People Space AI course content

Cactus vs Succulent vs Tree Classifier

Madison Kim

Git repository (Model & Dataset) https://github.com/yjyjy131/plant_classifier_model.git
Git repository (for Voila) https://github.com/yjyjy131/plants_classifiter.git
Binder https://mybinder.org/v2/gh/yjyjy131/plants_classifiter.git/HEAD

Acquiring the datas

- Scraped the google images by using Selenium

  I automatically scraped images for a dataset in a cloud environment.

  - Insert user's search keyword into the query part of url
  - Make the widest window, and scroll all the way down
  - Scrap the images' src from the image div of html all at once
  - Make them to zip file and download to the local computer

- Bing Image API
  - Set the maximum numbers of files that can be downloaded to 1,000 and downloa d a lot of image files

I finally got 4,632 images for the dataset !

Record of experiment results

Brief Dataset Cleaning

- Avoid duplicated images in data crawling

  I predicted that there would be a lot of duplicated image files, so I tried not to create du plicate datas as much as possible.

  - When scraped datas, use the OR operator to allow various images to be retrieved
  - After scraped datas, sort the image files order by file size. The same files would ha ve similar or same file size, so it's easier to find duplicate images.

- Cannot load some images on jupyter notebook & model
  - Delete broken files that I cannot open
  - The image files over about 30KB were too big to load. So use the image edit tool (Honey View) and adjust all of the files' size to about 20KB or less.

- Mislabeled data
  - Because I just scraped images at once as a search keyword, so there were some mislabeled datas in the dataset. For Example, a cactus image is in the succulent fo lder. So I relabeled them.

- Useless datas
  - Delete datas that don't have any relations about cactus, succulent and tree

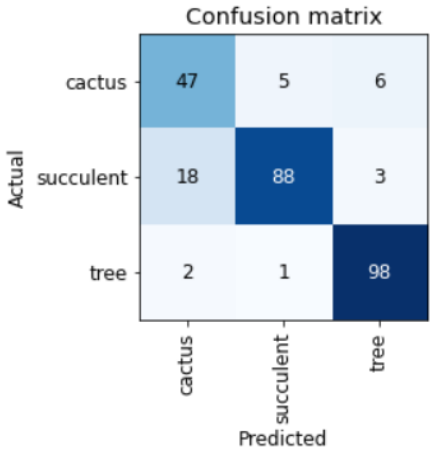
With this process, I was able to make about 3,000 datas for training the model !

First Model training

- confusion matrix

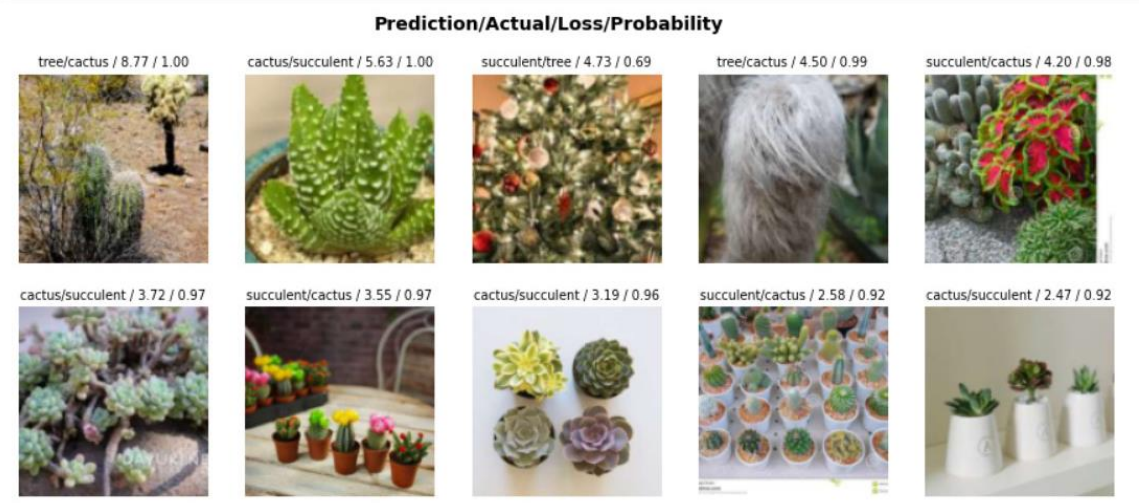
epoch	train_loss	valid_loss	error_rate	time
0	1.410495	0.442991	0.160448	00:13

epoch	train_loss	valid_loss	error_rate	time
0	0.651593	0.355492	0.126866	00:15
1	0.520199	0.378430	0.156716	00:15
2	0.462295	0.389066	0.152985	00:16
3	0.391855	0.353640	0.130597	00:16



I trained my model with dataset collected so fat. The error rate was 0.13 high. In addition, the pr ediction rate of cactus was very low, as I can see in the confusion matrix. When looking at the m atrix (1, 0), it seems to be confused between cactus and succulents.

- 10 images with the highest loss



After analyzing the top 10 losses images, I decided to delete the datas in the following cases.

- Multiple categories in one picture

There were some cases of different kinds of plants in the one picture. Like the first pictur e, there is actually a tree in the picture, and the model predicted that the picture was a tr ee. In this case, I deleted photos containing multiple categories from the dataset because something can exactly confuse learning.

- So many one plants in one picture

(445)

After looking at the three datas of succulent, there were many pictures of succulent plants in several pots like the matrix . (Maybe people who raise succulents like this way of taking pictures.) Therefore, my model thought that succulents could not distinguish the plant's shape, and it predicted that it was succulents when there were many pots. Therefore, I assume that the cactus in multiple pots could not be determined as in the first picture. Therefore, most of the potted plants arranged in matrix arrangements were deleted from the dataset.

- Delete the confused images

I was very shocked to see Loss images, because I found that these can really be seen as succulen t, tree and succulent! So I deleted all images of categories that were ambiguous.

- Part of the plant, not the whole figure

The reason why this picture looks succulent is that it is only partially enlarged, and it's hard to re cognize the entire appearance of the tree. Therefore, I removed the images with only one part th at was enlarged.

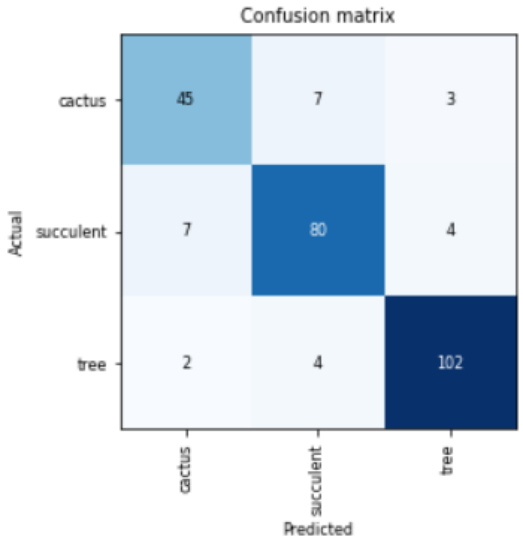
Through data cleaning, I reduced the dataset to about 1,500 data, and I trained the model again!

Second Model training

- confusion matrix

epoch	train_loss	valid_loss	error_rate	time
0	1.365873	0.539711	0.157480	00:11

epoch	train_loss	valid_loss	error_rate	time
0	0.591304	0.301392	0.118110	00:14
1	0.483318	0.318656	0.110236	00:14
2	0.390655	0.307453	0.118110	00:14
3	0.312243	0.297081	0.106299	00:15



I trained the model again with a new set of datas. I could see that the error rate dropped by 0.05 points! Confusion matrix also reduced the number of ‘cactus confused with succulents’ by abou t 50%. However, the figures of prediction in the matrix were not balanced. The cactus was too s mall at 45, and the tree was too much at 102.

- 10 images with the highest loss

After seeing these Loss images, I deleted photos that the model might be confused abou t. (left to right, picture number in order)

1, 3, 9, 10 - hard to tell exactly what kind of plant it is

2, 7 - I think I didn't clean all the duplicate data before. So I checked it again and remov ed only the duplicate data.

I didn't erase the rest of the pictures because I thought they all had a clear appearance a nd were well labeled.

- Test prediction with real photos

I tested the model with pictures I actually took that I didn't use to train it.

(My mother's hobby is raising some succulents, so I could easily get real datas!)

Prediction: tree; Probability: 0.9999

Prediction: cactus; Probability: 0.8264

Prediction: tree; Probability: 0.9041

It could recognize real trees very accurately. But I found out that two problems occurred. The sec ond picture shows a succulent plant named Stookie (although it looks a little different from ordi nary succulent) . Therefore, it was incorrectly predicted to be cactus. Also, the third picture is a c actus, not a tree. Maybe I overtrained the tree too much.

- Unbalanced dataset

After seeing the result of the confusion matrix and my prediction test, I realize that the balance of my dataset was not good. In tact, there were only 250 cactus pictures, but about 500 datas for succulents. Also the rest datas (about 700) were all tree datas! I guess that’s why my model onl y recognized trees well, but didn’t recognize cactus. Perhaps the problem was the unbalanced d ataset, so I did some following tasks.

- Add various , clear cactus pictures

- Delete some typical succulents, and add some snake plants (Stookies and Sansevierias)

I was careful to add too many sanke things, cause they actually don't have typical shapes of succ ulents.

- Delete some tree datas that can give confusions

- Balancing the number of each categories' datas

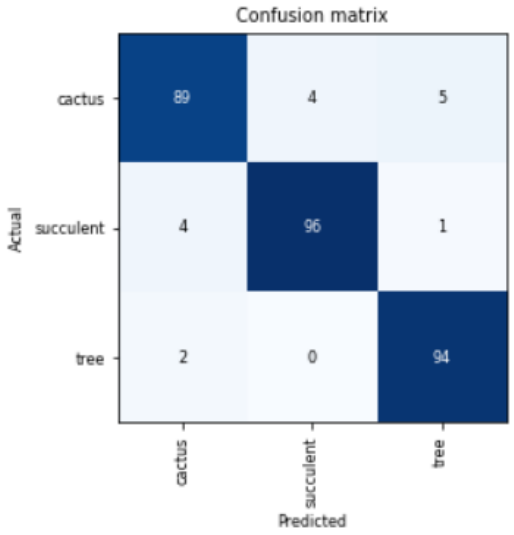
Now I have about 1500 balanced datas (500 cactus, 500 succulents, 500 trees) !

Third Model training

- confusion matrix

epoch	train_loss	valid_loss	error_rate	time
0	1.391104	0.351665	0.122034	00:13

epoch	train_loss	valid_loss	error_rate	time
0	0.637653	0.260290	0.084746	00:17
1	0.545796	0.254474	0.084746	00:17
2	0.443314	0.229113	0.064407	00:17
3	0.388013	0.214341	0.054237	00:18

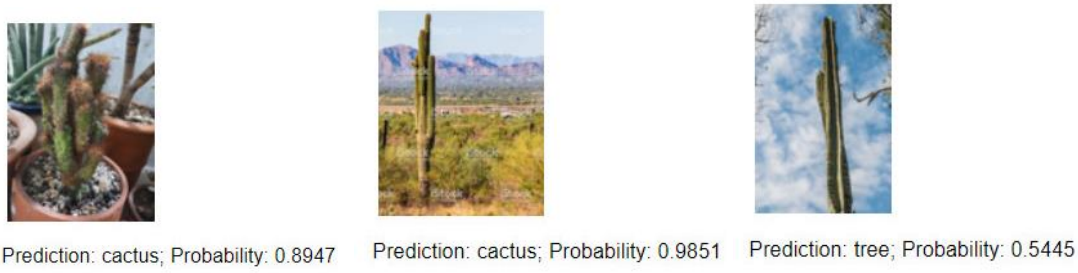


I got an amazing result! The error rate almost halved to 0.05. In addition, similar predictive figure s were obtained for cactus, succulents, and trees in the confusion matrix.

- 10 images with the highest loss

I deleted number 1 and 6 pictures because the model might be confused about what it is. I also deleted datas with too many letters or drawings, such as 2 and 8. This is because I think that rec ognizing drawings is a different problem to distinguish pictures of real plants. I left the rest of th e pictures as I thought they had the right shape for each category.

- Test prediction with real photos



I have only one cactus in my home, so I used some other cactus datas that I don't use in the trai ning. When tested on the previous model, it recognized my cactus as a tree, but now it can be r ecognized as a cactus ! But it's still a little confused between the cactus and the tree.

- Tree vs Cactus

To solve the problem of confusing trees and cactus, I added tall, long shaped cactus data. And t he trees were removed, leaving only the trees with the typical appearance.

At this point, I thought the dataset had enough diverse, clear, and balanced data, so I continued to use the dataset without modifying it anymore. Instead, I tried to improve the accuracy of the model while adjusting the model and the epoch of it.

Resnet & Epoch.

- Resnet 18 ’s epoch

I used resent 18 while I was doing all of this process. So first, I increased the number of epochs f or resnet18 to 10 epochs.

epoch	train_loss	valid_loss	error_rate	time
0	1.342631	0.564394	0.180887	00:13

epoch	train_loss	valid_loss	error_rate	time
0	0.746736	0.312008	0.105802	00:16
1	0.618998	0.287934	0.098976	00:16
2	0.523153	0.229447	0.064846	00:17
3	0.426554	0.250185	0.078498	00:17
4	0.365245	0.228946	0.068259	00:18
5	0.302795	0.251738	0.051195	00:22
6	0.269518	0.238539	0.054608	00:22
7	0.220500	0.228756	0.058020	00:26
8	0.191942	0.222629	0.054608	00:43
9	0.174081	0.220064	0.058020	00:26

I thought increasing the epoch would help to reduce the error rate, but if it went above a certain level, the error rate went up or maintained. Several experiments have shown that 4 to 5 epochs a re most appropriate.

- Resnet 34

epoch	train_loss	valid_loss	error_rate	time
0	1.328231	0.345512	0.129693	00:18

epoch	train_loss	valid_loss	error_rate	time
0	0.661442	0.229531	0.095563	00:27
1	0.490245	0.230721	0.044369	00:29
2	0.410319	0.185637	0.054608	00:29
3	0.326339	0.147278	0.037543	00:29
4	0.265441	0.141203	0.030717	00:29

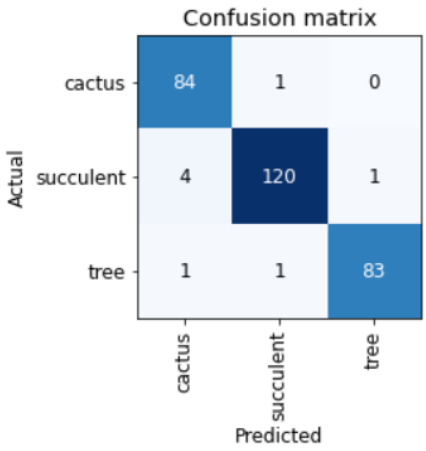
Wow! I could have the lowest error rate using Resnet 34. Maybe it's because it has more learned neuron layers. But later, I tried Resnet 50, but the error rate increased. I guess this is probably be cause of overfitting. Finally, the final data model was trained using the dataset and Resnet 34.

Final Model training

- confusion matrix

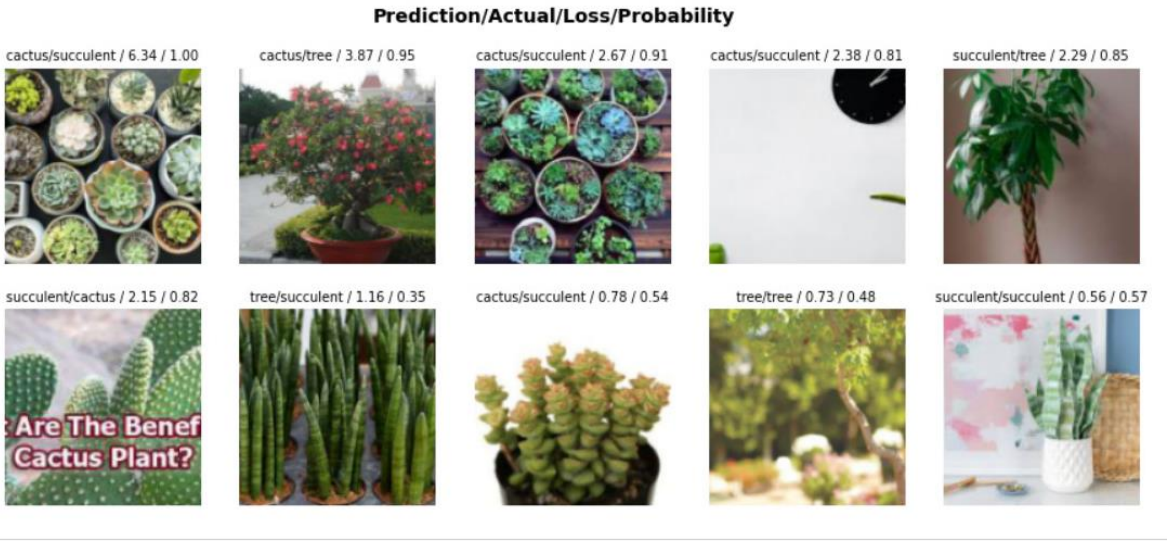
epoch	train_loss	valid_loss	error_rate	time
0	1.201589	0.297172	0.091525	00:18

epoch	train_loss	valid_loss	error_rate	time
0	0.759435	0.194894	0.061017	00:28
1	0.548360	0.134314	0.057627	00:28
2	0.439205	0.119896	0.030508	00:28
3	0.371041	0.103832	0.020339	00:29
4	0.295273	0.104620	0.027119	00:30



I was able to make the model with the lowest error rate so far. Although there seems to be a littl e bit of overfit on the succulent in the confusion matrix. But there were almost no incorrect predi ctions between the actual label and other labels!

- 10 images with the highest loss



Now I couldn't see the loss images anymore that I could see on the top 10 Loss image lis t before! Some datas (num 9, 10) are well predicted but here because it has low product

ivity. The main reason these almost images were printed here seems to be that there is n ot only one plant, and it is difficult to recognize multiple plants or clearly the whole plant .

- Test prediction with real photos

Prediction: tree; Probability: 0.9916

Prediction: tree; Probability: 1.0000

Prediction: tree; Probability: 1.0000

Prediction: succulent; Probability: 0.9997

Prediction: succulent; Probability: 0.9971

Prediction: succulent; Probability: 0.9812

Prediction: succulent; Probability: 0.9997

Prediction: succulent; Probability: 0.9998

Prediction: succulent; Probability: 0.9942

Prediction: cactus; Probability: 0.8947

Prediction: cactus; Probability: 0.9719

Prediction: cactus; Probability: 0.9999

Now I can distinguish trees, succulents, and cactus very well ! Some pictures of trees even had 1 00 percent accuracy. And the prediction rate for cactus classification is also increased.

Overfitting

Finally, I wanted to reduce the error rate to 0.01 or less, so I trained the model continuously with the same dataset. However, the error rate did not continue to decrease from 0.02, but the error r ate increased again and again. I guess it's because of overfitting. So I just use the model I traine d in the previous step.

Conclusion

- Clear dataset

When the model learns, it is very important to give clear and non-duplicate datas. The images sh ould have the appropriate file size and should be able to clearly show what label they belong to. In addition, images with too many clusters can interfere with learning. Collecting clear data is an important point.

- Model training

High model layers do not necessarily improve accuracy. Fine tuning so many times did not impr ove accuracy, too. I think that it is important to find appropriate points where the model can be well trained.