# 《Python 与深度学习》第二次实验报告

少年班学院 PB19000202 张炜琛

2022.5

## 1. 实验环境

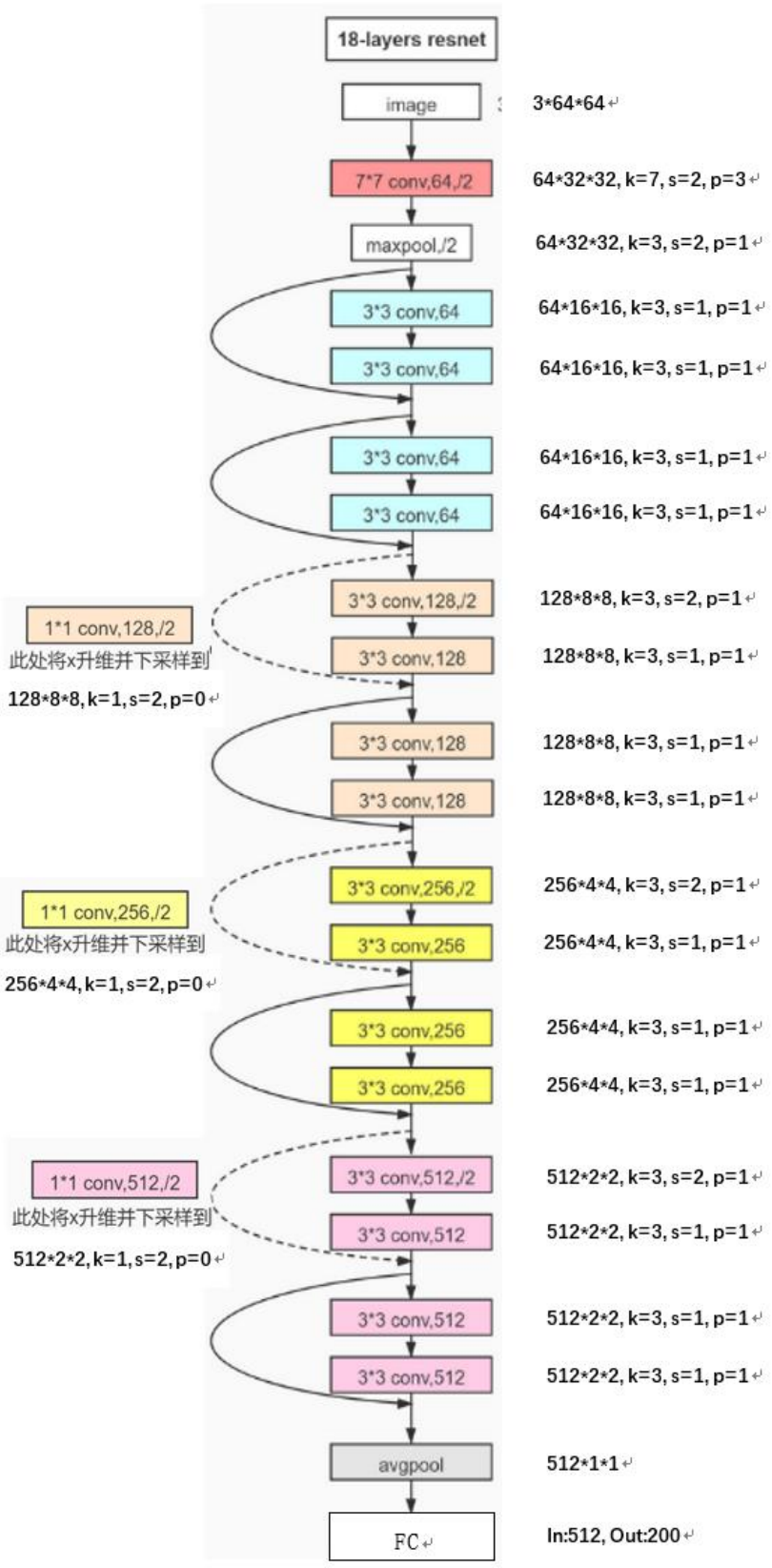
Windows10, python 3,9,7(‘base’: conda 4.10.3), Visual Studio Code 1.66, argparse 1.1, torch 1.11.0+cu113, torchvision 0.12.0+cu113, tensorboard 2.9.0.

## 2. 实验任务

将适用于 Imagenet 的 pytorch 图片分类训练范例代码，修改为适用 Tiny-Imagnet，并进行评估，模型限定为 resnet18.

## 3. 网络结构

本实验使用 Resnet18 模型，经过简单修改，适用于 Tiny-Imagenet-200，对于输入 3*64*64 的图片，经过计算，其网络结构和各层输出如下图 Fig.1.




注：本网络参考图片输入大小 3*224*224 的 resnet18 网络绘制，原网址：https://blog.csdn.net/weixin_36979214/article/details/108879684

Fig.1 Resnet18 网络结构和各层输出结果

## 4. 代码改动

### 4.1 Output 修改

```
135 137     if args.pretrained:
136 138         print("=> using pre-trained model '{}'".format(args.arch))
137 139         model = models.__dict__[args.arch](pretrained=True)
    140 +       # Finetune Final few layers to adjust for tiny imagenet input
    141 +       model.avgpool = nn.AdaptiveAvgPool2d(1)
    142 +       num_ftrs = model.fc.in_features
    143 +       model.fc = nn.Linear(num_ftrs, 200)
138 144     else:
139 145         print("=> creating model '{}'".format(args.arch))
140 146         model = models.__dict__[args.arch]()
    147 +       # Finetune Final few layers to adjust for tiny imagenet input
    148 +       model.avgpool = nn.AdaptiveAvgPool2d(1)
    149 +       num_ftrs = model.fc.in_features
    150 +       model.fc = nn.Linear(num_ftrs, 200)
```

修改 resnet18 的 FC 层，将输出改为 200 维。

### 4.2 验证数据集的改写（val_reformat.py）

```
8  +target_folder = 'C:\\Users\\86187\\Documents\\GitHub\\Weiczh-s-1st\\hw2tinyimage\\tiny-imagenet-200\\val\\'
9  +val_dict = {}
10 +with open(target_folder + 'val_annotations.txt', 'r') as f:
11 +    for line in f.readlines():
12 +        split_line = line.split('\t')
13 +        val_dict[split_line[0]] = split_line[1]
14 +paths = glob.glob(target_folder + 'images/*')
15 +paths[0].split('\\')[-1]
16 +for path in paths:
17 +    file = path.split('\\')[-1]
18 +    folder = val_dict[file]
19 +    if not os.path.exists(target_folder + str(folder)):
20 +        os.mkdir(target_folder + str(folder))
21 +for path in paths:
22 +    file = path.split('\\')[-1]
23 +    folder = val_dict[file]
24 +    dest = target_folder + str(folder) + '\\' + str(file)
25 +    move(path, dest)
26 +os.remove(os.path.join(target_folder, 'val_annotations.txt'))
27 +rmdir(os.path.join(target_folder, 'images'))
```

Tiny-Imagenet-200 的训练集图片是以其图片文件夹名称为标签分类，而验证集的图片全部存放在同一文件夹内。为了将验证集的图像标签对应到训练集上，我们利用 val_annotations.txt 对验证集文件夹进行改写，使其具有与训练集文件夹相同的形式，如下图所示：


4.3 增加 TesnorBoard 代码

```
23 +from torch.utils.tensorboard import SummaryWriter
```

Import 相关库函数。

```
83 +writer = SummaryWriter()

296 +    writer.close()

278 299 def train(train_loader, model, criterion, optimizer, epoch, args):

344 +    writer.add_scalar('Loss_train', losses.avg, epoch)
345 +    writer.add_scalar('Acc5_train', top5.avg, epoch)

325 -def validate(val_loader, model, criterion, args):
348 +def validate(val_loader, model, criterion, epoch, args):

386 +    writer.add_scalar('Loss_val', losses.avg, epoch)
387 +    writer.add_scalar('Acc5_val', top5.avg, epoch)
365 388     return top1.avg
```

在 train()函数和 validate()函数定义结尾，加入代码分别记录其每一个 epoch 的 loss 和 acc5 数据。因为 writer.add_scalar()需要计数 epoch 作为变量，所以在 validate 定义中加入 epoch 作为变量。

```
258 -        acc1 = validate(val_loader, model, criterion, args)
269 +        acc1 = validate(val_loader, model, criterion, epoch, args)
```

因此，前面引用 validate()也需做简单改写。

4.4 图片剪裁变换

```
216 226     train_dataset = datasets.ImageFolder(
217 227         traindir,
218 228         transforms.Compose([
219     -           transforms.RandomResizedCrop(224),
220 229           transforms.RandomHorizontalFlip(),
    230 +         # transforms.RandomVerticalFlip(),
    231 +         # transforms.RandomRotation(10),
221 232           transforms.ToTensor(),
222 233           normalize,
223     -       ]))
    234 +       ])
    235 +   )

236 248     val_loader = torch.utils.data.DataLoader(
237     -       datasets.ImageFolder(valdir, transforms.Compose([
238     -           transforms.Resize(256),
239     -           transforms.CenterCrop(224),
    249 +       datasets.ImageFolder(valdir,
    250 +                            transforms.Compose([
240 251                                transforms.ToTensor(),
241 252                                normalize,
242 253                            ])),
```

Tiny-Imagenet-200 的图片已经事先做过拉伸和裁剪，统一了尺寸，故删除图片裁剪部分的代码。基于对提高训练性能和验证准确度的期望，通过图片的翻转、旋转做了一些简单的数据集增强。

4.5 其他核心代码改动

```
37      -parser.add_argument('--epochs', default=90, type=int, metavar='N',
    40  +parser.add_argument('--epochs', default=15, type=int, metavar='N',
```

调整 epoch 数量，15 个 epoch 足以使数据集 top5 准确度训练到 95%以上

```
53      -parser.add_argument('-p', '--print-freq', default=10, type=int,
54      -                    metavar='N', help='print frequency (default: 10)')
55      -parser.add_argument('--resume', default='', type=str, metavar='PATH',
    56  +parser.add_argument('-p', '--print-freq', default=20, type=int,
    57  +                    metavar='N', help='print frequency (default: 20)')
    58  +parser.add_argument('--resume', default='C:/Users/86187/checkpoint10.pth.tar', type=str, metavar='PATH',
```

将 print-freq 调整为 20，使命令行窗口的打印结果更加简洁。写入 checkpoint 的位置。

```
    81  +#model = models.resnet18(pretrained=True)
```

探索发现，如果使用 resnet18 预训练模型，学习率取 0.01，可以在 5 个 epoch 的训练内将验证集的 top5 准确度提升并稳定于 77%上下。

```
-    scheduler = StepLR(optimizer, step_size=30, gamma=0.1)
+    scheduler = StepLR(optimizer, step_size=10, gamma=0.1)
```

将 step_size 修改为 10，即没训练 10 个 epoch，学习率降至原先的 0.1 倍。

```
211     -   traindir = os.path.join(args.data, 'train')
212     -   valdir = os.path.join(args.data, 'val')
        220 +   dir = 'C:\\Users\\86187\\Documents\\GitHub\\Weiczh-s-1st\\hw2tinyimage'
        221 +   traindir = os.path.join(dir, args.data, 'train')
        222 +   valdir = os.path.join(dir, args.data, 'val')
```

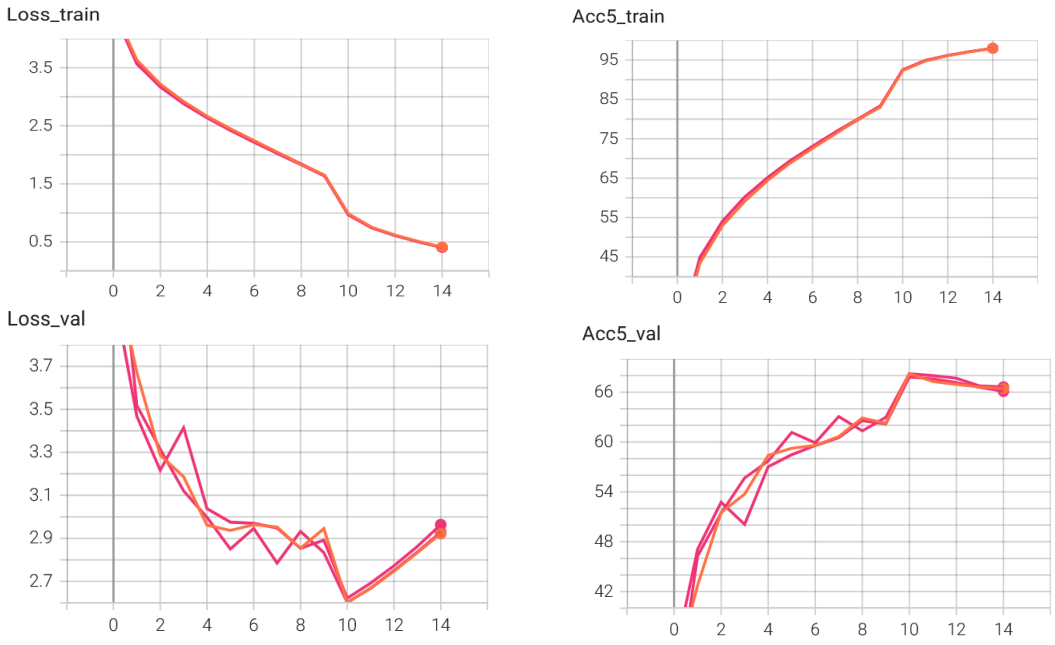
根据我的系统中 Tiny-Imagenet-200 数据集的位置，调整了 traindir 和 valdir.

```
+           if epoch in [4, 9, 14]:  # save checkpoint at the epoch we want
+               torch.save({
+                   'epoch': epoch + 1,
+                   'arch': args.arch,
+                   'state_dict': model.state_dict(),
+                   'best_acc1': best_acc1,
+                   'optimizer': optimizer.state_dict(),
+                   'scheduler': scheduler.state_dict()
+               }, 'checkpoint'+str(epoch+1)+'.pth.tar')
+   writer.close()
```

额外保存了 3 个 checkpoint 以便分析。分析内容见后文 5.3.

## 5. 实验结果与讨论

5.1(TensorBoard)Loss 和 top5 精度的变化曲线




参数设定：不加载 pretrained 模型，batch_size=256, lr=0.1 且每 10 个 epoch 降至 0.1 倍，momentum=0.9, weight_decay=1e-4，训练 15 个 epoch，训练 3 次。

对于训练集：随着 epoch 增加，训练集 Loss 呈现良好的递减趋势，acc5(top5

准确度)单调递增至 98%，并仍具有上升的趋势，在第 10 个 epoch(坐标为 9，因为 epoch 从 0 开始计数)处分别有陡峭的下降和上升，符合学习率 lr 从 0.1 降至 0.01 而产生的变化趋势。

对于验证集：随着 epoch 增加，验证集 loss 先呈下降趋势，并趋于平稳，然后在第 10 个 epoch 处具有陡峭的下降，符合我们对于训练的预知，符合训练饱和以及学习率 lr 突然下降产生的改变。而后 loss 值缓慢上升，考虑是存在过拟合而引发 loss 值上升。随着 epoch 增加，验证集 acc5 先快速上涨，并趋于平稳，然后在第 10 个 epoch 处具有陡峭的上涨，然后由于一定程度的过拟合，缓慢下降并趋于平稳，最终验证集 acc5 约为 67%.

5.2 对于提升验证集准确度的尝试

经过一系列尝试，最终发现，加载 pretrained 模型 resnet18，batch_size=256，lr=0.01，momentum=0.9，weight_decay=1e-4，并简单地增强了训练集后，验证集 top5 准确度可以在 5 个 epoch 内上升并稳定于约 77%，同时 top1 准确度可达到 51%，如下图所示：

```
(base) PS C:\Users\86187> python .\Documents\GitHub\Weiczh-s-1st\hw2tinyimage\main.py --evaluate --resume C:/Users/86187/model_best.pth-256notr5.tar
=> creating model 'resnet18'
=> loading checkpoint 'C:/Users/86187/model_best.pth-256notr5.tar'
=> loaded checkpoint 'C:/Users/86187/model_best.pth-256notr5.tar' (epoch 5)
Test: [ 0/40]   Time 14.523 (14.523)    Loss 1.5435e+00 (1.5435e+00)    Acc@1  62.50 ( 62.50)   Acc@5  82.81 ( 82.81)
Test: [20/40]   Time  0.024 ( 0.723)    Loss 2.0116e+00 (2.0485e+00)    Acc@1  49.22 ( 51.93)   Acc@5  76.95 ( 77.06)
 *  Acc@1 51.500 Acc@5 76.740
(base) PS C:\Users\86187>
```

训练集准确度高而验证集准确度相对较低，我认为除了数据集较小的缘故，还可能由于一定程度的过拟合影响。在 weight_decay(权值衰减以防止过拟合)之外，我尝试在 loss 函数中加入 l2 正则化项作为惩罚机制，使验证集 top5 准确度略微上升至 78%。由于时间关系，我并没有继续尝试作提升，但是我认为继续调整 weight_decay 和 l2 正则化项系数值，可以再小幅提升验证集的准确度。

5.3Checkpoint 的保存与评估

除了原代码为了断点训练设定的 checkpoint 保存，本实验额外保存了第 5、10、15 个 epoch 训练后的 checkpoint。如下图所示：


第 5 个 epoch 处于验证准确度上升阶段；第 10 个 epoch 处于当前学习率下(lr=0.1)验证准确度饱和阶段，且是降低学习率前的最后 1 个 epoch；第 15 个 epoch 处于降低学习率(lr=0.01)后的验证准确度饱和阶段。

使用代码中的 evaluate 选项，对这 3 个特殊的 checkpoint 作评估，评估结果如下图所示：

```
(base) PS C:\Users\86187> python .\Documents\GitHub\Weiczh-s-1st\hw2tinyimage\main.py --evaluate --resume C:/Users/86187/checkpoint5.pth.tar
=> creating model 'resnet18'
=> loading checkpoint 'C:/Users/86187/checkpoint5.pth.tar'
=> loaded checkpoint 'C:/Users/86187/checkpoint5.pth.tar' (epoch 5)
Test: [ 0/40]   Time 13.980 (13.980)    Loss 3.2970e+00 (3.2970e+00)    Acc@1  28.91 ( 28.91)   Acc@5  51.17 ( 51.17)
Test: [20/40]   Time  0.024 ( 0.696)    Loss 3.3185e+00 (3.2507e+00)    Acc@1  27.73 ( 28.20)   Acc@5  49.22 ( 53.16)
 *   Acc@1 28.380 Acc@5 53.880
(base) PS C:\Users\86187> python .\Documents\GitHub\Weiczh-s-1st\hw2tinyimage\main.py --evaluate --resume C:/Users/86187/checkpoint10.pth.tar
=> creating model 'resnet18'
=> loading checkpoint 'C:/Users/86187/checkpoint10.pth.tar'
=> loaded checkpoint 'C:/Users/86187/checkpoint10.pth.tar' (epoch 10)
Test: [ 0/40]   Time 15.100 (15.100)    Loss 2.1221e+00 (2.1221e+00)    Acc@1  47.27 ( 47.27)   Acc@5  73.05 ( 73.05)
Test: [20/40]   Time  0.024 ( 0.750)    Loss 3.0362e+00 (2.9394e+00)    Acc@1  33.20 ( 34.54)   Acc@5  59.77 ( 61.64)
 *   Acc@1 34.870 Acc@5 61.660
(base) PS C:\Users\86187> python .\Documents\GitHub\Weiczh-s-1st\hw2tinyimage\main.py --evaluate --resume C:/Users/86187/checkpoint15.pth.tar
=> creating model 'resnet18'
=> loading checkpoint 'C:/Users/86187/checkpoint15.pth.tar'
=> loaded checkpoint 'C:/Users/86187/checkpoint15.pth.tar' (epoch 15)
Test: [ 0/40]   Time 15.066 (15.066)    Loss 1.9094e+00 (1.9094e+00)    Acc@1  55.08 ( 55.08)   Acc@5  80.08 ( 80.08)
Test: [20/40]   Time  0.025 ( 0.750)    Loss 2.8544e+00 (2.9223e+00)    Acc@1  39.84 ( 40.70)   Acc@5  64.45 ( 66.61)
 *   Acc@1 40.710 Acc@5 66.470
(base) PS C:\Users\86187>
```