以對比學習為基礎結合深度圖之印刷電路板瑕疵檢測

In today's PCB manufacturing process, the most widely used defect detection method is the AOI system. However, this system cannot detect defects on the side or underneath the objects being tested. Therefore, we have introduced depth maps captured by an infrared camera and combined them with deep learning to train a defect detection model, with the aim of detecting defects that were previously undetectable using traditional methods. Additionally, in cases where the manufacturing process typically yields a very high pass rate, the number of defect samples available for training the model is much smaller compared to good samples. This can lead to the model being dominated by the characteristics of good samples and failing to accurately identify defects. To address this issue of imbalanced data sets, we employ a training method based on contrastive learning and special optimizations for defect samples, resulting in excellent results with a leakage rate of 0.77% and a overkill rate of 0.99 %.

吳映萱、謝君偉

Keywords: Defect Detection、Depth Map、Imbalanced Dataset、Contrastive Learning

1. Background

Today, the manufacturing of Printed Circuit Boards (PCB) is predominantly achieved through Surface Mount Technology (SMT), which involves soldering electronic components onto the PCB using solder paste. After SMT assembly, it is necessary to inspect the soldered components for defects. The most widely used method for such inspection is the Automatic Optical Inspection (AOI) system. AOI systems utilize image technology to compare the differences between the test object and standard images to determine whether the test object meets the specified standards. However, AOI systems require the adjustment of numerous machine parameters to accommodate different test objects. Additionally, since the test object's images are captured from above using a camera, they can only inspect surface defects and are unable to detect defects on the sides or underneath the test object.

2.Motivation

In light of the limitations of the aforementioned AOI system, we propose the training of a defect recognition model using machine learning. This approach aims to achieve higher accuracy in defect detection while increasing automation. To address the drawback of AOI systems not being able to detect defects on the sides or underneath the test object, we have introduced depth information for the test object to tackle this issue.

Methods

1. Defect Recognition Model

(1) Model's Task

To obtain image data of the components, we begin by using an edge detection algorithm to identify the components on the PCB, and then each component is individually segmented into separate images. The model's task is to classify each component image as either a good component or a defective component. To provide a more detailed understanding of the types of defects, defective components are further categorized into 11 classes, including empty solder, short circuit, location shift, and others.

(2) Evaluation Metrics

Given the advancements in SMT technology, modern PCB manufacturing achieves high yield rate. Therefore, the primary focus of the model's task is to accurately identify a small number of defective components while avoiding good components are misclassified as defective. As a result, the key metrics for this task are the leakage rate and overkill rate. The leakage rate refers to the ratio of defective components incorrectly classified as good components, while the overkill rate is the ratio of good components wrongly classified as defective components. Lower values for these two metrics indicate better performance in classifying component defects.

2. Depth Information of Components

(1) Acquiring and Presenting

We use an infrared camera to capture the depth information of the test objects on the PCB that the result is a grayscale depth map. Grayscale values range from 0 to 255, with lower values visually representing objects farther away from the camera, indicating greater depth. Higher values represent objects closer to the camera, indicating shallower depth, as shown in **Figure 1(b)** (Andreas Eitel et al., 2015).

(2) Utilizing

We aim for the model to simultaneously learn from both the RGB color information of the components and the depth information. To achieve this, we concatenate the RGB image obtained from a regular camera with the depth map. However, since the depth map has only one channel (grayscale), while the RGB image has three channels, simply concatenating them would result in the depth information contributing only 1/3 of the total information learned by the model. To balance the amount of information learned from both images, we use the colorjet method (Andreas Eitel et al., 2015) to convert the grayscale depth map into a 3-channel RGB image, as shown in **Figure 1(c)** (Andreas Eitel et al., 2015). Therefore, the input data for the model consists of a 6-channel image created by concatenating the RGB image from the regular camera with the converted 3-channel depth map.

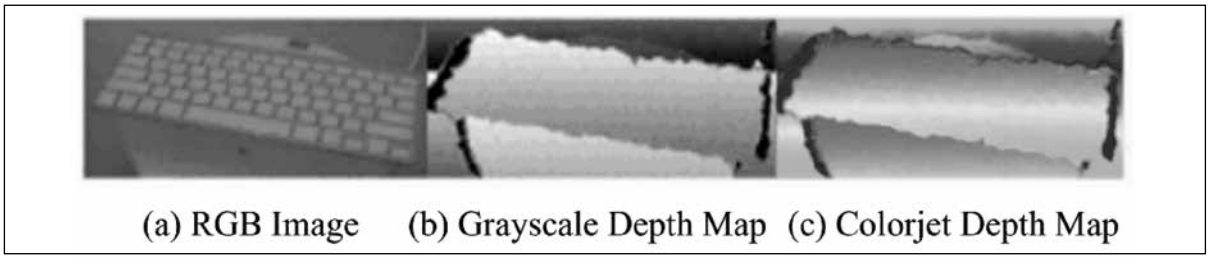

Figure 1. Different Images of the Same Object

(3) Advantages

In the electronics manufacturing industry, there are other defect detection methods that use depth information, known as 3D AOI. 3D AOI builds on the foundation of 2D AOI by incorporating multiple directional light sources with the use of Moiré fringes or laser projection to calculate a 3D model of the test object. However, 3D AOI, like 2D AOI, requires the adjustment of numerous parameters depending on the test object's characteristics. Additionally, the inclusion of an extra dimension of information significantly increases the computational load, which can reduce inspection speed.

On the other hand, the depth information we introduced is in the form of 2D images. Compared to 3D AOI, this approach allows maintaining essential

depth information while effectively reducing the computational load, thereby accelerating the inspection speed. This presents a potential advantage in terms of efficiency and speed for defect detection in electronic manufacturing.

3.Characteristics and Corresponding Methods of Class-imbalanced Dataset

(1) Class Imbalance

In today's PCB manufacturing, there is an extremely high yield rate, results in a class imbalance issue, where the majority of the data is from good components and there's a scarcity of defect component images. Training a conventional classification model on such imbalanced data leads to a situation where the model learns more about good components, and this can result in a bias where it tends to classify images as good components. In scenarios where the majority of the test data consists of good components, it can lead to high accuracy on paper but may fail to correctly classify defect components. Our experiments have also demonstrated that directly training a standard supervised classification model on such imbalanced data results in an extremely low overkill rate and a high leakage rate (as shown in **Table 1**), which means that the model tends to classify most images as good components, including some defective ones, leading to poor defect recognition performance.

Table 1. Predicting Results of Using General Classification Models

Backbone	Leakage rate (%)	Overkill rate (%)
MobileNetV3-lrage (Andrew Howard et al., 2019)	12.3288	0.1986
ResNet-18 (Kaiming He et al., 2016)	15.0684	0.1264
ResNet-101 (Kaiming He et al., 2016)	17.1233	0.1083

(2) Neural Network Model Training under Class Imbalance

1. Shortcomings of Traditional Methods

For addressing the issue of class imbalance, the traditional and most straightforward approach is re-sampling, which comes in two main forms: over-sampling the minority class or under-sampling the majority class. Both methods aim to equalize the class distribution through sampling. However, they each have their limitations.

Over-sampling, which involves replicating samples from the minority class, may lead to overfitting of the minority class. On the other hand, under-sampling, where samples from the majority class are reduced, may result in a significant reduction of information and cause a drop in the model's generalization ability.

2. Training Method based on Contrastive Learning

As mentioned in Section 貳、三、(一), supervised learning performs poorly when dealing with class-imbalanced training data. To address this issue, we have adopted a self-supervised learning approach, specifically leveraging contrastive learning. Many contrastive learning methods are built on the task of instance discrimination, where each sample is treated as a positive example and augmented views of the same data are considered positive, while other samples are treated as negatives. By comparing the differences between positive and negative samples, a feature extractor is trained to learn richer and more stable features.

Contrastive learning typically involves comparing each sample with a large number of negative samples. SimCLR (Ting Chen et al., 2020) achieves this by using extremely large batch sizes. In contrast, MoCo v2 (Xinlei Chen et al., 2020) stores a substantial number of negative samples in a queue, allowing training with smaller batch sizes while achieving better performance. Considering the advantages of MoCo v2, our training method is also based on this structure.

Our architecture, as shown in **Figure 2(a)**, consists of two networks with the same structure, referred to as the query network and the key network. Each network includes a feature extractor and a two-layer Multilayer Perceptron (MLP). Additionally, a fully connected layer is added to the feature extractor of the query network to serve as a classifier and a queue is used to store negative samples.

In the training process of the model, we denote the training dataset with n samples as D = $\{x_i, y_i\}$, where x_i represents the i-th image sample, and y_i represents the corresponding label. Taking x_i as an example, we perform two types of data augmentations to get two different views, x_i^q and x_i^k. These views are separately input into the query network and the key network, and obtain high-dimensional features z_i^q and z_i^k.

Additionally, we use a queue Q to dynamically collect historical features. The queue push features from the current batch while popping the oldest batch's features. Assuming the size of the queue Q is N, we can set that N is much larger than the batch size, allowing the queue to provide abundant negative samples. In summary, our contrastive learning loss L_{Con} is defined as:

$$L_{Con} = -log\left(\frac{exp\left(z_i^{q^T} z_i^k / \tau\right)}{exp\left(z_i^{q^T} z_i^k / \tau\right) + \sum_{zj \in Q} exp\left(z_i^{q^T} z_j^k / \tau\right)}\right) \quad (1)$$

where τ is a temperature hyperparameter. Through the contrastive learning loss L_{Con}, we can increase the similarity between the two views from the same sample and decrease the similarity between two different samples, allowing the feature extractor to learn better features.

After obtaining good features, the next step involves training a classifier that can correctly categorize these features. Typically, a classifier is trained using the cross-entropy loss. For a sample x_i, the feature obtained from the feature extractor F_q is denoted as $F_q(x_i)$. The cross-entropy loss is formulated as:

$$L_{CE} = -log\left(\frac{exp\,(F_q(x_j))}{\sum_{j=1}^{k} exp\,(F_q(x_j))}\right) \quad (2)$$

where k represents the number of classes in the dataset. However, in the case of class imbalance, the overwhelming number of samples in the majority class can lead to gradient updates dominated by the majority class in the cross-entropy loss, causing the model to lean towards the majority class.

To solve this situation, we train the classifier using balanced softmax loss (Jiawei Ren et al., 2020). Balanced softmax loss is based on the cross-entropy loss but incorporates class sample counts as weights to balance the loss for each class, achieving a balanced gradient effect. It is formulated as:

$$L_{BS} = -log\left(\frac{n_i exp\,(F_q(x_j))}{\sum_{j=1}^{k} n_j exp\,(F_q(x_j))}\right) \quad (3)$$

where n_i represents the number of samples in class to which sample x_i belongs. Through the use of balanced softmax loss, we successfully balance the learning of the classifier. Therefore, the overall loss for the training framework is:

$$L = L_{Con} + L_{BS} \quad (4)$$

To further enhance the model's performance, we implemented several optimizations. First, in the original MoCo v2 architecture, the parameters of the key network are dynamically updated by the query network using momentum updates. However, recent research (Xinlei Chen et al., 2020) suggests that not updating the key network's parameters at all can lead to better training results. Our experiments (see Section 參、四、(二)) also support this finding. Therefore, in our training framework, the key network does not undergo gradient updates. Furthermore, to mitigate any inherent biases in the model that could impact training results, we simultaneously train multiple models, their predictions are aggregate to form the final output,

which is referred to as ensemble learning.

During the testing phase, samples are not subjected to data augmentation. Instead, they are directly input into the feature extractor F_q of the query network to obtain features, which are then classified by the classifier. This process is similar to a standard classification model, as depicted in **Figure 2(b)**. Therefore, our training framework not only does not slow down the testing speed but also allows for improved classification results.

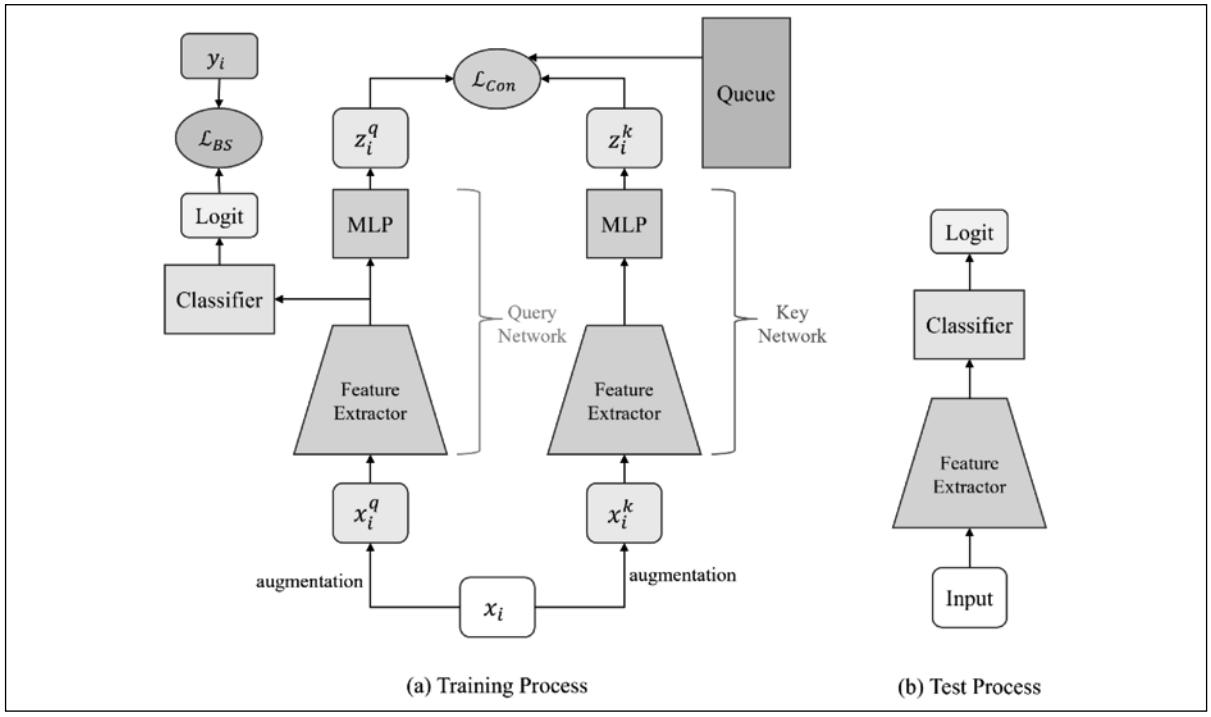

Figure 2. Training and Testing Architecture

4. Other Designs to Cope with Dataset Characteristics

(1) Lightweight Neural Network

Considering the need for fast and real-time industrial inspection, we adopted the lightweight neural network MobileNetV3-large (Andrew Howard et al., 2019) as the feature extractor. In addition to its speed advantages, we also found that the results obtained when training with MobileNetV3-large outperformed those achieved with deeper neural networks. We discuss the reasons for this in more detail in Section 參、四、(四).

(2) Optimized Design for Specific Categories

Upon reviewing the test results, we observed that the recognition rates for certain defect categories, such as "empty solder" and "location shift," were particularly poor. After a detailed investigation, we believe the reason for this lies in the fact that these defect categories require the observation of the entire image for accurate detection. For example, in the "empty solder" category, many images exhibit insufficient solder paste at one end of the component, leading to an uneven height at the two ends of the component. This results in a color gradient in the depth image. In the "location shift" category, some images involve slight component displacements. Both of these scenarios require the observation of the entire image to identify defects accurately. However, convolution-based neural networks, including the MobileNetV3-

large that we are using, are designed to observe only local image information.

To address this issue, we replaced the SE block (Jei Hu et al., 2018) in the MobileNetV3-large architecture with the Coordinate Attention block (CA block) (Qibin Hou et al., 2021) The SE block initially performs 2D global pooling to obtain features along the channel dimension and then applies channel attention, which results in improved performance with fewer computations. However, the SE block only considers inter-channel information and neglects positional information, making it unable to improve the problem of obtaining only local image information. In contrast, the CA block embeds positional information into channel attention, allowing a lightweight network to apply attention to the entire image without significantly increasing computational demands. Our experiments have demonstrated that by replacing the SE block in MobileNetV3-large with the CA block, we significantly enhance the recognition rates of defect categories that require observation of the entire image (see Section 參、四、(五)). This improvement has also positively impacted the overall model performance.

Results and Discussions

1. Dataset

The dataset we utilized comprises over 300 passive components. The training dataset consists of a total of 568,751 images, with 505,030 images of good components and 63,721 images of defective components. The testing dataset comprises 5,222 images, with 4,704 images of good components and 518 images of defective components. The dataset is categorized into 12 classes: good, location shift, short, empty solder, missing, inverted parts, tombstoning, row offset, opposite polarity, extra part, bridging, and other defects.

2. Implementation Details

The model utilizes a backbone consisting of MobileNetV3-large with SE block replaced by CA block. The queue size is set to 8192. A total of 4 model instances are used for ensemble learning. We use the SGD optimizer with a momentum of 0.9 and a weight decay of 0.0002. The learning rate starts at 0.02 and decreases using cosine annealing. The model is trained for a total of 400 epochs with a batch size of 64.

3. Experimental Results

As mentioned in Section 貳、一、(二), our evaluation metrics are leakage rate and overkill rate. The calculation for these metrics is as follows. Leakage rate: the number of samples where defective components of the 11 categories are misclassified as "good" divided by the total number of defective component samples; Overkill rate: the number of "good" component samples misclassified as any of the defective categories divided by the total number of "good" component samples. Our architecture achieved outstanding results on the test dataset with a leakage rate of 0.7722 % and an overkill rate of 0.9991 %.

4. Discussions

(1) Impact of Applying Depth Maps

To validate the impact of incorporating depth information and applying colorjet to depth maps, we compared the following methods: using only RGB images, using only colorjet-processed depth maps, concatenating RGB images with original grayscale depth maps (4-channel), and concatenating RGB images with colorjet-processed depth maps (6-channel). The experimental results are shown in **Table 2**, confirming that the combined use of RGB images and depth maps significantly improves the model's performance. Additionally, applying colorjet to depth maps further enhances the results.

Table 2. Comparison of Results from Different Input Images

Input Image	Leakage rate (%)	Overkill rate (%)
RGB Image	1.2445	1.9861
Depth Map	0.9871	2.1673
RGB Image + Grayscale Depth Map	0.9131	1.0620
RGB Image + Colorjet Depth Map	**0.7722**	**0.9991**

(2) Influence of Stop-gradient

As mentioned in Section 貳、三、(二) 2., in the original MoCo v2 architecture, the parameters of the key network are dynamically updated by the query network periodically. Therefore, we compared the practice of stopping gradient updates with the use of the original parameter updating method to demonstrate the effectiveness of stopping gradient updates in the key network. The results are presented in **Table 3**.

Table 3. Comparison of whether Key Network Performs Gradient Update or not

Gradient Update	Leakage rate (%)	Overkill rate (%)
Momentum Update	0.9784	1.5965
Stop-gradient	**0.7722**	**0.9991**

(3) The Effect of Ensemble Learning

We used a total of 4 models for the classification task and averaged the predictions of each model to obtain the final prediction. In this section, we compared the predictions of each individual model with the integrated predictions. The experimental results are shown in **Table 5**, demonstrating that ensemble learning can indeed improve model performance. Additionally, from **Table 4**, it can be observed that the predictions of Model 4 are particularly extreme, which can be attributed to neural network oscillations. However, through ensemble learning, we can effectively mitigate this issue.

Table 4. Results of Individual Model and Ensemble

Model	Leakage rate (%)	Overkill rate (%)
Ensemble	0.7722	0.9991
Model 1	0.9871	**0.9183**
Model 2	0.7961	1.2732
Model 3	0.9235	1.0646
Model 4	**0.6846**	1.9041

(4) Selection of Neural Network

In the selection of neural networks, we considered the model's prediction speed and chose the lightweight network MobileNetV3-lrage. We were pleasantly surprised to find that MobileNetV3-lrage outperformed deeper networks. The comparative results are shown in **Table 5**. For this situation, we believe there are two main reasons: First, due to the limited training data for our defect elements, deeper neural networks are more prone to overfitting the training data, resulting in poor performance on the test data; On the other hand, transformer-based networks lack the inductive biases that convolutional neural networks have, such as translation equivariance and locality. As a result, they require longer training times and may not effectively learn with limited training data (Zhengzhuo Xu et al., 2022).

Table 5. Comparison of Using Different Backbones

Backbone	Leakage rate (%)	Overkill rate (%)
MobileNetV3-lrage (CA block)	**0.7722**	**0.9991**
ResNet-18 (Kaiming He et al., 2016)	0.9438	1.2644
ResNet-101 (Kaiming He et al., 2016)	1.0725	1.0831
ConVNeXt_tiny (Zhuang Liu et al., 2022)	0.8687	1.2661
ConVNeXt_small (Zhuang Liu et al., 2022)	0.7722	1.3618
EfficientFormerv2_S0 (Yanyu Li et al., 2022)	1.0135	1.3021
EfficientFormerv2_S1 (Yanyu Li et al., 2022)	0.9652	1.4402

(5) The Effect of CA Block

In terms of feature extractor optimization, we compared the predictive results of models using the original MobileNetV3-lrage and the MobileNetV3-lrage with SE blocks replaced by CA blocks. The results are shown in **Table 6**. Besides the overall improvement in performance, we also analyzed the leakage rates for each category. We observed significant reductions in the leakage rates for defect categories that require observing the entire image, such as "empty solder" and "location shift," as well as improvements in other categories. This demonstrates that introducing global positional information through CA blocks has indeed helped the model in recognizing specific defect categories.

Table 6. The leakage rate (%) of each class whether the Backbone Replaces the CA block or not

Class	MobileNetV3-large	MobileNetV3-large (CA block)
Total	0.9473	0.7722
Location Shift	1.5628	1.3267
Short	0.1660	0.0415
Empty Solder	1.1898	1.0946
Missing	0.0000	0.0000
Inverted Parts	0.0000	0.0000
Tombstoning	0.0870	0.0870
Row Offset	1.0607	1.0607
Opposite Polarity	1.6730	0.8365
Extra Part	0.9456	0.0000
Bridging	1.1153	0.5583
Other	3.2904	2.4076

Conclusion

In this paper, we propose a precision printed circuit board (PCB) defect detection method based on contrastive learning and the integration of depth information. Our approach focuses on two key aspects: firstly, by incorporating depth maps captured through infrared imaging, we can detect defects on the sides and underneath the test objects, which are typically challenging for conventional AOI systems to assess. Secondly, addressing the issue of class imbalance due to the scarcity of defect samples in the training dataset, we utilize contrastive learning as the foundation and implement other optimization strategies. These efforts result in a robust classification model that achieves impressive detection performance with a leakage rate of 0.7722 % and an overkill rate of 0.9991 %.

References

1. Andreas Eitel, Jost Tobias Springenberg, Luciano Spinello, Martin Riedmiller, & Wolfram Burgard (2015, September). Multimodal Deep Learning for Robust RGB-D Object Recognition. International Conference on Intelligent Robots and Systems. Hamburg, Germany.
2. Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Jiawei Ren, Cunjun Yu, Shunan Sheng, Xiao Ma, Haiyu Zhao, Shuai Yi, & Hongsheng Li (2020, December). Balanced Meta-Softmax for Long-Tailed Visual Recognition. Conference on Neural Information Processing Systems. Online.
3. Jie Hu, Li Shen, Samuel Albanie, Gang Sun, & Enhua Wu (2018, June). Squeeze-and-Excitation Networks. Conference on Computer Vision and Pattern Recognition. Salt Lake City, USA.
4. Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun (2016, June). Deep Residual Learning for Image Recognition. Conference on Computer Vision and Pattern Recognition. City of Las Vegas, USA.
5. Qibin Hou, Daquan Zhou, & Jiashi Feng (2021, June). Coordinate Attention for Efficient Mobile Network Design. Conference on Computer Vision and Pattern Recognition. Nashville, Tennessee, USA.
6. Ting Chen, Simon Kornblith, Mohammad Norouzi, & Geoffrey Hinton (2020, July). A Simple Framework for Contrastive Learning of Visual Representations. International Conference on Machine Learning. Online.
7. Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, & Hartwig Adam (2019, October). Searching for MobileNetV3. International Conference on Computer Vision. Seoul, Korea.
8. Xinlei Chen, Haoqi Fan, Ross Girshick, & Kaiming He (2020). Improved Baselines with Momentum Contrastive Learning. arXiv preprint arXiv:2003.04297.
9. Xinlei Chen, & Kaiming He (2020). Exploring Simple Siamese Representation Learning (2020, June). Conference on Computer Vision and Pattern Recognition. Online.
10. Yanyu Li, Ju Hu, Yang Wen, Georgios Evangelidis, Kamyar Salahi, Yanzhi Wang, Sergey Tulyakov, & Jian Ren (2023, October). Rethinking Vision Transformers for MobileNet Size and Speed. International Conference on Computer Vision. Paris, France.
11. Zhengzhuo Xu, Ruikang Liu, Shuo Yang, Zenghao Chai, & Chun Yuan (2023, June). Learning Imbalanced Data with Vision Transformers. Conference on Computer Vision and Pattern Recognition. Vancouver Canada.
12. Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, & Saining Xie (2022, June). A ConvNet for the 2020s. New Orleans, USA.

作者簡介

吳映萱 / 國立陽明交通大學 智慧系統與應用研究所

謝君偉 / 國立陽明交通大學 智慧系統與應用研究所 / 指導老師