

Plant Seedling Classification

Presented by Group 16

Qinyan Li
PID: A53276822
q3li@ucsd.edu

Qian Wang
PID: A53235276
qiw018@ucsd.edu

Yunzhe Hu
PID: A53270645
yuhu@ucsd.edu

Zhanghan Liu
PID: A53283488
zh1031@ucsd.edu

ABSTRACT

Plant Seedling Classification is a Kaggle competition with the goal to classify the new unseen images into one of the twelve mentioned categories accurately. Convolutional neural networks (CNN) have outperformed conventional methods in modeling plant seedling classification. We intend to make prediction using Neural Network, feeding the model with the training set and try to tweak the number of hidden layer to get a acceptable accuracy, which is up to 0.9621. Customized CNN models with the combination of data augmentation, data cleaning help improve nearly state-of-art classification results.

1 INTRODUCTION

In recent years, image classification has become one of the most important problems in the machine learning field. Since computer hardware has seen great improvements, and various ML algorithms emerge over the years, computers are getting better and better at recognizing images. Solving image classification problems with deep learning becomes more realistic. Among all these machine learning algorithms, deep learning, especially convolutional neural networks (CNN) models gained great popularity on solving image classification problems. They will be the focus of our study in the project.

Our goal is to accurately differentiate weeds from crop seedlings based on their images in a reasonable time frame. There are challenges along the way to achieve this goal. For instance, some weeds looks nearly identical to the crops seedlings. Also, the image background contains a lot of noises, which could influence the accuracy of classification. Their impacts and the methods to handle them will be discussed in the following sections.

The ability to conduct fast and accurate differentiation on weeds and corp seedlings can help maintain better stewardship of environment effectively. And this technique can potentially assist the farmers to automate their tasks which ensures the corp yields.

2 RELATED WORK

Transfer Learning [1] is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on such these problems. Since the pre-trained models are trained on ImageNet, they are sensitive to certain features, such as edges and fit for classifying RGB images, such as Plant Seedlings Dataset.

Applying a pre-trained model should be a good start. We selected multiple pre-trained models from Keras. They are ResNet50 [8], VGG19 [6], Xception [3] and Inception. Compared to other pre-trained models, these three models performed well as feature selectors with modified connected to fully connected layers. We could classify the plant seedlings to appropriately, which would help recognize the difference between different field crops at seedling level and between the seedlings and weed.

Then we follow a Convolutional Neural Networks based approach. A convolutional neural network (CNN) contains 6 convolutional layers, pooling layers, and fully connected layers. The benefit of using CNNs is their ability to develop an internal representation of a two-dimensional image, which allows the model to learn position and scale in variant frames in plant classification [2]. This model shows the best performance in our project.

3 DATASET AND FEATURES

The database is recorded at Aarhus University Flakkebjerg Research station in a collaboration between University of Southern Denmark and Aarhus University [4]. There are approximately 960 unique plants that belong to 12 species at several growth stages, and the dataset comprises annotated RGB images with a physical resolution of roughly 10 pixels per mm. [4]. In the dataset, there are 4750 images as training set (80% as training and 20% as validation) and 794 images as test set. You could download the dataset we use in

this project at "<https://www.kaggle.com/c/plant-seedlings-classification/data>". Figure 1 shows sample input images belong to each species.



Figure 1: Sample images.

The preprocessing methods of the dataset including data augmentation, image segmentation and label conversion are discussed in detail in section 4.1. Features are extracted automatically from the deep neural network, without effect of manually extracting features.

After observing the dataset, we can get the distribution of data among species as shown in figure 2. From the figure, we can observe that there are some limitations for this dataset. First of all, the number of training set of each species is not large. Also, the dataset is imbalanced. For example, there are 655 training samples (14% of the total training set) belong to "loose silky-bent", while there are only 222 samples (5%) belong to "common wheat".

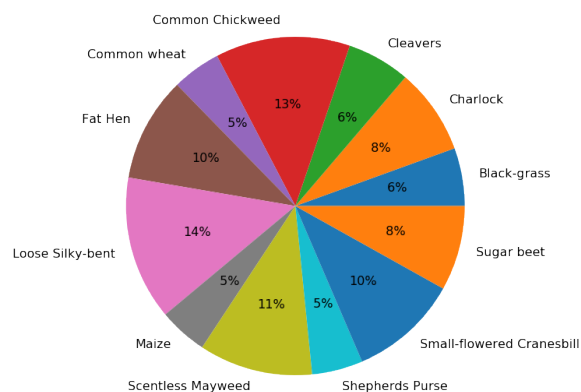


Figure 2: Distribution of data among species.

4 METHODS

4.1 Preprocessing

The training data were randomly sorted and divided into two parts, 80% for training and 20% for validation. All images were cropped to the same size (299 299) and normalized to range $[-1,1]$. For SVM inputs, images should be converted from RGB to grayscale and then flatten into one dimensional vectors.

4.1.1 Data Augmentation

Data Augmentation was employed to generate a larger training set. This was realized by performing random rotation, translation and flipping on the original inputs. An example is shown in Figure 3. The first image is the original image and the later ones are outputs of data augmentation.

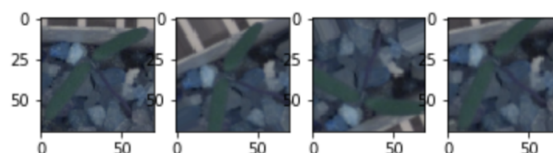


Figure 3: An example of image augmentation.

4.1.2 Image Segmentation

Since the images are taken in real environment, it's natural that there are a lot of noise. Thus, we need to remove the the noise and the background. Figure 4 shows the process of image segmentation. We first do image blurring to remove noises and get the result shown in the second image. Then, we convert RGB image into HSV, so that it can be easier to separate the color information from the luminance information. The conversion result is shown in the third image. Lastly, we create a mask to remove the background of the image and get the seed part for training and testing. The final result of the image segmentation process is shown in the last image.

4.1.3 Label Conversion

The labels are given in string in the dataset, like "Charlock" and "Cleavers", but strings are hard to process when training. Thus, we convert the string labels into the one-hot labels, and 1 indicates the species is detected, while 0 indicates not detected.

4.2 Classification models

In order to explore the best solution to this classification problem, several different models were employed to compare their performance.

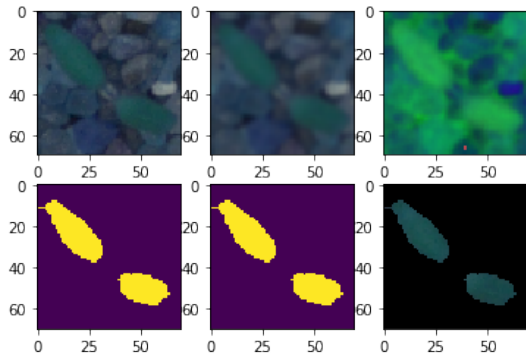


Figure 4: An example of image segmentation.

4.2.1 Support Vector Machine

Support Vector Machine (SVM) is a very popular machine learning model for classification. It works well on MNIST image classification task. In this project, SVM was employed first and used as a baseline.

4.2.2 Transfer Learning

Transfer Learning is a machine learning method that uses a network pretrained on one dataset to solve the classification task on another dataset. The final classification layers of the pretrained models should be detached and replaced by a classifier trained to solve the new task. In this project, three pretrained networks were implemented to extract features from input image. Their weights are all available on Keras.

VGG19: This is a convolutional neural network proposed in 2014. "19" stands for the number of weight layers in the network. It came second in the 2014 ImageNet classification challenge and has a top-5 error rate at 7.3% [6]. It outputs a 512 dimension feature vector.

Resnet50: Resnet50 is a residual network with 50 layers. Residual networks use skip connection to add the output from an earlier layer to a later layer. This architecture enables the network to mitigate the vanishing gradient problem. The Resnet model won the 2015 ImageNet challenge. It has a top-5 error rate at 7.02 % and the number of parameters reaches 25.6M [8]. It outputs a 2048 dimension feature vector.

Xception: Xception was proposed by the creator of Keras library. It is an extension of the Inception network which replaces the standard Inception modules with depthwise separable convolutions. It reached 5.5 % top-5 error rate on the ImageNet Dataset [3]. It outputs a 2048 dimension feature vector.

And two classifiers were compared. They were implemented by using scikit-learn library.

Logistic Regression: This is the most commonly used method for classification. It is very fast and usually performs well.

Fully-Connected Neural Network: Also known as artificial neural network(ANN), Multi-layer Perceptron(MLP). This is a neural network with one hidden layer. Here, the number of units for the hidden layer was set to be 1,000.

4.2.3 Customized Convolutional Neural Network

Besides using the pre-trained models, we also define our customized convolutional neural network. This model has 6 convolutional layers and 3 dense layers. Following each convolutional layer, there's a batch normalization layer and a RELU activation layer. There are a maxpooling layer and a dropout after every two convolutional layers. Also, there's a batch normalization layer, a dropout layer and a RELU activation layer after the first 2 dense layers. The last dense layer is followed by a softmax layer. To have a much clearer idea about our CNN model, please refer to figure 5.

| Layer (type) | Output Shape | Param # |
|---|---------------------|---------|
| conv2d_1 (Conv2D) | (None, 66, 66, 64) | 4864 |
| batch_normalization_1 (Batch Normalization) | (None, 66, 66, 64) | 256 |
| conv2d_2 (Conv2D) | (None, 62, 62, 64) | 102464 |
| max_pooling2d_1 (MaxPooling2) | (None, 31, 31, 64) | 0 |
| batch_normalization_2 (Batch Normalization) | (None, 31, 31, 64) | 256 |
| dropout_1 (Dropout) | (None, 31, 31, 64) | 0 |
| conv2d_3 (Conv2D) | (None, 27, 27, 128) | 204928 |
| batch_normalization_3 (Batch Normalization) | (None, 27, 27, 128) | 512 |
| conv2d_4 (Conv2D) | (None, 23, 23, 128) | 409728 |
| max_pooling2d_2 (MaxPooling2) | (None, 11, 11, 128) | 0 |
| batch_normalization_4 (Batch Normalization) | (None, 11, 11, 128) | 512 |
| dropout_2 (Dropout) | (None, 11, 11, 128) | 0 |
| conv2d_5 (Conv2D) | (None, 7, 7, 256) | 819456 |
| batch_normalization_5 (Batch Normalization) | (None, 7, 7, 256) | 1024 |
| conv2d_6 (Conv2D) | (None, 3, 3, 256) | 1638656 |
| max_pooling2d_3 (MaxPooling2) | (None, 1, 1, 256) | 0 |
| batch_normalization_6 (Batch Normalization) | (None, 1, 1, 256) | 1024 |
| dropout_3 (Dropout) | (None, 1, 1, 256) | 0 |
| flatten_1 (Flatten) | (None, 256) | 0 |
| dense_1 (Dense) | (None, 256) | 65792 |
| batch_normalization_7 (Batch Normalization) | (None, 256) | 1024 |
| dropout_4 (Dropout) | (None, 256) | 0 |
| dense_2 (Dense) | (None, 256) | 65792 |
| batch_normalization_8 (Batch Normalization) | (None, 256) | 1024 |
| dropout_5 (Dropout) | (None, 256) | 0 |
| dense_3 (Dense) | (None, 12) | 3084 |
| Total params: 3,320,396 | | |
| Trainable params: 3,317,580 | | |
| Non-trainable params: 2,816 | | |

Figure 5: Structure of the customized CNN.

5 RESULTS AND DISCUSSION

5.1 Experimental Setup

In this project, we have a training set with 4750 images and a test set 794 images. We split the training set into 80 percent for training and 20 percent for validation. Our experiment is divided into 3 parts: SVM as benchmark, transfer learning with different pretrained networks and classification methods and customized CNN Models. We trained all the model on UCSD Data Science / Machine Learning Platform (DSMLP) with GPU supported. To evaluate the training and validation results, we use accuracy. After that, we get test results from the CNN models.

SVM: Using SVM for image classification, we first did the image segmentation and then transformed the image from RGB to grayscale, flattened it into a 1-d array. It took about 40 minutes to train the SVM classifier and then another 20 minutes for classification.

Transfer Learning: Pretrained model weights were downloaded from Keras website, they could be also loaded directly from Keras library. We used these models to extract features from the input image and then used the extracted features as input for Logistic Regression/Neural Network classifier.

Customized CNN: To prevent overfitting, we first set the function to randomly changing the image characteristics during fitting (such as rotation, translation, flipping and zooming). When training the model, the learning rate will be reduced based on the changing of the validation accuracy. It will be reduced to prevent the convergence from too quick. Here, we train the model for 35 epochs with batch size 75. After each epoch, if the accuracy of the validation set is improved, the weights of the model will be stored for future use. Training this model on UCSD DSMLP, it took more than 4 hours to complete.

5.2 Results

We trained each of our models and get their corresponding training and test accuracy as Table 1 shown.

Table 1: Classification accuracy of different models

| Model | Train Accuracy | Test Accuracy |
|-------------------|----------------|---------------|
| SVM | 0.9992 | 0.1333 |
| Resnet50 + LR | 0.5522 | 0.5 |
| VGG19 + LR | 0.8678 | 0.8510 |
| Xception + LR | 0.9989 | 0.8813 |
| Xception + NN | 1.0 | 0.8906 |
| CNN(6 conv layer) | 0.9995 | 0.9621 |

Figure 6 shows the confusion matrix of the customized CNN prediction result.

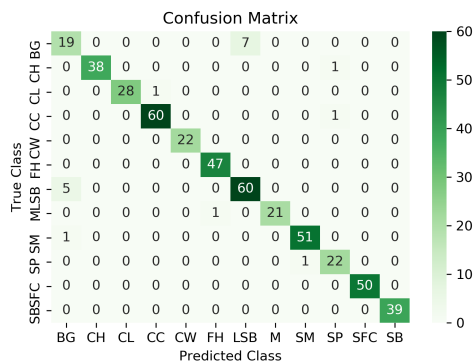


Figure 6: Confusion Matrix of customized CNN.

5.3 Discussion

From Table 1, we could see that SVM gained very high accuracy on training set, while performed poorly on test set. This indicates that the SVM classifier was overfitting. Since the SVM classifier had an input of size $299 \times 299 = 89401$, which is almost 24 times the size of the training set, it is very likely to fail due to the curse of dimensionality [7].

Among all the pretrained models, we could see that Xception had the best performance and reached 0.8813 for test accuracy. VGG19 had slightly worse performance and Resnet50 had a very poor performance. So features extracted by Resnet50 network were not good features for differentiating plant seedlings. As for the classifiers, Neural Network outperforms Logistic Regression by improving the test accuracy with 1%.

Our customized CNN model gives the best performance among all the other models. Especially, we focus on training a 6-conv-layer CNN model and reached our best score at 0.9621. Although our customized CNN model was much more shallow than those pretrained models, it beats all of them in this task. This may indicate that plant seedling classification task is essentially quite different from general image classification task in requiring more detailed and subtle information from the input image. So models worked well for ImageNet dataset could not extract useful features for this task. A customized CNN model was trained to extract the most useful feature for this task so it becomes the most successful.

From the confusion matrix, we can see that most of the species can be well predicted, however, there are some misprediction between Black-grass (BG) and Loose Silky-bent (LSB). Figure 7 provides some images from the two species classes. We can see that the seedlings of these two species are

