Determine the species of a seedling from an image

### 

**Submitted by**

**Yueran Liu(cwid:10434066)**

**Linan Dang(cwid:10442568)**

**Professor:Hong Man**

**Time:Dec 9, 2019**

### 

# Abstract

In this project, we used a dataset which shows 4750 labeled images showing plants of 12 different types. All images are quadratic but vary in size. It comprises annotated RGB images with a physical resolution of roughly 10 pixels per mm. and then we built the training model using VGG16 pretrained network to classify correctly the species with labels. Also we tried our best to eliminate the influence of excessive difference in variable values. In the future, we may try some other algorithms and test them to find the most suitable model.

Index terms: tensorflow, keras, VGG16

# Introduction

Differentiating a weed from a crop seedling can mean better crop yields and better stewardship of the environment. For several decades, researchers have been working on systems that aim to perform site-specific weed control. Although some systems are commercially available, a true commercial breakthrough of such systems is still to come, despite the construction of several prototypes and case studies showing promising results.

# Method

We would like to resize them such that each image has the same shape. Next, we will detect and segment the plant parts of the images, then normalize them such that each pixel is defined on the range [-1,1]. And there is an optional step which is to generate new images through rotations, translations and axis flippings, augmenting the original data. After computed these features, we will train and validate a basic logistic regression, using random forest or fully connected neural network model. Compared with the final result and find the best model. Finally, we will predict the species classes of the test images and write the submission file.

To achieve determination, first of all, we need to know what method we need. After filtrated and discussed, we chose to use tensorflow libraries.

The following command is import all libraries we use:

A screenshot of a cell phone

Description automatically generated

# Training image processing

Here, we used pyplot.rcParams to decide figure size with width=16 inches, height=10 inches.

The function below is to fetch all the images of training set from the directory.

A screenshot of a cell phone

Description automatically generated

We need to encode the text labels to numerical.

A screenshot of a cell phone

Description automatically generated

Then we reshaped the images to 4 dimensional tensors. (Model requires the input data to be in 4 dimensional format [no. of images, height, width, channels]) to split the data into Training and Validation to check the accuracy of the model on unseen data**.**

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

After splitted data, we known y means the label corresponding to each picture, numpy function np\_utils.to\_categorical transformed by Keras is used to change y to a one-hot form, you can think it as, y was a value before, between 0-9, and now a vector of size 10, which belongs to Which number is 1 at every position, and all other positions are 0.

A screenshot of a cell phone

Description automatically generated

Then we use Keras' built-in ImageDataGenerator to help us achieve image augmentation, used .flow to collect data and tag arrays to generate batch enhanced data.

A screenshot of a cell phone

Description automatically generated

# Pre-trained Network

VGG is a convolutional neural network model,Itt makes the improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. VGG has two structures, namely VGG16 and VGG19. There is no essential difference between the two, but the network depth is different.Here, we used VGG16 Pretrained Network.

A screenshot of a cell phone

Description automatically generated

Keras is a high-level neural network API written in Python. It can run with TensorFlow, CNTK, or Theano as the backend. Keras was developed with a focus on supporting fast experiments. Being able to turn your ideas into experimental results with minimal delay is the key to good research.we used Keras to help us processing image.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

During training, we saved the model's best weights by using ModelCheckpoint. The one with the minimum validation loss is saved. However, cause MACOS do not have GPU, it’s hardly for jupyter notebook running these code, so we change the platform with Google COLAB. Google Co-laboratory ,same as Jupyter notebook, is a free environment that can be used without any setup and runs entirely in the cloud. It supply the use of GPU.

A screenshot of a cell phone

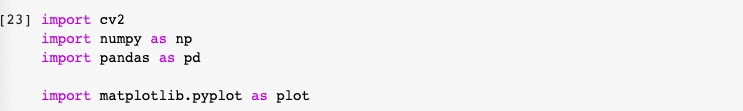
Description automatically generated

A close up of a newspaper

Description automatically generated

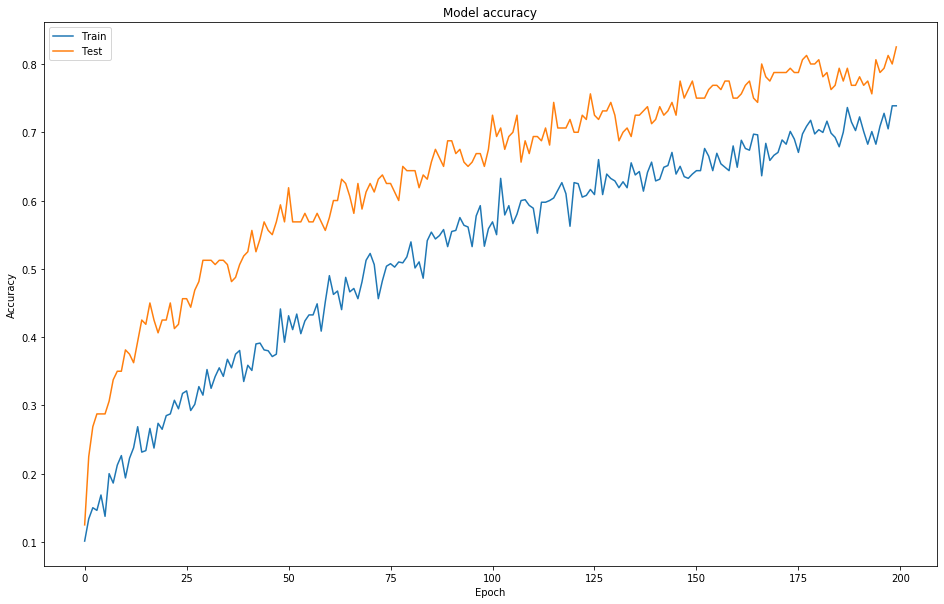
# model's accuracy and loss

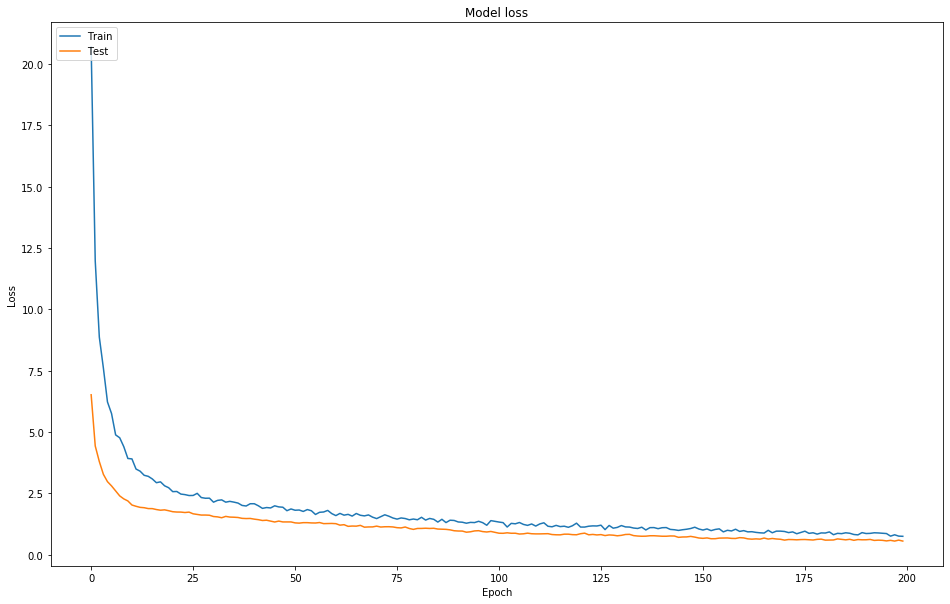
To show model's accuracy in a straight way, we can plotted the graph of model's accuracy and loss by using matplotlib.





the graph of model's accuracy and loss showing as below:



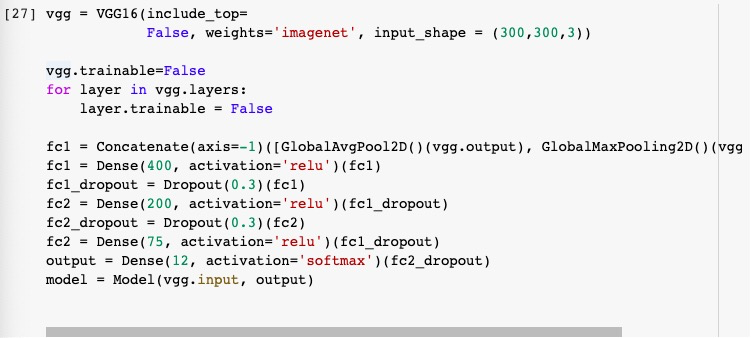


# Test image processing

We had processed with training data, and need toload the saved weights we need to define the same model architecture again. Also, make sure you do not compile the model this time.







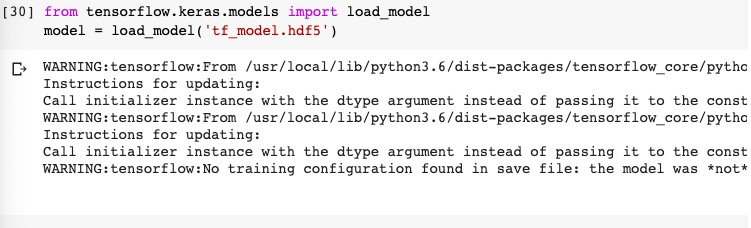
We created model by using model.load\_weights and loaded weights from model file.The structure of the model should be constant.



And used tensorflow.keras.models.save\_model() to save entire model named as tf\_model.hdf5 and load it by using tensorflow.keras.models



We then use keras.models.load\_model(filepath) to re-instantiate our model. load\_model will also take care of compiling the model using the saved training configuration (unless the model was never compiled in the first place).

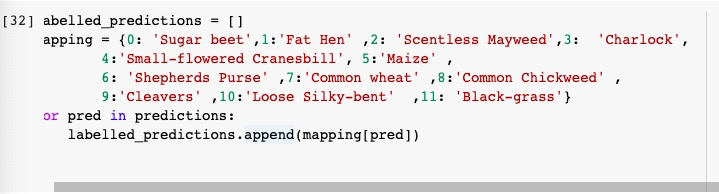


# model prediction

Here, the model predicts the new images using function model.predict()



With the predictions we have made,we can prepare submission file.

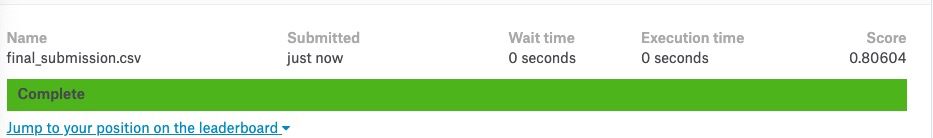






Result

With the file, finally got correct rate up to 0.80604.



# Conclusion

Our programming tool is used by Python which has comprehensive standard library, like pandas, numpy, keras. They can help us deal with the machine learning problem effectively. Given the large number of training data, and test data, first, we would like to change the format which is convenient for us to handle the data. Since machine learning models only understand data in numbers. The relationship is important among data files, and we tried different algorithm which can help us have more correct result. We use K-mean algorithm first because it is easy to understand. We add all the score belong to the track and calculate the mean of the scores. Repeating the above steps, unfortunately, we find that the result only changes a little and sometimes it becomes even worse. So, we decided to use another algorithm VGG16 and ensemble the result. And after several times to test the parameters, we find that when we take the highest 9 score and take the mean, we have the best result which can achieve 80% accuracy. After that, we use the random forest algorithm to combine with other algorithm to ensemble our result and have other good result.

In this project, Yueran Liu is responsible for the data-preprocess part, while Linan Dang is for VGG16 Network model. As to improvements and the final report script, Both of us work together.