skv2109-resnet50-V100

December 12, 2019

[13]:
```
from __future__ import print_function, division

import torch
import torch.nn as nn
import torch.optim as optim
from torch.optim import lr_scheduler
from torch.optim.lr_scheduler import ReduceLROnPlateau
import numpy as np
import torchvision
from torchvision import datasets, models, transforms
import matplotlib.pyplot as plt
import time
import os
import copy
import datetime
plt.ion()   # interactive mode

PATH = os.getcwd() + "/checkpoint/latestmodelv2.pt"
```

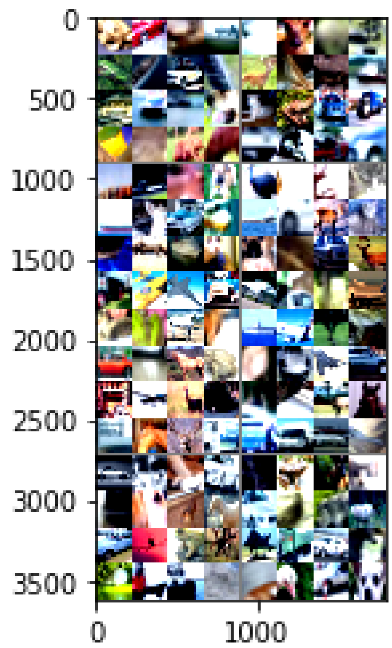
[14]:
```
data_transforms = {
    'train': transforms.Compose([
        transforms.RandomResizedCrop(224),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
    'val': transforms.Compose([
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
}
batch_size = 128
trainset = torchvision.datasets.CIFAR10(root='./data', train=True,
                                        download=True,␣
 ↪transform=data_transforms['train'])
```

```
trainloader = torch.utils.data.DataLoader(trainset, batch_size=batch_size,
                                          shuffle=True, num_workers=8)

testset = torchvision.datasets.CIFAR10(root='./data', train=False,
                                       download=True,␣
 ↪transform=data_transforms['val'])
testloader = torch.utils.data.DataLoader(testset, batch_size=batch_size,
                                         shuffle=False, num_workers=8)

dataloaders = {'train': trainloader , 'val': testloader }

classes = ('plane', 'car', 'bird', 'cat',
           'deer', 'dog', 'frog', 'horse', 'ship', 'truck')
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
dataset_sizes = {'train': len(trainset) , 'val': len(testset) }
```

```
Files already downloaded and verified
Files already downloaded and verified
```

[15]:
```
import matplotlib.pyplot as plt
import numpy as np

# functions to show an image


def imshow(img):
    img = img / 2 + 0.5     # unnormalize
    npimg = img.numpy()
    plt.imshow(np.transpose(npimg, (1, 2, 0)))
    plt.show()


# get some random training images
dataiter = iter(trainloader)
images, labels = dataiter.next()

# show images
imshow(torchvision.utils.make_grid(images))
# print labels
print(' '.join('%5s' % classes[labels[j]] for j in range(4)))
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```




```
frog   car  deer plane
```

```
[16]: def train_model(model, criterion, optimizer, scheduler, num_epochs=25):
    since = time.time()

    #best_model_wts = copy.deepcopy(model.state_dict())
    best_acc = 0.0
    val_loss= 100
    val_acc = -1

    for epoch in range(num_epochs):
        print('Epoch {}/{}'.format(epoch, num_epochs - 1))
        print('-' * 10)

        # Each epoch has a training and validation phase
        for phase in ['train', 'val']:
            if phase == 'train':
                model.train()  # Set model to training mode
            else:
                model.eval()   # Set model to evaluate mode

            running_loss = 0.0
            running_corrects = 0

            # Iterate over data.
```

```
            for inputs, labels in dataloaders[phase]:
                inputs = inputs.to(device)
                labels = labels.to(device)

                # zero the parameter gradients
                optimizer.zero_grad()

                # forward
                # track history if only in train
                with torch.set_grad_enabled(phase == 'train'):
                    outputs = model(inputs)
                    _, preds = torch.max(outputs, 1)
                    loss = criterion(outputs, labels)

                    # backward + optimize only if in training phase
                    if phase == 'train':
                        loss.backward()
                        optimizer.step()

                # statistics
                running_loss += loss.item() * inputs.size(0)
                running_corrects += torch.sum(preds == labels.data)
            if phase == 'train':
                scheduler.step(val_loss)

            epoch_loss = running_loss / dataset_sizes[phase]
            epoch_acc = running_corrects.double() / dataset_sizes[phase]
            if phase == 'val':
                val_loss = epoch_loss
                val_acc = epoch_acc

            print('{} Loss: {:.4f} Acc: {:.4f}'.format(
                phase, epoch_loss, epoch_acc))

            print( "Epoch Finish Time: ", datetime.datetime.now() )

            # deep copy the model
            if phase == 'val' and epoch_acc > best_acc:
                best_acc = epoch_acc
                #best_model_wts = copy.deepcopy(model.state_dict())

            torch.save({
                'epoch': epoch,
                'model_state_dict': model.state_dict(),
                'optimizer_state_dict': optimizer.state_dict(),
                'scheduler_state_dict': scheduler.state_dict(),
                'val_loss': val_loss,
```

```
                'val_acc' : val_acc,

            }, PATH)

        print()

    time_elapsed = time.time() - since
    print('Training complete in {:.0f}m {:.0f}s'.format(
        time_elapsed // 60, time_elapsed % 60))
    print('Best val Acc: {:4f}'.format(best_acc))

    # load best model weights
    model.load_state_dict(best_model_wts)
    return model
```

[5]:

```
#Model Def
m = models.resnet50()
m.fc = nn.Linear(2048, len(classes))

m = m.to(device)



criterion = nn.CrossEntropyLoss()

# Observe that only parameters of final layer are being optimized as
# opposed to before.
optimizer_conv = optim.SGD(m.parameters(), lr=0.1, momentum=0.9)

# Decay LR by a factor of 0.1 every 7 epochs
exp_lr_scheduler = lr_scheduler.StepLR(optimizer_conv, step_size=90, gamma=0.1)
#exp_lr_scheduler = ReduceLROnPlateau( optimizer_conv,patience=5,min_lr=0.5e-6)
```

[ ]:

```
print("Training Start Time: ", datetime.datetime.now() )
m = train_model(m, criterion, optimizer_conv, exp_lr_scheduler, num_epochs=350)
```

```
Training Start Time:  2019-12-11 16:58:32.410439
Epoch 0/349
----------
train Loss: 2.6184 Acc: 0.1775
Epoch Finish Time:  2019-12-11 17:01:01.067130
val Loss: 2.0597 Acc: 0.2350
Epoch Finish Time:  2019-12-11 17:01:13.198899

Epoch 1/349
----------
train Loss: 1.9770 Acc: 0.2533
Epoch Finish Time:  2019-12-11 17:03:42.360549
```

```
val Loss: 1.8755 Acc: 0.3122
Epoch Finish Time:  2019-12-11 17:03:55.719838

Epoch 2/349
----------
train Loss: 1.9158 Acc: 0.2809
Epoch Finish Time:  2019-12-11 17:06:24.931800
val Loss: 1.7174 Acc: 0.3657
Epoch Finish Time:  2019-12-11 17:06:38.229977

Epoch 3/349
----------
train Loss: 1.7752 Acc: 0.3411
Epoch Finish Time:  2019-12-11 17:09:07.227354
val Loss: 1.5761 Acc: 0.4260
Epoch Finish Time:  2019-12-11 17:09:20.837421

Epoch 4/349
----------
train Loss: 1.6874 Acc: 0.3771
Epoch Finish Time:  2019-12-11 17:11:50.469967
val Loss: 1.5051 Acc: 0.4499
Epoch Finish Time:  2019-12-11 17:12:03.913544

Epoch 5/349
----------
train Loss: 1.6073 Acc: 0.4127
Epoch Finish Time:  2019-12-11 17:14:33.408831
val Loss: 1.3672 Acc: 0.5110
Epoch Finish Time:  2019-12-11 17:14:46.796906

Epoch 6/349
----------
train Loss: 1.5332 Acc: 0.4434
Epoch Finish Time:  2019-12-11 17:17:16.148096
val Loss: 1.3165 Acc: 0.5237
Epoch Finish Time:  2019-12-11 17:17:29.451494

Epoch 7/349
----------
train Loss: 1.4304 Acc: 0.4873
Epoch Finish Time:  2019-12-11 17:19:58.171516
val Loss: 1.2061 Acc: 0.5689
Epoch Finish Time:  2019-12-11 17:20:10.723724

Epoch 8/349
----------
train Loss: 1.3539 Acc: 0.5161
```

```
Epoch Finish Time:  2019-12-11 17:22:39.072164
val Loss: 1.1154 Acc: 0.6118
Epoch Finish Time:  2019-12-11 17:22:51.289224

Epoch 9/349
----------
train Loss: 1.2704 Acc: 0.5483
Epoch Finish Time:  2019-12-11 17:25:19.054729
val Loss: 1.0791 Acc: 0.6083
Epoch Finish Time:  2019-12-11 17:25:31.286839

Epoch 10/349
----------
train Loss: 1.2108 Acc: 0.5695
Epoch Finish Time:  2019-12-11 17:27:59.681074
val Loss: 0.9515 Acc: 0.6661
Epoch Finish Time:  2019-12-11 17:28:11.886059

Epoch 11/349
----------
train Loss: 1.1540 Acc: 0.5897
Epoch Finish Time:  2019-12-11 17:30:40.085420
val Loss: 0.9170 Acc: 0.6838
Epoch Finish Time:  2019-12-11 17:30:52.229272

Epoch 12/349
----------
train Loss: 1.1068 Acc: 0.6097
Epoch Finish Time:  2019-12-11 17:33:20.616774
val Loss: 0.9451 Acc: 0.6753
Epoch Finish Time:  2019-12-11 17:33:32.792275

Epoch 13/349
----------
train Loss: 1.0648 Acc: 0.6262
Epoch Finish Time:  2019-12-11 17:36:00.924345
val Loss: 0.8217 Acc: 0.7166
Epoch Finish Time:  2019-12-11 17:36:13.044119

Epoch 14/349
----------
train Loss: 1.0238 Acc: 0.6388
Epoch Finish Time:  2019-12-11 17:38:41.357621
val Loss: 0.7852 Acc: 0.7301
Epoch Finish Time:  2019-12-11 17:38:53.671539

Epoch 15/349
----------
```

```
train Loss: 0.9959 Acc: 0.6509
Epoch Finish Time:  2019-12-11 17:41:21.929152
val Loss: 0.8741 Acc: 0.7273
Epoch Finish Time:  2019-12-11 17:41:34.023157

Epoch 16/349
----------
train Loss: 0.9664 Acc: 0.6606
Epoch Finish Time:  2019-12-11 17:44:02.432801
val Loss: 0.7904 Acc: 0.7326
Epoch Finish Time:  2019-12-11 17:44:14.546214

Epoch 17/349
----------
train Loss: 0.9360 Acc: 0.6721
Epoch Finish Time:  2019-12-11 17:46:42.794766
val Loss: 0.8149 Acc: 0.7377
Epoch Finish Time:  2019-12-11 17:46:54.992123

Epoch 18/349
----------
train Loss: 0.9136 Acc: 0.6781
Epoch Finish Time:  2019-12-11 17:49:23.388956
val Loss: 0.7509 Acc: 0.7550
Epoch Finish Time:  2019-12-11 17:49:35.604956

Epoch 19/349
----------
train Loss: 0.8885 Acc: 0.6883
Epoch Finish Time:  2019-12-11 17:52:03.759650
val Loss: 0.6932 Acc: 0.7689
Epoch Finish Time:  2019-12-11 17:52:15.951239

Epoch 20/349
----------
train Loss: 0.8661 Acc: 0.6979
Epoch Finish Time:  2019-12-11 17:54:44.457066
val Loss: 0.6890 Acc: 0.7663
Epoch Finish Time:  2019-12-11 17:54:56.691783

Epoch 21/349
----------
train Loss: 0.8419 Acc: 0.7074
Epoch Finish Time:  2019-12-11 17:57:24.626917
val Loss: 0.6919 Acc: 0.7777
Epoch Finish Time:  2019-12-11 17:57:37.450434

Epoch 22/349
```

```
----------
train Loss: 0.8255 Acc: 0.7131
Epoch Finish Time:  2019-12-11 18:00:05.615739
val Loss: 0.6450 Acc: 0.7986
Epoch Finish Time:  2019-12-11 18:00:17.886678

Epoch 23/349
----------
train Loss: 0.8034 Acc: 0.7187
Epoch Finish Time:  2019-12-11 18:02:46.189855
val Loss: 0.6093 Acc: 0.7993
Epoch Finish Time:  2019-12-11 18:02:58.353547

Epoch 24/349
----------
train Loss: 0.7739 Acc: 0.7316
Epoch Finish Time:  2019-12-11 18:05:26.281681
val Loss: 0.6614 Acc: 0.8024
Epoch Finish Time:  2019-12-11 18:05:38.403828

Epoch 25/349
----------
train Loss: 0.7607 Acc: 0.7346
Epoch Finish Time:  2019-12-11 18:08:06.688318
val Loss: 0.5865 Acc: 0.8079
Epoch Finish Time:  2019-12-11 18:08:18.772814

Epoch 26/349
----------
train Loss: 0.7433 Acc: 0.7407
Epoch Finish Time:  2019-12-11 18:10:47.111827
val Loss: 0.5443 Acc: 0.8113
Epoch Finish Time:  2019-12-11 18:11:00.612587

Epoch 27/349
----------
train Loss: 0.7332 Acc: 0.7441
Epoch Finish Time:  2019-12-11 18:13:28.735511
val Loss: 0.5247 Acc: 0.8199
Epoch Finish Time:  2019-12-11 18:13:41.034998

Epoch 28/349
----------
train Loss: 0.7131 Acc: 0.7532
Epoch Finish Time:  2019-12-11 18:16:09.266839
val Loss: 0.5410 Acc: 0.8191
Epoch Finish Time:  2019-12-11 18:16:21.445759
```

```
Epoch 29/349
----------
train Loss: 0.6966 Acc: 0.7565
Epoch Finish Time:  2019-12-11 18:18:49.644844
val Loss: 0.5452 Acc: 0.8149
Epoch Finish Time:  2019-12-11 18:19:01.873098

Epoch 30/349
----------
train Loss: 0.6885 Acc: 0.7596
Epoch Finish Time:  2019-12-11 18:21:30.109204
val Loss: 0.5358 Acc: 0.8264
Epoch Finish Time:  2019-12-11 18:21:42.337508

Epoch 31/349
----------
train Loss: 0.6721 Acc: 0.7656
Epoch Finish Time:  2019-12-11 18:24:10.630978
val Loss: 0.5194 Acc: 0.8267
Epoch Finish Time:  2019-12-11 18:24:22.916506

Epoch 32/349
----------
train Loss: 0.6597 Acc: 0.7696
Epoch Finish Time:  2019-12-11 18:26:52.063671
val Loss: 0.4814 Acc: 0.8413
Epoch Finish Time:  2019-12-11 18:27:04.385118

Epoch 33/349
----------
train Loss: 0.6452 Acc: 0.7753
Epoch Finish Time:  2019-12-11 18:29:32.935087
val Loss: 0.4728 Acc: 0.8474
Epoch Finish Time:  2019-12-11 18:29:45.289503

Epoch 34/349
----------
train Loss: 0.6321 Acc: 0.7801
Epoch Finish Time:  2019-12-11 18:32:13.902546
val Loss: 0.4625 Acc: 0.8478
Epoch Finish Time:  2019-12-11 18:32:26.472512

Epoch 35/349
----------
train Loss: 0.6211 Acc: 0.7836
Epoch Finish Time:  2019-12-11 18:34:54.857496
val Loss: 0.4654 Acc: 0.8512
Epoch Finish Time:  2019-12-11 18:35:07.171881
```

```
Epoch 36/349
----------
train Loss: 0.6124 Acc: 0.7860
Epoch Finish Time:  2019-12-11 18:37:35.945768
val Loss: 0.4693 Acc: 0.8447
Epoch Finish Time:  2019-12-11 18:37:48.259271

Epoch 37/349
----------
train Loss: 0.5991 Acc: 0.7891
Epoch Finish Time:  2019-12-11 18:40:17.922118
val Loss: 0.5938 Acc: 0.8515
Epoch Finish Time:  2019-12-11 18:40:30.345584

Epoch 38/349
----------
train Loss: 0.5852 Acc: 0.7960
Epoch Finish Time:  2019-12-11 18:42:58.820989
val Loss: 0.5585 Acc: 0.8452
Epoch Finish Time:  2019-12-11 18:43:11.132618

Epoch 39/349
----------
train Loss: 0.5760 Acc: 0.7989
Epoch Finish Time:  2019-12-11 18:45:39.884199
val Loss: 0.4471 Acc: 0.8589
Epoch Finish Time:  2019-12-11 18:45:52.120801

Epoch 40/349
----------
train Loss: 0.5670 Acc: 0.8021
Epoch Finish Time:  2019-12-11 18:48:20.946013
val Loss: 0.4334 Acc: 0.8591
Epoch Finish Time:  2019-12-11 18:48:33.278581

Epoch 41/349
----------
train Loss: 0.5618 Acc: 0.8042
Epoch Finish Time:  2019-12-11 18:51:01.946488
val Loss: 0.4248 Acc: 0.8575
Epoch Finish Time:  2019-12-11 18:51:14.251181

Epoch 42/349
----------
train Loss: 0.5488 Acc: 0.8086
Epoch Finish Time:  2019-12-11 18:53:43.025127
val Loss: 0.4575 Acc: 0.8573
```

```
Epoch Finish Time:  2019-12-11 18:53:55.442049

Epoch 43/349
----------
train Loss: 0.5409 Acc: 0.8125
Epoch Finish Time:  2019-12-11 18:56:23.925126
val Loss: 0.3943 Acc: 0.8715
Epoch Finish Time:  2019-12-11 18:56:36.054758

Epoch 44/349
----------
train Loss: 0.5282 Acc: 0.8169
Epoch Finish Time:  2019-12-11 18:59:04.542381
val Loss: 0.3928 Acc: 0.8701
Epoch Finish Time:  2019-12-11 18:59:16.900842

Epoch 45/349
----------
train Loss: 0.5266 Acc: 0.8164
Epoch Finish Time:  2019-12-11 19:01:45.458885
val Loss: 0.4139 Acc: 0.8664
Epoch Finish Time:  2019-12-11 19:01:57.839997

Epoch 46/349
----------
train Loss: 0.5140 Acc: 0.8206
Epoch Finish Time:  2019-12-11 19:04:26.414041
val Loss: 0.4216 Acc: 0.8755
Epoch Finish Time:  2019-12-11 19:04:38.770392

Epoch 47/349
----------
train Loss: 0.5066 Acc: 0.8238
Epoch Finish Time:  2019-12-11 19:07:07.702479
val Loss: 0.4471 Acc: 0.8714
Epoch Finish Time:  2019-12-11 19:07:20.916604

Epoch 48/349
----------
train Loss: 0.5112 Acc: 0.8228
Epoch Finish Time:  2019-12-11 19:09:49.605429
val Loss: 0.3987 Acc: 0.8726
Epoch Finish Time:  2019-12-11 19:10:02.036989

Epoch 49/349
----------
train Loss: 0.4912 Acc: 0.8288
Epoch Finish Time:  2019-12-11 19:12:30.444715
```

```
val Loss: 0.4132 Acc: 0.8756
Epoch Finish Time:  2019-12-11 19:12:42.789623

Epoch 50/349
----------
train Loss: 0.4903 Acc: 0.8302
Epoch Finish Time:  2019-12-11 19:15:11.429015
val Loss: 0.3857 Acc: 0.8771
Epoch Finish Time:  2019-12-11 19:15:23.768725

Epoch 51/349
----------
train Loss: 0.4882 Acc: 0.8293
Epoch Finish Time:  2019-12-11 19:17:52.346008
val Loss: 0.4366 Acc: 0.8715
Epoch Finish Time:  2019-12-11 19:18:04.749803

Epoch 52/349
----------
train Loss: 0.4765 Acc: 0.8336
Epoch Finish Time:  2019-12-11 19:20:33.129359
val Loss: 0.4295 Acc: 0.8843
Epoch Finish Time:  2019-12-11 19:20:45.492608

Epoch 53/349
----------
train Loss: 0.4713 Acc: 0.8371
Epoch Finish Time:  2019-12-11 19:23:14.018037
val Loss: 0.4494 Acc: 0.8768
Epoch Finish Time:  2019-12-11 19:23:26.355518

Epoch 54/349
----------
train Loss: 0.4656 Acc: 0.8372
Epoch Finish Time:  2019-12-11 19:25:54.685538
val Loss: 0.3544 Acc: 0.8857
Epoch Finish Time:  2019-12-11 19:26:07.032356

Epoch 55/349
----------
train Loss: 0.4589 Acc: 0.8402
Epoch Finish Time:  2019-12-11 19:28:35.748193
val Loss: 0.5248 Acc: 0.8811
Epoch Finish Time:  2019-12-11 19:28:48.084056

Epoch 56/349
----------
train Loss: 0.4592 Acc: 0.8400
```

```
Epoch Finish Time:  2019-12-11 19:31:16.586864
val Loss: 0.3590 Acc: 0.8831
Epoch Finish Time:  2019-12-11 19:31:28.987874

Epoch 57/349
----------
train Loss: 0.4455 Acc: 0.8446
Epoch Finish Time:  2019-12-11 19:33:57.504408
val Loss: 0.4406 Acc: 0.8847
Epoch Finish Time:  2019-12-11 19:34:09.916301

Epoch 58/349
----------
train Loss: 0.4454 Acc: 0.8453
Epoch Finish Time:  2019-12-11 19:36:39.037595
val Loss: 0.4302 Acc: 0.8822
Epoch Finish Time:  2019-12-11 19:36:51.377671

Epoch 59/349
----------
train Loss: 0.4366 Acc: 0.8473
Epoch Finish Time:  2019-12-11 19:39:20.005432
val Loss: 0.4603 Acc: 0.8797
Epoch Finish Time:  2019-12-11 19:39:32.358254

Epoch 60/349
----------
train Loss: 0.4338 Acc: 0.8481
Epoch Finish Time:  2019-12-11 19:42:01.166233
val Loss: 0.3644 Acc: 0.8834
Epoch Finish Time:  2019-12-11 19:42:13.514846

Epoch 61/349
----------
train Loss: 0.4316 Acc: 0.8502
Epoch Finish Time:  2019-12-11 19:44:42.217870
val Loss: 0.3611 Acc: 0.8858
Epoch Finish Time:  2019-12-11 19:44:54.481202

Epoch 62/349
----------
train Loss: 0.4252 Acc: 0.8509
Epoch Finish Time:  2019-12-11 19:47:23.211905
val Loss: 0.3958 Acc: 0.8824
Epoch Finish Time:  2019-12-11 19:47:35.603500

Epoch 63/349
----------
```

```
train Loss: 0.4152 Acc: 0.8564
Epoch Finish Time:  2019-12-11 19:50:04.402927
val Loss: 0.3380 Acc: 0.8973
Epoch Finish Time:  2019-12-11 19:50:17.258621

Epoch 64/349
----------
train Loss: 0.4163 Acc: 0.8555
Epoch Finish Time:  2019-12-11 19:52:45.767396
val Loss: 0.3344 Acc: 0.8942
Epoch Finish Time:  2019-12-11 19:52:58.209615

Epoch 65/349
----------
train Loss: 0.4100 Acc: 0.8552
Epoch Finish Time:  2019-12-11 19:55:26.763476
val Loss: 0.3701 Acc: 0.8878
Epoch Finish Time:  2019-12-11 19:55:39.063012

Epoch 66/349
----------
train Loss: 0.4158 Acc: 0.8553
Epoch Finish Time:  2019-12-11 19:58:07.791910
val Loss: 0.3484 Acc: 0.8935
Epoch Finish Time:  2019-12-11 19:58:20.374437

Epoch 67/349
----------
train Loss: 0.4014 Acc: 0.8617
Epoch Finish Time:  2019-12-11 20:00:48.704128
val Loss: 0.3489 Acc: 0.8931
Epoch Finish Time:  2019-12-11 20:01:01.056395

Epoch 68/349
----------
train Loss: 0.3981 Acc: 0.8600
Epoch Finish Time:  2019-12-11 20:03:29.657283
val Loss: 0.3335 Acc: 0.8965
Epoch Finish Time:  2019-12-11 20:03:43.162434

Epoch 69/349
----------
train Loss: 0.3961 Acc: 0.8613
Epoch Finish Time:  2019-12-11 20:06:11.813002
val Loss: 0.3439 Acc: 0.8948
Epoch Finish Time:  2019-12-11 20:06:24.213779

Epoch 70/349
```

```
----------
train Loss: 0.3879 Acc: 0.8646
Epoch Finish Time:  2019-12-11 20:08:52.972539
val Loss: 0.3781 Acc: 0.8940
Epoch Finish Time:  2019-12-11 20:09:05.291951

Epoch 71/349
----------
train Loss: 0.3860 Acc: 0.8658
Epoch Finish Time:  2019-12-11 20:11:33.938245
val Loss: 0.3519 Acc: 0.8906
Epoch Finish Time:  2019-12-11 20:11:46.334231

Epoch 72/349
----------
train Loss: 0.3857 Acc: 0.8632
Epoch Finish Time:  2019-12-11 20:14:15.087848
val Loss: 0.3842 Acc: 0.8909
Epoch Finish Time:  2019-12-11 20:14:27.493385

Epoch 73/349
----------
train Loss: 0.3784 Acc: 0.8674
Epoch Finish Time:  2019-12-11 20:16:56.061284
val Loss: 0.3728 Acc: 0.8920
Epoch Finish Time:  2019-12-11 20:17:08.496155

Epoch 74/349
----------
train Loss: 0.3689 Acc: 0.8704
Epoch Finish Time:  2019-12-11 20:19:37.515663
val Loss: 0.3505 Acc: 0.8945
Epoch Finish Time:  2019-12-11 20:19:49.931664

Epoch 75/349
----------
train Loss: 0.3720 Acc: 0.8688
Epoch Finish Time:  2019-12-11 20:22:18.481449
val Loss: 0.3656 Acc: 0.8913
Epoch Finish Time:  2019-12-11 20:22:30.840309

Epoch 76/349
----------
train Loss: 0.3645 Acc: 0.8721
Epoch Finish Time:  2019-12-11 20:24:59.217487
val Loss: 0.3567 Acc: 0.8958
Epoch Finish Time:  2019-12-11 20:25:11.569501
```

```
Epoch 77/349
----------
train Loss: 0.3663 Acc: 0.8732
Epoch Finish Time:  2019-12-11 20:27:40.165086
val Loss: 0.3806 Acc: 0.9004
Epoch Finish Time:  2019-12-11 20:27:52.419750

Epoch 78/349
----------
train Loss: 0.3642 Acc: 0.8738
Epoch Finish Time:  2019-12-11 20:30:21.036959
val Loss: 0.3337 Acc: 0.8995
Epoch Finish Time:  2019-12-11 20:30:33.451756

Epoch 79/349
----------
train Loss: 0.3545 Acc: 0.8763
Epoch Finish Time:  2019-12-11 20:33:02.944269
val Loss: 0.3412 Acc: 0.8970
Epoch Finish Time:  2019-12-11 20:33:15.256263

Epoch 80/349
----------
train Loss: 0.3482 Acc: 0.8789
Epoch Finish Time:  2019-12-11 20:35:43.398911
val Loss: 0.3889 Acc: 0.9022
Epoch Finish Time:  2019-12-11 20:35:55.688272

Epoch 81/349
----------
train Loss: 0.3506 Acc: 0.8793
Epoch Finish Time:  2019-12-11 20:38:24.425183
val Loss: 0.3272 Acc: 0.9024
Epoch Finish Time:  2019-12-11 20:38:36.699912

Epoch 82/349
----------
train Loss: 0.3463 Acc: 0.8787
Epoch Finish Time:  2019-12-11 20:41:04.966973
val Loss: 0.3370 Acc: 0.9012
Epoch Finish Time:  2019-12-11 20:41:17.201137

Epoch 83/349
----------
train Loss: 0.3421 Acc: 0.8795
Epoch Finish Time:  2019-12-11 20:43:45.633234
val Loss: 0.3687 Acc: 0.8968
Epoch Finish Time:  2019-12-11 20:43:57.949521
```

```
Epoch 84/349
----------
train Loss: 0.3429 Acc: 0.8831
Epoch Finish Time:  2019-12-11 20:46:26.199867
val Loss: 0.3273 Acc: 0.9046
Epoch Finish Time:  2019-12-11 20:46:38.643671

Epoch 85/349
----------
train Loss: 0.3449 Acc: 0.8801
Epoch Finish Time:  2019-12-11 20:49:07.179416
val Loss: 0.3145 Acc: 0.9043
Epoch Finish Time:  2019-12-11 20:49:19.536813

Epoch 86/349
----------
train Loss: 0.3417 Acc: 0.8816
Epoch Finish Time:  2019-12-11 20:51:48.186768
val Loss: 0.3129 Acc: 0.9038
Epoch Finish Time:  2019-12-11 20:52:00.610806

Epoch 87/349
----------
train Loss: 0.3353 Acc: 0.8831
Epoch Finish Time:  2019-12-11 20:54:29.041465
val Loss: 0.3286 Acc: 0.9031
Epoch Finish Time:  2019-12-11 20:54:41.443835

Epoch 88/349
----------
train Loss: 0.3265 Acc: 0.8860
Epoch Finish Time:  2019-12-11 20:57:10.043704
val Loss: 0.3466 Acc: 0.9024
Epoch Finish Time:  2019-12-11 20:57:22.399278

Epoch 89/349
----------
train Loss: 0.3228 Acc: 0.8875
Epoch Finish Time:  2019-12-11 20:59:50.737625
val Loss: 0.3347 Acc: 0.9048
Epoch Finish Time:  2019-12-11 21:00:03.879488

Epoch 90/349
----------
val Loss: 0.3550 Acc: 0.9013
Epoch Finish Time:  2019-12-11 21:02:44.523112
```

```
Epoch 91/349
----------
train Loss: 0.3258 Acc: 0.8875
Epoch Finish Time:  2019-12-11 21:05:13.244854
val Loss: 0.3411 Acc: 0.9012
Epoch Finish Time:  2019-12-11 21:05:25.704505

Epoch 92/349
----------
train Loss: 0.3247 Acc: 0.8861
Epoch Finish Time:  2019-12-11 21:07:54.356787
val Loss: 0.3283 Acc: 0.9013
Epoch Finish Time:  2019-12-11 21:08:06.799709

Epoch 93/349
----------
train Loss: 0.3167 Acc: 0.8888
Epoch Finish Time:  2019-12-11 21:10:35.571690
val Loss: 0.3675 Acc: 0.8992
Epoch Finish Time:  2019-12-11 21:10:47.930734

Epoch 94/349
----------
train Loss: 0.3194 Acc: 0.8888
Epoch Finish Time:  2019-12-11 21:13:16.684905
val Loss: 0.3482 Acc: 0.9021
Epoch Finish Time:  2019-12-11 21:13:29.185812

Epoch 95/349
----------
train Loss: 0.3167 Acc: 0.8912
Epoch Finish Time:  2019-12-11 21:15:57.920617
val Loss: 0.3545 Acc: 0.9044
Epoch Finish Time:  2019-12-11 21:16:10.282539

Epoch 96/349
----------
train Loss: 0.3065 Acc: 0.8939
Epoch Finish Time:  2019-12-11 21:18:39.165267
val Loss: 0.3512 Acc: 0.9057
Epoch Finish Time:  2019-12-11 21:18:51.434102

Epoch 97/349
----------
train Loss: 0.3109 Acc: 0.8904
Epoch Finish Time:  2019-12-11 21:21:19.925346
val Loss: 0.3528 Acc: 0.9055
Epoch Finish Time:  2019-12-11 21:21:32.245842
```

```
Epoch 98/349
----------
train Loss: 0.3064 Acc: 0.8931
Epoch Finish Time:  2019-12-11 21:24:00.834615
val Loss: 0.3840 Acc: 0.9030
Epoch Finish Time:  2019-12-11 21:24:13.207680

Epoch 99/349
----------
train Loss: 0.3050 Acc: 0.8945
Epoch Finish Time:  2019-12-11 21:26:41.734511
val Loss: 0.3988 Acc: 0.9017
Epoch Finish Time:  2019-12-11 21:26:54.054036

Epoch 100/349
----------
train Loss: 0.3074 Acc: 0.8938
Epoch Finish Time:  2019-12-11 21:29:23.195275
val Loss: 0.3397 Acc: 0.9068
Epoch Finish Time:  2019-12-11 21:29:35.576665

Epoch 101/349
----------
train Loss: 0.3008 Acc: 0.8953
Epoch Finish Time:  2019-12-11 21:32:04.405611
val Loss: 0.3602 Acc: 0.9054
Epoch Finish Time:  2019-12-11 21:32:16.861395

Epoch 102/349
----------
train Loss: 0.2987 Acc: 0.8971
Epoch Finish Time:  2019-12-11 21:34:45.491932
val Loss: 0.3461 Acc: 0.9048
Epoch Finish Time:  2019-12-11 21:34:57.915141

Epoch 103/349
----------
train Loss: 0.2995 Acc: 0.8960
Epoch Finish Time:  2019-12-11 21:37:26.459731
val Loss: 0.3897 Acc: 0.9072
Epoch Finish Time:  2019-12-11 21:37:38.814082

Epoch 104/349
----------
train Loss: 0.2953 Acc: 0.8980
Epoch Finish Time:  2019-12-11 21:40:07.393969
val Loss: 0.3673 Acc: 0.9071
```

```
Epoch Finish Time:  2019-12-11 21:40:19.738706

Epoch 105/349
----------
train Loss: 0.2918 Acc: 0.8983
Epoch Finish Time:  2019-12-11 21:42:49.532734
val Loss: 0.3588 Acc: 0.9047
Epoch Finish Time:  2019-12-11 21:43:01.901192

Epoch 106/349
----------
train Loss: 0.2931 Acc: 0.8976
Epoch Finish Time:  2019-12-11 21:45:30.583056
val Loss: 0.3730 Acc: 0.9066
Epoch Finish Time:  2019-12-11 21:45:42.833816

Epoch 107/349
----------
train Loss: 0.2928 Acc: 0.8980
Epoch Finish Time:  2019-12-11 21:48:11.505103
val Loss: 0.3625 Acc: 0.9065
Epoch Finish Time:  2019-12-11 21:48:23.831236

Epoch 108/349
----------
train Loss: 0.2910 Acc: 0.8985
Epoch Finish Time:  2019-12-11 21:50:52.508003
val Loss: 0.3734 Acc: 0.9055
Epoch Finish Time:  2019-12-11 21:51:04.804564

Epoch 109/349
----------
train Loss: 0.2898 Acc: 0.8988
Epoch Finish Time:  2019-12-11 21:53:33.324525
val Loss: 0.3628 Acc: 0.9016
Epoch Finish Time:  2019-12-11 21:53:45.577068

Epoch 110/349
----------
train Loss: 0.2858 Acc: 0.9008
Epoch Finish Time:  2019-12-11 21:56:14.346959
val Loss: 0.3745 Acc: 0.9064
Epoch Finish Time:  2019-12-11 21:56:27.179712

Epoch 111/349
----------
train Loss: 0.2847 Acc: 0.9010
Epoch Finish Time:  2019-12-11 21:58:55.418437
```

```
val Loss: 0.4926 Acc: 0.9072
Epoch Finish Time:  2019-12-11 21:59:07.804510

Epoch 112/349
----------
train Loss: 0.2801 Acc: 0.9016
Epoch Finish Time:  2019-12-11 22:01:36.269044
val Loss: 0.3286 Acc: 0.9111
Epoch Finish Time:  2019-12-11 22:01:48.688898

Epoch 113/349
----------
```

[12]:

```python
#  Pipe Break- reloading from checkpoint
# Model Def
m = models.resnet50()
m.fc = nn.Linear(2048, len(classes))
m = m.to(device)
criterion = nn.CrossEntropyLoss()

# Observe that only parameters of final layer are being optimized as
# opposed to before.
optimizer_conv = optim.SGD(m.parameters(), lr=0.001, momentum=0.9)

# Decay LR by a factor of 0.1 every 7 epochs
exp_lr_scheduler = lr_scheduler.StepLR(optimizer_conv, step_size=100, gamma=0.1)
#exp_lr_scheduler = ReduceLROnPlateau( optimizer_conv,patience=5,min_lr=0.5e-6)

checkpoint = torch.load(PATH)
m.load_state_dict(checkpoint['model_state_dict'])

optimizer_conv.load_state_dict(checkpoint['optimizer_state_dict'])
#exp_lr_scheduler.load_state_dict(checkpoint['optimizer_state_dict'])
epoch = checkpoint['epoch']
loss = checkpoint['val_loss']

print("Training Start Time: ", datetime.datetime.now(), "Epoch: ", str(epoch) )
m = train_model(m, criterion, optimizer_conv, exp_lr_scheduler,␣
 ↪num_epochs=350-(epoch+1))
```

```
Training Start Time:  2019-12-11 22:27:32.785435 Epoch:  121
Epoch 0/227
----------
train Loss: 0.2735 Acc: 0.9062
Epoch Finish Time:  2019-12-11 22:30:01.075008
val Loss: 0.3399 Acc: 0.9119
Epoch Finish Time:  2019-12-11 22:30:13.471261
```

```
Epoch 1/227
----------
train Loss: 0.2583 Acc: 0.9106
Epoch Finish Time:  2019-12-11 22:32:41.856042
val Loss: 0.3091 Acc: 0.9137
Epoch Finish Time:  2019-12-11 22:32:54.379867

Epoch 2/227
----------
train Loss: 0.2527 Acc: 0.9124
Epoch Finish Time:  2019-12-11 22:35:23.019365
val Loss: 0.3076 Acc: 0.9166
Epoch Finish Time:  2019-12-11 22:35:35.465019

Epoch 3/227
----------
train Loss: 0.2450 Acc: 0.9154
Epoch Finish Time:  2019-12-11 22:38:04.089098
val Loss: 0.3082 Acc: 0.9169
Epoch Finish Time:  2019-12-11 22:38:16.340043

Epoch 4/227
----------
train Loss: 0.2405 Acc: 0.9174
Epoch Finish Time:  2019-12-11 22:40:45.004221
val Loss: 0.3202 Acc: 0.9160
Epoch Finish Time:  2019-12-11 22:40:58.481369

Epoch 5/227
----------
train Loss: 0.2326 Acc: 0.9196
Epoch Finish Time:  2019-12-11 22:43:27.096721
val Loss: 0.3033 Acc: 0.9177
Epoch Finish Time:  2019-12-11 22:43:39.761191

Epoch 6/227
----------
train Loss: 0.2354 Acc: 0.9192
Epoch Finish Time:  2019-12-11 22:46:08.413311
val Loss: 0.2967 Acc: 0.9196
Epoch Finish Time:  2019-12-11 22:46:21.007886

Epoch 7/227
----------
train Loss: 0.2362 Acc: 0.9188
Epoch Finish Time:  2019-12-11 22:48:49.618508
val Loss: 0.3066 Acc: 0.9197
Epoch Finish Time:  2019-12-11 22:49:02.125116
```

```
Epoch 8/227
----------
train Loss: 0.2275 Acc: 0.9211
Epoch Finish Time:  2019-12-11 22:51:30.729317
val Loss: 0.3033 Acc: 0.9193
Epoch Finish Time:  2019-12-11 22:51:43.197769

Epoch 9/227
----------
train Loss: 0.2265 Acc: 0.9212
Epoch Finish Time:  2019-12-11 22:54:12.009268
val Loss: 0.3009 Acc: 0.9197
Epoch Finish Time:  2019-12-11 22:54:24.514688

Epoch 10/227
----------
```

```
       ␣
 ↪---------------------------------------------------------------------------

        KeyboardInterrupt                         Traceback (most recent call␣
 ↪last)

        <ipython-input-12-6e56cfd2731b> in <module>
         23
         24 print("Training Start Time: ", datetime.datetime.now(), "Epoch: ",␣
 ↪str(epoch) )
    ---> 25 m = train_model(m, criterion, optimizer_conv, exp_lr_scheduler,␣
 ↪num_epochs=350-(epoch+1))


        <ipython-input-10-c76245a1704b> in train_model(model, criterion,␣
 ↪optimizer, scheduler, num_epochs)
         39                     if phase == 'train':
         40                         loss.backward()
    ---> 41                         optimizer.step()
         42
         43                 # statistics


        /opt/anaconda3/lib/python3.7/site-packages/torch/optim/lr_scheduler.py␣
 ↪in wrapper(*args, **kwargs)
         34             def wrapper(*args, **kwargs):
         35                 opt._step_count += 1
    ---> 36                 return func(*args, **kwargs)
         37             wrapper._with_counter = True
```

```
     38             return wrapper


    /opt/anaconda3/lib/python3.7/site-packages/torch/optim/sgd.py in␣
↪step(self, closure)
     98                     else:
     99                         buf = param_state['momentum_buffer']
--> 100                         buf.mul_(momentum).add_(1 - dampening, d_p)
    101                     if nesterov:
    102                         d_p = d_p.add(momentum, buf)


    KeyboardInterrupt:
```

[17]:

```python
#  Pipe Break- reloading from checkpoint
# Model Def
m = models.resnet50()
m.fc = nn.Linear(2048, len(classes))
m = m.to(device)
criterion = nn.CrossEntropyLoss()

# Observe that only parameters of final layer are being optimized as
# opposed to before.
optimizer_conv = optim.SGD(m.parameters(), lr=0.001, momentum=0.9)

# Decay LR by a factor of 0.1 every 7 epochs
exp_lr_scheduler = lr_scheduler.StepLR(optimizer_conv, step_size=100, gamma=0.1)
#exp_lr_scheduler = ReduceLROnPlateau( optimizer_conv,patience=5,min_lr=0.5e-6)

checkpoint = torch.load(PATH)
m.load_state_dict(checkpoint['model_state_dict'])

optimizer_conv.load_state_dict(checkpoint['optimizer_state_dict'])
#exp_lr_scheduler.load_state_dict(checkpoint['optimizer_state_dict'])
epoch = checkpoint['epoch']
loss = checkpoint['val_loss']

print("Training Start Time: ", datetime.datetime.now(), "Epoch: ", str(epoch) )
m = train_model(m, criterion, optimizer_conv, exp_lr_scheduler,␣
↪num_epochs=350-(epoch+1))
```

```
Training Start Time:  2019-12-12 02:05:37.842983 Epoch:  9
Epoch 0/339
----------
train Loss: 0.2219 Acc: 0.9236
Epoch Finish Time:  2019-12-12 02:08:06.397568
val Loss: 0.2981 Acc: 0.9207
```

```
Epoch Finish Time:  2019-12-12 02:08:18.852337

Epoch 1/339
----------
train Loss: 0.2229 Acc: 0.9229
Epoch Finish Time:  2019-12-12 02:10:47.530429
val Loss: 0.3006 Acc: 0.9205
Epoch Finish Time:  2019-12-12 02:11:00.007218

Epoch 2/339
----------
train Loss: 0.2202 Acc: 0.9238
Epoch Finish Time:  2019-12-12 02:13:28.538922
val Loss: 0.2978 Acc: 0.9215
Epoch Finish Time:  2019-12-12 02:13:40.887699

Epoch 3/339
----------
train Loss: 0.2240 Acc: 0.9239
Epoch Finish Time:  2019-12-12 02:16:09.732234
val Loss: 0.3016 Acc: 0.9214
Epoch Finish Time:  2019-12-12 02:16:22.263576

Epoch 4/339
----------
```

```
       ␣
↪---------------------------------------------------------------------------

       KeyboardInterrupt                         Traceback (most recent call␣
↪last)

       <ipython-input-17-6e56cfd2731b> in <module>
        23
        24 print("Training Start Time: ", datetime.datetime.now(), "Epoch: ",␣
↪str(epoch) )
   ---> 25 m = train_model(m, criterion, optimizer_conv, exp_lr_scheduler,␣
↪num_epochs=350-(epoch+1))


       <ipython-input-16-c76245a1704b> in train_model(model, criterion,␣
↪optimizer, scheduler, num_epochs)
        39                     if phase == 'train':
        40                         loss.backward()
   ---> 41                         optimizer.step()
        42
        43                 # statistics
```

```
/opt/anaconda3/lib/python3.7/site-packages/torch/optim/lr_scheduler.py␣
↪in wrapper(*args, **kwargs)
     34             def wrapper(*args, **kwargs):
     35                 opt._step_count += 1
---> 36                 return func(*args, **kwargs)
     37             wrapper._with_counter = True
     38             return wrapper


/opt/anaconda3/lib/python3.7/site-packages/torch/optim/sgd.py in␣
↪step(self, closure)
     98                     else:
     99                         buf = param_state['momentum_buffer']
--> 100                         buf.mul_(momentum).add_(1 - dampening, d_p)
    101                     if nesterov:
    102                         d_p = d_p.add(momentum, buf)


KeyboardInterrupt:
```

[30]:
```
#model for evaluation
#Model Def
m = models.resnet50()
m.fc = nn.Linear(2048, len(classes))
m = m.to(device)

criterion = nn.CrossEntropyLoss()

# Observe that only parameters of final layer are being optimized as
# opposed to before.
optimizer_conv = optim.SGD(m.parameters(), lr=0.001, momentum=0.9)

# Decay LR by a factor of 0.1 every 7 epochs
exp_lr_scheduler = lr_scheduler.StepLR(optimizer_conv, step_size=100, gamma=0.1)

checkpoint = torch.load(PATH)
m.load_state_dict(checkpoint['model_state_dict'])
optimizer_conv.load_state_dict(checkpoint['optimizer_state_dict'])
#exp_lr_scheduler.load_state_dict(checkpoint['optimizer_state_dict'])
epoch = checkpoint['epoch']
loss = checkpoint['val_loss']
m.eval()
```

[30]:
```
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3),
bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1,
ceil_mode=False)
  (layer1): Sequential(
    (0): Bottleneck(
      (conv1): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (downsample): Sequential(
        (0): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): Bottleneck(
      (conv1): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (2): Bottleneck(
      (conv1): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
```

```
track_running_stats=True)
      (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
  )
  (layer2): Sequential(
    (0): Bottleneck(
      (conv1): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (downsample): Sequential(
        (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): Bottleneck(
      (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (2): Bottleneck(
      (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
```

```
      (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (3): Bottleneck(
      (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
  )
  (layer3): Sequential(
    (0): Bottleneck(
      (conv1): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (downsample): Sequential(
        (0): Conv2d(512, 1024, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): Bottleneck(
      (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
```

```
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (2): Bottleneck(
      (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (3): Bottleneck(
      (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (4): Bottleneck(
      (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (5): Bottleneck(
      (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
```

```
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
  )
  (layer4): Sequential(
    (0): Bottleneck(
      (conv1): Conv2d(1024, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (downsample): Sequential(
        (0): Conv2d(1024, 2048, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): Bottleneck(
      (conv1): Conv2d(2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (2): Bottleneck(
      (conv1): Conv2d(2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
```

```
1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc): Linear(in_features=2048, out_features=10, bias=True)
)
```

[31]:
```python
from PIL import Image
from torchvision import transforms

def eval_image( filepath ):
    input_image = Image.open(filepath )
    preprocess = transforms.Compose([
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.
 ↪225]),
    ])
    input_tensor = preprocess(input_image)
    input_batch = input_tensor.unsqueeze(0) # create a mini-batch as expected␣
 ↪by the model

    # move the input and model to GPU for speed if available
    if torch.cuda.is_available():
        input_batch = input_batch.to('cuda')
        m.to('cuda')

    with torch.no_grad():
        output = m(input_batch)
    # The output has unnormalized scores. To get probabilities, you can run a␣
 ↪softmax on it.
    #print(torch.nn.functional.softmax(output[0], dim=0))
    _, preds = torch.max(output, 1)
    return preds[0]
```

[32]:
```python
from os import listdir
classes_new = ('airplane', 'automobile', 'bird', 'cat',
           'deer', 'dog', 'frog', 'horse', 'ship', 'truck')
image_paths = [
    listdir("DSF_HW5_wild_images/%d_%s" % ( num, c_name ))
```

```
    for num, c_name in enumerate(classes_new)
]

correct = 0.0
total = 0.0
for actual_class,files in enumerate(image_paths):
    for image_filepath in files:
        fpath = "DSF_HW5_wild_images/%d_%s/" % ( actual_class,␣
↪classes_new[actual_class] )
        pred_label = eval_image( fpath + image_filepath )
        if pred_label == actual_class:
            correct += 1.0

        total += 1.0

print( "Wild Accuracy: ", correct / total )
```

```
Wild Accuracy:  0.98
```

[36]:
```
device
```

[36]:
```
device(type='cuda', index=0)
```