

Image Classification of Plant Pathology 2021

Jinyi Shang

Yu Cao

The George Washington University

Content

1. Introduction.....	1
2. Description of Dataset.....	3
2.1 Overview of Dataset.....	3
2.2 Exploratory Data Analysis.....	4
2.2.1 Duplicates Values.....	4
2.2.2 Encoder.....	4
2.3 Data Expansion.....	5
3. Model and Algorithm.....	7
3.1 VGG Model.....	7
3.2 ResNet Model.....	8
3.3 Ensemble Learning.....	10
4. Experimental Setup.....	12
4.1 Create Data Pipeline.....	12
4.1.1 Transform Class Stup.....	12
4.1.2 Customized Dataset Class Setup.....	12
4.2 Model Configuration.....	13
4.3 Metrics.....	14
4.4 Hyper-parameter.....	15
4.5 Over-fitting.....	16
5. Results.....	17
5.1 VGG16.....	17
5.2 ResNet50.....	17
5.3 Ensemble.....	19
6. Summary and Conclusions.....	21
7. Reference.....	22
Appendix.....	24

1. Introduction

Apples are one of the most important temperate fruit crops in the world. They contain a lot of vitamin C. It can improve the body's immunity, as well as reduce blood pressure, if people eat it regularly. As a result, apples have become a daily staple for many people. This not only provides huge benefits for consumers, but also for growers. However, foliar (leaf) diseases pose a major threat to the overall productivity and quality of apple orchards. This has the potential to cause huge losses for growers.

The U.S. apple industry, annually worth \$15 billion, experiences millions of dollars in annual losses due to various biotic and abiotic stresses, ongoing stress management, and multi-year impacts from the loss of fruit-bearing trees. Over the growing season, apple orchards are under constant threat from a large number of insects, as well as fungal, bacterial, and viral pathogens, particularly in the northeastern United States. Depending on the incidence and severity of infection by diseases and insects, impacts range from unappealing cosmetic appearance, low marketability, and poor quality of fruit, to decreased yield or complete loss of fruit or trees, causing huge economic losses. Early pest and disease detection are critical for appropriate and timely deployment of disease and pest management programs.

Currently, disease and pest detection in commercial apple orchards relies on manual scouting by crop consultants and service providers. Unfortunately, very few experienced scouts are available, forcing them to cover many large orchards within a narrow time frame. Scouts require a great deal of expertise and training before they can be efficient and accurate in diagnosing an orchard. Generally, they are first trained using images of disease symptoms and insect damage, but due to the presence of a great number of variables in an actual orchard, they need considerable time to familiarize themselves with the many symptom classes caused by either the age and type of infected tissues or the stage of the disease or pest cycle, as well as by changing weather, geographical variances, and cultural differences. Many symptoms of diseases, pests, and abiotic stresses in an apple orchard are distinct enough to differentiate based on visual symptoms alone. However, several disease symptoms look similar enough to each other that it is difficult to accurately determine their cause. At the same time, visual symptoms of a single disease or particular insect can vary greatly between apple varieties, due to differences in leaf color, morphology, and physiology. In addition to the time spent in the orchard, a scout spends a significant amount of time on each client, entering the scouting report, interpreting results, and providing recommendations for action. Overall, human scouting is usually time consuming, expensive, and in some cases, prone to errors.

In recent years, digital imaging and machine learning have shown great potential to speed up plant disease diagnosis. In this report, we use deep learning to train models to categorize apple plant pathology categories. Transfer learning method is selected here. We apply pre-trained model approach to use parts of the pre-trained model. The models include VGG model and ResNet model in this research. These two models are developed by a challenging image classification task, ImageNet 1000-class photograph classification competition. These models can take days or weeks to train on modern hardware. They can be downloaded and incorporated directly into new models that expect image data as input. In order to improve the performance of models, some approaches are applied, such as deleting duplicates, encoding, transforming, image augmentation and so on.

We are going to break everything into logical steps that allow us to ensure the cleanest, most realistic data for our model to make accurate predictions from:

1. Load Data and Packages in Kaggle
2. Preprocessing of Dataset
 - 2.1. Analyzing
 - 2.2. Removing Duplicates
 - 2.3. Data Expansion
3. Modeling and Categorizing
 - 3.1. VGG16 Model
 - 3.2. ResNet50 Model
 - 3.3. Ensemble Model

2. Description of Dataset

2.1 Overview of the dataset

We obtained data from Plant Pathology 2021-FGVC8 challenge competition in Kaggle. Plant Pathology 2020-FGVC7 challenge competition had a pilot dataset of 3,651 RGB images of foliar disease of apples. For Plant Pathology 2021-FGVC8, dataset has significantly increased the number of foliar disease images and added additional disease categories. The dataset contains approximately 23,000 high-quality RGB images of apple foliar diseases, including a large expert-annotated disease dataset. This dataset reflects real field scenarios by representing non-homogeneous backgrounds of leaf images taken at different maturity stages and at different times of day under different focal camera settings.

The train.csv file contains two columns, image and labels. Image indicates the image ID and labels represents the target classes, a space delimited list of all diseases found in the image.

The original train.csv file contains 18632 images. The detailed number of each disease is listed in Table 1.

Table 1. Counts of Each Disease

Disease	Counts
scab	4826
healthy	4624
frog_eye_leaf_spot	3181
rust	1860
complex	1602
powdery_mildew	1184
scab frog_eye_leaf_spot	686
scab frog_eye_leaf_spot complex	200
frog_eye_leaf_spot complex	165
rust frog_eye_leaf_spot	120
rust complex	97
powdery_mildew complex	87

It is clearer to see the number and proportion of different categories in a pie chart. The dataset has series imbalance problem. Disease like scab and healthy plant contain almost half of all images, while diseases like powdery_mildew contains only 6.35% and some other categories containing more than one disease even account for less than 1%. So, we will try some methods to deal with it.

Label distribution

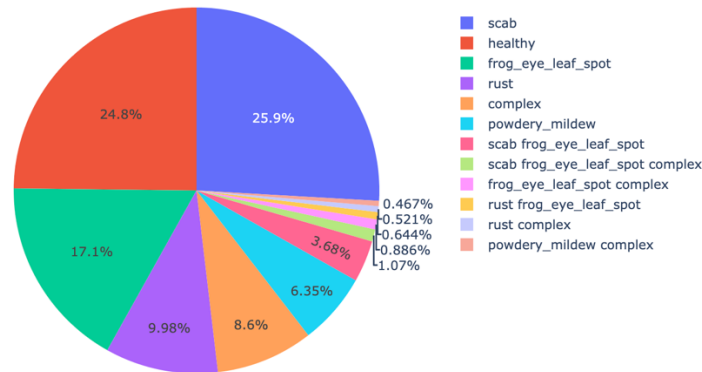


Figure 1. Pie Chart of Categories

2.2 Exploratory Data Analysis

2.2.1 Duplicates Values

Duplicates are always harmful for training process: differently labeled duplicates produce noise in the dataset, while equally labeled duplicates lead to data leakage. Duplicate data will affect the statistical results and mislead decision makers. In the dataset, there are more than 50 duplicates.

In this report, we used the result of a notebook in Kaggle to solve duplicates (<https://www.kaggle.com/nickuzmenkov/pp2021-duplicates-revealing/output>). In his notebook, firstly he saved downsampled images to boost performance because of long time taken by computing hash over original images of very high quality. Then, he used hash to find duplicated images and save them into a csv file. We use this csv file directly to get a dataset without duplicates.

2.2.2 Encoder

In this report, we take different targets as different labels instead of using multi-labels. That is label 'complex' as one category, label 'frog_eye_leaf_spot' as one category, and label 'frog_eye_leaf_spot complex' as one category. So, there are total of 12 categories in the dataset, including 'complex', 'frog_eye_leaf_spot', 'frog_eye_leaf_spot complex',

‘healthy’, ‘powdery_mildew’, ‘powdery_mildew complex’, ‘rust’, ‘rust complex’, ‘rust frog_eye_leaf_spot’, ‘scab’, ‘scab frog_eye_leaf_spot’, ‘scab frog_eye_leaf_spot complex’. Thus, the labels are encoded into 0 to 11 depending on different categories.

Figure 2 gives the example of the 12 categories image:

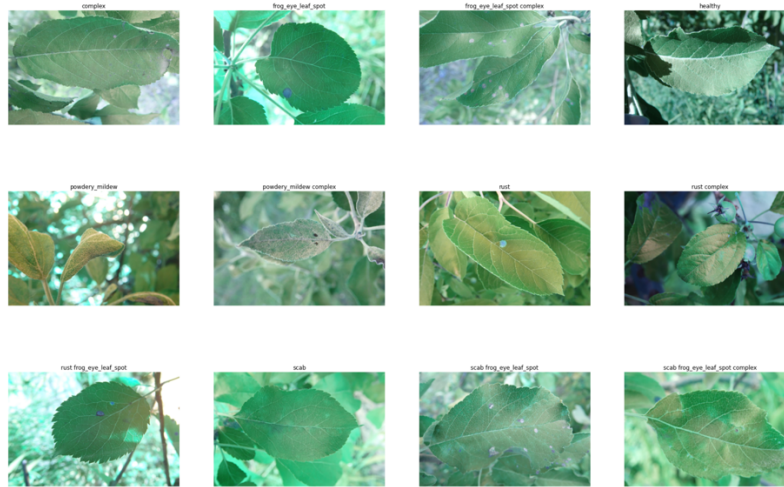


Figure 2. Examples of 12 Categories' Images

3. Model and Algorithm

3.1 VGG Model

VGG is the brainchild of the Visual Geometry Group in Oxford. This network is the related work in ILSVRC 2014, the main work is to prove that increasing the depth of the network can affect the final performance of the network to a certain extent. VGG has two structures, namely VGG16 and VGG19. There is no essential difference between the two, only the network depth is different.

One improvement over AlexNet is that VGG16 uses several 3x3 convolution cores in a row instead of the larger convolution cores in AlexNet (11x11, 7x7, 5x5). Using a stacked small convolution kernel is better than using a large convolution kernel, because multiple nonlinear layers can increase the depth of the network to ensure learning more complex patterns at a lower cost (fewer parameters).

In short, in VGG, 3 3 by 3 convolution kernels are used to replace 7 by 7 convolution kernels, and 2 3 by 3 convolution kernels are used to replace 5 by 5 convolution kernels. The main purpose of this is to improve the depth of the network and the effect of the neural network to some extent under the condition of ensuring the same perceptual field.

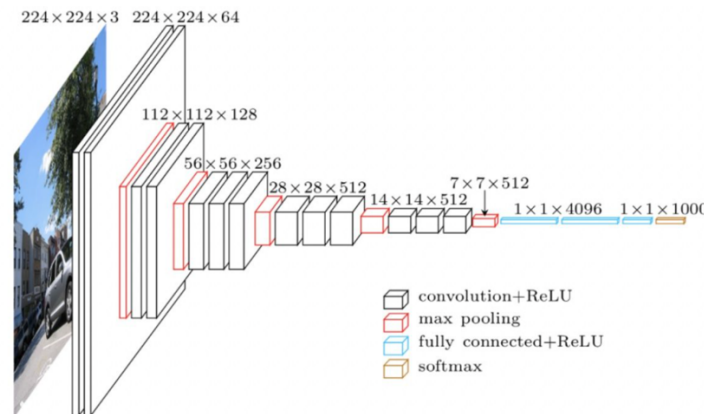


Figure 3. Structure of VGG16 Model

Figure 3 shows the simple structure of VGG16 model. The model can be simply divided into five stages, each layer is composed of two convolutional cores pooled, and is finally connected by three layers for classification.

As the Figure 3 and Table 2 show, processing of VGG16 is: firstly, input the picture of $224 \times 224 \times 3$, and make two convolution and ReLU function through 64 3×3 convolution cores, and the size of the convolution becomes $224 \times 224 \times 64$. Then, max pooling is carried out, and the pooling unit size is 2×2 (the effect is to halve the image size), and the pooled

size is changed to 112x112x64. Then, double convolution and ReLU function through 128 3x3 convolution kernels, and the size becomes 112x112x128. After this, 2x2 max pooling is conducted, and the size is changed to 56x56x128. Then, make three convolution and ReLU function through 256 3x3 convolution kernel, and the size becomes 56x56x256, followed by 2x2 max pooling, and the size is changed to 28x28x256. Then, through 512 3x3 convolution kernels for three times and ReLU function, the size becomes 28x28x512. And conduct 2x2 max pooling and change the size to 14x14x512. After that, through 512 3x3 convolution kernels for three times and ReLU function, the size becomes 14x14x512. Conduct 2x2 max pooling and change the size to 7x7x512. Finally, full connection with layer 2 1x1x4096 and layer 1 1x1x1000 and RELU function and output 1000 forecast results through Softmax.

Table 2. Architecture of VGG Model

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

3.2 ResNet Model

The proposal of Deep Residual Network (ResNet) was a milestone in the history of CNN images.

From experience, the depth of the network is crucial to the performance of the model. When the number of network layers is increased, the network can carry out more complex feature pattern extraction, so better results can be obtained theoretically when the model is deeper.

The experiment found that the depth network presented a degradation problem: with the increase of the depth of the network, the network accuracy appeared saturation, or even decreased. The problem is clearly not caused by overfitting, because not only does the test error become higher as the network deepens, so does its training error. This can be seen from Figure 4.

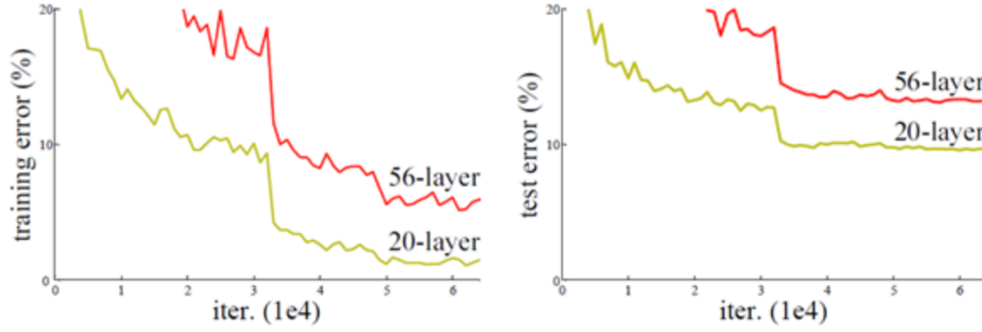


Figure 4. Degradation Problem

When the number of layers of the traditional neural network increases from 20 to 56, both the training error and the test error of the network increase significantly, that is to say, the performance of the network degenerates significantly with the increase of the depth. ResNet was created to solve this degradation problem.

Residual learning is proposed to solve the degradation problem. For a stacking layer structure, when the input is x , the feature learned is noted as $H(x)$. Now we hope that it can learn residual $F(x) = H(x) - x$, so that the original learning feature is actually $F(x) + x$. The reason for this is that it is easier to learn residuals than to learn the original features directly. When the residual is 0, the stack layer only does the identity mapping, and at least the network performance will not decrease. In fact, the residual will not be 0, which also makes the stack layer learn new features on the basis of input features, so as to have better performance. The structure of residual learning is shown in Figure 5. This is a kind of "short" in an electric circuit, so it's a shortcut connection.

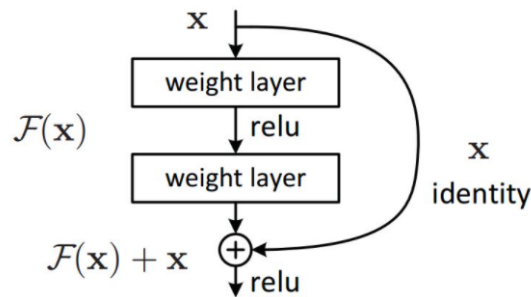


Figure 5. The Basic Unit of Residual Study

In this report, we use ResNet50 model. Table 3 shows the architecture of it. There are five big convolutional layers. The first layer consists of a convolution layer with 64 7 by 7 filters and 2 stride and a subsampling layer of max pooling. Then, there are four big convolutional layers. They consist of 9 layers, 12 layers, 18 layers and 9 layers respectively. Finally, there is a fully connected layer.

Table 3. Architecture of ResNet

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

4. Experimental Setup

4.3 Metrics

Because our validation dataset is imbalance. So, accuracy is not an appropriate metrics to be used here. And because the competition uses mean f1-score. So, we will use f1-score as our metrics to judge the performance.

First, we introduce two concepts: Precision = True Positive / (True Positive + False Positive) and Recall = True Positive / (True Positive + False Negative). Recall is defined as the ratio of the total number of correctly classified positive examples divide to the total number of positive examples. Precision can be calculated by dividing the total number of correctly classified positive examples by the total number of predicted positive examples.

$F1 = 2 * (Precision * Recall) / (Precision + Recall)$. Now that the dataset is imbalanced, f1 score might be a better measure to use if we need to seek a balance between precision and recall.

4.4 Hyper-parameter

In this model, we use ReduceLROnPlateau() function in torch.optim.lr_scheduler to determine training parameter, learning rate. When the evaluation index of the network is not improved, the network performance can be improved by reducing the learning rate of the network.

5. Results

5.2 ResNet50

For the Resnet50 results, we plot the train_val accuracy, train_val loss and learning rate decay graphs and here are the results:

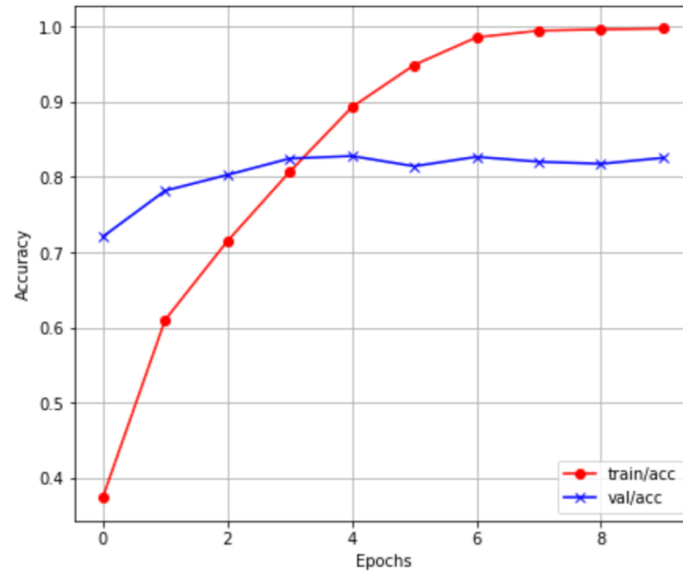


Figure 7. Accuracy of Train and Validation Datasets

Figure 7 shows the accuracy lines of train and validation datasets. We can see that the accuracy of both train and validation datasets increase as the epoch increase. And at about the fifth epoch, the accuracy of validation becomes stable and not increase obviously. The accuracy of validation dataset passes 0.8.

Figure 8 represents the loss lines of train and validation datasets. The loss of train and validation dataset decrease as the epoch increase. And the loss of validation dataset also become stable when it reaches the fifth epoch. It is less than 0.6.

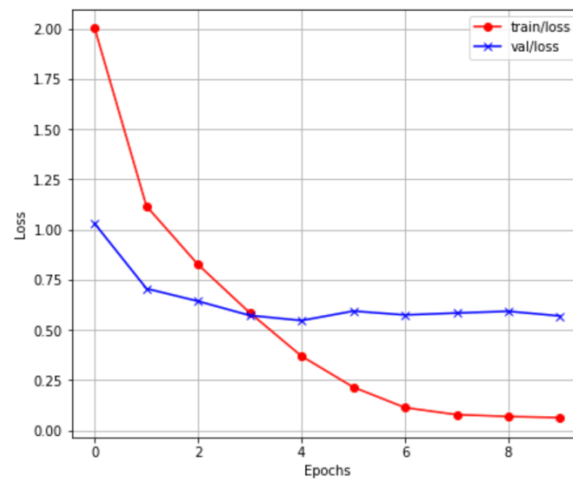


Figure 8. Loss of Train and Validation Datasets

The validation loss is 0.57, f1 score of it is 0.82, which performs well. And here is the best local result:

```
val Loss: 0.5716 Acc: 0.8248
The f1 score is 0.8215232721433382
Learning Rate is 1e-05
```

And here is the learning rate plot:

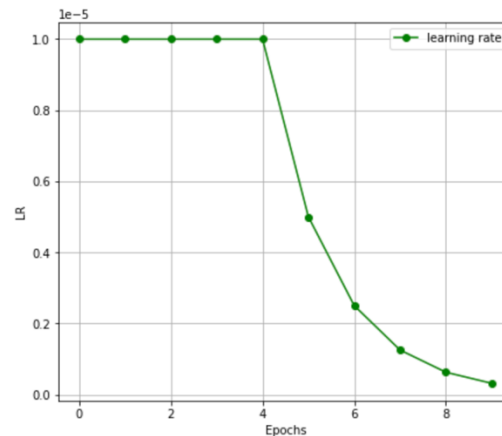


Figure 9. Learning Rate

Figure 9 shows the plot of learning rate. It decreases from the fourth epoch as epochs increase. So, the model might be overfitting since the fifth epoch. So, we submit the model with 5 epochs into Kaggle to train on test dataset. However, the score on the leader board is low:

fi_ressub (version 6/9) a day ago by Yu Cao18 From Notebook [fi_ressub]	Succeeded	0.257
--	-----------	-------

The score is only 0.257. Then, we tried several methods to figure this problem. We applied the second expanded dataset to the model. Because the first one has a lot of transformation, which will generate many noises. This might be the reason why the result in test dataset is not very well. And in local, the loss of validation dataset is 1.1496 and f1 score is 0.649. The result on leaderboard is only 0.197, less than the first score. So, change the expanded dataset didn't solve this issue.

Then, we also take the depth of the model into consideration. If the model is too deep, the result might be not good. So, we tried ResNet18 model as a substitute to decrease the data complexity and the result is worse. The local validation dataset loss is 1.3389 and f1 score is 0.600. And the f1 score on the leaderboard is 0.119. So, the depth of the model might still not the reason why the model performs bad on test dataset. So, this is a direction that we can continue to study in the future.

7. References

- [1] “Plant Pathology 2021 - FGVC8.” Kaggle, www.kaggle.com/c/plant-pathology-2021-fgvc8.
- [2] Brownlee, Jason. “A Gentle Introduction to Transfer Learning for Deep Learning.” Machine Learning Mastery, 16 Sept. 2019, machinelearningmastery.com/transfer-learning-for-deep-learning/#:~:text=Transfer learning is a machine,model on a second task.&text=Common examples of transfer learning,your own predictive modeling problems.
- [3] “13.2. Fine-Tuning¶ Colab [Mxnet] Open the Notebook in Colab Colab [Pytorch] Open the Notebook in Colab Colab [Tensorflow] Open the Notebook in Colab.” 13.2. Fine-Tuning - Dive into Deep Learning 0.16.3 Documentation, d2l.ai/chapter_computer-vision/fine-tuning.html.
- [4] Huilgol, Purva. “Accuracy vs. F1-Score.” Medium, Analytics Vidhya, 24 Aug. 2019, medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2.
- [5] Nickuzmenkov. “PP2021 - Duplicates Revealing.” Kaggle, Kaggle, 6 Apr. 2021, www.kaggle.com/nickuzmenkov/pp2021-duplicates-revealing.
- [6] Kuboko. “PP2021_PyTorch_VGG-16_Fine-tune_Inference.” Kaggle, Kaggle, 25 Mar. 2021, www.kaggle.com/kuboko/pp2021-pytorch-vgg-16-fine-tune-inference.
- [7] Thapa, R., Zhang, K., Snaveley, N., Belongie, S., & Khan, A. (2020). The Plant Pathology Challenge 2020 data set to classify foliar disease of apples. *Applications in Plant Sciences*, 8(9). doi:10.1002/aps3.11390
- [8] Compendium of Apple and Pear Diseases and Pests, Second Edition. (2014). doi:10.1094/9780890544334
- [9] Bessin, R., P. McManus, G. Brown, and J. Strang. 1998. Midwest tree fruit pest management handbook. University of Kentucky, Lexington, Kentucky, USA
- [10] Judd, G. J., A. L. Knight, and A. M. El-Sayed. 2017. Development of kairomone-based lures and traps targeting *Spilonota ocellana* (Lepidoptera: Tortricidae) in apple orchards treated with sex pheromones. *Canadian Entomologist* 149: 662–676.
- [11] Li, X., S. Geng, H. Chen, C. Jung, C. Wang, H. Tu, and J. Zhang. 2017. Mass trapping of apple leafminer, *Phyllonorycter ringoniella* with sex pheromone traps in apple orchards. *Journal of Asia-Pacific Entomology* 20: 43–46.

- [12] Barbedo, J. G. A. 2014. An automatic method to detect and measure leaf disease symptoms using digital image processing. *Plant Disease* 98: 1709–1716.
- [13] Dai, P., X. Liang, Y. Wang, M. L. Gleason, R. Zhang, and G. Sun. 2019. High humidity and age-dependent fruit susceptibility promote development of *Trichothecium* black spot on apple. *Plant Disease* 103: 259–267.
- [14] Mahlein, A.-K. 2016. Plant disease detection by imaging sensors: Parallels and specific demands for precision agriculture and plant phenotyping. *Plant Disease* 100: 241–251.
- [15] Praveengovi. (2021, April 19). Plant Pathology-Detail-EDA Pytorch. Retrieved from <https://www.kaggle.com/praveengovi/plant-pathology-detail-eda-pytorch>

Appendix

GitHub Repo:

- ❑ Final-Project-Group2 (<https://github.com/Yu-Cao2019/Final-Project-Group2>)
 - ❑ Group-Proposal/
 - ❑ Group_Proposal.pdf
 - ❑ Final-Group-Project-Report/
 - ❑ Final_Group_Project_Report.pdf
 - ❑ Final-Group-Presentation/
 - ❑ Final_Group_Presentation.pdf
 - ❑ Code/
 - ❑ README.md
 - ❑ eda-plant-pathology-2021.ipynb
 - ❑ failed-ensemble.ipynb
 - ❑ model-combination.ipynb
 - ❑ model-plant-pathology-2021.ipynb
 - ❑ new-eda.ipynb
 - ❑ resnet.ipynb
 - ❑ ressub.ipynb
 - ❑ val-aug.ipynb
 - ❑ vgg-v1.ipynb
 - ❑ Jinyi-Shang-Individual-Report /
 - ❑ Jinyi-Shang-final-project.pdf
 - ❑ Code/
 - ❑ RF.py
 - ❑ GBR.py
 - ❑ README.md
 - ❑ Yu-Cao-Individual-Report/
 - ❑ Yu-Cao-final-project.pdf
 - ❑ Code/
 - ❑ RandomizedSearchCV_Neural_Network.py
 - ❑ Neural Network.py
 - ❑ README.md
 - ❑ README.md