Cassava Leaf Disease Classification: Final Report

G026 (s2041285, s2099657)

Abstract

Cassava, which is a popular kind of crop in Africa. It could be used in many areas and withstand harsh farming conditions. However, there are some types of diseases that make poor harvest. To cope with this problem, we need to identify the type of cassava disease and remedy it. Nowadays, image classification techniques could be applied to help identify cassava disease. Recently, using image classification to detect cassava disease has been attached great importance as a Kaggle competition. We selected this topic as our project, in which, firstly we tried several common single-type deep neural networks including ResNet, DenseNet, ResNext, EfficientNet B3 and Vision Transformer. Then based on their performance, we did ensembling classification using models with the top three highest validation accuracy. It shows that ensembling model performs much better than single-type models. Based on the ensembling model, we also investigated several generalization and regularization methods including Fmix, however, experiment results show that these methods fail to improve the validation performance.

1. Introduction

Cassava is an important food security crop in Africa and at least 80% of African farmers grow Cassava. Cassava has many uses, for example, lots of cassava based dishes are consumed in Africa, besides, cassava could also be processed to ingredients such as animals' feed and glue. Although Cassava has the strong ability to tolerate harsh farming conditions, there are various types of cassava leaf diseases resulting poor harvest. To identify cassava leaf disease types using traditional human inspection methods requires much labor efforts and is time consuming. Thus, many researchers apply data science methods to automate this disease classification problem based on the photos taken by farmers. However, different photo qualities and few photos for some type of disease are obstacles to improve the classification accuracy. To address this, a Kaggle competition called **Cassava Leaf Disease Classification** has been held many years. In this project, we used this competition as our research topic and developed deep neural networks for Cassava leaf disease classification.

This project aims to apply deep learning algorithm to do cassava disease classification. The first part of our work is to train some single-type models such as ResNet (He et al., 2016), EfficientNet (Tan & Le, 2019) and ResNext (Xie et al., 2017), DenseNet (Huang et al., 2017) and vision transformer (Dosovitskiy et al., 2020). Inspired by (Chen et al., 2020), in which deep transfer learning for plant disease detection was proposed and validated to be efficient and effective in classifying plant images, we used the pre-trained models for parameter initialization and fine-tuned on our dataset.

The second part in our project is to pick three single-type models with best validation performance and ensemble them to do classification. We also tried different weights for the three models in classification. Based on the ensembling model with best weight settings, we investigated several generalization and regularization techniques such as Fmix (Harris et al., 2020) and label smoothing (Müller et al., 2019). Finally, we picked the model with the best validation performance and used it for test accuracy.

To avoid the models from being overfitting to training dataset, image augmentation techniques are used. In this project, some basic image augmentation techniques are applied such as image rotation, flip and RandomCrop. Moreover, in test phase, test time augmentation (TTA) is used to create several augmented copies of each image, then an average of the prediction for each copy is used as the final decision. The objectives of our project could be concluded as follows:

- To investigate the performance of some popular single-type deep learning image classification models in this cassava disease classification task and analyse their validation performance.
- To investigate the performance of ensembling models of the single-type models, and to compare the validation performance of ensembling models with single-type models.
- To further explore some data augmentation methods such as Fmix and regularization techniques such as label smoothing.

The rest of this report has 5 sections. Section 2 describes the dataset of our task, as well as training and testing procedures. Section 3 gives an overview of the methodology. Experiment details, results and analysis are stated in Section 4. Section 5 states related work for plant disease classification and finally, section 6 concludes the work.

2. Data set and task

2.1. Data Set

In the experiments, the dataset used is cassava-leaf-classification dataset (cassava leaf disease dataset). The dataset contains 21,397 labeled images of cassava plant, these images were collected from photos taken by farmers in Uganda.

There are five classes in this dataset and each class represents a disease type or healthy state. The images in this dataset are of size 800×600. The five classes are: Cassava Bacterial Blight (CBB) as class 0, Cassava Brown Streak Disease (CBSD) as class 1, Cassava Green Mottle (CGM) as class 2, Cassava Mosaic Disease (CMD) as class 3 and Healthy as class 4. Figure 1 shows the example images of each class from the dataset. Table 1 gives the number of training images from each class. We can notice that the classes are unbalanced. The number of images for class 3 is more than a half of total images, which is much more than images of the other four classes.

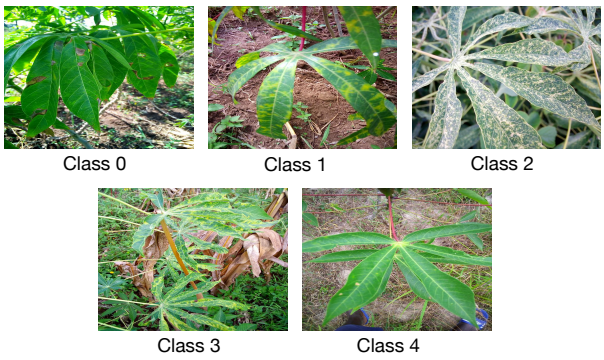

Figure 1. Examples of images with five classes. The top three images are of class 0 (CBB), class 1 (CBSD), class 2 (CGM). The bottom two images are of class 3 (CMD) and class 4 (Healthy).

Total Num	class 0	class 1	class 2	class 3	class 4
21,397	1087	2189	2386	13158	2577

Table 1. Number of Training Images of Each Class.

2.2. Training and testing procedures

Firstly, we used **StratifiedKFold** to divide the dataset into five folds beforehand. Then, one fold is selected randomly as validation dataset and the other four folds are used as training dataset. Since the test dataset provided by Kaggle is not open for public analysis, we measured the performance of models based on the validation dataset. At last, the performance of final model was tested on the public and private board provided by Kaggle as test accuracy.

3. Methodology

3.1. Data Processing

Data augmentation, as a regularizer to help mitigate overfitting, is adopted in this project. We directly use some functions provided in the **Albumentations** library. In precise, a sequence of data augmentation methods is implemented sequentially for each image. In this project, pre-processing of images needs to be done in training. For each input image, a region with a specified size is cropped randomly within the original image. Then for the cropped image, operations such as transpose, flip operations, rotate, and normalize are implemented to make training images various, which could enhance the generalization ability of a model.

3.2. Model Architecture

The architecture of the vision model in this project is as shown in figure 2. Firstly, singlt-type vision models such as Vision Transformer (Dosovitskiy et al., 2020), EfficientNet (Tan & Le, 2019) are explored. Secondly, ensemble learning is adopted to combine these single-type vision models to improve the classification ability since ensemble modeling aggregates multiple models and could reduce the generalization error. Considering the limited computation resources, we will implement ensembling in testing phase, in precise, the weighted average of the single-type model's prediction is used as the final decision.

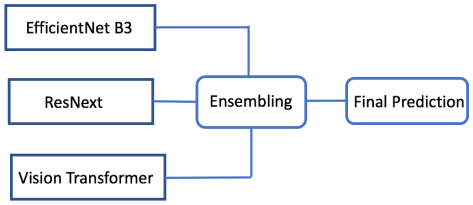

Figure 2. Ensembling Network.

3.3. Transfer Learning

Transfer learning is to apply pre-trained models to specific tasks. Pre-trained models are usually trained on large amounts of images such as ImageNet (Deng et al., 2009) and thus could learn common characteristics. Deep transfer learning is validated to be efficient and effective in classifying plant images (Chen et al., 2020), as a result, we use the pre-trained models for parameter initialization and fine-tune on our dataset.

3.4. Vision Transformer

Vision Transformer (ViT) (Dosovitskiy et al., 2020) is adapted from Transformer (Vaswani et al., 2017), which is used in natural language processing (NLP) tasks. The structure of ViT is shown in figure 3. Unlike Transformer in NLP, ViT only contains the encoder, in particular, firstly input image is split into a sequence of patches and embedded with position embeddings, then the resulting sequence of vectors are fed into a standard Encoder of the Transformer. Experiment results validate that ViT could achieve excellent performance without any convolutional neural network module.

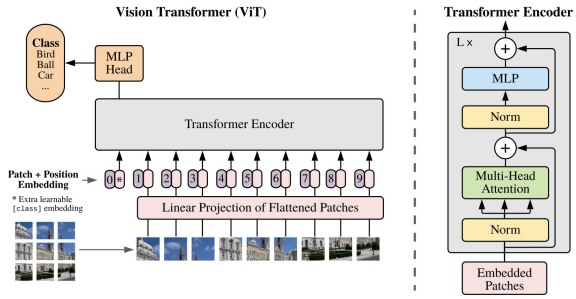

Figure 3. Structure of Vision Transformer (Dosovitskiy et al., 2020)

3.5. EfficientNet

For convolutional neural networks, scaling up is widely used for better accuracy. Making neural networks deeper by using more layers is a common method. For example, using ResNet with 18 layers to ResNet with 101 layers. Moreover, making width larger and increasing resolution are also two scaling up methods. Usually when we try these scaling methods, we need to find the parameters manually, which is tedious and not sure to reach the optimum state. EfficientNet (Tan & Le, 2019), which aims to scale all dimensions of depth, width and resolution using a simple and effective coefficient. For example, if computational resources is 2^N times more, we could increase the network depth by α^N, width by β^N and image resolution by θ^N, where α, β, θ are constant coefficients decided on original model.

3.6. ResNext

ResNext (Xie et al., 2017) adopts a multi-branch architecture, in precise, it aggregates a set of transformations with the same topology. Moreover, ResNet is adopted in this architecture to enhance the ability of the network to converge. Experiments show that increasing the number of sub-branch is a more effective way to improve the capacity and performance compared to increasing depth and width of the neural network.

3.7. ResNet

Residual Network is firstly proposed in (He et al., 2016). Performance tends to get saturated and might degrade in deep neural network, to address this bottleneck, ResNet proposes to fit a residual mapping by adding identity mappings. To implement residual learning, several layers in the network comprise a block, in which a shortcut connection from the input to the output is built and basic element-wise addition is implemented to combine the input and output. It has been validated through the experiments that deep residual learning could break the bottleneck of saturated performance and achieve higher accuracy in many downstream tasks, moreover, residual learning also have good generalization performance.

3.8. DenseNet

Densely connected convolutional network is proposed in (Huang et al., 2017), as an improved version of deep residual learning. As the CNN gets deeper, information flow is prone to vanishing, thus DenseNet passes the output of each layer to all subsequent layers, which could bring several advantages, including less overall parameters, improved flow of gradients through the network and better regularizing effect. It is validated through experiments that DenseNet could achieve high accuracy with considerably less parameters due to the reuse of feature maps, it could also mitigate overfitting on dataset such as SVHN, ImageNet, CIFAR.

3.9. Techniques

In order to improve the performance of the final classifier, the following techniques are used during training or inference. In this subsection, several techniques are introduced as follows:

- **Test Time Augmentation**(TTA) is adopted. TTA creates several augmented copies of each image during test phase, then an average of the prediction for each copy is used as the final decision. By implementing TTA, classification accuracy could be improved since classification errors may be averaged out from prediction. TTA is especially useful when the model is not that confident.

- **Fmix** (Harris et al., 2020), a popular data augmentation method. Fmix firstly generates random binary masks by applying a threshold to low frequency images, and then combines two images and corresponding labels using this binary mask. Since the binary mask could take on multiple shapes and Fmix does not affect training time, Fmix is widely used. Fmix is used in data processing phase, thus the model with Fmix applied should be re-trained. Figure 4 shows an example of combining two initial images (the left and center images) into a new image (the right one) with Fmix.

Figure 4. An example of combining two images by using Fmix. The left image is the image to be trained, it is combined with the center image using Fmix. The image on the right is the generated image, which is used during training.

- **Label Smoothing** (Müller et al., 2019) could be used to prevent the network from becoming over-confident. In particular, when calculating the loss of the model, a weighted average of the cross-entropy loss and the uniform distribution over predicted labels are used as the final loss.

4. Experiments

This section contains experiments on four parts: hyper-parameter settings, experiments on single-type models, ensemble models and additional training techniques. The first part states hyper-parameter settings in training. All the models have the same hyper-parameter settings to make results comparable. The second part compares several deep neural networks on validation accuracy. The networks investigated are Vision Transformer (Dosovitskiy et al., 2020), EfficientNet b3 (Tan & Le, 2019), ResNext (Xie et al., 2017) and DenseNet (Huang et al., 2017). The third part is about ensembling the single-type models, which is to select several single-type models with top validation performance, and use the weighted average of single-type models' scores as the final decision. The final part of the experiments tries to explore generalization techniques including Fmix (Harris et al., 2020) and regularization method such as label smoothing (Müller et al., 2019), aiming to see whether these techniques are helpful to improve accuracy.

4.1. Hyper-parameter Settings

In the experiment, we used the same hyper-parameters for consistent test results and comparable analysis. In data processing, a sequence of data operations {RandomResizedCrop, Transpose, HorizontalFlip, VerticalFlip, ShiftScaleRotate, HueSaturaionValue, RandomBrightnessContrast, Normalize, CorseDropoutCutout, Cutout} from **albumentations** library with specified probabilities was adopted. During training, we used a batch size of 16, and the Adam optimizer with initial learning rate of 1e-4 and minimum learning rate of 1e-6. We trained 10 epochs based on shuffled data for all models. For vision transformer, the input size is 384×384. For other CNN models in the experiment, the input size is 512×512. During testing phase, we adopted Test Time Augmentation (TTA) to improve the performance, the number of image copies we used was 4.

4.2. Experiments on single type neural networks

The experiment in this section aims to explore overall and single-class performance on single-type models. We picked ResNet (He et al., 2016) as the baseline model. In addition, we also picked DenseNet (Huang et al., 2017), EfficientNet B3 model (Tan & Le, 2019), ResNext (Xie et al., 2017) and vision transformer (Dosovitskiy et al., 2020) for comparison. These models are trained using the pre-trained models in **timm** package[1]. We used **resnet101** architecture for ResNet, **vit_base_patch16_384** architecture for vision transformer, **tf_efficientnet_b3_ns** architecture for EfficientNet model, **densenet201** architecture for DenseNet and **ig_resnext101_32x8d** architecture for ResNext. The output layer of these models are changed to the number of classes in this task which is 5. Figure 5 shows the training curves of the top three models with best performances which are EfficientNet B3, ResNext and Vision transformer.

[1]https://github.com/rwightman/pytorch-image-models/tree/master/timm/models

For each type of model, we trained 10 epochs. From the figure 5, it could be seen that the models could achieve quick convergence in the first two epochs, which means that using pre-trained models could fit into our task more quickly. Finally we picked the models of the epoch with the largest validation accuracy.

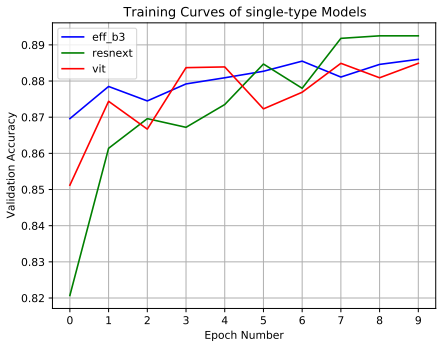

Figure 5. Training Curves For Three Single-Type Models: EfficientNet B3, ResNext and Vison Transformer.

Table 2 shows the overall validation performance of the five models, as well as single-class validation performance. It shows that the other four models have better validation performance than the baseline, which is ResNet. This is in our expectation because ResNet has simpler structure than other models. From single-class performance, all the models perform best on class 3 while perform worst on class 0. This is because the number of training images in class 3 is the largest with 10526 while the smallest for class 0 with only 870. Thus, we can see unbalanced performance on different classes. Class 2 and 4 have similar number of images, however it shows that class 4 has worse performance but more training instances than class 2. It is possible that class 4 contains much noise in the training images, which is difficult to learn the main features of this class, thus is less accurate in validation set.

4.3. Experiments on ensemble neural networks

The experiment in this section aims to explore the ensembling power, which means to combine multiple models in testing phase. Each test image is tested by selected single-type models separately, then the weighted average of the probability score from each model is used as the final decision.

The cassava competition provides a test picture shown in figure 6, which is hard to tell whether the leaf is healthy of type 4 or has disease of type 2. Using single-type model prediction, vision transformer predicts the picture as disease type 2. For other CNN models, they predict the picture as type 4 which is healthy. From the much different structures of vision transformer and the prediction outcome of this image, it seems that vision transformer learns different characteristics compared to other CNN models. From table 2, EfficientNet B3 and ResNext have the top two validation accuracy. Thus we picked the two CNN models and vision transformer to do ensembling.

Model Type	Validation Accuracy	class 0	class 1	class 2	class 3	class 4
ResNet(baseline)	0.859	0.555	0.733	0.784	0.943	0.733
Efficientnet b3	0.886	0.610	0.765	0.820	0.964	0.766
DenseNet	0.882	0.633	0.781	0.845	0.963	0.696
ResNext	0.892	0.638	0.799	0.818	0.965	0.775
ViT	0.885	0.610	0.788	0.803	0.962	0.767

Table 2. Validation Accuracies of Single-Type Models.

Model1	Model2	Model3	Validation Accuracy	class 0	class 1	class 2	class 3	class 4
ViT(1/3)	ResNext(1/3)	Effb3(1/3)	0.940	0.812	0.879	0.891	0.984	0.862
ViT(0.4)	ResNext(0.3)	Effb3(0.3)	0.930	0.803	0.869	0.862	0.983	0.833
ViT(0.3)	ResNext(0.4)	Effb3(0.3)	0.938	0.789	0.874	0.891	0.984	0.859
ViT(0.3)	ResNext(0.3)	Effb3(0.4)	0.938	0.803	0.877	0.889	0.984	0.859

Table 3. Validation Accuracies of Ensemble Model with different weight ratios.

Figure 6. Test image example

Table 3 shows the overall validation performance, as well as single-class validation performance of the ensembling of the three selected single-type models using different weight combinations. The overall validation accuracies from the ensembling models are all much higher than single-type models. For ensembling, it gets the final prediction result from multiple single-type models. For each single-type model, it may be weak to predict some certain type of classes. Using ensembling, it could mitigate the disadvantage by complementing each other's weakness and be more robust. This could be validated through validation performance on class 0. Single-type models perform poorly on class 0 as table 2 shows, where the validation accuracies do not exceed 0.64. However, from table 3, we can observe that ensembling could achieve validation accuracies of class 0 higher than 0.789, which is a huge improvement without increasing the number of training instance. Moreover, we can also see an improvement in validation accuracies for other classes.

Different weight combinations on the three single-type models are also tried. From table 3, the weight ratio 1:1:1 could achieve the best validation performance. We also tried to give higher weights to different models and observed that when giving higher weights to one of the single-type models, their validation accuracies drops compared to ratio 1:1:1. It is likely that by giving more weights to a particular single-type model, its prediction weakness could be enlarged and thus influence the overall prediction results.

4.4. Experiments on multiple techniques

The experiment in this section aims to explore the performance of multiple techniques, including Fmix (Harris et al., 2020) and label smoothing (Müller et al., 2019). In this experiment, the baseline model we used is the ensembling of vision transformer, ResNext and EfficientNet B3 with weight ratio 1:1:1 as in table 3.

First we retrained the vision transformer model by adding Fmix and used the newly trained vision transformer to replace the original one in baseline. The experiment result in table 4 shows that the validation performance drops by using new vision transformer with Fmix. Then we retrained the other two models by adding Fmix and used the three newly trained Fmix models to see the performance, which performed even much worse in validation accuracy.

For worse validation performance by using Fmix, it is possible that Fmix may combines images from different classes. For example, a healthy leaf image may be combined with an image of certain type of disease, which could make the model learn false healthy leaf features since the learnt healthy leaf features may contain leaf disease information. Moreover, for combination of images with different disease types, models could also wrongly learn features from other classes. Therefore, Fmix could not improve the validation accuracy in this task.

Then we tested whether label smoothing is helpful in this task. First we retrained the vision transformer model by adding label smoothing and used the newly trained vision transformer to replace the original one in baseline. The experiment result in table 4 shows that the validation performance drops by using new vision transformer with label smoothing. For label smoothing, it is a regularization method, aiming to prevent the training to be overfitting to the training dataset. However, it may also cause model underfitting when the training dataset is large enough for the model, thus result in lower validation accuracy.

To summarize, neither Fmix nor label smoothing could

model1	model2	model3	Validation Accuracy	class 0	class 1	class 2	class 3	class 4
ViT	ResNext	Effb3	0.940	0.812	0.879	0.891	0.984	0.862
ViT(fmix)	ResNext	Effb3	0.926	0.780	0.849	0.860	0.983	0.822
ViT(Smooth)	ResNext	Effb3	0.934	0.780	0.868	0.885	0.984	0.847
ViT(fmix)	ResNext(fmix)	Effb3(fmix)	0.916	0.761	0.831	0.847	0.979	0.798

Table 4. Validation Accuracies of combination of techniques with weight ratio 1:1:1.

improve validation accuracy for our task.

4.5. Experiments on Test Dataset

Based on the experiments on previous sections, we picked the final model with the best validation performance. The final model is ensembling of vision transformer, ResNext and EfficientNet B3 with weighting ratio 1:1:1 with hyper-parameters in section 4.1. The overall validation accuracy is 0.940. Finally we test the final model using test set from both public leader board and private leader board. The final test accuracy on public leader board is 0.905, and on private leader board is 0.901.

5. Related work

The transfer learning idea in this project was inspired by (Chen et al., 2020), in which deep transfer learning for plant disease detection was proposed and validated to be efficient and effective even in classifying plant images with complex background conditions. In (Chen et al., 2020), the authors used VGGNet (Simonyan & Zisserman, 2014) as the initial model and then strengthened this model by replacing the last layers with Inception model (Chollet, 2017), this is because Inception model has the advantage of reducing the overall parameters and computation of the model while keeping the learning ability. Moreover, the pre-trained VGGNet model was adopted to initialize the parameters of the developed model. This transfer learning was finally tested on both public and collected dataset, the experimental results showed that this transfer learning method could not only improve learning efficiency but also enhance the learning ability of the model. In this project, we do not add other image deep learning modules to the pre-trained model due to the limitation of computation resources. In future work, we plan to research on extending the pre-trained model by using compact deep neural module for further performance improvement.

In computer vision, the disease of plant detection is commonly solved through deep learning method. For example, a deep learning network was developed (Yu & Son, 2020) for apple leaf disease detection, in precise, a feature segmentation sub-network is trained jointly with a spot-aware classification sub-network. First by passing the feature segmentation subnetwork, segments of the apple leaf could be achieved. Then the segmented apple leaf is concatenated with the original image and fed into the spot-aware classification subnetwork. By using segmentation, this network could focus more attention on the leaf part where the disease symptoms appear and ignore surrounding noise that could affect prediction. The paper shows that it could achieve state-of-the-art image classification accuracy for apple leaf disease.

Besides the deep learning technique, Surampalli.etc. (Ashok et al., 2020) adopts traditional techniques including pre-processing images with Gaussian filter, feature extraction through discrete wavelet transform (DWT) and segmentation to assist the training of the convolutional neural network for tomato leaf disease classification. Using pre-processing and feature extraction could reduce image noise and get rid of blurring areas, thus enhance tomato leaf image. Therefore, detailed contents of tomato leaf could be retrieved, which is helpful for prediction. Moreover, using segmentation could determine the boundary of tomato leaf and could make convolutional neural network focus more on tomato leaf areas. Experimental results show that these traditional image-processing methods could contribute to a reliable and accurate model for tomato leaf disease classification.

It could be found from the above two literature that the key to improving the accuracy of plant disease detection is to detect the area of images where the disease symptoms appear. To achieve this, we could use both deep learning model or traditional segmentation method. In this project, we do not have ground truth segmentation maps for training segmentation network, thus no segmentation was implemented. In the future, we could collect plant images with ground truth segmentation, which could be used to train a segmentation neural network. Moreover, in real life, plant leaf images might be of various qualities and have much noise, which makes classification more complex. To get rid of these difficult conditions, applying image pre-processing techniques to reduce noise and enhance image details is still worth exploring.

6. Conclusions

Our work aims to classify cassava leaf disease which is from Kaggle competition. First, we divided the given dataset into training and validation dataset with Stratified-KFold method. Then we fixed hyper-parameter settings in training for consistent results and comparable analysis. We adopted transfer learning in our work, in precise, we used pre-trained models for parameter initialization and fine-tuned the models on our dataset. During the experiment, first we tried several popular single-type models including ResNet, DenseNet, ResNext, EfficientNet B3

and Vision Transformer, and compared their overall and single-class validation accuracy. Based on their validation performance, we chose top three models and ensembled them using weighted average to do prediction. We also tried different weight ratios of these models. Moreover, some generalization and regularization techniques including Fmix and label smoothing were investigated. Our key findings through the experiments are as follows:

- As experiment result shows, using pre-trained model for parameter initialization could achieve quicker convergence within 10 epochs, which greatly improves training efficiency considering limited computing resources.
- From validation performance for each class, classes with much fewer training instances have poor performance using single-type models. Unbalanced performance for each class due to unbalanced dataset is a key point to improve the overall performance.
- Ensembling of single-type models could help improve the performance of classes with much fewer training instances, thus improve the overall accuracy. Moreover, the ensembling we implemented during test phase does not affect the training time or require large computing resources.
- The performance by using Fmix and label smoothing in ensembling drops, which does not match our initial expectation.

References

Ashok, Surampalli, Kishore, Gemini, Rajesh, Velpula, Suchitra, S, Sophia, SG Gino, and Pavithra, B. Tomato leaf disease detection using deep learning techniques. In *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, pp. 979–983. IEEE, 2020.

Chen, Junde, Chen, Jinxiu, Zhang, Defu, Sun, Yuandong, and Nanehkaran, Yaser Ahangari. Using deep transfer learning for image-based plant disease identification. *Computers and Electronics in Agriculture*, 173:105393, 2020.

Chollet, François. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

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

Dosovitskiy, Alexey, Beyer, Lucas, Kolesnikov, Alexander, Weissenborn, Dirk, Zhai, Xiaohua, Unterthiner, Thomas, Dehghani, Mostafa, Minderer, Matthias, Heigold, Georg, Gelly, Sylvain, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Harris, Ethan, Marcu, Antonia, Painter, Matthew, Niranjan, Mahesan, and Hare, Adam Prügel-Bennett Jonathon. Fmix: Enhancing mixed sample data augmentation. *arXiv preprint arXiv:2002.12047*, 2(3):4, 2020.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Huang, Gao, Liu, Zhuang, Van Der Maaten, Laurens, and Weinberger, Kilian Q. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Müller, Rafael, Kornblith, Simon, and Hinton, Geoffrey. When does label smoothing help? *arXiv preprint arXiv:1906.02629*, 2019.

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Tan, Mingxing and Le, Quoc. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pp. 6105–6114. PMLR, 2019.

Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N, Kaiser, Lukasz, and Polosukhin, Illia. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

Xie, Saining, Girshick, Ross, Dollár, Piotr, Tu, Zhuowen, and He, Kaiming. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

Yu, Hee-Jin and Son, Chang-Hwan. Leaf spot attention network for apple leaf disease identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 52–53, 2020.