哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

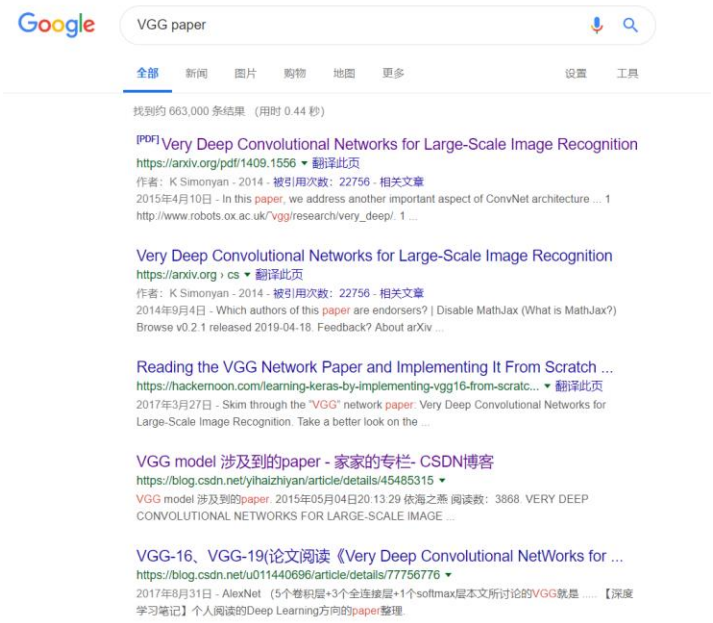
lab3

实验报告

组员姓名学号	1160300122-谢根琳 1160300426-李国建
班号	1603107

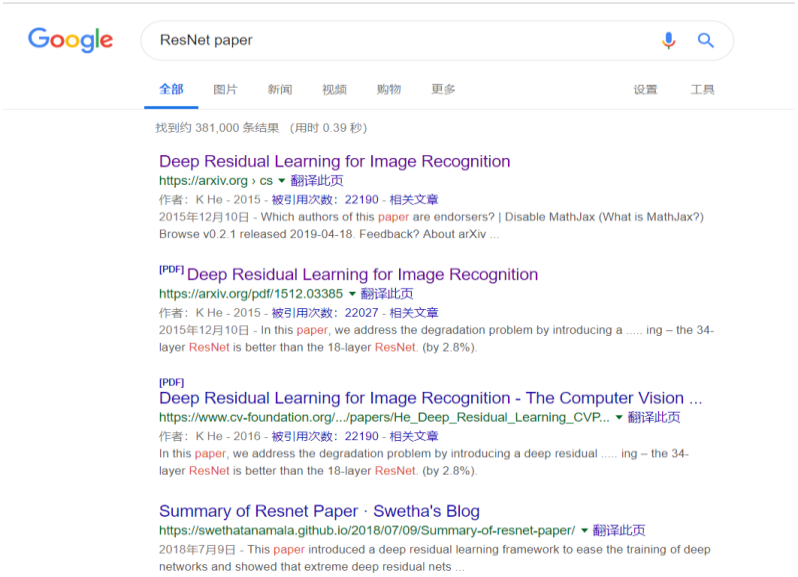
一、 论文搜索

- VGG 搜索过程：
  谷歌搜索 ： VGG paper
  得到如下结果：



点击第一个就是 VGG 的论文，但还不是很确定，点击第四个，发现二者的论文是一个，确定找到该论文：Very Deep Convolutional Networks for Large-Scale Image Recognition
链接：https://arxiv.org/pdf/1409.1556.pdf

- 谷歌搜索：ResNet paper



前两条都指向一个 paper：Deep Residual Learning for Image Recognition
链接：https://arxiv.org/pdf/1512.03385.pdf

二、 手动实现 dataset

继承了 torch.utils.data.Dataset， 然后重写 __len__ 方法 和 __getitem__ 方法。在初始化的函数中读取整个数据集，将数据和标签分别存储在 data 和 targets 属性中。利用 cifar-10 官网提供的数据集读取方式，读取整个数据集，然后分别在 init 和 getitem 方法中对数据进行处理。返回格式参考 dataset 函数。

三、 argparse 的使用

```
parser = argparse.ArgumentParser()
parser.add_argument('--cuda', default='GPU', type=str, choices=['GPU', 'CPU'], help='选择使用cpu/gpu')
args = parser.parse_args()
if args.cuda == 'GPU':
    device = torch.device('cuda')
elif args.cuda == 'CPU':
    device = torch.device('cpu')
```

利用如上的代码段进行设置，参数名为 cuda，默认值设为’ GPU’ ,当为‘GPU’时，torch.device(‘cuda’)；当将 cuda 参数输入为‘CPU’时，torch.device(‘cpu’)。

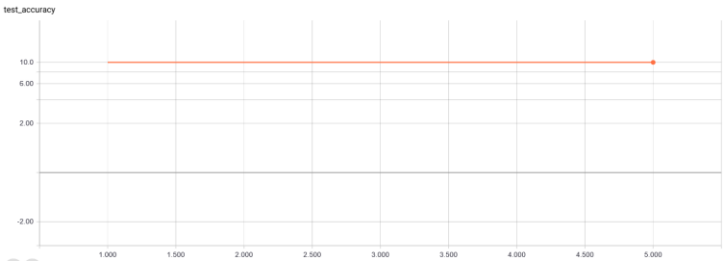
四、 VGG-11 的 LR 对比：

选用 Adam 作为优化器。均为经过 5 个 epoch 的结果。由对比可发现，当学习率设为 0.05 和 0.01 都出现学习率过高而导致的 loss 不变的情况，同时学习率为 0.0001 时 loss 下降就会变慢，这就属于学习率设置地稍低的问题。而当将学习率设置为 0.001 时，可以达到较好的效果。但是加上 BN 层后进行批标准化就可以改善这个现象。

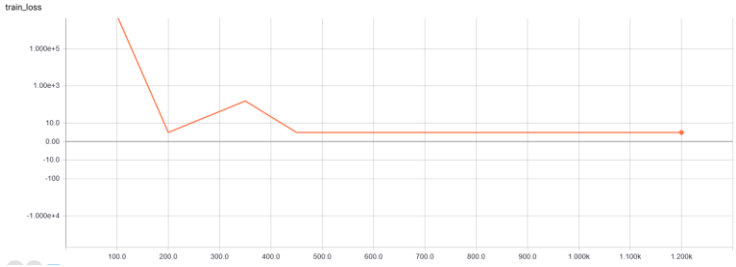
1） LR 设为 0.05

```
===> epoch: 1/5
train:
*****训练集预测准确率为: 10.212%
test:
*****测试集预测准确率为: 10.000%

===> epoch: 2/5
train:
*****训练集预测准确率为: 9.956%
test:
*****测试集预测准确率为: 10.000%

===> epoch: 3/5
train:
*****训练集预测准确率为: 9.790%
test:
*****测试集预测准确率为: 10.000%

===> epoch: 4/5
train:
*****训练集预测准确率为: 9.964%
test:
*****测试集预测准确率为: 10.000%

===> epoch: 5/5
train:
*****训练集预测准确率为: 10.030%
test:
*****测试集预测准确率为: 10.000%
*****测试集最高准确率为: 10.000%
```

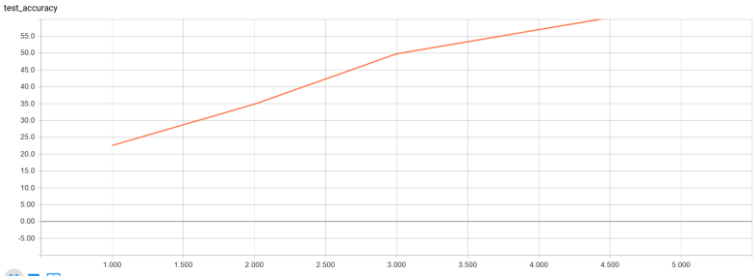
测试集准确率曲线：



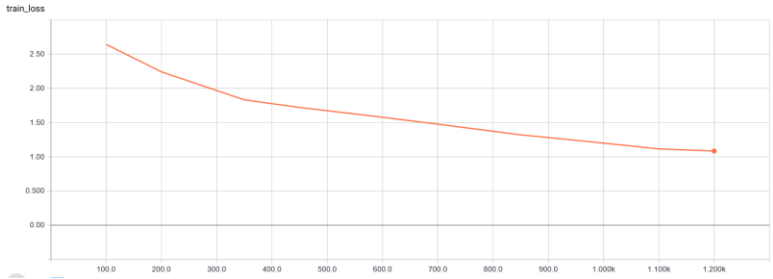
训练集 loss 曲线：



加 BN 层测试集准确率曲线：



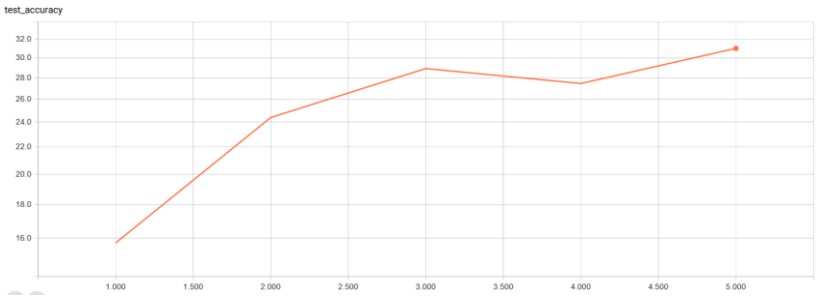
加 BN 层训练集 loss 曲线：



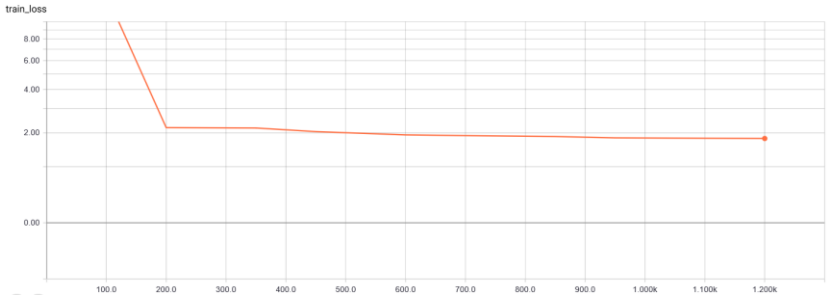
2） LR 设为 0.01

```
===> epoch: 1/5
train:
*****训练集预测准确率为: 13.426%
test:
*****测试集预测准确率为: 15.740%

===> epoch: 2/5
train:
*****训练集预测准确率为: 20.062%
test:
*****测试集预测准确率为: 24.370%

===> epoch: 3/5
train:
*****训练集预测准确率为: 26.576%
test:
*****测试集预测准确率为: 28.910%

===> epoch: 4/5
train:
*****训练集预测准确率为: 28.746%
test:
*****测试集预测准确率为: 27.440%

===> epoch: 5/5
train:
*****训练集预测准确率为: 30.374%
test:
*****测试集预测准确率为: 31.020%
*****测试集最高准确率为: 31.020%
```

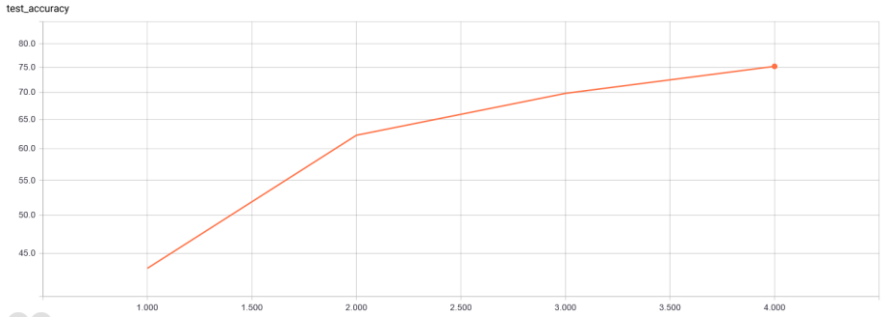
测试集准确率曲线：



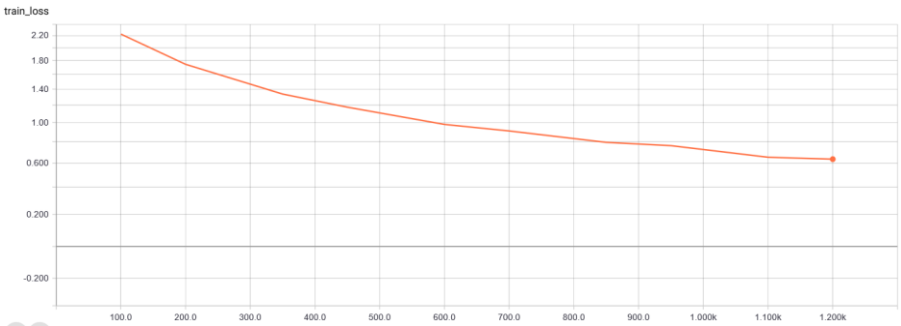
训练集 loss 曲线：



加 BN 层测试集准确率曲线：



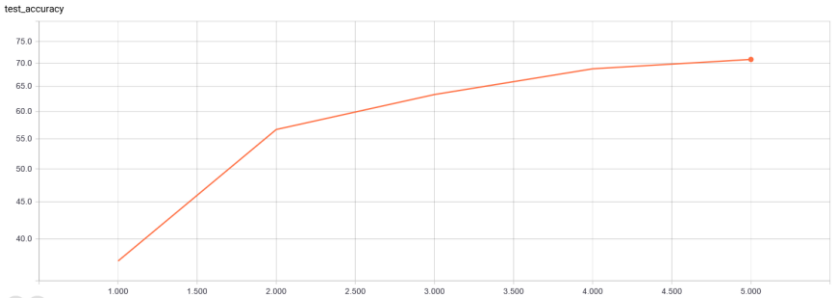
加 BN 层训练集 loss 曲线：



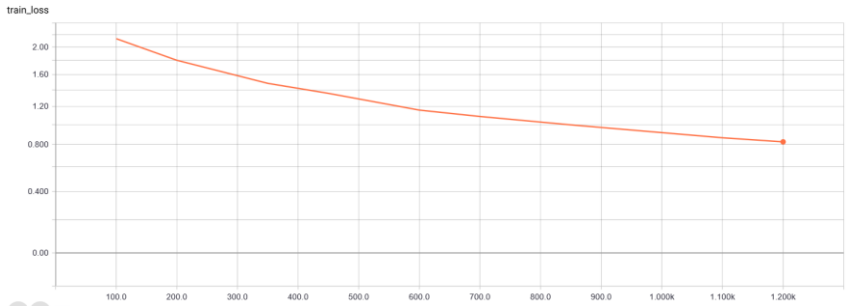
3） LR 设为 0.001

```
===> epoch: 1/5
train:
*****训练集预测准确率为: 25.326%
test:
*****测试集预测准确率为: 37.270%

===> epoch: 2/5
train:
*****训练集预测准确率为: 48.064%
test:
*****测试集预测准确率为: 56.670%

===> epoch: 3/5
train:
*****训练集预测准确率为: 59.862%
test:
*****测试集预测准确率为: 63.350%

===> epoch: 4/5
train:
*****训练集预测准确率为: 65.312%
test:
*****测试集预测准确率为: 68.760%

===> epoch: 5/5
train:
*****训练集预测准确率为: 70.320%
test:
*****测试集预测准确率为: 70.840%
*****测试集最高准确率为: 70.840%
```

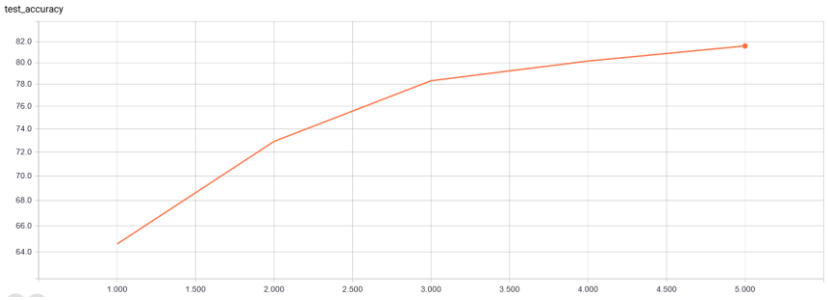
测试集准确率曲线：



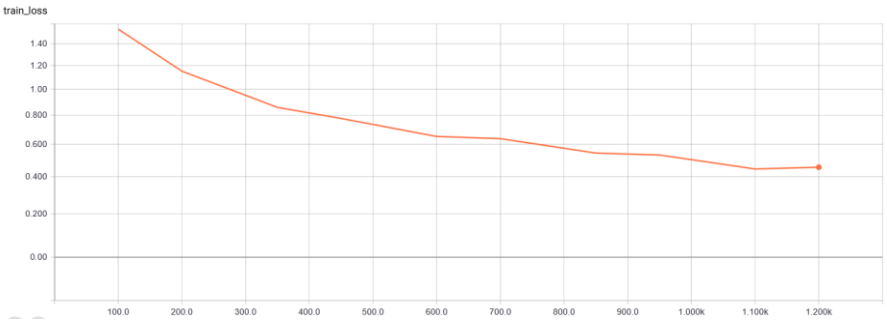
训练集 loss 曲线：



加 BN 层测试集准确率曲线：



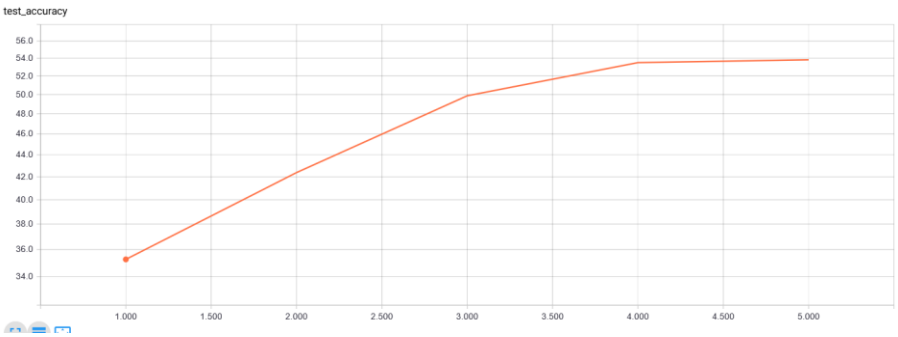
加 BN 层训练集 loss 曲线：



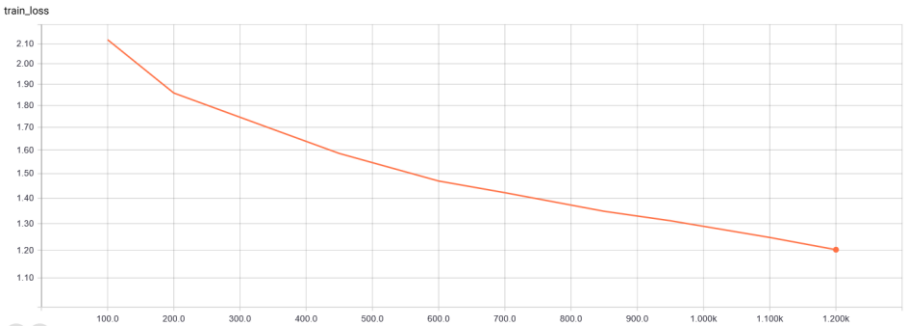
4) LR 设为 0.0001

```
===> epoch: 1/5
train:
*****训练集预测准确率为: 25.550%
test:
*****测试集预测准确率为: 35.260%

===> epoch: 2/5
train:
*****训练集预测准确率为: 38.760%
test:
*****测试集预测准确率为: 42.380%

===> epoch: 3/5
train:
*****训练集预测准确率为: 46.362%
test:
*****测试集预测准确率为: 49.870%

===> epoch: 4/5
train:
*****训练集预测准确率为: 51.176%
test:
*****测试集预测准确率为: 53.500%

===> epoch: 5/5
train:
*****训练集预测准确率为: 55.212%
test:
*****测试集预测准确率为: 53.820%
*****测试集最高准确率为: 53.820%
```

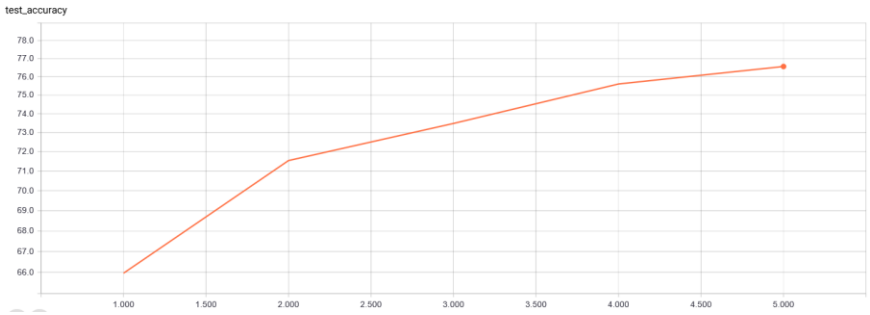
测试集准确率曲线：



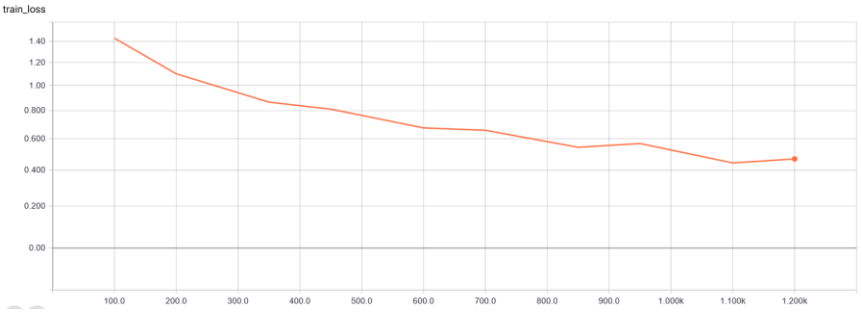
训练集 loss 曲线：



加 BN 层测试集准确率曲线：



加 BN 层训练集 loss 曲线：



五、 VGG-11 训练时选用 SGD、Adam 和 RMSprop 优化器的对比

选取学习率 lr=0.001，即 LR 对比中的最佳 LR。

由下对比可发现，当把学习率设置为 0.001 时，以 SGD 作为优化器，可能是学习率设置的过高，而导致 loss 震荡不收敛，同时准确率也一直仅有百分之十,而将学习率设置的更小时，发现还是一样的现象，可能是局部收敛出不来，所以不太懂出现这种现象的原因。当在卷积层中加入 BN 层后，每一层卷积层都进行批标准化，发现可以避免上述现象的发生。

相比较而言使用 Adam 有着较好的表现，RMSprop 在后期出现 loss 减少变缓的现象，但是最终也能够满足 5 次 epoch 之内达到 50%的准确率。

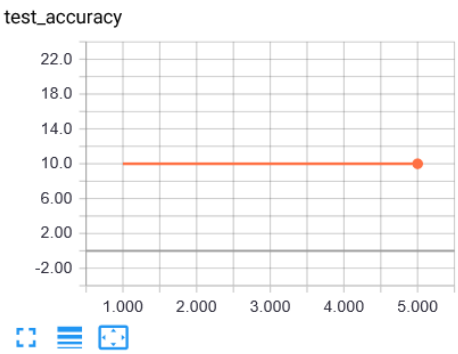
综上，选用 Adam 作为优化器时可以得到最好的效果，5 次 epoch 后（未加 BN 层），测试集的准确率可以达到 70.84%，加入 BN 层，可达到 80.6%；而选用 RMSprop 作为优化器效果居中，5 次 epoch 后，测试集准确率达到 51.19%，加入 BN 层，可达到 77.21%；其后加入 BN 层后选用 SGD 作为优化器的效果最差，5 次 epoch 后，测试集准确率可以达到 59.81%。

- SGD

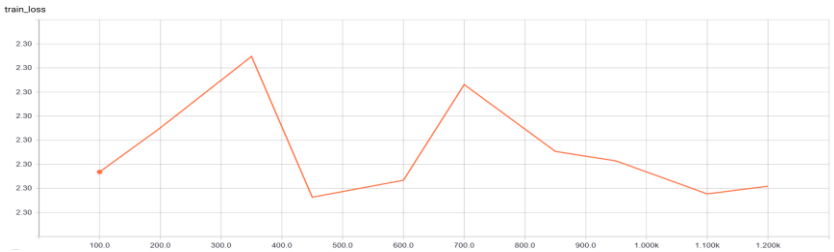
1. 未使用 BN 层进行批标准化

```
===> epoch: 1/5
train:
*****训练集预测准确率为: 10.000%
test:
*****测试集预测准确率为: 10.000%

===> epoch: 2/5
train:
*****训练集预测准确率为: 10.000%
test:
*****测试集预测准确率为: 10.000%

===> epoch: 3/5
train:
*****训练集预测准确率为: 10.000%
test:
*****测试集预测准确率为: 10.000%

===> epoch: 4/5
train:
*****训练集预测准确率为: 10.000%
test:
*****测试集预测准确率为: 10.000%

===> epoch: 5/5
train:
*****训练集预测准确率为: 10.000%
test:
*****测试集预测准确率为: 10.000%
*****测试集最高准确率为: 10.000%
```

测试集准确率曲线：



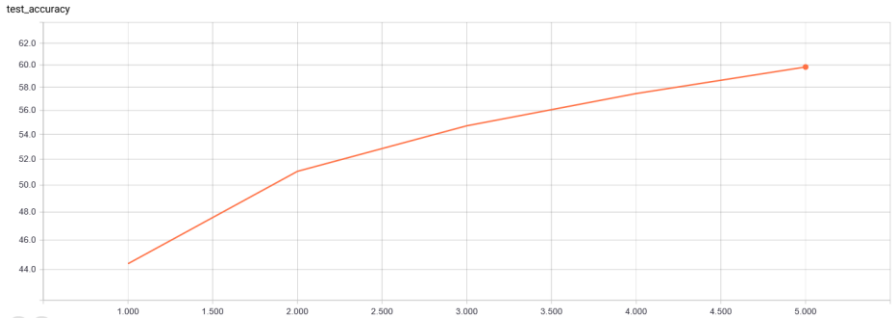
训练集 loss 曲线：



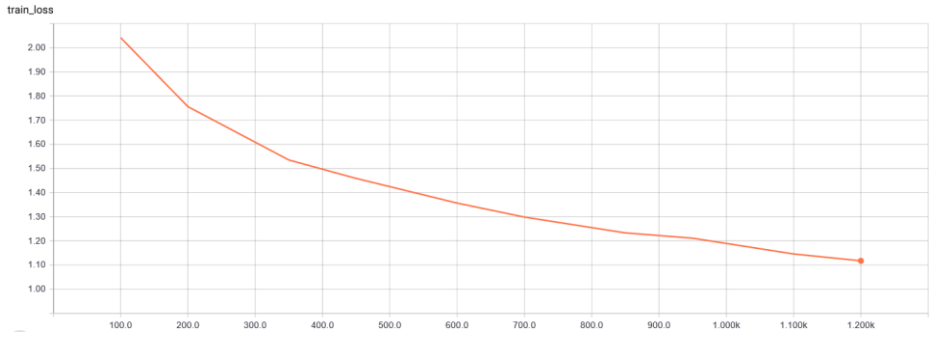
2. 使用 BN 层进行批标准化

```
===> epoch: 1/5
train:
*****训练集预测准确率为: 35.716%
test:
*****测试集预测准确率为: 44.390%

===> epoch: 2/5
train:
*****训练集预测准确率为: 48.272%
test:
*****测试集预测准确率为: 51.060%

===> epoch: 3/5
train:
*****训练集预测准确率为: 53.816%
test:
*****测试集预测准确率为: 54.710%

===> epoch: 4/5
train:
*****训练集预测准确率为: 57.606%
test:
*****测试集预测准确率为: 57.450%

===> epoch: 5/5
train:
*****训练集预测准确率为: 60.720%
test:
*****测试集预测准确率为: 59.810%
*****测试集最高准确率为: 59.810%
```

测试集准确率曲线：



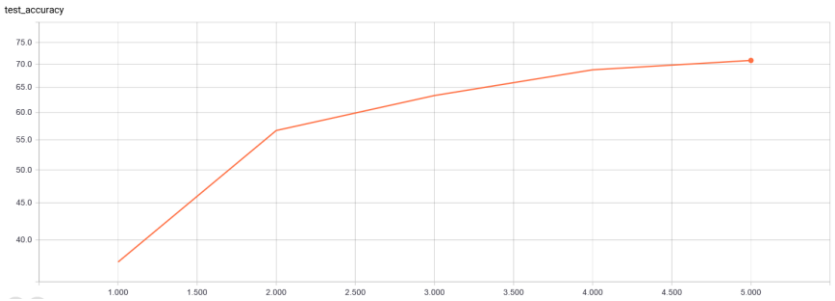
训练集 loss 曲线：



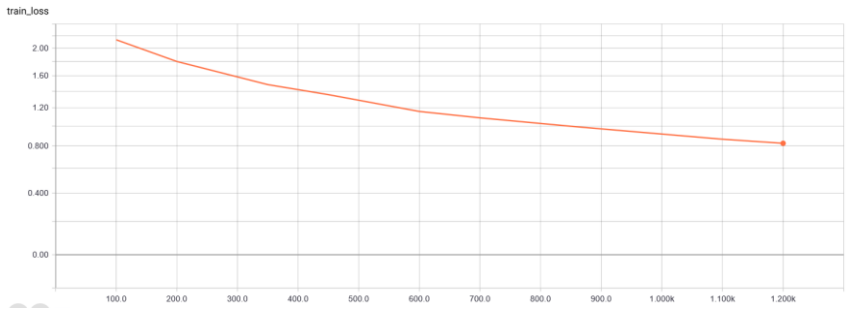
- Adam

```
===> epoch: 1/5
train:
*****训练集预测准确率为: 25.326%
test:
*****测试集预测准确率为: 37.270%

===> epoch: 2/5
train:
*****训练集预测准确率为: 48.064%
test:
*****测试集预测准确率为: 56.670%

===> epoch: 3/5
train:
*****训练集预测准确率为: 59.862%
test:
*****测试集预测准确率为: 63.350%

===> epoch: 4/5
train:
*****训练集预测准确率为: 65.312%
test:
*****测试集预测准确率为: 68.760%

===> epoch: 5/5
train:
*****训练集预测准确率为: 70.320%
test:
*****测试集预测准确率为: 70.840%
*****测试集最高准确率为: 70.840%
```

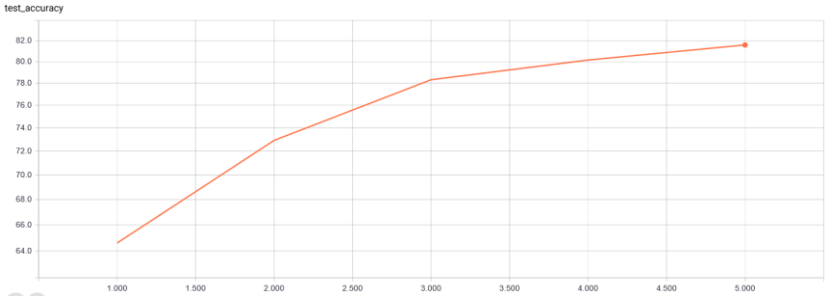
测试集准确率曲线：



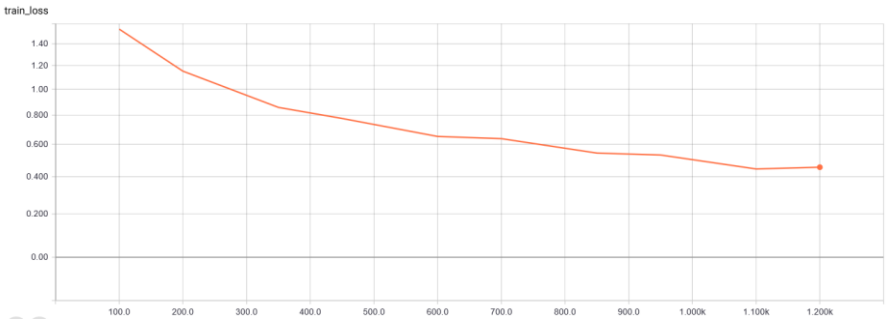
训练集 loss 曲线：



加 BN 层测试集准确率曲线：



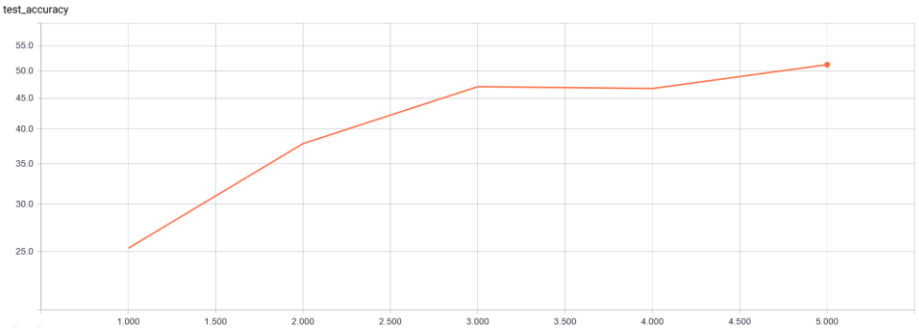
加 BN 层训练集 loss 曲线：



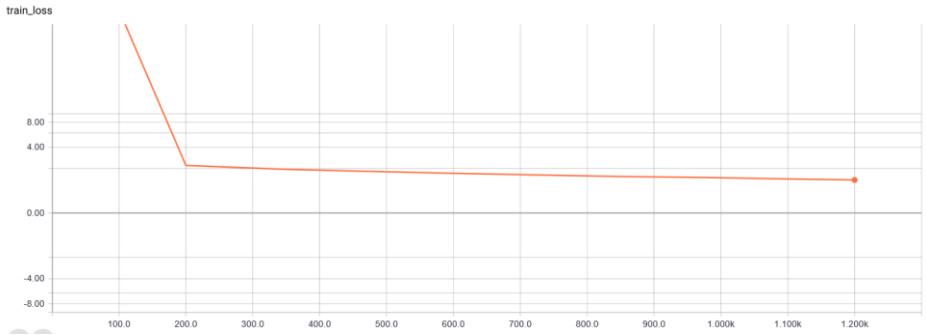
- RMSprop

```
===> epoch: 1/5
train:
*****训练集预测准确率为: 14.562%
test:
*****测试集预测准确率为: 25.340%

===> epoch: 2/5
train:
*****训练集预测准确率为: 31.214%
test:
*****测试集预测准确率为: 37.830%

===> epoch: 3/5
train:
*****训练集预测准确率为: 40.388%
test:
*****测试集预测准确率为: 47.050%

===> epoch: 4/5
train:
*****训练集预测准确率为: 46.986%
test:
*****测试集预测准确率为: 46.720%

===> epoch: 5/5
train:
*****训练集预测准确率为: 53.224%
test:
*****测试集预测准确率为: 51.190%
*****测试集最高准确率为: 51.190%
```

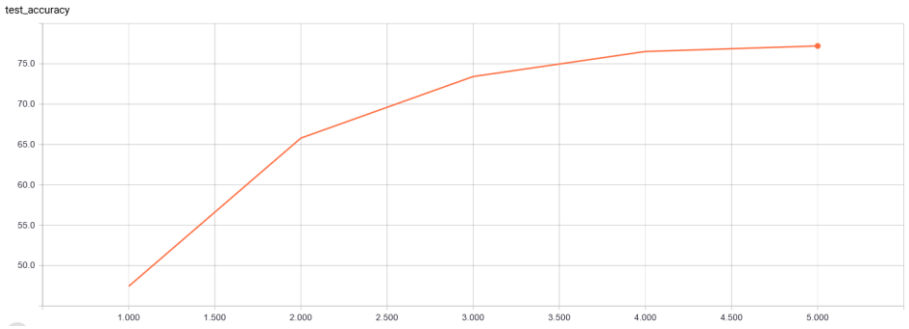
测试集准确率曲线：



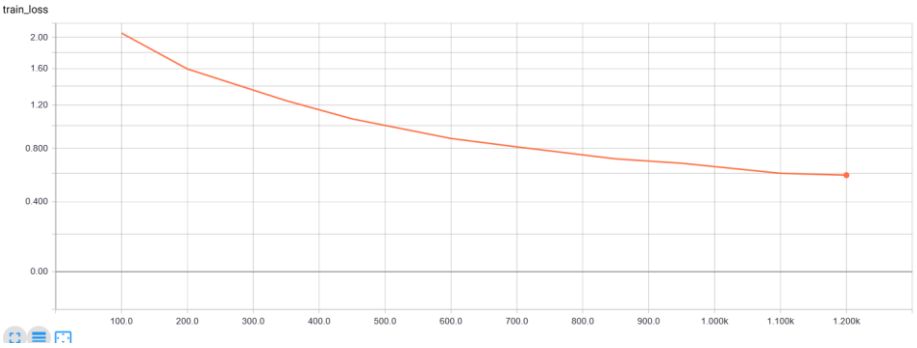
训练集 loss 曲线：



加 BN 层测试集准确率曲线：



加 BN 层训练集 loss 曲线：



六、 ResNet 的网络结构

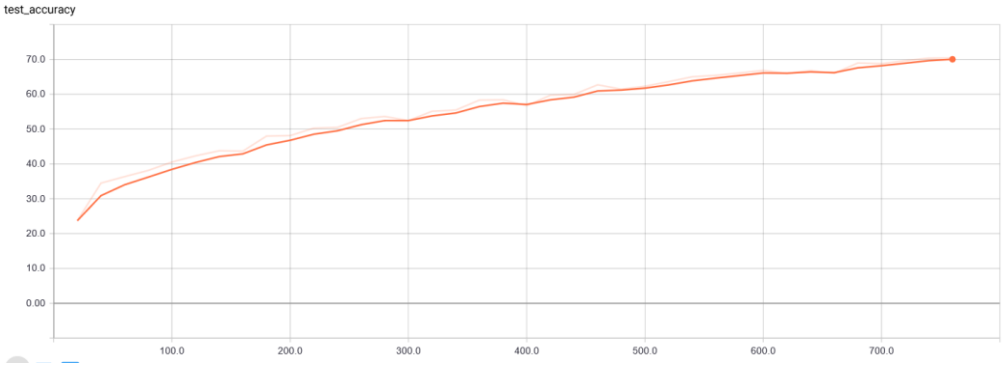
ResNet18 的结构如下：

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
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

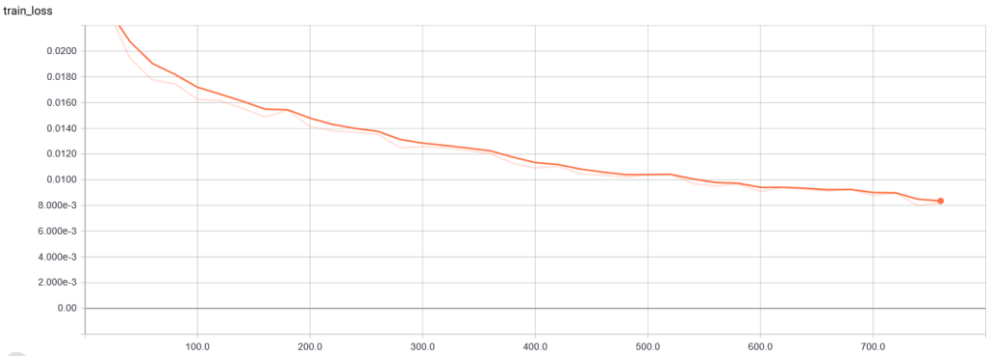
网络中有很多重复的单元，它们都有跨层直连的 shortcut。ResNet 中，将一个跨层直连的单元称为 Residual block。 ResNet18 中，每两个 Residual block 组成一个 layer。除第一层卷积外，有 4 个 layer。

七、 ResNet 的测试集准确率曲线和 loss 曲线

测试集准确率曲线：



训练集 loss 曲线：



八、 成员分工

1160300122-谢根琳：实现 VGG-11、VGG-11 的对比实验、使用 argparse 设定 CPU/GPU

1160300426-李国建：实现 ResNet-18、手动实现 dataset、论文搜索

各自完成自己所写部分的报告。谢根琳对代码和报告进行整合。