Solving Semi-Conductor Classification Problem by Light-weighted Model with Stratified Convolutions

FANG Linjiajie, Liu Yiyuan, Wang Qiyue, Wang Ya

Abstract

In this project, we solve the defective semiconductor detection problem using deep neural networks on Nexperia Kaggle dataset.We mainly face three challenges when building models to do the task. The first challenge comes from the extremely imbalanced number of available data between good and defect samples.(roughly 27,000 v.s. 7,000). The second locates in the lacking of powerful machine to solve the large computer vision problem. Since we do not have enough GPU's to train a large model (over 10 M parameters) on the large data set (over 2GB raw data images) based on the present large models, such as ResNet50, VGG etc. , We build our own light-weighted model with far less parameters and train the model using the method of boostrapping to overcome the difficulties of both the low RAM of GPU and the imbalanced quantities of class samples. The third challenge comes from the diverse scales of flaw areas, which motivates us to build a stratified convolution structure to deal with different kinds of defect shapes. Although our model only has number of parameters about 4 times less than the simplest ResNet18, we achieved 0.9983 AUC on the Leaderboard. Since we use fewer parameters and do early stop on cheap laptop training device, our contest score might be higher if we can access more powerful machines. By virtue of the simpleness of our model, it is nature to extend it so that it can also quickly find bounding boxes of defect areas. Even though we do not finish this part in the limited time using our personal machine, we borrow the idea of YOLO(You Look Only Once) to easily establish a model framework to predict bounding boxes. The proposed model details will be given in the codes rather than our reports (since still undergoing).

1 Introduction

Nexperia-one of the biggest Semi-conductor company in the world, is facing a problem about how to effectively detect semi-conductors with accumulated images and hope to improve human based anomaly detection by deep learning methods. Although it is a simple binary classification problem, this task is much more difficult in reality than it seems to be. Firstly, the semiconductor has very different shapes. Figure.1 gives two examples of very different shapes of semiconductors. Additionally, the defect ones have very different kinds of flaws from tiny spots to overall area damage, see Figure.2. We totally have 27,420 good samples and 7,039 defect samples. Each sample image is a gray image of size 267×275. So we need to efficiently train a model with such large amount of data (over 2 GB storage of raw images and over 20 GB storage is required if transformed to the model recognizable tensor format) and avoid the influences of imbalanced data.

To address those difficulties, we design our own light-weighted model based on resnet structure and apply boostrapping to create balanced training set during each training epoch. We also develop a stratified convolution structure to handle different scales of defect areas. This proposal of model architecture is much flexible and we can easily extend it to have the same output form as YOLO

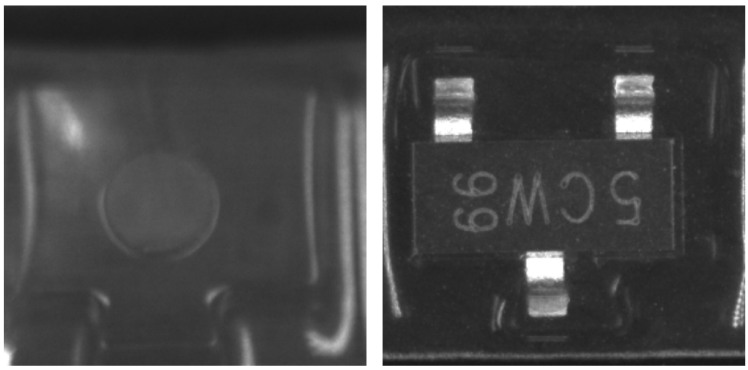

Figure 1: Two very different kinds of semiconductors. The left semiconductor seems to have no metal feet. The right semiconductor have letters on it to show the different models.

(Jposeph etc., CVPR, 2016) so that detecting bounding boxes on flaw areas becomes possible. We give the model structure in the code while move the training part of the YOLO model to the future work.

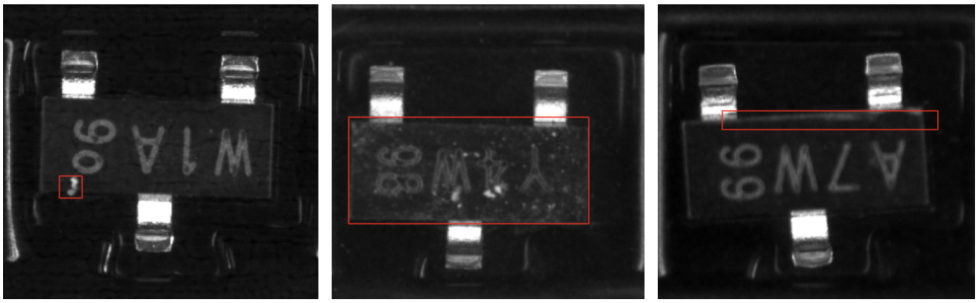

Figure 2: Examples of three different defect semiconductors. The bounding boxes are given by the data file defect_area.csv. The left image gives a example of a tiny flaw on the semiconductor. The middle image gives the example that the whole face of the semiconductor is damaged. The right image shows that the damaged area is a very thin strip.

2 Boostrapping

There are only 7,039 defect sample images in the dataset, compared to 27,420 good sample images, making it a problem to build a model to make binary classification since the model will see good samples much more frequently and make biased prediction. Even a 'blind guess of all good' can achieve 80% accuracy. Ordinary deep learning framework without AUC target can easily make such bias.

To remedy such imbalance of data set, we propose the technique of boostrapping during each training epoch. We randomly sample the same number (with replacement) of defect images and good images and mix them together to form a group of data set with balanced number of defect samples and good samples. In our experiment, we randomly sample N = 5,000 defect ones as well as good ones to form a group of a data set with balanced 10,000 samples. After each epoch, we again apply boostrapping to form another different set of training data. It can be easily seen that many defect samples might be trained by multiple epochs, which helps to balance the deficiencies of the quantities compared to good ones.

We have also tried to sample a smaller or a larger amount N = 2,000 or 7,000. The smaller boostrapping data set is more stable in training the model but converge much slower, the larger training set improves much faster but becomes really sensitive to the learning rate. Even our N = 5,000 setting shows a period of unstable training during the 20-th epoch (Figure .6). The problem is well settled after we apply the technique of learning rate annealing.

As comparison, we also trained another model using data by sampling without replacement. The Leaderboard results shows that it has slightly lower performance (auc=0.99696) compared to the method using boostrapping (auc=0.99779)

3 Stratified Convolution

Figure 2 shows that the damage areas scale from a very different range. In the ordinary setting we use 3×3 convolution filters with stride 1 and padding 1 in all convolution layers, even though such setting can already achieve a good auc performance, it is reasonable to consider how to deal with damaged ares with different scales. Figure 3 gives an example for testing the flaws. Our model can successfully detect larger flaws while fail to detect a minor flaw on the semiconductor.

To solve the problem of detecting different scales of defect areas, we propose a technique called stratified convolution to give the model the power of discriminating flaws with different scales. Figure 4 gives the architecture of the stratified convolutions. Similar to the idea of scattering net, we use four filters with different kernel sizes and concatenate the output results. Then we use a single convolution layer with kernel size 1 to process the concatenated hidden neurons. One drawback of this setting is that the training speed becomes much lower and we need more parameters (3,118,497 parameters) than the ordinary model (2,878,241 parameters). But the number of parameters is still much less than networks such as resnet18 (which requires over 11,177,025 parameters).

In our experiment, we train 100 epochs for stratified convolution model and 140 epochs for the normal model since the training time is much longer to deal with stratified convolution.

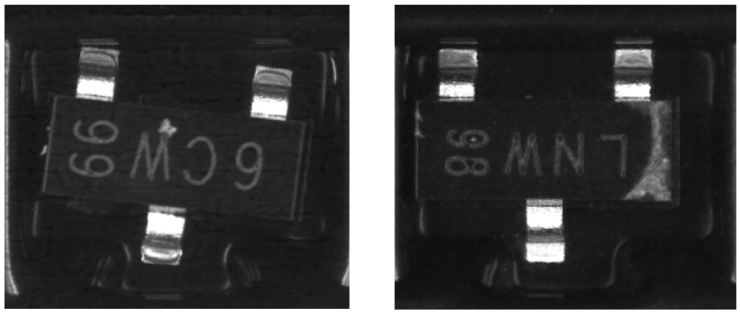

Figure 3: The predicted defect score given by our model with ordinary overall 3×3 convolution filters. Both figures give examples of obvious defect products. But the left semiconductor with minor flaws is failed to be detected by our normal model given a low defect score.

4 Model

4.1 Model architecture

It is believed that CNN is effective in pattern recognition and should be effective for this problem. However, the problem of gradient explosion and vanishing weaken its effectiveness until it is solved by Resnet. By using the 'shortcut connection', the network is learning residual mapping rather than original mapping. Intuitively, defect semiconductors are good semiconductors with some 'flaw residue' on it. Thus the choice of resnet as our basic model module is very nature.

Figure.5 gives the architecture of our model. We can either use normal convolution layer with fewer parameters or use stratified convolutions with more parameters. we limit the maximal channel depth to be 128 to simplify our model. The last two layers are fully connected layers and a scalar output is transformed by the sigmoid function to get the defect score. To make the training more stable. The pixels in the original image data, which are represented by gray scale from 0 to 255, are normalized by 255 to get input features between 0 and 1. As for models requiring RGB channels (such as resnet18), we train it on data by simply duplicating the gray channel by three times.

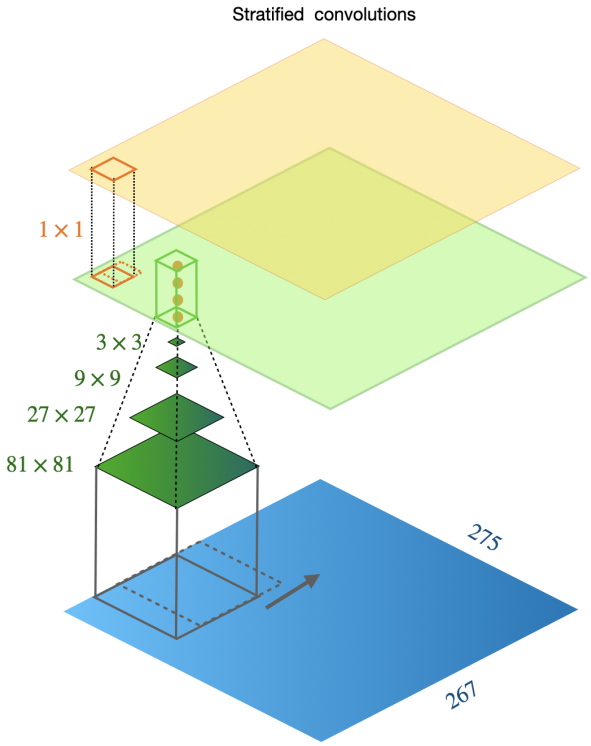

Figure 4: The architecture of the stratified convolution layer. We use 4 different filters with kernel size 3, 9, 27, 81 respectively and followed by one simple convolution layer with kernel size 1 to merge the results.

During each training epoch, we perform boostrapping on both defect and good data to get a balanced training data with 10,000 samples. We do batch training with batch equals to 100. So the training data feeded into the model is of shape $100 \times 1 \times 267 \times 275$.

4.2 Parameters and Simplicity of our model

Different from the traditional models, for example, Resnet18 with 11,177,025 parameters, Resnet50 with 25,557,032 parameters and VGG11 with 132,863,336 parameters, our designed normal model has only parameters 2,878,241, which is of size fraction $\frac{1}{4}$, $\frac{1}{10}$ and $\frac{1}{50}$.The reduction of model parameters increases the simplicity of the model and the efficiency for training on such large data set even on a personal laptop.

Additionally, we are not satisfied with merely classification of the defect ones and good ones. We also wish to build a model to quickly give bounding boxes of where the defect areas are. Inspired by the idea of YOLO, which stands for a simple end-to-end object-detection model, we can easily modify our final two layers to give bounding boxes with (x, y, w, h, C) (coordinates of bounding boxes, confidence score). Due to the limited availability of machines and current main task on Kaggle contest, we give our model of yolo-version in the code file without training. In the future we can train and test our idea based on such data.

4.3 Performance of our original model

We train the model using binary cross entropy loss and Adam optimizer. The learning rate is initiated to be 0.005 and annealed by a factor of 0.5 every 20 epochs. After trained by 140 epochs (with a single GPU with an hour), the normal model achieves nearly 100% training accuracy. The overall performance in training phase is given in Figure.6

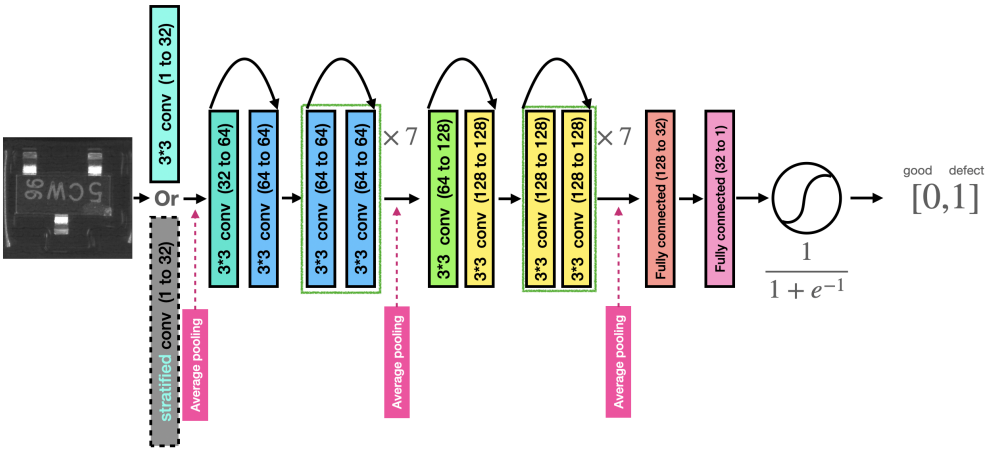

Figure 5: The architecture of our model. Each input image is a tensor of shape $1\times267\times275$. Compared to Resnet18, we simplified the maximal hidden state to have only 128 channels rather than 256 or 512 channels. To reduce the size of the model, we also perform average pooling more frequently, say, we perform average pooling once we transformed to a hidden state with deeper channel. In the end we use two fully connected layer and sigmoid activation to get the defect score. Similar to the project1, we use the hidden vector before the last fully connected layer as the extracted feature vector for further analysing. The first convolution layer can be implemented by stratified convolution or normal convolution.

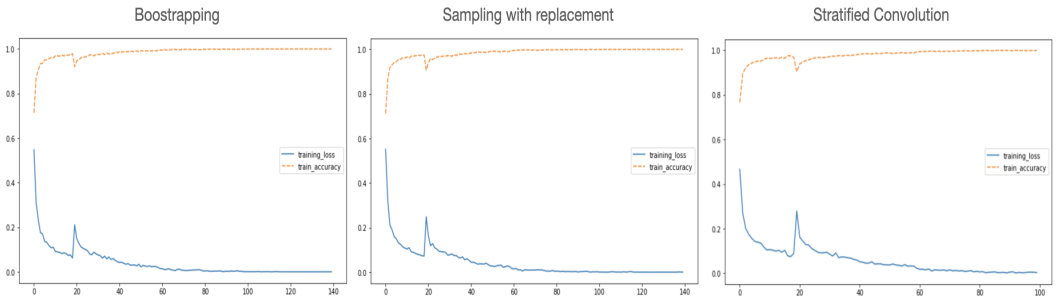

Figure 6: The model performance during the training epochs. The left figure shows the training performance of the normal model with boostrapping technique. The middle figure shows the result of normal model of sampling without replacement. The right figure shows the result of model with stratified convolution structure. In all three models, the loss drops rapidly at the very beginning but becomes really unstable after the 20 epochs. The later drop of the loss benefits from the annealed learning rate. After the 140 epoch, the first two models maintains 100% training accuracy and zero training error (loss). Since the training of stratified convolution model requires far more time, so we do early stop after 100 epochs.

Since our initial learning rate is set to be high enough, the training loss drop rapidly in the first few epochs but abruptly becomes unstable. The second stage of steady improvement appears when learning rate is annealed after 20 epochs. Such control of the learning rate is effective to make the model training more stable.

4.4 Comparison of models

Due to the lacking of powerful machines, we cannot train large deep nets on our large data set in this project by the reason of low RAM or the vast amount of training time. As illustration, we compare models with different sample sizes, with or without replacement, normal convolution or stratified convolution. We also trained Resnet18 with replacing the final fully connected layer (512-dim to 1000-dim) by (512-dim to 1-dim) on our data set. Table1 gives the comparison results.

Table 1: Models comparison. 'Normal' refers to models with normal convolution layers with kernel size 3; 'Stratified' refers to our model replacing the initial convolution layer by stratified convolution layers; 2000, 5000 and 7000 refers to the number of good/defect samples contained in each training epoch; 'B' stands for bootstrapping and 'S' stands for sampling without replacement.

Models	parameters	AUC on Leaderboard
ResNet18-B	11,177,025	0.99587
Normal-2000B	2,878,241	0.99631
Normal-7000B	2,878,241	0.99710
Normal-5000S	2,878,241	0.99696
Normal-5000B	2,878,241	0.99779
Stratified-5000B	3,118,497	**0.99830**

Among the trained models in our experiment, all models achieve very high auc on the Leaderboard. Resnet18 performs the worst compared to other models. Our model equips with stratified convolution structure gives the best performance. To check whether our model solved the flaws scale problem in Figure 3. Figure 7

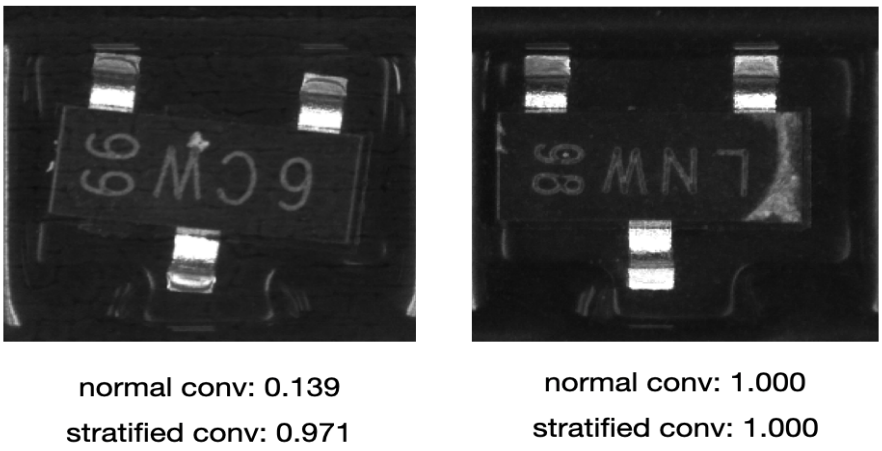

Figure 7: Check whether our stratified model can successfully detect the small flaws. After applying stratified model, we see that the model predicts the defect score to be 0.97 with much higher confidence of defect. These images are manually found from the test set and are far more convicing.

4.5 Activation vectors of Resnet

Similar to the project1, we hope to investigate the feature vectors extracted by our model. For simplicity, we only consider the hidden vector before the last fully connected layer of the normal model as the extracted features, which is of dimension 32.

To visualize the activation vectors, we use both PCA and t-SNE to reduce the dimension from 32 to 2. Figure.8 shows the visualization of 200 activation vectors. Interestingly, the visualization pattern looks almost linear in the two plotting factors.

5 Conclusion

In this project, we successfully build a light-weighted resnet-based model to detect defect semiconductors. To solve the imbalance among the training set and the heavy task of training large image data, we propose the boostrapping technique. Our experiment result shows the efficiency of adopting boostrapping in the training stage. To deal with the diversity of defect areas, we propose stratified convolution structure to give the model the power to distinguish defect areas with different scales. To further detect bounding boxes of defect areas, we propose a YOLO-version of our model in the code file and move the training tasks to the future work.

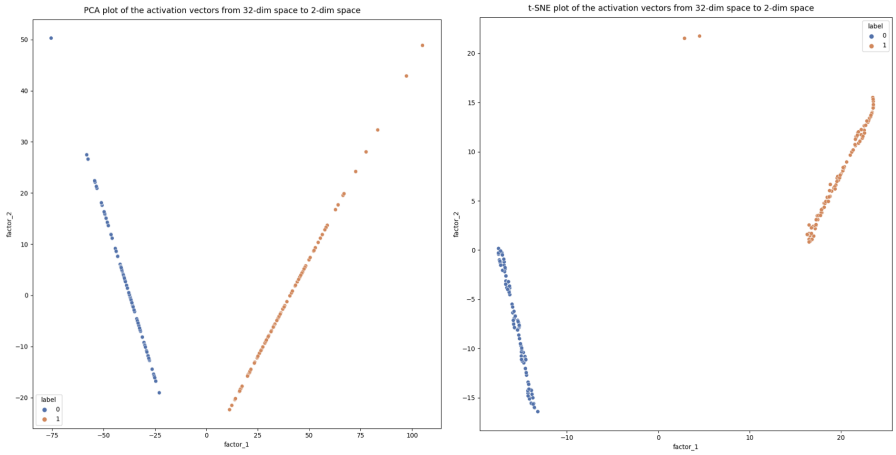

Figure 8: The visualization of the activation vectors extracted by our normal model. We compare both PCA and tSNE methods to visualize the vectors in 2-dim space. Both visualizations shows a linear pattern between two selected visualization components.

6 Contributions

- FANG Linjiajie: Programming and network designing
- Liu Yiyuan: Project presentation
- Wang Qiyue: Visualization of results and traditional methods
- Wang Ya: Outcome analysis and summarising

  All authors contributes to the report-writing.

References

[1] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi CVPR(2016) You Only Look Once: Unified, Real-Time Object Detection

[2] He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing; Sun, Jian CVPR(2016) Deep Residual Learning for Image Recognition