Shuohe Ren

Bryan Plummer

CS542

12/12/2020

Final Project Lab Report

This semester, I am taking Dr. Plummer's CS 542, and the final project is to solve two tasks addressing an image recognition topic. There is a dataset containing around 1K categories of plants with only about 250,000 images. The first task is about image classification: we care to identify around 250,000 images. The second task is Semi-Supervised/Few-Shot Learning. It is more advanced because we have some images we think contain the plant, but we only have a few labels. Our new goal is to develop an AI model that can learn from just these labeled examples. Aside from modifying and selecting the models, I also learned to use a website named "SCC" in Boston University providing us a chance to process some big data faster. After finishing this project, not only did my background of selecting models grow stronger, but I am also able to use a more decent tool to process some big datas.

After finishing task 1, I learned to increase the accuracy by adjusting the properate variable, and to use SCC for big data. Initially, my plan was to download the zip file, and ran offline so that I tried Anaconda and Colab, but my computer was not capable to handle so many images, and professor provided with a second solution: using a 32*32 model instead of the original version. I tried Dr. Plummer's code on the 12/10 while it could not run the last step: allocating the data into histograms since the routing was different from the one from SCC. Moreover, my classmate Sining Shen told me that the accuracy would be very low since the pictures are not as clear as the original one. Then, I just made an appointment on the website

named SCC. I used the code from Piazza and it looked plausible. To increase the accuracy, I changed the variable "epoch" and "learning rate" without switching to a new model. Now, I found out if we use a higher "epoch", it costs more time while the accuracy can have a higher record (if we fixed the model). Therefore, although my result was not perfect, but it was just about the time for running.

Although the second task has less pictures and the model can be modified from the first one, it is still tricky and more advanced since it is semi-supervised. I considered using my model among ResNet50, VGG16, VGG19, MobileNetV2, DenseNet201 because they are pre-trained with imagenet, and I picked ResNet50 since for model task1 and task2 does not have significant difference. However, we have to classify the dataset into two parts, labeled and unlabeled. It is also very interesting to allocate the part from the labeled dataset as much as we could to get the stable result and then start classifying the unlabeled result. For the variable epoch, I set 10 as a try. I spent around half an hour running this program from the website"SCC". I had two accuracy: the first one is 0.8 and the second was around 0.67 so I think it is much better than the code without change. After finishing task 2, I learned one more example for a semi-supervised model.

I also had a new experience using "SCC". Using "SCC" is much faster than using colab, or it is almost impossible to run from "Colab" so I am very glad to know this website. Here are some fun I would like to share, and some issues I suggest. I personally prefer a set short time slot and only one core so that I can have less time waiting. When I had a bug, I could fix them very quickly although the running time was slow so I can still do some other work when waiting for the results to come out. However, I suggest Boston University to have a better algorithm for the SCC user. I had a very long wait during the due days which made me a little bit disappointed and

nervous. I also suggest CS542 to put less pictures for the task one since my computer was very hot when doing the homework.

I am very glad to have this interesting project from learning by comparing better models, a new way to handle some big data. (I will put some data and diagrams to make it easier for the grader.)

50	thevour	0.42509	6	6h
51	Yutong Wang11	0.44700	3	4h
52	Luyin Fu	0.44715	8	6h
53	Yunshuang Tao	0.44990	15	2d
54	ZackHui	0.46961	2	11h
55	Callen Bragdon	0.48049	8	2d
56	Daisy Yu	0.48495	4	5h
57	Ali Hus	0.52434	5	2d
58	Max Cohen	0.52482	5	3d
59	Straw Hats	0.62075	6	8d
60	Winnie Xi Jr.	0.66580	2	36m

Your Best Entry ↑

Your submission scored 0.66580, which is an improvement of your previous score of 0.99313. Great job! Tweet this!

61	BlackWhiteAI	0.67317	12	2d
62	CharlieZ	0.68126	5	9h
63	Ali Siahkamari	0.68197	1	13d
64	Joe Iacoviello	0.69622	1	1d
65	Mikhail Kouzminov	0.70043	1	12h
66	Tinghe Cui	0.70345	13	6m
	provided code (no changes)	0.73922		

	baseline	0.88807		
4	mz	0.87254	15	5h
5	YifeiF	0.86356	27	2d
6	Bian Yuanchong	0.86274	15	7h
7	cs542Nov2020	0.86029	11	1d
8	Dav7566	0.85375	8	1d
9	Teletubbies	0.84885	13	4d
10	Marjament	0.84722	20	4d
11	Rock&Roll	0.84150	27	2d
12	frankchz	0.83905	22	18h
13	Pikaest	0.83333	14	3d
14	NoAcclmSad	0.83088	7	16h
15	Sining Shen	0.82516	1	2d
16	Qiancheng Fu	0.82026	38	2d
17	Winnie Xi Jr.	0.81781	3	13h

baseline

Your Best Entry ↑

Your submission scored 0.66666, which is not an improvement of your best score. Keep trying!

```
Epoch 1/5
5751/5751 [==============================] - 273s 48ms/step - loss: 5.4811 - accuracy: 0.0572 - sparse_top_k_categorical_accuracy: 0.1642 - val_loss: 4.9731 - val_accuracy: 0.0922 - val_sparse_top_k_categorical_accuracy: 0.2514
Epoch 2/5
5751/5751 [==============================] - 195s 34ms/step - loss: 4.5622 - accuracy: 0.1309 - sparse_top_k_categorical_accuracy: 0.3184 - val_loss: 4.5841 - val_accuracy: 0.1335 - val_sparse_top_k_categorical_accuracy: 0.3209
Epoch 3/5
5751/5751 [==============================] - 194s 34ms/step - loss: 4.0629 - accuracy: 0.1918 - sparse_top_k_categorical_accuracy: 0.4132 - val_loss: 4.4414 - val_accuracy: 0.1505 - val_sparse_top_k_categorical_accuracy: 0.3502
Epoch 4/5
5751/5751 [==============================] - 194s 34ms/step - loss: 3.5862 - accuracy: 0.2592 - sparse_top_k_categorical_accuracy: 0.5010 - val_loss: 4.5250 - val_accuracy: 0.1505 - val_sparse_top_k_categorical_accuracy: 0.3516
Epoch 5/5
5751/5751 [==============================] - 193s 34ms/step - loss: 3.0468 - accuracy: 0.3467 - sparse_top_k_categorical_accuracy: 0.5991 - val_loss: 4.9211 - val_accuracy: 0.1409 - val_sparse_top_k_categorical_accuracy: 0.3361
```

```
Iteration 1
Epoch 1/2
6/6 [==============================] - 2s 277ms/step - loss: 3.1210 - accuracy: 0.0355 - val_loss: 3.1184 - val_accuracy: 0.0568
Epoch 2/2
6/6 [==============================] - 1s 92ms/step - loss: 3.0405 - accuracy: 0.0780 - val_loss: 3.0780 - val_accuracy: 0.0671
Epoch 1/2
6/6 [==============================] - 1s 92ms/step - loss: 3.0386 - accuracy: 0.0496 - val_loss: 3.0386 - val_accuracy: 0.0775
Epoch 2/2
6/6 [==============================] - 1s 91ms/step - loss: 2.9283 - accuracy: 0.0638 - val_loss: 3.0007 - val_accuracy: 0.1033
Epoch 1/2
6/6 [==============================] - 1s 93ms/step - loss: 2.8425 - accuracy: 0.1418 - val_loss: 2.9626 - val_accuracy: 0.1153
Epoch 2/2
6/6 [==============================] - 1s 92ms/step - loss: 2.7560 - accuracy: 0.1560 - val_loss: 2.9245 - val_accuracy: 0.1308
Epoch 1/2
6/6 [==============================] - 1s 92ms/step - loss: 2.8424 - accuracy: 0.1064 - val_loss: 2.8882 - val_accuracy: 0.1515
Epoch 2/2
6/6 [==============================] - 1s 92ms/step - loss: 2.7289 - accuracy: 0.1844 - val_loss: 2.8516 - val_accuracy: 0.1738
Epoch 1/2
6/6 [==============================] - 1s 93ms/step - loss: 2.6512 - accuracy: 0.2411 - val_loss: 2.8152 - val_accuracy: 0.1790
Epoch 2/2
6/6 [==============================] - 1s 94ms/step - loss: 2.6449 - accuracy: 0.2695 - val_loss: 2.7796 - val_accuracy: 0.1945
Epoch 1/2
6/6 [==============================] - 1s 92ms/step - loss: 2.5545 - accuracy: 0.3050 - val_loss: 2.7442 - val_accuracy: 0.2100
Epoch 2/2
6/6 [==============================] - 1s 97ms/step - loss: 2.5119 - accuracy: 0.3121 - val_loss: 2.7085 - val_accuracy: 0.2375
Epoch 1/2
6/6 [==============================] - 1s 94ms/step - loss: 2.4500 - accuracy: 0.3475 - val_loss: 2.6742 - val_accuracy: 0.2633
Epoch 2/2
6/6 [==============================] - 1s 95ms/step - loss: 2.3983 - accuracy: 0.3759 - val_loss: 2.6387 - val_accuracy: 0.2737
Epoch 1/2
6/6 [==============================] - 1s 94ms/step - loss: 2.3792 - accuracy: 0.3830 - val_loss: 2.6054 - val_accuracy: 0.2857
Epoch 2/2
6/6 [==============================] - 1s 93ms/step - loss: 2.3899 - accuracy: 0.4043 - val_loss: 2.5730 - val_accuracy: 0.3046
Epoch 1/2
6/6 [==============================] - 1s 93ms/step - loss: 2.2604 - accuracy: 0.4752 - val_loss: 2.5393 - val_accuracy: 0.3201
Epoch 2/2
6/6 [==============================] - 1s 93ms/step - loss: 2.3054 - accuracy: 0.3972 - val_loss: 2.5071 - val_accuracy: 0.3287
number of unlabeled samples remained: 3788
Epoch 1/2
6/6 [==============================] - 1s 97ms/step - loss: 2.1969 - accuracy: 0.5106 - val_loss: 2.4754 - val_accuracy: 0.3408
Epoch 2/2
6/6 [==============================] - 1s 98ms/step - loss: 2.1951 - accuracy: 0.4965 - val_loss: 2.4452 - val_accuracy: 0.3580
Epoch 1/2
6/6 [==============================] - 1s 94ms/step - loss: 2.0511 - accuracy: 0.5887 - val_loss: 2.4142 - val_accuracy: 0.3769
Epoch 2/2
6/6 [==============================] - 1s 93ms/step - loss: 2.0669 - accuracy: 0.5248 - val_loss: 2.3852 - val_accuracy: 0.3890
```

```
[ ] hist7 = model_list[-1].fit(train_ds,validation_data=val_ds,epochs=10,shuffle=True)
```

```
Epoch 1/10
6/6 [==============================] - 1s 94ms/step - loss: 1.0251 - accuracy: 0.8227 - val_loss: 1.5999 - val_accuracy: 0.6196
Epoch 2/10
6/6 [==============================] - 1s 92ms/step - loss: 1.0501 - accuracy: 0.8227 - val_loss: 1.5904 - val_accuracy: 0.6213
Epoch 3/10
6/6 [==============================] - 1s 92ms/step - loss: 1.0547 - accuracy: 0.7801 - val_loss: 1.5815 - val_accuracy: 0.6248
Epoch 4/10
6/6 [==============================] - 1s 93ms/step - loss: 0.9799 - accuracy: 0.8227 - val_loss: 1.5727 - val_accuracy: 0.6265
Epoch 5/10
6/6 [==============================] - 1s 93ms/step - loss: 1.0387 - accuracy: 0.8298 - val_loss: 1.5627 - val_accuracy: 0.6282
Epoch 6/10
6/6 [==============================] - 1s 93ms/step - loss: 0.9999 - accuracy: 0.8156 - val_loss: 1.5535 - val_accuracy: 0.6299
Epoch 7/10
6/6 [==============================] - 1s 94ms/step - loss: 0.9475 - accuracy: 0.8511 - val_loss: 1.5449 - val_accuracy: 0.6334
Epoch 8/10
6/6 [==============================] - 1s 93ms/step - loss: 0.9318 - accuracy: 0.8582 - val_loss: 1.5358 - val_accuracy: 0.6334
Epoch 9/10
6/6 [==============================] - 1s 92ms/step - loss: 0.9426 - accuracy: 0.8369 - val_loss: 1.5275 - val_accuracy: 0.6351
Epoch 10/10
6/6 [==============================] - 1s 93ms/step - loss: 0.9532 - accuracy: 0.8440 - val_loss: 1.5196 - val_accuracy: 0.6351
```

```
[ ] np.save('hist7.npy',hist7.history)
```