Plant Seedlings Classification

The Trung Le

A1784927

School of Computer Science, The University of Adelaide

Abstract

"Plant Seedlings Classification" is a famous problem in Kaggle. In order to solve this problem, I analysed the dataset, removed the noise and the background of the images , transforming the categorical variables into numerical variables and reduce the over-fitting problem by random-generating the image of the train dataset. After that, I splitted the train dataset into train set and validation set, then I built a convolutional neural network (CNN) model with three Conv2D layers and applied it on both train set and validation set. Finally, I applied this model on test dataset and submitted the result into Kaggle. For this problem, my CNN model has the accuracy of 99.57% on train dataset and 67.02% on validation dataset. For test dataset, my model has the MeanFScore of 0.68261 on Kaggle.

Introduction

These days, increasing agricultural efficiency is a globally aim since the population is rising dramatically. Hence, the demand of automated crop stewardship increases rapidly since it can decrease the cost of agricultural labour and increase crop yields. To achieve this goal, the ability of identifying each plant species is very important.

In this project, I analysed the image and build a model that can predict the species of each plant through image. I used the dataset of "Plant Seedlings Classification" on Kaggle.

Plant Seedlings Database

This data set is hosted by Aarhus University Department of Engineering Signal Processing Group and it is available in Kaggle[1]. This problem has 3803 images in train dataset which belong to 12 plant species and 794 images in test dataset. There is one output variable in this problem – the species of each image.

[1] https://www.kaggle.com/c/plant-seedlings-classification/overview

Implementation Step

All my coding is performed in Jupyter Notebook using a Python kernel.

In order to solve this problem, I followed 4 steps:

Step 1: Data analysis – collect the data, analysing the dataset and understanding the construct of dataset.

Step 2: Data preprocessing – removing the background of images, label encoding the categorical variable (in this problem, the categorical variable is the output variable) and reduce the over-fitting problem.

Step 3: Build model – building model using the train dataset to predict the target variable.

Step 4: Apply on test dataset – Apply the model built in step 3 on test dataset and submit the result on Kaggle.

Data Analysis

The aim of this problem is using the image of plant seedlings at various stages of grown to predict the plant species. Since there are 12 plant species, this problem is a classification problem.

There are 3803 images in train dataset and 794 images in test dataset. I stored every image in train dataset and test dataset and reshape the image to the size of (50,50). This action can help the Convolutional neural network model training faster in "Build model" step.

In the train dataset, each species has different number of images. The "Loose Silky-bent" has the highest number of image (524 image), the "Common wheat" and the "Maize" both has the lowest number of image (177 image).

Loose Silky-bent	524
Common Chickweed	489
Scentless Mayweed	413
Small-flowered Cranesbill	397
Fat Hen	380
Charlock	312
Sugar beet	308
Cleavers	230
Black-grass	211
Shepherds Purse	185
Maize	177
Common wheat	177

Figure 1: Number of images of each species

From figure 1, it can be seen that this dataset is unbalanced. Some species has a large number of images and some species only has a few images.

Data Preprocessing

There are three problems of this dataset. The first problem is the background of each image, which can lead to misclassification problem. The second problem is the output variable is image's labels, which is a categorical variable. The third problem is the model can over-fit the train dataset. To increase the accuracy of my model, I need to deal with these three problems.

Firstly, I removed the background of each image. In this dataset, every plant in each image is green. Hence, I created a mask, which have the upper limit and lower limit of range of green colour, then removed the other part of the image. This process has five steps:

- Step 1: Using gaussian blur for remove noise of the image.
- Step 2: Converting RGB image to HSV image. The colour range is represented easier in HSV colour range than in RGB colour range.
- Step 3: Create the mask which have the upper limit and lower limit of range of green colour.
- Step 4: Create the Boolean mask.
- Step 5: Apply the Boolean mask on each image.
- Step 6: Normalize the value from [0, 255] to [0, 1] for faster training the Convolutional neural network model.

Secondly, to deal with the categorical output, I created a class array containing 12 values, then I used label encoder to encode every value by position in this array. For example, 'Black-grass' value is encoded to [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0].

Lastly, to solve the over-fitting problem, I created an image generator which randomly rotate and flip each image. In this generator, the random rotation is from 0 degree to 180 degree, the horizontal and vertical flip is also enabled.

Building Models

Before building models, I split the train dataset into 2 set – train set and validation set. Train set contains 80% data of train dataset and validation set contains 20% data of train dataset. The reason of this splitting is I wanted to use the train set to train the models, then used validation set to evaluate the models.

For this dataset, I built a convolutional neural network (CNN) model with three Conv2D layers. The first layer has 64 filters, the second layer has 128 filters, and the last layers has 256 filters. The activation for these layers is "relu". After each Conv2D layers, I used batch normalization layer. Between these three Conv2D layer, I used two MaxPooling2D layers. After the last

Conv2D layer, I used a Flatten layer and a Dense layer with 12 classes. All filters size is (5,5). The summary of my CNN model is shown in figure 2 below.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 46, 46, 64)        4864
_________________________________________________________________
batch_normalization (BatchNo (None, 46, 46, 64)        256
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 23, 23, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 19, 19, 128)       204928
_________________________________________________________________
batch_normalization_1 (Batch (None, 19, 19, 128)       512
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 9, 9, 128)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 5, 5, 256)         819456
_________________________________________________________________
batch_normalization_2 (Batch (None, 5, 5, 256)         1024
_________________________________________________________________
flatten (Flatten)            (None, 6400)              0
_________________________________________________________________
dense (Dense)                (None, 12)                76812
=================================================================
Total params: 1,107,852
Trainable params: 1,106,956
Non-trainable params: 896
```

Figure 2: CNN model

After building the CNN model, I applied this model on the train set and validation set. I ran the model with the batch size of 50 and the epochs of 50. After running, the best accuracy of this model on train set is 99.57% and on validation set, the best accuracy is 67.02%.

Apply on test dataset

After fitting the train dataset to the CNN model, I applied this model on test dataset. Firstly, I removed the background from every image on the test dataset. After that, I applied the CNN

model on test dataset and predicted the species of every image. Lastly, I stored the Id and the predict output of each image to a .csv file and submit it on Kaggle.

In Kaggle, my submissions are evaluated on MeanFScore, which actually is micro-averaged F1-score. The MeanFScore of my model on Kaggle is 0.68261.

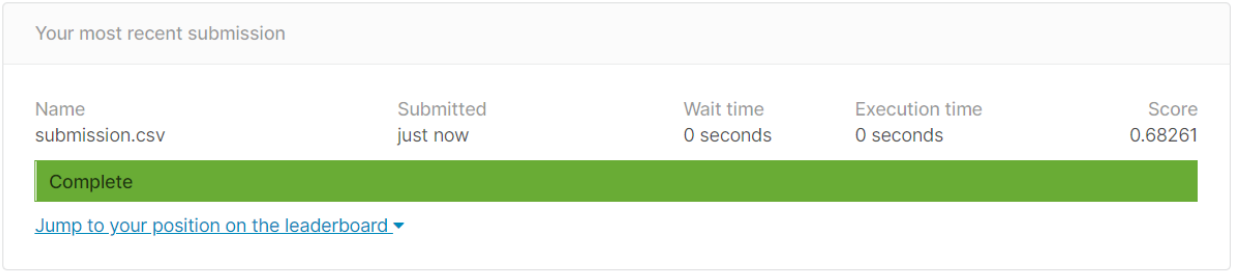

Figure 3: Kaggle Result

References

[1] https://www.kaggle.com/c/plant-seedlings-classification/overview