

Wafer defect pattern recognition based on machine learning method

Charlie Han, 2020013002

Weiyang-Active 02

Summary: Wafer defect pattern recognition is an important part of producing chips. Based on two ideas, this study built a convolutional neural network to identify defect patterns on wafers, introduced the DropBlock layer, and used cross-validation and grid search methods to confirm the optimal values of hyperparameters. The final recognition accuracy was as high as 95.3%. In addition, this article uses an autoencoder to enhance the input image, which can increase the accuracy by 10.2% compared to traditional image enhancement methods. This paper also proves that convolutional neural networks are better than ordinary neural networks in wafer defect pattern

Key words: recognition problems. Defect pattern recognition; convolutional neural network; autoencoder; machine learning; wafer

1 Introduction

Wafer is an important material for manufacturing chips. It is composed of single crystal silicon and is generally round in shape. The quality of the wafers needs to be inspected before subsequent processing. If there are any defects on the surface of the wafer, the wafer cannot be used to process chips; if the wafer has no defects, circuits can be etched on the wafer to produce chips. The problem solved in this report is how to determine whether the wafer has defects and the corresponding defect types based on the wafer pictures.

1.1 Dataset description

In this report, the data set used is provided by J. Wang, C. Xu et al. [1]. The data set contains a total of 38015 pictures and The corresponding label, the size of each picture is ; If the size is 0, it means that there is no corresponding defect. There are nine basic defect forms, as shown in Figure 1. Its Pattern Type is Single-type, which means that each defect pattern only contains one typical basic defect. Among them, the first type of Normal corresponds to a defect-free situation; the remaining eight types of defects each correspond to a unique Pattern Label, and the non-zero item indicates the defect type.

In addition to the 9 basic defects, each basic defect can also be combined with each other to form a mixed-type defect . Mixed defects included in this data set include: 2 mixed-type, 3 mixed-type, 4 mixed-type. Typical hybrid defects are shown in Figure 2. The Pattern Label of mixed defects marks all included defect types as 1, otherwise it is 0.

In the data set, there are a total of 38 defect types, including 9 basic defects and 29 hybrid defects, and each defect corresponds to about 1000 samples.

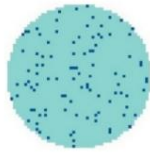


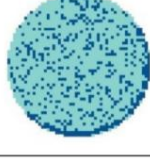
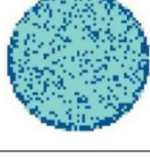
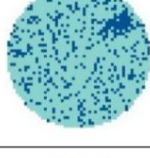
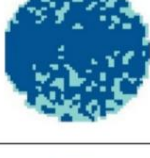
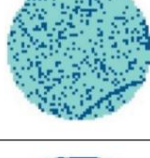

No.	Pattern Type	Pattern Name	Pattern ID	Pattern Label	Illustration	Amount	Index
1	Single-type (9)	Normal	C1	[0 0 0 0 0 0 0]		1000	33866-34865
2		Center(C)	C2	[1 0 0 0 0 0 0]		1000	12000-12999
3		Donut(D)	C3	[0 1 0 0 0 0 0]		1000	24000-24999
4		Edge_Loc(EL)	C4	[0 0 1 0 0 0 0]		1000	25000-25999
5		Edge_Ring(ER)	C5	[0 0 0 1 0 0 0]		1000	26000-26999
6		Loc(L)	C6	[0 0 0 0 1 0 0]		1000	32000-32999
7		Near_Full(NF)	C7	[0 0 0 0 0 1 0]		866	33000-33865
8		Scratch(S)	C8	[0 0 0 0 0 0 1]		1000	37015-38014
9		Random(R)	C9	[0 0 0 0 0 0 0 1]		149	34866-35014

Figure 1 Basic defect types of wafers


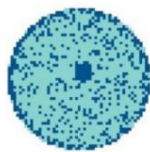







No.	Pattern Type	Pattern Name	Pattern ID	Pattern Label	Illustration	Amount	Index
1	2 mixed-Type	C+EL	C10	[1 0 1 0 0 0 0]		1000	2000-2999
2		C+ER	C11	[1 0 0 1 0 0 0]		1000	4000-4999
3		C+L	C12	[1 0 0 0 1 0 0]		1000	10000-10999
1	3 mixed-Type	C+EL+L	C23	[1 0 1 0 1 0 0]		1000	6000-6999
2		C+EL+S	C24	[1 0 1 0 0 0 1 0]		2000	0-1999
3		C+ER+L	C25	[1 0 0 1 1 0 0 0]		1000	8000-8999
1	4 mixed-Type	C+L+ER+S	C36	[1 0 0 1 1 0 1 0]		1000	7000-7999
2		D+L+EL+S	C37	[0 1 1 0 1 0 1 0]		1000	17000-17999
3		D+L+ER+S	C38	[0 1 0 1 1 0 1 0]		1000	19000-19999

Figure 2 Typical mixed defect types of wafers

2 method introduction

The core idea of this research is to use wafer pictures ($1 \times 52 \times 52$) is used as the model input, the wafer defect label () is used as the model output, and the difference between the predicted label and the actual label is used as the model evaluation criterion. This study first preprocessed the input images, and then used different ideas and methods to build a convolutional neural network model, which will be introduced one by one below.

The process of this development is shown in Figure 3. First, read the data and convert it into an appropriate format; then build and train an autoencoder neural network to enhance the input image; then build a convolutional neural network based on two ideas , and continuously repeat the cycle of training the model, modifying hyperparameters, changing the model structure and optimization methods, and analyzing the results until the optimal network model is found. This study also compared the effects of convolutional neural networks with ordinary artificial neural networks, proving the superiority of convolutional neural networks in handling this problem.

The core innovations of this research are: 1. Using autoencoders for image enhancement, the effect is much better than traditional image enhancement methods ; 2. Different from the research methods of defect pattern recognition, this article proposes two construction methods. The idea of convolutional neural network and compares the advantages and disadvantages of the two. 3. The DropBlock layer is introduced to optimize the traditional convolutional neural network structure. 4. When designing the model structure, the activation function uses the combination of Tanh+Sigmoid+ReLU to try to solve the vanishing gradient problem; 5. Use GPU training to speed up model training; 6. When training the model, use the test set's Loss is used as the end condition of training to avoid overfitting.

In addition, this study was conducted strictly in accordance with the research paradigm, using standardized formats to store preprocessed data, training models, etc., and using the cross-validation method when dividing the training set, test set, and validation set. Refer to Papers in this field define the loss function, general structure and model evaluation criteria of the model.

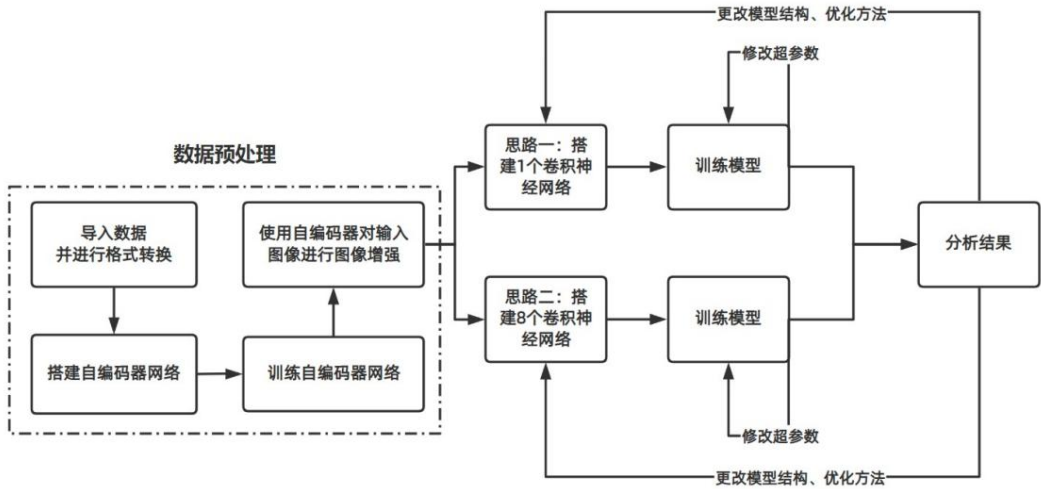


Figure 3 Wafer defect pattern recognition research flow chart

2.1 Data preprocessing

2.1.1 Data reading

First, read and store the data file in .npz format into numpy.array format, and divide it into a picture set and a label set. The size of the picture set is $C \times H \times W$ and the size of the label set is $1 \times H \times W$. In order to process the atlas into the format (C- Channel, H-Height, W-Weight), the shape of the atlas needs to be reset to $38015 \times 1 \times 52 \times 52$.

2.1.2 Image enhancement

By investigating the classic methods of image processing, the author found that image enhancement of input images can very effectively highlight the main features of the image and delete some redundant information, which can significantly improve the performance of network input. The quality of images is very helpful in improving network accuracy. Therefore, the author performed image enhancement on the input image, and compared its corresponding model accuracy with the model accuracy without image enhancement in the "Conclusion" section, proving that image enhancement can be used as a preprocessing method. The image enhancement method used in this study draws on the ideas of Kin et al. [2], that is, using an autoencoder for image enhancement instead of using traditional image enhancement methods. On this basis, pattern filtering is further used on the generated images.

Autoencoder is an unsupervised generation model. The algorithm model contains two main parts: Encoder (encoder) and Decoder (decoder). The function of the Encoder is to encode high-dimensional input into low-dimensional latent variables, thereby forcing the neural network to learn the most informative features; the function of the Decoder is to restore the latent variables of the hidden layer to the initial dimension. What we hope is that the output of the decoder can perfectly or approximately restore the original input, that is

The principle that the autoencoder can perform image enhancement is that the Encoder can capture the most informative features of the image, delete other redundant features, and then the Decoder can restore the original image to the greatest extent. Both Encoder and Decoder use convolutional neural networks here (the principle will be described in detail later). Encoder consists of a convolution layer and a maximum pooling layer; Decoder consists of a deconvolution layer, consisting of an upsampling layer and a deconvolution layer. After trying, the model structure used in this study is shown in Table 1. The training process of the autoencoder is the same as that of the other unsupervised networks, and you only need to input images. The loss function of the autoencoder is designed as:

$$L(\theta) = \frac{1}{N} \sum_{i=1}^N \frac{1}{2} \|y_i - \hat{y}(x_i)\|_2^2 + \frac{\lambda}{2} (\|W\|_F^2 + \|W'\|_F^2) \tag{1}$$

in, $\theta = \{W, b, W', b'\}$, N is the batch size, is the real image, is the predicted image, and are the weight values of the Decoder and Encoder respectively. The first term of the loss function penalizes the distance between the predicted image and the real image, which corresponds to our goal of making the predicted image as close as possible to the real image; the latter term is a regular term to avoid overfitting caused by excessive weight.

Table 1 Convolutional neural network structure of autoencoder

Layer (type)	Output Shape	Param #
InputLayer	(None, 1, 52, 52)	0
Conv2D	(None, 64, 52, 52)	3584
MaxPooling2D	(None, 64, 26, 26)	0
Conv2DTranspose	(None, 64, 26, 26)	73856
UpSampling2D	(None, 64, 52, 52)	0
Conv2DTranspose	(None, 1, 52, 52)	3462

Compared with traditional image enhancement methods (such as rotation, cropping, sharpening, blurring, etc.), the absolute advantage of the autoencoder is that once the model training is completed, it can automatically identify the features in a type of image and add them. Extraction, no manual selection of image enhancement methods is required. For the images generated by the autoencoder, pattern filtering is further used, and the image enhancement effect is shown in Figure 3. It can be seen that through image enhancement, the noise of the original image is effectively eliminated, while its main features are retained and enhanced, which is consistent with theoretical analysis and expected results.

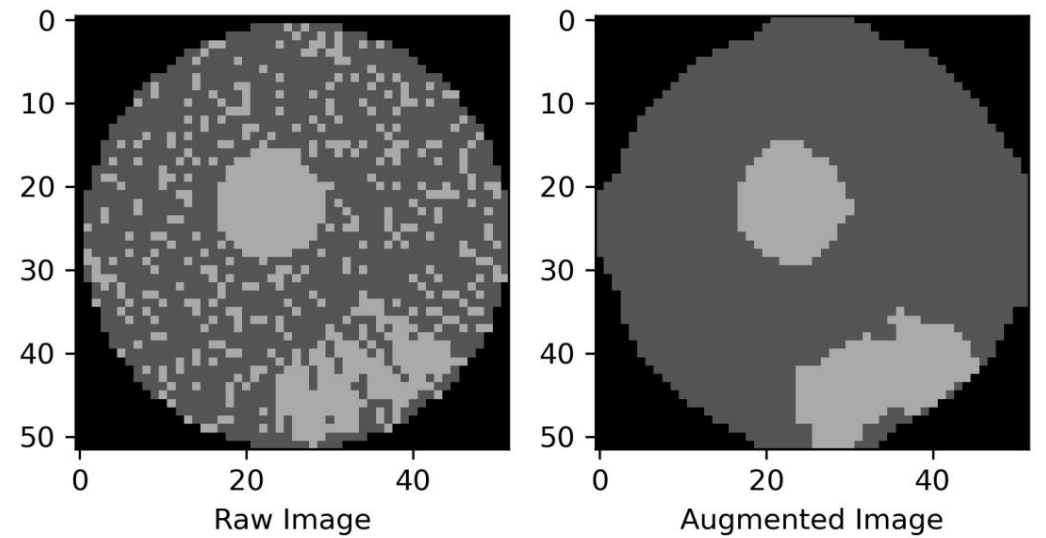


Figure 4 Image enhancement results based on autoencoder and pattern filtering

2.1.3 Training data generation

In order to make the training data richer and make it easier for the model to capture data features, this article rotates and flips the original data to generate more training data. Each picture in the original data set was rotated 90°, 180°, 270° and flipped horizontally and vertically, turning the original one picture into six pictures with different angles while retaining the original feature.

The significance of rotating and flipping the original data is: 1. The newly generated data has the same defect pattern but different angles, which makes it easier for the model to identify defects of the same type but different angles, and it is easier for the model to discover the essence of the defect pattern and reduce the Risk of over-fitting; 2. Expand the data to six times the size of the original data. More sufficient data is expected to achieve better model effects.

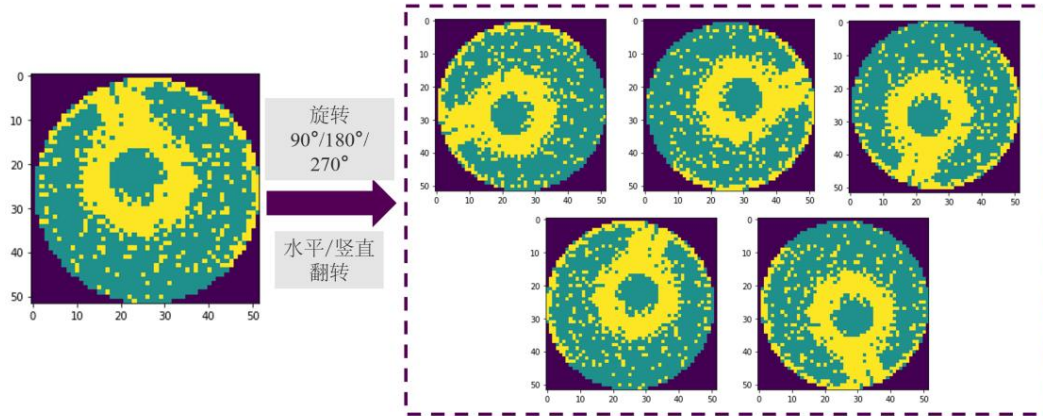


Figure 5 Rotate and flip the original data

2.1.4 Division of training set, validation set and test set

First, all data are randomly shuffled. Use 80% of the original data (30412 items) as the training set and verification set, and use the remaining 20% (7603 items) as the test set. Use 5-fold cross-validation on the first 80% of the data, that is, divide this part of the data into 5 subsets, select one of the sets as the validation set each time, and train on the remaining 4 sets. The pseudo code as follows:

Initialization : $fold \leftarrow \{fold_i | i \in [1, 5]\}, j \leftarrow 1, t \leftarrow total_epoch$

while $j \leq 5$

$validate_set \leftarrow fold_j, train_set \leftarrow fold - fold_j$

$\beta \leftarrow \vec{0}, i \leftarrow 1$

while $i \leq t$

$\beta^{(i+1)} \leftarrow \beta^{(i)} + \alpha \sum_{i=1}^m [y_i - g(\beta^{(i)T} \mathbf{x}_i)] \mathbf{x}_i$ #SGD

$i \leftarrow i + 1$

end

Validate on validate_set

end

The main purpose of 5-fold cross-validation is to determine the values of hyperparameters. The specific method is to calculate the average loss on all validation sets for each set of hyperparameters, and take the smallest set of hyperparameters as the value to be used. The hyperparameters used. Compared with just dividing the training set and the test set, 5-fold cross-validation can largely avoid set division problems caused by uneven data distribution, and avoid learning failures caused by inconsistent data distribution in the test set and training set. Therefore, this study used the 5-fold cross-validation method to generate the training set, validation set and test set.

2.2 Machine learning model construction

Since there are eight defect modes in this problem, and the correlation between different defect modes is very small, they can be approximately regarded as independent (Multiple defects may exist at the same time), so the author believes that there are two ideas in building the model: 1. Build a network and predict 8 labels at the same time, corresponding to the existence of 8 defects; 2. Build The 8 networks are trained separately, and each network is responsible for predicting whether a defect exists. Both ideas are theoretically feasible, so this research plan adopts both ideas and compares the advantages and disadvantages of the two from the perspective of defect identification accuracy.

2.2.1 Convolutional Neural Network (CNN)

Compared with ordinary neural networks, convolutional neural networks (CNN) use convolutional layers instead of linear layers in at least one hidden layer. The nature of the convolution operation makes convolutional neural networks particularly suitable for processing data with grid-like topology. This is an important reason for using convolutional neural networks in this study. Typical convolutional neural networks include convolutional layers, pooling layers, flattening layers and fully connected layers (linear layers).

The so-called convolution layer, in two-dimensional image processing, uses a kernel matrix to perform a convolution operation on the original matrix. The size of the kernel matrix is smaller than the original matrix. The kernel matrix is constantly moving and is a block matrix corresponding to the position of the original matrix. The operation of multiplying the corresponding elements of the matrix results in a scalar as the corresponding element of the new matrix. Its mathematical expression is:

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(i + m, j + n) K(m, n) \quad (2)$$

Where S is the new matrix obtained by the convolution operation, K is the kernel matrix, and I is the original matrix. Apply a matrix I to a

$F_W \times F_H$ Perform a convolution operation on the kernel matrix K. The kernel matrix moves with a step of S_H

For the padding operation, the size of the resulting matrix S is:

$$W' = \lfloor \frac{W - F_W + 2 \times P}{S_W} \rfloor + 1 \quad (3)$$

$$H' = \lfloor \frac{H - F_H + 2 \times P}{S_H} \rfloor + 1 \quad (4)$$

The so - called pooling layer, assuming that the size of the pooling kernel is , that is, for each small block matrix, the maximum value of the $\mathbb{R}^{W \times H} \rightarrow \mathbb{R}$ element is output.

For the flattening layer, the new $C \times H \times D$ The three-dimensional data is converted into one-dimensional data and then input into the fully connected layer. Network updates method is still backpropagation.

2.2.2 DropBlock layer

This article uses the DropBlock layer to replace the Dropout layer in the convolutional neural network. The DropBlock layer was proposed by Google in 2018 [4]. Its main purpose is to solve the over-fitting problem caused by the traditional Dropout layer in the convolutional neural network. DropBlock is a structural form of Dropout that is very effective in regularizing convolutional networks. DropBlock puts the Features in a Block , that is , an adjacent area in the Feature Map, together for drop . Dropout is widely used as a regularization technique for fully connected layers, but it is usually not very effective for convolutional layers. The reason is that adjacent position elements in the feature map of the convolutional layer share semantic information in space, so although a certain A unit is dropped , but its adjacent elements can still retain the semantic information of that position, and the information can still circulate in the convolutional network. Therefore, for convolutional networks, we need a structural form of DropBlock to regularize, that is, drop by block. The visual representation of DropBlock is as shown below:

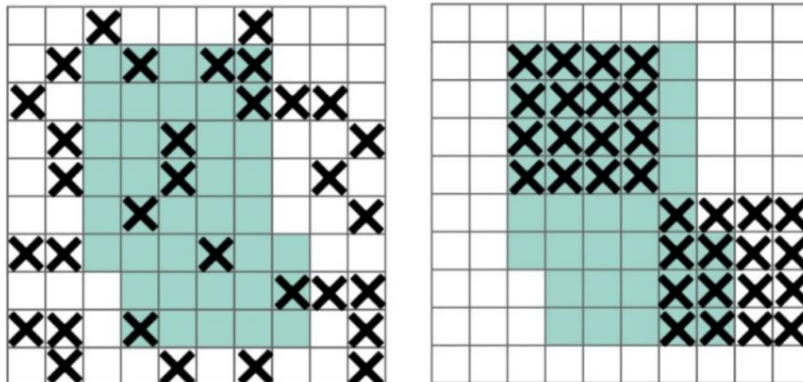


Figure 6 Comparison of the effects of Dropout layer (left) and DropBlock layer (right)

Since the complete mathematical expression and design of the DropBlock layer is relatively complex, this article will not elaborate on it.

2.2.3 Network design

Through research, the author believes that among the most popular convolutional network structures currently on the market, considering the image size and data volume in this study, the model framework of the VGG network is more suitable [3], so it is VGG builds a convolutional neural network for the framework. Although the network is very complex and has a huge amount of parameters, based on the mathematical foundation of deep learning and examples from previous ImageNet competitions, it can be seen that even if the number of parameters is larger than the amount of data, the network can still be well trained and perform well. . Through continuous attempts, this study finally selected three network structures, as follows:

```
(VGG1): Sequential( (0):
  Conv2d(1, 32, kernel_size=(6, 6), stride=(1, 1), padding=(3, 3))
  (1): ReLU(inplace=True)
  (2): Conv2d(32, 32, kernel_size=(6, 6), stride=(1, 1), padding=(3, 3))
  (3): Tanh(inplace=True)
  (4): MaxPool2d(kernel_size=5, stride=2, padding=0, dilation=1, ceil_mode=False)
  (5): DropBlock2D(block_size=7, keep_prob=0.7)
  (6): Conv2d(32, 16, kernel_size=(6, 6), stride=(1, 1), padding=(3, 3))
  (7): ReLU(inplace=True)
  (8): Conv2d(16, 16, kernel_size=(6, 6), stride=(1, 1), padding=(3, 3))
  (9): Tanh(inplace=True)
  (10): MaxPool2d(kernel_size=5, stride=2, padding=0, dilation=1, ceil_mode=False)
  (11): DropBlock2D(block_size=5, keep_prob=0.7)
  (12): Conv2d(16, 16, kernel_size=(6, 6), stride=(1, 1), padding=(3, 3))
  (13): ReLU(inplace=True)
  (14): Conv2d(16, 16, kernel_size=(6, 6), stride=(1, 1), padding=(3, 3))
  (15): Tanh(inplace=True)
  (16): MaxPool2d(kernel_size=5, stride=2, padding=0, dilation=1, ceil_mode=False)
  (17): LazyBatchNorm2d(0, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
)
(VGG2): Sequential(
  (0): Conv2d(32, 32, kernel_size=(4, 4), stride=(1, 1), padding=(1, 1))
  (1): ReLU(inplace=True)
  (2): Conv2d(32, 32, kernel_size=(4, 4), stride=(1, 1), padding=(1, 1))
  (3): Sigmoid(inplace=True)
  (4): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  (5): Conv2d(32, 3, kernel_size=(4, 4), stride=(1, 1), padding=(1, 1))
  (6): ReLU(inplace=True)
  (7): Conv2d(3, 3, kernel_size=(4, 4), stride=(1, 1), padding=(1, 1))
  (8): Sigmoid(inplace=True)
  (9): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
)
(fc1): Linear(in_features=400, out_features=300, bias=True) (fc2):
Linear(in_features=300, out_features=200, bias=True) (fc3): Linear(in_features=300,
out_features=8, bias=True) (droupout): Dropout(p=0.5, inplace=False) (fla):
Flatten(start_dim=1, end_dim=-1) (sig): Sigmoid()
```

The final output of the network is a 8×1 vector, corresponding to the eight-dimensional defect label. Each component has been passed through Sigmoid()

The function is converted into a real number within [0,1], which can be considered as a probability: if the component is greater than 0.5, it is considered that the defect mode exists,

that is, the corresponding component of the predicted label is 1; if the component is less than or equal to 0.5, it is considered that This defect mode does not exist, that is, the

corresponding component of the predicted label is 0. On this basis, the loss function of the network uses Hamming distance, which is defined as:

$$L(y, \hat{y}) = \frac{1}{8} \sum_{i=1}^8 I(y_i \neq \hat{y}_i) \quad (5)$$

SGD+Momentum is used to update network parameters. Its parameter update method is:

$$w_k = w_{k-1} - \alpha v_{w,k-1} \quad (6)$$

$$b_k = b_{k-1} - \alpha v_{b,k-1} \quad (7)$$

in,

$$v_{w,k} = \sum_{i=0}^{k-1} \beta^i w^i \quad (8)$$

$$v_{b,k} = \sum_{i=0}^{k-1} \beta^i b^i \quad (9)$$

This algorithm is equivalent to introducing the previous weight into the parameter update of the current round in the form of exponential decay, accumulating "dynamic Quantity", which is also the origin of the name of its algorithm. The larger the hyperparameter, the greater the momentum.

Through Grid Search and continuous cross-validation, the learning rate was finally determined and decayed at a rate in the 50th, 100th, and 150th rounds of training (i.e.), the hyperparameters in the Momentum algorithm $\beta = 0.95$.

2.3 Model evaluation

To evaluate the accuracy of the model, this study mainly uses Hamming distance and accuracy. Hamming distance has been defined in equation (5). There are different accuracy definitions for different model building methods. For idea 1 (that is, building only one network and predicting 8 labels), the accuracy is defined as:

$$A_1 = \frac{\sum_n I_n(\hat{y} = y)}{N} \quad (10)$$

$$I_n(\hat{y} = y) = \begin{cases} 1, & y_i = \hat{y}_i \text{ for } i \in [1, 8] \\ 0, & \text{otherwise} \end{cases} \quad (11)$$

For idea 2, its accuracy is defined as:

$$A_2 = \frac{\sum_n I_n'(\hat{y} = y)}{N} \quad (12)$$

$$I_n'(\hat{y} = y) = \begin{cases} 1, & y = \hat{y} \\ 0, & \text{otherwise} \end{cases} \quad (13)$$

Assuming that the existence of each defect is independent of each other, and for the second idea, the prediction accuracy of each network for its corresponding defect is equal, then there should be:

$$A_1 = (A_2)^8 \quad (14)$$

In actual situations, the above formula may not be established exactly, but it should be correct in order of magnitude. When the accuracy of idea 2 can reach a higher level of 85%, the accuracy of idea 1 can only reach 49.8%. Therefore, the accuracy of the first idea may always be low, but this does not mean that there is a problem with the model of the first idea. When testing the network, you can use equation (14) to estimate whether there are problems with the network design.

3 Results and analysis

This study also compared the model accuracy of Idea 1 and Idea 2, and also studied the impact of different hyperparameter settings on model performance within each idea. Since the impact of model structure on model performance is very complex, and the model performs very well under the selected model structure, model structure is not studied here as a variable that affects model performance. This study will compare the model accuracy in three cases: image enhancement using autoencoders, image enhancement using traditional methods, and image enhancement methods not being used, and proves the results proposed in this study. The advantages of image enhancement method based on autoencoder. In addition, this study also compared convolutional neural networks with ordinary neural networks, proving the superiority of convolutional neural networks in wafer defect pattern recognition problems.

In addition, this article uses a GPU to train the model, which greatly speeds up model training. When training the model, early stopping settings are used to avoid overfitting; a 5-fold cross-validation method is used to find the optimal hyperparameters. The use of the above methods has greatly improved the accuracy and training efficiency of the model.

3.1 Convolutional neural network model results of two ideas

3.1.1 Convolutional neural network training process

First, take idea 1 (that is, building a convolutional neural network to predict 8 labels) as an example to illustrate the training process of the network. The loss function of the network is defined by equation (5). In this part, the inputs to the network are all images enhanced by the autoencoder image.

The left picture of Figure 5 shows the change of the network's loss function with the training rounds. It can be seen that the initial loss is very large, but after about 10 rounds of training, it dropped to a smaller range. Since the sample size of the training set is larger, the Loss in each round (epoch) is larger, so the absolute value of Train Loss is greater than the absolute value of Test Loss. Here, the absolute value of Loss is meaningless because it is also affected by the batch size. You only need to pay attention to the changing trend of Loss. Since the Test Loss decreases approximately monotonically, it can be judged that there is no overfitting problem. The condition for terminating training set here is that the change in the magnitude of the two Test Loss is less than 0.05, so the training is terminated early and the preset training rounds of 100 are not reached.

The right picture of Figure 5 shows the training process more intuitively. Among them, the definition of accuracy is given by equations (10) and (11). It can be seen that the accuracy rate and the loss function change almost simultaneously, which is consistent with intuition; however, the fluctuation amplitude of the accuracy rate is greater than that of the loss function. This is because the numerator of the accuracy rate is defined as the number of "correct judgments", which requires 8 labels. Only if everything is correct can it be counted as "correct judgment". It can be seen from the accuracy rate that the model accuracy of the first idea is very high, reaching about 95%. The meaning is: the model can perfectly identify 95% of all defect patterns contained in the data without any external constraints and no prior information (for example, if there is a type A defect, there cannot be a type B defect). This is very difficult. Accuracy based on idea 1. , calculated back according to equation (14), this means that the accuracy of a single model in idea 2 can reach 99.4%!

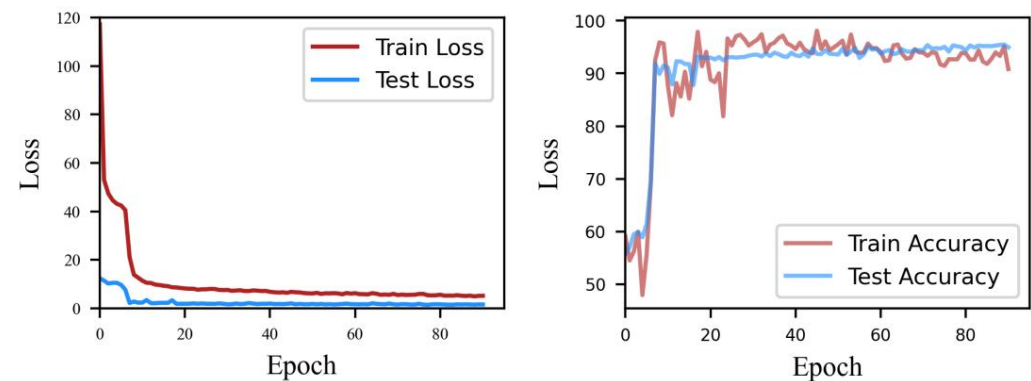


Figure 7 The loss function and accuracy of the convolutional neural network based on idea 1 change with training rounds

The above shows the accuracy and loss of the convolutional neural network with the first idea under the optimal hyperparameters. At this time, $\alpha = 0.015$ the learning rate does not decay at the rate of the 50th, 100th and 150th round of training (i.e., the Momentum algorithm hyperparameters. During the actual training $\beta = 0.95$ process, this set of optimal parameters is continuously found through cross-validation and Grid Search. The following table shows the optimal loss and optimal accuracy values of the model test set for different hyper-parameter values. Table 2 shows the performance of the model in Idea 1 under different hyperparameters. Intuitively, a larger value means that the model training rate is faster, but it means that the model is likely to be unable to accurately reach the optimal point or miss the optimal point, which is consistent with the data in the table. At that time, the model had the largest loss and the lowest accuracy. The larger the value, the greater the momentum of the model, and you can see that the model performed optimally at that time.

Table 2 Loss function and accuracy of convolutional neural network under different hyperparameters

α	β	Loss	Accuracy
0.015	0.95	1.43	95.3%
0.015	0.90	1.50	93.7%
0.015	0.80	1.79	90.3%
0.020	0.95	1.48	94.0%
0.025	0.95	1.85	88.7%
0.010	0.95	1.44	95.2%

The convolutional neural training process of idea 2 is similar to idea 1 and will not be described in detail below.

3.1.2 Comparison of model results between Idea 1 and Idea 2

The 8 model structures of Idea 2 are similar to Idea 1, but simpler:

```
(VGG1): Sequential(
  (0): Conv2d(1, 32, kernel_size=(6, 6), stride=(1, 1), padding=(3, 3))
  (1): ReLU(inplace=True)
  (2): Conv2d(32, 32, kernel_size=(6, 6), stride=(1, 1), padding=(3, 3))
  (3): ReLU(inplace=True)
  (4): MaxPool2d(kernel_size=5, stride=2, padding=0, dilation=1, ceil_mode=False)
  (5): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
)

(VGG2): Sequential(
  (0): Conv2d(32, 32, kernel_size=(4, 4), stride=(1, 1), padding=(1, 1))
  (1): ReLU(inplace=True)
  (2): Conv2d(32, 32, kernel_size=(4, 4), stride=(1, 1), padding=(1, 1))
  (3): ReLU(inplace=True)
  (4): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  (5): Conv2d(32, 3, kernel_size=(4, 4), stride=(1, 1), padding=(1, 1))
  (6): ReLU(inplace=True)
  (7): Conv2d(3, 3, kernel_size=(4, 4), stride=(1, 1), padding=(1, 1))
  (8): ReLU(inplace=True)
  (9): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
)

(fc1): Linear(in_features=20000, out_features=300, bias=True) (fc2):
Linear(in_features=300, out_features=200, bias=True) (fc3): Linear(in_features=300,
out_features=8, bias=True) (droupout): Dropout(p=0.5, inplace=False) (fla):
Flatten(start_dim=1, end_dim=-1) (sig): Sigmoid()
```

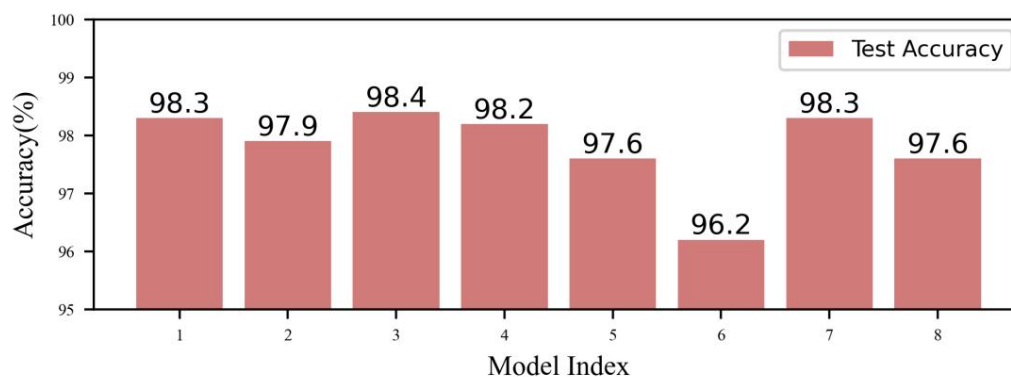


Figure 8 Accuracy rate of convolutional neural network based on idea 2

The model accuracy of idea 2 is shown in Figure 6. The highest model accuracy is 98.4%, the lowest is 96.2%, and the level difference is 2.2%, which are all higher than the accuracy of the model in Idea 1. According to equation (14), assuming that each defect is independent of each other, the maximum accuracy that can be achieved by the theoretical model can be calculated :

$$A_1^* = 83.8\% \tag{15}$$

However, the actual accuracy of the model based on the first idea is:

$$A_1 = 95.3\% \tag{16}$$

Trying to analyze this situation, the author believes that it mainly shows that the existence of each defect is not independent. The deep neural network in the first idea has the ability to find the rules between different defects from the input data, so it is accurate. The accuracy is higher than the product of the accuracy of individual models. This also shows that when dealing with multi-label prediction problems, end-to-end deep neural networks should be preferred instead of solving the problem separately.

3.2 Effect of image enhancement

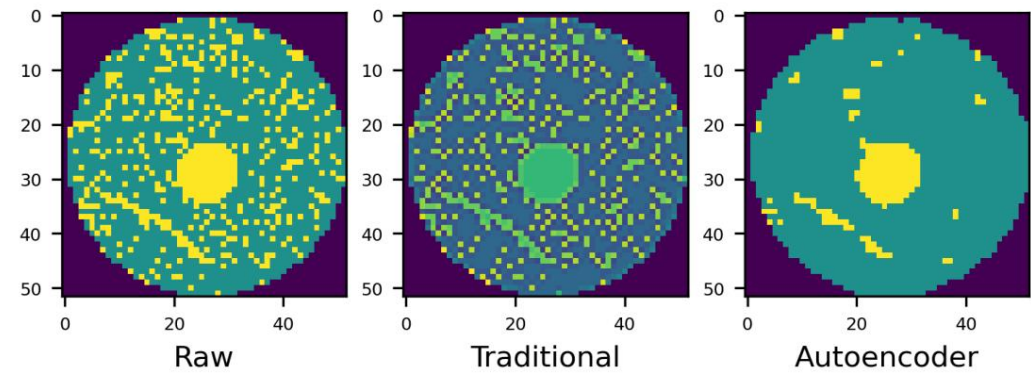


Figure 9 Comparison of the effects of two image enhancement methods

Figure 7 intuitively shows the comparison of the effects of Autoencoder image enhancement and traditional image enhancement methods (pattern filtering + sharpening). It can be seen that image enhancement based on autoencoder can better retain the original features and filter out redundant noise. This can help the convolutional neural network better discover the defect features of the image, which is very meaningful. The defect label corresponding to the example in Figure 7 is [1 0 0 0 0 1 0]. Comparing Figure 1, we can find that it has two defect modes: Center (center defect) and Scratch (scratch defect). This is different from Figure 1. 7 The features retained by the encoder image enhancement in the rightmost picture are exactly the same.

When other conditions are exactly the same, three image enhancement methods, Raw, Traditional, and Autoencoder, are used to process the input image, and the accuracy curves of the three test sets are obtained, as shown in Figure 8. It can be seen that the input corresponding model generated by the Autoencoder method has the highest accuracy, about 10% higher than the Traditional method, and about 13% higher than Raw (that is, no image enhancement). This proves that the Autoencoder method can effectively improve the accuracy of the model.

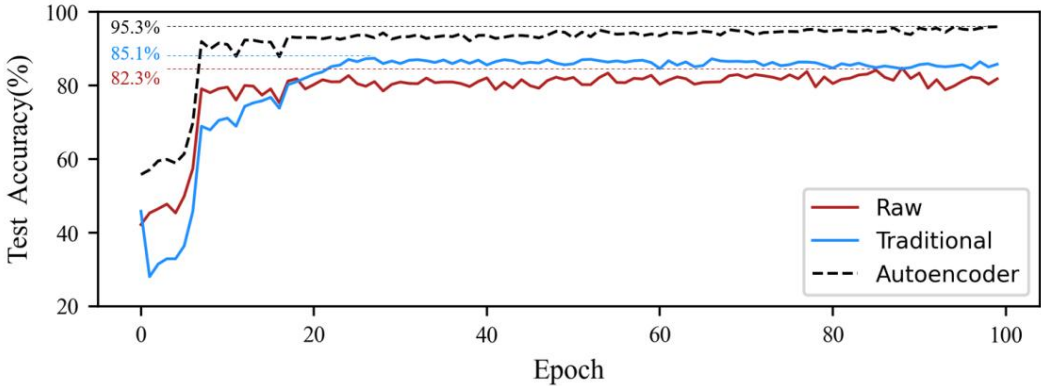


Figure 10 Model accuracy corresponding to input generated by different image enhancement methods

3.3 Effect of DropBlock layer

Replace the DropBlock layer in the original model with the Dropout layer, and control the remaining variables unchanged. The DropBlock layer will be used with The Loss and Accuracy of the convolutional neural network using the Dropout layer are compared. The results are shown in Table 3:

Table 3 Comparison of performance between DropBlock layer and Dropout layer

	Loss	Accuracy
使用DropBlock层	1.43	95.3%
使用Dropout层	1.67	91.8%

It can be seen that the DropBlock layer can indeed improve model performance. Analyzing the reasons, in the problem of wafer defect identification, since the pixels corresponding to each defect are relatively concentrated and the distribution of defects is relatively sparse, discarding the entire area is more promising than randomly discarding pixels. Save the overall characteristics of the defect, which in turn helps the model identify defect characteristics.

3.4 Comparison with ordinary neural network models

This part compares the convolutional neural network model in the first idea with the ordinary neural network model. Control other conditions are not Change, the structure of the ordinary neural network model is as follows:

```
(fc1): Linear(in_features=2704, out_features=256, bias=True) (relu1):
ReLU(inplace=True) (fc2):
Linear(in_features=256, out_features=512, bias=True) (relu2):
ReLU(inplace=True) (fc3):
Linear(in_features=512, out_features=8, bias=True) (droupout):
Dropout(p=0.5, inplace=False) (sig): Sigmoid()
```

The idea of designing an ordinary neural network structure is to ensure that the number of model parameters is the same as the number of parameters of the convolutional neural network model in Idea 1. The training process of ordinary neural networks is exactly the same as that of convolutional neural networks, and the cross-validation method is also used to determine the optimal hyperparameters. Compare the performance of the convolutional neural network model in Idea 1 with that of the ordinary neural network model, as shown in Table 3. It can be seen that the convolutional neural network model is far better than the ordinary neural network, which has an accuracy of only 9.7%. This is mainly because ordinary neural networks are difficult to grasp the topological characteristics of high-dimensional data and are limited to the properties of linear layers. Ordinary neural networks can only flatten high-dimensional data into one-dimensional data as input, so it is difficult to discover Geometric characteristics of this high-dimensional data. In this problem, the wafer defects are precisely characterized by the geometric characteristics of the image pixels. Therefore, ordinary neural networks cannot be used for this problem.

Table 4 Comparison of model performance between convolutional neural network and ordinary neural network

	Loss	Accuracy
卷积神经网络	1.43	95.3%
普通神经网络	12.78	9.7%

4 Conclusion

This study is based on the data set provided by J. Wang, C. Xu et al., using Autoencoder and Convolutional Neural Network (CNN) to identify defect patterns on wafers, and in the specific network The DropBlock layer is used in the structure instead of the Dropout layer, and a combination of Tanh, Sigmoid, and ReLU functions are designed as activation functions to alleviate the vanishing gradient problem; in addition, Hamming Loss is used as the loss function, and SGD+Momentum is used as the optimizer to make The cross-validation method and Grid Search were used to determine the values of the hyperparameters, and during training, whether the loss of the test set was reduced was used as a condition to determine whether the training needed to end early (Early Stopping). The final recognition accuracy was as high as 95.3%. This study uses an autoencoder to enhance the input image, and proves that the image enhancement method based on the autoencoder can greatly improve the accuracy of the model. Compared with the traditional image enhancement method, it can increase by 10.2 % accuracy, compared with no image enhancement, the accuracy can be improved by 13%. This article will compare the use of the DropBlock layer with the convolutional neural network using the Dropout layer, and prove that using the DropBlock layer can improve the recognition accuracy by about 3.5% in this problem. On this basis, this study built a convolutional neural network based on two ideas: building one network to predict eight labels and building eight networks to predict eight labels respectively. Research has proven that the method of building a convolutional neural network is more suitable for dealing with multi-label problems. Although the latter has a higher single-label recognition accuracy, the accuracy of eight networks is

The product of rates is much lower than the accuracy of the convolutional neural network in Idea 1, about 11.5% lower. In addition, this study also compared the model effects of convolutional neural networks and ordinary neural networks, and proved that convolutional neural networks are more suitable for wafer defect pattern recognition problems.

The core innovations of this study are: 1. Using autoencoders for image enhancement, the effect is much better than traditional image enhancement methods ; 2. Different from the research methods of defect pattern recognition, this paper proposes two The idea of building a convolutional neural network and comparing the advantages and disadvantages of the two. 3. The DropBlock layer is introduced to optimize the traditional convolutional neural network structure. 4. When designing the model structure, the activation function uses the combination of Tanh+Sigmoid+ReLU to try to solve the vanishing gradient problem; 5. Use GPU training to speed up model training; 6. When training the model, use the test set's Loss is used as the end condition o

In addition, this study was conducted strictly in accordance with the research paradigm, using standardized formats to store preprocessed data, training models, etc., and using the cross-validation method when dividing the training set, test set, and validation set. Refer to Papers in this field define the loss function, general structure and model evaluation criteria of the model, and compare the model used in this study with ordinary neural network models, making the research more rigorous and comprehensive.

The data sets, training models, and code files of this study are all attached to the folder.

5References _

ŷ1ŷ J. Wang, C. Xu, Z. Yang, J. Zhang and X. Li, "Deformable Convolutional Networks for Efficient Mixed-type Wafer Defect Pattern Recognition," in IEEE Transactions on Semiconductor Manufacturing, DOI: 10.1109/TSM.2020.3020985.

ŷ2ŷ Kin Gwn Lore, Adedotun Akintayo, Soumik Sarkar, "LLNet: A deep autoencoder approach to natural low-light image enhancement", Pattern Recognition, Volume 61,

ŷ3ŷ Karen Simonyan, Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", <https://arxiv.org/abs/1409.1556>

ŷ4ŷ Golnaz Ghiasi, Tsung-Yi Lin, Quoc V. Le, "DropBlock: A regularization method for convolutional networks", <https://arxiv.org/abs/1810.12890>

6 Course Gains and Suggestions

I gained a lot from this course. It is mainly divided into the following three aspects.

One of my gains is that I have expanded my knowledge and made the knowledge system more complete. Before taking this course, I had a rough I have learned some knowledge about machine learning and artificial intelligence, but most of it is fragmented and superficial. By studying this course, I established a blueprint in this field. On the one hand, I deeply understood the connections and boundaries between complex concepts such as artificial intelligence, machine learning, deep learning, and pattern recognition. On the other hand, I established a A knowledge framework has been established to understand the classification and characteristics of various technologies. I think this is crucial for me to get started in the fields of artificial intelligence and machine learning. I am very grateful to have such a course that can help me sort out this huge and complex framework.

My second gain is that I have a clearer understanding of the principles of machine learning. Before, I always thought that machine learning was a very vague and magical method, but through studying this course, I understood the clear mathematical principles behind machine learning methods. After class, I successfully designed a small program to use the gradient descent method to calculate a minimum quadratic problem based on the formulas I learned in class. The sense of accomplishment is incomparable.

My third gain is that by completing two major assignments, my coding ability, data processing ability, and machine learning ability have all improved by leaps and bounds. When completing the mid-term assignment, I went from being a "layman" who couldn't use Python to drawing to successfully completing the visualization assignment through self-study. When completing the final assignment, I learned a lot about convolutional neural networks by myself. It took me nearly ten days to learn the principles and how to use the code to implement it. But in these ten days, My ability to write code and understand machine learning algorithms have been greatly improved. I am very grateful for the two major assignments, which "forced" me to grow so much.

The suggestion for this course is that we can assign some small homework at ordinary times to help us consolidate the knowledge learned in class and at the same time Train your ability to write code. Thank you very much to Mr. Jiang and Mr. Liu for your one-semester teaching. You have suffered so much!