Final Report for STATS 507 Project

Yueyang Zhang Jingyan Lu Zixin Lu

***Abstract*—This project presents approaches for plant seedlings classification with a dataset that contains 5539 images of plants belonging to 12 different species. We first implement the data processing with background subtraction, data augmentation, normalization and balancing methods. After that, we fit the Convolutional Neural Network (CNN) models as well as two pretrained models VGG-16 and Resnet-152 to classify the images into 12 classes. By comparing the 4-layer CNN model and the pre-trained models, the 4-layer CNN model has better performance. Since there are two species difficult to classify, we utilized the ensemble model and finally achieved a test accuracy of 95.63%.**

I. Introduction

A. Background

Plants are important sources of food in the world, and weed control is considered a major part of agriculture. Successful cultivation of crops depends largely on the efficacy of weed control. Weeds compete vigorously with the crop for nutrients and water during the first six to eight weeks. Annual yield losses occur as a result of weed infestations in cultivated crops. Those losses can vary from 10% to 100% with types of weed, types of crop, and many other factors involved. This explains why weed control is extremely important in real life and why correct weed identification plays a critical part in this process. In modern ages, proper automation of the farming process can improve efficiency and save labour resources. It's meaningful to introduce computer vision and neural network techniques into the process of weed identification[8].

B. Research Question and Datasets

Our group project is based on a dataset provided by Aarhus University Signal Processing group, in collaboration with University of Southern Denmark. This dataset contains 5,539 images of crop and weed seedlings. The images are grouped into 12 classes and each class has different size as shown in the table 1. These classes represent common plant species in Danish agriculture. Each class contains RGB images that show plants at different growth stages. Our goal is to build a CNN model to identify the 12 plant species based on the dataset. The images are in various sizes and are in png format. Figure 1 shows some random samples from each class.

These pictures capture the plants at their early growth stage. This allows farmers to conduct weeding before the weeds start competing with crops. Also, performing image classification at this stage is easier because there is less overlapping of plant leaves[5].

We can see some classes are quite different from each other, while some are very close. And the background of all pictures

TABLE I
Size of Each Class

Species Class	Size	Type
Black-grass	309	Weed
Charlock	452	Weed
Cleavers	335	Weed
Common Chickweed	713	Weed
Common wheat	253	Crop
Fat Hen	538	Weed
Loose Silky-bent	762	Weed
Maize	257	Crop
Scentless Mayweed	607	Weed
Shepherds Purse	274	Weed
Small-flowered Cranesbill	576	Weed
Sugar beet	463	Crop

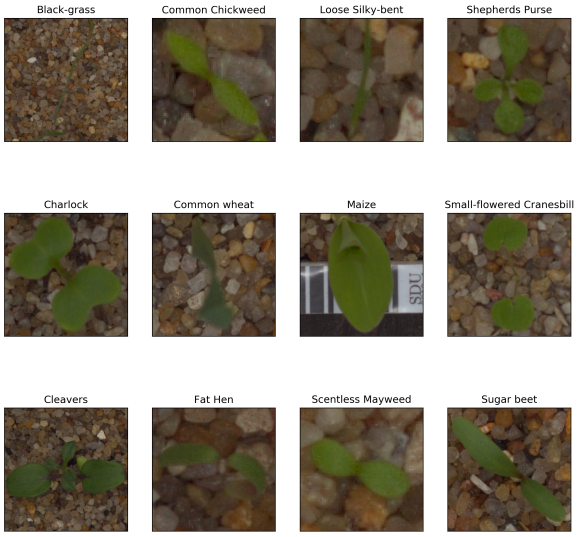
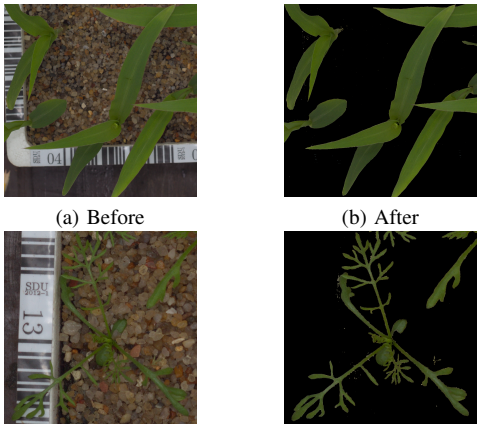


Fig. 1. Some sample images in 12 classes



(a) Before (b) After

(c) Before (d) After

Fig. 2. Example of Images Before and After Background Subtraction

are irrelevant and very similar. Besides, we can see that the data size is relatively small compared to common image classification problems. And the data size of different classes are unbalanced. All of these problems are the difficulties we will try to address with various methods including data augmentation, color based background subtraction and image normalization.

II. Methods and Analysis

A. Color-based Background Subtraction

The plant pictures in our dataset have a similar background, which includes stones and is irrelevant to the plant itself. Because all the plants here have green color and though the saturation and value of the greens do vary, they are mostly restricted within a small range. We can consider using color to subtract the irrelevant background.

We created a mask by specifying a range of possible color values of the seedlings to be captured, and then we apply this mask to all the pictures. Now we are able to produce foreground seedling images with the backgrounds subtracted. Figure 2 are images of a subset of the seedlings before and after background subtraction.

The segmentation looks fine. We got rid of the stones and the container's pixels. But we should be aware that while we subtract the irrelevant background, inevitably, some relevant features are ignored, such as some dark rhizome of the plants. So this is a trade-off between the improvement brought by the subtraction of the background, and the damage of the origin features. We need to choose the color range carefully.

Table II illustrates that the prediction performance has an obvious boost after subtracting background for 3, 4 and 5 convolution layers model with other tuning parameters being the same. Hence, we applied background subtraction to all the models we will illustrate later including the CNN models we set up and the pre-trained models.

B. Data Augmentation

To get a good result with a neural network model, it is important that parameters are tuned in the right way. In image classification problems, there is always a huge amount of parameters. And to avoid overfitting, it is critical to have enough large size of input data. Our dataset of around 5000 pictures is not large enough. To get more data, we can just make some minor alterations to the existing dataset, such as flips, rotations and adding random noises. This method can reduce the amount of irrelevant features in the dataset and help address the problem of overfitting.

For large dataset, people often choose to do online augmentation, or augmentation on the fly. This method would perform transformations on the mini-batches that would be about to feed to the model. In this case, the size of the dataset is not actually increased.

For small datasets like our case, we could use offline augmentation, which actually increases the size of the dataset by a factor equal to the number of transformations we perform. For offline augmentation, we choose flip, rotation, resize as our transformation methods. By controlling the angle of rotation, the direction of flip and scale parameter of resize, we can increase our dataset to many times its original size. Figure 3 is an example showing what a single picture could produce after offline augmentation.

TABLE II
Comparisons of Test Accuracy Before and After Background Subtraction

Test Accuracy(%)	Convolution Layer	Background Subtraction
82.11	3	NO
90.84	3	YES
89.38	4	NO
93.96	4	YES
83.96	5	NO
91.92	5	YES

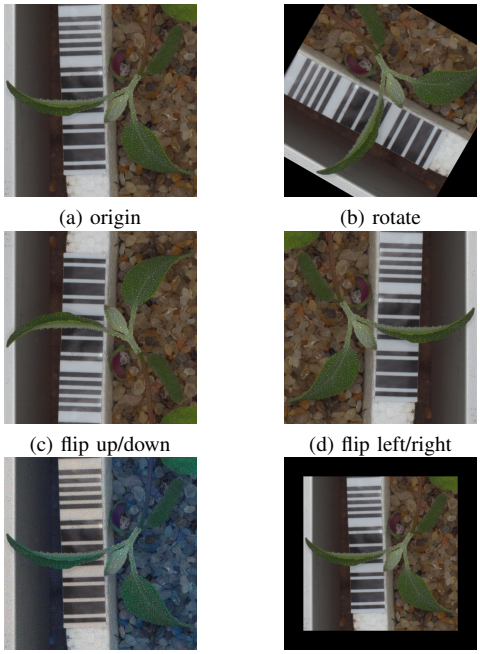
C. Normalization

Data normalization is an important step which ensures that each input parameter (pixel, in image case) has a similar data distribution. This makes convergence faster while training the CNN network. After normalization, our data are all in the range [-1,1]. We implement this normalization based on torch.transforms.Normalize function.

D. Balancing Technology

Oversampling and undersampling are used to adjust the class distribution of a dataset. Here we can see from table I, that the classes in our dataset are quite unbalanced. While the majority class, such as Loose Silky-bent, has more than 700 images, the minority class like Maize, just have around 200 images.

In our project, we combined balancing techniques and data augmentation mentioned above. The thought of our data augmentation process is to produce flipped, rotated and rescaled images randomly from original images. To balance the data, we produce different numbers of new images for different



(a) origin (b) rotate (c) flip up/down (d) flip left/right (e) add noise (f) rescale

Fig. 3. Example of Images produced by augmentation

TABLE III
SIZE OF EACH CLASS IN TRAIN

Species Class	Size
Black-grass	1500
Charlock	1500
Cleavers	1500
Common Chickweed	1500
Common wheat	1211
Fat Hen	1499
Loose Silky-bent	1498
Maize	1239
Scentless Mayweed	1499
Shepherds Purse	1358
Small-flowered Cranesbill	1498
Sugar beet	1499

categories according to a certain proportion. The goal is to make the size of each category similar in our train dataset. The size of each class after balancing is shown in table III

III. CNN BASED NEURAL NETWORK MODEL

A. Four-layer CNN model

1) Batch normalisation: Batch normalisation can fix the inputs to layers in the same range. This can reduce variance of input data distribution so as to accelerate learning process. Model with batch normalisation can achieve the same performance with that without batch normalisation in much less training epoch. The equation is shown below[7].

$$y = \frac{x - \mu}{\sqrt{\sigma^2 + \varepsilon}}\gamma + \beta$$

Where μ is the mean of the batch x and σ is the standard deviation, γ and β are parameters which could be updated after every batch, ε is a very small constant used to avoid zero-division[4]. During test, instead of using the mean, μ and standard deviation, σ of test data set, μ and σ of training data are applied.

2) Activation function: The rectified linear unit(RELU) is a commonly used activation function in image classification area. Compared with sigmoid or tanh, it has 1 or 0 as gradient which solve the gradient vanishing issues in sigmoid and tanh[2]. Applying the rectifier function could increase the non-linearity in the images and so capture the non-linear features better. In addition, the computation is much faster.

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases}$$

3) Max-pooling: Pooling is an approach to reduce the size of the feature map. For max-pooling, it will select features with the largest value to represent n x n neighbourhoods in the feature map. It can make the model invariant to translation and rotation in some cases.

4) Final architecture: The final architecture of the 4-layer CNN model can be seen from table IV. The input data is resized as 224 x 224 in pre-processing with 3 channels.

Our model consists of 4 convolution layers and 3 fully connected linear layers. Each convolution layer is followed by a batch normalization layer and a max pooling layer. In the convolutional layers, each kernel size is set as 5 and each stride is 1. Only the first layer has a padding of 2. Output channels are increased so as to represent more various features. Activation function is set as ReLU to accelerate the learning process and avoid gradient vanishing. The output layer is log softmax. Therefore, the neural network will output the possibilities of input belonging to each class in log softmax format. Loss function is set as negative likelihood so as to calculate cross entropy of the output distribution.

We set SGD optimizer with an adaptive learning rate. The initial learning rate is 0.05, and it will be divided by 5 every 5 epochs. This will push network to learn more detailed features.

5) Convergence: From the line chart of training loss and validation loss(Figure 4), we can see the convergence starts from the sixth epoch and there is no significant sign of overfitting. Therefore, we do not apply techniques to avoid overfitting such as regularization.

6) Comparison and analysis: Table V shows the CNN models with many sets of parameters including the number of convolution and linear layers, batch size, learning rate, and LR_decay(the iteration of learning rate decay). The decay rate we set is 0.1. It means that if the iteration is 5, the learning rate is divided by 10 every 5 epochs. Initially, networks with

TABLE IV
ARCHITECTURE OF THE SELECTED MODEL

Layer (type)	Output Shape	Param
Conv2d-1	[64, 12, 224, 224]	912
BatchNorm2d-2	[64, 12, 224, 224]	24
ReLU-3	[64, 12, 224, 224]	0
MaxPool2d-4	[64, 12, 112, 112]	0
Conv2d-5	[64, 24, 108, 108]	7,224
BatchNorm2d-6	[64, 24, 108, 108]	48
ReLU-7	[64, 24, 108, 108]	0
MaxPool2d-8	[64, 24, 54, 54]	0
Conv2d-9	[64, 48, 50, 50]	28,848
BatchNorm2d-10	[64, 48, 50, 50]	96
ReLU-11	[64, 48, 50, 50]	0
MaxPool2d-12	[64, 48, 25, 25]	0
Conv2d-13	[64, 96, 21, 21]	115,296
BatchNorm2d-14	[64, 96, 21, 21]	192
ReLU-15	[64, 96, 21, 21]	0
MaxPool2d-16	[64, 96, 10, 10]	0
Linear-17	[64, 1200]	11,521,200
Linear-18	[64, 128]	153,728
Linear-19	[64, 12]	1,548

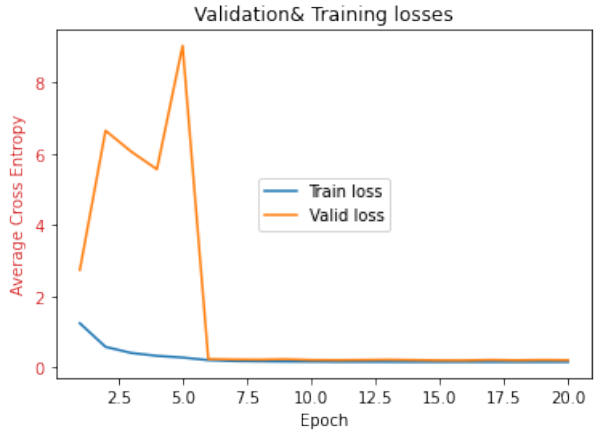


Fig. 4. Training loss and Validation loss.The orange and blue line respectively represent validation and training loss.

2 convolution layers and 3 linear layers are constructed with validation accuracy 85%. Since the prediction performance is not satisfactory, we increased the number of convolution layer. From the models with 3, 4 and 5 convolution layers, the model with 4 layers has the highest test accuracy. Therefore, we set the number of convolution layer as 4 and started to try different batch size, learning rate and iteration of learning rate decay. In tests, increasing the linear layer did not have obvious contribution to the final performance. 4 or 5 linear layers may not affect the validation accuracy significantly. Compared with the model with batch size of 64, the one of 128 seems to

TABLE V
TESTED MODELS

Test Accuracy(%)	Convolution Layer	Linear Layer	Batch Size	Learning Rate	LR_ Decay
85.12	2	3	64	0.05	5
90.84	3	3	64	0.05	5
93.96	**4**	**3**	**64**	**0.05**	**5**
91.92	5	3	64	0.05	5
82.29	4	3	64	0.01	10
85.23	4	3	64	0.01	7
89.17	4	3	128	0.05	5
92.34	4	4	64	0.05	5
93.01	4	5	64	0.05	5

Note: LR Decay in the table refers to the iteration of learning rate decay.

TABLE VI
PARAMETERS OF PRETRAINED MODEL

Model	Parameter Size
VGG-16	134,309,708
Charlock	58,168,396

perform worse. Finally, we chose the model with 4 convolution layers and 4 linear layers. The batch size, learning rate and iteration of learning rate decay are 64, 0.05 and 5 respectively.

The depth of the convolution part does affect the performance. A naive model consisting of 2 convolution layers with 10 output channels cannot represent all potential features of the dataset. However, too many convolutional layers tend to result in overfitting. This issue will be mitigated by adding dropout and residual layers, but this method will increase the complexity of the whole model. Therefore, the final architecture selects a suitable number of convolution layers and keeps the model computationally efficient.

B. Different pretrained model

In our project, we tried VGG-16 and Resnet-152 model, which were developed based on deep convolutional neural network to solve our classification problem. Their parameter sizes are shown in Table VI. Here are some detailed introductions.

1) VGG-16: VGGNet is a deep convolutional neural network developed by Oxford University Computer Vision Group (Visual Geometry Group) and researchers from Google DeepMind[11]. VGGNet explores the relationship between the depth of the convolutional neural network and its performance. By repeatedly stacking 33 small convolution kernels and 22 maximum pooling layers, VGGNet successfully constructed a convolutional neural network with a depth of 16 to 19 layers. Compared with the previous state-of-the-art network structure, VGGNet has greatly reduced the error rate.In the VGGNet paper, 33 small convolution kernels and 22 maximum pooling kernels are all used to improve performance by continuously deepening the network structure.[12]The meaning of 16 in the

TABLE VII
FIRST TEN LAYER ARCHITECTURE OF VGG-16 MODEL

Layer (type)	Output Shape	Param
Conv2d-1	[64, 64, 224, 224]	1,792
ReLU-2	[64, 64, 224, 224]	0
Conv2d-3	[64, 64, 224, 224]	36,928
ReLU-4	[64, 64, 224, 224]	0
MaxPool2d-5	[64, 64, 112, 112]	0
Conv2d-6	[64, 128, 112, 112]	73,856
ReLU-7	[64, 128, 112, 112]	0
Conv2d-8	[64, 128, 112, 112]	147,584
ReLU-9	[64, 128, 112, 112]	0
MaxPool2d-10	[64, 128, 56, 56]	0

TABLE VIII
FIRST TEN LAYER ARCHITECTURE OF RESNET-152 MODEL

Layer (type)	Output Shape	Param
Conv2d-1	[64, 64, 112, 112]	9,408
BatchNorm2d-2	[64, 64, 112, 112]	128
ReLU-3	[64, 64, 112, 112]	0
MaxPool2d-4	[64, 64, 56, 56]	0
Conv2d-5	[64, 64, 56, 56]	7,4,096
BatchNorm2d-6	[64, 64, 56, 56]	128
ReLU-7	[64, 64, 56, 56]	0
Conv2d-8	[64, 64, 56, 56]	36,864
BatchNorm2d-9	[64, 64, 56, 56]	128
ReLU-10	[64, 64, 56, 56]	0

VGG-16 network is: there are 16 layers containing parameters, which contain a total of about 138 million parameters. The advantage is that it simplifies the structure of the convolutional neural network and the disadvantage is that the number of trained features is very large.[1] Table VII is the architecture of VGG-16 model.(we just show the first 10 lines in the model)

2) Resnet-152: ResNet was first proposed by four Chinese, Kaiming He etc, who were in Microsoft Research Institute.[6] They successfully trained a 152-layer neural network by using ResNet Unit, which has more accuracy and less parameters than VGGnet.The main idea of ResNet is to increase the direct connection channel in the network, which is also the idea of Highway Network. The previous network structure was a non-linear transformation of the performance input, while Highway Network allowed to retain a certain proportion of the output of the previous network layer. The idea of ResNet is very similar to that of Highway Network, allowing the original input information to be passed directly to the subsequent layers.[10]Two types of residual modules are used in the ResNet network structure. One is to connect two 3 * 3 convolutional networks in series as a residual module, while the other is 1 * 1, 3 * 3, 1 * 1, three convolutional networks are connected in series as a residual module.There are 152 network layers in Resnet-152.[9]) It is realized by stacking the above residual modules.Table VIII is the architecture of the Resnet-152 model.(we just show the first 10 lines in the model)

3) Training Process: For transfer learning, we first load these two models with pre-trained weights which have been trained on ImageNet. Then, we freeze all weights in existing layers in the two models and replace the fully connected layer with a custom fully connected layer where the number of outputs matches the number of classes of our dataset. Finally, we only train the customized fully connected layer on the dataset. The dataset has been splitted into three subsets: 1500*12 of images for training, 40*12 for validation and 40*12 for testing. All images have been center-cropped to 224*224*3 to match the input size of the two models. Image augmentation has been applied to all training images. Cross-entropy loss has been selected as the loss function and we use stochastic gradient descent in the training process.

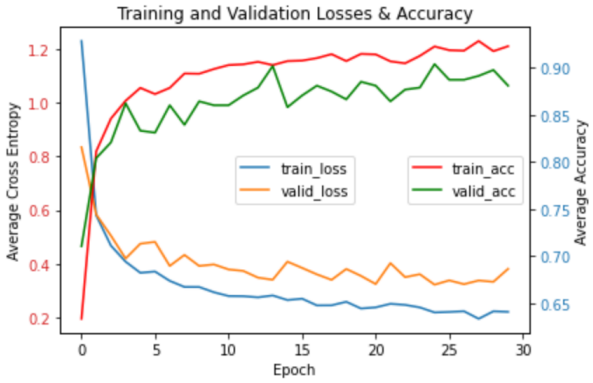


Fig. 5. VGG-16 Training and Validation Losses & Accuracy

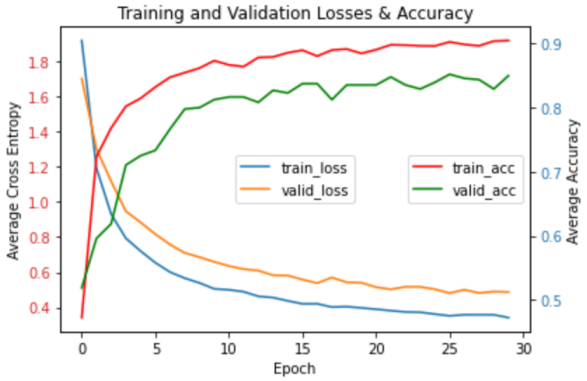


Fig. 6. Resnet-152 Training and Validation Losses & Accuracy

The VGG-16 model accuracy(Figure 5) indicates that the model performance is not that good. Although both the train loss and validation loss reach convergence, the validation accuracy has much fluctuation train accuracy. Because we only train the last layer in the model, so the performance does not meet our expectation.

The Resnet-152 model accuracy(Figure 6) indicates that the

TABLE IX
CONFUSION MATRIX

	Actual Positive	Actual Negative
Predicted Positive	TP	FP
Predicted Negative	FN	TN

model performance is quite good. Both the train loss and validation loss reach convergence and the validation accuracy has less fluctuation during the process. We can compare the pretrained model and our 4-Layer CNN model to see if there is any space to improve our model.

IV. EVALUATING METHODS

A. Classification Accuracy

The classfication accuracy can be defined as the numbers of correctly classified patterns divided by the total number of patterns. In order to compare classification performance for each class and have a deeper understanding of our model performance, we not only calculate the accuracy rate for entire test dataset, but also calculate it for each class.

B. Confusion Matrix and F1 Score

A confusion matrix of size n x n, where n is the number of different classes, is a common approach of performance evaluation in machine learning area[13]. It consists of 2 dimensions, "actual" and "predicted". Each row of the matrix represents the prediction of one instance. Each column represents one actual class. For binary classification, the matrix contains 4 basic elements as shown in table IX.

True Positives (TP): Cases we predicted as class A and they are actually from class A.

False Positives (FP): Cases we predicted as class A and they are actually from class B.

True Negatives (TN): Cases we predicted as class B and they are actually from class B.

False Negatives (FN): Cases we predicted as class B and they are actually from class A.

Confusion matrix for multi-classification is similar, and it is of great significance since lots of parameters are computed based on it, including accuracy, precision, recall and F1 score.

Accuracy evaluates how often the classifier is correct.

$$Accuracy = \frac{TP + TN}{Total}$$

Precision evaluates how often it is correct when the prediction is class A (Figure 7).

$$Precision = \frac{TP}{TP + FP}$$

Recall evaluates how often it predicts class A when the class is class A.

$$Recall = \frac{TP}{TP + FN}$$

F1 score is a weighted average of the recall and precision. It can be seen as a measure of test accuracy.

$$F1 = \frac{2 * (precision * recall)}{precision + recall}$$

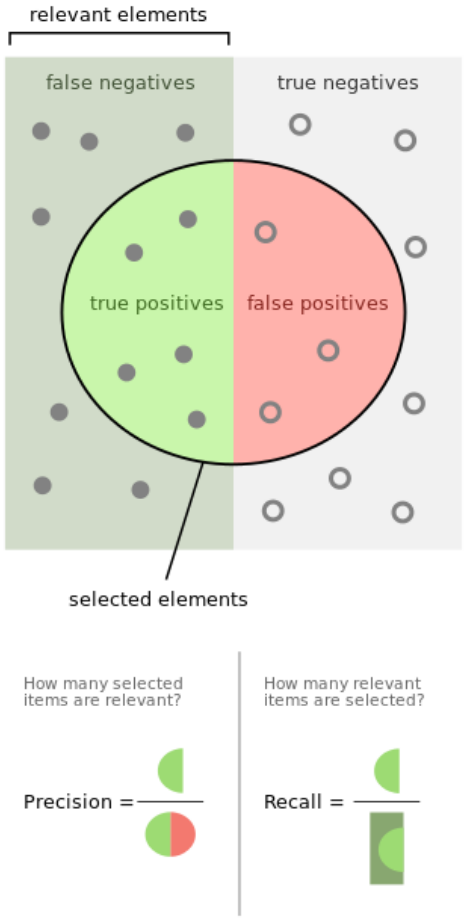


Fig. 7. Precision and recall[3]

V. RESULTS AND EVALUATION

A. Results of Pretrained Models

The confusion matrix of VGG-16 model(Figure 8) indicates that the performance of VGG-16 is quite fair.The total test accuracy is 79.3%. The classification performances are relatively bad for Black-grass, Silky-bent, Common-chickweed and Shepherd-purse. The macro F1-score is 0.889.

The confusion matrix of Resnet-152 model(Figure 9) indicates that the performance of Resnet-152 is much better than that of VGG-16. The total test accuracy is 88.1%. The macro F1-score is 0.9524. This is pretty high. The lack of accuracy rate is mainly due to poor classification between black-grass and silky-bent.

B. Result of the 4-layer CNN model

After convergence of train and validation losses, the test accuracy fluctuates within a narrow range around 93.5%, and the highest test accuracy from the training process is 93.96%. The marco F1-score is 0.961, which can be used to compare with the pretrained models. The confusion matrix of the selected model(Figure 10) indicates that the overall

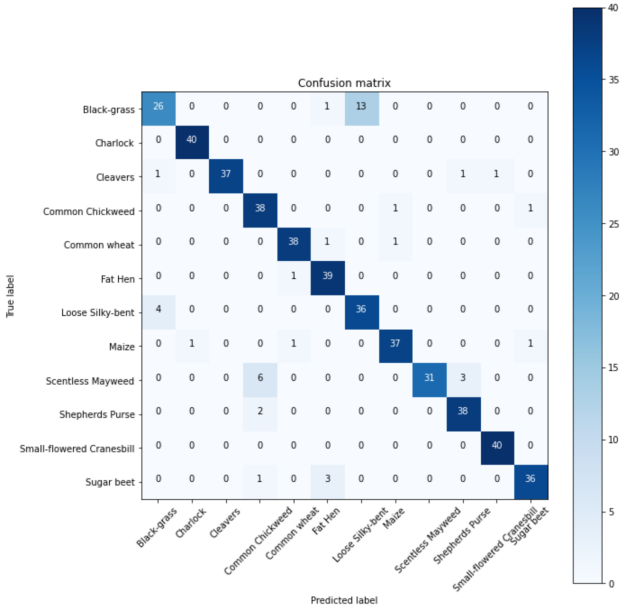


Fig. 8. Confusion matrix of VGG-16 model

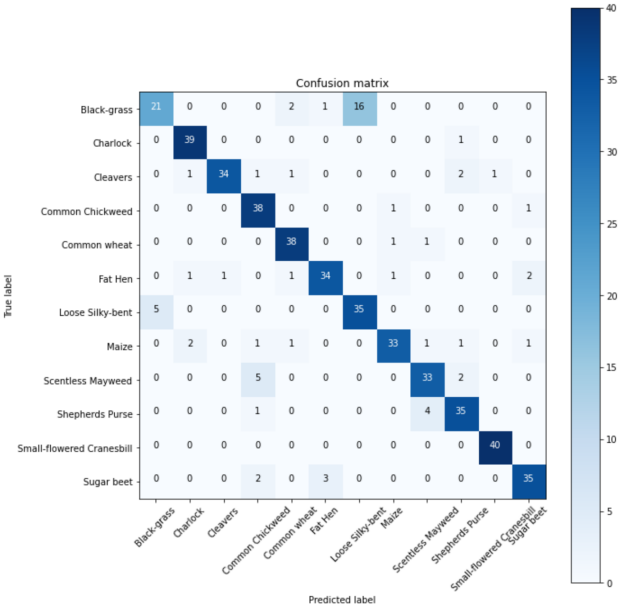


Fig. 9. Confusion matrix of Resnet-152 model

performance of the 4-layer Convolutional Neural Network model is fairly good.

The table X summarizes all the results obtained for each category of plants.

C. Comparison Analysis

Compared the result above,we find that the 4-layer CNN model has better performance based on test accuracy and F1 score than the two pretrained models, VGG-16 and Resnet-152. So we choose the 4-layer CNN model as our final model(model A).

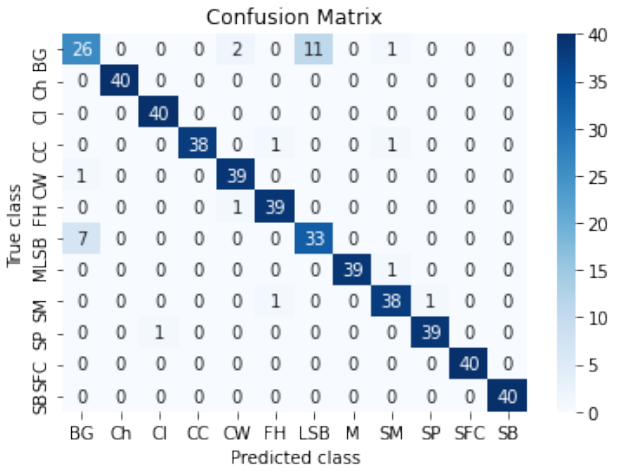


Fig. 10. Confusion matrix of the 4-layer CNN model. The tick labels are the abbreviation of the plants' name. CC: Common Chickweed; SFB: small-flowered Cranesbill; M: Maize; LSB: Loose Silky-bent; BG: Black-grass; CW: Common wheat; SM: Scentless Mayweed; Cl: Cleavers; Ch: Charlock; SP: Shepherds Purse; FH: Fat Hen; SB: Sugar beet

TABLE X
CLASSIFICATION RESULTS OF EACH CLASS

Species Class	Precision(%)	Recall(%)	F1(%)
Black-grass	76.47	65.00	70.27
Charlock	100.0	100.0	100.0
Cleavers	97.56	100.0	98.77
Co-Chickweed	100.0	95.00	97.44
Co-wheat	92.86	97.50	95.12
Fat Hen	95.12	97.50	96.30
LS-bent	75.00	82.50	78.57
Maize	100.0	97.50	98.73
Sce-Mayweed	92.68	95.00	93.83
Shep-Purse	97.50	97.50	97.50
SF-Cranesbill	100.0	100.0	100.0
Sugar beet	100.0	100.0	100.0
Average	93.93%	93.96%	93.88%

TABLE XI
CONFUSION MATRIX FOR BINARY CASE

	Actual Crop	Actual Weed
Predicted Crop	118	3
Predicted Weed	2	357

VI. ADDITIONAL ANALYSES AND IMPROVEMENTS

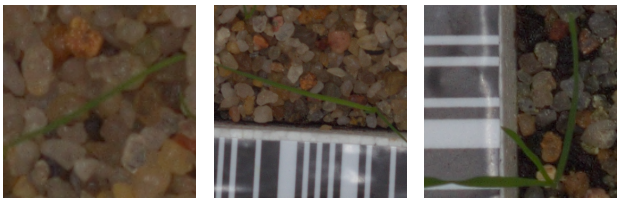
If our goal is only to classify weeds and crops, our model obtains a F1 score of 97.93%, which is quite good.(see Table XI). But for the 12 classes classification problem, the classification performance still have room for improvement.

A. Results Analysis

From both the confusion matrix and the table, we can see the classification performance is bad for two classes: loose silky-

(a) (b) (c)

Fig. 11. Example Images of Black-grass



(a) (b) (c)

Fig. 12. Example Images of Loose Silky-bent

bent and black-grass. This situation can be seen in almost all of our experiments with different models. This is not random.

When the data is not balanced, the misclassification rate is lower for Loose Silky-bent and higher for Black-grass. This is because the size of Loose Silky-bent sample 762 is much higher than 309, the size of Black-grass sample. And the model tends to classify images to Loose Silky-bent class more. When the data is balanced, this tendency disappears, these two classes both get high misclassification rates. To find the reason, we take a look at the original samples of the two classes.

From the images above(see Figure 11 and Figure12), we can see that these two kinds of plant seedling have very similar slender leaves. They are difficult to distinguish even for human beings when they are at the low growth stages.

B. Ensemble model

To increase the classification performance, we think of training a second CNN model (we call is model B), only to classify these two species. We used the same network structure as the previous 4-layers CNN model. But this time we only use images of loose silky-bent and black-grass in train and validation sets in training process. To obtain more additional artificial images and so prevent overfitting, this time we augmented the data by mainly rotating each original train image on angles in the range from 16 to 196 with step 5. Finally the training data size is 110342, which contains 54916 images of label black-grass and 55426 images of label loose silky-bent. Training process is shown in Figure 13.

C. Improvement

To get a final prediction of test data, we first classified the test images using model A. The input data of model B is formed of those images that were classified as loose silky-bent and black-grass by model A. And we combined the the other 10 classes' classification results of model A and the two classes' classification results of model B. The final result is shown below. (Figure 14 & Table XII)

From the above results we can see that the misclassification between loose silky-bent and black-grass is improved a little.

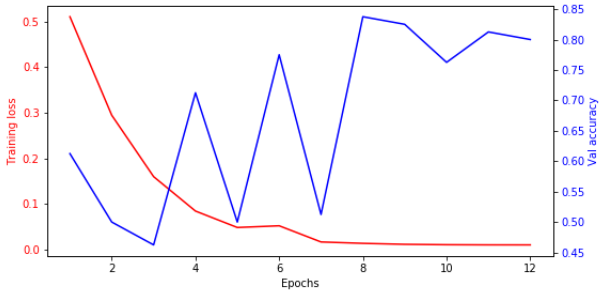


Fig. 13. model B Training and Validation Accuracy

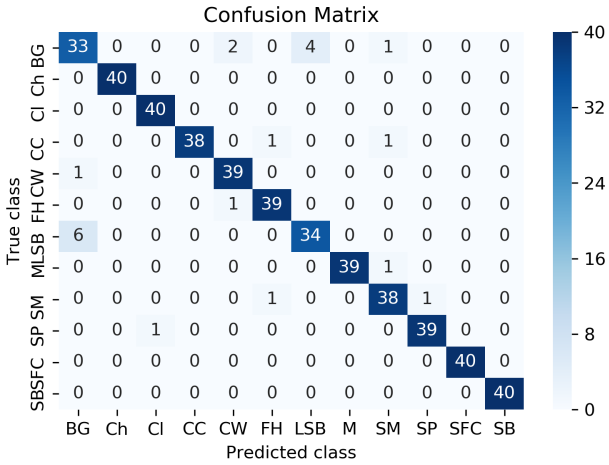


Fig. 14. Final test confusion matrix obtained by model A and B

TABLE XII
CLASSIFICATION RESULTS OF EACH CLASS

Species Class	Precision(%)	Recall(%)	F1(%)
Black-grass	82.50	82.50	82.50
Charlock	100.0	100.0	100.0
Cleavers	97.56	100.0	98.77
Co-Chickweed	100.0	95.00	97.44
Co-wheat	92.86	97.50	95.12
Fat Hen	95.12	97.50	96.30
LS-bent	89.47	85.00	87.18
Maize	100.0	97.50	98.73
Sce-Mayweed	92.68	95.00	93.83
Shep-Purse	97.50	97.50	97.50
SF-Cranesbill	100.0	100.0	100.0
Sugar beet	100.0	100.0	100.0
Average	95.64%	95.62%	95.61%

Now the number of black-grass images misclassified as loose silky-bent has been reduced from 11 to 4, and the number of loose silky-bent images misclassified as black-grass has been reduced from 7 to 6.

Our supporting reason to apply model B to test set is that it can improve the classification performance on validation set. Here we will leave out the details.

VII. Conclusion

An accurate plant classification method could help weed control in agricultural areas. This project considers the problem of plants recognition based on CNN models. The most successful convolutional neural network achieved a F1 score of 0.938 and overall accuracy of 93.96% on the testing dataset.

The original dataset contains approximately 5539 images of plants belonging to 12 species. We augmented and balanced the dataset by applying offline data augmentation techniques such as rotation, flipping and zooming. We used pre-trained VGG-16 and Resnet-152 models, but the performance is not satisfactory. We also built our own CNN based model. And the final selected model A is a four-layer CNN model. This model achieves good performance with an accuracy of 93.96% on the testing dataset.

We noticed a high misclassification rate between two particular classes: Black-grass and Loose Silky-Bent. This is due to these species' high similarity. We train an additional model B to deal with this problem. Model B only focuses on classifying the two classes and improved the previous performance a little. Finally we achieved a F1 score of 0.956 and overall accuracy of 95.63% on the testing dataset.

VIII. Future Work

Although we have achieved a comparable good prediction performance on the plant seedling dataset, there is still some room for improvement.For example, for pretrained model part, we can try more efficient pretrained model,like FixEfficient Net.

In the future, we might want to apply the model beyond this dataset to see the prediction performance in the reality.

IX. Contribution

Every member in our group devotes a lot to the project and behaves as a team. Each of us not only take responsibility to a specific part, but also help others when there is a need. We completed the project report together using online cooperative document platform overleaf. Here is a general division of work:

A. Yueyang Zhang

Mainly focus on data processing including background subtraction, data augmentation and balancing, and model evaluation part. Write the sections Introduction, Methods and Analysis, Additional Analysis and Conclusion of the report. Responsible for oral presentation.

B. Zixin Lu

Mainly focus on the part of pretrained model. Write the sections pretrained model and corresponding Results and Evaluation of the report. Responsible for the presentation PPT.

C. Jingyan Lu

Focus on the CNN model part. Tune the parameters and finally fit the 4-layer model. Write the sections four-layer CNN model, Evalation methods, Results and Evaluation of the report.

References

[1] Mohamad Aqib Haqmi Abas et al. *VGG16 for Plant Image Classification with Transfer Learning and Data Augmentation*. 2018. DOI: 10.14419/ijet.v7i4.11.20781. URL: https://www.sciencepubco.com/index.php/ijet/article/view/20781.

[2] Abien Fred Agarap. "Deep Learning using Rectified Linear Units (ReLU)". In: (Mar. 2018).

[3] Wikimedia Commons. *File:Precisionrecall.svg — Wikimedia Commons, the free media repository*. [Online; accessed 26-April-2020]. 2020. URL: %5Curl%7Bhttps://commons.wikimedia.org/w/index.php?title=File:Precisionrecall.svg&oldid=406342651%7D.

[4] Mads Dyrmann, Henrik Karstoft, and Henrik Midtiby. "Plant species classification using deep convolutional neural network". In: *Biosystems Engineering* 151 (Nov. 2016), pp. 72–80. DOI: 10.1016/j.biosystemseng.2016.08.024.

[5] Thomas Giselsson et al. "A Public Image Database for Benchmark of Plant Seedling Classification Algorithms". In: (Nov. 2017).

[6] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2016.

[7] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. Feb. 2015.

[8] Helmut Kopka and Patrick Daly. *A Guide To LaTeX*. Jan. 2004, p. 597. ISBN: 0321173856.

[9] Sajja Tulasi Krishna and Hemantha Kumar Kalluri. *Deep learning and transfer learning approaches for image classification*. 2019.

[10] S. Natesan, C. Armenakis, and U. Vepakomma. *RESNET-BASED TREE SPECIES CLASSIFICATION USING UAV IMAGES*. 2019. DOI: 10.5194/isprs-archives-XLII-2-W13-475-2019. URL: https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLII-2-W13/475/2019/.

[11] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015.

[12] Srikanth Tammina. *Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images*. Oct. 2019. DOI: 10.29322/IJSRP.9.10.2019.p9420.

[13] Sofia Visa et al. “Confusion Matrix-based Feature Selection.” In: vol. 710. Jan. 2011, pp. 120–127.