在 Tiny-ImageNet 数据集上训练 Resnet 模型

Jonathon Chow *

（中国科学技术大学数学科学学院）

2022/5/17

摘要

Tiny-ImageNet 是一个小型的、常用于教学的数据集，包括 200 个类型，10 万张训练图片和 1 万张验证图片。在本次实验中，以 PyTorch 官网基于 ImageNet 的示例代码为基础，在其上修改以适配 Tiny-ImageNet，并且限定模型为 resnet18。

本次实验的代码地址参见https://github.com/JONATHONCHOW/tiny-imagenet_resnet18

前言

本次实验主要完成了 resnet18.py、update_val.py、main.py、compare.py，分别实现了分析 resnet18 模型结构、修改训练数据集 val、改动示例代码使适配 Tiny-ImageNet、比较两个 checkpoint 差异。

任务 1：分析 resnet18 模型结构

利用 PyTorch 的模型解析的包，再根据 Tiny-ImageNet 图片大小（3*64*64）编写代码。

```
model = models.__dict__['resnet18']()
input_num = model.fc.in_features
```

*E-mail: jonathonchow23@gmail.com

```
model.fc = nn.Linear(input_num, 200)
# gpu to cpu
summary(model, (3, 64, 64), device = "cpu")
```

运行文件 resnet18.py，得到各层的名称及图片经过各层处理后的中间结果的大小。

```
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 32, 32]           9,408
       BatchNorm2d-2           [-1, 64, 32, 32]             128
              ReLU-3           [-1, 64, 32, 32]               0
         MaxPool2d-4           [-1, 64, 16, 16]               0
            Conv2d-5           [-1, 64, 16, 16]          36,864
       BatchNorm2d-6           [-1, 64, 16, 16]             128
              ReLU-7           [-1, 64, 16, 16]               0
            Conv2d-8           [-1, 64, 16, 16]          36,864
       BatchNorm2d-9           [-1, 64, 16, 16]             128
             ReLU-10           [-1, 64, 16, 16]               0
       BasicBlock-11           [-1, 64, 16, 16]               0
           Conv2d-12           [-1, 64, 16, 16]          36,864
      BatchNorm2d-13           [-1, 64, 16, 16]             128
             ReLU-14           [-1, 64, 16, 16]               0
           Conv2d-15           [-1, 64, 16, 16]          36,864
      BatchNorm2d-16           [-1, 64, 16, 16]             128
             ReLU-17           [-1, 64, 16, 16]               0
       BasicBlock-18           [-1, 64, 16, 16]               0
           Conv2d-19            [-1, 128, 8, 8]          73,728
      BatchNorm2d-20            [-1, 128, 8, 8]             256
             ReLU-21            [-1, 128, 8, 8]               0
           Conv2d-22            [-1, 128, 8, 8]         147,456
      BatchNorm2d-23            [-1, 128, 8, 8]             256
           Conv2d-24            [-1, 128, 8, 8]           8,192
      BatchNorm2d-25            [-1, 128, 8, 8]             256
             ReLU-26            [-1, 128, 8, 8]               0
       BasicBlock-27            [-1, 128, 8, 8]               0
           Conv2d-28            [-1, 128, 8, 8]         147,456
      BatchNorm2d-29            [-1, 128, 8, 8]             256
             ReLU-30            [-1, 128, 8, 8]               0
           Conv2d-31            [-1, 128, 8, 8]         147,456
      BatchNorm2d-32            [-1, 128, 8, 8]             256
             ReLU-33            [-1, 128, 8, 8]               0
       BasicBlock-34            [-1, 128, 8, 8]               0
           Conv2d-35            [-1, 256, 4, 4]         294,912
      BatchNorm2d-36            [-1, 256, 4, 4]             512
             ReLU-37            [-1, 256, 4, 4]               0
           Conv2d-38            [-1, 256, 4, 4]         589,824
      BatchNorm2d-39            [-1, 256, 4, 4]             512
           Conv2d-40            [-1, 256, 4, 4]          32,768
      BatchNorm2d-41            [-1, 256, 4, 4]             512
```

```
             ReLU-42              [-1, 256, 4, 4]               0
        BasicBlock-43             [-1, 256, 4, 4]               0
            Conv2d-44             [-1, 256, 4, 4]         589,824
       BatchNorm2d-45             [-1, 256, 4, 4]             512
             ReLU-46              [-1, 256, 4, 4]               0
            Conv2d-47             [-1, 256, 4, 4]         589,824
       BatchNorm2d-48             [-1, 256, 4, 4]             512
             ReLU-49              [-1, 256, 4, 4]               0
        BasicBlock-50             [-1, 256, 4, 4]               0
            Conv2d-51             [-1, 512, 2, 2]       1,179,648
       BatchNorm2d-52             [-1, 512, 2, 2]           1,024
             ReLU-53              [-1, 512, 2, 2]               0
            Conv2d-54             [-1, 512, 2, 2]       2,359,296
       BatchNorm2d-55             [-1, 512, 2, 2]           1,024
            Conv2d-56             [-1, 512, 2, 2]         131,072
       BatchNorm2d-57             [-1, 512, 2, 2]           1,024
             ReLU-58              [-1, 512, 2, 2]               0
        BasicBlock-59             [-1, 512, 2, 2]               0
            Conv2d-60             [-1, 512, 2, 2]       2,359,296
       BatchNorm2d-61             [-1, 512, 2, 2]           1,024
             ReLU-62              [-1, 512, 2, 2]               0
            Conv2d-63             [-1, 512, 2, 2]       2,359,296
       BatchNorm2d-64             [-1, 512, 2, 2]           1,024
             ReLU-65              [-1, 512, 2, 2]               0
        BasicBlock-66             [-1, 512, 2, 2]               0
AdaptiveAvgPool2d-67              [-1, 512, 1, 1]               0
            Linear-68                   [-1, 200]         102,600
================================================================
```

任务 2：改动示例代码使适配 Tiny-ImageNet

以 PyTorch 官网基于 ImageNet 的示例代码 main.py 为基础，在其上修改以适配 Tiny-ImageNet。[1]

2.1 修改训练数据集 val

利用 os 库先将 val 文件夹重命名为 originval 文件夹，再重新创建 val 文件夹。

```
os.rename("./tiny-imagenet-200/val", "./tiny-imagenet-200/originval")
os.mkdir("./tiny-imagenet-200/val")
```

利用 train 文件夹的结构，构造 val 文件夹，使得更新后的 val 文件夹结构与 ImageNet 的相同。

[1] main.py 的改动说明参见附录。

```python
for filename in os.listdir("./tiny-imagenet-200/train"):
    if filename in os.listdir("./tiny-imagenet-200/originval"):
        pass
    else:
        os.mkdir(os.path.join("./tiny-imagenet-200/val", filename))
```

利用 val_annotations.txt 文件重新修订标签。

```python
with open("./tiny-imagenet-200/originval/val_annotations.txt") as f:
    labels = f.readlines()
for i, label in enumerate(labels):
    filename = label.split("\t")[1]
    src = "./tiny-imagenet-200/originval/images/val_" + str(i) + ".JPEG"
    dst = "./tiny-imagenet-200/val/" + filename + "/val_" + str(i) +".JPEG"
    shutil.copyfile(src, dst)
```

在运行 main.py 之前需要先运行文件 update_val.py。

2.2 其他修改

在 main.py 中修改 output 维数、删除对图片进行伸缩和裁剪的代码。

```python
# change output dimension from 1000 to 200
input_num = model.fc.in_features
model.fc = nn.Linear(input_num, 200)
```

```python
# transforms.RandomResizedCrop(224),
# transforms.RandomHorizontalFlip(),
# transforms.Resize(256),
# transforms.CenterCrop(224),
```

任务 3：在 TensorBoard 中观察训练参数

在代码中增加 torch.utils.tensorboard 的代码，利用函数传参，使得能在 TensorBoard 中观察训练集 Loss、训练集精度、验证集 Loss、验证集精度的变化。

```python
writer.add_scalar('Train/Loss', losses_train, epoch)
writer.add_scalar('Train/Top5', acc5_train, epoch)
writer.add_scalar('Train/Top1', acc1_train, epoch)
```

```
writer.add_scalar('Validate/Loss', losses, epoch)
writer.add_scalar('Validate/Top5', acc5, epoch)
writer.add_scalar('Validate/Top1', acc1, epoch)
```

任务 4：运行程序并分析曲线变化

4.1 运行程序

运行 main.py，为将 resnet18 在训练集上的精度（Top5）训练到 95% 以上，batch_size 为默认值 256，设置 epochs=25，事实上大约 12 个 epochs 就可以达到精度要求。

在任务端[2]的 Command Prompt 中输入命令[3]。

```
python main.py --epochs 25 tiny-imagenet-200
```

4.2 利用 TensorBoard 分析曲线变化

在任务端的 Command Prompt 中输入命令打开 TensorBoard。

```
tensorboard --logdir=runs
```

得到训练集 Loss、训练集精度（Top1、Top5）、验证集 Loss、验证集精度（Top1、Top5）的变化曲线图，参见图 1、图 2。

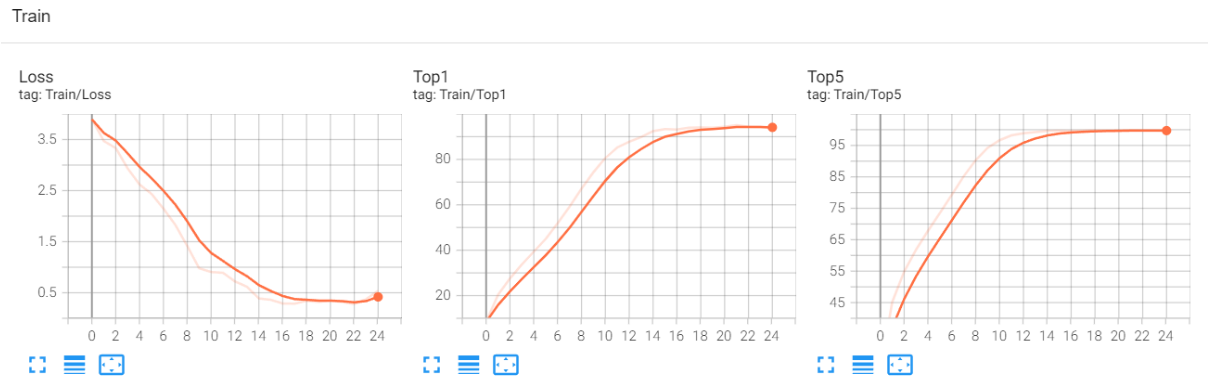


图 1: TensorBoard_Train

[2]注意不能是本地端的 Command Prompt。

[3]也可以在 IDE 的运行配置中设置形参。

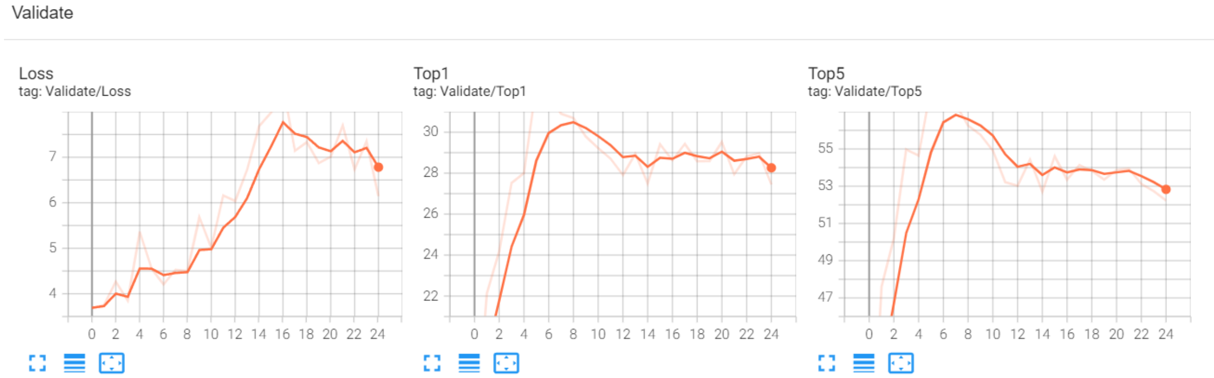


图 2: TensorBoard_Validate

训练集：Loss 曲线单调下降，这是利用梯度下降法的原因。Top1 和 Top5 曲线单调上升，模型在大约 15 个 epochs 处就趋于稳定，精度分别达到了 90% 和 95% 以上。

验证集：模型存在轻微的过拟合现象，这是训练时间过长的原因。Loss 曲线在大约 15 个 epochs 处达到峰值，然后轻微下降趋于平稳。Top1 和 Top5 曲线在大约 8 个 epochs 处达到峰值，然后轻微下降趋于平稳，精度分别在 30% 和 55% 左右。

任务 5：保存 checkpoint 并评估差异

5.1 保存 checkpoint

保存每个 epoch 的 checkpoint 总共 25 个，以及验证集 Top1 曲线峰值时的 checkpoint。

```
save_checkpoint({
    'epoch': epoch + 1,
    'arch': args.arch,
    'state_dict': model.state_dict(),
    'best_acc1': best_acc1,
    'optimizer': optimizer.state_dict(),
    'scheduler': scheduler.state_dict()
    }, is_best, "checkpoint{}.pth.tar".format(epoch + 1))
```

利用 checkpoint 可以方便地进行断点重启[4]，比如从第 10 个 epoch 开始继续运行（需要恢复第 10 个 epoch[5]运行结束后的断点 checkpoint10.pth.tar），在任务端的 Command Prompt 中输入如下命令。

[4]注意并不需要手工赋予参数 start_epoch 的值，这是由于 checkpoint 中已经有 start_epoch 的值。
[5]注意第 i 个 epoch 相当于 epoch i 也相当于 Epoch: [i-1]。

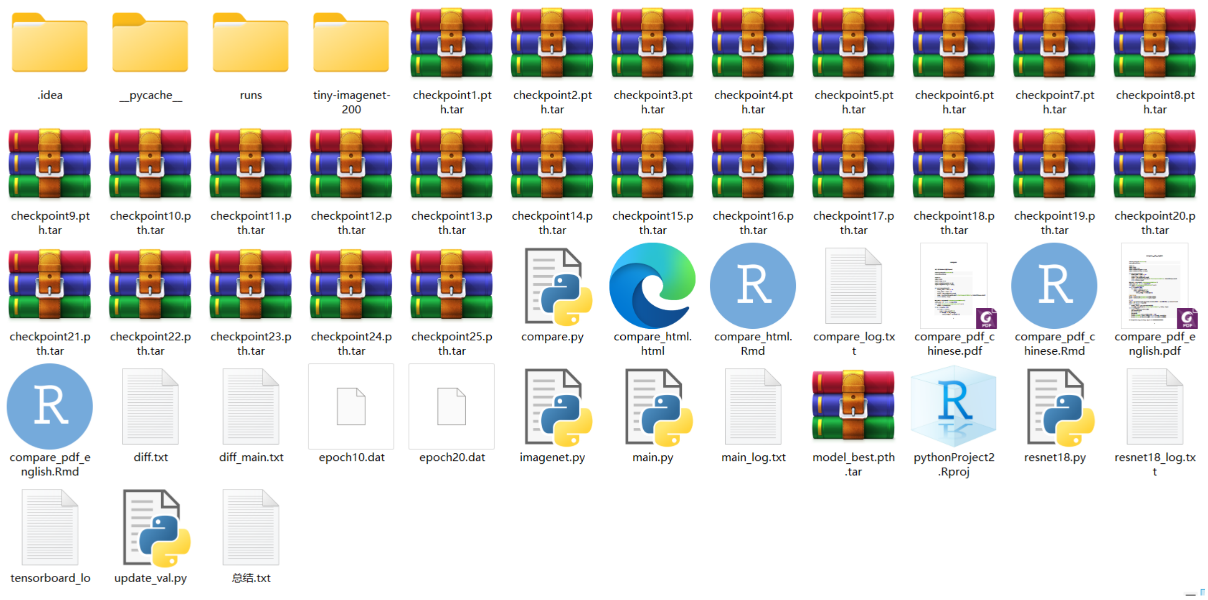


图 3: checkpoint

```
    python main.py --epochs 25 --resume checkpoint10.pth.tar tiny-imagenet-200
```

5.2 评估不同 checkpoint 的差异

使用代码中的 –evaluate 选项，编写代码以保存指定 epoch（相当于指定 checkpoint）处的训练结果。

```
if args.evaluate:
    torch.save(save, "epoch" + str(args.start_epoch) + ".dat")
    return
```

在任务端的 Command Prompt 中输入如下命令以保存第 10 个和第 20 个 epoch 处的训练结果，存储为文件 epoch10.dat 和 epoch20.dat。

```
python main.py --resume checkpoint10.pth.tar -e tiny-imagenet-200
python main.py --resume checkpoint20.pth.tar -e tiny-imagenet-200
```

对比两次评估的差异：由于验证集存在轻微的过拟合现象，第 10 个 epoch 处的准确率比第 20 个 epoch 处的准确率略高一些。

```
Epoch: [9]
Test: [ 0/40]   Time  5.740 ( 5.740)    Loss 3.0113e+00 (3.0113e+00)    Acc@1  38.67 ( ...
    38.67)   Acc@5  62.89 ( 62.89)
Test: [10/40]   Time  0.023 ( 0.562)    Loss 3.7513e+00 (3.5626e+00)    Acc@1  26.56 ( ...
    31.36)   Acc@5  55.47 ( 58.03)
Test: [20/40]   Time  0.054 ( 0.313)    Loss 3.4789e+00 (3.7384e+00)    Acc@1  32.03 ( ...
    29.32)   Acc@5  56.25 ( 55.77)
Test: [30/40]   Time  0.053 ( 0.226)    Loss 3.9178e+00 (3.7831e+00)    Acc@1  29.30 ( ...
    29.07)   Acc@5  56.25 ( 55.12)
 *   Acc@1 29.790 Acc@5 55.760
```

```
Epoch: [19]
Test: [ 0/40]   Time  5.696 ( 5.696)    Loss 4.1153e+00 (4.1153e+00)    Acc@1  35.16 ( ...
    35.16)   Acc@5  61.72 ( 61.72)
Test: [10/40]   Time  0.023 ( 0.557)    Loss 4.6135e+00 (4.6518e+00)    Acc@1  31.25 ( ...
    30.01)   Acc@5  56.25 ( 56.89)
Test: [20/40]   Time  0.039 ( 0.310)    Loss 4.7686e+00 (4.8993e+00)    Acc@1  25.78 ( ...
    27.79)   Acc@5  53.91 ( 53.39)
Test: [30/40]   Time  0.029 ( 0.224)    Loss 5.8639e+00 (4.9703e+00)    Acc@1  19.14 ( ...
    27.70)   Acc@5  45.31 ( 52.19)
 *   Acc@1 28.590 Acc@5 53.350
```

利用 os 库对文件地址操作、利用 numpy 库将 tensor 转化成易于处理的数据类型、利用 matplotlib 库显示图像，编写程序得到 compare.py。

```python
for i in range(n):
    figure = mpimg.imread(files[i])
    plt.imshow(figure)
    plt.axis("off")
    plt.show()
    print("answer:", index_word[index[i] // 50])
    print("epoch10:", [index_word[j] for j in pred1[:,index[i]]])
    print("epoch20:", [index_word[j] for j in pred2[:,index[i]]])
```

运行 compare.py，找出其中 10 张评判结果不同的图片如下。

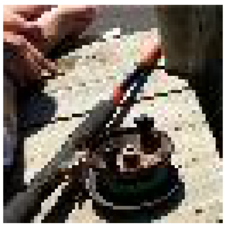
```
answer: altar
epoch10: ['barrel, cask', 'candle, taper, wax light', 'lampshade, lamp shade', 'pill ...
    bottle', 'pizza, pizza pie']
epoch20: ['walking stick, walkingstick, stick insect', 'cockroach, roach', "dragonfly, ...
    darning needle, devil's darning needle, sewing needle, snake feeder, snake ...
    doctor, mosquito hawk, skeeter hawk", 'broom', 'slug']
```

```
answer: cannon
epoch10: ['maypole', 'suspension bridge', 'cannon', 'projectile, missile', 'lawn ...
    mower, mower']
epoch20: ['crane', 'flagpole, flagstaff', 'maypole', 'cannon', 'standard poodle']
```

```
answer: freight car
epoch10: ['lifeboat', 'trolleybus, trolley coach, trackless trolley', 'school bus', ...
    'go-kart', 'moving van']
epoch20: ['trolleybus, trolley coach, trackless trolley', 'school bus', 'crane', ...
    'lifeboat', 'chest']
```

```
answer: moving van
epoch10: ['suspension bridge', 'triumphal arch', 'cannon', 'dam, dike, dyke', 'school ...
    bus']
```


```
epoch20: ['triumphal arch', 'dam, dike, dyke', 'butcher shop, meat market', 'bullet ...
    train, bullet', 'suspension bridge']
```



```
answer: reel
epoch10: ['rugby ball', 'kimono', 'reel', 'military uniform', 'binoculars, field ...
    glasses, opera glasses']
epoch20: ['reel', 'rugby ball', 'goose', 'gondola', 'sports car, sport car']
```

```
answer: sewing machine
epoch10: ['abacus', 'dining table, board', 'freight car', 'organ, pipe organ', ...
    'barbershop']
epoch20: ['barbershop', 'scoreboard', 'abacus', 'torch', 'oboe, hautboy, hautbois']
```

```
answer: torch
epoch10: ['snail', 'fly', 'bee', 'acorn', 'tailed frog, bell toad, ribbed toad, tailed ...
    toad, Ascaphus trui']
epoch20: ['reel', 'bullfrog, Rana catesbeiana', 'tailed frog, bell toad, ribbed toad, ...
    tailed toad, Ascaphus trui', 'European fire salamander, Salamandra salamandra', ...
    'black stork, Ciconia nigra']
```

```
answer: wooden spoon
epoch10: ['wooden spoon', 'espresso', 'slug', 'bathtub, bathing tub, bath, tub', ...
    'drumstick']
epoch20: ['espresso', 'mashed potato', 'plate', 'wooden spoon', 'Chihuahua']
```

```
answer: cauliflower
```

```
epoch10: ['poncho', 'bee', 'cauliflower', 'lemon', "spider web, spider's web"]
epoch20: ['bee', 'pomegranate', 'sulphur butterfly, sulfur butterfly', "spider web, ...
    spider's web", 'mushroom']
```

```
answer: orange
epoch10: ['school bus', 'crane', 'frying pan, frypan, skillet', 'wok', 'police van, ...
    police wagon, paddy wagon, patrol wagon, wagon, black Maria']
epoch20: ['moving van', 'goldfish, Carassius auratus', 'jellyfish', 'crane', 'beacon, ...
    lighthouse, beacon light, pharos']
```

参考文献

[1] 郑歆慰：Python 与深度学习基础课件

[2] https://github.com/pytorch/examples/blob/main/imagenet/main.py

[3] https://pytorch.org/docs/stable/tensorboard.html

附录

main.py 的改动说明如下。[6]

```
diff --git
a / main.py
b / main.py
index
44e06
d8.
.297
```

[6] 参见网址 https://github.com/JONATHONCHOW/tiny-imagenet_resnet18/commit/e4971e55a84200bc75e660927c3d57de9b23ec50 或文件 diff_main.txt

```
baf7
100644
--- a / main.py
+++ b / main.py


@ @-19

, 12 + 19, 16 @ @
import torchvision.transforms as transforms
import torchvision.datasets as datasets
import torchvision.models as models

+
from torch.utils.tensorboard import SummaryWriter

+
+writer = SummaryWriter()

model_names = sorted(name for name in models.__dict__
                     if name.islower() and not name.startswith("__")
                     and callable(models.__dict__[name]))

parser = argparse.ArgumentParser(description='PyTorch ImageNet Training')
+ # enter parameters in "Run" or "Command Line"
parser.add_argument('data', metavar='DIR', default='imagenet',
                    help='path to dataset (default: imagenet)')
parser.add_argument('-a', '--arch', metavar='ARCH', default='resnet18',
@ @ -138, 6 + 142, 10 @ @


def main_worker(gpu, ngpus_per_node, args):
    print("=> creating model '{}'".format(args.arch))
    model = models.__dict__[args.arch]()


+  # change output dimension from 1000 to 200
+     input_num = model.fc.in_features
+     model.fc = nn.Linear(input_num, 200)
+
if not torch.cuda.is_available():
    print('using CPU, this will be slow')
elif args.distributed:


    @ @-213

, 8 + 221, 8 @ @


```

```
def main_worker(gpu, ngpus_per_node, args):
    train_dataset = datasets.ImageFolder(
        traindir,
        transforms.Compose([
            -           transforms.RandomResizedCrop(224),
            -           transforms.RandomHorizontalFlip(),
            + # transforms.RandomResizedCrop(224),
            + # transforms.RandomHorizontalFlip(),
            transforms.ToTensor(),
            normalize,
        ]))


@ @-230

, 8 + 238, 8 @ @


def main_worker(gpu, ngpus_per_node, args):
    val_loader = torch.utils.data.DataLoader(
        datasets.ImageFolder(valdir, transforms.Compose([
            -           transforms.Resize(256),
            -           transforms.CenterCrop(224),
            + # transforms.Resize(256),
            + # transforms.CenterCrop(224),
            transforms.ToTensor(),
            normalize,
        ])),
                @ @ -247, 13 + 255, 20 @ @

    def main_worker(gpu, ngpus_per_node, args):

        train_sampler.set_epoch(epoch)


# train for one epoch
-        train(train_loader, model, criterion, optimizer, epoch, args)
+        losses_train, acc5_train, acc1_train = train(train_loader, model, criterion, ...
    optimizer, epoch, args)

# evaluate on validation set
-        acc1 = validate(val_loader, model, criterion, args)
+        losses, acc5, acc1 = validate(val_loader, model, criterion, args)

scheduler.step()

+        writer.add_scalar('Train/Loss', losses_train, epoch)
+        writer.add_scalar('Train/Top5', acc5_train, epoch)
+        writer.add_scalar('Train/Top1', acc1_train, epoch)
+        writer.add_scalar('Validate/Loss', losses, epoch)
```

```
+         writer.add_scalar('Validate/Top5', acc5, epoch)
+         writer.add_scalar('Validate/Top1', acc1, epoch)
+
# remember best acc@1 and save checkpoint
is_best = acc1 > best_acc1
best_acc1 = max(acc1, best_acc1)


@ @-267

, 7 + 282, 7 @ @


def main_worker(gpu, ngpus_per_node, args):
    'best_acc1': best_acc1,
    'optimizer': optimizer.state_dict(),
    'scheduler': scheduler.state_dict()


-}, is_best)
+}, is_best, "checkpoint{}.pth.tar".format(epoch + 1))

def train(train_loader, model, criterion, optimizer, epoch, args):
    @ @-315

    , 6 + 330, 7 @ @

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

    if i % args.print_freq == 0:
        progress.display(i)


+
return loss, top5.avg, top1.avg


def validate(val_loader, model, criterion, args):


    @ @-342

, 6 + 358, 17 @ @


def validate(val_loader, model, criterion, args):
    output = model(images)
    loss = criterion(output, target)
```

```
+ # compute output in evaluation
+
if args.evaluate:
    + # find max
+                 _, pred = output.topk(5, 1, True, True)
+                 pred = pred.t()
+
if i == 0:
    +                         save = pred
+ else:
+                         save = torch.cat((save, pred), 1)
+
continue
+
# measure accuracy and record loss
acc1, acc5 = accuracy(output, target, topk=(1, 5))
losses.update(loss.item(), images.size(0))


@ @-355

, 9 + 382, 13 @ @


def validate(val_loader, model, criterion, args):
    if i % args.print_freq == 0:
        progress.display(i)


+
if args.evaluate:
    +             torch.save(save, "epoch" + str(args.start_epoch) + ".dat")
+
return
+
progress.display_summary()

-
return top1.avg
+
return loss, top5.avg, top1.avg


def save_checkpoint(state, is_best, filename='checkpoint.pth.tar'):


    @ @-454

, 4 + 485, 4 @ @

```

```
def accuracy(output, target, topk=(1,)):
    if __name__ == '__main__':


-       main()
\ No
newline
at
end
of
file
+       main()
```