.

CNN Based Multiclass Classification with ensemble of PCB Defects

310706009 歐芷欣 310706036 余冠德

1 Introduction

The electronics industry is at the forefront of industrial progress, yet faces many difficult market pressures. It must innovate – quickly, reliably, and economically – while maintaining the lowest possible error rates

The electronics industry consists of a variety of fields – from chip production to PCB production, complex mounting and inspection processes for computers, smart phones and other mobile devices. Nearly every electronic device contains a PCB. Often, several PCBs are required for the device. During manufacturing, the PCB goes through several process steps before it is added to the electronic device during assembly. Different sensor solutions play a role in this. However, it is not only highly complex 2D or 3D camera solutions, for example, for inspecting SMT components, that are important. As the PCB board goes through various stages of production, such as from a bare board to fully populated with components, various technologies need to be used for detection and positioning. The optimal use of these technologies ensures that a wide range of requirements can be satisfied in order to optimize processes and machines.

The PCB industry is ever-growing and according to [1], the need is for reliable and accurate vision system because PCBs manufactured today has much more component density than earlier and it continues to grow. Study by [2] shows that PCB industry is expected to reach US$ 93.9 billion in 2017. Manufacturers want their products defect free to satisfy their customer needs and remain up par with the competitors. Once a product is manufactured in a plant, it has to go through a quality assurance process to ensure that the product that is being sold to the customer does not have any defects. In this process the product under inspection is tested for defects and flaws. Most manufacturers prefer means of non-destructive inspection system such as Computer Vision for their products quality assurance.

Ever growing PCB industry requires automation during manufacturing process to produce defect free products. Machine Vision is widely used as popular means of inspection to find defects in PCBs. However, it is still largely dependent on user input to select algorithm set for the PCB under inspection prior to the beginning of the process. People will feel fatigued from inspecting the same product for a long time, which will lead to a decrease in the ability to distinguish and increase the chance of defective products being shipped. Continuous increase in computation power of computers and image quality of image acquisition devices demands new methods for further automation.

In the current application, automatic optical inspection (AOI) equipment is commonly used for defect detection, which has the characteristic of "it is better to kill a hundred mistakes than to let one go".

Convolutional neural networks (CNNs) are one of the most common deep learning network architectures because the Convolutional layer and Pooling layer in the network architecture enhance the relationship between pattern recognition and adjacent data, making the application of CNNs to video, audio and other signal types of data types effective. The result is that convolutional neural networks can be applied to video, audio, and other signal types with good results. With the rapid development of computing performance in recent years, CNNs are often used to solve various computer vision tasks, including focal lesion assisted judgment in smart medicine, self-driving cars, etc. The precise judgment ability is one of the state-of-the-art in computer vision.

In view of the many advantages of CNN, we would like to use the powerful image recognition ability of CNN to reduce the error rate of AOI to save the time of human review, and use AI to filter out the inaccurate images of AOI prediction, that is, the part that AOI identifies as defective but is actually not defective, and use AI to assist in defect detection to improve the quality of PCB.

2 Related Work

2.1 MBConv block

Before diving into the architecture of CoAtNet, a brief introduction to CNNs, with this part, originated from studies of the visual cortex of the brain in the 1980s, conducted by two Nobel Prize winners, David Hubel and Torsten Wiesel. These two scientists showed that the local receptive fields of neurons are small, that they only respond to specific areas of the visual field, and that these different areas may overlap. In addition, some neurons respond only to specific patterns (i.e., horizontal lines), while some neurons respond to specific complex patterns that are combinations of lower-level patterns. Inspired by these ideas, in 1998 Yann LeCun introduced the LeNet architecture, which utilized convolutional and pooling layers for the first time. It did not receive much attention at first, but a few years later AlexNet was introduced and CNNs were able to perform well in the ImageNet competition. With the increase in computational power, CNNs have grown even more, and a large part of computer vision research has shifted to CNNs, and several networks have been introduced over the years, such as VGG, ResNet, Inception, EfficientNet, MobileNet, etc.

The basic element of a CNN is the convolutional layer. In the convolution layer, neurons are not connected to every pixel in their input image, but only to pixels in their receptive field. During training, learnable filters or kernels that are convolved on the image are used. Each filter learns to recognize specific patterns, while lower-level filters provide the underlying support for more complex patterns. Given a 224 × 224 × 3 RGB image, a convolution layer with three 3 × 3 × 3 filters is used. This means sliding each of the n filters over the image and performing a convolution operation, as shown in the figure below. The results of the convolution are stacked together to form the $H_{out} \times W_{out} \times n$ output. The width W_{out} and height H_{out} of the output depend on the values of the kernel size and step (the size of the kernel step as it moves across the image) and the padding (how the borders of the image are handled).

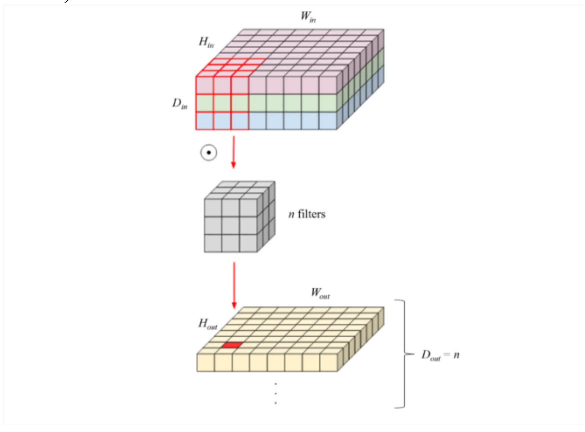


Figure 1: Normal convolution

2.2 Depthwise Separable Convolution

Normal convolution can require a lot of computational effort. For this reason, Google (it's always them) introduced deep separable convolution into their MobileNet architecture. This type of convolution divides the process into two steps: a first deep convolution followed by a point-by-point convolution. The final output has the same size as the output of classical convolution, but requires much less computation due to the 1 × 1 convolution.

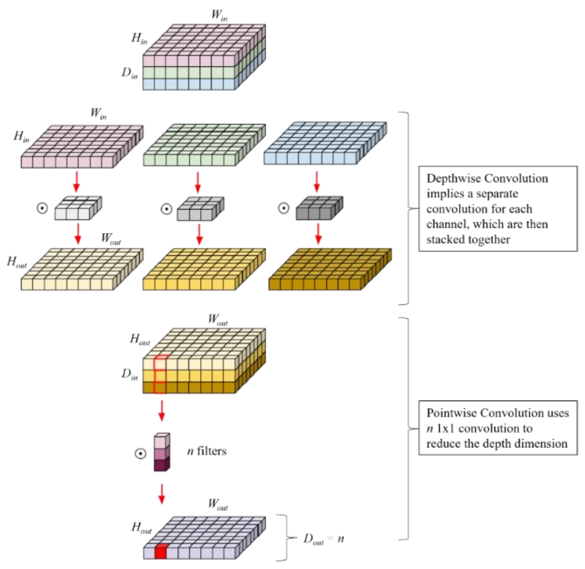

W_{in}
H_{in}
D_{in}
W_{out}
H_{out}
Depthwise Convolution implies a separate convolution for each channel, which are then stacked together
n filters
Pointwise Convolution uses n 1x1 convolution to reduce the depth dimension
$D_{out} = n$

Fig 2. Depth-wise convolution. Filters and image have been broken into three different channels and then convolved separately and stacked thereafter

In the second version of MobileNetv2, the same authors introduced two main ideas about deeply separable convolutions:

Inverted residuals: This technique allows lower layers to access information from the previous layer by skipping connections. The first step expands the input using 1 × 1 convolution, since the subsequent deep convolution has greatly reduced the number of parameters. The later 1 × 1 convolution performs compression to match the initial number of channels. This residual join connects narrow layers instead of the wide layers of the original residual block, hence the name Inverted.

Linear Bottleneck: Using the inverted residual block above, it is possible to compress the layers that skip the connection links to speed up the computation, but this can hurt the performance of the network. Therefore, the authors introduced the idea of a linear bottleneck, where the last convolution of the residual block would contain a linear output before being added to the initial activation.

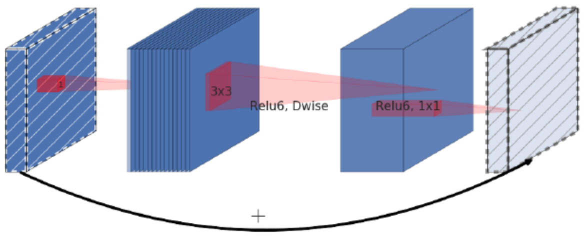


Fig3: An inverted residual block connects narrow layers with a skip connection while layers in between are wide

2.3 Self-attention and Vision Transformer

Transformer was first introduced in the paper Attention Is All You Need [6]. These architectures are based on a self-attentive mechanism for learning relationships between sequence elements.

Specifically, the Encoder architecture of the Transformer is shown below (the decoding part is not covered and is not relevant for the purpose of this paper). In the input embedding the inputs are tokenized, each token is mapped to an integer in the vocabulary, then each integer is mapped to a k-dimensional vector that the model will learn during training, and the next step is to add a positional encoding based on a sine function to handle the word order in the sentence. Finally, the output is obtained by N_x blocks with the addition of multi-headed self-attentive and feedforward networks, residual connectivity and layer normalization layers.

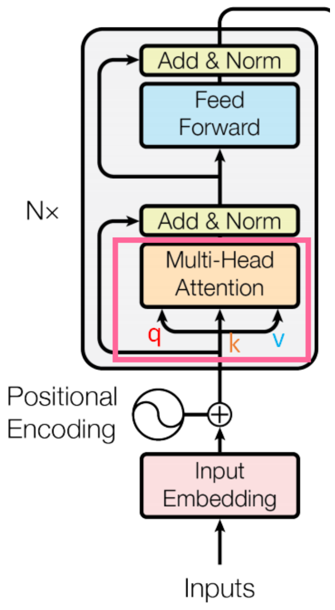


Fig4: Structure of encoder

The multi-headed attention block computes the self-attention multiple times using different weight matrices, and then joins the results together and resizes them to the embedding dimension using another trainable matrix, so that a vector of the same size as the input can be output and it's passed to the next block. Each embedding must have three associated representations, query (Q), key (K) and value (V), which are obtained by multiplying the input by three trainable weight matrices. Intuitively, Q is the representation of the current word used to score all other words (we only care about the query being processed.) K is like a bunch of tags that we match with Q when searching for related words. V contains the actual word representation.

We can also think of the operation of this module as searching through an archive: queries (Q) are like sticky notes for topic research. The key (K) is like a label for a folder. When a match is made the contents of the folder are taken out, and these contents are the value (V) vector.

The scalar product of Q and K is computed to obtain the Score (S) matrix, which represents the degree of relevance of each key to each query, and this matrix is scaled and passed through a row-wise SoftMax function. The scalar product between S and V is computed to obtain the contextual embedding, i.e., the self-attentive matrix. This matrix is passed to the position-wise (meaning it takes one token at a time, but the weights are shared and the different results are stacked in one matrix) FFN after dropout, and residual concatenation and layer normalization, which first scales the input and then compresses it, similar to the MBConv block above (this similarity is also a feature of a feature of CoAtNet) and finally activated by ReLU.

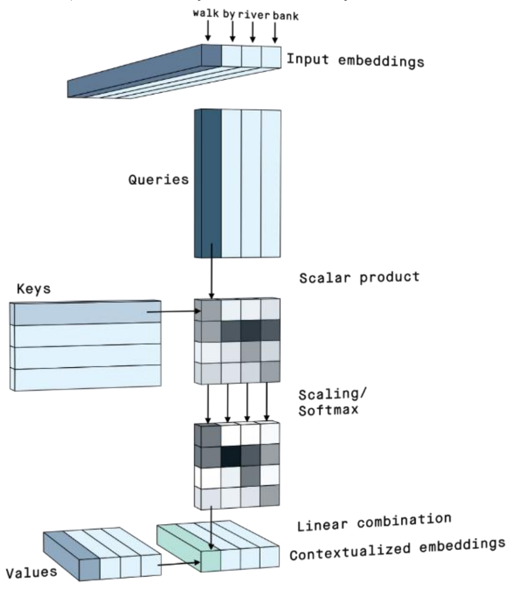


Fig5: Self-attention mechanism

This architecture is characterized by

- Can deal with complete sequences, thus learning long-range relationships
- Can be easily parallelized
- Can scale to high complexity models and large-scale datasets.

That is why the discovery of Transformer networks in the NLP domain has generated a great deal of interest in the computer vision community. However, visual data follow a typical structure, thus requiring new network designs and training schemes. As a result, different authors have proposed their own implementation of a Transformer model applied to vision, but the SOTA has only been achieved by the Vision Transformer (ViT). This architecture focuses on small patches of the image, which are treated as tokens. Each patch in the input image is flattened using a linear projection matrix, and a learnable positional embedding is added to it. This positional embedding is 1-dimensional and considers the input as a sequence of patches in the raster order. In CoAtNet this will be substituted by the relative position between patches instead of their absolute position, leading to the definition of relative self-attention. In order to perform classification, an extra-learnable classification token is added to the sequence (denoted with a * in the image below). The ViT encoder, similarly to the original version discussed above, consists of multiple blocks of self-attention, normalization, and fully connected layers with residual connections. In each attention block, multiple heads can capture different connectivity patterns. The fully connected Multi-Layer Perceptron head at the classification output provides the desired class prediction.

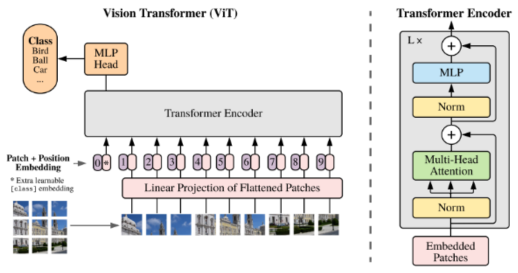


Fig6: Vision Transformer (ViT)

2.4 EfficientNet

Convolutional Neural Networks (ConvNets) are commonly developed at a fixed resource budget, and then scaled up for better accuracy if more resources are available. In this paper, we systematically study model scaling and identify that carefully balancing network depth, width, and resolution can lead to better performance.[3] Therefore, we apply EfficientNet, which is a new scaling method that uniformly scales all dimensions of depth/width/resolution using a simple yet highly effective compound coefficient.

2.5 CoAtNet

CoAtNet, which mixes deep convolution and self-attention and makes new design about vertical layout, is the model we adopt to compare to EfficientNet to figure out the performance of models between both of them.

2.5.1 Mixing Deep Convolution and Self-Attention

One of the main limitations of ViT that emerged in the original paper [4] is its impressive data-hunger. Indeed, while ViT has shown exciting results with the enormous JFT300M dataset, its performance is still inferior to classical CNNs with a low amount of data. This suggests that Transformers might be missing the generalization capability possessed by

CNNs, and thus requires a significant amount of data to compensate. Nevertheless, attention models present a higher model capacity compared to CNNs.

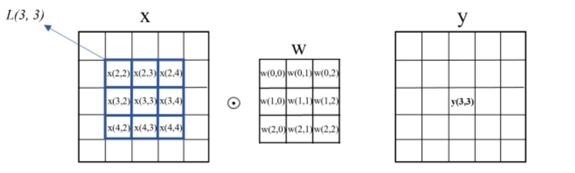
The aim of CoAtNet is therefore to blend the pros of CNNs and Transformers into a single architecture, in terms of 1) generalization and 2) model capacity, but what is the right way to mix CNN and Transformer?

The first idea is to take advantage of the already discussed MBConv block, which uses depthwise convolution with inverted residual bottleneck, an expansion-compression scheme is in common with the FFN module of the Transformer. In addition to this similarity, both depthwise convolution and self-attention can be expressed as a per-dimension weighted sum of values in a pre-defined receptive field. Depthwise convolution can be expressed as:

$$y_i = \sum_{j \in \mathcal{L}(i)} w_{i-j} \odot x_j$$

Fig7: Depthwise convolution

where x_i and y_i are the input and output at position i respectively, w_{i-j} is the weight matrix at position (i - j) and L(i) is a local neighborhood of i. As already seen, depthwise convolution performs convolution for each channel separately. In the illustration below, is shown an example of how to compute y_i with i=(3,3), for one channel, with the above formula:



$$\mathbf{y(3,3)} = x(2,2) * w(0,0) + x(2,3) * w(0,1) + x(2,4) * w(0,2) + x(3,2) * w(1,0) + \\ + x(3,3) * w(1,1) + x(3,4) * w(1,2) + x(4,2) * w(2,0) + x(4,3) * w(2,1) + x(4,4) * w(2,2)$$

Fig8: Visualization of depthwise convolution

In comparison, self-attention allows the receptive field not to be a local neighborhood and computes weights based on pairwise similarity followed by a SoftMax function:

$$y_i = \sum_{j \in \mathcal{G}} \underbrace{\frac{\exp\left(x_i^\top x_j\right)}{\sum_{k \in \mathcal{G}} \exp\left(x_i^\top x_k\right)}}_{A_{i,j}} x_j \quad \text{(self-attention)},$$

Fig9: formulation of Self-attention

where G indicates the global space and x_i, x_j are two pairs (for example two patches of an image). An over-simplified image (omitting the multi-head Q, K, and V projections) just for the sake of understanding is shown below: each patch is compared with every other patch in the same image to produce a self-attention matrix.

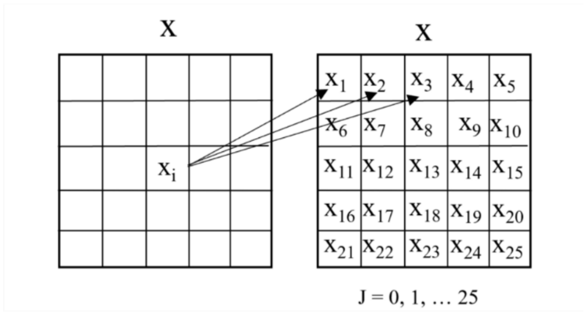


Fig10: matrix of self-attention

Let's define the pros and cons which came to mind after analyzing these two formulas:

Input-Adaptive Weighting: the matrix w_{i-j} is an input-independent static value, while the attention weights A_{ij} depend on the representation of the input. This makes self-attention more prone to capture the relationships between different elements in the input, at the cost of the risk to overfit when data are limited.

Translation Equivariance: the convolution weight w_{i-j} cares about the relative shift between i and j rather than the specific values of i and j. This translation equivariance is proved to improve generalization under a limited size dataset.

Global Receptive Field: the larger receptive field used in self-attention provides more contextual information compared to the local receptive field of CNN.

In summary, an optimal architecture involves the Input-Adaptive Weighting and Global Receptive Field characteristics of self-attention and the Translation Equivariance of CNNs. The idea proposed by the authors is to sum a global static convolution kernel with the adaptive attention matrix, after or before the SoftMax initialization:

$$y_i^{\text{post}} = \sum_{j \in \mathcal{G}} \left(\frac{\exp\left(x_i^\top x_j\right)}{\sum_{k \in \mathcal{G}} \exp\left(x_i^\top x_k\right)} + w_{i-j} \right) x_j \quad \text{or} \quad y_i^{\text{pre}} = \sum_{j \in \mathcal{G}} \frac{\exp\left(x_i^\top x_j + w_{i-j}\right)}{\sum_{k \in \mathcal{G}} \exp\left(x_i^\top x_k + w_{i-k}\right)} x_j$$

Fig11: formulation of the adaptive Self-attention

The pre-normalization version, which was used in the paper, also corresponds to a particular variant of relative self-attention, which we already mentioned before.

2.5.2 Vertical Layout Design

The authors decided to use convolution to perform down-sampling and global relative attention operations only after the size of the feature maps is small enough to be processed. And there are two ways to perform down-sampling:

- Dividing the images into patches as in the ViT model and stack relative self-attention blocks. This model was used as a comparison with the original ViT implementation.
- Using a multi-stage layout with gradual pooling. This approach is divided into 5 stages, but the first two, which are a classic convolutional layer and a MBConv block, used to lower the dimensionality, are here combined in a single one for simplicity, named S0. The last three stages can be either a Convolution or a Transformer block, resulting in 4 combinations: S0-CCC, S0-CCT, S0-CTT and S0-TTT

In the original paper, this resulted in 5 models which were compared in terms of generalization (the gap between training loss and evaluation accuracy) using 1.3M images and model capacity (the ability to fit a large training dataset) using more than 3B images.

Generalization capability: S0-CCC ≈ S0-CCT ≥ S0-CTT > S0-TTT ≫ ViT

Model capacity: S0-CTT ≈ S0-TTT > ViT > S0-CCT > S0-CCC

With regard to generalization, it was found that the more Convolutional layers, the smaller the gap. Concerning model capacity, the results suggest that simply adding more Transformer blocks does not automatically mean better generalization. The S0-CTT, which is shown in the image below, was selected as the best compromise between these two capabilities.

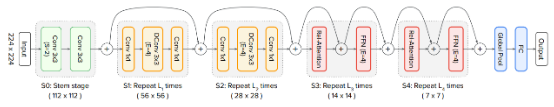


Fig12: CoAtNet by using S0-CTT structure

2.6 PCB Defect Detection

Recently, in 2022, the newest journal paper contributed in PCB defect detection classification is from *Wireless Communications and Mobile Computing*. The paper presents a powerful and precise computer vision model for automated classification of defect product from standard product. Human operators and inspectors without digital aid must spend inordinate amounts of time poring over visual data, especially in high volume production environments. Their model works quickly and accurately in sparing defective product from entering doomed operations that would otherwise incur waste in the form of wasted worker-hours, tardy disposition, and field failure. They use a convolutional neural network (CNN) with the Visual Geometry Group with 16 layers (VGG16) architecture and train it on the Printed Circuit Board (PCB) dataset with 3175 RBG images. The resultant trained model, assisted by finely tuned optimizers and learning rates, classifies defective product with 97.01% validating accuracy.

Extracting circuit information from real-world PCB images and getting high accuracy from classification in PCB defect is a challenge in the electronic manufacturing industry. These images usually contain various components and a lot of scratch by physical composition. Our project utilizes state-of-art machine learning methods cooperating with ensemble methods and classical image processing techniques to build a feasible and reliable analysis system. Our work and contributions are in 3 main parts:

- **Higher accuracy in validation and test set**: As the latest paper of PCB defect classification, we mentioned before, we got higher accuracy with hybrid model that containing two models and three ensemble methods.
- **Ensemble method in the PCB field**: We first proposed the structure of wiring multiple differentiable base learners in the field of PCB defect detection, which we can find much information about the same structure in the past papers.
- **Implementation of CoAtNet in PCB defect detection classification**: We implement the latest version of CoAtNet

that was proposed by Google, and first try to apply the model in PCB defect detection.

3 Problem formulation

We want to apply the CNN-based method to classify the defects of the images generated by the automatic optical inspection of PCBs, but there are many hidden issues in the PCB process that seem to be unrelated but actually affect the learning of the algorithm, such as image pixels, light sources that cause overexposure, camera shooting height, etc. Therefore, in addition to using CNN as the basic model method, we intend to add integrated learning, which means that the voting system will pass the same job to three AI models for judgment, and use the majority decision to confirm whether the judgment of the AOI is correct or not.

Also, we propose three ensemble method to get the results from outputs.

In this project, we want to identify several different models and give the final result based on voting, avoiding too much faith in a single model that leads to many unnecessary discards.

3.1 Dataset

We implement the dataset from the competition—Defect Classifications of AOI from AIdea [5]. we divide the training set into following three parts:

- train_images: 1,769 images (in PNG format) for the training.
- train.csv: contains 2 fields, ID and Label.
    - ID: the file name of the image.
- Label: defect classification category (shown in Table1).
- validation_images: 379 images (in PNG format) for validation
- test_images: the image data (PNG format) required for the test, 380 images in total.
- test.csv: contains 1 field, ID.
    - ID: the file name of the image.

Label index	Type of picture
0	Normal
1	Void
2	Horizontal
3	Vertical Defect
4	Edge Defect
5	Particle

Table 1: label corresponding table

4 Methodology

4.1 Voting Classifier [8]

Voting is popularly used ensemble methods. Basically, voting fits numbers that you defined base estimators independently, and the final prediction takes average over the predictions from all base estimators.

Using the Voting Classifier, each model is trained and predictions are made on the basis of aggregating the results of each basic model. We use Hard Voting to determine the final results based on the prediction calculations of the output categories, while using Voting to aggregate prediction categories or prediction probabilities.

4.2 Soft Gradient Boosting Classifier [9]

Gradient Boosting Machine(sGBM) is proposed by wiring multiple differentiable base learners together, by injecting both local and global objectives inspired from gradient boosting, all base learners can then be jointly optimized with linear speed-up. When using differentiable soft decision trees as base learner, such device can be regarded as an alternative version of the (hard) gradient boosting decision trees with extra benefits. Experimental results showed that, sGBM enjoys much higher time efficiency with better accuracy, given the same base learner in both on-line and off-line settings.

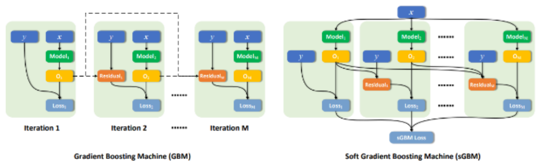

Iteration 1
Iteration 2
Iteration M
Gradient Boosting Machine (GBM)
sGBM Loss
Soft Gradient Boosting Machine (sGBM)

Fig13: Graphical illustration of GBM and sGBM

4.3 Snapshot Ensemble [10]

As each weak model needs to be trained from scratch, resulting in costly training for traditional ensemble methods. Each weak model actually corresponds to a local minimum on the performance surface, and this problem would be solved if there was a way to find multiple local minima without having to start training from scratch each time. The learning rate adjustment strategy [12,13] developed based on the study of performance surfaces is just the right way to explore multiple local minima during the training process. In the paper, the authors use the learning strategy SGDR [13], which has two parts: first it decreases from a max_lr to base_lr by cosine annealing, then restart, which resets the learning rate to max_lr, and finally repeats these two parts periodically throughout the learning process. The learning rate change curve is roughly as follows.

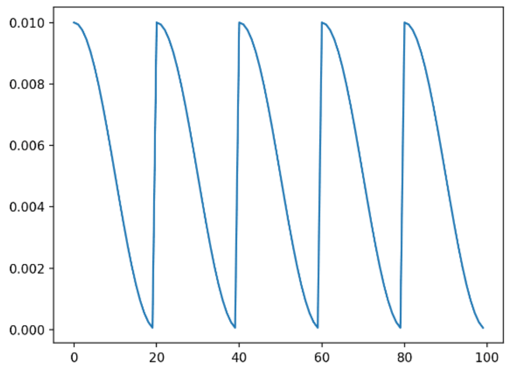


Fig14: Cosine Annealing

After each restart, the model starts exploring other local optima, and the model converges to a local optimum after cosine annealing. This would end up with a weak model from each cycle in the training process, which does not have to be retrained each time, but is just a snapshot of the training process (where the flag is inserted in the figure below).

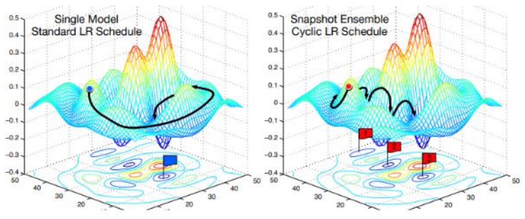


Fig15: Left: Illustration of SGD optimization with a typical learning rate schedule. The model converges to a minimum at the end of training. Right: Illustration of Snapshot Ensembling. The model undergoes several learning rate annealing cycles, converging to and escaping from multiple local minima. We take a snapshot at each minimum for test-time ensembling

4.4 Framework

We implement two models with three ensemble that we have mentioned before. The work flow of our project is as below:

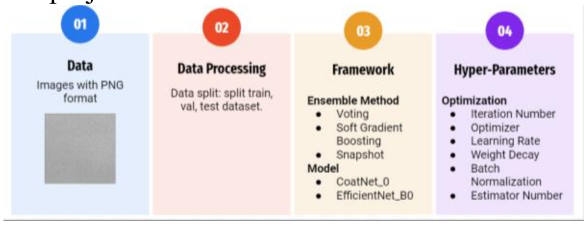


Fig15: Workflow of our system

5 Experimental results

First, we implement only single machine to do detection, the result is as below:

model	Best accuracy of test set
Efficientnet_B0	98.7%
CoAtNet_0	98.4%

Table 2: result of single machine

We can observe the results and figure out that EfficientNet gets better performance than CoAtNet. In addition, we visualize the picture by using grad-cam after processed through the prediction.

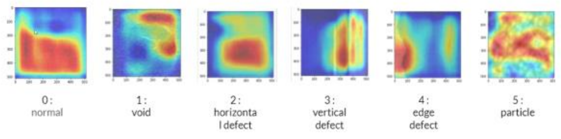


Fig15: feature maps visualization from EfficientNet

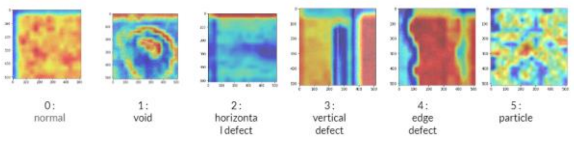


Fig15: feature maps visualization from CoAtNet

Moreover, we implement the three ensemble method, the results are as below:

.

Model / Ensemble method	Efficientnet_B0	CoAtNet
Voting Classifier	100%	98.947%
Soft Gradient Boosting	99.474%	98.843%
Snapshot Ensemble	99.211%	99.211%

Table 3: result ensemble of two models

Model / Ensemble method	Efficientnet_B0	CoAtNet
Voting Classifier	100%	98.947%
Soft Gradient Boosting	99.474%	99.211%
Snapshot Ensemble	99.211%	99.474%

Table 4: result ensemble of three models

In conclusion, we can know from the results in two side. In the view of models, in general, EffiecientNet got higher accuracy than coAtNet, and CoAtNet perform better only when using Snapshot Ensemble of three models. In the side of Ensemble method, surprisingly, Voting Classifier got both 100% with both ensemble of two and three models, Soft Gradient Boosting have higher accuracy when ensemble 2 models, and last but not least, Snapshot Ensemble have higher accuracy when ensemble 3 models.

References

[1] T. Chen, J. Zhang, and Y. Zhou, “A Smart Machine Vision System for PCB Inspection,” Eng. Intell. Syst., pp. 513–518, 2001.

[2] C. Szymanski and M. R. Stemmer, “Automated PCB Inspection in Small Series Production based on SIFT Algorithm,” IEEE Int. Symp. Ind. Electron., vol. 2015–Septe, pp. 594–599, 2015.

[3] Mingxing Tan, Quoc V. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,” arXiv:1905.11946 (https://arxiv.org/abs/1905.11946)

[4]: Zihang Dai, Hanxiao Liu, Quoc V. Le, Mingxing Tan, “CoAtNet: Marrying Convolution and Attention for All Data Sizes,” arXiv:1905.11946 (https://arxiv.org/abs/2106.04803)

[5]: (Dataset) https://aidea-web.tw/topic/a49e3f76-69c9-4a4a-bcfc-c882840b3f27?lang=en

[6]: Attention Is All You Need arxiv 1706.03762

[7]: Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale,” arXiv:2010.11929 (https://arxiv.org/abs/2010.11929)

[8] Zhou, Zhi-Hua. Ensemble Methods: Foundations and Algorithms. CRC press, 2012.

[9] Feng, Ji, et al. Soft Gradient Boosting Machine. ArXiv, 2020.

[10] Huang Gao, Sharon Yixuan Li, Geoff Pleisset, et al., “Snapshot Ensembles: Train 1, Get M for Free.” ICLR, 2017.

[11] Garipov, Timur, et al. Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs. NeurIPS, 2018.

[12] Cyclical learning rates for training neural networks

[13] SGDR: stochastic gradient descent with restartsAlfred. V. Aho and Jeffrey D. Ullman. 1972. The Theory of Parsing, Translation and Compiling, volume 1. Prentice-Hall, Englewood Cliffs, NJ.

[14] Sara A. Althubiti, Fayadh Alenezi, S. Shitharth, Sangeetha K., Chennareddy Vijay Simha Reddy, "Circuit Manufacturing Defect Detection Using VGG16 Convolutional Neural Networks", Wireless Communications and Mobile Computing, vol. 2022, Article ID 1070405, 10 pages, 2022. https://doi.org/10.1155/2022/1070405