Project 6
Deep learning by PyTorch

Due date: 23:59 Saturday 4/11th (2020)

My id in Kaggle is sepide sarajian.

Layer no	Layer type	Kernel size	Input|output Dimension	Input | output channels
1	conv2d	5	32 | 28	3 | 64
2	Batch normalization	-	28 | 28	-
3	relu	-	28 | 28	-
4	conv2d	5	28 | 24	64 | 256
5	Batch normalization	-	24 | 24	-
6	relu	-	24 | 24	-
7	Max pooling	2	24 | 12	-
8	conv2d	5	12 | 8	256 | 512
9	Batch normalization	-	8 | 8	-
10	relu	-	8 | 8	-
11	conv2d	5	8 | 4	512 | 1024
12	Batch normalization	-	4 | 4	-
13	relu	-	4 | 4	-

14	Average pooling	4	4 | 1	-

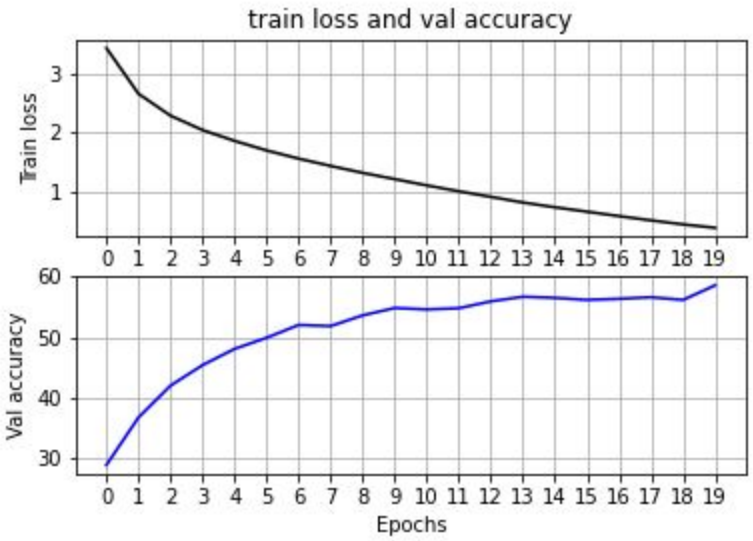
First I try running the network without the normalization which got the following result:

```
[1] loss: 3.426
Accuracy of the network on the val images: 27 %
[2] loss: 2.654
Accuracy of the network on the val images: 35 %
[3] loss: 2.292
Accuracy of the network on the val images: 42 %
[4] loss: 2.051
Accuracy of the network on the val images: 45 %
[5] loss: 1.857
Accuracy of the network on the val images: 47 %
[6] loss: 1.699
Accuracy of the network on the val images: 49 %
[7] loss: 1.571
Accuracy of the network on the val images: 50 %
[8] loss: 1.442
Accuracy of the network on the val images: 52 %
[9] loss: 1.323
Accuracy of the network on the val images: 52 %
[10] loss: 1.205
Accuracy of the network on the val images: 54 %
[11] loss: 1.105
Accuracy of the network on the val images: 54 %
[12] loss: 1.003
Accuracy of the network on the val images: 55 %
[13] loss: 0.910
Accuracy of the network on the val images: 55 %
[14] loss: 0.819
Accuracy of the network on the val images: 55 %
[15] loss: 0.729
Accuracy of the network on the val images: 56 %
[16] loss: 0.652
Accuracy of the network on the val images: 56 %
[17] loss: 0.576
Accuracy of the network on the val images: 56 %
[18] loss: 0.499
Accuracy of the network on the val images: 57 %
[19] loss: 0.450
Accuracy of the network on the val images: 56 %
[20] loss: 0.390
Accuracy of the network on the val images: 56 %
Finished Training
```

And I got 0.54300 accuracy in the test set from Kaggle.
Then I added the normalization and got the following:
Result of training for 20 batches which gave 58% accuracy on the validation set:

```
[1] loss: 3.441
Accuracy of the network on the val images: 28 %
[2] loss: 2.665
Accuracy of the network on the val images: 36 %
[3] loss: 2.293
Accuracy of the network on the val images: 41 %
[4] loss: 2.048
Accuracy of the network on the val images: 45 %
[5] loss: 1.864
Accuracy of the network on the val images: 48 %
[6] loss: 1.701
Accuracy of the network on the val images: 49 %
[7] loss: 1.562
Accuracy of the network on the val images: 52 %
[8] loss: 1.438
Accuracy of the network on the val images: 51 %
[9] loss: 1.320
Accuracy of the network on the val images: 53 %
[10] loss: 1.214
Accuracy of the network on the val images: 54 %
[11] loss: 1.107
Accuracy of the network on the val images: 54 %
[12] loss: 1.007
Accuracy of the network on the val images: 54 %
[13] loss: 0.912
Accuracy of the network on the val images: 55 %
[14] loss: 0.817
Accuracy of the network on the val images: 56 %
[15] loss: 0.735
Accuracy of the network on the val images: 56 %
[16] loss: 0.661
Accuracy of the network on the val images: 56 %
[17] loss: 0.585
Accuracy of the network on the val images: 56 %
[18] loss: 0.513
Accuracy of the network on the val images: 56 %
[19] loss: 0.444
Accuracy of the network on the val images: 56 %
[20] loss: 0.387
Accuracy of the network on the val images: 58 %
Finished Training
```

I got 0.56799 accuracy on the test database from my submission in Kaggle.



Part 2:

I tried adding L2 normalization and adam optimizer but got lower accuracy:
`Test Loss: 0.1628 Test Accuracy 0.2967`

For the final result I made the following changes to the code:

1. Setting the gradients to True

```
    #Set gradients to false
    if feature_extracting:
      for param in self.resnet18.parameters():
          param.requires_grad = True
```

2. Setting the hyperparameters:

```
NUM_EPOCHS = 40
LEARNING_RATE = 0.001
BATCH_SIZE = 10
```

3. Data augmentation:

```
data_transforms = {
    'train': transforms.Compose([
```

```
        transforms.Resize(256),
        # transforms.CenterCrop(224),
        transforms.RandomCrop(224),
        transforms.RandomHorizontalFlip(),
        #TODO: Transforms.RandomResizedCrop() instead of CenterCrop(),
RandomRoate() and Horizontal Flip()
        transforms.ToTensor(),
        #TODO: Transforms.Normalize()
    ]),
    'test': transforms.Compose([
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        #TODO: Transforms.Normalize()
    ]),
}
```

```
RESNET_LAST_ONLY = False #Fine tunes only the last layer. Set to False to
fine tune entire network
```

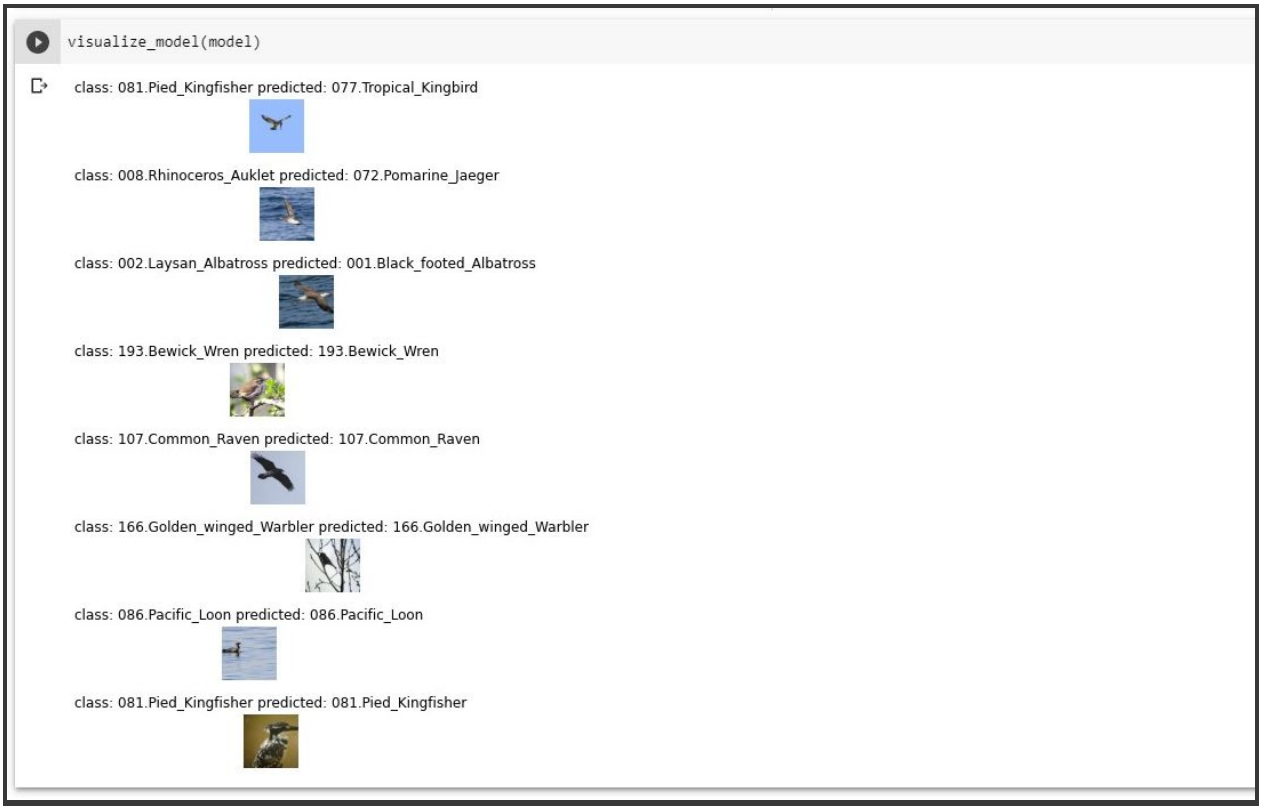
These are the result I got with fine tuning the whole resNet:
(staring from epoch 21)

```
TRAINING Epoch 21/40 Loss 0.0179 Accuracy 0.9830
TRAINING Epoch 22/40 Loss 0.0169 Accuracy 0.9850
TRAINING Epoch 23/40 Loss 0.0159 Accuracy 0.9850
TRAINING Epoch 24/40 Loss 0.0135 Accuracy 0.9890
TRAINING Epoch 25/40 Loss 0.0135 Accuracy 0.9890
TRAINING Epoch 26/40 Loss 0.0123 Accuracy 0.9903
TRAINING Epoch 27/40 Loss 0.0108 Accuracy 0.9907
TRAINING Epoch 28/40 Loss 0.0099 Accuracy 0.9933
TRAINING Epoch 29/40 Loss 0.0095 Accuracy 0.9923
TRAINING Epoch 30/40 Loss 0.0087 Accuracy 0.9927
Test Loss: 0.1424 Test Accuracy 0.6337
TRAINING Epoch 31/40 Loss 0.0084 Accuracy 0.9950
TRAINING Epoch 32/40 Loss 0.0075 Accuracy 0.9947
TRAINING Epoch 33/40 Loss 0.0083 Accuracy 0.9927
TRAINING Epoch 34/40 Loss 0.0071 Accuracy 0.9947
```

```
TRAINING Epoch 35/40 Loss 0.0074 Accuracy 0.9933
TRAINING Epoch 36/40 Loss 0.0065 Accuracy 0.9940
TRAINING Epoch 37/40 Loss 0.0061 Accuracy 0.9963
TRAINING Epoch 38/40 Loss 0.0064 Accuracy 0.9937
TRAINING Epoch 39/40 Loss 0.0059 Accuracy 0.9957
TRAINING Epoch 40/40 Loss 0.0050 Accuracy 0.9973
Finished Training
```

```
Test Loss: 0.1405 Test Accuracy 0.6456
```

Visualisation:




```
RESNET_LAST_ONLY = True #Fine tunes only the last layer. Set to False to
fine tune entire network
```

Then I tried only training on the final layer:
(staring from epoch 21)

```
Test Loss: 0.1423 Test Accuracy 0.6268
TRAINING Epoch 21/40 Loss 0.0167 Accuracy 0.9873
```

```
TRAINING Epoch 22/40 Loss 0.0160 Accuracy 0.9887
TRAINING Epoch 23/40 Loss 0.0142 Accuracy 0.9880
TRAINING Epoch 24/40 Loss 0.0141 Accuracy 0.9883
TRAINING Epoch 25/40 Loss 0.0133 Accuracy 0.9913
TRAINING Epoch 26/40 Loss 0.0115 Accuracy 0.9897
TRAINING Epoch 27/40 Loss 0.0115 Accuracy 0.9897
TRAINING Epoch 28/40 Loss 0.0108 Accuracy 0.9920
TRAINING Epoch 29/40 Loss 0.0098 Accuracy 0.9933
TRAINING Epoch 30/40 Loss 0.0083 Accuracy 0.9963
Test Loss: 0.1433 Test Accuracy 0.6337
TRAINING Epoch 31/40 Loss 0.0084 Accuracy 0.9940
TRAINING Epoch 32/40 Loss 0.0072 Accuracy 0.9967
TRAINING Epoch 33/40 Loss 0.0079 Accuracy 0.9947
TRAINING Epoch 34/40 Loss 0.0077 Accuracy 0.9933
TRAINING Epoch 35/40 Loss 0.0071 Accuracy 0.9937
TRAINING Epoch 36/40 Loss 0.0068 Accuracy 0.9930
TRAINING Epoch 37/40 Loss 0.0067 Accuracy 0.9950
TRAINING Epoch 38/40 Loss 0.0066 Accuracy 0.9937
TRAINING Epoch 39/40 Loss 0.0055 Accuracy 0.9970
TRAINING Epoch 40/40 Loss 0.0054 Accuracy 0.9960
Finished Training
```

```
Test Loss: 0.1433 Test Accuracy 0.6403
```

Visualisation:

```
visualize_model(model)
```

```
class: 024.Red_faced_Cormorant predicted: 024.Red_faced_Cormorant
```

```
class: 195.Carolina_Wren predicted: 198.Rock_Wren
```

```
class: 080.Green_Kingfisher predicted: 080.Green_Kingfisher
```

```
class: 183.Northern_Waterthrush predicted: 183.Northern_Waterthrush
```

```
class: 183.Northern_Waterthrush predicted: 183.Northern_Waterthrush
```

```
class: 185.Bohemian_Waxwing predicted: 185.Bohemian_Waxwing
```

```
class: 096.Hooded_Oriole predicted: 096.Hooded_Oriole
```

```
class: 059.California_Gull predicted: 064.Ring_billed_Gull
```