AlexNet 应用于 cifar 数据集实验报告

15331416 赵寒旭

目录

1. 修改的 AlexNet 结构

1）AlexNet 详细结构

Modified AlexNet for Cifar(3FC)					
input	32*32*3				
layer1	Conv2d	kernel	channel	padding	stride
		11*11*3	64	5	4
	Relu	inplace=True			
	MaxPool2d	kernel_size		stride	
		2*2		2	
layer2	Conv2d	kernel	channel	padding	stride
		5*5*64	192	2	default
	Relu	inplace=True			
	MaxPool2d	kernel_size		stride	
		2*2		2	
layer3	Conv2d	kernel	channel	padding	stride
		3*3*192	384	1	default
	Relu	inplace=True			
layer4	Conv2d	kernel	channel	padding	stride
		3*3*384	256	1	default
	Relu	inplace=True			
layer5	Conv2d	kernel	channel	padding	stride
		3*3*256	256	1	default
	Relu	inplace=True			
	MaxPool2d	kernel_size		stride	
		2*2		2	
fully-connected	Dropout				
	Linear	256->4096			
	Relu	inplace=True			
	Dropout				
	Linear	4096->4096			
	Relu	inplace=True			
	Linear	4096->10			

2）参数初始化

learning_rate = 0.1

momentum = 0.9

weight_decay = 0.0005

损失函数使用交叉熵，训练过程使用带动量的随机梯度下降法。

```
230  # 交叉熵损失函数
231  criterion = nn.CrossEntropyLoss()
232  # 随机梯度下降
233  optimizer = optim.SGD(model.parameters(), lr=0.1, momentum=0.9, weight_decay=5e-4)
```

3）测试及网络参数调整

（1）增强数据集

```
####数据预处理
transform = transforms.Compose([
                                transforms.RandomCrop(32, padding=4), # 随机剪裁
                                transforms.RandomHorizontalFlip(), # 随机水平翻转
                                transforms.ToTensor(),#转为tensor
                                transforms.Normalize((0.5,0.5,0.5),(0.5,0.5,0.5)),#归一化
                            ])
```

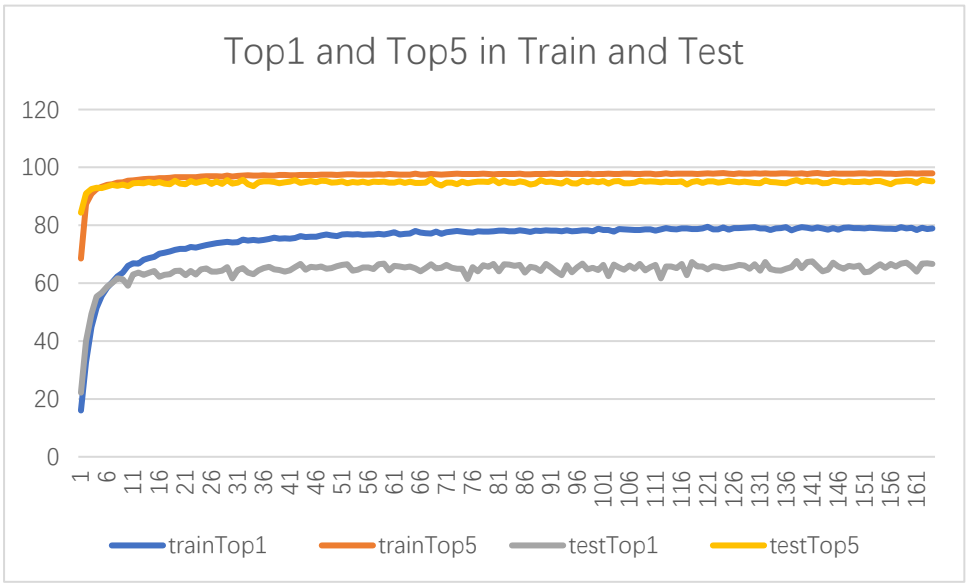
（2）修改学习率

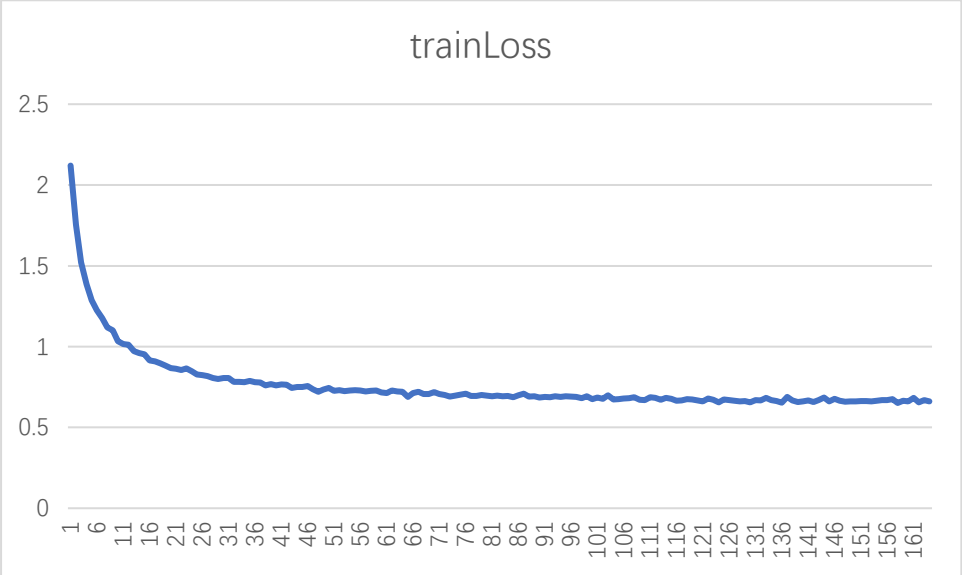
```
def adjust_learning_rate(optimizer, epoch):
    if epoch in [81, 122]:
        for param_group in optimizer.param_groups:
            param_group['lr'] = param_group['lr'] * 0.1
```

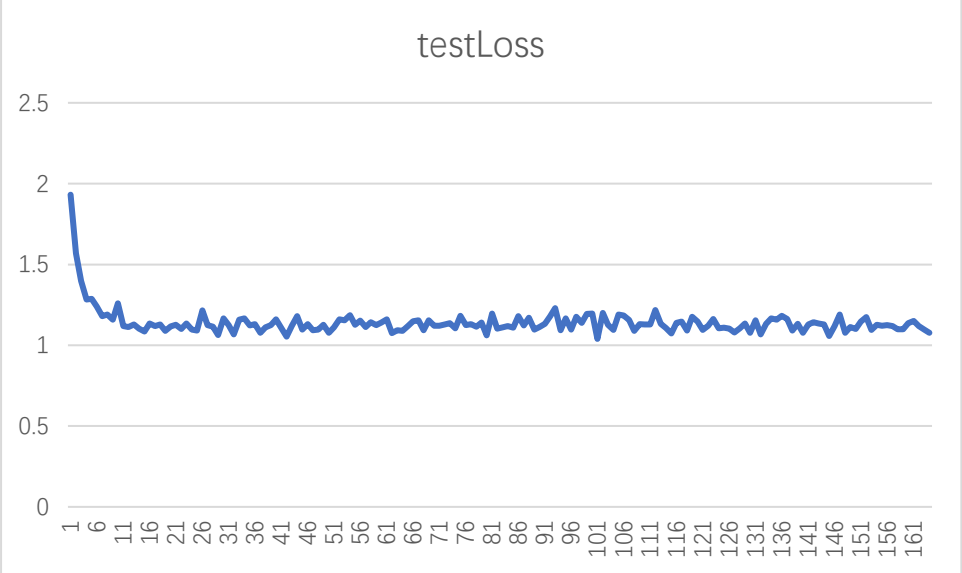
4）结果对比展示

（1）无数据增强，无学习率调整

Best Accuracy		
train	Top-1	79.48
	Top-5	98.044
test	Top-1	67.65
	Top-5	96.01

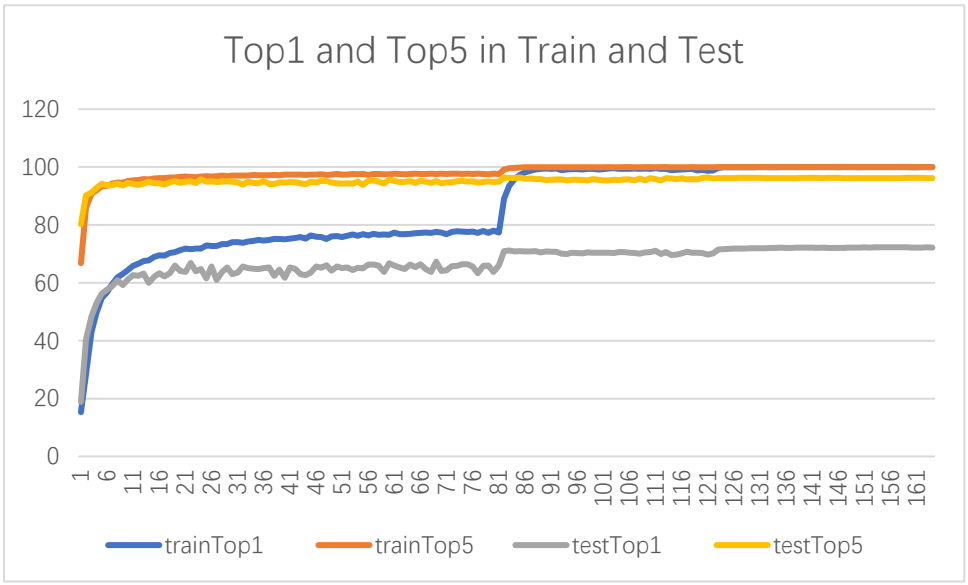


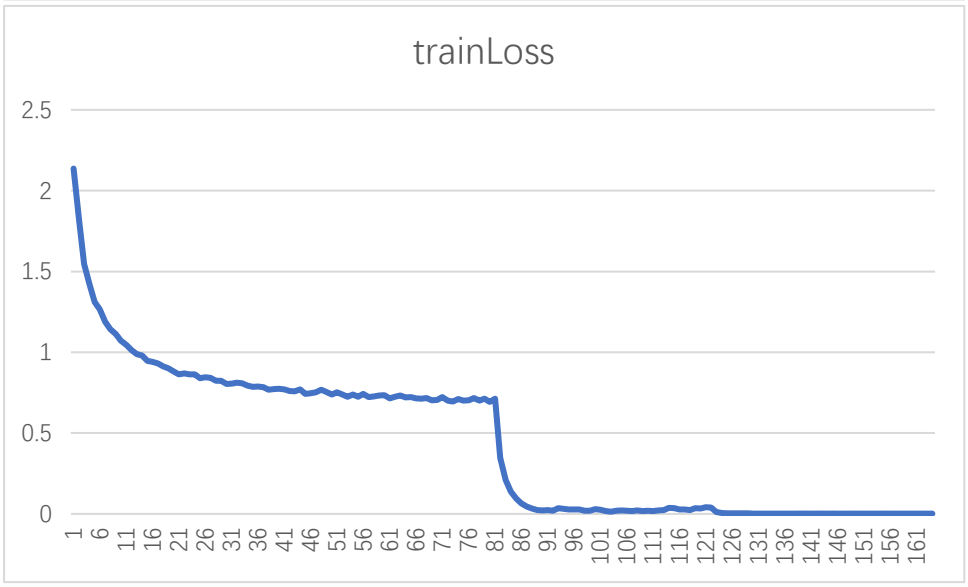


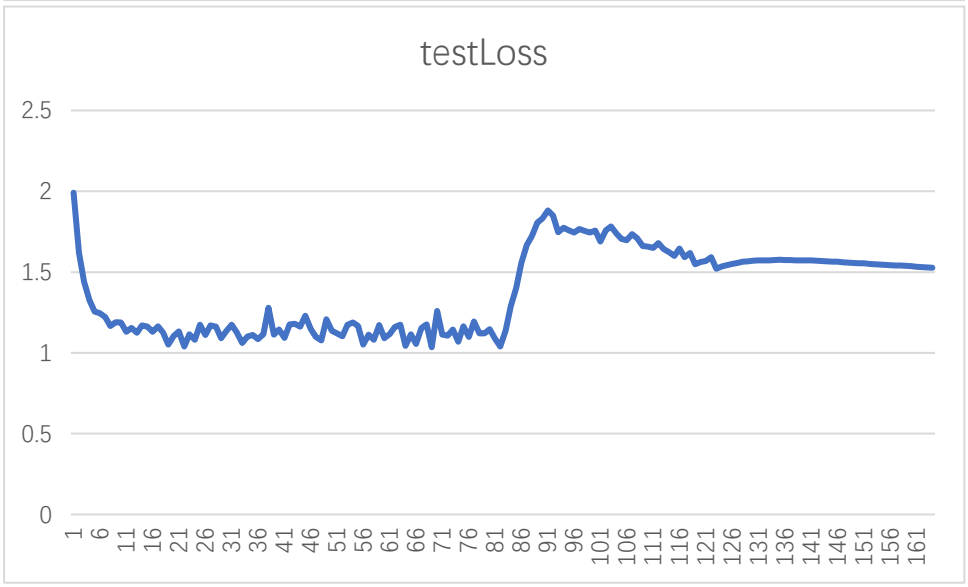


（2）无数据增强，有学习率调整

Best Accuracy		
train	Top-1	100
	Top-5	100
test	Top-1	72.17
	Top-5	96.46

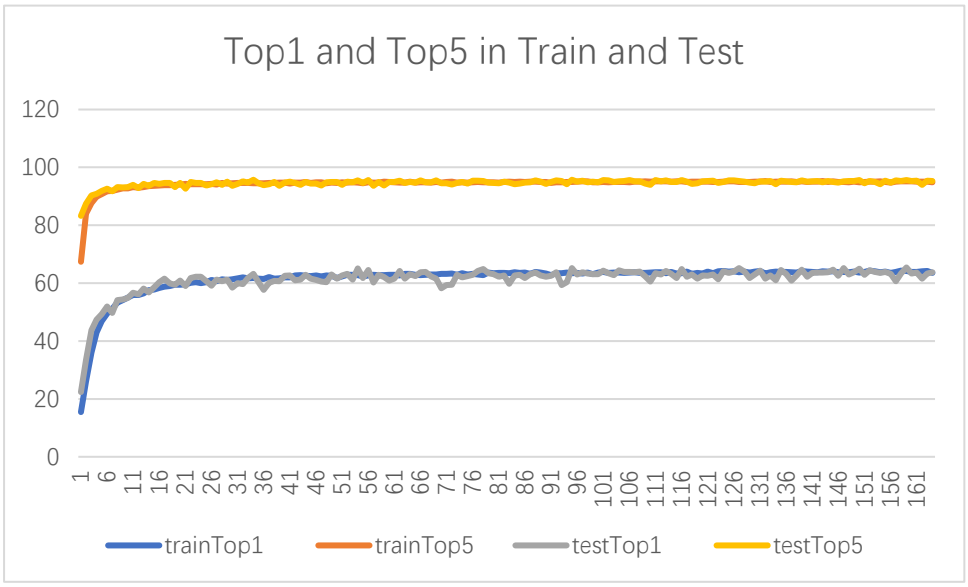

Top1 and Top5 in Train and Test
120
100
80
60
40
20
0
1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101 106 111 116 121 126 131 136 141 146 151 156 161
trainTop1
trainTop5
testTop1
testTop5

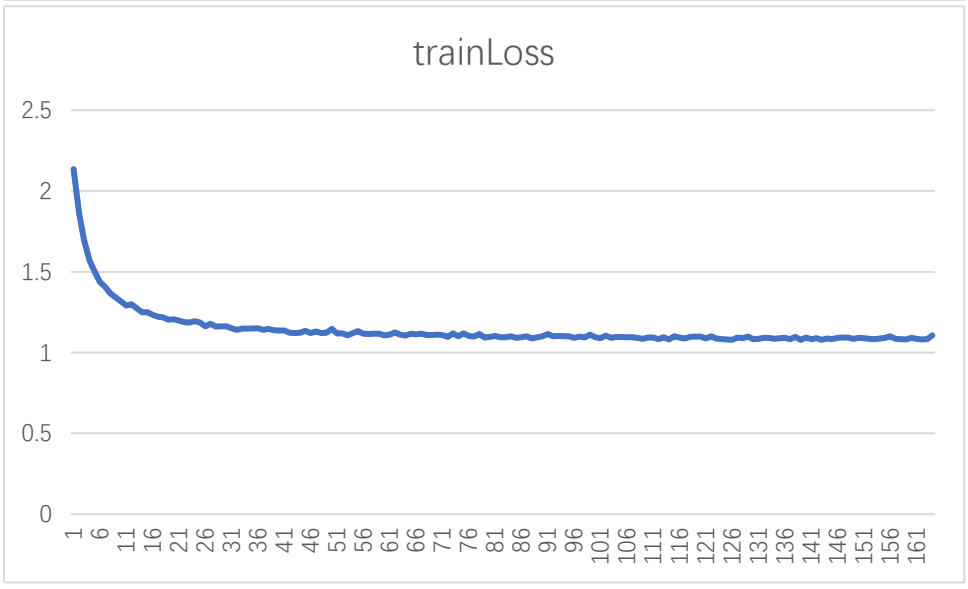

trainLoss
2.5
2
1.5
1
0.5
0
1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101 106 111 116 121 126 131 136 141 146 151 156 161

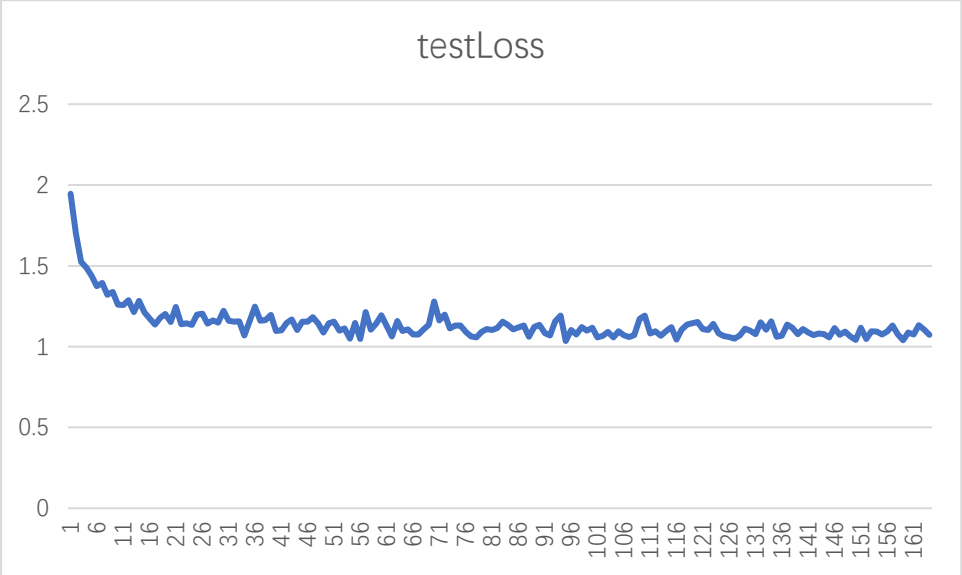

testLoss
2.5
2
1.5
1
0.5
0
1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101 106 111 116 121 126 131 136 141 146 151 156 161

（3）有数据增强，无学习率调整

Best Accuracy		
train	Top-1	64.344
	Top-5	95.28
test	Top-1	65.25
	Top-5	95.73

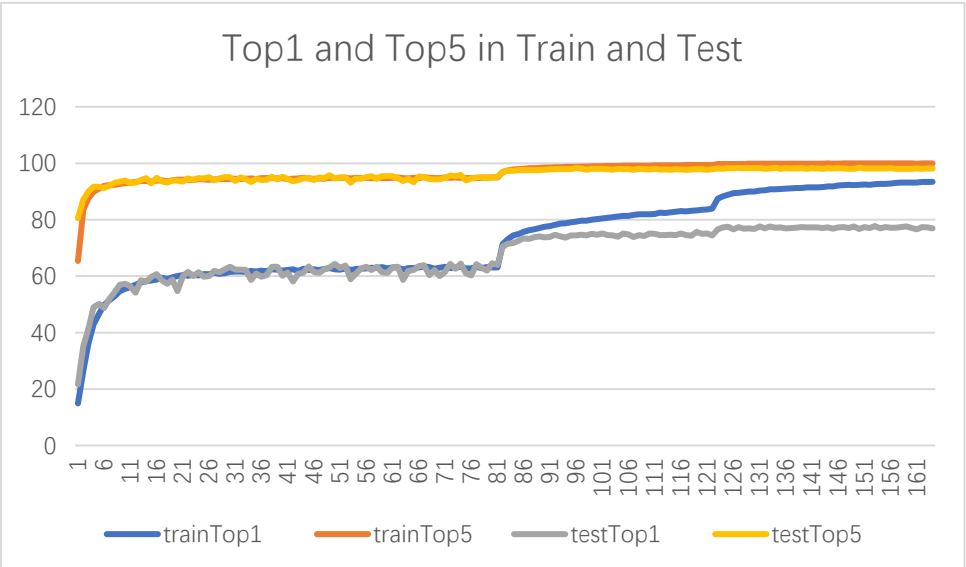


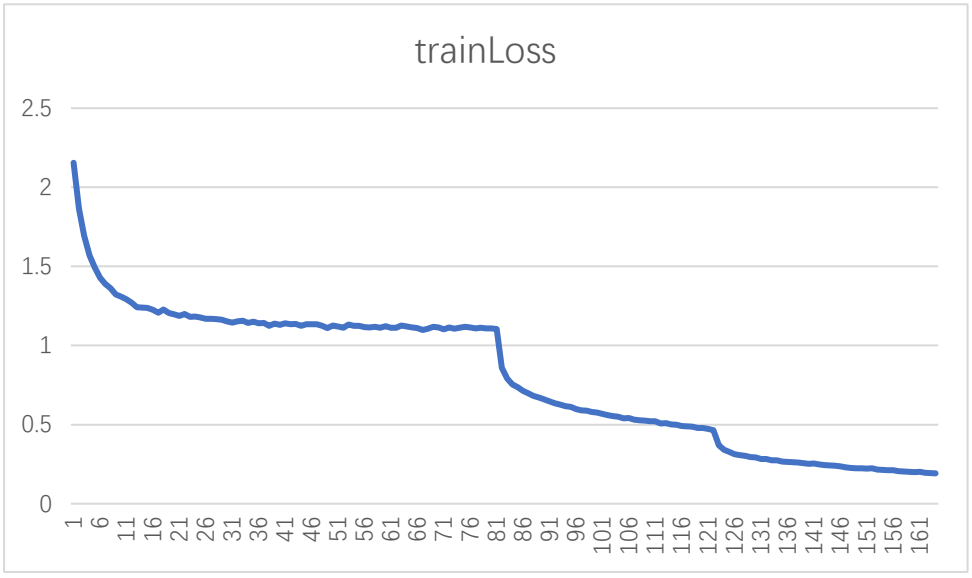


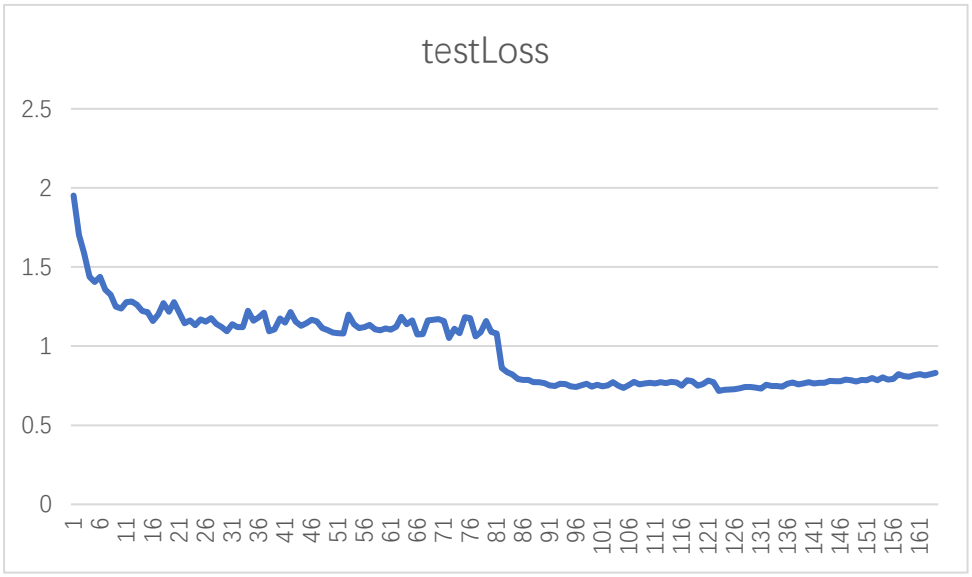


（4）有数据增强，有学习率调整

Best Accuracy		
train	Top-1	92.35
	Top-5	99.904
test	Top-1	77.61
	Top-5	98.43




trainLoss
2.5
2
1.5
1
0.5
0
1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101 106 111 116 121 126 131 136 141 146 151 156 161


testLoss
2.5
2
1.5
1
0.5
0
1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101 106 111 116 121 126 131 136 141 146 151 156 161

2. 全连接层结构调整

1）AlexNet 详细结构

Modified AlexNet for Cifar(1FC)					
input	32*32*3				
layer1	Conv2d	kernel	channel	padding	stride
		11*11*3	64	5	4
	Relu	inplace=True			
	MaxPool2d	kernel_size		stride	
		2*2		2	
layer2	Conv2d	kernel	channel	padding	stride
		5*5*64	192	2	default
	Relu	inplace=True			
	MaxPool2d	kernel_size		stride	
		2*2		2	
layer3	Conv2d	kernel	channel	padding	stride
		3*3*192	384	1	default
	Relu	inplace=True			
layer4	Conv2d	kernel	channel	padding	stride
		3*3*384	256	1	default
	Relu	inplace=True			
layer5	Conv2d	kernel	channel	padding	stride
		3*3*256	256	1	default
	Relu	inplace=True			
	MaxPool2d	kernel_size		stride	
		2*2		2	
fully-connected	256 -> 10				

2）参数初始化

learning_rate = 0.1

momentum = 0.9

weight_decay = 0.0005

损失函数使用交叉熵，训练过程使用带动量的随机梯度下降法。

```
# 交叉熵损失函数
criterion = nn.CrossEntropyLoss()
# 随机梯度下降
optimizer = optim.SGD(model.parameters(), lr=0.1, momentum=0.9, weight_decay=5e-4)
```

3）测试及网络参数调整

（1）增强数据集

```
####数据预处理
transform = transforms.Compose([
                transforms.RandomCrop(32, padding=4), # 随机剪裁
                transforms.RandomHorizontalFlip(), # 随机水平翻转
                transforms.ToTensor(),#转为tensor
                transforms.Normalize((0.5,0.5,0.5),(0.5,0.5,0.5)),#归一化
            ])
```

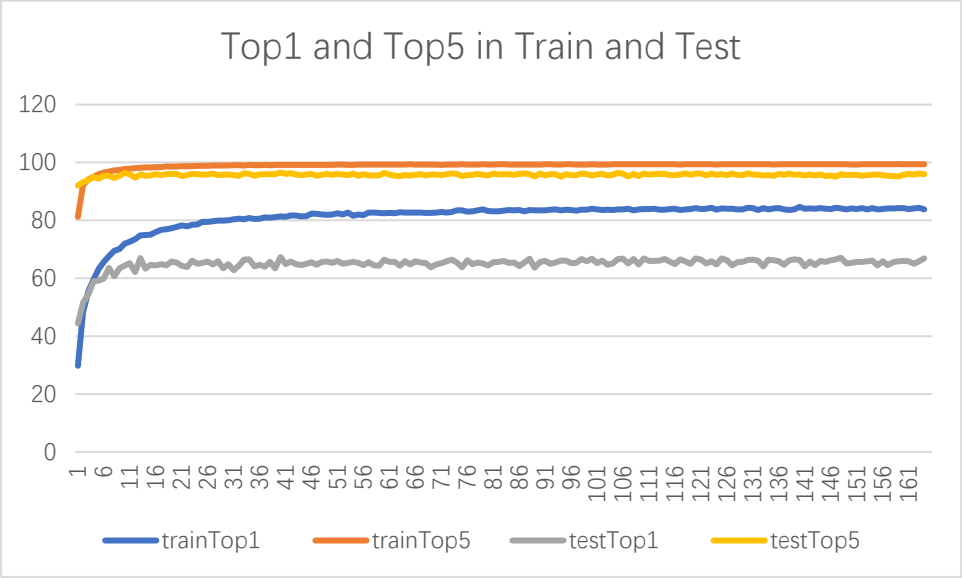
（2）修改学习率

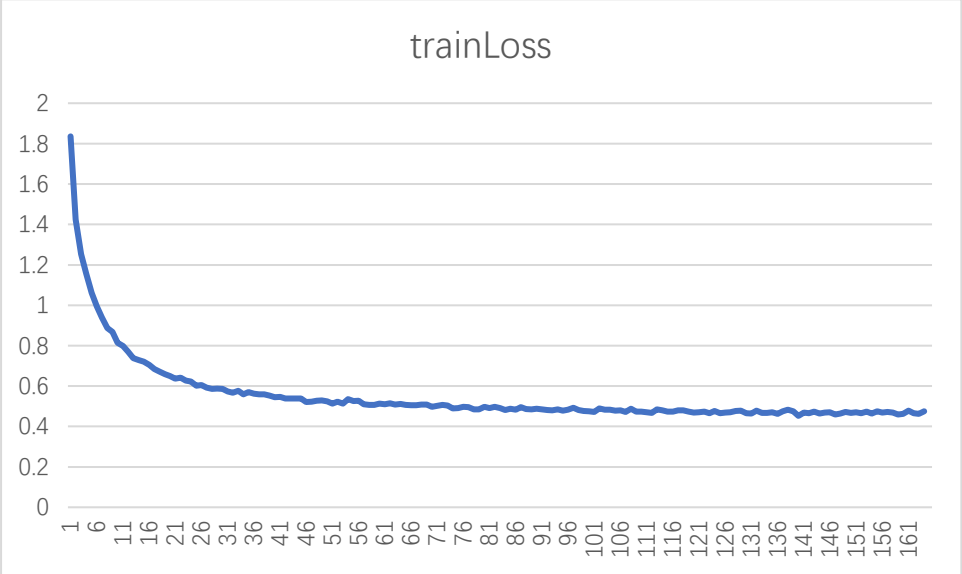
```
def adjust_learning_rate(optimizer, epoch):
    if epoch in [81, 122]:
        for param_group in optimizer.param_groups:
            param_group['lr'] = param_group['lr'] * 0.1
```

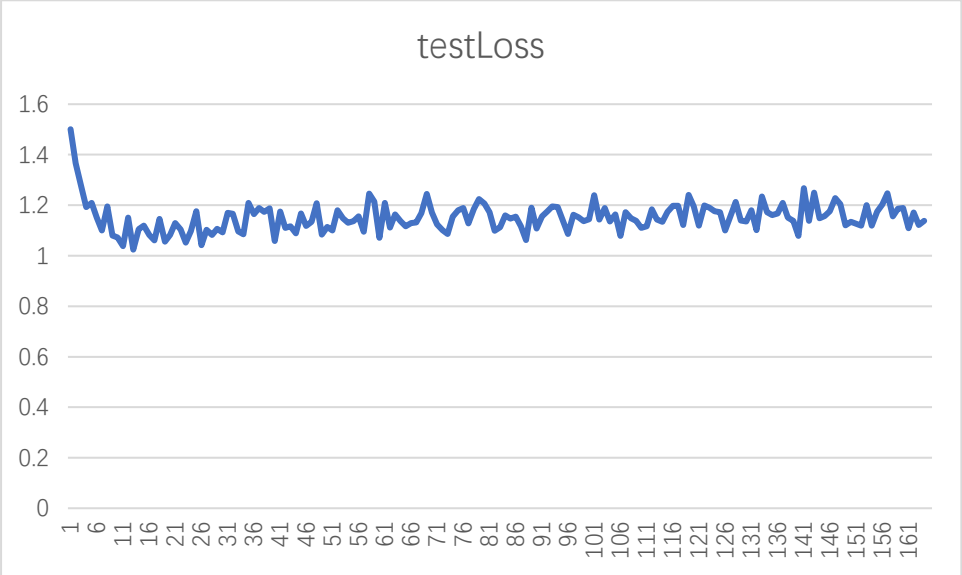
4）结果对比展示

（1）无数据增强，无学习率调整

Best Accuracy		
train	Top-1	84.674
	Top-5	99.444
test	Top-1	67.31
	Top-5	96.51

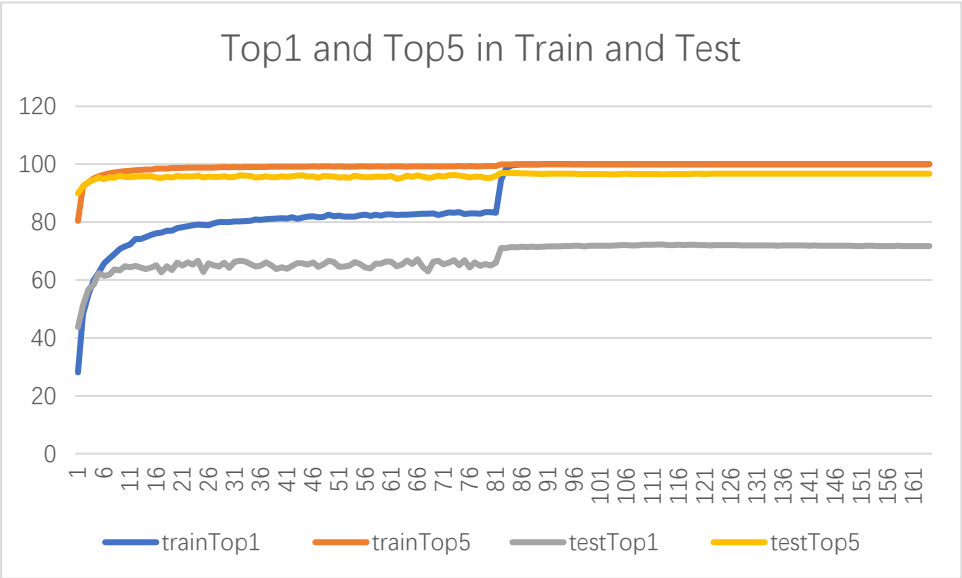


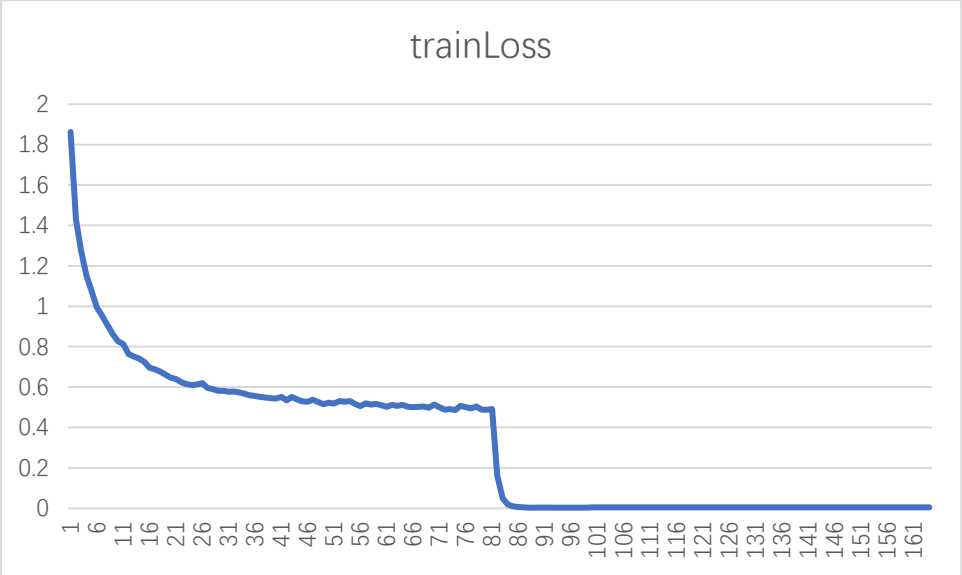


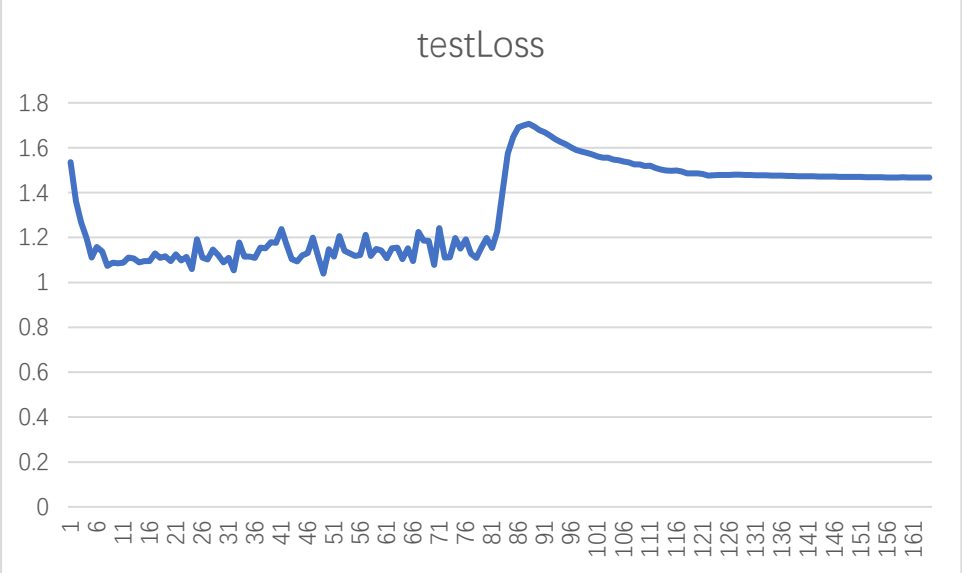


（2）无数据增强，有学习率调整

Best Accuracy		
train	Top-1	100
	Top-5	100
test	Top-1	72.26
	Top-5	96.96

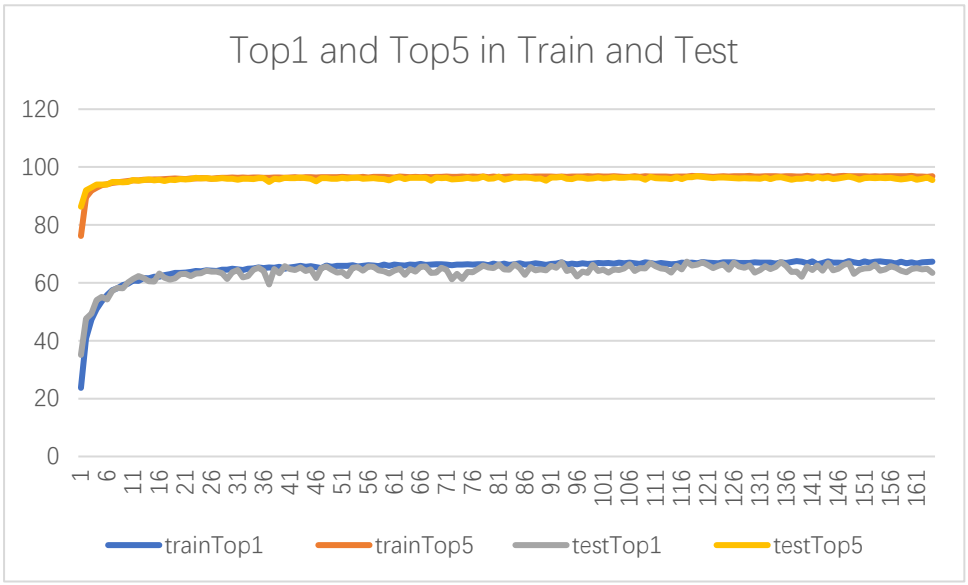


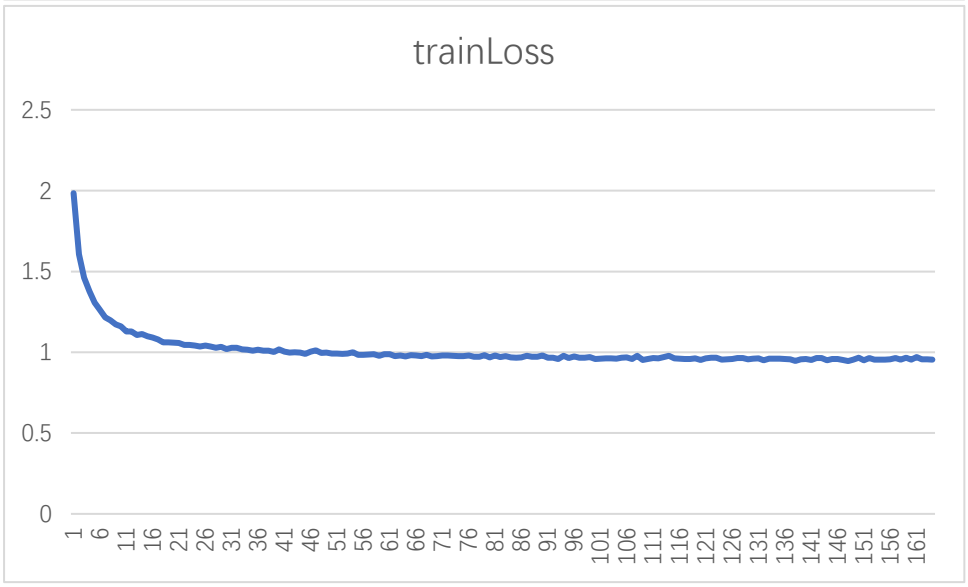


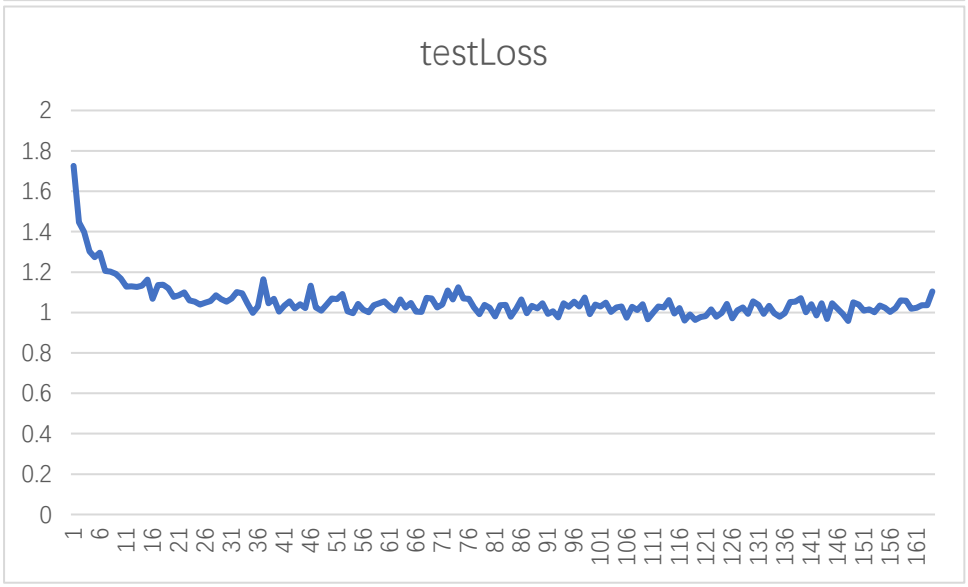


（3）有数据增强，无学习率调整

Best Accuracy		
train	Top-1	67.524
	Top-5	96.974
test	Top-1	67.33
	Top-5	96.9

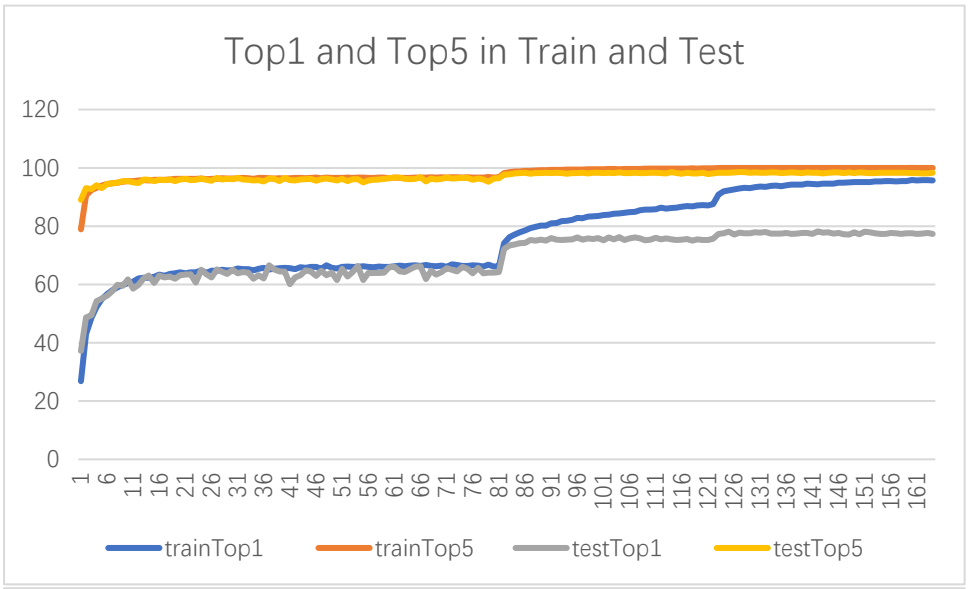

Top1 and Top5 in Train and Test
120
100
80
60
40
20
0
1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101 106 111 116 121 126 131 136 141 146 151 156 161
trainTop1
trainTop5
testTop1
testTop5

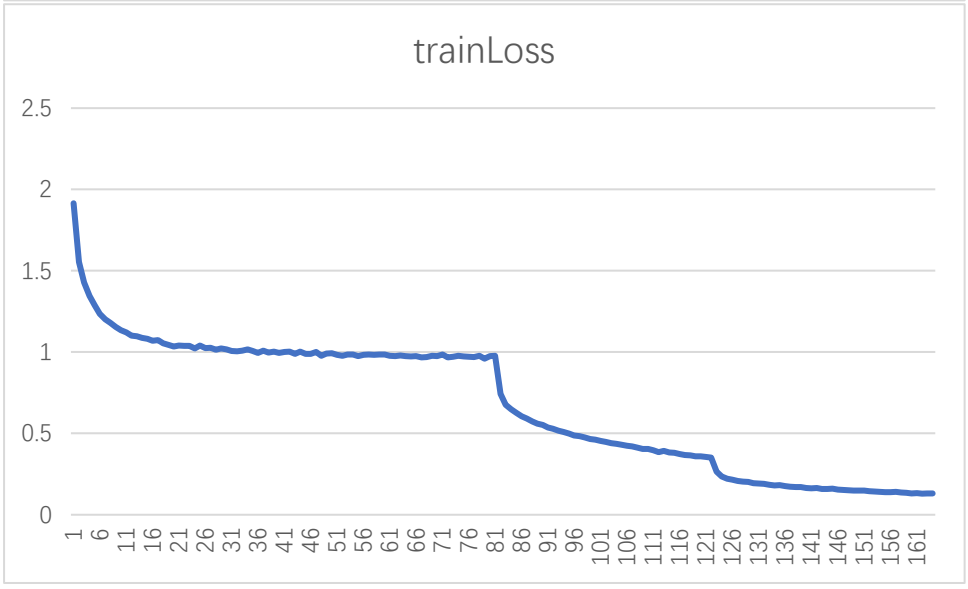

trainLoss
2.5
2
1.5
1
0.5
0
1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101 106 111 116 121 126 131 136 141 146 151 156 161

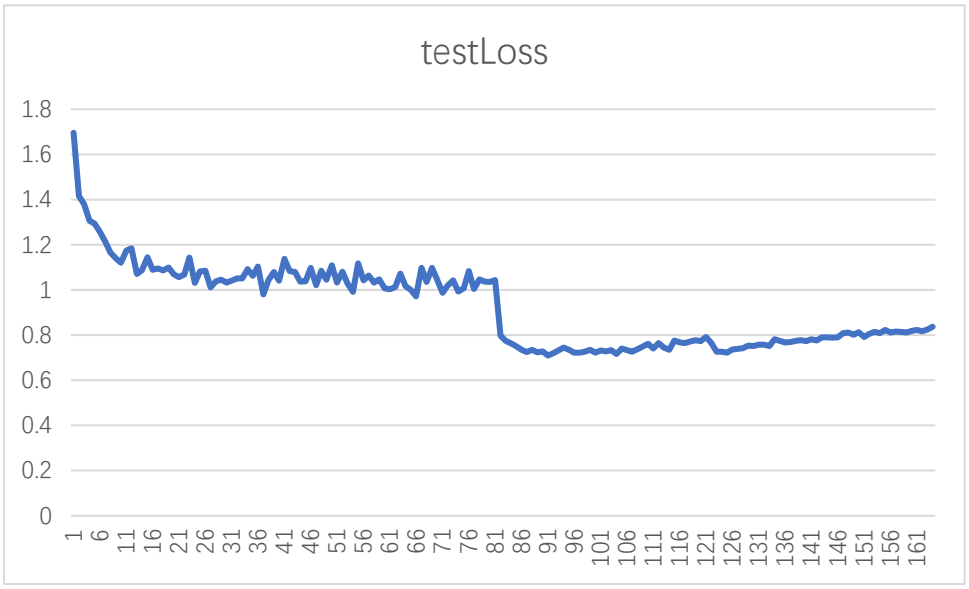

testLoss
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101 106 111 116 121 126 131 136 141 146 151 156 161

（4）有数据增强，有学习率调整

Best Accuracy		
train	Top-1	95.164
	Top-5	99.984
test	Top-1	78.2
	Top-5	98.56






testLoss
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1
6
11
16
21
26
31
36
41
46
51
56
61
66
71
76
81
86
91
96
101
106
111
116
121
126
131
136
141
146
151
156
161