Lab 3 - Diabetic Retinopathy Detection

0516069 翁英傑

1. Introduction (20%)

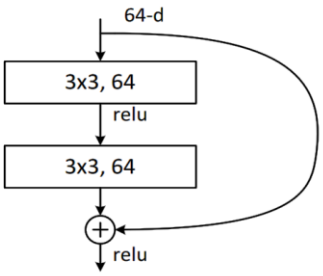
Implement a resnet18 and resnet50 to train on Diabetic Retinopathy Detection dataset. Compare with the pre trained model from torchvision that was pre trained on Imagenet dataset.

2. Experiment set up (30%)

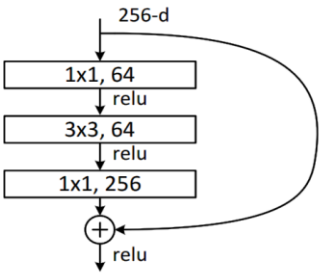
A. ResNet :

ResNet can be composed by two blocks :

Basic block



Bottleneck



Where resnet18 with basic block and resnet50 with bottleneck

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

B. Dataloader :

```
self.transform_train = transforms.Compose([
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
])

self.transform_test = transforms.Compose([
        transforms.ToTensor(),
])
```

Transform data from PIL into tensor, which normalize the value to [0, 1]. Also random flip the data while training with probability = 0.5.

C. Confusion Matrix :

It counts the percentage of what class was a specific class was classified into. The larger the diagonal is, the better the performance is.

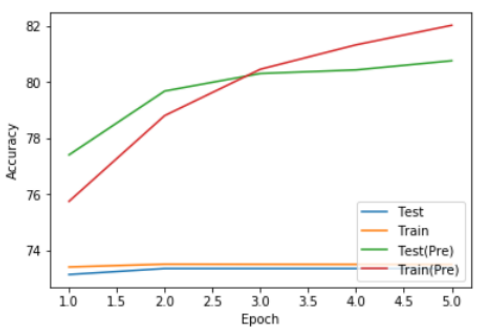
3. Experimental results (30%)

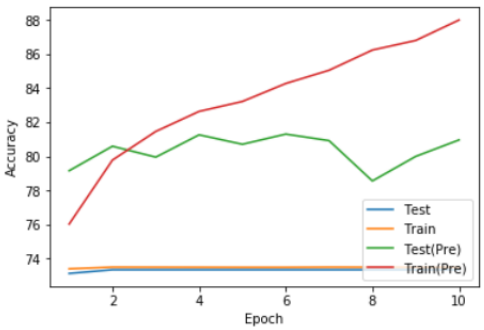
A. Highest accuracy :

```
Epoch: 6
 [==============================================================> 3513/3513 ms | Tot: 10m16s | Loss: 0.489 | Acc: 83.515% (23467/28099))0)
 [==============================================================> 879/879 11ms | Tot: 45s777ms | Loss: 0.575 | Acc: 82.021% (5762/7025)
Saving..
```

B. Compare two structure with pre-trained weight and without :

ResNet18 ResNet50





Hyper Parameters :

Batch size = 8

Learning rate = 0.001

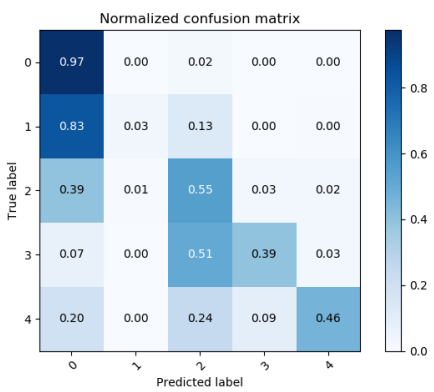
Epochs = 2000 (ResNet18) / 10 (ResNet50)

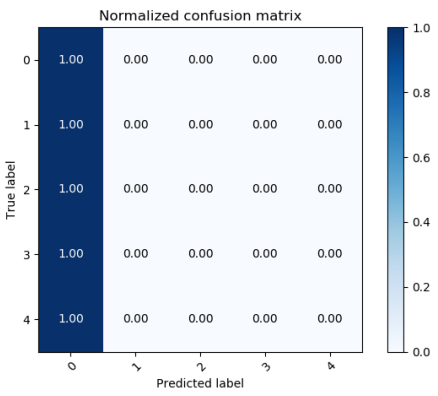
Optimizer = SGD (momentum=0.9)

Loss function = Cross Entropy Loss

C. Confusion Matrix :

ResNet50 (pre-trained) ResNet18





Total Accuracy : 82.021% Total Accuracy : 73.3%

4. Discussion (20%)

From the confusion matrix, we can see that network tends to learn to classify everything to class 0. This is cause by the unbalance data numbers in each class. Most of the data are from class 0 obviously. So a pre trained weight could be useful.

We can clearly see that in this dataset, initial value of the network is very critical. With the pre-train weight on Imagenet, the network has started with a decent feature extractor, which leads a successful training.

Another problem with this dataset is overfitting. We can clearly see from the accuracy graph, that testing accuracy stops increasing even with the training accuracy increasing. Dropout is the solution to this situation.