Orchid Type Recognition and Classification

[1]*Chao-Hsun Yang* (楊朝勛), [1]*Chiou-Shann Fuh* (傅楸善)

[1]Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, Taiwan,
jellopray860928@gmail.com fuh@csie.ntu.edu.tw

ABSTRACT

We attended a competition of AI to correctly predict breeds of orchids out of orchid's photographs. We use Pytorch pre-train model and apply some strategy of deep learning, and we get a prediction above-average score compared with other participating groups. In this essay, we will go through the process of how we modify the model to make it better and better.

Keywords: TBrain, Orchid

1. INTRODUCTION

Orchids in Taiwan have been planted for a long time, and widely known for their quantity and quality in the world. Taiwan has the foremost technology for cultivating new breeds of orchids, and it has the greatest amount of breeds of orchids compared with other countries. With number of breeds increasing, the recognition of different breeds become a tough problem. Nowadays, only experts can distinguish the breeds among the orchids, but it is time-consuming and exhausting. Since no robust and rapid recognition system has been invented, we like to use deep learning to create a model for distinguishing the breeds, and relieve the workloads of the experts.

We choose this subject based on a competition from 1 April 2022 to 6 June 2022. It is held on "TBrain AI實戰吧" platform. We have two months to train a model for predicting orchids' breed. The competition prize will be awarded to the top three participating groups. We rank 58th among 743 groups on this competition.

2. METHODOLOGY

The dataset contains 2,190 images with 219 breeds. There are 10 images in a breed in Fig. 1. To begin training a model, we split it into train, validation and test data, which have 1,533, 219, 438 photographs, respectively.

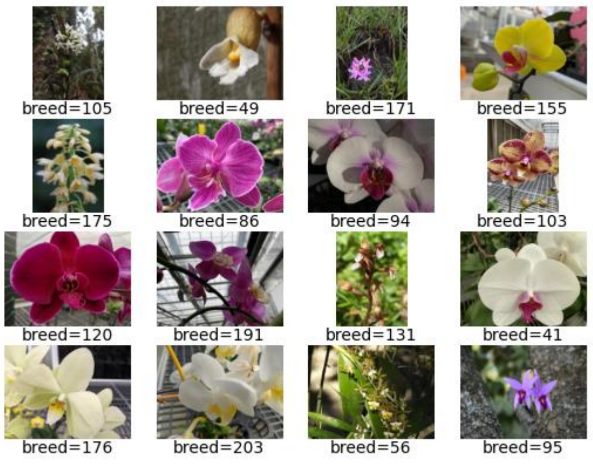


Fig. 1. Examples of orchid images.

From 1 April 2022 to 12 April 2022, we used

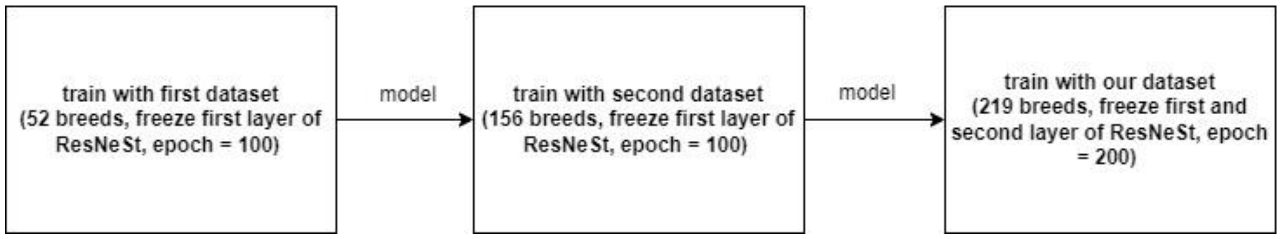

train with first dataset
(52 breeds, freeze first layer of ResNeSt, epoch = 100)
model
train with second dataset
(156 breeds, freeze first layer of ResNeSt, epoch = 100)
model
train with our dataset
(219 breeds, freeze first and second layer of ResNeSt, epoch = 200)

Fig. 2. Steps of training with three datasets.

a Pytorch pre-train model, ResNet50. It is a deep learning model and mostly used for image classification. With some parameter set, such as epochs = 50 and batch size = 32, we got 72% correctness on test data. Then we augmented train data by rotating or flipping original photographs, so the numbers of photographs for training became larger, and we got 78% correctness on testing data.

The problem we encounter for this model is overfitting, we see, after 50 epochs, 100% correctness on training data. Thus, we try to add more photographs for each breed. Ideally we want to increase the number of photographs of each breed to 50, but telling the breed of all photographs from Internet is difficult. Besides, representations of the breeds in our dataset are numerical. They act as huge barriers to the work since we cannot use these labels to search for more photographs. We did find thousands of photographs, but we did not have a proper way to automatically convert their identity to the numerical labels. Finally, we abandoned this idea.

Another approach to handle the overfitting is cross validation. We do not know whether validation data can represent all photographs of orchids that the model wants to classify. If it is not a representative, we may find a totally different validation accuracy when we select other photographs as validation data. With cross validation, we can solve this problem. We divide training data into *k* subsets, each subset will be treated as validation data once and remaining *k*-1 subsets will be used for training. After complete training with each subset is validation data once, we will have *k* models. Their parameter is recorded with the highest validation accuracy of their own validation data. Finally, we use these models to predict the breeds of photographs from test data and apply majority rule to get the result of breed prediction.

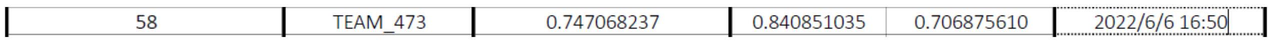
From 12 April 2022 to 25 April 2022, we added 10-fold cross validation to the training, but did not improve the result. Then we tried different data augmentation such as affine transformation, still did not see major improvement.

After doing some research, we start training with different models. For example, we replaced ResNet50 to ResNeXt50 and Inception v3, both are Pytorch pre-train models. After we increased the number of epochs to 200 and decreased the batch size to 16, the accuracy of testing data with ResNeXt50 became 82%.

From 25 April 2022 to 23 May 2022, we continually tried different models. This time ResNeSt become our major model, because it is based on ResNeXt plus the idea of split attention. We assumed a better accuracy with this model, but the result did not fulfill our expectation. We realized that maybe constant learning rate was a problem, so we made learning rate decay in half every 75 epochs. With base learning rate = 0.001 and SGD (Stochastic Gradient Descent) optimization been used, we got 83.1% accuracy with this idea.

We notice that transfer learning is often used for pre-train model. Transfer learning means leaving some parameters of model unchanged during learning period on current dataset. It is often implemented when the dataset is relatively small. ResNeSt has 4 major layers, each layer includes convolution, ReLU function, batch normalization, and so on. We make the first layer frozen, so the parameter of this layer would stay the same. Unfortunately, the accuracy was 83.1%, which was virtually identical to discounting transfer learning.

Besides, we find two open datasets of orchid photographs, they are also used for breed classification. The numbers of breeds are 52 and 156, respectively. Since they are different among three datasets, we cannot combine all photographs together and train as a whole.



58	TEAM_473	0.747068237	0.840851035	0.706875610	2022/6/6 16:50

Fig. 3. Score of the competition (ranks 58th among 743 teams, team name, final score, score of public dataset, score of private dataset, last uploaded time).

Instead, we train the model with each dataset one by one. The steps of training are in Fig. 2. Because first and second dataset are not what we want to predict in the end, we just wish our model learns more about orchid's shape or other features from these extra data. Therefore, we made epoch 100 to train with these datasets, rather than 200. The accuracy of testing data gets a little bit improved. Maybe it is a sign to show us the magnitude of the size of data, and distinguishing each dataset to different steps of training seems feasible. If we can find more datasets related to orchid, we will include them into the training process.

Previously, we resized the photographs into 224 × 224 pixels recommended by Pytorch. The original size of the photographs is bigger than 224 × 224 pixels, so the photographs need to be condensed and probably lose some spatial information between the pixels. We also find the width and the length of the photographs are either 480 or 640 pixels. To keep the completeness of spatial information, we abandon resize function. Instead, we crop the periphery of the photographs and retain the center. After cropping, all photographs are reduced to 480 × 480 pixels. We still use those two datasets to help fine-tune the model, and finally we have 85.4% accuracy on our test data.

From 23 May 2022 to 5 June 2022, we modified the program for cross validation. We use 5-fold cross validation for training. Previously the model went through a complete 5- fold cross validation learning at an epoch. It is not very feasible because it will update model's parameters 5 times. Model will then be easily overfitting. This time we changed the order that we let each fold do a complete training, so we would get a model from a fold. Eventually we got 5 models. Each model had its own knowledge about orchid photographs. We used these models to predict test data and applied majority rule to get the final prediction result. Because of time constraint, we make epoch 50 for each fold. The total number of epochs are 250, which is more than earlier training only a little. We got 85.8% accuracy with this idea. If we see the single model's accuracy for test data, it will locate between 81% to 83%.

Finally, we train the model with the whole data which consists of 2,190 photographs, because we want model to recognize different photographs as much as possible. We still apply 5-fold cross validation and majority rule in our model, and it is ready to predict photographs published on platform at 6 June 2022.

	Accuracy	Macro F1-score
ResNet50	78%	
ResNeXt50	82%	
Inception v3	78%	
ResNeSt50	83.1%	0.819
ResNeSt50 (layer1 unchanged)	83.1%	0.822
ResNeSt50 (layer1, layer2 unchanged, two datasets added)	83.8%	0.824
ResNeSt50 (layer1, layer2 unchanged, two datasets added, 480 × 480)	85.4%	0.834
ResNeSt50 (layer1, layer2 unchanged, two datasets added, 480 × 480, 5-fold cv)	85.8%	0.846

Table 1. Accuracy of testing data.

3. COMPETITION RESULT

On 6 June 2022, two datasets are published on platform for prediction. One is public dataset, and it contains 40,285 photographs. The prediction score of this dataset will be shown on the Net, so every participating group can see it immediately. The other is private dataset, and it contains 41,425 photographs. Its prediction score will be hidden until the end of the competition. These two scores account for 30% and 70% of the final score, respectively.

Among 743 teams, we received 0.840851 and 0.706875 as our scores of public and private datasets, respectively. Final score is 0.747068. We rank 58th on this competition. The score of the 1st team is 0.877667, which means there are still many methods to improve model and keep trying.

4. CORRECT EXAMPLES

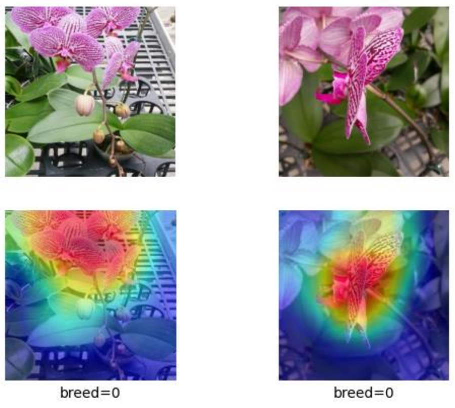


Fig. 4. Examples of correctly predicted images (breed = 0).

We use Grad-CAM to see if our model makes predictions bases on the right area of the picture. We check the weights of its final layer, drawing a heat map and combine original photograph and heat map together as a composite photograph. The redder the color shows on the photograph, the more important this pixel is for making the prediction.

We can see our model mainly focuses on the right area of original photograph in Fig. 4. In this case it is not surprising because flowers occupy many pixels and the colors are bright.

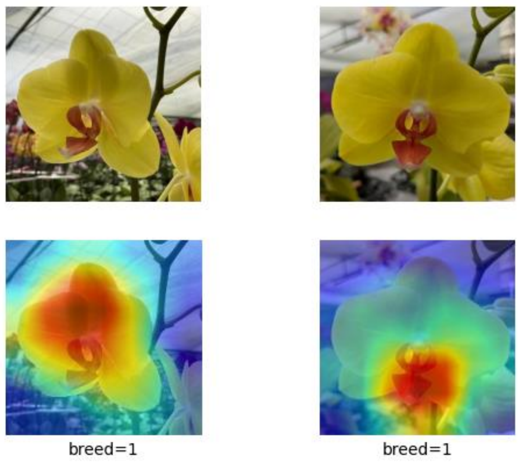


Fig. 5. Examples of correctly predicted images (breed = 1).

Another example (Fig. 5.) shows that bigger flowers are likely to get correct prediction.

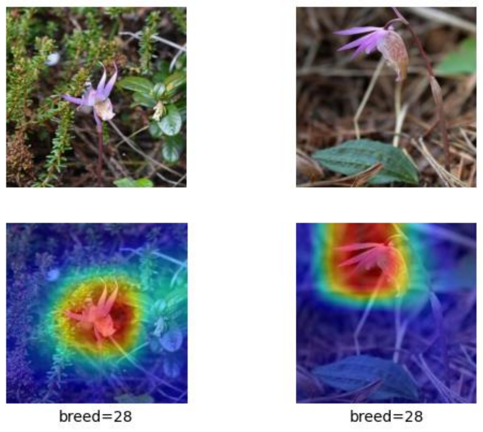


Fig. 6. Examples of correctly predicted images (breed = 28).

In Fig. 6 even small flowers can be detected and get correct predictions. In these photographs, there are green leaves and brown leaves covering much area, but the model views them as backgrounds, which is correct, probably because leave colors are common among all photographs, our model learns that leaves are not significant for classification.

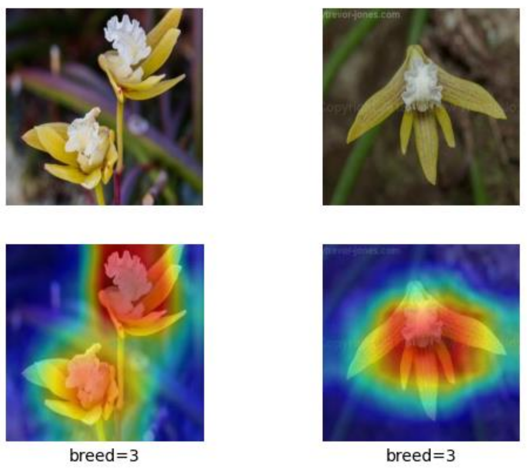


Fig. 7. Examples of correctly predicted images (breed = 3).

We see two photographs in Fig. 7 with the same breed of orchid, but with different number of flowers and angles of shot. Our model can still understand they represent the same breed of orchid and make the right prediction for both photographs.

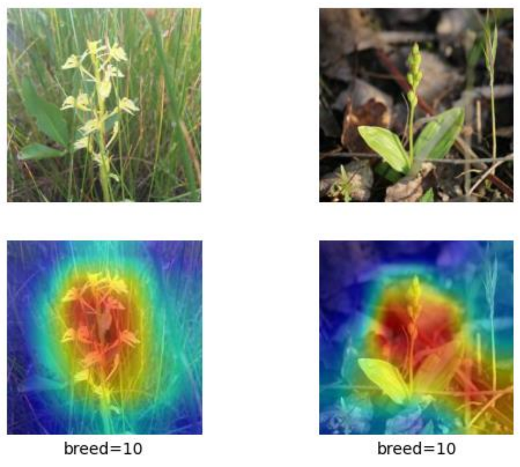


Fig. 8. Examples of correctly predicted images (breed = 10).

We see two photographs in Fig. 8 with the same breed of orchid, but flowers in the left photograph are in full bloom, and one in the right photograph is not. Our model can still understand they represent the same breed of orchid and make the right prediction for both photographs.

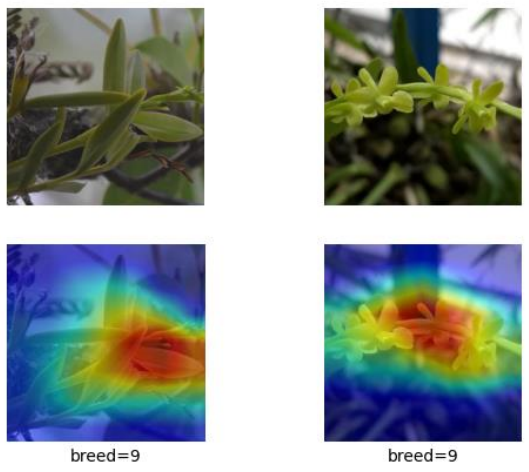


Fig. 9. Examples of correctly predicted images (breed = 9).

Fig. 9 shows that although color of flowers is green, same with color of leaves, our model can still understand they represent flowers instead of leaves and make a correct prediction.

5. INCORRECT EXAMPLES

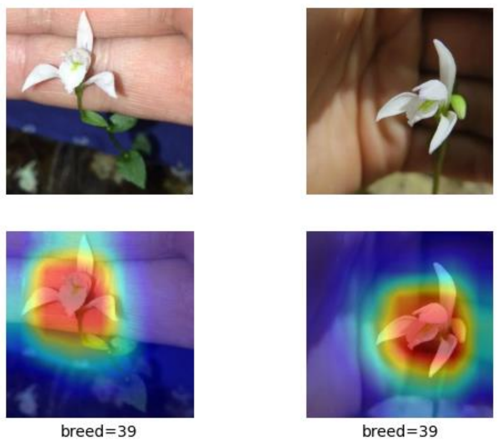


Fig. 10. Examples of incorrectly predicted images (breed = 39, predicted breed = 81).

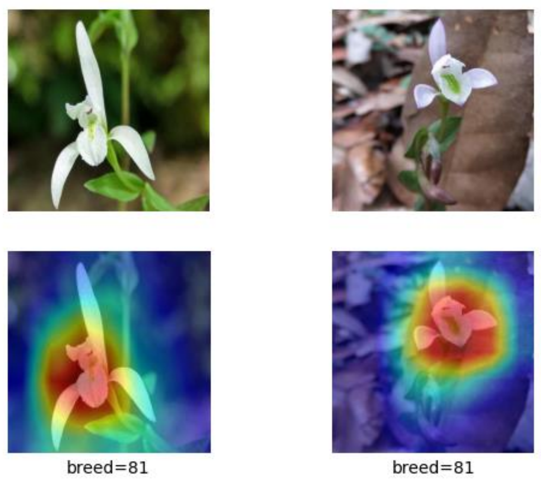


Fig. 11. Examples of incorrectly predicted images (breed = 81, predicted breed = 39).

We see orchids in Figs. 10 and 11 are very alike. Although our model predicts mainly by the flower's location, it gets these two breeds of photographs mixed up. Probably because photographs of train data have some parts different from test data, such as texture, color temperature of photographs, and camera location. We use 5-fold cross validation to deal with this problem, but the problem remains.

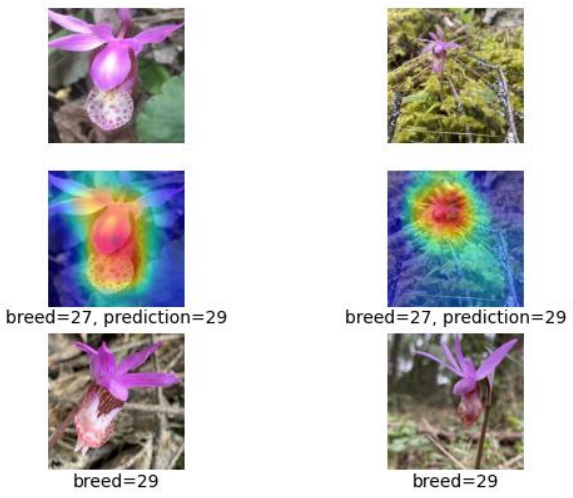


Fig. 12. Examples of incorrectly predicted images (breed = 81, predicted breed = 39).

From Figs. 12, we see no matter how small flowers are in photographs, our model will detect them. We do not need to worry our model make predictions without paying attention on less important area. However, we cannot correctly predict the orchid of certain breeds. Our model thinks the breed of the top two photographs is 29. Maybe the architecture of our model cannot find the difference between the top two photographs and the bottommost two photographs, so modification of our model is necessary.

6. DISCUSSION

We notice Google has a cloud computing resource. It will automatically train a model with our dataset, so we do not need to have any knowledge of machine learning, try to modify parameter or worry about which model is the best for our dataset. We will use this resource to get a model, and consider it as a baseline.

We find that the accuracies of the validation data with different models are almost identical. Maybe because the validation data are biased. We have tried cross validation once, but did not see any improvement. Maybe there is something wrong in our program. Later we changed the order of epoch iteration and 5-fold iteration, and we eventually and successfully apply cross validation to our model. We notice that majority rule is only one of the strategies for final prediction, so we will try others in the future.

This time we did not change the structure of the model. We understand we can add and delete some layers or put our invented structure on the pre-trained model. Thus, we will follow state-of-the-art ideas in journals, try to modify them and add them to our model.

7. CONCLUSION

Now, deep learning model plays an important role in classification task. Different architectures of model will affect the result of prediction. So far we have only tried Pytorch pre-train model, and already encountered many problems. It demonstrates that deep learning is a challenging work, but it is also interesting when the accuracy is getting higher that we know our thought is correct and the modification is toward the right direction. We cannot imagine how much pleasure we will receive when we try to alter inside layers of a model, but it is no doubt fascinating and worth the best try.

8. REFERENCES

[1] Residual Leaing:認識 ResNet 與他的冠名後繼者 ResNeXt 、ResNeSt：
https://medium.com/ai-blog-tw/deep-learning-residual-leaning%E8%AA%8D%E8%AD%98resnet%E8%88%87%E4%BB%96%E7%9A%84%E5%86%A0%E5%90%8D%E5%BE%8C%E7%B9%BC%E8%80%85resnext-resnest-6bedf9389ce

[2] 資料增強 – 我全都要,jpg：
https://ithelp.ithome.com.tw/articles/10273884

[3] ORCHID FLOWER DATASET：
https://ieee-dataport.org/open-access/orchid-flower-dataset#files

[4] Orchid Flowers Dateset：
https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/0HNECY

[5] K Fold Cross Validation with Pytorch and sklearn：
https://medium.com/dataseries/k-fold-cross-validation-with-pytorch-and-sklearn-d094aa00105f

[6] Inception 系 列 – InceptionV2, InceptionV3：
https://medium.com/ching-i/inception-%E7%B3%BB%E5%88%97-inceptionv2-inceptionv3-93cd42054d23

[7] Transfer Learning 轉移學習：
https://medium.com/%E6%88%91%E5%B0%B1%E5%95%8F%E4%B8%80%E5%8F%A5-%E6%80%8E%E9%BA%BC%E5%AF%AB/transfer-learning-%E8%BD%89%E7%A7%BB%E5%AD%B8%E7%BF%92-4538e6e2ffe4

[8] [機器學習 ML NOTE]Overfitting 過渡學習：
https://medium.com/%E9%9B%9E%E9%9B%9E%E8%88%87%E5%85%94%E5%85%94%E7%9A%84%E5%B7%A5%E7%A8%8B%E4%B8%96%E7%95%8C/%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92-ml-note-overfitting-%E9%81%8E%E5%BA%A6%E5%AD%B8%E7%BF%92-6196902481bb

[9] Group Normalization in Pytorch (With Examples)：
https://wandb.ai/wandb_fc/GroupNorm/reports/Group-Normalization-in-Pytorch-With-Examples---VmlldzoxMzU0MzMy

[10] 機器/深度學習:損失函數 (loss function) – Huber Loss 和 Focal Loss：
https://chih-sheng-huang821.medium.com/%E6%A9%9F%E5%99%A8-%E6%B7%B1%E5%BA%A6%E5%AD%B8%E7%BF%92%E6%90%8D%E5%A4%B1%E5%87%BD%E6%95%B8-loss-function-huber-loss%E5%92%8C-focal-loss-bb757494f85e

[11] [機器學習 ML NOTE]SGD, Momentum, AdaGrad, Adam Optimizer：
https://medium.com/%E9%9B%9E%E9%9B%9E%E8%88%87%E5%85%94%E5%85%94%E7%9A%84%E5%B7%A5%E7%A8%8B%E4%B8%96%E7%95%8C/%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92ml-note-sgd-momentum-adagrad-adam-optimizer-f20568c968db

[12]（轉）優化時該用 SGD, 還是用 Adam?
https://blog.csdn.net/S20144144/article/details/103417502

[13] Swin Transformer：
https://huggingface.co/docs/transformers/model_doc/swin

[14] RESNEST：
https://pytorch.org/hub/pytorch_vision_resnest/

[15] [精進魔法] Regularization：減少 Overfitting，提高模型泛化能力：
https://ithelp.ithome.com.tw/articles/10203371

[16] nadermx/backgroundremover：
https://github.com/nadermx/backgroundremover

[17] TRANSFER LEARNING FOR COMPUTER VISION TUTORIAL：
https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html

[18] pytorch 中的 pre-train 函數模型引用及修改：
https://blog.csdn.net/whut_ldz/article/details/78845947

[19][Day 19] 自動化機器學習 - AutoML：

https://ithelp.ithome.com.tw/articles/10275842

[20][手把手教學] 快速利用 AutoML Vision 提升 PCB 瑕疵檢測精準度：https://ikala.cloud/cloud-automl-vision-aoi-pcb-defect-detection/

[21][手把手教學] 快速啟動 Cloud AutoML Vision ：Google 最新機器學習產品：https://ikala.cloud/cloud-automl-vision-quick-start/

[22]christianversloot/machine-learning-articles ：https://github.com/christianversloot/machine-learning-articles/blob/main/how-to-use-k-fold-cross-validation-with-pytorch.md

[23] pytorch-K 折交叉驗證過程說明及實現 ：https://blog.csdn.net/foneone/article/details/104445320

[24] 交叉驗證(Cross-validation, CV) ：https://chih-sheng-huang821.medium.com/%E4%BA%A4%E5%8F%89%E9%A9%97%E8%AD%89-cross-validation-cv-3b2c714b18db

[25] 交叉驗證：https://zh.m.wikipedia.org/zh-tw/%E4%BA%A4%E5%8F%89%E9%A9%97%E8%AD%89

[26] Python 基礎資料視覺化—Matplotlib：https://medium.com/@yuhsuan_chou/python-%E5%9F%BA%E7%A4%8E%E8%B3%87%E6%96%99%E8%A6%96%E8%A6%BA%E5%8C%96-matplotlib-401da7d14e04

[27] [Python]資料視覺化 M05─運用 matplotlib 完成多圖同時呈現：https://ithelp.ithome.com.tw/articles/10211489

[28]【python】如何將 matplotlib 的標題置于圖片下方：https://ithelp.ithome.com.tw/articles/10211489

[29] Python 讀取與寫入 CSV 檔案教學與範例：https://blog.gtwang.org/programming/python-csv-file-reading-and-writing-tutorial/

[30] TBrain AI 實戰吧：https://tbrain.trendmicro.com.tw/

[31] 尋找花中君子 - 蘭花種類辨識及分類競賽：https://tbrain.trendmicro.com.tw/Competitions/Details/20

[32] 使用 Grad-CAM 解釋卷積神經網路的分類依據：https://medium.com/%E6%89%8B%E5%AF%AB%E7%AD%86%E8%A8%98/grad-cam-introduction-d0e48eb64adb

[33] Class Activation Map methods implemented in Pytorch ：https://github.com/jacobgil/pytorch-grad-cam

[34] 如何在 PyTorch 上使用 GradCAM 進行神經網路分類依據視覺化?：https://yanwei-liu.medium.com/pytorch-with-grad-cam-6a92a54bfaad

[35] Grad-Cam 实现流程（pytorch）：https://blog.csdn.net/weixin_41735859/article/details/106474768

[36] 在 Python 中從字串中刪除某些字元：https://www.delftstack.com/zh-tw/howto/python/remove-certain-characters-from-string-python/

[37] 十五分鐘認識正規表達式，解決所有文字難題：https://5xruby.tw/posts/15min-regular-expression

[38] Regex 使用 ^ 符號排除特定字元：https://matthung0807.blogspot.com/2017/08/caret.html

[39] regular expression 如何 match「不含指定字串的任意字串」？：https://social.msdn.microsoft.com/Forums/zh-TW/53373cdd-e03f-4fed-b065-233e658261a2/regular?forum=237

[40] Day 12: 正規表示式(Regular Expression)：https://ithelp.ithome.com.tw/articles/10222163

[41]［料理佳餚］Regular Expression（正則表達式）的比對「不包含」：https://dotblogs.com.tw/supershowwei/2021/07/11/135346