AI CUP 2022 Fall Competition: Crop Image Recognition

TEAM 2016 FROM OPML

Jing-En Hung, Jia-Wei Liao, Yen-Jia Chen,
Yi-Cheng Hung, and Shang-Yen Lee

December 18, 2022

Contents

List of Figures

List of Tables

Abstract

Image classification has been widely used in engineering, agriculture, and medical applications. Nowadays, with the rapid development of deep neural network and the computing power of graphic cards, the performance of image classification has been greatly improved. In particular, pattern recognition plays an important role in image classification. Even though this field looks very mature, there are still some intractable problems, such as lack of labeled data and lack of understanding of species. In this paper, we introduce a pipeline that can find the most suitable process for each task systematically. Using this method, our final scores are 0.9328596 and 0.9344163 in public and private datasets, respectively. In addition, our overall ranking is 8th out of 153 teams. Our source codes are available at https://github.com/Jia-Wei-Liao/Crop_Classification and https://github.com/yenjia/AIdea_crops. The model weights can be downloaded from our Google Drive folder at https://drive.google.com/drive/folders/1cZTBzgOuuf3ms6V__71nWlOXtvmTOTxZ?usp=sharing and https://drive.google.com/drive/folders/1qCnjAqN5TmdW-kPOtgxRGA4cRyJAJgT7?usp=share_link.

Keywords: Fine-grained Classification, Convolution Neural Network, Transformer, Auto-augmentation, Test Time Augmentation, and Grad-CAM.

Chapter 1

Introduction

At present, although there are complete AI data sets in related fields such as people's livelihood, industry, and medical care, they are relatively lacking in the agricultural field. In order to meet the needs of applying AI technology to smart agriculture in the future, it is necessary to invest a large amount of professional manpower in agricultural related information. collection and analysis operations.

Training models to predict crop yield can be a challenging task for a number of reasons. One of the main challenges is that crop growth and yield can be affected by many factors, including weather, soil conditions, and the presence of pests and diseases. This can make it difficult to predict crop yields with high accuracy, as models can ultimately only perform as well as the data they were trained on. Also, the quality of the training data may vary, which also affects the performance of the model.

In the training model phase, we have 89,514 images in our training data for a total of 33 categories. We have 2,000 to 3,000 training images per class at the current phase. Fig. 1.1 show each categories number and Fig. 1.2 demonstrates the part of our training dataset.

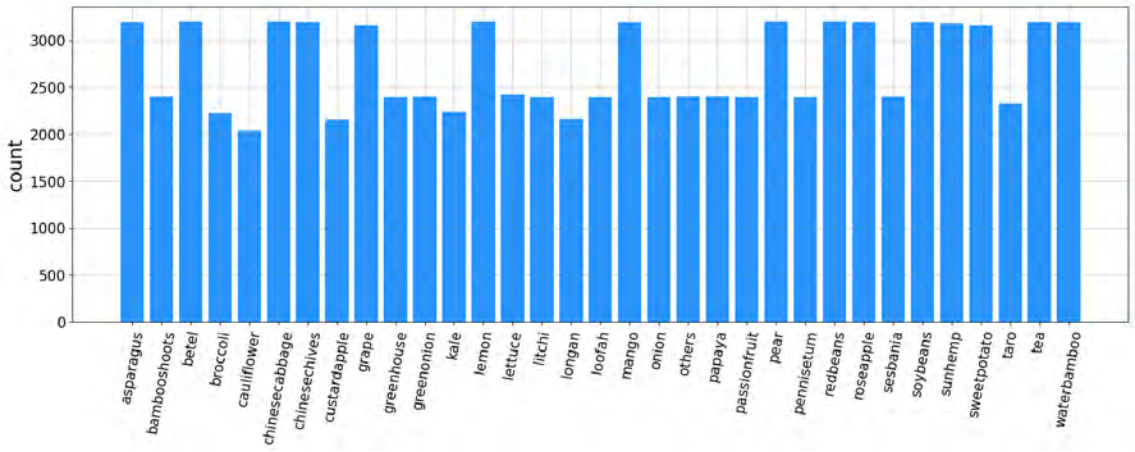


Figure 1.1: Crops images count bar plot.

Figure 1.2: Crops images in the training dataset.

We have a total of 22,308 public and private verification sets. By using an ensemble of models based on CNN and Transformer architectures and voting on their predictions, we are able to achieve results of 0.9328596 and 0.9344163 in public and private, respectively.

Chapter 2

Related Work

2.1 Fine-grained Classification

Fine-grained image classification that focuses on differentiating between hard-to-distinguish object classes and doing the detailed inference [1, 2, 3] are also highly relevant to this competition. Furthermore, there are also lots of literature related to orchid classification tasks including treat extracted feature and color as an important information for orchid types classification task [4] proposed by Pulung Nurtantio Andono et al in, Phylogeny and classification of the orchid family [5] and Features of pollinaria and orchid classification [6] proposed by RL Dressler in 1993 and 1986, respectively, etc.

2.2 ImageNet Pre-training

ImageNet dataset contains 14,197,122 annotated images according to the WordNet hierarchy, is a large-scale database designed for visual object recognition such as image classification and automatic object clustering [7]. This dataset has been used in many research, and the data of many pre-trained models also come from ImageNet.

2.3 Convolutional Neural Network

Convolutional Neural Network (CNN) was proposed in 1990s, but it was limited by the computing power and amount of data at that time. It was not taken seriously until the early 2000s. Basic CNN consists of convolution layer, pooling layer, and fully-connected layer. In later works, there are various CNN-based networks. Such as AlexNet, VGG, and GoogleNet deepened the network architecture. ResNet imports the residual block [8], connecting the input layer and the layer close to the output. Dense Convolutional Network (DenseNet) connects each layer to every other layer in a feed-forward fashion [9]. Increased the depth of the network and improve the accuracy.

2.4 Transformer

The concept of Transformer was proposed by Ashish Vaswani et al. in 2017 [10], which is used to deal with Natural Language Processing (NLP) tasks. The advantage of Transformers is that it can process sequential input data and import attention mechanisms. In later image tasks, the concept of dealing with sequential inputs and attention mechanisms also be applied in various networks, such as Vision Transformer (ViT), Swin Transformer. Figure 2.1 shows architecture of ViT model.

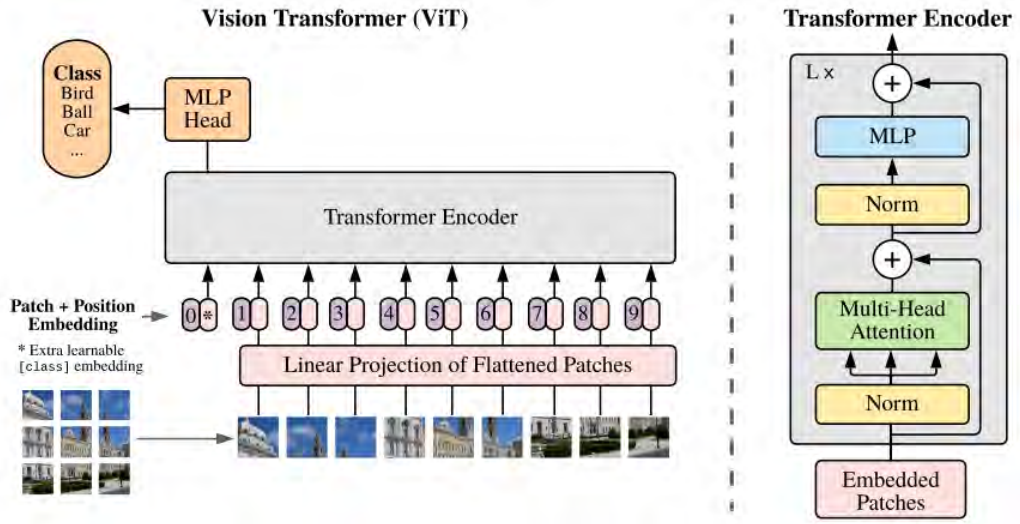


Figure 2.1: Architecture of ViT model.

Chapter 3

Proposed Method

3.1 Data Pre-processing

3.1.1 Data Cleaning

We noticed that there are repeated images in each category, and the overall statistics are about 1,800. Compared with the overall number, we think the impact is not large, so ignore this problem. However, the appearance of the same image in different categories may affect the correctness of the model's judgment. We found that there were 11 pairs of such situations, and these 11 pairs of data were removed from the data set as shown in Fig. 3.1.

Figure 3.1: The left figure is `ae1df7c4-704b-4206-9c94-f093a4390ea2.jpg` in sesbania and right is `e1965d51-6fba-4900-ba5a-b48b41c23e85.jpg` in sunhemp.

3.1.2 Data Splitting

First, we split our training data into three parts, training, validation and testing as shown in Fig. 3.2. At the same time, we ensure that the data distribution in each part is similar. Then we apply hold-out method in Fig. 3.3 and choose several model weights ensemble the results to reduce the variance, so the performance of the model is less sensitive to the single model weights.

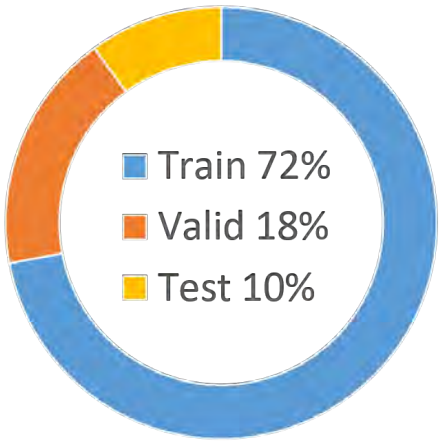


Figure 3.2: The ratio of the training set, validation set, and test set.

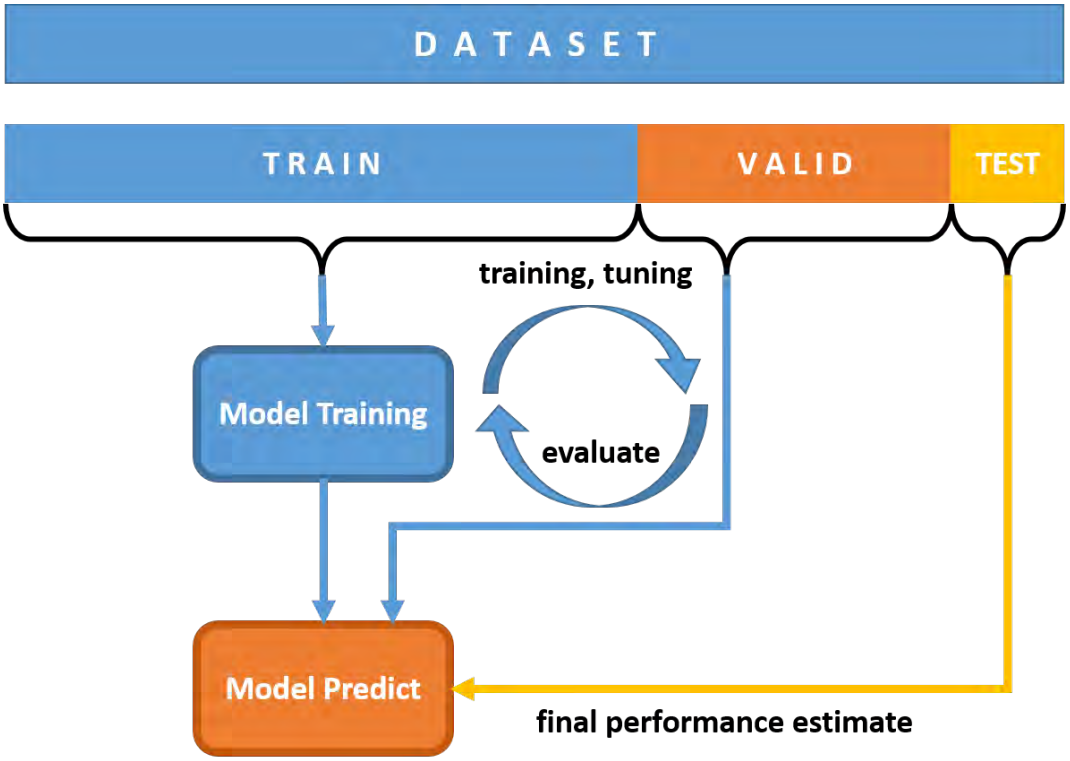


Figure 3.3: An illustration of the hold-out method.

3.1.3 Image Mask

Since the most interested region in color space are focus on color green region, we convert RGB to HSV (hue, saturation and value) representations and extract the value in $35 \leq H \leq 77$, $43 \leq S \leq 255$, and $46 \leq V \leq 255$ constrained.

Figure 3.4: Image and mask represent the interested region.

3.1.4 Image Resize

The image resolution varies significantly, ranging from 700 pixels to over 3000 pixels. In order to maintain consistency in the number of pixels when reading the data, we need to resize it in (1080, 1080) or (512,512).

3.1.5 Image Normalize

Images are represented as tensors with pixel values ranging from 0 to 255. Since the model's weights are pre-trained on ImageNet, we apply the z-score transform to pixel values with the mean of (0.485, 0.456, 0.406) and standard deviation of (0.229, 0.224, 0.224) the same as in ImageNet.

3.1.6 Standard Augmentation

To avoid over-fitting situations and enhance the robustness of the model, we use the following techniques, including

1. **Randomly flipping:** With probability 0.25, we flip the image left and right, and with probability 0.25, we flip the image up and down.
2. **Randomly rotating:** With probability 0.25, we rotate the image counterclockwise by 90 degrees.

3.1.7 Auto-augmentation

Auto-augmentation [11] is the method of adding data based on reinforcement learning, adding an automatic augmentation strategy, and establishing multiple sub-strategies under the strategy to make the data change in different color types or geometric types. We divided the data augmentation transformations into two categories: eight color transformations and five geometric transformations. These transformations were further divided into 25 sub-strategies, and during each mini-batch, sub-policies were selected with uniform probability to augment the data.

Color Space Transform		Geometry Transform
Brightness	Invert	Rotate
Color	Equalize	TranslateX
Contrast	Solarize	TranslateY
Autocontrast	Posterize	ShearX
		ShearY

Table 3.1: Transformations for the auto-augmentation.

3.2 Model Architecture

In the model part, besides ResNet family, we used EfficientNet [12], RegNet [13], and Swin Transformer [14], which are the state-of-the-art (SOTA) models in the image classification task.

3.2.1 EfficientNet

EfficientNet is proposed by Mingxing Tan, Quoc V. Le in 2019. Scaling up convolution networks is widely used to achieve better accuracy such as ResNet [15] scaling up the network's depth (ResNet18 to ResNet152). Wide Residual Network (WRN) [16] scaling up the network's width (increasing the number of output channels). Moreover, scaling up the image resolution also gives better accuracy. The goal of EfficientNet is to scale up depth, width, and resolution simultaneously, as shown in Fig 3.5. The larger the input image, the deeper layer and the more channels are model needs, which is to ensure that the receptive field is large enough to capture more features. And we can see EfficientNet gives a great accuracy, as shown in Fig 3.6.

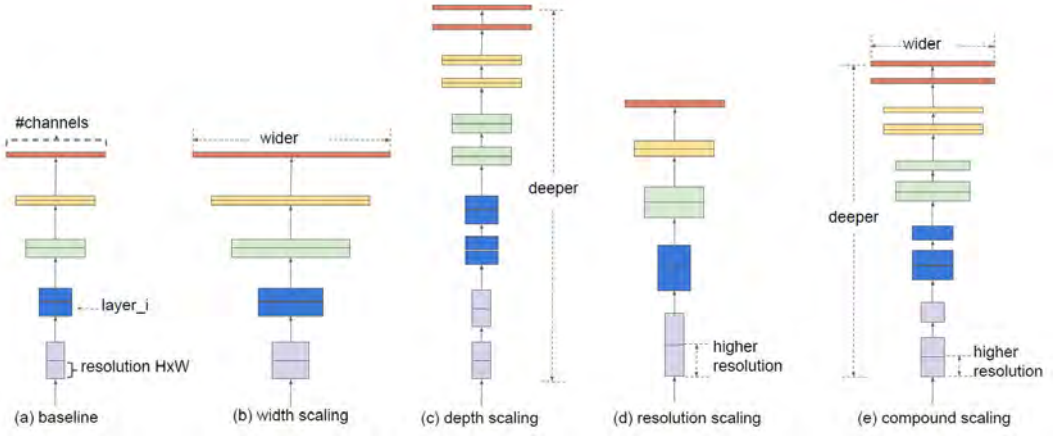


Figure 3.5: An illustration of model scaling.

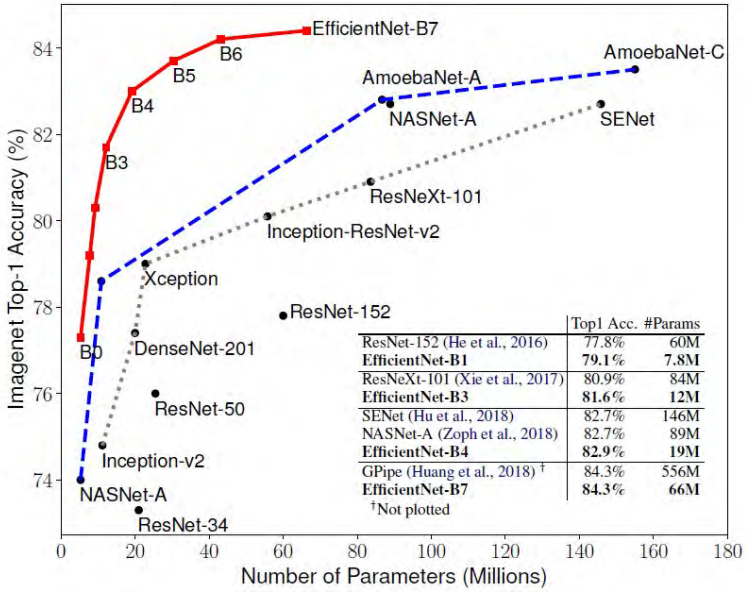


Figure 3.6: Model size versus ImageNet accuracy.

3.2.2 RegNet

RegNet is a network that uses NAS (Network Architecture Search) for a specific task. In prior work, they have designed a process to automatically search the best architecture from the ”design space”, and the architecture mentioned in this paper is one of the searched networks called ”RegNet-Y-16GF”. 16GF means 1.6×10^{10} FLOPs and Y means adding a new SE (Squeeze-and-Excitation) structure to enhance performance. However, this series of works cannot search out a specific network for a specific task with limited resources, so we did not perform an architectural search.

3.2.3 Swin Transformer

Swin Transformer is proposed by Ze Liu in 2021, which achieved the best paper award of ICCV2021. It uses the window-based multi-head self-attention (W-MSA) to reduce linear time complexity. To reinforce the connectivity of different windows, it proposes a window shift mechanism. Figure 3.7 is the architecture of Swin Transformer, there are four stages in the Swin Transformer and each stage is composed of a patch merging layer and two successive Swin Transformer blocks which are the feature extractor and maintain the output size as the input. The patch merging layer uses the inverse pixel shuffle to attain dimension reduction. Its role is as a maximum poling in the CNN model. In Figure 3.8, we show the Swin Transformer block, which is composed of layer normalization (LN), (shift) window-based multi-head self-attention ((S)W-MSA), and multilayer perceptron (MLP). Moreover, it also has the residual mechanism after (S)W-MSA and MLP layers.

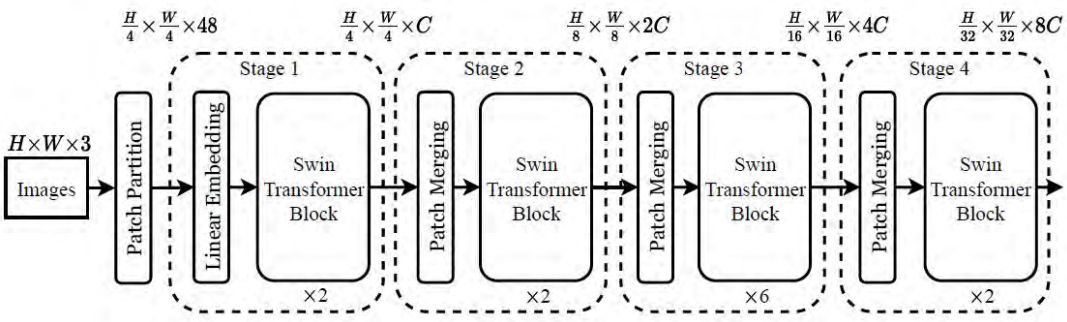


Figure 3.7: Architecture of Swin Transformer.

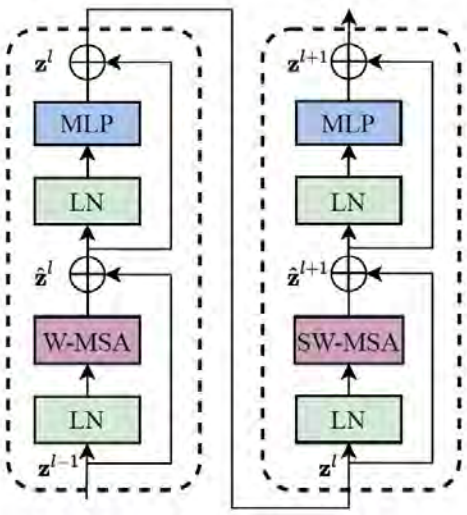


Figure 3.8: Two successive Swin Transformer block.

3.3 Loss Function

For loss functions, we attempt cross-entropy loss (CE), and focal loss (FL).

Cross-entropy loss is the most popular loss function in classification tasks. It can be written in the form

$$\mathcal{L}_{CE}(\hat{y}, y) = -\sum_{i=1}^{c} y_i \log \hat{y}_i,$$

where y is the probability of prediction, $\hat{y}$ is the one-hot label, and c is the number of categories.

Focal loss [17] was proposed in 2018, which is used to solve the data imbalance problem. It is defined as

$$\mathcal{L}_{FL}(\hat{y}, y) = -\alpha(1 - \hat{y}_c)^{\gamma} \log \hat{y}_c,$$

where c is the category of y, $\hat{y}_c$ is the probability of category c in $\hat{y}$ and $\alpha, \gamma \geq 0$ are hyper-parameters.

3.4 Optimization

We mainly use SGD and AdamW [18] as our optimizer. AdamW is a stochastic optimization method that modifies Adam's [19] typical implementation of weight decay by decoupling weight decay from the gradient update. The algorithm is as follows

Algorithm AdamW

Input: objective $f(\theta)$, initial weight θ_0, learning rate η, weight decay rate λ, β_1, β_2, and ε.
Initial: first moment $m_0 \leftarrow 0$, and second moment $v_0 \leftarrow 0$.
for $t = 1$ to ... **do**
  $g_t \leftarrow \nabla_\theta f(\theta_{t-1})$
  $\theta_t \leftarrow \theta_{t-1} - \eta\lambda\theta_{t-1}$
  $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
  $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
  $\hat{m}_t \leftarrow \dfrac{m_t}{1 - \beta_1^t}$
  $\hat{v}_t \leftarrow \dfrac{v_t}{1 - \beta_2^t}$
  $\theta_t \leftarrow \theta_{t-1} - \eta \dfrac{\hat{m}_t}{\sqrt{\hat{v}_t + \varepsilon}}$
end for
return θ_t

To further improve results and let the model converge to the global minimum, we adjust the learning rate by cosine decay with the warm-up method. It represents by the following:

$$\eta(T_{cur}) = \begin{cases} \left(\frac{T_{cur}}{T_{warm\_up}}\right)\eta_{max}, & \text{if } T_{cur} \leq T_{warm\_up} \\ \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})\left(1 + \cos\left(\frac{T_{cur} - T_{warm\_up}}{T_{max} - T_{warm\_up}}\pi\right)\right), & \text{if } T_{cur} > T_{warm\_up} \end{cases},$$

where η_{max} is the maximum learning rate, η_{min} is the minimum learning rate, T_{cur} is the current iteration, T_{warm_up} is the warm up iteration and T_{max} is the maximum iteration.

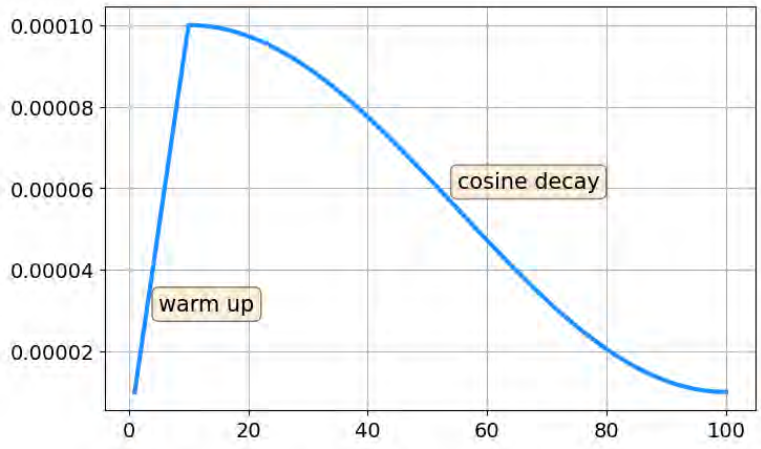


Figure 3.9: Cosine decay with the warm-up method.

Chapter 4

Experimental Results

4.1 Performance Evaluation

Confusion matrix is a tool for visually predicting network performance. As shown in Fig. 4.1, it consists of True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN). In particular, TP is the number of prediction which is predicted positive and the corresponding Ground Truth (GT) is also positive. TN is the number of predictions which is predicted negative and the corresponding GT is also negative. FP is the number of predictions that is predicted positive, and the corresponding GT is negative. FN is the number of predictions which is predicted negative, and the corresponding GT is positive.

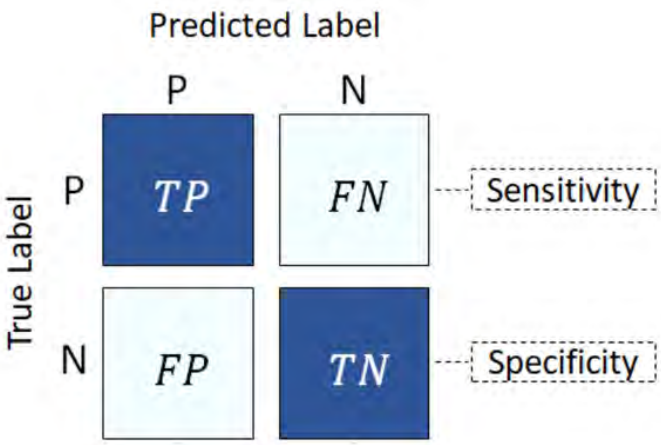


Figure 4.1: An illustration of the confusion matrix.

After knowing the meaning of the confusion matrix, we can calculate the Weighted-Precision (WP), Precision, and Recall by Eq. 4.4, Eq. 4.1, and Eq. 4.2, respectively. Further, we can calculate the F1 score by using the formula as shown in Eq. 4.3. In this competition, WP is used to the evaluation metric and all categories' F1 score must be above 0.7.

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{4.1}$$

$$\text{Recall} = \frac{TP}{TP + FN}, \tag{4.2}$$

$$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{4.3}$$

$$\text{WP} = \frac{1}{N} \sum_{i=1}^{c} (\text{Precision}_i \times (TP_i + FN_i)) \tag{4.4}$$

where c is the number of categories, and N is number of images.

4.2 Validation Results

First, we use the DenseNet121 architecture to set the baseline for this competition. With several combinations of optimizer and learning rate, we can get the best validation accuracy of 0.8373 as shown in Table 4.1, which is a good start. After obtaining the baseline scores, we tried many augmentation methods, deep learning models, optimizers and loss functions and tried to find what combination would give the best performance.

Model	B.S	Size	Loss	Optimizer	L.R.	Best WP
DenseNe-t121	550	384	CE	AdamW	1e-4	0.8373

Table 4.1: Validation accuracy for different models. B.S. denotes the batch size, and L.R. denotes the learning rate.

Table 4.2 shows the validation accuracy for our experiments with different models, batch size, loss function, and learning rate. From the table, we can see that most of the models can achieve good performance during training. During the training, we found that increasing the size of the image helped to improve the performance, but it also brought some problems: training time, GPU memory usage, etc. will increase significantly. In addition, we found that Auto Augmentation can help increase performance and reduce overfitting. Lastly, we also find that even though the performance of each model does not vary much, the results show that different models are good at different categories. Therefore, we aggregated the results of all models by ensemble, and the performance was improved a bit more.

Model	Fold	B.S.	Size	Loss	Optimizer	L.R.	Best WP	Public Score
EfficientNet-B0	0	20	1080	FL	AdamW	3e-4	0.9037	0.9167
EfficientNet-B0	0	10	1080	CE	SGD	3e-4	**0.9065**	**0.9192**
EfficientNet-B0	1	10	1080	CE	SGD	3e-4	0.9039	0.9123
EfficientNet-B0	2	10	1080	CE	SGD	3e-4	0.9005	0.9182
EfficientNet-B0	3	10	1080	CE	SGD	3e-4	0.9024	0.9168
EfficientNet-B0	4	10	1080	CE	SGD	3e-4	0.8791	-
EfficientNet-B0	5	10	1080	CE	SGD	3e-4	0.8787	-
RegNet-Y-16GF	0	256	512	FL	AdamW	1e-4	0.9047	0.9158
Swin-S	0	32	1080	FL	AdamW	1e-4	0.8971	0.9117
Swin-B	0	192	512	FL	AdamW	1e-4	0.8952	0.9140
Swin-B	0	24	1080	FL	AdamW	1e-4	0.9030	0.9164

Table 4.2: The public score and private score for different models. B.S. denotes the batch size, and L.R. denotes the learning rate.

Based on the results in Table 4.2, we selected the top 4 EfficientNet models and the remaining four models (RegNet, Swin-S, Swin-B-512, and Swin-B-1080) for ensembling. This resulted in a public score of 0.9328596 and a private score of 0.9344163, achieving ranks of 9 and 8, respectively.

4.3 Discussion

4.3.1 Analysis of Validation Results by Using t-SNE

t Distributed Stochastic Neighborhood Embedding (t-SNE) helps visualize classification results. Fig. 4.2 shows the results of the Swin-B model on the validation dataset with an accuracy of about 0.91. Different colors in the figure represent different categories, and it is obvious that the model has effectively classified the data into the correct category. This shows that the predictive performance of the model is strong.

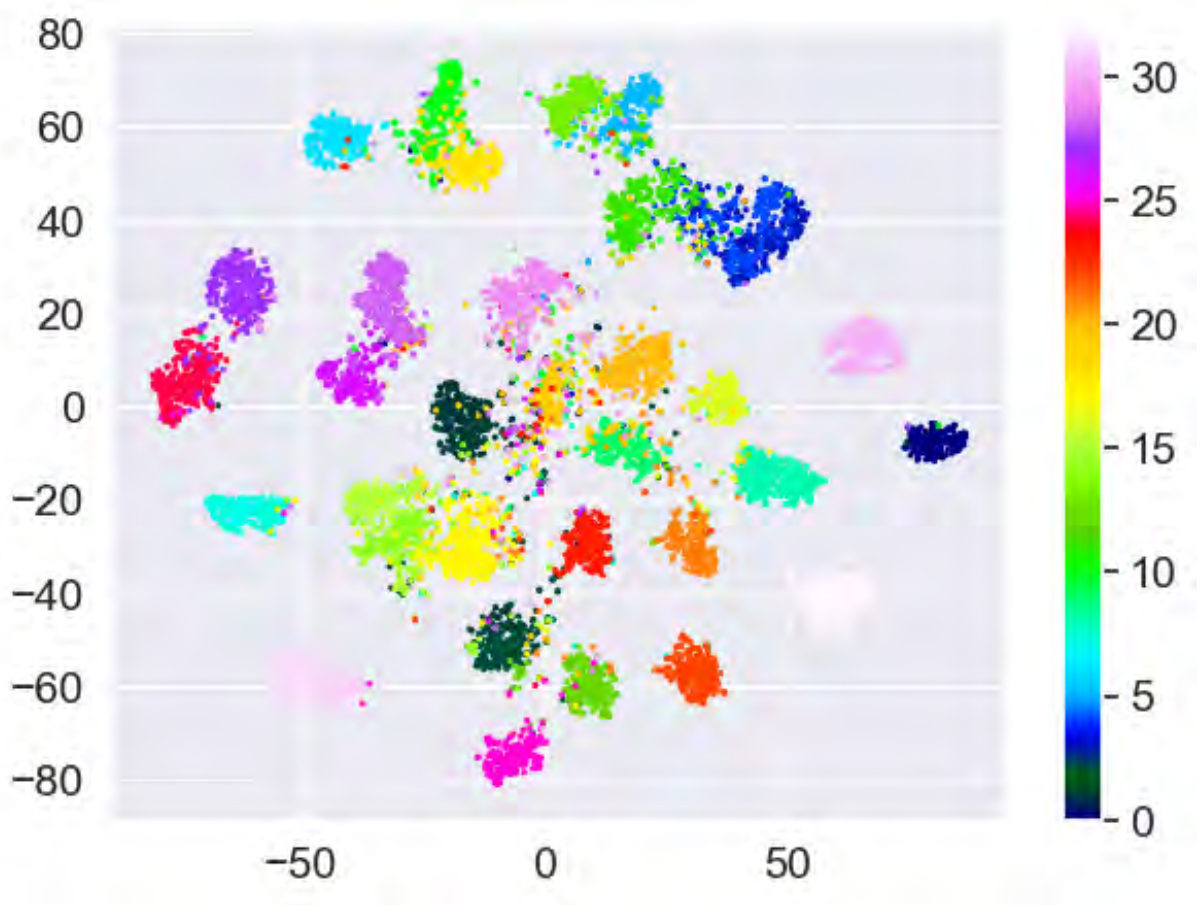


Figure 4.2: t-SNE for single model results.

4.3.2 Analysis of Testing Results by Using Correlation Matrix

RegNet Result

Fig. 4.3 shows the inference results of the RegNet model on the test dataset, with an accuracy of about 0.90.

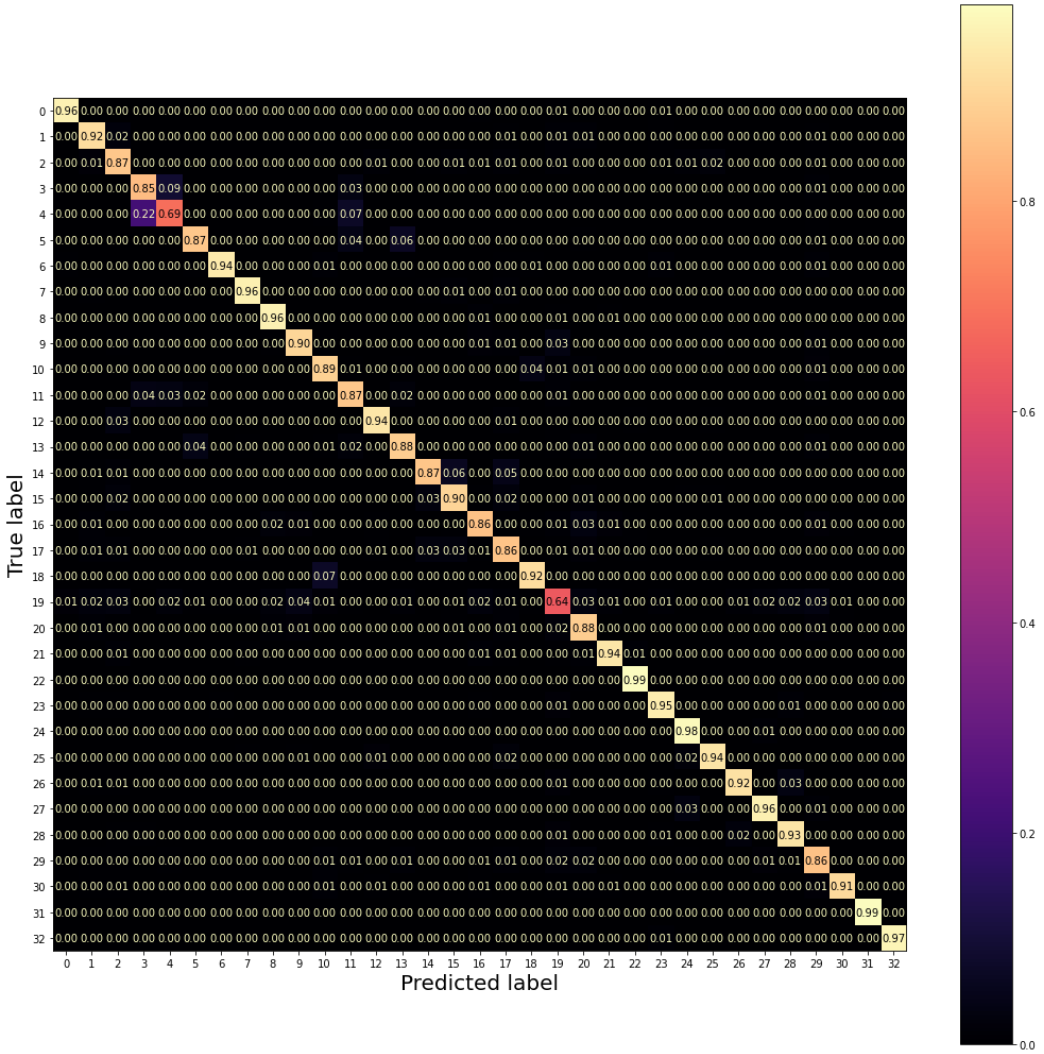


Figure 4.3: Correlation matrix for RegNet.

Ensemble Result

Fig. 4.4 shows the correlation matrix of the ensemble results for the three models. Here, we use the mean ensemble method for each prediction by using the corresponding probability. The test accuracy can be raised to 0.91 in our test dataset.

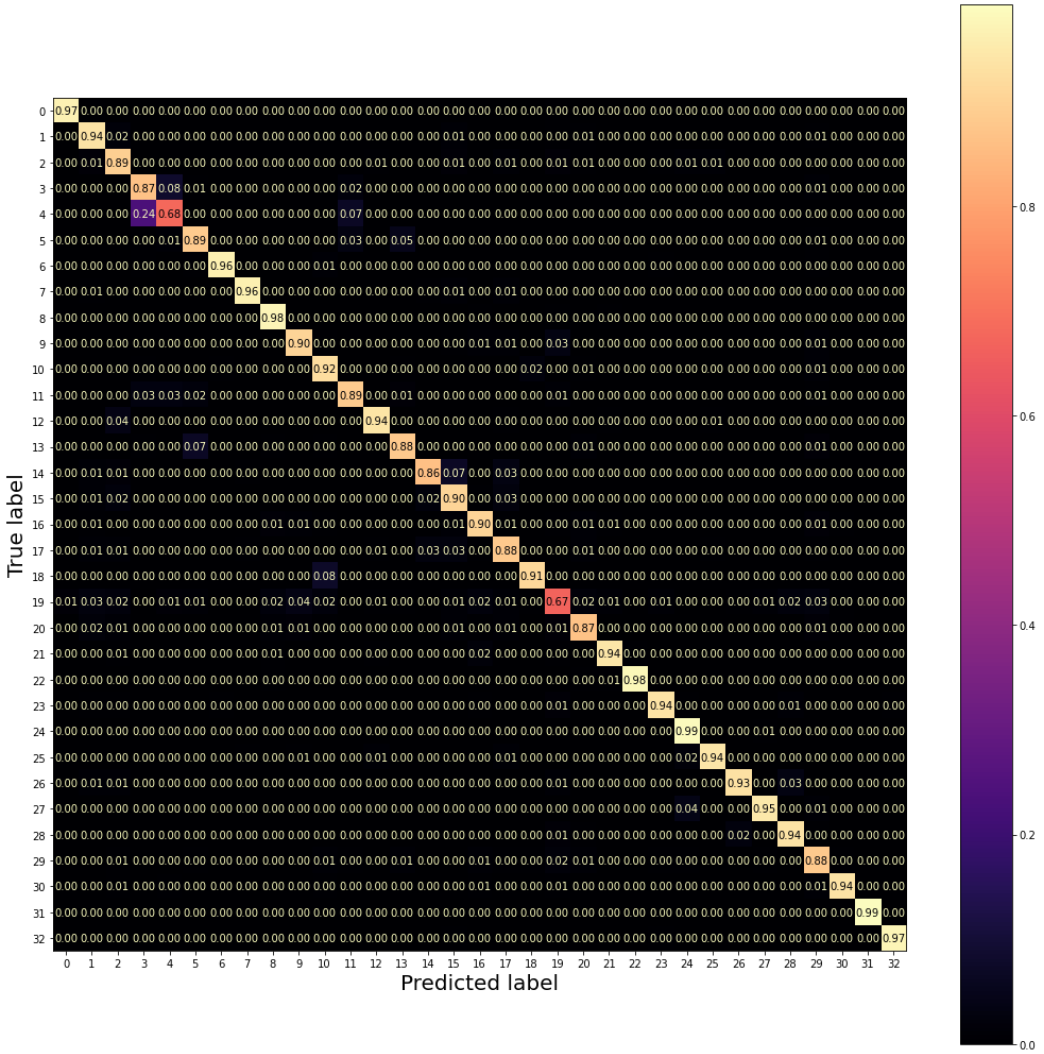
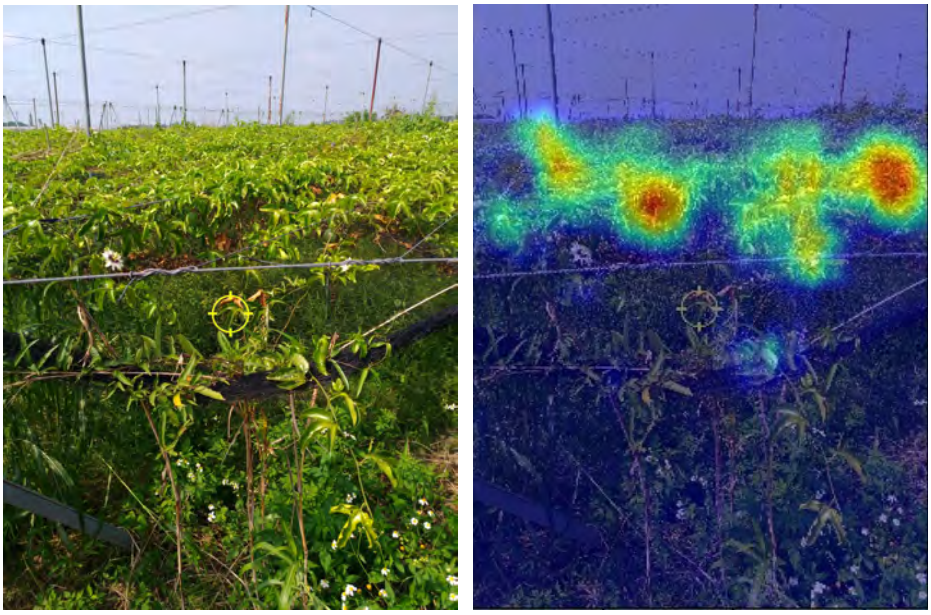


Figure 4.4: Correlation matrix for ensemble results of three models (Swin-S, Swin-B, and RegNet).

Compared to the two correlation matrices above, we can find that the categories of prediction errors are not the same before and after doing the ensemble. Therefore, we think that the results can be improved a lot if the ensemble technique is handled well.

4.3.3 Visualization of Prediction Result s by Using Grad-CAM

We also draw the Gradient Cam and observe whether the region for model doing the inference is reasonable. Fig. 4.5 shows an example of gradient cam which focus on the specify region. In the below case, The model's prediction is for the category of "passion-fruit", and upon visually inspecting the image, it can be determined that the flowering part, as well as the immature green fruit in the image, are part of the passionfruit. The Grad-CAM also focuses on the passionfruit tree, and the redder area corresponds to the fruit area in the original image, indicating that the model has successfully captured the focus.



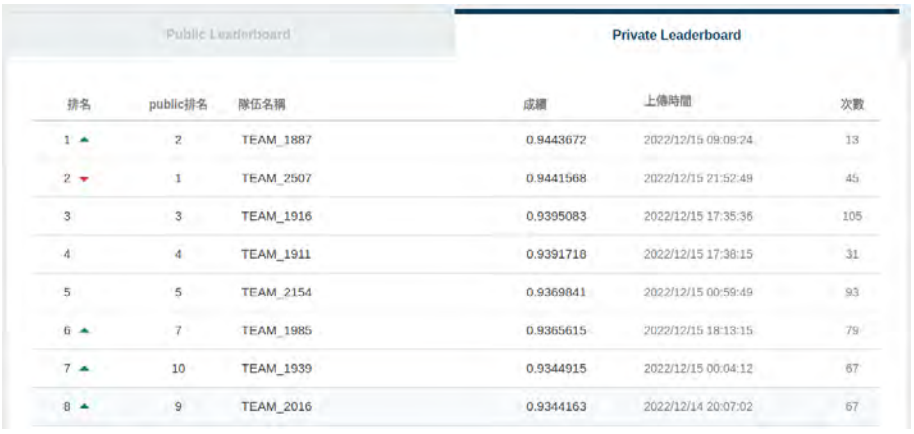
(a) Original image (b) Grad-CAM image

Figure 4.5: The Grad-CAM of image `033005e5-fb60-43ed-8e96-deb6f7d0137f.jpg` in public dataset.

Chapter 5

Conclusion

In this competition, we explored a variety of models, applied auto-augmentation techniques to diversify the dataset, and trained with CNN base models such as EfficientNet and Transformer-base like Swin model. The experimental results showed that in large data and using SGD as the optimizer had better convergence than AdamW. We also found that the higher the image resolution, the better the training performance as measured by the WP score. In addition, we observed that Swin performed better on lower resolution data compared to CNN base, while CNN base showed better training efficiency and performance on high resolution data. Subsequently, we verified the stability of the model and the advantage of ensemble combination by applying test-time augmentation (TTA) and ensemble to both public and private datasets. Finally, we achieved a public ranking of 9th and a private ranking of 8th, corresponding to scores of 0.9328596 and 0.9344163, respectively.


Public Leaderboard Private Leaderboard

排名	public排名	隊伍名稱	成績	上傳時間	次數
1 ▲	2	TEAM_1887	0.9443672	2022/12/15 09:09:24	13
2 ▼	1	TEAM_2507	0.9441568	2022/12/15 21:52:49	45
3	3	TEAM_1916	0.9395083	2022/12/15 17:35:36	105
4	4	TEAM_1911	0.9391718	2022/12/15 17:38:15	31
5	5	TEAM_2154	0.9369841	2022/12/15 00:59:49	93
6 ▲	7	TEAM_1985	0.9365615	2022/12/15 18:13:15	79
7 ▲	10	TEAM_1939	0.9344915	2022/12/15 00:04:12	67
8 ▲	9	TEAM_2016	0.9344163	2022/12/14 20:07:02	67

Figure 5.1: Aidea leaderboard.

Bibliography

[1] Ze Yang, Tiange Luo, Dong Wang, Zhiqiang Hu, Jun Gao, and Liwei Wang. Learning to navigate for fine-grained classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 420–435, 2018.

[2] Michael Fleischman and Eduard Hovy. Fine grained classification of named entities. In *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002.

[3] Peiqin Zhuang, Yali Wang, and Yu Qiao. Learning attentive pairwise interaction for fine-grained classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13130–13137, 2020.

[4] Pulung Nurtantio Andono, Eko Hari Rachmawanto, Nanna Suryana Herman, and Kunio Kondo. Orchid types classification using supervised learning algorithm based on feature and color extraction. *Bulletin of Electrical Engineering and Informatics*, 10(5):2530–2538, 2021.

[5] Robert L Dressler. *Phylogeny and classification of the orchid family*. Cambridge University Press, 1993.

[6] ROBERT L Dressler. Features of pollinaria and orchid classification. *Lindleyana*, 1(2):125–130, 1986.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[9] Laurens van der Maaten Gao Huang, Zhuang Liu and Kilian Q. Weinberger. Densely connected convolutional networks. *CVPR*, 2017.

[10] Vaswani A. Shazeer N. Parmar N. Uszkoreit J. Jones L. Gomez A. N. ... Polosukhin I. Attention is all you need. *arXiv:1706.03762*, 2017.

[11] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

[12] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[13] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollar. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[14] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[16] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[17] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, 2017.

[18] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Appendix A

Environment and Device

A.1 Operating System

Our system is Ubuntu 22.04, which is a Linux distribution based on Debian and composed mostly of free and open-source software. Further, we use Python and Bash, which is a Unix shell and command language written by Brian Fox for the GNU Project as a free software replacement for the Bourne shell as our programming language.

A.2 Hardware Information

The following list contains our hardware information.

CPU	GPU
Intel(R) Xeon(R) Gold 6342 CPU @ 2.80GHz	GeForce GTX 3090
Intel(R) Xeon(R) Gold 5220	NVIDIA Quadro RTX8000
Intel(R) Xeon(R) Silver 4210R CPU	NVIDIA RTX A6000

A.3 Third Party Libraries

In this competition, we have utilized the following third-party libraries.

Library	Version
scikit-learn	0.21.3
numpy	1.18.5
opencv-python	4.6.0.66
Pillow	6.2.0
scipy	1.4.1
torch	1.13.0+cu116
torchvision	0.14.0+cu116
torchaudio	0.13.0
tqdm	4.55.1
tensorboard	2.11.0
pandas	1.2.0
PyYAML	5.1.2

(a) **Jia-Wei-Liao**'s repository with Python 3.7.4.

Library	Version
scikit-learn	1.0.1
numpy	1.22.3
torch	1.13.0+cu117
torchvision	0.14.0+cu117
torchaudio	0.13.0
torchmetrics	0.6.1
pandas	1.3.4
monai	0.9.0
torch-tb-profiler	0.3.1
pytorch-lightning	1.6.5
lightning-bolts	0.5.0
PyYAML	6.0

(b) **yenjia**'s repository with Python 3.9.7.

Table A.1: Version of our using libraries.

Appendix B

Reproduce the Best Result

In this section, we demonstrate how to reproduce the best result.

B.1 Jia-Wei-Liao's Repository

1. Download the Repository:

```
git clone https://github.com/Jia-Wei-Liao/Crop_Classification.git
```

2. Download the model weights from google drive and move into the assigned place in folder structure.

3. Setup Environment: Run the following commend

```
conda create --name crop_cls python=3.7.4
source activate crop_cls
pip install -r requirements.txt
```

4. Reproduce the Result:
   Enter `Crop_Classification` folder and execute the following code

```
bash script/infer.sh
```

Then you can get the predictions csv file in `submission` directory.

B.2 yenjia's Repository

1. Download the Repository:

```
git clone https://github.com/yenjia/AIdea_crops.git
```

2. Download the model weights from google drive and move into the assigned place in folder structure.

3. Setup Environment: Run the following commend

```
conda create --name crop_cls_2 python=3.9.7
source activate crop_cls_2
pip install -r requirements.txt
```

4. Reproduce the Result:

   (a) Give the config file path in `infer_public.py`

   Note that there is a line in `infer_public` that needs to be modified for usage.

   ```
   data_list = json.load(open("../datalist/public_private.json"))
   ```

   Please give the correct JSON file for the data to be inferred

   (b) The JSON file is list structure. Each element is dict structure in list in the format `{"image": image_path}`. image_path should be replaced by the **absolute path**. The example datalist is in the datalist folder. (public_private.json)

   Enter `AIdea_crops/command` folder and execute the following code

```
source reproduce.sh
```

Then you will get the predictions csv file in `submission` directory.

B.3 Merge Submission Files

In order to reproduce the result, moving 4 csv files from yenjia's repository `private` to Jia-Wei-Liao's repository `Crop_Classfication/submission`. Then run the following command, then you can get the `result.csv` file, which is our final submission.

```
python generate_merge_csv.py
```

Appendix C

Contact Information

C.1 Team Information

Team Name	Public Score	Public Rank	Private Score	Private Rank
TEAM_2016	0.9328596	9 / 153	0.9344163	8 / 153

C.2 Team Member

Name	Institution	Phone	Email
黃靖恩 * (Jing-En Huang)	國立臺灣師範大學數學系 (National Taiwan Normal University Mathematics)	0927619756	40840210s@gapps.ntnu.edu.tw
廖家緯 (Jia-Wei Liao)	國立臺灣大學資訊工程研究所 (National Taiwan University Computer Science and Information Engineering)	0939580280	d11922016@csie.ntu.edu.tw
陳彥嘉 (Yen-Jia Chen)	國立臺灣大學資料科學學程 (National Taiwan University Data Science Degree Program)	0911412486	r10946007@ntu.edu.tw
洪翊誠 (Yi-Cheng Hung)	國立陽明交通大學應用數學所 (National Yang Ming Chiao Tung University Applied Mathematics)	0958618583	yicheng.sc10@nycu.edu.tw
李尚宴 (Shang-Yen Lee)	國立臺灣師範大學數學系 (National Taiwan Normal University Mathematics)	0978361975	40740211s@gapps.ntnu.edu.tw

* team leader.

C.3 Advisor and Practitioner

Name	Institution	Position	Email
樂美亨 (Mei-Heng Yueh)	國立臺灣師範大學數學系 (National Taiwan University University Mathematics）	副教授	yue@ntnu.edu.tw
陳育熙 (Yu-Hsi Chen)	國立陽明交通大學應用數學所 (National Yang Ming Chiao Tung University Applied Mathematics)	研究助理	yuhsi44165.sc09@nycu.edu.tw