**Introduction**

# **Comparison between Traditional method and Deep Learning method**

PKU 2022 Spring Semester CVDL homework

School ID : 2102010644

Name : LEE DONG WOOK

College：Department of Engineering

We make a model that can classify Plant Seeding Using two method which is Traditional and Deep Learning method respectively. By comparing two method, check out each method of features and distinctions.

## **Submission**

Traditional Method must have feature extraction and classifier. Deep Learning method must have network, optimizer, augmentation, reguralization. Check each charateristic property by comparing each method and model.

## **Dataset**

We use dataset of Kaggle (<https://www.kaggle.com/comp>etitions/plant-seedlings-classification). trainset images are 4750, testset images are 794, and there are 200~600 images per class. Classes are consisted of 12 classes. Image pixel size is (3, 1899, 1900). More description is described in “data\_analysis.ipynb” file in github. (https://github.com/dwsmart32/DL\_Model\_collect/blob/master/Plant\_Seeding/data\_analysis.ipynb)

# **Traditional method**

At first, features and descriptor can be calculated by using SIFT function in cv2 module. As Image below, because of the existence Background, it can be seen that discriptors are dotted on background, not a plant which is rather important for proper classification. So deleting background method is used and easily can be checked it is more reasonable for dotting discriptors.

텍스트이(가) 표시된 사진

자동 생성된 설명

Fig. 1 Change of white discriptor with Background deletion method

Mask are calculated by using HSV features of each image. Black color is setted for the area that is not a plant. Gaussian Blur are applied to make image sharpen. Performance difference is compared in each method.

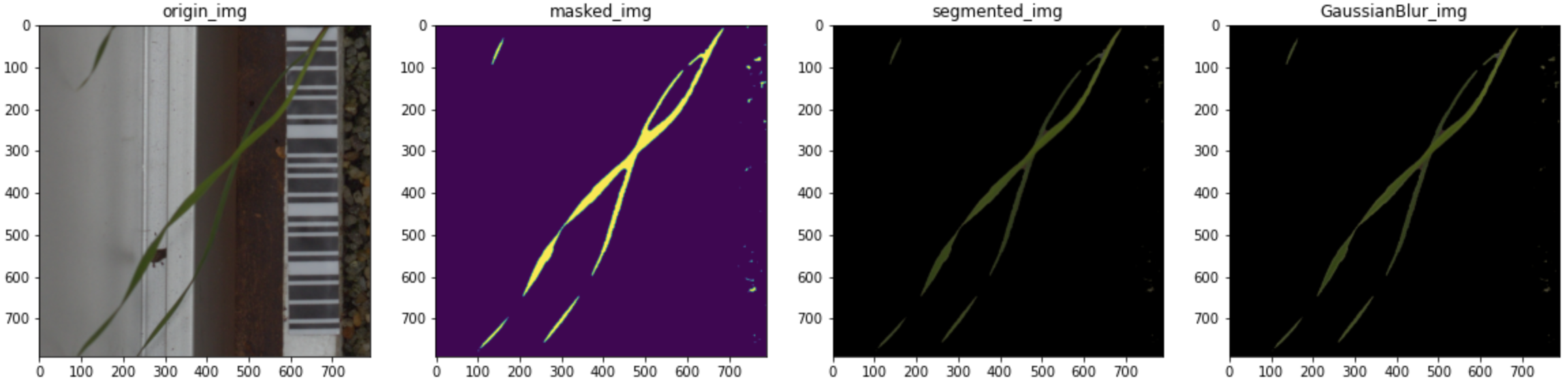


Fig. 2 Masking method

## **SIFT & K-mean Clustering with SVM**

All descriptor are caculated by SIFT method . All descriptor is vetorized and center point are calculated by using K-mean clustering method in condition which is that num\_cluster is 60 and image resize in (128, 256) . Support Vector Machine(SVM) model is used as training model. Linear model and Gaussian kernel are used for comparing. For exact comparing in same condition, seed is selected as 42 during Kmeans and training.

### Background without deletion

Images which are not deleted it’s Background are used. Number of Cluster is 60 for K-mean Clustering, C is 1, gamma is 0.008. Random state is 42. so that make the higest performance.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class name | Precision |  | Class name | Precision |
| Black-grass | 0.25 |  | Black-grass | 0.25 |
| Charlock | 0.16 |  | Charlock | 0.20 |
| Cleavers | 0.18 |  | Cleavers | 0.50 |
| Common Chickweed | 0.17 |  | Common Chickweed | 0.24 |
| Common wheat | 0.27 |  | Common wheat | 0.00 |
| Fat Hen | 0.25 |  | Fat Hen | 0.29 |
| Loose Silky-bent | 0.23 |  | Loose Silky-bent | 0.23 |
| Maize | 0.10 |  | Maize | 0.00 |
| Scentless Mayweed | 0.29 |  | Scentless Mayweed | 0.36 |
| Shepherds Purse | 0.00 |  | Shepherds Purse | 0.00 |
| Sf - Cranesbill | 0.12 |  | Sf - Cranesbill | 0.24 |
| Sugar beet | 0.35 |  | Sugar beet | 0.23 |
| Total(%) | 21.97 |  | Total(%) | 25.73 |

Fig. 3 mAP in Linear kernel and Gaussian kernel w/o B.M.

### Background deletion

Images which is deleted it’s Background are used. Nuber of Cluster is 60 for K-mean Clustering, C is 1, gamma is 0.008. Random state is 42. Hypreparameters are also tuned as well.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class name | Precision |  | Class name | Precision |
| Black-grass | 0.31 |  | Black-grass | 0.58 |
| Charlock | 0.49 |  | Charlock | 0.59 |
| Cleavers | 0.33 |  | Cleavers | 0.52 |
| Common Chickweed | 0.48 |  | Common Chickweed | 0.51 |
| Common wheat | 0.56 |  | Common wheat | 0.72 |
| Fat Hen | 0.62 |  | Fat Hen | 0.70 |
| Loose Silky-bent | 0.67 |  | Loose Silky-bent | 0.71 |
| Maize | 0.31 |  | Maize | 0.50 |
| Scentless Mayweed | 0.64 |  | Scentless Mayweed | 0.63 |
| Shepherds Purse | 0.33 |  | Shepherds Purse | 0.62 |
| Sf - Cranesbill | 0.72 |  | Sf - Cranesbill | 0.80 |
| Sugar beet | 0.51 |  | Sugar beet | 0.54 |
| Total(%) | 55.29 |  | Total(%) | 63.25 |

Fig. 4 mAP in Linear kernel and Gaussian kernel with B.M.

## **Hog with SVM**

Hog function in cv2 module is used for getting feature descriptor. Image is also resized in (128, 256) and changed in gray scale. SVM model is used as training model. Linear model and Gaussian kernel are used for comparing. For exact comparing in same condition, seed is selected as 42 during Kmeans and svm training.

### Background not deleted

Images which aren’t deleted it’s Background are used. Number of Cluster is 60 for K-mean Clustering, C is 6. gamma is 0.015. Random state is 42. Hyperparameter is tuned by a lot of steps so that make the higest performance.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class name | Precision |  | Class name | Precision |
| Black-grass | 0.12 |  | Black-grass | 0.29 |
| Charlock | 0.22 |  | Charlock | 0.28 |
| Cleavers | 0.13 |  | Cleavers | 0.29 |
| Common Chickweed | 0.27 |  | Common Chickweed | 0.28 |
| Common wheat | 0.09 |  | Common wheat | 0.50 |
| Fat Hen | 0.17 |  | Fat Hen | 0.21 |
| Loose Silky-bent | 0.17 |  | Loose Silky-bent | 0.24 |
| Maize | 0.25 |  | Maize | 0.56 |
| Scentless Mayweed | 0.31 |  | Scentless Mayweed | 0.40 |
| Shepherds Purse | 0.06 |  | Shepherds Purse | 0.00 |
| Sf - Cranesbill | 0.32 |  | Sf - Cranesbill | 0.32 |
| Sugar beet | 0.21 |  | Sugar beet | 0.19 |
| Total(%) | 20.50 |  | Total(%) | 26.99 |

### Fig. 5 mAP in Linear kernel and Gaussian kernel w/o B.M.

### Background deleted

Images which is deleted its Background are used. Nuber of Cluster is 60 for K-mean Clustering, C is 6, gamma is 0.015. Random state is 42. Hypreparameters are also tuned as well.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class name | Precision |  | Class name | Precision |
| Black-grass | 0.36 |  | Black-grass | 0.39 |
| Charlock | 0.57 |  | Charlock | 0.71 |
| Cleavers | 0.46 |  | Cleavers | 0.65 |
| Common Chickweed | 0.53 |  | Common Chickweed | 0.65 |
| Common wheat | 0.28 |  | Common wheat | 0.54 |
| Fat Hen | 0.53 |  | Fat Hen | 0.68 |
| Loose Silky-bent | 0.67 |  | Loose Silky-bent | 0.70 |
| Maize | 0.51 |  | Maize | 0.63 |
| Scentless Mayweed | 0.55 |  | Scentless Mayweed | 0.67 |
| Shepherds Purse | 0.37 |  | Shepherds Purse | 0.53 |
| Sf - Cranesbill | 0.81 |  | Sf - Cranesbill | 0.84 |
| Sugar beet | 0.58 |  | Sugar beet | 0.72 |
| Total(%) | 54.18 |  | Total(%) | 67.36 |

Fig. 6 mAP in Linear kernel and Gaussian kernel with B.M.

# **Deep Learning Method**

## **Model**

Resnet50 and VGG16 model are used as transfer learning. Also, MLP are designed in 2 ways which are Simple CNN net and Unet respectively.

Final Model : Unet

Parameter size (MB): 3.11

Estimated Total Size (MB): 13.76

## **Optimizer**

3 kinds of Optimizer which are SGD, Adam, Adagrad are tried. Adam is the best and then Adagrad is also good and SGD is not proper for the model that used in this experiment. So, Adam has been chosen in final model. Learning rate is setted for 0.002 by tuning hyperparameter. In case of Adagrad, there is a problem in torch library that optimizer is only calculated in CPU, We should use CPU in training model with Adagrad optimizer.

Optimizer : Adam (lr : 0.002)

## **augmentation**

Firstly, resizing method is used. Since Resnet50 and VGG16 network is based on (224,224) image. So we make input images to (224, 224) size and also (96,96) for Simplenet and Unet. Secondly, We make all pictures normalize. All dataset of pixels mean value is 0.3 and standard variation value is also 0.1. So, we make all pixels value normalize for enhancing speed and performance. Thirdly, We also use Centercrop, GaussianBlur, Flip, but performace is rather worse. So only resizing and normalization is seleted as augmentation . Details are in data\_analysis.ipynb file.

Resize : (224, 224) or (96,96)

Normalize(RGB) : ((0.3, 0.3, 0.3), (0.1. 0.1, 0.1))

## **Regularization**

We use weight decay for avoiding weight differs dramatically and set gamma value as 0.1.

Weight decay(gamma) : 0.1

Dropout : 0.5 for Unet

## **Other Specification**

For Hyperparamter tuning torch.manual\_seed(42) is used. CrossEntropyLoss is also used as loss function. epochs value is 30. Batch size is 32 and output Dense Layer is 12. Experiment is done in condition which is AMD Ryzen 5 3600 CPU, NVDIA GeForce GTX 1660 SUPER.

## **Performance and Result**

Accuracy and Loss of datasets are tracked by tensorboard library. Accuracy and Loss are same as below.

### Resnet50

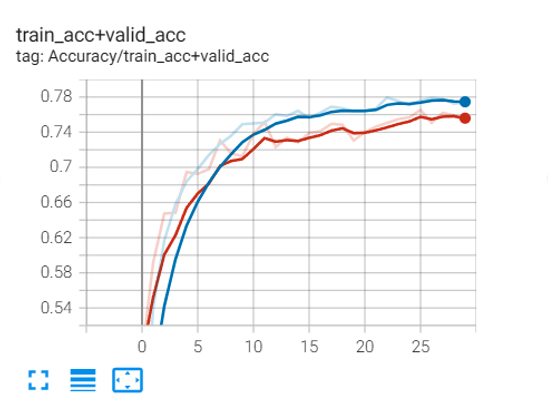
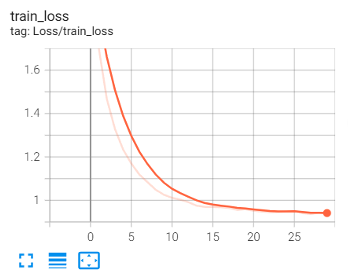


Fig. 7 training loss and accuracy of training/validation set in resnet50

Train dataset loss : 0.97 (at epoch 30)

Train dataset acc : 0. 78

Valid dataset acc : 0.77

Testset acc : 72.16 %

### VGG16

### 

Fig. 8 training loss and accuracy of training/validation set in vgg16

Train dataset loss : 1.13(at epoch 20)

Train dataset acc : 0. 65

Valid dataset acc : 0.67

Testset acc : 63.35 %

### Unet

### 

Fig. 9 training loss and accuracy of training/validation set in Unet

Train dataset loss : 0.09215 (at epoch 50)

Train dataset acc : 0. 9703

Valid dataset acc : 0.7926

Testset acc : 81.17 %

# **Conclusion**

Background deletion method was efficient for enhance the accuracy. Simultaneously, Deep learning method has good performance than Traditional method which is SIFT and HOG. We can check that just simple unet and not customized vgg16 model is sufficient for getting more than 80% accuracy. Also, training spending time in Deep learning method is much than traditional method. We should make customized Deep learning model or custimized data preprocessing skills only for this dataset for enhancing performance and also reducing training time.

# **References**

1. Pytorch\_tutorial:https://pytorch.org/tutorials/beginner/blitz/cifar10\_tutorial.html
2. tensorflow\_tutorial:https://www.tensorflow.org/tutorials/images/classification
3. vlfeat开源library: https://www.vlfeat.org/
4. sklearn开源library: https://scikit-learn.org
5. opencv开源library: https://opencv.org/
6. dwsmart32 github : <https://github.com/dwsmart32/DL_Model_collect>
7. docongminh github : https://github.com/docongminh/Classification-SIFT-SVM/blob/39b8aca52271b4718793431acdf5714c46fd1e85/build\_model.py#L26