Assignment 2

Wanyi Su
Student #: 301445656

Total # of Kaggle submission: 8.

Part 1

1. csv file has been submitted on Kaggle.
2. Kaggle information:
    - Name: Wanyi Su
    - Base performance: Best accuracy is 0.698 (best Kaggle submission)
3. Answers for grading scheme

Layer structure:

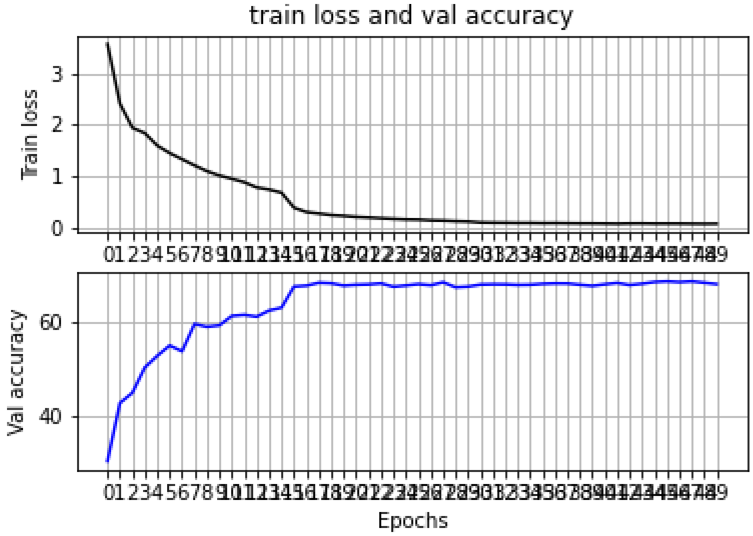
I chose VGGNet-16 as the layer structure to modify from, since they are both for the classification purpose. I made BaseNet network deeper by adding more (conv-normalization-ReLU)*4-avgpooling layers, followed by (linear-normalization-ReLU-dropout)*2-linear as FC layer. To avoid overfitting, I add a dropout layer in the FC layer. Also, I found expanding the number of channels helps performance improvement, which is similar to what VGGNet does. Since we want to see different learning behavior among epoches, we use a learning rate scheduler to decrease learning rate over different epochs starting from 0.001 with gamma=1.

Layer No.	Layer Type	Kernel Size (for conv layers)	Input | Output dimension	Input | Output Channels (for conv layers)
1	Conv2d	3	32 | 32	3 | 64
2	BatchNorm2d	-	32 | 32	-
3	ReLU	-	32 | 32	-
4	Conv2d	3	32 | 32	64 | 128
5	BatchNorm2d	-	32 | 32	-
6	ReLU	-	32 | 32	-
7	Conv2d	3	32 | 32	128 | 128
8	BatchNorm2d	-	32 | 32	-

9	ReLU	-	32 | 32	-
10	AvgPool2d	2	32 | 16	-
11	Conv2d	3	16 | 16	128 | 256
12	BatchNorm2d	-	16 | 16	-
13	ReLU	-	16 | 16	-
14	Conv2d	3	16 | 16	256 | 256
15	BatchNorm2d	-	16 | 16	-
16	ReLU	-	16 | 16	-
17	Conv2d	3	16 | 16	256 | 256
18	BatchNorm2d	-	16 | 16	-
19	ReLU	-	16 | 16	-
20	AvgPool2d	2	16 | 8	-
21	Conv2d	3	8 | 8	256 | 512
22	BatchNorm2d	-	8 | 8	-
23	ReLU	-	8 | 8	-
24	Conv2d	3	8 | 8	512 | 512
25	BatchNorm2d	-	8 | 8	-
26	ReLU	-	8 | 8	-
27	Conv2d	3	8 | 8	512 | 512
28	BatchNorm2d	-	8 | 8	-
29	ReLU	-	8 | 8	-
30	AvgPool2d	2	8 | 4	-
31	Conv2d	3	4 | 4	512 | 1024
32	BatchNorm2d	-	4 | 4	-

33	ReLU	-	4 | 4	-
34	Conv2d	3	4 | 4	1024 | 1024
35	BatchNorm2d	-	4 | 4	-
36	ReLU	-	4 | 4	-
37	Conv2d	3	4 | 4	1024 | 1024
38	BatchNorm2d	-	4 | 4	-
39	ReLU	-	4 | 4	-
40	AvgPool2d	2	4 | 2	-
41	Linear	-	4096 | 4096	-
42	BatchNorm1d	-	4096 | 4096	-
43	ReLU	-	4096 | 4096	-
44	Dropout	-	4096 | 4096	-
45	Linear	-	4096 | 4096	-
46	BatchNorm1d	-	4096 | 4096	-
47	ReLU	-	4096 | 4096	-
48	Dropout	-	4096 | 4096	-
49	Linear	-	4096 | 100	-

Plot:



Training loss experiences a rapid decrease from beginning to around the 16th epoch, then decreases slowly after below 0.1. Validation accuracy has an opposite trend to training loss. It increases at a rapid speed until the 16th epoch, after which it increases at a stable and slow speed towards 0.69. This performance trend corresponds to the learning rate scheduler used here, which has a larger lr in the first 15 epochs and a smaller lr after. This indicates the model functions and fits well to the dataset.

Ablation study:

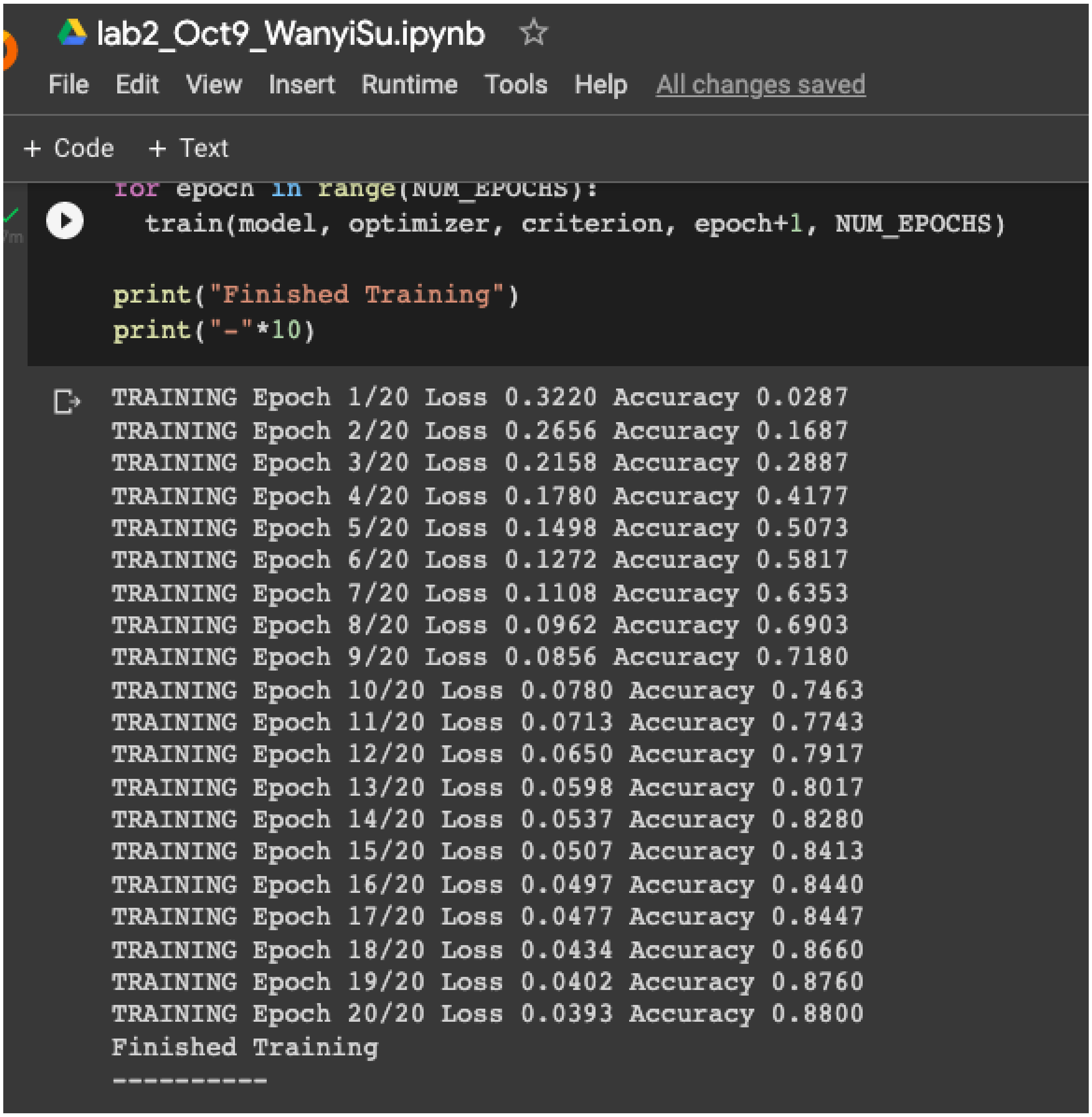
After some parameter setting and layer structure changes, we have the test accuracy (get from Kaggle submission) performance improvement of 14.4% from 0.61 to 0.698. The improvement reasons can be deeper layer structure in each conv block, larger output channels, dynamic learning rates, and larger batch size. Below are some details of the ablation study.

Trial (test accuracy 0.61 on kaggle)	Trial (test accuracy 0.698 on kaggle)
we use five conv blocks, with each block consisting of two (Conv2d-BatchNorm2d-ReLU) sets followed by a maxpooling layer, then finally a FC layer consisting of one (Linear-BatchNorm1d-ReLU-Linear) block.	we use four conv blocks, but make each block deeper, with each block consisting of three (Conv2d-BatchNorm2d-ReLU) sets followed by a avgpooling layer, then finally a FC layer consisting of two (Linear-BatchNorm1d-ReLU) block and then Linear layer.
Output channel is 512. Dimension is then expanded into 512*1*1(512) after the last conv block before before being pulled into FC layer and predict for the 100 classes.	We improve the output channel to 1024.. Dimension is then expanded into a much larger number of 1024*2*2 (4096) after the last conv block as VGGNet does before being

	pulled into FC layer to predict for the 100 classes.
We use 0.001 as the fixed learning rate and keep it the same through the whole training process.	We use a scheduler to dynamically change the learning rate among epochs through the training process, with 0.01 as the starting learning rate. scheduler = MultiStepLR(optimizer, [15, 30], gamma=0.1)
Set batch_size as 32.	Double batch_size as 64.

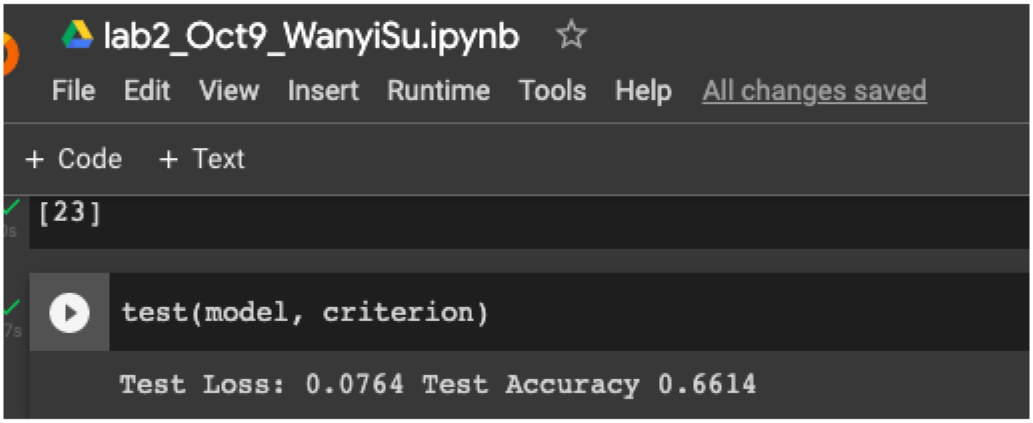
Part 2

1. Accuracies and screenshots
   - By fine-tuning the whole network (w/ RESNET_LAST_ONLY = False):
     - Train accuracy = 88%
     - Test accuracy = 66.14%
     - Screenshots as below:



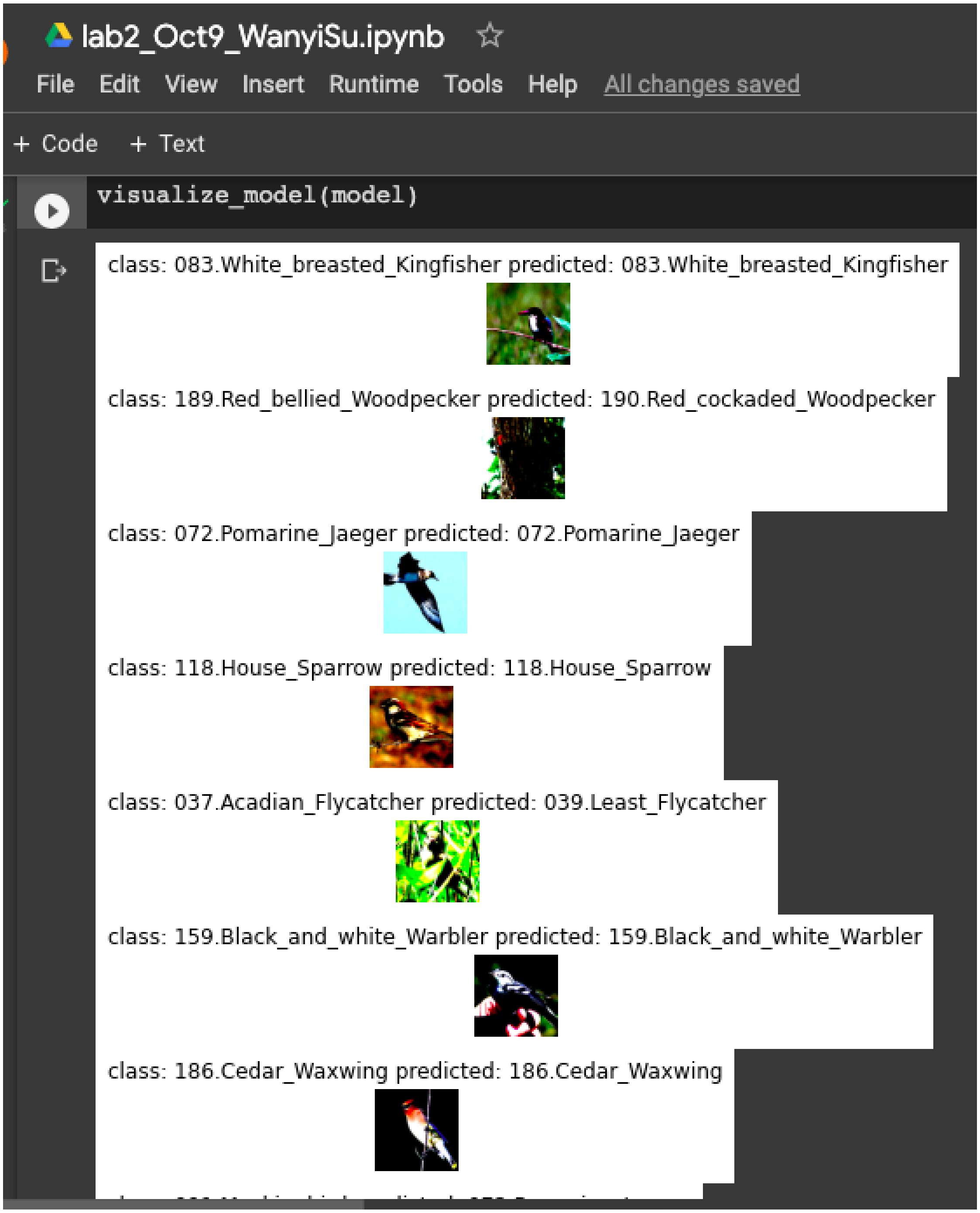
```
for epoch in range(NUM_EPOCHS):
  train(model, optimizer, criterion, epoch+1, NUM_EPOCHS)

print("Finished Training")
print("-"*10)
```

```
TRAINING Epoch 1/20 Loss 0.3220 Accuracy 0.0287
TRAINING Epoch 2/20 Loss 0.2656 Accuracy 0.1687
TRAINING Epoch 3/20 Loss 0.2158 Accuracy 0.2887
TRAINING Epoch 4/20 Loss 0.1780 Accuracy 0.4177
TRAINING Epoch 5/20 Loss 0.1498 Accuracy 0.5073
TRAINING Epoch 6/20 Loss 0.1272 Accuracy 0.5817
TRAINING Epoch 7/20 Loss 0.1108 Accuracy 0.6353
TRAINING Epoch 8/20 Loss 0.0962 Accuracy 0.6903
TRAINING Epoch 9/20 Loss 0.0856 Accuracy 0.7180
TRAINING Epoch 10/20 Loss 0.0780 Accuracy 0.7463
TRAINING Epoch 11/20 Loss 0.0713 Accuracy 0.7743
TRAINING Epoch 12/20 Loss 0.0650 Accuracy 0.7917
TRAINING Epoch 13/20 Loss 0.0598 Accuracy 0.8017
TRAINING Epoch 14/20 Loss 0.0537 Accuracy 0.8280
TRAINING Epoch 15/20 Loss 0.0507 Accuracy 0.8413
TRAINING Epoch 16/20 Loss 0.0497 Accuracy 0.8440
TRAINING Epoch 17/20 Loss 0.0477 Accuracy 0.8447
TRAINING Epoch 18/20 Loss 0.0434 Accuracy 0.8660
TRAINING Epoch 19/20 Loss 0.0402 Accuracy 0.8760
TRAINING Epoch 20/20 Loss 0.0393 Accuracy 0.8800
Finished Training
----------
```



```
test(model, criterion)
```

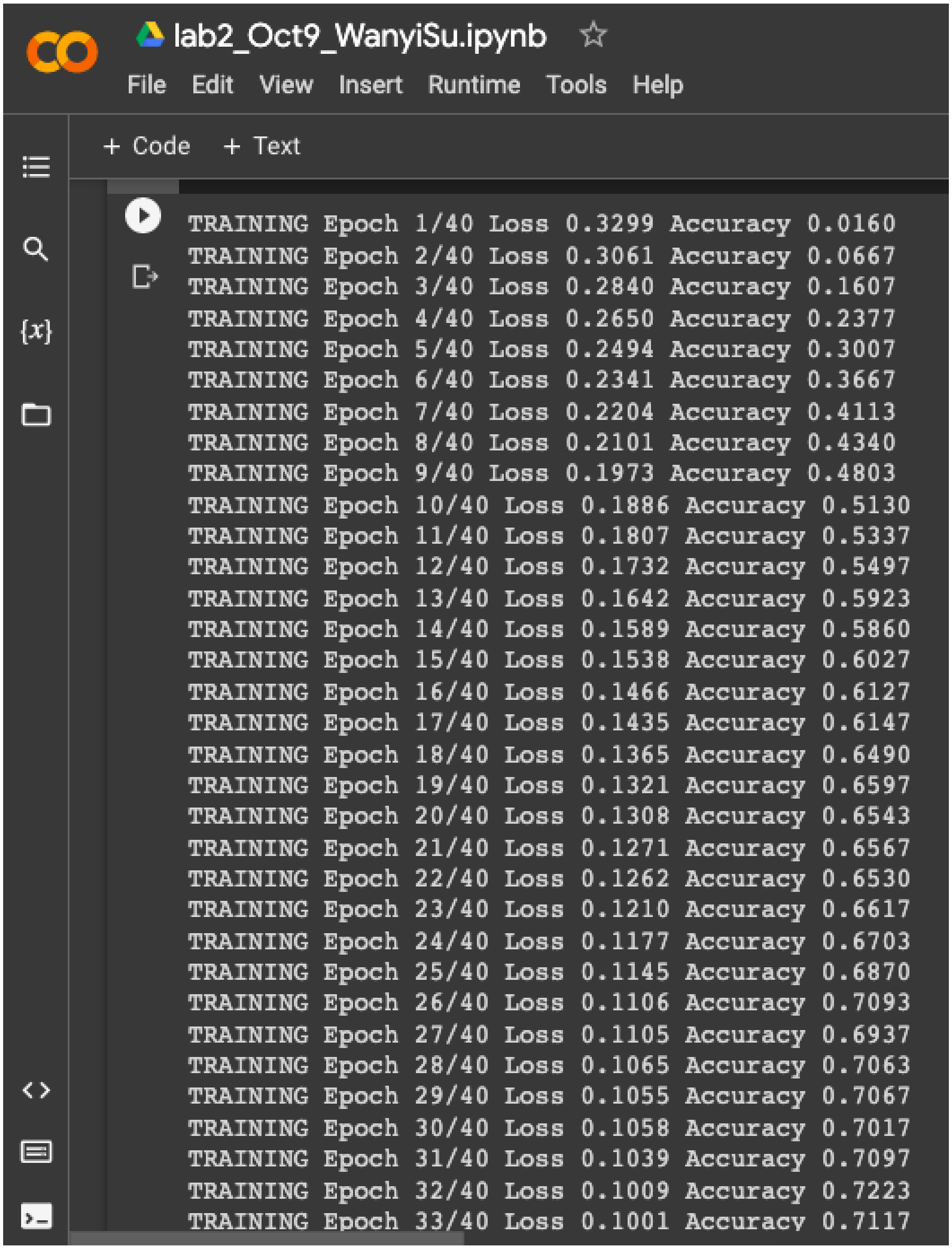
```
Test Loss: 0.0764 Test Accuracy 0.6614
```



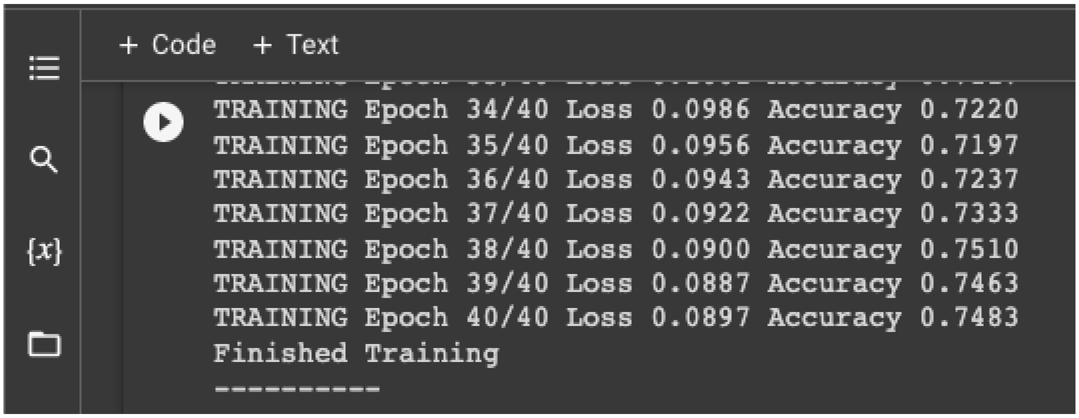
```
visualize_model(model)
```

```
class: 083.White_breasted_Kingfisher predicted: 083.White_breasted_Kingfisher
class: 189.Red_bellied_Woodpecker predicted: 190.Red_cockaded_Woodpecker
class: 072.Pomarine_Jaeger predicted: 072.Pomarine_Jaeger
class: 118.House_Sparrow predicted: 118.House_Sparrow
class: 037.Acadian_Flycatcher predicted: 039.Least_Flycatcher
class: 159.Black_and_white_Warbler predicted: 159.Black_and_white_Warbler
class: 186.Cedar_Waxwing predicted: 186.Cedar_Waxwing
```

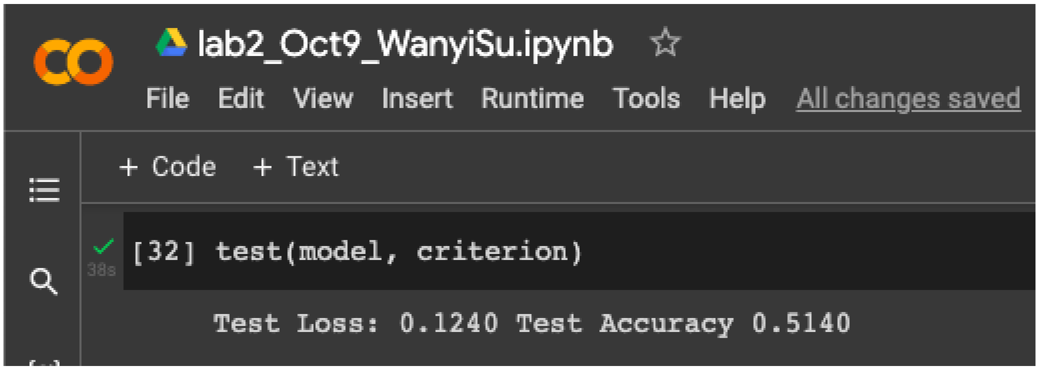
- As a fixed feature extractor (fine-tuning only last FC layer w/ RESNET_LAST_ONLY = True)
    - Train accuracy = 74.83%
    - Test accuracy = 51.4%
    - Screenshots as below:



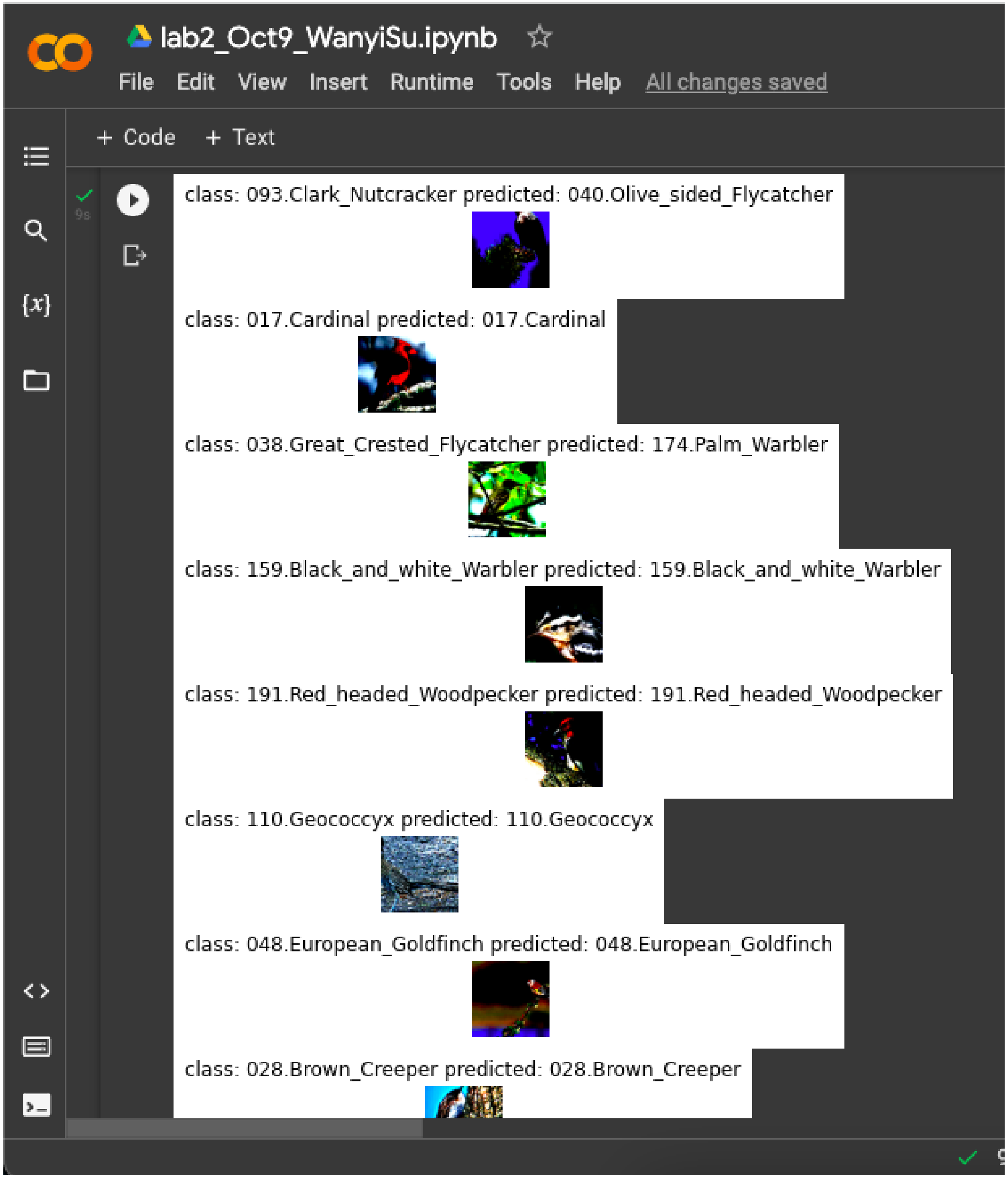
```
TRAINING Epoch 1/40 Loss 0.3299 Accuracy 0.0160
TRAINING Epoch 2/40 Loss 0.3061 Accuracy 0.0667
TRAINING Epoch 3/40 Loss 0.2840 Accuracy 0.1607
TRAINING Epoch 4/40 Loss 0.2650 Accuracy 0.2377
TRAINING Epoch 5/40 Loss 0.2494 Accuracy 0.3007
TRAINING Epoch 6/40 Loss 0.2341 Accuracy 0.3667
TRAINING Epoch 7/40 Loss 0.2204 Accuracy 0.4113
TRAINING Epoch 8/40 Loss 0.2101 Accuracy 0.4340
TRAINING Epoch 9/40 Loss 0.1973 Accuracy 0.4803
TRAINING Epoch 10/40 Loss 0.1886 Accuracy 0.5130
TRAINING Epoch 11/40 Loss 0.1807 Accuracy 0.5337
TRAINING Epoch 12/40 Loss 0.1732 Accuracy 0.5497
TRAINING Epoch 13/40 Loss 0.1642 Accuracy 0.5923
TRAINING Epoch 14/40 Loss 0.1589 Accuracy 0.5860
TRAINING Epoch 15/40 Loss 0.1538 Accuracy 0.6027
TRAINING Epoch 16/40 Loss 0.1466 Accuracy 0.6127
TRAINING Epoch 17/40 Loss 0.1435 Accuracy 0.6147
TRAINING Epoch 18/40 Loss 0.1365 Accuracy 0.6490
TRAINING Epoch 19/40 Loss 0.1321 Accuracy 0.6597
TRAINING Epoch 20/40 Loss 0.1308 Accuracy 0.6543
TRAINING Epoch 21/40 Loss 0.1271 Accuracy 0.6567
TRAINING Epoch 22/40 Loss 0.1262 Accuracy 0.6530
TRAINING Epoch 23/40 Loss 0.1210 Accuracy 0.6617
TRAINING Epoch 24/40 Loss 0.1177 Accuracy 0.6703
TRAINING Epoch 25/40 Loss 0.1145 Accuracy 0.6870
TRAINING Epoch 26/40 Loss 0.1106 Accuracy 0.7093
TRAINING Epoch 27/40 Loss 0.1105 Accuracy 0.6937
TRAINING Epoch 28/40 Loss 0.1065 Accuracy 0.7063
TRAINING Epoch 29/40 Loss 0.1055 Accuracy 0.7067
TRAINING Epoch 30/40 Loss 0.1058 Accuracy 0.7017
TRAINING Epoch 31/40 Loss 0.1039 Accuracy 0.7097
TRAINING Epoch 32/40 Loss 0.1009 Accuracy 0.7223
TRAINING Epoch 33/40 Loss 0.1001 Accuracy 0.7117
```



```
TRAINING Epoch 34/40 Loss 0.0986 Accuracy 0.7220
TRAINING Epoch 35/40 Loss 0.0956 Accuracy 0.7197
TRAINING Epoch 36/40 Loss 0.0943 Accuracy 0.7237
TRAINING Epoch 37/40 Loss 0.0922 Accuracy 0.7333
TRAINING Epoch 38/40 Loss 0.0900 Accuracy 0.7510
TRAINING Epoch 39/40 Loss 0.0887 Accuracy 0.7463
TRAINING Epoch 40/40 Loss 0.0897 Accuracy 0.7483
Finished Training
----------
```



```
[32] test(model, criterion)

     Test Loss: 0.1240 Test Accuracy 0.5140
```



```
class: 093.Clark_Nutcracker predicted: 040.Olive_sided_Flycatcher
class: 017.Cardinal predicted: 017.Cardinal
class: 038.Great_Crested_Flycatcher predicted: 174.Palm_Warbler
class: 159.Black_and_white_Warbler predicted: 159.Black_and_white_Warbler
class: 191.Red_headed_Woodpecker predicted: 191.Red_headed_Woodpecker
class: 110.Geococcyx predicted: 110.Geococcyx
class: 048.European_Goldfinch predicted: 048.European_Goldfinch
class: 028.Brown_Creeper predicted: 028.Brown_Creeper
```

2. Hyperparameter settings:
    - By fine-tuning the whole network
        - Batch_size = 16,
        - Learning_rate = 0.001,
        - Resnet_last_only = False,
        - num_epochs = 20
    - By fine-tuning only the last FC layer
        - Batch_size = 16,
        - Learning_rate = 0.001,
        - Resnet_last_only = True,
        - num_epochs = 40