macro defects, especially Mura [2–7].

With the advances in technology, pixel sizes are getting smaller [8, 9]. Using a 15-inch panel with Full HD (FHD) resolution, commonly used in notebook computers, as an example, the pixel size is about 0.19 mm × 0.20 mm. According to Curcio et al. [10], the size limit that the human naked eye can distinguish is about 0.35 ARC points (arc-minute), which is equivalent to 300 pixels per inch (ppi), about 0.08 mm. To use the light-on test to sort panels based on the number of defective pixels, inspectors need to work in a high luminance ratio (low ambient light and high local illumination) environment for long working hours and search for defects that are close to the human limit. On the other hand, using automatic optical inspection (AOI) technology to assist visual inspection may reduce the risk of damage to inspectors' eyes and can also improve the efficiency of the panel sorting process. Such a process reflects the trend of defect sorting processes in Industry 4.0. AOI technology relies on visual sensing equipment, such as photosensitive charge coupled device (CCD) cameras, to replace the human eye for object detection. In general, the AOI process can be divided into four stages: image acquisition, image preprocessing, image classification, and output results. In this process, the efficiency of the image preprocessing and image classification determines the efficiency of the entire

process. Many studies have discussed topics related to the assistance of AOI in decision-making [11–14].

Few studies have focused on the detection of defective pixels in TFT-LCD panels. Guo et al. [15] developed a process that used notch filter and threshold segmentation method to detect subpixel defects on LCD screen images. Çelik et al. [16] proposed a real-time defective pixel detection system for LCD TV products using a deep learning algorithm. They compared the ability to detect pixel-level defects by using RetinaNet [17], M2Det [18], and Yolov3 [19] network and found that RetinaNet based architecture provides balanced results in terms of accuracy and use of time. However, their model could only detect the defective pixels but did not furthermore categorize them into their corresponding type. According to ISO/TR 9241-310 standards, defective pixels can be divided into full-bright, full-dark, half-bright, and half-dark pixels. Each pixel on a color TFT-LCD panel is composed of individual red, green, and blue subpixels. Thus, TFT-LCD panel defects can include 12 different combinations of defective pixels. ISO/TR 9241 also describes the ISO's tolerance for LCD panels in different classes. For example, class 0 panels are completely defect-free (i.e., no full or subpixel defects are allowed), while class 1 to class 3 panels may allow five to 50 defects. Currently, the best-quality panels (commonly referred to as Grade-A panels in practice) sold by TFT-LCD panel manufacturers to notebook computer manufacturers are class 1 (and above) panels, which means macrodefects have been screened out but there may still exist a small number of defective pixels.

Many defect classification studies on TFT-LCD panels were made from the perspective of TFT-LCD manufacturers. This study was motivated by a practical problem raised by a notebook computer manufacturer. It is known that notebook computers sold in the market could be divided by several grades in terms of price, ranging from several hundreds of US dollars to several thousands, under the same brand name. Therefore, to maximize the product value to its customers, a notebook computer manufacturer would want to put the best-quality TFT-LCD panels to its top-grade models and leave the ones with a couple of defects but still counted as grade-A panels to lower-grade models. The type, location, and number of defective pixels are critical for panel classifications. Traditionally, this panel sorting task was conducted by human eyes using light-on tests. With the recent advanced high-resolution camera and data processing technologies, AOI became an alternative. This paper aimed at constructing a multicategory classification model based on deep learning that could be used to improve the efficiency of model construction to identify defective pixels in TFT-LCD panels while maintaining high classification accuracy. Real TFT-LCD images provided by a brand-name computer manufacturer in Taiwan were applied to demonstrate the feasibility and effectiveness of the proposed model.

The follow-up structure of this paper is as follows. Section 2 presents the preliminaries, including deep learning, the convolutional neural network (CNN), and edge detection. The model construction method and the construction process are introduced in Section 3. The numerical result of the model is shown in Section 4, and the conclusion is discussed in Section 5.

2. Preliminaries

2.1. Deep Learning. Deep learning topics regarding multilayer perceptrons are once again gaining attention from researchers, mainly due to the use of nonlinear activation functions, which gives multilayer perceptrons the ability to solve linear indivisible problems that neural networks could not handle in the past [20]. Multilayer perceptrons have been widely used in various research areas, such as plant breeding [21], public construction project cost forecasts [22], landslide susceptibility [23], flood predictions [24], and thermal power plants [25].

In the case of model training for deep learning, in order to prevent overfitting, a sufficient amount of data is necessary. Tanner and Wong [26] first proposed the concept of data augmentation by using geometric transformation of the original dataset to increase the amount of data. This has become a widely applied method to solve the problem of insufficient data. The augmentation method is based on geometric transformations, such as rotation, flipping, or scaling, of the original images to accumulate the required data by using the existing dataset. This method could effectively solve the problem of a lack of training data without hurting the accuracy of the model training. For example, Cui et al. [27] and Ding et al. [28] applied the data augmentation method in topics regarding deep learning model training, showing the importance of data augmentation for training deep neural networks and how it can improve network performance by providing sufficient data.

Recently, deep learning has been proven to be a useful approach in image classification. For example, Lin et al. [29] used the CNN for wheat leaf disease classification, and Shao et al. [30] developed a novel convolutional deep belief network (DBN) to diagnose electric locomotive bearing faults. CNN is a commonly used mode in deep learning and has been successfully applied in many areas such as image processing, image recognition, medical analytics, noise reduction, and more. Liang et al. [31] applied CNN to classify blood cell images, Cui et al. [32] classified tire defects based on CNNs, Isogawa et al. [33] used CNN to perform noise reduction, and Lin et al. [29] constructed a CNN to identify wheat leaf diseases based on images. All these studies showed that CNN is a good fit for image recognition.

Deep learning is a kind of technology that implements machine learning and uses neural networks as the architecture for learning, extraction, and shaping multilevel representation data. During the history of deep learning's development, Hinton et al. [34] successfully trained neural networks by applying restricted Boltzmann machines (RBM) for the development of DBNs and also developed a solution for backpropagation issues. A brief introduction of RBM, DBN, and multilayer perceptrons is as follows.

2.1.1. RBM. An RBM is a two-layer stochastic network. Its two layers are a visible layer and a hidden layer, as shown in Figure 1. Nodes inside the layers are not connected to one

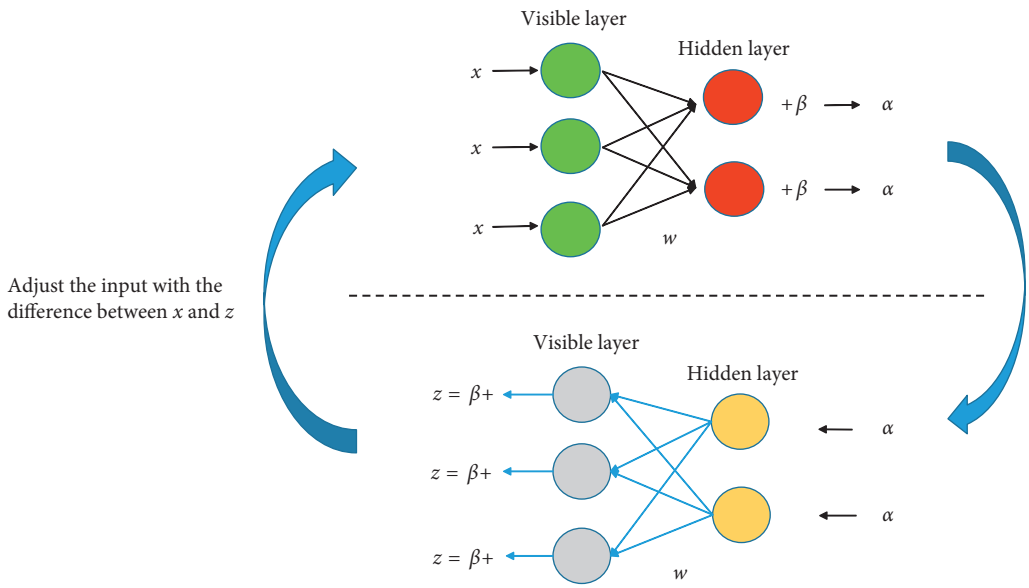

Figure 1: Basic RBM structure.

another. Input value x is sent into the network, and then each input value is multiplied by weight w and summarized with bias value β. The summarized value becomes the output of the hidden layer through the activation function. After this step is complete, output α is treated as the new input of the hidden layer and goes back through the hidden layer, after which it is multiplied by weight w, then moved to the visible layer and was added by a new random bias β, then, it became output z through the visible layer. The difference between the original input x and z is the baseline of the model adjustment. Figure 1 shows the basic structure of an RBM.

There are many relative types of research regarding RBM. For example, Chen et al. [35] applied RBM to enhance the performance of deep learning, Huang and Wang [36] used it to accelerate Monte Carlo simulations, and Liu et al. [37] inferred microRNA targets using RBM. Although RBM can be used to perform unsupervised learning and can be used for various purposes, it is still time consuming and faces efficiency issues in training, which is why researchers then stacked RBM to solve its lack of efficiency, thus ushering in the development of neural networks.

2.1.2. DBN. DBN is a powerful method of feature extraction proposed by Hinton and Salakhutdinov [38] that consists of a stack of RBMs. It can be interpreted as a probabilistic generative model where each node is called a probabilistic logic node (as it outputs the probability for each class). A generative pretraining step is used to help get rid of the overfitting problem. The basic structure of DBN is shown in Figure 2, which shows multiple layers being trained at the same time [38]. The output of the first stage becomes the input of the succeeding stage and continues until the last layer is reached. The continuous calculation and modification of the weights during the process decreases the training time of the neural network, which is the main reason why DBN is so widely applied in various areas. For example, Zhao et al. [39] constructed a high-resolution SAR image classification model based on discriminant DBN, and Shao et al. [30] applied DBN to diagnose electric locomotive bearing faults. DBNs have better efficiency and more powerful learning ability features compared to RBMs, allowing DBNs to perform unsupervised learning and possess the ability to handle nonlinearly separable problems. However, the size of the DBN network expands dramatically with the complexity of the target, which inevitably results in increased computing hardware consumption.

2.1.3. Multilayer Perceptron. The concept of the perceptron was first proposed by Rosenblatt [40] as a feedforward neural network expressed by the feature vector (a binary linear classifier). A real vector input is mapped into a binary output value. The perceptron concept proposed by Rosenblatt [40] is shown in Figure 3.

Rosenblatt [41] stacked multiple perceptrons to form a multilayer perceptron and create a basic deep learning model. The multilayer perceptron consists of multiple node layers, each of which contains a number of independent neurons that are all connected to the lower layer. Each neuron in the lower layer gets a weight value in the upper layer and is trained through a backpropagation algorithm. Since the multilayer perceptron can only solve the problem of linear separability, the problem of linear indivisibility is solved by the nonlinear activation function. In image recognition, Lin et al. [42] reconstructed the algorithm of lost frames of multiview videos in wireless multimedia sensor networks using multilayer perceptrons.

2.2. CNN. A convolutional neural network (CNN) is a type of artificial neural network that is often used in image recognition and classification. LeCun et al. [43] proposed the concept of CNN, which is a feedforward neural network

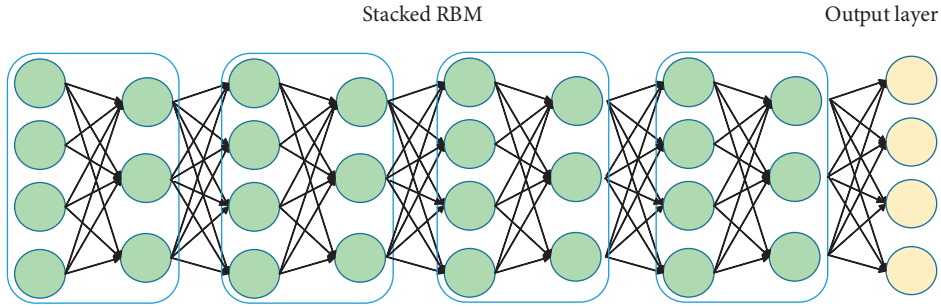

Figure 2: The basic DBN structure.

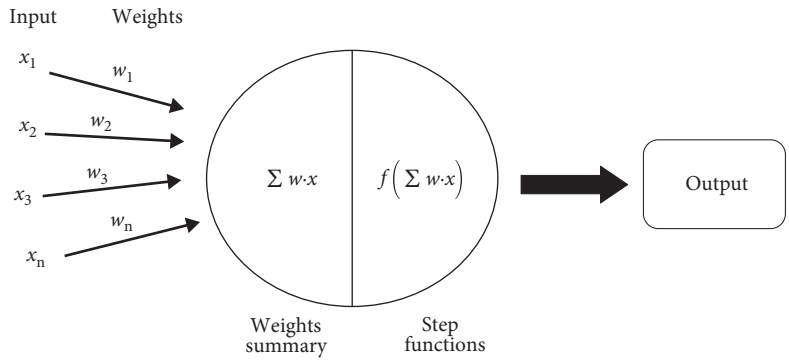

Figure 3: Multilayer perceptron concept.

divided into a convolutional layer, a pooling layer, a flatten layer, a fully connected layer, and a loss function layer. The structure is as follows.

2.2.1. Convolutional Layer. The convolutional layer is a set of parallel feature maps. The original image is scanned through a custom convolution kernel, and the scanned area is dot-sorted. The convolution kernel continuously multiplies the numbers in the kernel and the picture. The feature map is then obtained as the output at the end of this process. Figure 4 shows the basic concept of the convolution process.

2.2.2. Pooling Layer. The pooling layer combines the output of one cluster of anterior neurons with individual neurons of the lower layer, thus reducing the size of the input image to lower the dimension of the feature map. Commonly used pooling methods include max-pooling, average-pooling, and min-pooling. The max-pooling method selects the maximum absolute value of the pooling feature generated by the convolutional layer. Yang et al. [44] stated that max-pooling can better preserve features in images and can outperform other pooling methods (such as average-pooling), making it more applicable in neural networks that involve image processing. Because max-pooling has better efficiency in image processing, this study chose the max-pooling method as the main pooling method. This study divided blocks into $2 * 2$ windows, after which the maximum value in each block was taken as the corresponding output block to reduce the number of features. This method could not only reduce the number of parameters and calculations but also indirectly slow down the phenomenon of overfitting. The process of a max-pooling example is shown in Figure 5.

2.2.3. Flatten Layer. After going through the convolutional and pooling layers, some pooled feature maps will be obtained from the input images. In order to transform the feature maps into the input of the fully connected layer (which is a one-dimension vector), a flatten layer is required to flatten the pooled map. A concept of the flatten layer is shown in Figure 6.

2.2.4. Fully Connected Layer. After the feature is flattened, the network can then proceed to the fully connected layer. The fully connected layer gathers every feature map acquired from the previous layers connected to it and puts them into combinations. These combinations then help the model create probabilities of the category to which each input image should be assigned. These probabilities are input to the last layer and are used to present the final result.

2.2.5. Loss Function Layer. The loss function layer is the last layer of the network to present the output results with loss functions. Commonly used loss functions are softmax and sigmoid. The sigmoid function takes any kind of real number and returns it into a value that falls between 0 and 1. Since the probability only exists in the range of 0 to 1, the sigmoid function has better performance in predicting probabilities as the output. The softmax function takes the input of a vector consisting of K real numbers and then outputs the result to a probability distribution consisting of K

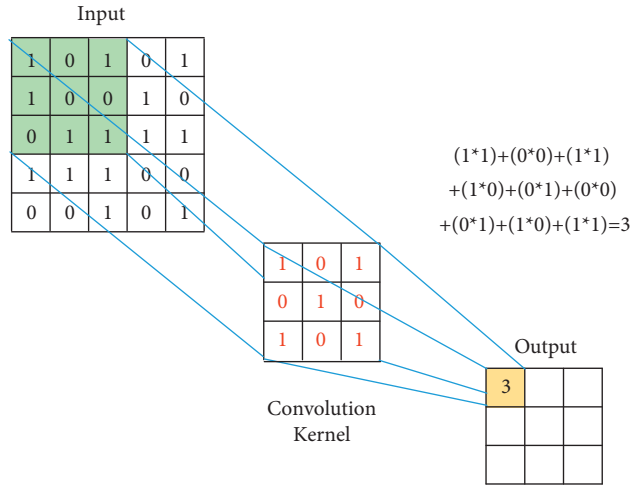

Figure 4: Basic concept of convolution.

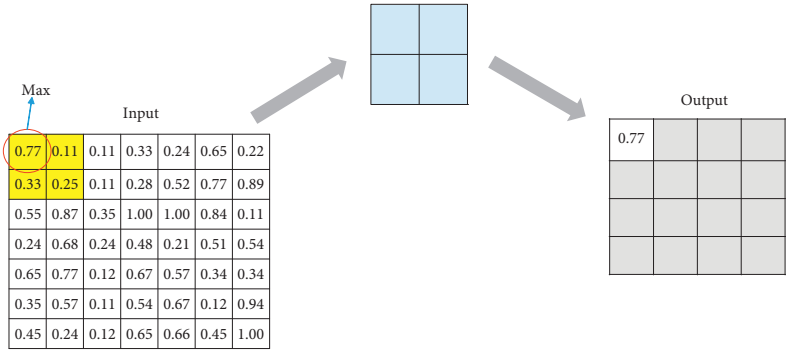

Figure 5: Using a 2×2 window to implement the max pooling process.

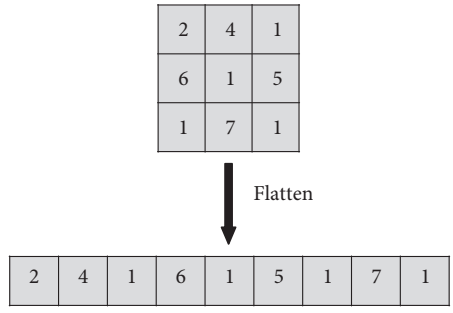

Figure 6: Flatten layer concept.

probabilities. All the output adds up to 1, and because of this, the softmax function is often used in processing problems that require a categorical output.

2.3. Edge Detection. Edge detection includes a variety of mathematical methods to identify an edge or a specific line in a digital image. Generally, an edge is a boundary that connects to two separate regions where the luminance of an image changes significantly or has discontinuities [45]. One of the most applied methods is Canny edge detection [46], due to its ability to extract features without disturbing them and also its low error rate at locating edges [47], making it valuable for image processing. A number of researchers have implemented Canny edge detection. Lee et al. [48] applied it for advanced mobile vision, and Zhou et al. [49] used Canny edge detection to develop visual odometry. The important steps of Canny edge detection are as follows [46]:

Step 1. Noise reduction

No edge detection techniques are likely to have good outputs using unprocessed images, so the first step is to reduce the noise in the original images.

Step 2. Compute gradients

The Canny algorithm takes the difference between neighboring $2 * 2$ areas to calculate the gradient magnitude and gradient direction in the image, which is then used in the following hysteresis process.

Step 3. Hysteresis thresholding

Hysteresis is a method of connecting the polyline generated by pixel iteration and then checking whether it is an edge. If it is confirmed as an edge, it is necessary to then check whether there are other edges in the surrounding area of the position. Canny edge detection also uses two different thresholds: a high threshold and a low threshold. If the pixel is larger than the low threshold, it is marked as an edge. Pixels that are greater than the low threshold and the high threshold at the same time are marked as edge pixels. The entire process stops if there are no other changes in the image.

3. Research Methods

Deep learning can solve the problem of low feature extraction efficiency in machine learning, but it often ends up with several binary models, making it hard to be implemented for practical use. Therefore, construction of a multicategory classification model is inevitable. In order to construct a multicategory classification mode for TFT-LCD panels, this study built a model for four types of defects (full-bright pixels, full-dark pixels, half-bright pixels, and half-dark pixels) under three colors (red, green, and blue), along with defect-free images containing the three colors. Figure 7 organized several examples of defective pixels considered in this study. This allowed users to go through only one multicategory model instead of needing numerous binary classification models to classify the defects, as shown in Figure 8.

The process of building the model is shown in Figure 9. The inputs of this model were captured by optical cameras, which were used to form a deep learning model and obtain features for further picture classification. The output of this model shows the categories to which each picture should belong.

The steps of the proposed model construction are as follows:

3.1. Step 1: Preprocess the Data. The efficiency of deep learning is related to the computing device; the more power the device has, the more effective the model can be. However, it is important to reach a balance between them. If the input image contains too many pixels, there may be insufficient computer memory to train the model. Since the pixel-level defects are extremely small, it is possible to cut the image captures into several smaller segments according to the hardware limitations of the computing device. Entering a smaller image can also help shorten the training time of the model.

Another common issue found in defect-identifying problems is unbalanced data, as the number of images with defects is far less than the number of images without defects. If all images were fed into model training, the resulting model will be likely to classify all images as normal images, resulting in high overall classification accuracy but low validation accuracy of minority categories, which is useless. As images that contain defects are extremely hard to collect, the data augmentation method can be used to increase defect images to avoid unbalanced data.

In practice, when the AOI was implemented, there will be cases with slight angle differences on the images caused by misalignment of the panel. In order to simulate this situation while augmenting images, the rotation technique was implemented to create misaligned images. Moreover, this study flipped the images to simulate the situation in which defective pixels may appear on different locations of the images. By rotating and/or flipping images, the proliferated images could be close to the original image without distortion.

After gathering the data, this study used numbers to label the images, as shown in Table 1. The entire image dataset was then split into two sets: a training set (80%), and a testing set (20%), for further model training and testing.

3.2. Step 2: Construct the Neural Networks. A convolutional neural network is composed of multiple neural network layers. While constructing layers in the neural network, the dimensions of the input images must be carefully calculated and modified, and the number of features that each convolution layer extracts, along with the size of the convolution kernel (kernel size) and the type of activation function, have to be set beforehand. There are many kinds of layers in a CNN, each of which has different construction orders due to its functionality.

The first layer is the convolution. The number of convolution kernels and the size of each kernel are set in this layer, which are used to determine the number and size of the features to be acquired. The activation function used in model training is determined in this layer as well. When the accuracy of the training data and the verification data reach a certain level but the testing data shows that the classification result is relatively low, the model does not have the right feature values. When that happens, it is necessary to increase the convolution layer to get the desired features of the image.

Next are the pooling layers. The main purpose of pooling is to reduce the number of parameters, which can reduce the computation time and also reduce the possibility of overfitting. Of the many pooling options available, this study used max pooling, as it has the ability to preserve the position of the feature, which is important for defect classification.

After the pooling layers comes to the construction of the fully connected layer. The main purpose of this layer is to collect the features obtained by the previous layers and classify them to facilitate model training and output. In order to avoid overfitting, a dropout function is often placed after the fully connected layer, which uses a probability value as the input. During model training, the neuron gets the probability controlled by dropout layers and has been turned off that according to these probabilities, then, prevents the occurrence of correlation between the feature values and resulting in overfitting.

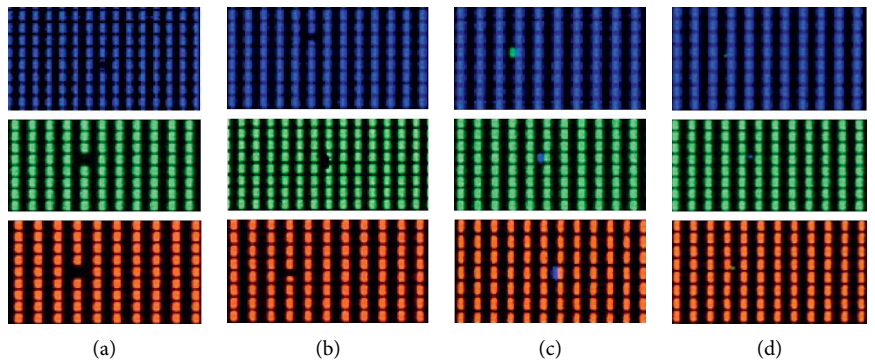

Figure 7: Examples of 12 types of defective pixels.

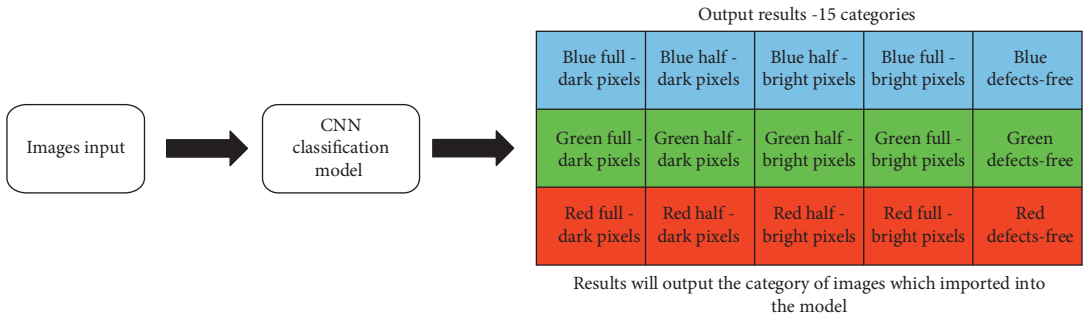

Figure 8: Multicategory classification model concept.

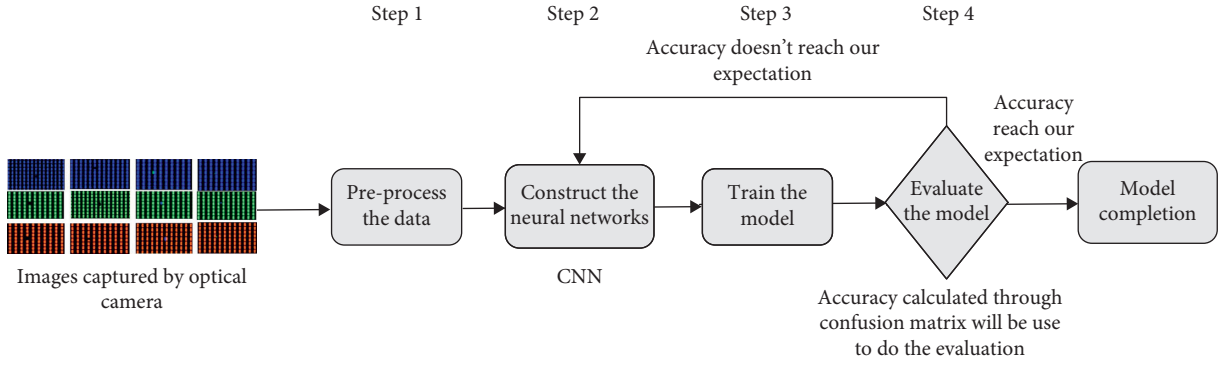

Figure 9: Model construction process.

Table 1: Image label codes.

Label	0	1	2	3	4
Actual category	Blue full-dark pixels	Blue half-dark pixels	Blue half-bright pixels	Blue full-bright pixels	Blue defects-free
Label	5	6	7	8	9
Actual category	Green full-dark pixels	Green half-dark pixels	Green half-bright pixels	Green full-bright pixels	Green defects-free
Label	10	11	12	13	14
Actual category	Red full-dark pixels	Red half-dark pixels	Red half-bright pixels	Red full-bright pixels	Red defects-free

At last, the flatten layers are established and used to transform all the data acquired through the networks into a one-dimensional vector. The final output demonstrates the classification of defects through the softmax function.

3.3. Step 3: Train the Model. This study divided the data into training data, verification data, and testing data. The training and verification data were used to build the model, and the testing data were used to verify the classification ability of the model. The choice of the optimizer is an important part of

model training and can be used to calculate the variety of gradients. This study used Adam as the main optimizer. Xiao et al. [50] showed that it can have a positive impact in preventing overfitting and is applicable to most classification problems.

3.4. Step 4: Evaluate the Model. In the last step, the model was tested on classifying the testing data. Because this part of data was not included model training, the model has never seen it before testing, making it useful to create the baseline for the accuracy of the model. For each of the images that were classified as defective, the Canny edge detection technique was applied to mark the position of the defects.

This paper used the confusion matrix to evaluate the classification results yielded by the model. Unlike the traditional confusion matrix approach which is only used in cases with two categories, this study intended to output a confusion matrix with fifteen categories, which got differences in the definition of positive and negative categories. The positive category in this study referred to the category users intend to evaluate; the negative category, on the other hand, referred to any categories not being evaluated by users. As shown in Table 2, if this study wanted to get the result for category 0, the positive category would be category 0, and the negative category would be the set of all other categories. The sensitivity, specificity, and total prediction accuracy (TPA) are evaluating measures generated by the confusion matrix and are important in identifying misjudged outputs from the model. These measures are shown in equations (1) to (3).

Specificity is the rate of how many images with negative labels are correctly classified into the negative category, as shown in the following:

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}. \tag{1}$$

Sensitivity is the rate of how many images with positive labels are correctly classified into the positive category, as shown in the following:

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{2}$$

TPA calculates the rate of the overall classification correctness, including false negatives and false positives, as shown in the following:

$$\text{Total prediction accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{3}$$

4. Experiment and Analysis

This paper used data captured from TFT-LCD 15.6-inch panels provided by a brand-name notebook manufacturer in Taiwan to test the efficiency and effectiveness of the proposed method. Each original TFT-LCD panel size was 15.6 inches and had a brightness of 300 cd, a frequency of 60 Hz, and a resolution of 1920 ∗ 1080-FHD. Using a high-resolution industrial camera, limited by the field of view (FOV), each 15.6-inch panel was captured 294 times and yielded 294 images with the size of 3684 ∗ 4912 pixels (18.10 Mpix). This study constructed the model using Python 3.6 as the code language and used the Anaconda compiler to perform model training on a machine. The hardware used included an Intel core i9-9700K@3.60 GHz CPU, 64 GB DDR4-2400 RAM, a GTX-1080 graphic card, and the Windows 10 operating system.

4.1. TFT-LCD Defect Classification Process. The TFT-LCD defect classification process included three steps: data processing, defect classification, and classification completion.

Step 1: data processing

First, because of the limitations of the computing device, this study equally segmented the original TFT-LCD images into 100 pieces before inputting the images into the model. The model then resized the images for further use.

Step 2: defect classification

The model then classified the input images and output the classification result to the confusion matrix. The model then moved images that were classified as having defects for edge detection.

Step 3: classification completion

The model performs Canny edge detection in the last step of the classifying process, during which it marks the positions of the defects on the images. The process is complete after edge detection.

The basic process of defect classification using the deep learning model is shown in Figure 10.

4.2. Experiment Result. Per the experiment settings, 294 images with the size of 3684 ∗ 4912 pixels could be resulted from scanning a 15.6-inch panel. Since the defective pixel is very small, to lighten the computing burden, this study further divided each image into 100 pieces, each of which of size 368 ∗ 491 pixels. Moreover, to cut down the computing complexity more while still preserving important features of various defective pixels, the size of each image was reduced to 140 ∗ 140 pixels. Therefore, one 15.6-inch TFT-LCD panel is represented by 29,400 images. Since the panels under inspection are all Grade-A panels purchased from TFT-LCD manufacturers, at most 10 defective pixels of various types, distributed at most 10 images if existed, are expected to have per ISO/TR 9241 standards. Therefore, it is expected to see that the images containing defective pixels are very rare and thus result in a serious class imbalance problem. In fact, it took the notebook manufacturer several months to complete the collection of images with 12 types of defective pixels.

To deal with the class imbalance problem, this paper randomly rotated images with defective pixels by degrees between 0 and 10 degrees and flipped them to generate more images. For each type of defective pixels, 400 images were prepared, including the original defective images. For 15 categories (the defect-free ones included), 6000 images were prepared for model training. Table 3 shows the details of the

Table 2: Confusion matrix definition.

	Actual value	
Classification value	Positives	Negatives
Positives	TP (true positive)	FP (false positive)
Negatives	FN (false negative)	TN (true negative)

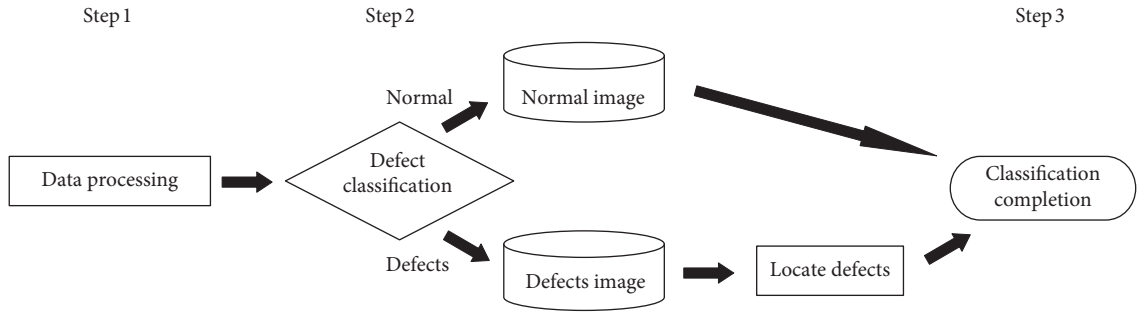

Figure 10: Defect classification process.

Table 3: Details of the materials used in this study—original dataset.

	Image color		Number of images included
Original-dataset	Red	Full-dark defect	400
		Half-dark defect	400
		Half-bright defect	400
		Full-bright defect	400
		Defect-free	400
	Green	Full-dark defect	400
		Half-dark defect	400
		Half-bright defect	400
		Full-bright defect	400
		Defect-free	400
	Blue	Full-dark defect	400
		Half-dark defect	400
		Half-bright defect	400
		Full-bright defect	400
		Defect-free	400

6000 images, named as the original dataset, used for model training and testing.

The original dataset was split into a training dataset and a testing dataset at the ratio of 80% to 20%, resulting in 4800 images in the training dataset and 1200 images in the testing dataset. The labels were split in a likewise fashion. After this study trained the model using the datasets prepared in the previous steps, the result of the model training confusion matrix was as shown in Figure 11. The results of the sensitivity and specificity are shown in Table 4.

In practical use, the specificity, sensitivity, positive predictive value (PPV), and negative predictive value (NPV) are more important than the overall accuracy of the model. If a defect-free image is wrongly classified as a defective image in practical use, it may cause the downgrade of the original TFT-LCD panel. Low sensitivity and specificity may cause the downgrade or the incorrect grading of a TFT-LCD panel and bring significant losses to the company. Therefore, making sure that defect-free images can be correctly classified was the main focus of this study.

As shown in the confusion matrix in Figure 11 and the sensitivity and specificity results shown in Table 5, the model that trained with 6000 images got 77.38% of sensitivity in the blue half-bright category and 76.74% in the blue full-bright category, and many others with sensitivities under 90%, indicating that the result could cause a significant amount of false classification and clearly needed to be improved.

After investigating the misclassified images, it is found that many of them resulted from the image segmentation in which an image of size $3684 * 4912$ was divided into 100 pieces of images with a size of $368 * 491$. During the cutting process, some of the pixels were cut, causing some defect-free images to be classified as defective, or even more, defects were wrongfully classified as another type. Figure 12 demonstrated a sample in which a full-dark blue defect was classified as a half-dark blue defect due to this imaging cutting process.

The model developed by this study was based on deep learning. One of the advantages of deep learning is that preprocessing is not required. Therefore, this study increased the datasets in order to train the model with acceptable sensitivity and specificity.

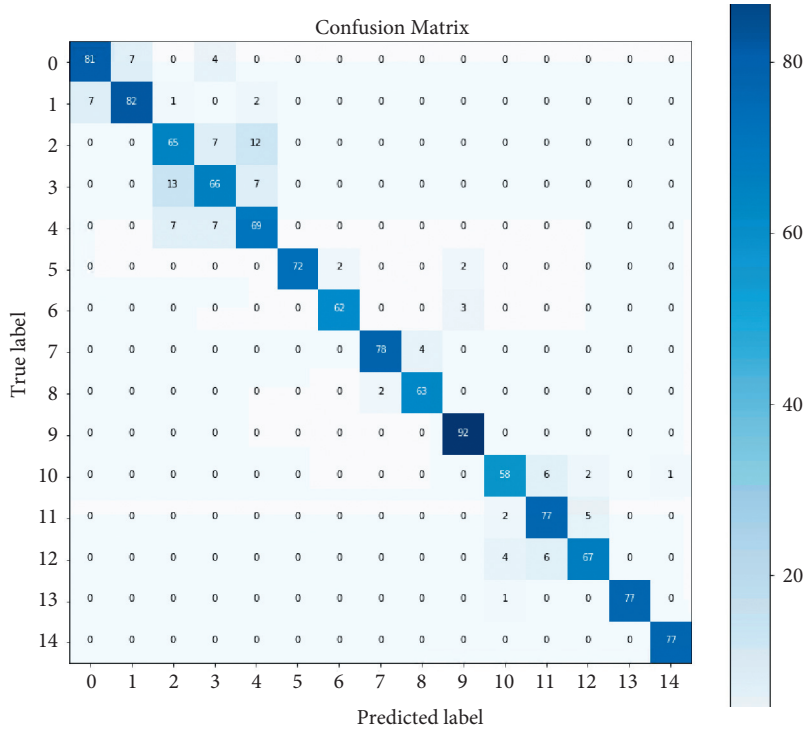

Figure 11: Confusion matrix of the original dataset.

Table 4: Sensitivity and specificity results—original dataset.

Label in the model	Category	Specificity (%)	Sensitivity (%)	TPA (%)
0	Blue full-dark defect	99.37	88.04	
1	Blue half-dark defect	99.37	89.13	
2	Blue half-bright defect	98.12	77.38	
3	Blue full-bright defect	98.38	76.74	
4	Blue defect-free	98.12	83.13	
5	Green full-dark defect	100	94.74	
6	Green half-dark defect	99.82	95.38	
7	Green half-bright defect	99.82	95.12	96.60
8	Green full-bright defect	99.65	96.92	
9	Green defect-free	99.55	100	
10	Red full-dark defect	99.38	86.57	
11	Red half-dark defect	98.92	91.67	
12	Red half-bright defect	99.38	87.01	
13	Red full-bright defect	100	86.52	
14	Red defect-free	99.91	100	

This study extracted images that had been wrongly classified and generated more images based on them to augment the original dataset. The number of misclassified images and the newly augmented images of each defect type were organized in Table 5. To balance the dataset, 400 defect-free images also needed to be added. These newly added images along with the original dataset were combined as a new dataset, called the modified dataset, for further use.

The modified dataset contained 800 images in each category. The dataset was split into the training and testing datasets at the ratio of 80% to 20%, resulting in 9600 images in the training and 2400 images in the testing. The confusion matrix resulted in the testing data by using the model trained by the modified dataset is shown in Figure 13, and the results of the sensitivity and specificity for the modified dataset are shown in Table 6.

The result shown in Table 6 indicated a huge improvement in the sensitivity and specificity, which also indicated that increasing the number of defective images helped the model training significantly. The sensitivity of all categories, as shown in Table 6, were all increased, compared with the ones in Table 4. The lowest value of the sensitivities is 96.97%, which happened for a green half-dark defect. In the defect-free categories, the green and red colors all got 100% of sensitivity, while the blue one got 97.69% in sensitivity, indicating that the classification ability increased significantly. A ten-fold cross-validation resulted in 98.41% TPA.

Table 5: Number of augmented images.

Image color	Defect type	Amounts of misclassified images	Number of images generated
Red	Full-dark defect	11	400
	Half-dark defect	10	400
	Half-bright defect	19	400
	Full-bright defect	20	400
	Defect-free	0	400
Green	Full-dark defect	4	400
	Half-dark defect	3	400
	Half-bright defect	4	400
	Full-bright defect	2	400
	Defect-free	0	400
Blue	Full-dark defect	8	400
	Half-dark defect	7	400
	Half-bright defect	10	400
	Full-bright defect	1	400
	Defect-free	0	400

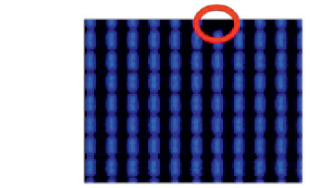

Figure 12: Sabotaged defect sample.

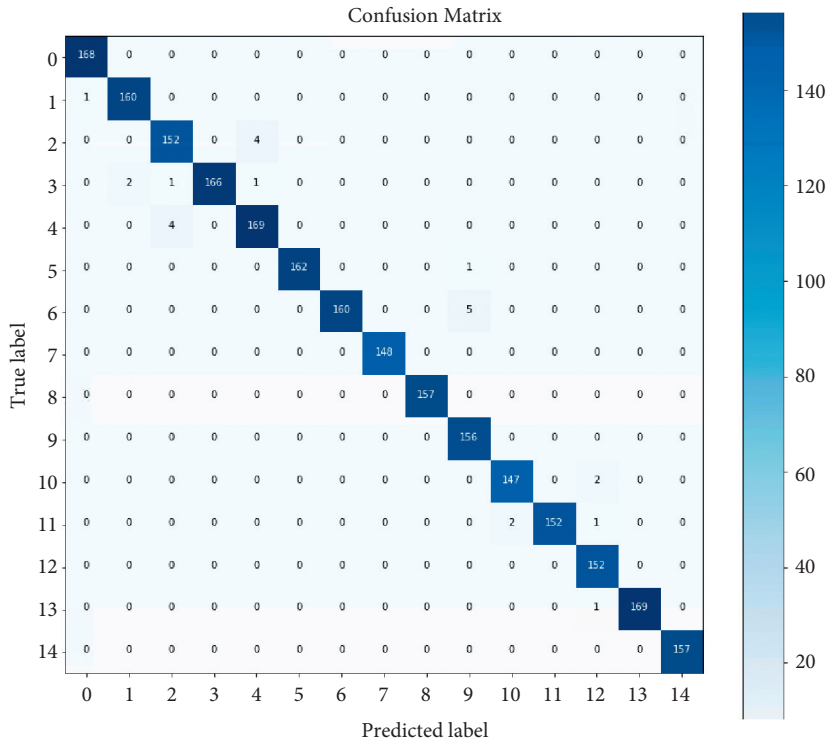

Figure 13: Confusion matrix of the modified dataset.

If all defect categories (categories 0, 1, 2, 3, 5, 6, 7, 8, 10, 11, 12, and 13) were combined together as "defects" category and all defect-free categories (categories 4, 9, and 14) were combined as "defect-free" category, the classification model was reduced to a binary classification one. The values of specificity and sensitivity resulting by using the original and

Table 6: Result of sensitivity and specificity results—modified dataset.

Label in the model	Category	Specificity (%)	Sensitivity (%)	TPA (%)
0	Blue full-dark defect	99.96	100	
1	Blue half-dark defect	99.91	99.38	
2	Blue half-bright defect	99.78	97.44	
3	Blue full-bright defect	100	97.65	
4	Blue defect-free	99.78	97.69	
5	Green full-dark defect	100	99.39	
6	Green half-dark defect	100	96.97	
7	Green half-bright defect	100	100	98.90
8	Green full-bright defect	100	100	
9	Green defect-free	99.73	100	
10	Red full-dark defect	99.91	98.66	
11	Red half-dark defect	100	98.06	
12	Red half-bright defect	99.82	100	
13	Red full-bright defect	100	99.41	
14	Red defect-free	100	100	

Table 7: Comparison between the defect and defect-free categories of the two models.

	Original dataset		Modified dataset	
	Defects (%)	Defect-free (%)	Defects (%)	Defects (%)
Specificity	63.90	97.15	Specificity	63.90
Sensitivity	88.43	94.44	Sensitivity	88.43

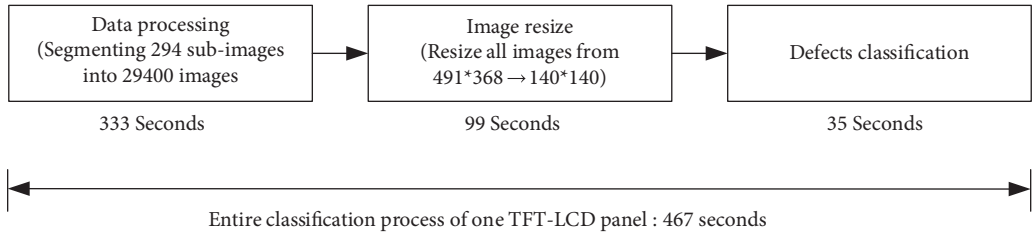

Figure 14: Time related information of the model.

the modified datasets were summarized in Table 7. The numbers shown in Table 7 reflect the ability of the model, which had a sensitivity and specificity of over 95%, indicating that the model could accurately classify images to the correct categories with a low error rate.

This study trained a model that resulted in an accuracy of 98.9% and had high sensitivity, specificity, and TPA. The model also had high efficiency, as shown in Figure 14. It only took 467 seconds to complete the entire classification process of a 15.6-inch TFT-LCD panel. The time could be shortened with the proper design of parallel computing. It is also possible to design the screen sorting process properly to keep up the speed of the assembly line of notebook computers. Compared to the traditional inspection methods based on the human eyes, this study could segment a large number of panel images and inspect them in a short period of time with high accuracy, and then output defect images with the defect locations marked by the Canny edge detection. Images that were wrongly classified could also be input into the model for reclassification. All of the above showed that the model created by this study possessed capabilities that traditional methods lack and could definitely have a positive impact.

5. Conclusions

TFT-LCD panel detection and classification are important issues not only for TFT-LCD panel manufacturers but also for the companies that purchased ready-made panels for further product assembly, such as notebook computer manufacturers. Due to the fact that micodefects, especially defective pixels, were still allowed to exist in top-graded TFT-LCD panels supplied by TFT-LCD manufacturers, sorting and categorizing the panels are inevitable tasks for notebook computer manufacturers. The accuracy of the classification would affect the profit of the notebook computer manufacturer and the market competitiveness of the brand.

This study developed a defect classification model working with AOI by using a convolutional neural network of deep learning approach for identifying defective pixels on TFT-LCD panels to replace manual inspection relied on light-on tests by human eyes. To our knowledge, there were few similar studies that work on subpixel defect identification from the perspective of notebook computer manufacturers, the customers of TFT-LCD manufacturers. By using real images provided by a notebook computer

manufacturer in Taiwan, this study demonstrated that the proposed method could effectively identify various defective pixels under light-on tests so that the TFT-LCD panels could be properly graded to be assembled with different grades of computers. The simulation results showed that the proposed multicategory classification model reached 98.9% total prediction accuracy and had great sensitivity and specificity in the actual panel image detection and classification. Moreover, it took 467 seconds to screen one panel by using the proposed model executing on personal computers. This time could be significantly shortened using parallel computing since each subimage could be inspected independently. The traditional manual screening requires inspectors to look for defects with a size close to human-eye limitation. Using AOI to replace manual inspection is not only a more efficient solution but also a must-go direction.

Data Availability

The TFT-LCD panel detection data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors would like to thank the Ministry of Science and Technology, Taiwan, for financially supporting this research under Contract No. MOST 110-2221-E-A49-120 and MOST 110-2410-H-145-001.

References

[1] K. Nakashima, "Hybrid inspection system for LCD color filter panels," in *Proceedings of the 10th Anniversary (IMTC/94), Advanced Technologies in I & M, 1994 IEEE Instrumentation and Measurement Technology Conference*, pp. 689–692, IEEE, 1994.

[2] T.-Y. Li, J.-Z. Tsai, R.-S. Chang, L.-W. Ho, and C.-F. Yang, "Pretest gap mura on TFT LCDs using the optical interference pattern sensing method and neural network classification," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 9, pp. 3976–3982, 2013.

[3] C. Ngo, Y. J. Park, J. Jung, R. U. Hassan, and J. Seok, "A new algorithm on the automatic TFT-LCD mura defects inspection based on an effective background reconstruction," *Journal of the Society for Information Display*, vol. 25, no. 12, pp. 737–752, 2017.

[4] H. Yang, S. Mei, K. Song, B. Tao, and Z. Yin, "Transfer-learning-based online Mura defect classification," *IEEE Transactions on Semiconductor Manufacturing*, vol. 31, no. 1, pp. 116–123, 2018.

[5] H. Yang, K. Song, S. Mei, and Z. Yin, "An accurate Mura defect vision inspection method using outlier-prejudging-based image background construction and region-gradient-based level set," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 4, pp. 1704–1721, 2018.

[6] S. Mei, H. Yang, and Z. Yin, "Unsupervised-learning-based feature-level fusion method for mura defect recognition," *IEEE Transactions on Semiconductor Manufacturing*, vol. 30, no. 1, pp. 105–113, 2017.

[7] H.-P. Lu and C.-T. Su, "CNNs combined with a conditional GAN for mura defect classification in TFT-LCDs," *IEEE Transactions on Semiconductor Manufacturing*, vol. 34, no. 1, pp. 25–33, 2021.

[8] L. N. Zhang, X. Dang, L. Feng, and J. H. Yang, "Efficient secret image sharing scheme with authentication and cheating prevention," *Mathematical Problems in Engineering*, vol. 2021, Article ID 9274415, 2021.

[9] H. B. Mahmud, V. Katiyar, and M. Nagai, "Improved consistency of an automated multisatellite method for extracting temporal changes in flood extent," *Mathematical Problems in Engineering*, vol. 2021, Article ID 6164161, 2021.

[10] C. A. Curcio, K. R. Sloan, R. E. Kalina, and A. E. Hendrickson, "Human photoreceptor topography," *The Journal of Comparative Neurology*, vol. 292, no. 4, pp. 497–523, 1990.

[11] J. Karangwa, L. Kong, D. Yi, and J. Zheng, "Automatic optical inspection platform for real-time surface defects detection on plane optical components based on semantic segmentation," *Applied Optics*, vol. 60, no. 19, pp. 5496–5506, 2021.

[12] J. M. Runji and C. Y. Lin, "Markerless cooperative augmented reality-based smart manufacturing double-check system: Case of safe PCBA inspection following automatic optical inspection," *Robotics and Computer-Integrated Manufacturing*, vol. 64, Article ID 101957, 2020.

[13] F. M. Tzu, "Effectiveness of electrical and optical detection at pixel circuit on thin-film transistors," *Micromachines*, vol. 12, no. 2, Article ID 135, 2021.

[14] L. Traxler, L. Ginner, S. Breuss, and B. Blaschitz, "Experimental comparison of optical inline 3D measurement and inspection systems," *IEEE Access*, vol. 9, pp. 53952–53963, 2021.

[15] L. Y. Guo, S. N. Li, W. J. Hu et al., "Sub-pixel level defect detection based on notch filter and image registration," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 32, no. 6, Article ID 1854016, 2018.

[16] A. Çelik, A. Küçükmanisa, A. Sümer, A. T. Çelebi, and O. Urhan, "A real-time defective pixel detection system for LCDs using deep learning based object detectors," *Journal of Intelligent Manufacturing*, Early Access, 2020.

[17] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, 2020.

[18] Q. Zhao, T. Sheng, Y. Wang et al., "M2Det: A single-shot object detector based on multi-level feature pyramid network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 9259–9266, Honolulu, HI, USA, January-February 2019.

[19] A. Farhadi and J. Redmon, "Yolov3: An incremental improvement," in *Computer Vision and Pattern Recognition*Vol. 1804, Springer, Berlin, Germany, 2018.

[20] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, pp. 303–314, 1989.

[21] G. Inocente, D. D. Garbuglio, and P. M. Ruas, "Multilayer perceptron applied to genotypes classification in diallel studies," *Scientia Agricola*, vol. 79, no. 3, Article ID e20200365, 2022.

[22] A. Pessoa, G. Sousa, L. M. F. Maues, F. C. Alvarenga, and D. D. Santos, "Cost forecasting of public construction projects using multilayer perceptron artificial neural networks: a case

study," *Ingenieria e Investigacion*, vol. 41, no. 3, Article ID e87737, 2021.

[23] B. T. Pham, D. T. Bui, H. R. Pourghasemi, P. Indra, and M. B. Dholakia, "Landslide susceptibility assesssment in the Uttarakhand area (India) using GIS: a comparison study of prediction capability of naïve bayes, multilayer perceptron neural networks, and functional trees methods," *Theoretical and Applied Climatology*, vol. 128, no. 1-2, pp. 255–273, 2017.

[24] Y. Wang, Z. C. Fang, H. Y. Hong, R. Costache, and X. Z. Tang, "Flood susceptibility mapping by integrating frequency ratio and index of entropy with multilayer perceptron and classification and regression tree," *Journal of Environmental Management*, vol. 289, Article ID 112449, 2021.

[25] H. X. Wang, L. Sui, M. Y. Zhang, F. F. Zhang, F. Y. Ma, and K. Sun, "A novel input variable selection and structure optimization algorithm for multilayer perceptron-based soft Sensors," *Mathematical Problems in Engineering*, vol. 2021, Article ID 5517289, 2021.

[26] M. A. Tanner and W. H. Wong, "The calculation of posterior distributions by data augmentation," *Journal of the American Statistical Association*, vol. 82, no. 398, pp. 528–540, 1987.

[27] X. D. Xiaodong Cui, V. Goel, and B. Kingsbury, "Data augmentation for deep neural network acoustic modeling," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 9, pp. 1469–1477, 2015.

[28] J. Ding, B. Chen, H. W. Liu, and M. Y. Huang, "Convolutional neural network with data augmentation for SAR target recognition," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 3, pp. 364–368, 2019.

[29] Z. Lin, S. Mu, F. Huang et al., "A unified matrix-based convolutional neural network for fine-grained image classification of wheat leaf diseases," *IEEE Access*, vol. 7, pp. 11570–11590, 2019.

[30] H. Shao, H. Jiang, H. Zhang, and T. Liang, "Electric locomotive bearing fault diagnosis using a novel convolutional deep belief network," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 3, pp. 2727–2736, 2018.

[31] G. Liang, H. Hong, W. Xie, and L. Zheng, "Combining convolutional neural network with recursive neural network for blood cell image classification," *IEEE Access*, vol. 6, pp. 36188–36197, 2018.

[32] X. H. Cui, Y. Liu, Y. Zhang, and C. X. Wang, "Tire defects classification with multi-contrast convolutional neural networks," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 32, no. 4, Article ID 1850011, 2018.

[33] K. Isogawa, T. Ida, T. Shiodera, and T. Takeguchi, "Deep shrinkage convolutional neural network for adaptive noise reduction," *IEEE Signal Processing Letters*, vol. 25, no. 2, pp. 224–228, 2018.

[34] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[35] C. L. P. Chen, C.-Y. Zhang, L. Chen, and M. Gan, "Fuzzy restricted Boltzmann machine for the enhancement of deep learning," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 6, pp. 2163–2173, 2015.

[36] L. Huang and L. Wang, "Accelerated Monte Carlo simulations with restricted Boltzmann machines," *Physical Review B*, vol. 95, no. 3, Article ID 035105, 2017.

[37] Y. Liu, J. Luo, and P. Ding, "Inferring microRNA targets based on restricted Boltzmann machines," *IEEE Journal of Biomedical and Health Informatics*, vol. 23, no. 1, pp. 427–436, 2019.

[38] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[39] Z. Zhao, L. Jiao, J. Zhao, J. Gu, and J. Zhao, "Discriminant deep belief network for high-resolution SAR image classification," *Pattern Recognition*, vol. 61, pp. 686–701, 2017.

[40] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[41] F. Rosenblatt, *Principles of Neurodynamics. Perceptrons and the Theory of Brain Mechanisms*, Cornell Aeronautical Lab Inc, Buffalo, NY, USA, 1961.

[42] T.-L. Lin, H.-W. Tseng, Y. Wen, F.-W. Lai, C.-H. Lin, and C.-J. Wang, "Reconstruction algorithm for lost frame of multiview videos in wireless multimedia sensor network based on deep learning multilayer perceptron regression," *IEEE Sensors Journal*, vol. 18, no. 23, pp. 9792–9801, 2018.

[43] Y. LeCun, B. Boser, J. S. Denker et al., "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[44] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *Proceedings of the 2009 IEEE Conference on computer vision and pattern recognition*, pp. 1794–1801, IEEE, Miami, FL, USA, June 2009.

[45] W. Frei and C. C. Chung-Ching Chen, "Fast boundary detection: a generalization and a new algorithm," *IEEE Transactions on Computers*, vol. C-26, no. 10, pp. 988–998, 1977.

[46] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986.

[47] S. Kaur and I. Singh, "Comparison between edge detection techniques," *International Journal of Computer Applications*, vol. 145, no. 15, pp. 15–18, 2016.

[48] J. Lee, H. Tang, and J. Park, "Energy efficient Canny edge detector for advanced mobile vision applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 4, pp. 1037–1046, 2018.

[49] Y. Zhou, H. Li, and L. Kneip, "Canny-VO: Visual odometry with RGB-D Cameras based on geometric 3-D-2-D edge alignment," *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 184–199, 2019.

[50] B. Xiao, Y. Liu, and B. Xiao, "Accurate state-of-charge estimation approach for lithium-ion batteries by gated recurrent unit with ensemble optimizer," *IEEE Access*, vol. 7, pp. 54192–54202, 2019.