CMPT 412

Project 2

Deep learning by PyTorch

Instructor : Yasutaka Furukawa

Name: Kaikun Fang

Student ID: 301416542

Note: Use 1 free late-day

Note: Use 1 free late-day

Part 1: Improving BaseNet on CIFAR100

1. Kaggle Group Name: 🍺

Members: Cheng Hu (301435966), Kaikun Fang (301416542)

2. Best accuracy:

The highest score of our group on Kaggle is 0.747

3. Layer Structure Table:

Layer No.	Layer Type	Kernel size (for conv layer)	Input | Output dimension	Input | Output Channels (for conv layers)
1	Conv2d	5	32 | 32	3 | 64
2	BatchNorm2d		32 | 32	
3	ReLU		32 | 32	
4	Conv2d	5	32 | 32	64 | 64
5	BatchNorm2d		32 | 32	
6	ReLU		32 | 32	
7	Conv2d	5	32 | 32	64 | 64
8	BatchNorm2d		32 | 32	
9	ReLU		32 | 32	
10	MaxPool2d	2	32 | 16	
11	Conv2d	5	16 | 16	64 | 128
12	BatchNorm2d		16 | 16	
13	ReLU		16 | 16	
14	Conv2d	5	16 | 16	128 | 128
15	BatchNorm2d		16 | 16	
16	ReLU		16 | 16	
17	Conv2d	5	16 | 16	128 | 128
18	BatchNorm2d		16 | 16	
19	ReLU		16 | 16	
20	Conv2d	5	16 | 16	128 | 256
21	BatchNorm2d		16 | 16	
22	ReLU		16 | 16	

23	Conv2d	5	16 | 16	256 | 256
24	BatchNorm2d		16 | 16	
25	ReLU		16 | 16	
26	Conv2d	5	16 | 16	256 | 256
27	BatchNorm2d		16 | 16	
28	ReLU		16 | 16	
29	MaxPool2d	2	16 | 8	
30	Conv2d	5	8 | 8	256 | 512
31	BatchNorm2d		8 | 8	
32	ReLU		8 | 8	
33	Conv2d	5	8 | 8	512 | 512
34	BatchNorm2d		8 | 8	
35	ReLU		8 | 8	
36	Conv2d	5	8 | 8	512 | 512
37	BatchNorm2d		8 | 8	
38	ReLU		8 | 8	
39	Conv2d	5	8 | 8	512 | 512
40	BatchNorm2d		8 | 8	
41	ReLU		8 | 8	
42	MaxPool2d	2	8 | 4	
43	Linear		8192 | 2000	
44	BatchNorm1d			
45	ReLU		2000 | 2000	
46	Dropout			
47	Linear		2000 | 2000	
48	BatchNorm1d			
49	ReLU		2000 | 2000	
50	Dropout			
51	Linear		2000 | 100	

I conceived my model structure based on the idea of VGG16 model, mainly every three layers ("Conv2d", "BatchNorm2d" and "ReLU") form a group, and then repeat the stacked groups to form three large groups. The last layer of each big group is "MaxPool2d", and the fc layer is processed after the operation of the three big groups.

4. plot.png:

To make it easier to debug the structure and parameters, I set the epoch at 20 at the beginning.

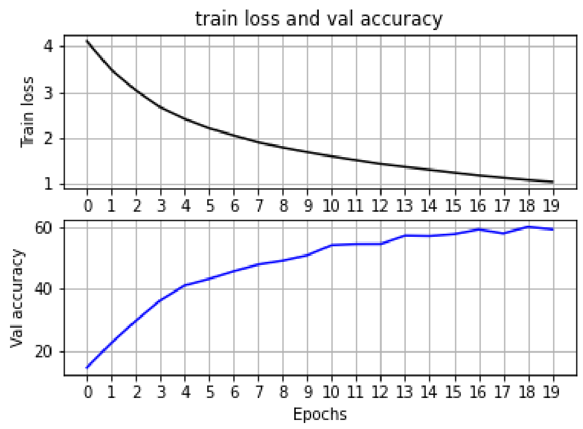


Figure 1 Epochs = 30

After several attempts at the latter, I found that keeping the structure and parameters unchanged, setting the epochs at 80 gives the best efficiency without leading to wasted computational units. val images has an accuracy of around 66% and achieves a score of 0.685 when uploaded to kaggle.

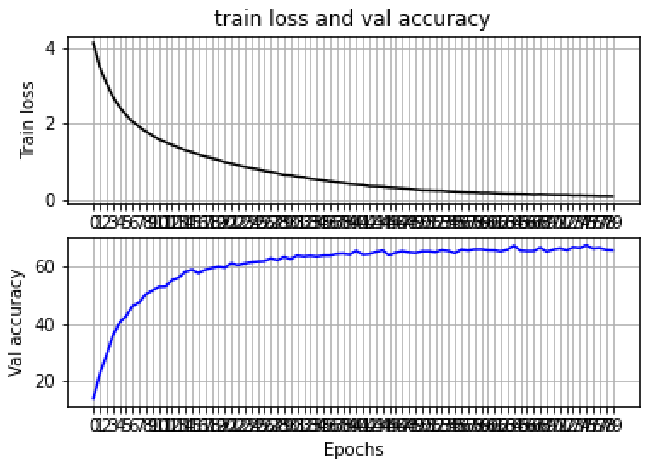


Figure 2 Epochs = 80

5. Ablation study:

I conducted an ablation study on the effect of Data normalization. Because of the limitation of the number of available computational units, I set the epoch at 20 for the experiment. I used the version with Normalize (transforms.Normalize ([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])) as the experimental group and the version without Normalize as the control group, while keeping the layer structure and other parameters the same. Running results :

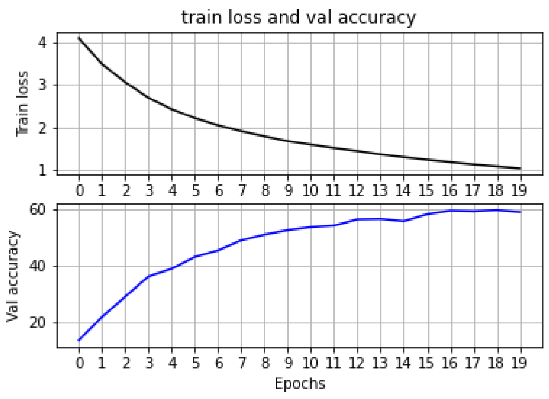


Figure 3 No Normalize

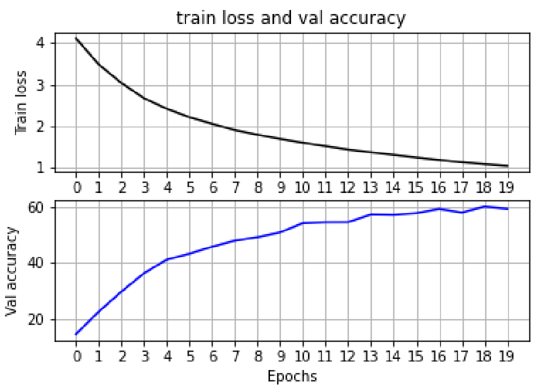
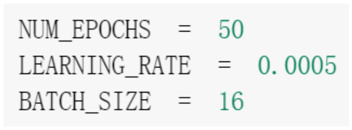


Figure 4 Normalize

From the data in the table, it appears that the difference between the control and experimental groups is not large. However, after uploading the CSV files to kaggle, it can be found that there is a gap between the two in terms of accuracy. The accuracy of the code without Normalize is 0.611, while the accuracy of the code with Normalize is 0.639. increases 2.8%. The difference should be more obvious if we increase the value of epochs.

Part 2: Transfer Learning

Base on resnet18. After many attempts, I set hyperparameter to the following values. (batch_size =16, learning_rate =0.0005, num_epochs =50).

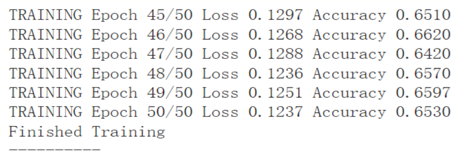


```
NUM_EPOCHS  =  50
LEARNING_RATE  =  0.0005
BATCH_SIZE  =  16
```

- **ResNet as fixed feature extractor** vs. **fine-tuning whole network:**

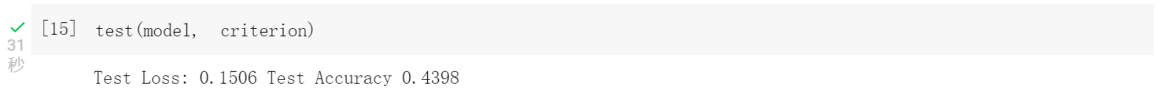
1. When ‘`RESNET_LAST_ONLY = True`’:

   Train Accuracy: About 65%



```
TRAINING Epoch 45/50 Loss 0.1297 Accuracy 0.6510
TRAINING Epoch 46/50 Loss 0.1268 Accuracy 0.6620
TRAINING Epoch 47/50 Loss 0.1288 Accuracy 0.6420
TRAINING Epoch 48/50 Loss 0.1236 Accuracy 0.6570
TRAINING Epoch 49/50 Loss 0.1251 Accuracy 0.6597
TRAINING Epoch 50/50 Loss 0.1237 Accuracy 0.6530
Finished Training
----------
```

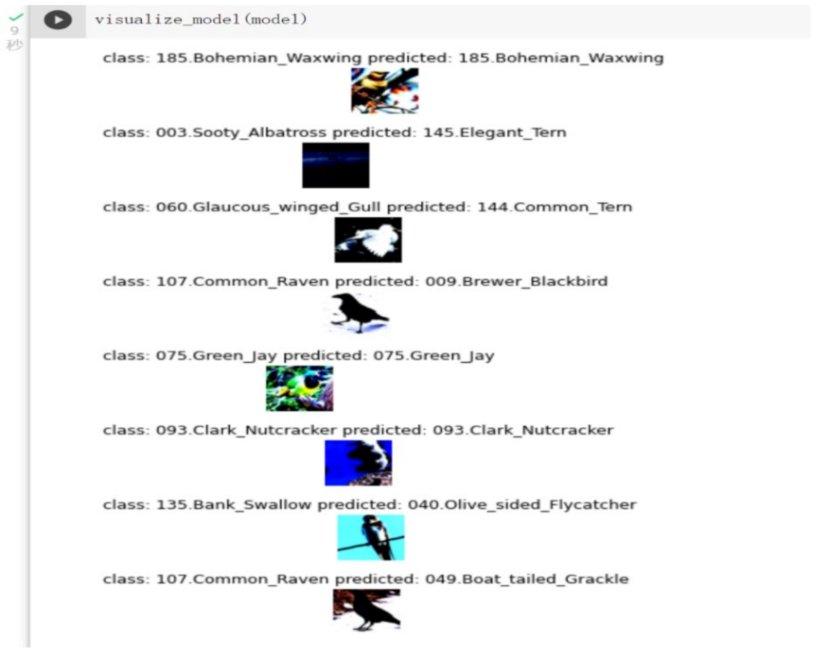
   Test Accuracy: 43.98%



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

      Test Loss: 0.1506 Test Accuracy 0.4398
```

   Visualizing:

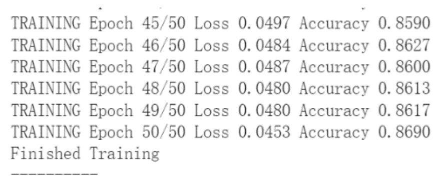


```
visualize_model(model)

class: 185.Bohemian_Waxwing predicted: 185.Bohemian_Waxwing
class: 003.Sooty_Albatross predicted: 145.Elegant_Tern
class: 060.Glaucous_winged_Gull predicted: 144.Common_Tern
class: 107.Common_Raven predicted: 009.Brewer_Blackbird
class: 075.Green_Jay predicted: 075.Green_Jay
class: 093.Clark_Nutcracker predicted: 093.Clark_Nutcracker
class: 135.Bank_Swallow predicted: 040.Olive_sided_Flycatcher
class: 107.Common_Raven predicted: 049.Boat_tailed_Grackle
```

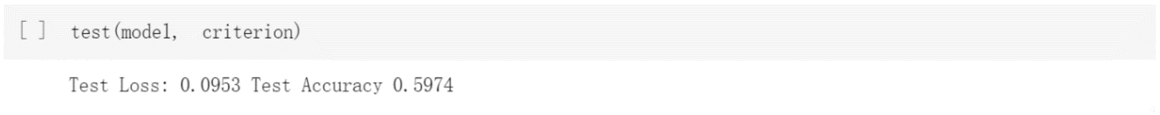
2. When ‘`RESNET_LAST_ONLY = False`’:

Train Accuracy: Above 86%



```
TRAINING Epoch 45/50 Loss 0.0497 Accuracy 0.8590
TRAINING Epoch 46/50 Loss 0.0484 Accuracy 0.8627
TRAINING Epoch 47/50 Loss 0.0487 Accuracy 0.8600
TRAINING Epoch 48/50 Loss 0.0480 Accuracy 0.8613
TRAINING Epoch 49/50 Loss 0.0480 Accuracy 0.8617
TRAINING Epoch 50/50 Loss 0.0453 Accuracy 0.8690
Finished Training
----------
```

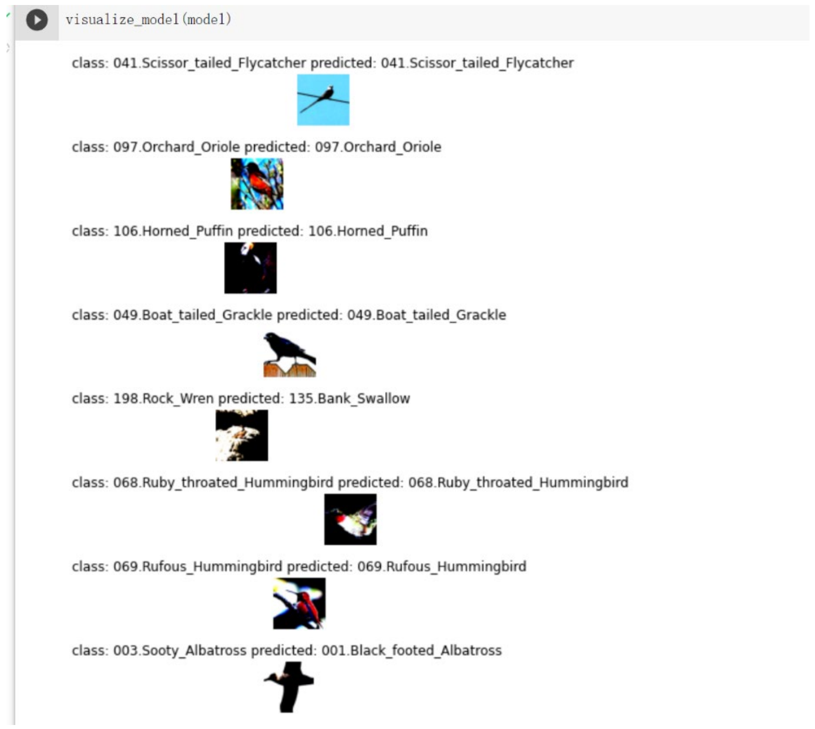
Test Accuracy: 59.74%



```
[ ]  test(model,  criterion)

     Test Loss: 0.0953 Test Accuracy 0.5974
```

Visualizing:



```
visualize_model(model)
```

class: 041.Scissor_tailed_Flycatcher predicted: 041.Scissor_tailed_Flycatcher

class: 097.Orchard_Oriole predicted: 097.Orchard_Oriole

class: 106.Horned_Puffin predicted: 106.Horned_Puffin

class: 049.Boat_tailed_Grackle predicted: 049.Boat_tailed_Grackle

class: 198.Rock_Wren predicted: 135.Bank_Swallow

class: 068.Ruby_throated_Hummingbird predicted: 068.Ruby_throated_Hummingbird

class: 069.Rufous_Hummingbird predicted: 069.Rufous_Hummingbird

class: 003.Sooty_Albatross predicted: 001.Black_footed_Albatross

Statement: The code inspiration and principles are derived from the videos and materials provided by the teacher. Discussions were also held with my group member(Cheng Hu 301435966).