Semi-conductor Image Classification

Zhixian CHEN
Dept of Mathematics
HKUST
Clear Water Bay
Student ID: 20573039
`zchencz@connect.ust.hk`

Shunkang ZHANG
Dept of Mathematics
HKUST
Clear Water Bay
Student ID: 20580484
`szhangcj@connect.ust.hk`

Yueqi QIAN
Dept of Mathematics
HKUST
Clear Water Bay
Student ID: 20669292
`yqianai@connect.ust.hk`

Abstract

In this project, we aim to use the state-of-art deep learning models to do the binary classification task. In the data processing state, we adjust the number of positive and negative samples to achieve a balance dataset. Based on this pre-processing, we further train the fully-connected convolution neural network to classify the bad semi-conductor. In order to get the better optima, we also try to several different stochastic gradient method and search the learning rate to further improve the model performance. What's more, instead of using the accuracy as the measure metric, we also use the F1 score as the supplement metric. Finally, we analysis the experiment result and give the future work.

1 Problem Setting

1.1 Nexperia Kaggle in-class Contest

Nexperia is one of the biggest Semi-conductor company in the world. They will produce billions of semi-conductors every year. But unfortunately, they are facing a hard problem now which is anomaly detection of the semi-conductors. A small number of devices may have defects, e.g. Fig. 1 shows a good example and a bad example. Nexperia accumulated lots of image data using online digital camera and hoped that human based anomaly detection could be greatly improved by the state-of-the-art deep learning technique. So with the data they provide to us, we launch this in-class Kaggle contest in a purpose to test various machine learning methods, esp. deep learning, to solve this real world problem.

1.2 Data Description

The training data contains 3000 bad semi-conductor images and 27000 good semi-conductor images. The labels are 1 for bad ones and 0 for good ones respectively. The dataset is extremely imbalance and we will introduce how to solve this problem in the following section. Due to the online contest, we cannot access to the test dataset directly.

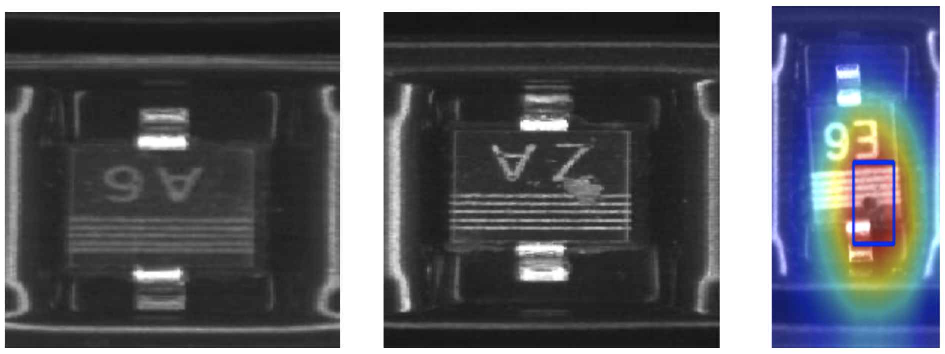


Figure 1: Examples of Nexperia image data. Left: a good example; Middle: a bad example; Right: a visualization based on heatmap

2 Data Preprocessing

In the data pre-processing stage, we mainly focus on how to solve the imbalance dataset problem. If we directly train the deep learning model based on the original dataset, our model will put more attention to the good semi-conductor. In such case, we cannot have a good generalization performance. Under the above consideration, we copy the bad semi-conductor several times to get a balance dataset. We do not resample the dataset, because we do not want to lose any information about the good semi-conductor. What's more, we also do the data augmentation to further increase the model generalization. We rescale the original image, zoom in part of the training images and also do the horizontal flip operand.

3 Performance Metric

Accuracy is not the metric to use when working with an imbalanced dataset. We have seen that it is misleading. There are metrics that have been designed to tell you a more truthful story when working with imbalanced classes. We also use the following metric in our experiment.

- Confusion Matrix: A breakdown of predictions into a table showing correct predictions (the diagonal) and the types of incorrect predictions made (what classes incorrect predictions were assigned).
- Precision: A measure of a classifiers exactness.
- Recall: A measure of a classifiers completeness
- F1 Score (or F-score): A weighted average of precision and recall.

4 Model Architecture and Optimizers

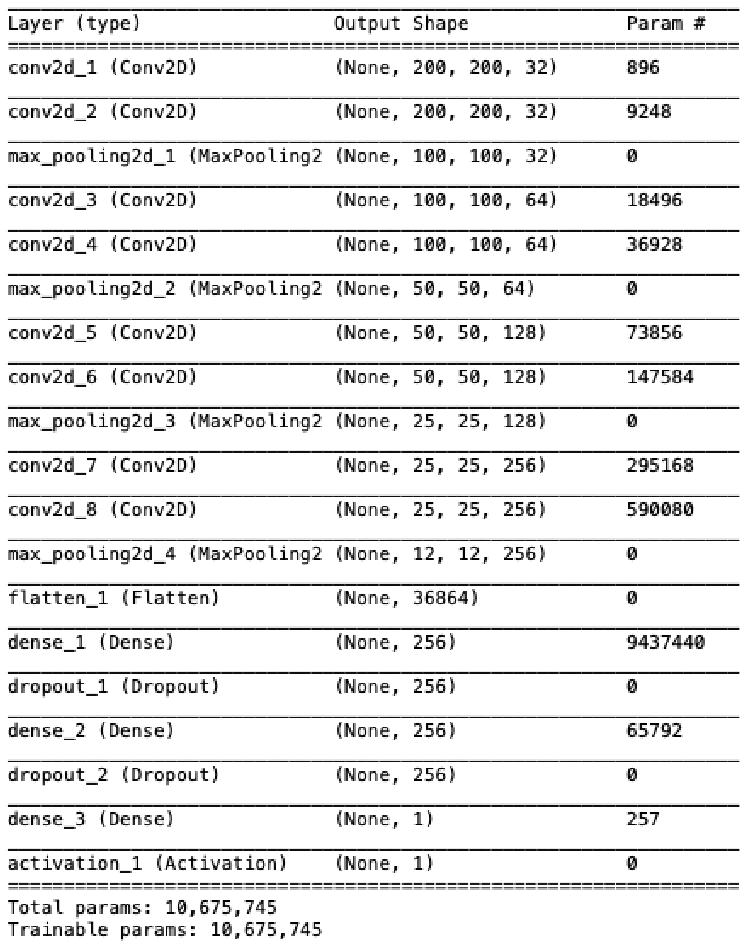
4.1 Model Architecture

We use the fully-connected convolution operation to capture the image feature and use the max-pooling operation to reduce the load of information. In order to get a better generalization performance, we also add dropout operation into our model in case of over fitting. In a word, we build a 15-layer deep neural network and the details of architecture are as the following.

4.2 Optimizers

In order to train a better deep learning model, we also try to different optimizers to see which can achieve a better performance. What's more, we also try to search the learning rate strategy to boost the final performance. In our experiment, we use the following optimizers.

- RMSprop: The central idea of RMSprop is keep the moving average of the squared gradients for each weight. And then we divide the gradient by square root the mean square.
- Adam: Adam can be looked at as a combination of RMSprop and Stochastic Gradient Descent with momentum. It uses the squared gradients to scale the learning rate like RMSprop and it takes advantage of momentum by using moving average of the gradient instead of gradient itself like SGD with momentum.
- Nadam: Nadam is the Adam optimizer with Nesterov momentum.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 200, 200, 32)      896
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 200, 200, 32)      9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 100, 100, 32)      0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 100, 100, 64)      18496
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 100, 100, 64)      36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 50, 50, 64)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 50, 50, 128)       73856
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 50, 50, 128)       147584
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 25, 25, 128)       0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 25, 25, 256)       295168
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 25, 25, 256)       590080
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 12, 12, 256)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 36864)             0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               9437440
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               65792
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 257
_________________________________________________________________
activation_1 (Activation)    (None, 1)                 0
=================================================================
Total params: 10,675,745
Trainable params: 10,675,745
```

Figure 2: Fully-connected convolution neural network structure

5 Experiment Result

According to the above discussion, we train our fully-connected convolution neural network based on different experiment setting and different objective function. Finally, we get the inference result as the following tables.

Table 1: Accuracy of Validation Data

Acc	RMSprop	Adam	Nadam
BCE	96.43	96.03	94.93
MSE	96.00	96.20	95.83

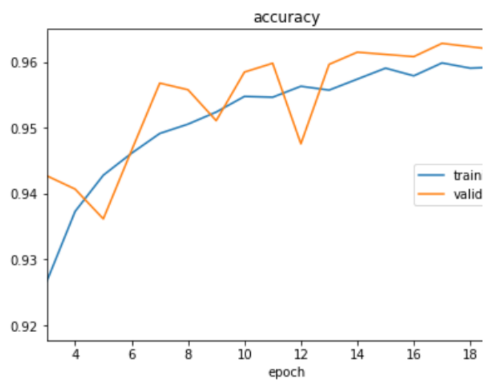


Figure 3: Accuracy During Training Progress

Table 2: F1-Score of Validation Data

F1	RMSprop	Adam	Nadam
BCE	89.10	88.45	86.99
MSE	87.87	88.86	88.38

We train our deep learning model based on three different objective functions: binary cross entropy, binary cross weighted entropy and mean square error. We train our model 20 epoch on both balance and imbalance dataset to do further comparison.

Table 3: AUC Test Result Trained on Imbalanced Data

AUC	RMSprop	Adam	Nadam
BCE	88.24	88.61	85.72
MSE	82.61	89.74	84.19

Table 4: AUC Test Result Trained on Balanced Data

AUC	RMSprop	Adam	Nadam
BCE	88.63	90.54	92.44
MSE	92.00	91.17	91.46

From the above tables, we see that the combination of binary cross entropy and RMSprop optimizer can give us the best inference performance, which can achieve 96.433% accuracy. Compared with the imbalance training dataset, it is easy to see that our simple data adjustment method is very effective.

6 Conclusion and Future Improvement

In this project, we try to solve the imbalance training dataset problem and also try several different combination of data augmentation. We build our own fully-connected convolution neural network deep learning model to do the binary classification task. During the experiment, we also try to use different optimizers and learning rate adjustment strategy to further boost the generalization performance.

There is still room for us to further improve our classification model. For example, in the data pre-processing stage, we simply copy the bad semi-conductor images several times to get a balance dataset. However, in this way, we force our neural network to remember the same bad semi-conductor several times, which actually cause over-fitting problem. We think it is better for us to check if there are some good resampling method which can help us train several deep neural network separately. After that, we can use the ensemble strategy to get the full model.

References

[1] Susto, G. A., Terzi, M., & Beghi, A. (2017). Anomaly detection approaches for semiconductor manufacturing. Procedia Manufacturing, 11, 2018-2024.

[2] Sun, Y., Wong, A. K., & Kamel, M. S. (2009). Classification of imbalanced data: A review. International Journal of Pattern Recognition and Artificial Intelligence, 23(04), 687-719.

[3] Huang, C., Li, Y., Change Loy, C., & Tang, X. (2016). Learning deep representation for imbalanced classification. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 5375-5384).

[4] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[5] Dozat, T. (2016). Incorporating nesterov momentum into adam.

[6] De, S., Mukherjee, A., & Ullah, E. (2018). Convergence guarantees for RMSProp and ADAM in non-convex optimization and an empirical comparison to Nesterov acceleration.

[7] Mannor, S., Peleg, D., & Rubinstein, R. (2005, August). The cross entropy method for classification. In Proceedings of the 22nd international conference on Machine learning (pp. 561-568). ACM.