Resnet18-TinyImageNet测试报告

王玉峰 *PB19020517*

1 代码改动

```
import shutil
import torch
import torch.nn as nn
import torch.nn.parallel
import torch.backends.cudnn as cudnn
import torch.distributed as dist
import torch.optim
from torch.optim.lr_scheduler import StepLR
import torch.multiprocessing as mp
import torch.utils.data
import torch.utils.data.distributed
import torchvision.transforms as transforms
import torchvision.datasets as datasets
import torchvision.models as models
from torch.utils.tensorboard import SummaryWriter
writer=SummaryWriter()
```

Fig. 1: 导入必要的包

导入shutil和tensorboard
*shutil*用于修改验证集数据的存储方式，将对应类别的图片存放到对应类别的文件夹内。
*tensorboard*用于记录训练过程中的准确率、误差的变化，以及绘制出网络结构。

```python
parser.add_argument('--data', metavar='DIR', default='E:\\OneDrive - USTC\\Python\
                    help='path to dataset (default: imagenet)')
parser.add_argument('-a', '--arch', metavar='ARCH', default='resnet18',
                    choices=model_names,
                    help='model architecture: ' +
                        ' | '.join(model_names) +
                        ' (default: resnet18)')
parser.add_argument('-j', '--workers', default=0, type=int, metavar='N',
                    help='number of data loading workers (default: 4)')
parser.add_argument('--epochs', default=90, type=int, metavar='N',
                    help='number of total epochs to run')
parser.add_argument('--start-epoch', default=0, type=int, metavar='N',
                    help='manual epoch number (useful on restarts)')
parser.add_argument('-b', '--batch-size', default=200, type=int,
                    metavar='N',
                    help='mini-batch size (default: 256), this is the total '
                         'batch size of all GPUs on the current node when '
                         'using Data Parallel or Distributed Data Parallel')
parser.add_argument('--lr', '--learning-rate', default=0.1, type=float,
                    metavar='LR', help='initial learning rate', dest='lr')
parser.add_argument('--momentum', default=0.9, type=float, metavar='M',
                    help='momentum')
parser.add_argument('--wd', '--weight-decay', default=1e-4, type=float,
                    metavar='W', help='weight decay (default: 1e-4)',
                    dest='weight_decay')
parser.add_argument('-p', '--print-freq', default=10, type=int,
                    metavar='N', help='print frequency (default: 10)')
parser.add_argument('-r','--resume', default='checkpoint.pth.tar', type=str, metav
```

Fig. 2: 参数更改

$--data$更改了项目数据路径；

$--workers$取消了多线程加载，防止加载数据时因线程冲突报错；

$--batch-size$更改为200，防止显存爆炸；

$--resume$新增了断点文件路径，防止程序突然中断后重启无法继续上次的训练结果。

```python
    # set output dimension as 200
    model.avgpool=nn.AdaptiveAvgPool2d((1,1))
    model.fc=nn.Linear(512,200)
    writer.add_graph(model,torch.rand([1,3,64,64],dtype=torch.float32))
    # print(model)
```

Fig. 3: 输出维度

更改输出维度为200（训练集数据有200类），并且将模型结构保存到Tensorboard日志。

```python
    #get img-class info
    anno='E:\\OneDrive - USTC\\Python\\Python and Deep learning\\tiny-imagenet-20
    imgs=[]
    classes=[]
    with open(anno,'r') as f:
        while True:
            con=f.readline()
            if con:
                classes.append(con.split('\t')[1])
                imgs.append(con.split('\t')[0])
            else:
                break
    f.close()

    # change file path
    images='E:\\OneDrive - USTC\\Python\\Python and Deep learning\\tiny-imagenet-
    if os.path.isdir(os.path.join(images,'images')):
        for i in range(len(classes)):
            try:
                os.mkdir(os.path.join(images,classes[i]))
            except:
                pass

        for i in range(len(imgs)):
            shutil.copyfile(os.path.join(images,'images',imgs[i]),os.path.join(ima
        shutil.rmtree(os.path.join(images,'images'))

    val_dataset=datasets.ImageFolder(valdir, transforms.Compose([transforms.Resize
    val_loader = torch.utils.data.DataLoader(val_dataset,
        batch_size=args.batch_size, shuffle=False,
        num_workers=args.workers, pin_memory=True)
```

Fig. 4: 验证集数据导入

如前所述，此处我们新建了名称为类名的文件夹，并且将对应类别的图片移动到对应类别的文件夹内，最后使用$dataset.ImageFolder()$加载数据集。

```python
    # add logs
    # writer.add_image(tag='training images{}'.format(epoch),img_tensor=output,glo
    writer.add_scalar(tag='train_acc5',scalar_value=acc5[0],global_step=epoch)
    writer.add_scalar(tag='train_loss',scalar_value=loss.item(),global_step=epoch)
```

Fig. 5: 添加tensorboard日志

2 网络结构展示

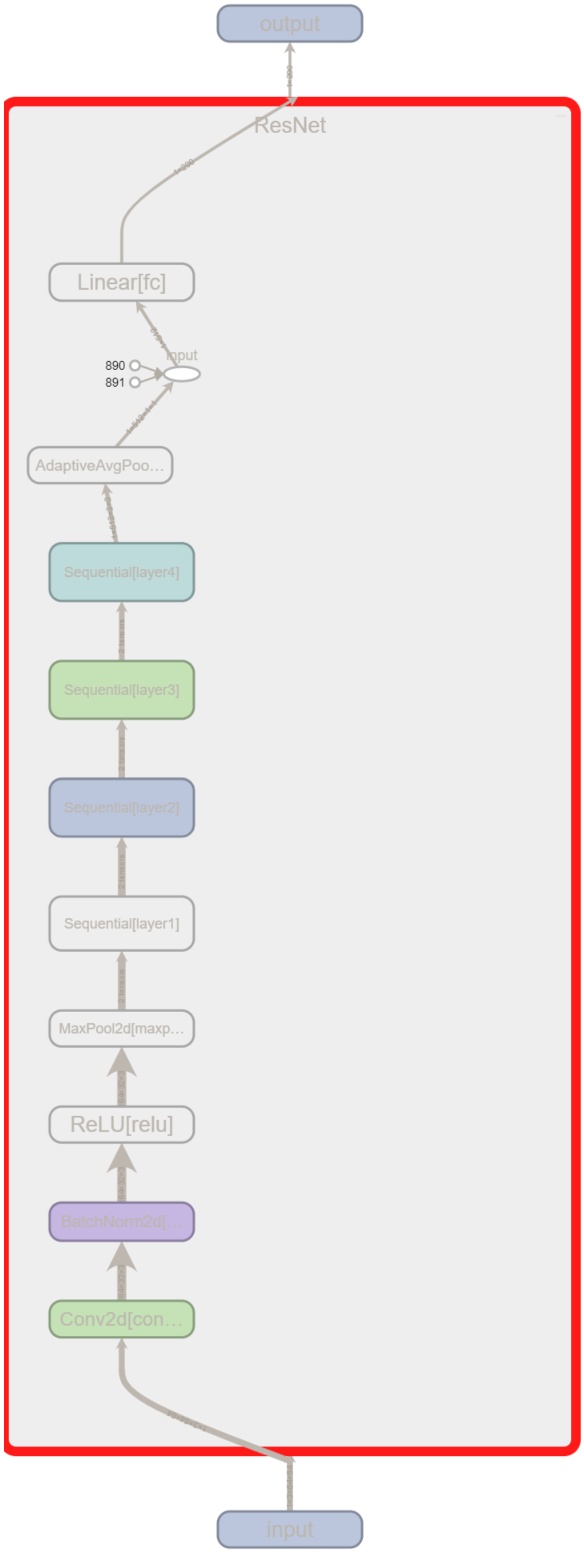

Fig. 6: 网络结构

从Tensorboard保存的网络结构上可以看出，输入层数据维度为$1\times3\times64\times64$,接下来数据进入第一个2维卷

积层，2维卷积层的输出数据维度为$1\times64\times32\times32$，接下来数据进入2维批正则化层，该层数据的输出维度为$1\times64\times32\times32$,然后数据进入激活函数ReLU，输出维度不变，再然后依次进入池化层，三个全连接层，其输出数据维度为$1\times512\times2\times2$,然后数据进入二元自适应均值汇聚层，输出维度为$1\times512\times1\times1$，然后这些数据被压平，得到1×512维的输出，最后是一个线性变换层，获得维度为1×200的输出。（所有数据均可在图上找到，数据维度与箭头颜色过于相近不太好分辨）

3 训练结果分析（横轴是小时）

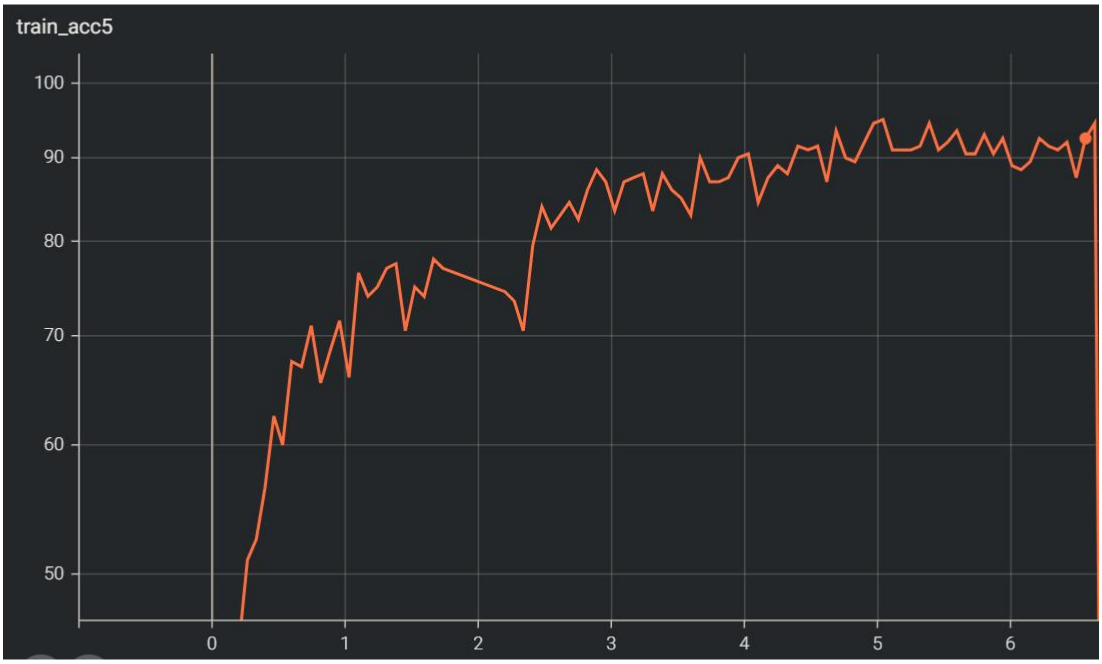

Fig. 7: 训练集准确率曲线（90 epochs）

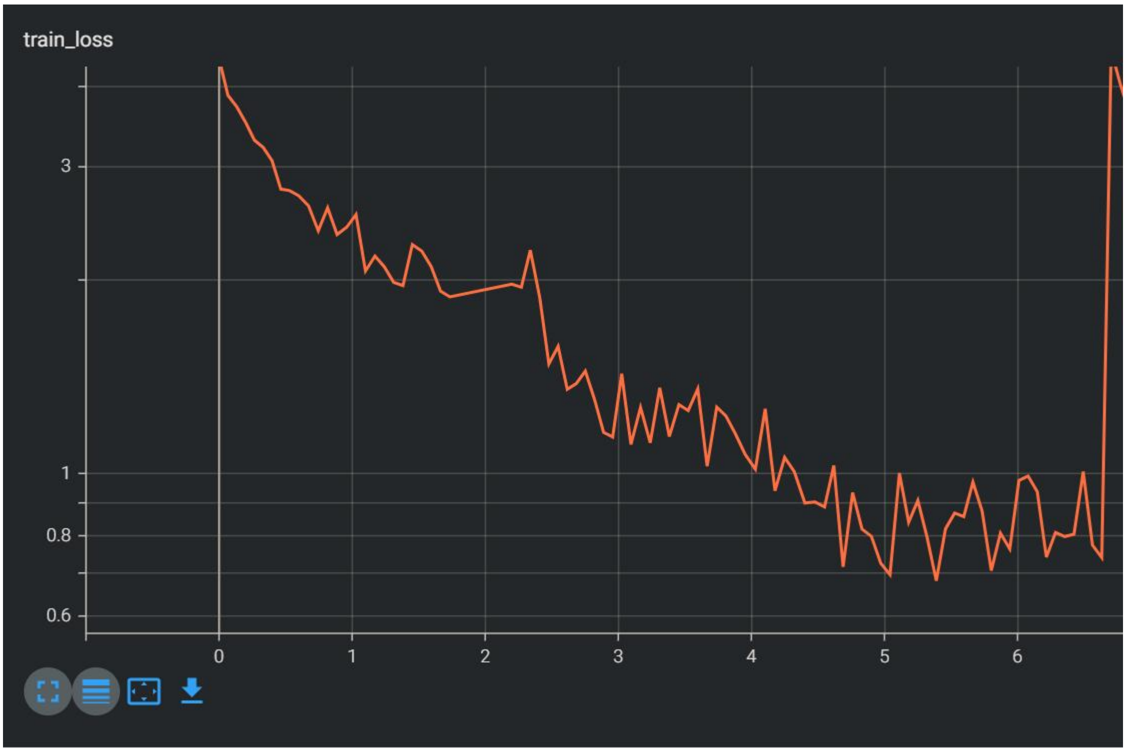

Fig. 8: 训练集误差曲线（90 epochs）

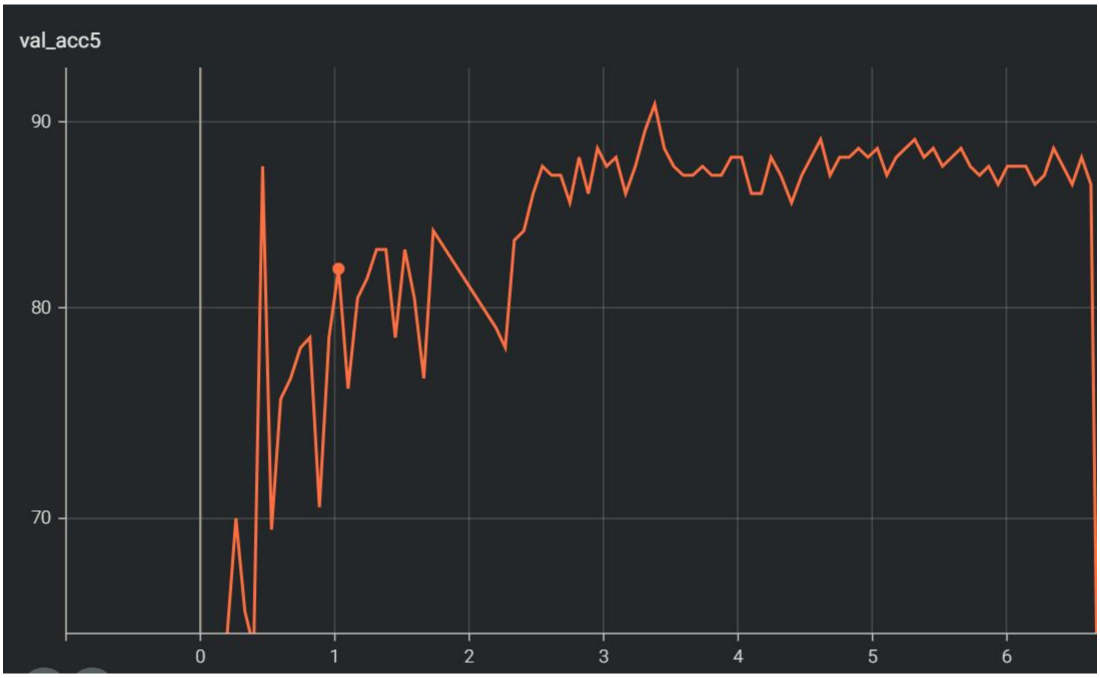

Fig. 9: 验证集准确率曲线（90 epochs）

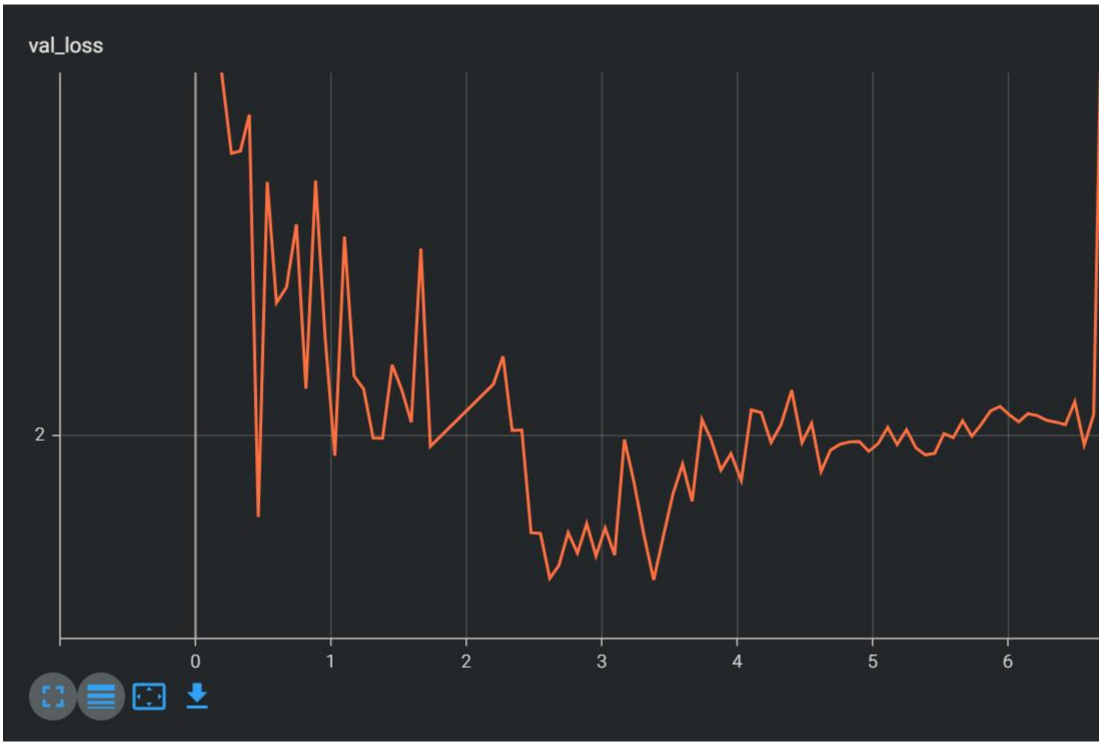

Fig. 10: 验证集误差曲线（90 epochs）

显然随着训练次数的增加，准确率也在随之上升，当上升到一定数值时由于准确率不收敛，我们需要适当降低学习率（×1/10），在此之后准确率进一步上升，误差进一步下降，经过几次循环后准确率趋于稳定值95%附近（最高准确率96.71%），最终准确率的上限取决于epoch、batch-size以及模型本身的性质。

4 过程比较

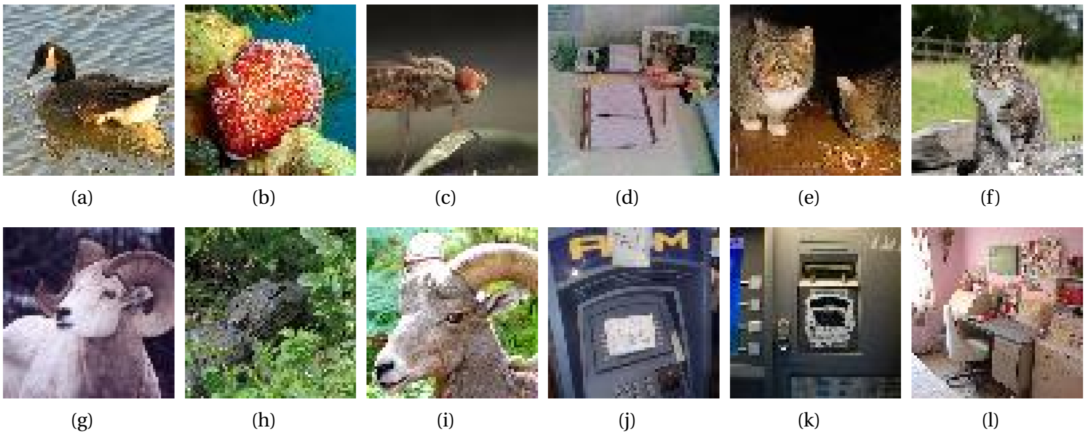

Fig. 11: 12个判断结果不一的样本（在不同epoch中）

```
E:\OneDrive - USTC\Python\Python and Deep learning\Image-Net\imagenet>python main.py -e
=> creating model 'resnet18'
=> loading checkpoint 'checkpoint01.pth.tar'
=> loaded checkpoint 'checkpoint01.pth.tar' (epoch 38)
Test: [ 0/50]   Time  2.908 ( 2.908)    Loss 1.1079e+00 (1.1079e+00)    Acc@1  68.00 ( 68.00)    Acc@5  89.00 ( 89.00)
Test: [10/50]   Time  0.361 ( 0.603)    Loss 1.7460e+00 (1.4441e+00)    Acc@1  54.50 ( 64.59)    Acc@5  88.50 ( 86.50)
Test: [20/50]   Time  0.372 ( 0.494)    Loss 1.7802e+00 (1.5332e+00)    Acc@1  61.00 ( 63.48)    Acc@5  78.00 ( 84.57)
Test: [30/50]   Time  0.364 ( 0.453)    Loss 1.8976e+00 (1.5963e+00)    Acc@1  52.50 ( 62.10)    Acc@5  76.50 ( 83.39)
Test: [40/50]   Time  0.365 ( 0.432)    Loss 1.4586e+00 (1.6621e+00)    Acc@1  64.50 ( 60.70)    Acc@5  85.50 ( 82.33)
 *   Acc@1 61.250 Acc@5 83.020
```

Fig. 12: 第38epoch的验证集正确率

```
E:\OneDrive - USTC\Python\Python and Deep learning\Image-Net\imagenet>python main.py -e
=> creating model 'resnet18'
=> loading checkpoint 'checkpoint.pth.tar'
=> loaded checkpoint 'checkpoint.pth.tar' (epoch 76)
Test: [ 0/50]   Time  2.915 ( 2.915)    Loss 1.1167e+00 (1.1167e+00)    Acc@1  72.50 ( 72.50)    Acc@5  91.00 ( 91.00)
Test: [10/50]   Time  0.372 ( 0.598)    Loss 1.8274e+00 (1.5496e+00)    Acc@1  58.00 ( 65.45)    Acc@5  86.00 ( 85.91)
Test: [20/50]   Time  0.364 ( 0.488)    Loss 1.9823e+00 (1.6766e+00)    Acc@1  61.50 ( 63.88)    Acc@5  80.50 ( 83.83)
Test: [30/50]   Time  0.360 ( 0.449)    Loss 2.2436e+00 (1.7648e+00)    Acc@1  46.00 ( 61.71)    Acc@5  76.00 ( 82.58)
Test: [40/50]   Time  0.371 ( 0.429)    Loss 1.5382e+00 (1.8305e+00)    Acc@1  64.50 ( 60.82)    Acc@5  85.50 ( 81.55)
 *   Acc@1 61.170 Acc@5 82.280
```

Fig. 13: 第76epoch的验证集正确率

显然两次checkpoint得到的预测结果不一样，按理来说应该epoch越大准确率越高，但是在这个实验中由于batch-size设置等原因，当准确率达到82%之后，模型收敛速度很慢，所以出现准确率跳动的情形在所难免，但是从我们之前给出的准确率曲线数据来看，虽然准确率存在一定方差，但是总体上准确率是随着训练次数的增加而增大的。

5 其他总结

1.batch-size 对神经网络参数收敛的速度影响非常明显，并且越大的batch-size更有可能得到越大的准确率
2.一张好的显卡异常重要（尤其是大显存）
3.越复杂的网络稳定性越高，初期收敛的速度也越快，但是训练所需的时间也越长

6 说明

代码改动记录以及改动后的代码已经上传到Github，