project_VGG

November 28, 2022

[2]:
```
import argparse
import os
import time
import shutil

import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
import torch.backends.cudnn as cudnn


import torchvision
import torchvision.transforms as transforms

from models import *


global best_prec
use_gpu = torch.cuda.is_available()
print('=> Building model...')



batch_size = 128
model_name = "VGG16_quant4bit"
model = VGG16_quant()
model.features[24] = QuantConv2d(256, 8, kernel_size=3, stride=1, padding=1,␣
 ↪bias=False)
model.features[25] = nn.BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True,␣
 ↪track_running_stats=True)
model.features[27] = QuantConv2d(8, 8, kernel_size=3, stride=1, padding=1,␣
 ↪bias=False)
model.features[28] = nn.Sequential()
model.features[30] = QuantConv2d(8, 512, kernel_size=3, stride=1, padding=1,␣
 ↪bias=False)
```

```
#model.features[24] = QuantConv2d(512, 512, kernel_size=3, stride=1, padding=1,␣
 ↪bias=False)
#model.features[26] = QuantConv2d(512, 512, kernel_size=3, stride=1, padding=1,␣
 ↪bias=False)
#model.features[28] = QuantConv2d(512, 512, kernel_size=3, stride=1, padding=1,␣
 ↪bias=False)
print(model)

normalize = transforms.Normalize(mean=[0.491, 0.482, 0.447], std=[0.247, 0.243,␣
 ↪0.262])


train_dataset = torchvision.datasets.CIFAR10(
    root='./data',
    train=True,
    download=True,
    transform=transforms.Compose([
        transforms.RandomCrop(32, padding=4),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        normalize,
    ]))
trainloader = torch.utils.data.DataLoader(train_dataset, batch_size=batch_size,␣
 ↪shuffle=True, num_workers=2)


test_dataset = torchvision.datasets.CIFAR10(
    root='./data',
    train=False,
    download=True,
    transform=transforms.Compose([
        transforms.ToTensor(),
        normalize,
    ]))

testloader = torch.utils.data.DataLoader(test_dataset, batch_size=batch_size,␣
 ↪shuffle=False, num_workers=2)


print_freq = 100 # every 100 batches, accuracy printed. Here, each batch␣
 ↪includes "batch_size" data points
# CIFAR10 has 50,000 training data, and 10,000 validation data.

def train(trainloader, model, criterion, optimizer, epoch):
    batch_time = AverageMeter()
    data_time = AverageMeter()
```

```
    losses = AverageMeter()
    top1 = AverageMeter()

    model.train()

    end = time.time()
    for i, (input, target) in enumerate(trainloader):
        # measure data loading time
        data_time.update(time.time() - end)

        input, target = input.cuda(), target.cuda()

        # compute output
        output = model(input)
        loss = criterion(output, target)

        # measure accuracy and record loss
        prec = accuracy(output, target)[0]
        losses.update(loss.item(), input.size(0))
        top1.update(prec.item(), input.size(0))

        # compute gradient and do SGD step
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        # measure elapsed time
        batch_time.update(time.time() - end)
        end = time.time()


        if i % print_freq == 0:
            print('Epoch: [{0}][{1}/{2}]\t'
                  'Time {batch_time.val:.3f} ({batch_time.avg:.3f})\t'
                  'Data {data_time.val:.3f} ({data_time.avg:.3f})\t'
                  'Loss {loss.val:.4f} ({loss.avg:.4f})\t'
                  'Prec {top1.val:.3f}% ({top1.avg:.3f}%)'.format(
                   epoch, i, len(trainloader), batch_time=batch_time,
                   data_time=data_time, loss=losses, top1=top1))



def validate(val_loader, model, criterion ):
    batch_time = AverageMeter()
    losses = AverageMeter()
    top1 = AverageMeter()
```

```
    # switch to evaluate mode
    model.eval()

    end = time.time()
    with torch.no_grad():
        for i, (input, target) in enumerate(val_loader):

            input, target = input.cuda(), target.cuda()

            # compute output
            output = model(input)
            loss = criterion(output, target)

            # measure accuracy and record loss
            prec = accuracy(output, target)[0]
            losses.update(loss.item(), input.size(0))
            top1.update(prec.item(), input.size(0))

            # measure elapsed time
            batch_time.update(time.time() - end)
            end = time.time()

            if i % print_freq == 0:  # This line shows how frequently print out␣
↪the status. e.g., i%5 => every 5 batch, prints out
                print('Test: [{0}/{1}]\t'
                  'Time {batch_time.val:.3f} ({batch_time.avg:.3f})\t'
                  'Loss {loss.val:.4f} ({loss.avg:.4f})\t'
                  'Prec {top1.val:.3f}% ({top1.avg:.3f}%)'.format(
                   i, len(val_loader), batch_time=batch_time, loss=losses,
                   top1=top1))

    print(' * Prec {top1.avg:.3f}% '.format(top1=top1))
    return top1.avg


def accuracy(output, target, topk=(1,)):
    """Computes the precision@k for the specified values of k"""
    maxk = max(topk)
    batch_size = target.size(0)

    _, pred = output.topk(maxk, 1, True, True)
    pred = pred.t()
    correct = pred.eq(target.view(1, -1).expand_as(pred))

    res = []
    for k in topk:
        correct_k = correct[:k].view(-1).float().sum(0)
```

```
        res.append(correct_k.mul_(100.0 / batch_size))
    return res


class AverageMeter(object):
    """Computes and stores the average and current value"""
    def __init__(self):
        self.reset()

    def reset(self):
        self.val = 0
        self.avg = 0
        self.sum = 0
        self.count = 0

    def update(self, val, n=1):
        self.val = val
        self.sum += val * n
        self.count += n
        self.avg = self.sum / self.count


def save_checkpoint(state, is_best, fdir):
    filepath = os.path.join(fdir, 'checkpoint.pth')
    torch.save(state, filepath)
    if is_best:
        shutil.copyfile(filepath, os.path.join(fdir, 'model_best.pth.tar'))


def adjust_learning_rate(optimizer, epoch):
    """For resnet, the lr starts from 0.1, and is divided by 10 at 80 and 120␣
 ↪epochs"""
    adjust_list = [60, 80]
    if epoch in adjust_list:
        for param_group in optimizer.param_groups:
            param_group['lr'] = param_group['lr'] * 0.1

#model = nn.DataParallel(model).cuda()
#all_params = checkpoint['state_dict']
#model.load_state_dict(all_params, strict=False)
#criterion = nn.CrossEntropyLoss().cuda()
#validate(testloader, model, criterion)
```

```
=> Building model…
VGG_quant(
  (features): Sequential(
    (0): QuantConv2d(
```

```
      3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
      (weight_quant): weight_quantize_fn()
    )
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): QuantConv2d(
      64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
      (weight_quant): weight_quantize_fn()
    )
    (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (5): ReLU(inplace=True)
    (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (7): QuantConv2d(
      64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
      (weight_quant): weight_quantize_fn()
    )
    (8): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (9): ReLU(inplace=True)
    (10): QuantConv2d(
      128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
      (weight_quant): weight_quantize_fn()
    )
    (11): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (12): ReLU(inplace=True)
    (13): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (14): QuantConv2d(
      128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
      (weight_quant): weight_quantize_fn()
    )
    (15): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (16): ReLU(inplace=True)
    (17): QuantConv2d(
      256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
      (weight_quant): weight_quantize_fn()
    )
    (18): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (19): ReLU(inplace=True)
    (20): QuantConv2d(
      256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
      (weight_quant): weight_quantize_fn()
```

```
    )
    (21): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (22): ReLU(inplace=True)
    (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (24): QuantConv2d(
      256, 8, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
      (weight_quant): weight_quantize_fn()
    )
    (25): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (26): ReLU(inplace=True)
    (27): QuantConv2d(
      8, 8, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
      (weight_quant): weight_quantize_fn()
    )
    (28): Sequential()
    (29): ReLU(inplace=True)
    (30): QuantConv2d(
      8, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
      (weight_quant): weight_quantize_fn()
    )
    (31): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (32): ReLU(inplace=True)
    (33): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (34): QuantConv2d(
      512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
      (weight_quant): weight_quantize_fn()
    )
    (35): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (36): ReLU(inplace=True)
    (37): QuantConv2d(
      512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
      (weight_quant): weight_quantize_fn()
    )
    (38): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (39): ReLU(inplace=True)
    (40): QuantConv2d(
      512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
      (weight_quant): weight_quantize_fn()
    )
    (41): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
```

```
    (42): ReLU(inplace=True)
    (43): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (44): AvgPool2d(kernel_size=1, stride=1, padding=0)
  )
  (classifier): Linear(in_features=512, out_features=10, bias=True)
)
Files already downloaded and verified
Files already downloaded and verified
```

[3]:
```python
# This cell won't be given, but students will complete the training

lr = 4e-2
weight_decay = 1e-4
epochs = 100
best_prec = 0

#model = nn.DataParallel(model).cuda()
model.cuda()
criterion = nn.CrossEntropyLoss().cuda()
optimizer = torch.optim.SGD(model.parameters(), lr=lr, momentum=0.9,␣
 ↪weight_decay=weight_decay)
#cudnn.benchmark = True

if not os.path.exists('result'):
    os.makedirs('result')
fdir = 'result/'+str(model_name)
if not os.path.exists(fdir):
    os.makedirs(fdir)


for epoch in range(0, epochs):
    adjust_learning_rate(optimizer, epoch)

    train(trainloader, model, criterion, optimizer, epoch)

    # evaluate on test set
    print("Validation starts")
    prec = validate(testloader, model, criterion)

    # remember best precision and save checkpoint
    is_best = prec > best_prec
    best_prec = max(prec,best_prec)
    print('best acc: {:1f}'.format(best_prec))
    save_checkpoint({
        'epoch': epoch + 1,
        'state_dict': model.state_dict(),
```

```
        'best_prec': best_prec,
        'optimizer': optimizer.state_dict(),
    }, is_best, fdir)
```

```
/opt/conda/lib/python3.9/site-packages/torch/nn/functional.py:718: UserWarning:
Named tensors and all their associated APIs are an experimental feature and
subject to change. Please do not use them for anything important until they are
released as stable. (Triggered internally at
/pytorch/c10/core/TensorImpl.h:1156.)
  return torch.max_pool2d(input, kernel_size, stride, padding, dilation,
ceil_mode)
Epoch: [0][0/391]       Time 0.409 (0.409)      Data 0.242 (0.242)      Loss
2.3846 (2.3846)    Prec 10.156% (10.156%)
Epoch: [0][100/391]     Time 0.055 (0.058)      Data 0.002 (0.004)      Loss
2.3539 (3.0093)    Prec 14.062% (11.889%)
Epoch: [0][200/391]     Time 0.054 (0.056)      Data 0.001 (0.003)      Loss
2.1668 (2.5918)    Prec 17.969% (14.385%)
Epoch: [0][300/391]     Time 0.054 (0.056)      Data 0.001 (0.002)      Loss
2.0692 (2.4084)    Prec 22.656% (16.323%)
Validation starts
Test: [0/79]    Time 0.241 (0.241)      Loss 2.0212 (2.0212)    Prec 24.219%
(24.219%)
 * Prec 22.950%
best acc: 22.950000
Epoch: [1][0/391]       Time 0.264 (0.264)      Data 0.212 (0.212)      Loss
1.7874 (1.7874)    Prec 28.906% (28.906%)
Epoch: [1][100/391]     Time 0.058 (0.059)      Data 0.002 (0.004)      Loss
1.9209 (1.9188)    Prec 21.094% (23.987%)
Epoch: [1][200/391]     Time 0.057 (0.057)      Data 0.002 (0.003)      Loss
1.9143 (1.9014)    Prec 26.562% (24.510%)
Epoch: [1][300/391]     Time 0.059 (0.057)      Data 0.002 (0.002)      Loss
1.8668 (1.8936)    Prec 27.344% (24.912%)
Validation starts
Test: [0/79]    Time 0.199 (0.199)      Loss 1.8372 (1.8372)    Prec 28.906%
(28.906%)
 * Prec 28.320%
best acc: 28.320000
Epoch: [2][0/391]       Time 0.279 (0.279)      Data 0.217 (0.217)      Loss
1.8545 (1.8545)    Prec 31.250% (31.250%)
Epoch: [2][100/391]     Time 0.055 (0.057)      Data 0.002 (0.004)      Loss
1.7258 (1.8012)    Prec 34.375% (30.701%)
Epoch: [2][200/391]     Time 0.057 (0.057)      Data 0.002 (0.003)      Loss
1.7187 (1.7903)    Prec 32.812% (30.597%)
Epoch: [2][300/391]     Time 0.055 (0.057)      Data 0.002 (0.002)      Loss
1.6987 (1.7731)    Prec 32.031% (31.554%)
Validation starts
Test: [0/79]    Time 0.244 (0.244)      Loss 1.6547 (1.6547)    Prec 39.062%
```

```
(39.062%)
 * Prec 33.040%
best acc: 33.040000
Epoch: [3][0/391]       Time 0.284 (0.284)      Data 0.230 (0.230)      Loss
1.7388 (1.7388)    Prec 36.719% (36.719%)
Epoch: [3][100/391]     Time 0.054 (0.059)      Data 0.001 (0.004)      Loss
1.6285 (1.6534)    Prec 36.719% (35.860%)
Epoch: [3][200/391]     Time 0.053 (0.057)      Data 0.002 (0.003)      Loss
1.5552 (1.6370)    Prec 42.188% (36.641%)
Epoch: [3][300/391]     Time 0.056 (0.056)      Data 0.002 (0.002)      Loss
1.5782 (1.6174)    Prec 42.188% (37.513%)
Validation starts
Test: [0/79]    Time 0.208 (0.208)      Loss 1.7196 (1.7196)    Prec 35.156%
(35.156%)
 * Prec 36.230%
best acc: 36.230000
Epoch: [4][0/391]       Time 0.251 (0.251)      Data 0.203 (0.203)      Loss
1.5380 (1.5380)    Prec 42.188% (42.188%)
Epoch: [4][100/391]     Time 0.055 (0.057)      Data 0.002 (0.004)      Loss
1.4363 (1.5182)    Prec 50.000% (42.404%)
Epoch: [4][200/391]     Time 0.055 (0.056)      Data 0.002 (0.003)      Loss
1.5221 (1.4972)    Prec 46.094% (43.412%)
Epoch: [4][300/391]     Time 0.055 (0.056)      Data 0.002 (0.002)      Loss
1.5234 (1.4787)    Prec 42.188% (44.417%)
Validation starts
Test: [0/79]    Time 0.233 (0.233)      Loss 1.3222 (1.3222)    Prec 45.312%
(45.312%)
 * Prec 46.650%
best acc: 46.650000
Epoch: [5][0/391]       Time 0.265 (0.265)      Data 0.221 (0.221)      Loss
1.4243 (1.4243)    Prec 46.094% (46.094%)
Epoch: [5][100/391]     Time 0.053 (0.057)      Data 0.002 (0.004)      Loss
1.2855 (1.3604)    Prec 57.031% (49.961%)
Epoch: [5][200/391]     Time 0.055 (0.056)      Data 0.002 (0.003)      Loss
1.2400 (1.3507)    Prec 56.250% (50.536%)
Epoch: [5][300/391]     Time 0.054 (0.056)      Data 0.002 (0.002)      Loss
1.1976 (1.3270)    Prec 57.812% (51.204%)
Validation starts
Test: [0/79]    Time 0.202 (0.202)      Loss 1.1099 (1.1099)    Prec 62.500%
(62.500%)
 * Prec 54.070%
best acc: 54.070000
Epoch: [6][0/391]       Time 0.265 (0.265)      Data 0.226 (0.226)      Loss
1.1275 (1.1275)    Prec 63.281% (63.281%)
Epoch: [6][100/391]     Time 0.053 (0.057)      Data 0.002 (0.004)      Loss
1.1724 (1.2075)    Prec 57.812% (56.915%)
Epoch: [6][200/391]     Time 0.056 (0.056)      Data 0.001 (0.003)      Loss
1.4105 (1.2049)    Prec 52.344% (56.713%)
```

```
Epoch: [6][300/391]     Time 0.053 (0.056)      Data 0.001 (0.002)      Loss
1.0673 (1.1899)    Prec 61.719% (57.247%)
Validation starts
Test: [0/79]    Time 0.226 (0.226)      Loss 1.0966 (1.0966)    Prec 63.281%
(63.281%)
 * Prec 56.990%
best acc: 56.990000
Epoch: [7][0/391]       Time 0.237 (0.237)      Data 0.200 (0.200)      Loss
1.3164 (1.3164)    Prec 56.250% (56.250%)
Epoch: [7][100/391]     Time 0.056 (0.057)      Data 0.001 (0.004)      Loss
1.1325 (1.1179)    Prec 54.688% (60.365%)
Epoch: [7][200/391]     Time 0.051 (0.055)      Data 0.001 (0.002)      Loss
0.9232 (1.1096)    Prec 67.188% (60.187%)
Epoch: [7][300/391]     Time 0.054 (0.055)      Data 0.002 (0.002)      Loss
1.0576 (1.0969)    Prec 57.031% (60.675%)
Validation starts
Test: [0/79]    Time 0.199 (0.199)      Loss 1.0309 (1.0309)    Prec 66.406%
(66.406%)
 * Prec 60.820%
best acc: 60.820000
Epoch: [8][0/391]       Time 0.267 (0.267)      Data 0.218 (0.218)      Loss
0.9442 (0.9442)    Prec 67.969% (67.969%)
Epoch: [8][100/391]     Time 0.054 (0.057)      Data 0.002 (0.004)      Loss
0.9704 (1.0039)    Prec 65.625% (63.993%)
Epoch: [8][200/391]     Time 0.053 (0.056)      Data 0.001 (0.003)      Loss
1.0403 (0.9964)    Prec 59.375% (64.486%)
Epoch: [8][300/391]     Time 0.053 (0.055)      Data 0.002 (0.002)      Loss
1.0532 (0.9900)    Prec 61.719% (64.626%)
Validation starts
Test: [0/79]    Time 0.208 (0.208)      Loss 0.9344 (0.9344)    Prec 67.188%
(67.188%)
 * Prec 65.700%
best acc: 65.700000
Epoch: [9][0/391]       Time 0.234 (0.234)      Data 0.194 (0.194)      Loss
0.8420 (0.8420)    Prec 69.531% (69.531%)
Epoch: [9][100/391]     Time 0.057 (0.057)      Data 0.002 (0.004)      Loss
0.8937 (0.9246)    Prec 70.312% (66.901%)
Epoch: [9][200/391]     Time 0.059 (0.056)      Data 0.002 (0.003)      Loss
1.1065 (0.9300)    Prec 64.844% (67.005%)
Epoch: [9][300/391]     Time 0.055 (0.056)      Data 0.002 (0.002)      Loss
0.8946 (0.9231)    Prec 66.406% (67.206%)
Validation starts
Test: [0/79]    Time 0.222 (0.222)      Loss 0.8512 (0.8512)    Prec 64.844%
(64.844%)
 * Prec 66.350%
best acc: 66.350000
Epoch: [10][0/391]      Time 0.305 (0.305)      Data 0.256 (0.256)      Loss
0.8171 (0.8171)    Prec 71.875% (71.875%)
```

```
Epoch: [10][100/391]    Time 0.054 (0.057)      Data 0.001 (0.004)      Loss
0.8730 (0.8658)    Prec 67.969% (69.539%)
Epoch: [10][200/391]    Time 0.052 (0.057)      Data 0.001 (0.003)      Loss
1.0068 (0.8641)    Prec 67.188% (69.648%)
Epoch: [10][300/391]    Time 0.053 (0.056)      Data 0.002 (0.003)      Loss
0.8638 (0.8611)    Prec 67.188% (69.708%)
Validation starts
Test: [0/79]    Time 0.222 (0.222)      Loss 0.9082 (0.9082)    Prec 69.531%
(69.531%)
 * Prec 66.760%
best acc: 66.760000
Epoch: [11][0/391]      Time 0.277 (0.277)      Data 0.234 (0.234)      Loss
0.8056 (0.8056)    Prec 74.219% (74.219%)
Epoch: [11][100/391]    Time 0.057 (0.058)      Data 0.001 (0.004)      Loss
0.7777 (0.8210)    Prec 71.094% (71.914%)
Epoch: [11][200/391]    Time 0.056 (0.057)      Data 0.002 (0.003)      Loss
0.8569 (0.8083)    Prec 71.094% (72.038%)
Epoch: [11][300/391]    Time 0.057 (0.056)      Data 0.002 (0.002)      Loss
0.9132 (0.8071)    Prec 66.406% (71.971%)
Validation starts
Test: [0/79]    Time 0.223 (0.223)      Loss 0.8094 (0.8094)    Prec 69.531%
(69.531%)
 * Prec 69.320%
best acc: 69.320000
Epoch: [12][0/391]      Time 0.235 (0.235)      Data 0.193 (0.193)      Loss
0.8251 (0.8251)    Prec 70.312% (70.312%)
Epoch: [12][100/391]    Time 0.057 (0.059)      Data 0.001 (0.004)      Loss
0.8530 (0.7700)    Prec 75.781% (73.414%)
Epoch: [12][200/391]    Time 0.056 (0.057)      Data 0.001 (0.003)      Loss
0.7038 (0.7606)    Prec 75.781% (73.453%)
Epoch: [12][300/391]    Time 0.056 (0.056)      Data 0.002 (0.002)      Loss
0.6465 (0.7634)    Prec 75.781% (73.360%)
Validation starts
Test: [0/79]    Time 0.214 (0.214)      Loss 0.7446 (0.7446)    Prec 75.000%
(75.000%)
 * Prec 72.120%
best acc: 72.120000
Epoch: [13][0/391]      Time 0.285 (0.285)      Data 0.246 (0.246)      Loss
0.7280 (0.7280)    Prec 74.219% (74.219%)
Epoch: [13][100/391]    Time 0.055 (0.058)      Data 0.001 (0.004)      Loss
0.7950 (0.7150)    Prec 68.750% (75.541%)
Epoch: [13][200/391]    Time 0.055 (0.057)      Data 0.001 (0.003)      Loss
0.5378 (0.7240)    Prec 79.688% (75.117%)
Epoch: [13][300/391]    Time 0.055 (0.056)      Data 0.001 (0.002)      Loss
0.6436 (0.7301)    Prec 78.125% (75.005%)
Validation starts
Test: [0/79]    Time 0.199 (0.199)      Loss 0.7461 (0.7461)    Prec 74.219%
(74.219%)
```

```
 * Prec 71.120%
best acc: 72.120000
Epoch: [14][0/391]      Time 0.265 (0.265)      Data 0.218 (0.218)      Loss
0.8804 (0.8804)    Prec 71.875% (71.875%)
Epoch: [14][100/391]    Time 0.050 (0.057)      Data 0.002 (0.004)      Loss
0.6579 (0.7025)    Prec 75.000% (76.153%)
Epoch: [14][200/391]    Time 0.055 (0.056)      Data 0.002 (0.003)      Loss
0.6763 (0.6977)    Prec 75.781% (76.275%)
Epoch: [14][300/391]    Time 0.054 (0.056)      Data 0.001 (0.002)      Loss
0.6387 (0.6911)    Prec 76.562% (76.498%)
Validation starts
Test: [0/79]    Time 0.207 (0.207)      Loss 0.7366 (0.7366)    Prec 72.656%
(72.656%)
 * Prec 75.280%
best acc: 75.280000
Epoch: [15][0/391]      Time 0.303 (0.303)      Data 0.254 (0.254)      Loss
0.5262 (0.5262)    Prec 80.469% (80.469%)
Epoch: [15][100/391]    Time 0.053 (0.058)      Data 0.002 (0.004)      Loss
0.5531 (0.6491)    Prec 82.812% (77.684%)
Epoch: [15][200/391]    Time 0.055 (0.056)      Data 0.001 (0.003)      Loss
0.6446 (0.6516)    Prec 82.812% (77.713%)
Epoch: [15][300/391]    Time 0.057 (0.056)      Data 0.001 (0.002)      Loss
0.6164 (0.6526)    Prec 76.562% (77.634%)
Validation starts
Test: [0/79]    Time 0.213 (0.213)      Loss 0.6141 (0.6141)    Prec 79.688%
(79.688%)
 * Prec 76.260%
best acc: 76.260000
Epoch: [16][0/391]      Time 0.288 (0.288)      Data 0.237 (0.237)      Loss
0.5688 (0.5688)    Prec 82.031% (82.031%)
Epoch: [16][100/391]    Time 0.057 (0.057)      Data 0.002 (0.004)      Loss
0.6380 (0.6152)    Prec 79.688% (79.138%)
Epoch: [16][200/391]    Time 0.055 (0.056)      Data 0.001 (0.003)      Loss
0.7263 (0.6175)    Prec 73.438% (78.910%)
Epoch: [16][300/391]    Time 0.055 (0.056)      Data 0.002 (0.002)      Loss
0.7878 (0.6209)    Prec 75.000% (78.880%)
Validation starts
Test: [0/79]    Time 0.224 (0.224)      Loss 0.6503 (0.6503)    Prec 79.688%
(79.688%)
 * Prec 76.210%
best acc: 76.260000
Epoch: [17][0/391]      Time 0.294 (0.294)      Data 0.247 (0.247)      Loss
0.5400 (0.5400)    Prec 80.469% (80.469%)
Epoch: [17][100/391]    Time 0.050 (0.058)      Data 0.002 (0.004)      Loss
0.4293 (0.5946)    Prec 83.594% (79.834%)
Epoch: [17][200/391]    Time 0.059 (0.056)      Data 0.002 (0.003)      Loss
0.6290 (0.5971)    Prec 77.344% (79.629%)
Epoch: [17][300/391]    Time 0.055 (0.056)      Data 0.002 (0.003)      Loss
```

```
0.6252 (0.5957)    Prec 78.125% (79.825%)
Validation starts
Test: [0/79]    Time 0.247 (0.247)      Loss 0.7067 (0.7067)    Prec 76.562%
(76.562%)
 * Prec 75.620%
best acc: 76.260000
Epoch: [18][0/391]      Time 0.283 (0.283)      Data 0.239 (0.239)      Loss
0.4576 (0.4576)    Prec 82.812% (82.812%)
Epoch: [18][100/391]    Time 0.053 (0.058)      Data 0.003 (0.004)      Loss
0.5142 (0.5806)    Prec 79.688% (80.159%)
Epoch: [18][200/391]    Time 0.054 (0.057)      Data 0.002 (0.003)      Loss
0.5655 (0.5779)    Prec 78.906% (80.461%)
Epoch: [18][300/391]    Time 0.055 (0.056)      Data 0.001 (0.002)      Loss
0.6351 (0.5763)    Prec 77.344% (80.477%)
Validation starts
Test: [0/79]    Time 0.199 (0.199)      Loss 0.6843 (0.6843)    Prec 81.250%
(81.250%)
 * Prec 75.640%
best acc: 76.260000
Epoch: [19][0/391]      Time 0.224 (0.224)      Data 0.176 (0.176)      Loss
0.4557 (0.4557)    Prec 83.594% (83.594%)
Epoch: [19][100/391]    Time 0.049 (0.057)      Data 0.001 (0.003)      Loss
0.6259 (0.5621)    Prec 74.219% (81.126%)
Epoch: [19][200/391]    Time 0.052 (0.056)      Data 0.002 (0.002)      Loss
0.5215 (0.5586)    Prec 80.469% (81.126%)
Epoch: [19][300/391]    Time 0.054 (0.056)      Data 0.001 (0.002)      Loss
0.4182 (0.5598)    Prec 85.938% (81.102%)
Validation starts
Test: [0/79]    Time 0.242 (0.242)      Loss 0.7021 (0.7021)    Prec 78.125%
(78.125%)
 * Prec 76.870%
best acc: 76.870000
Epoch: [20][0/391]      Time 0.252 (0.252)      Data 0.205 (0.205)      Loss
0.6467 (0.6467)    Prec 76.562% (76.562%)
Epoch: [20][100/391]    Time 0.055 (0.057)      Data 0.002 (0.004)      Loss
0.5915 (0.5278)    Prec 82.812% (82.379%)
Epoch: [20][200/391]    Time 0.056 (0.056)      Data 0.002 (0.003)      Loss
0.6274 (0.5279)    Prec 76.562% (82.183%)
Epoch: [20][300/391]    Time 0.059 (0.056)      Data 0.002 (0.002)      Loss
0.5563 (0.5255)    Prec 80.469% (82.143%)
Validation starts
Test: [0/79]    Time 0.227 (0.227)      Loss 0.5825 (0.5825)    Prec 78.906%
(78.906%)
 * Prec 76.480%
best acc: 76.870000
Epoch: [21][0/391]      Time 0.275 (0.275)      Data 0.226 (0.226)      Loss
0.5466 (0.5466)    Prec 80.469% (80.469%)
Epoch: [21][100/391]    Time 0.054 (0.057)      Data 0.001 (0.004)      Loss
```

```
0.5818 (0.5252)    Prec 77.344% (82.410%)
Epoch: [21][200/391]    Time 0.054 (0.056)      Data 0.001 (0.003)      Loss
0.4497 (0.5220)    Prec 84.375% (82.525%)
Epoch: [21][300/391]    Time 0.054 (0.056)      Data 0.001 (0.002)      Loss
0.6210 (0.5221)    Prec 85.938% (82.413%)
Validation starts
Test: [0/79]    Time 0.191 (0.191)      Loss 0.5223 (0.5223)    Prec 78.906%
(78.906%)
 * Prec 79.530%
best acc: 79.530000
Epoch: [22][0/391]      Time 0.279 (0.279)      Data 0.230 (0.230)      Loss
0.3922 (0.3922)    Prec 85.938% (85.938%)
Epoch: [22][100/391]    Time 0.055 (0.058)      Data 0.002 (0.004)      Loss
0.5122 (0.4855)    Prec 79.688% (83.362%)
Epoch: [22][200/391]    Time 0.057 (0.057)      Data 0.003 (0.003)      Loss
0.5561 (0.4878)    Prec 82.031% (83.442%)
Epoch: [22][300/391]    Time 0.053 (0.056)      Data 0.002 (0.003)      Loss
0.4787 (0.4915)    Prec 82.812% (83.467%)
Validation starts
Test: [0/79]    Time 0.215 (0.215)      Loss 0.6169 (0.6169)    Prec 78.906%
(78.906%)
 * Prec 80.050%
best acc: 80.050000
Epoch: [23][0/391]      Time 0.258 (0.258)      Data 0.208 (0.208)      Loss
0.4081 (0.4081)    Prec 87.500% (87.500%)
Epoch: [23][100/391]    Time 0.054 (0.057)      Data 0.001 (0.004)      Loss
0.5371 (0.4670)    Prec 82.812% (83.911%)
Epoch: [23][200/391]    Time 0.057 (0.057)      Data 0.002 (0.002)      Loss
0.4969 (0.4737)    Prec 81.250% (83.730%)
Epoch: [23][300/391]    Time 0.054 (0.056)      Data 0.002 (0.002)      Loss
0.4277 (0.4766)    Prec 83.594% (83.721%)
Validation starts
Test: [0/79]    Time 0.206 (0.206)      Loss 0.5653 (0.5653)    Prec 79.688%
(79.688%)
 * Prec 78.530%
best acc: 80.050000
Epoch: [24][0/391]      Time 0.261 (0.261)      Data 0.215 (0.215)      Loss
0.4436 (0.4436)    Prec 84.375% (84.375%)
Epoch: [24][100/391]    Time 0.055 (0.058)      Data 0.002 (0.004)      Loss
0.3637 (0.4698)    Prec 85.938% (84.019%)
Epoch: [24][200/391]    Time 0.056 (0.057)      Data 0.001 (0.003)      Loss
0.5716 (0.4693)    Prec 82.031% (84.173%)
Epoch: [24][300/391]    Time 0.054 (0.056)      Data 0.002 (0.002)      Loss
0.4931 (0.4676)    Prec 82.031% (84.253%)
Validation starts
Test: [0/79]    Time 0.227 (0.227)      Loss 0.7080 (0.7080)    Prec 75.000%
(75.000%)
 * Prec 78.370%
```

```
best acc: 80.050000
Epoch: [25][0/391]      Time 0.237 (0.237)      Data 0.190 (0.190)      Loss
0.4125 (0.4125)    Prec 82.812% (82.812%)
Epoch: [25][100/391]    Time 0.054 (0.057)      Data 0.002 (0.004)      Loss
0.4694 (0.4315)    Prec 84.375% (85.179%)
Epoch: [25][200/391]    Time 0.059 (0.056)      Data 0.001 (0.003)      Loss
0.5074 (0.4446)    Prec 82.031% (84.612%)
Epoch: [25][300/391]    Time 0.055 (0.056)      Data 0.002 (0.002)      Loss
0.5177 (0.4438)    Prec 82.031% (84.686%)
Validation starts
Test: [0/79]    Time 0.225 (0.225)      Loss 0.5876 (0.5876)    Prec 83.594%
(83.594%)
 * Prec 81.310%
best acc: 81.310000
Epoch: [26][0/391]      Time 0.288 (0.288)      Data 0.239 (0.239)      Loss
0.3989 (0.3989)    Prec 82.812% (82.812%)
Epoch: [26][100/391]    Time 0.055 (0.057)      Data 0.001 (0.004)      Loss
0.4005 (0.4191)    Prec 85.156% (85.729%)
Epoch: [26][200/391]    Time 0.057 (0.056)      Data 0.002 (0.003)      Loss
0.4420 (0.4265)    Prec 83.594% (85.510%)
Epoch: [26][300/391]    Time 0.056 (0.056)      Data 0.002 (0.002)      Loss
0.5334 (0.4308)    Prec 80.469% (85.379%)
Validation starts
Test: [0/79]    Time 0.221 (0.221)      Loss 0.5454 (0.5454)    Prec 80.469%
(80.469%)
 * Prec 80.820%
best acc: 81.310000
Epoch: [27][0/391]      Time 0.306 (0.306)      Data 0.258 (0.258)      Loss
0.2939 (0.2939)    Prec 92.188% (92.188%)
Epoch: [27][100/391]    Time 0.059 (0.058)      Data 0.002 (0.004)      Loss
0.5050 (0.3977)    Prec 82.031% (86.603%)
Epoch: [27][200/391]    Time 0.053 (0.056)      Data 0.001 (0.003)      Loss
0.5080 (0.4045)    Prec 82.031% (86.392%)
Epoch: [27][300/391]    Time 0.053 (0.056)      Data 0.001 (0.002)      Loss
0.5381 (0.4101)    Prec 85.938% (86.119%)
Validation starts
Test: [0/79]    Time 0.221 (0.221)      Loss 0.5349 (0.5349)    Prec 82.812%
(82.812%)
 * Prec 81.470%
best acc: 81.470000
Epoch: [28][0/391]      Time 0.240 (0.240)      Data 0.198 (0.198)      Loss
0.3404 (0.3404)    Prec 87.500% (87.500%)
Epoch: [28][100/391]    Time 0.057 (0.058)      Data 0.002 (0.004)      Loss
0.4359 (0.3837)    Prec 84.375% (87.152%)
Epoch: [28][200/391]    Time 0.055 (0.057)      Data 0.001 (0.003)      Loss
0.2885 (0.3963)    Prec 87.500% (86.524%)
Epoch: [28][300/391]    Time 0.056 (0.056)      Data 0.002 (0.002)      Loss
0.4454 (0.4002)    Prec 81.250% (86.342%)
```

```
Validation starts
Test: [0/79]    Time 0.218 (0.218)      Loss 0.4651 (0.4651)    Prec 85.938%
(85.938%)
 * Prec 80.890%
best acc: 81.470000
Epoch: [29][0/391]      Time 0.281 (0.281)      Data 0.235 (0.235)      Loss
0.3338 (0.3338)    Prec 89.062% (89.062%)
Epoch: [29][100/391]    Time 0.054 (0.059)      Data 0.004 (0.004)      Loss
0.4338 (0.3852)    Prec 85.156% (86.989%)
Epoch: [29][200/391]    Time 0.058 (0.058)      Data 0.001 (0.003)      Loss
0.3549 (0.3852)    Prec 88.281% (86.863%)
Epoch: [29][300/391]    Time 0.094 (0.060)      Data 0.002 (0.002)      Loss
0.4800 (0.3849)    Prec 82.812% (86.942%)
Validation starts
Test: [0/79]    Time 0.224 (0.224)      Loss 0.5590 (0.5590)    Prec 83.594%
(83.594%)
 * Prec 81.140%
best acc: 81.470000
Epoch: [30][0/391]      Time 0.267 (0.267)      Data 0.208 (0.208)      Loss
0.3628 (0.3628)    Prec 86.719% (86.719%)
Epoch: [30][100/391]    Time 0.085 (0.086)      Data 0.002 (0.004)      Loss
0.3686 (0.3610)    Prec 87.500% (87.833%)
Epoch: [30][200/391]    Time 0.083 (0.085)      Data 0.001 (0.003)      Loss
0.4295 (0.3745)    Prec 89.062% (87.181%)
Epoch: [30][300/391]    Time 0.085 (0.084)      Data 0.002 (0.002)      Loss
0.3262 (0.3766)    Prec 86.719% (87.256%)
Validation starts
Test: [0/79]    Time 0.231 (0.231)      Loss 0.4799 (0.4799)    Prec 82.031%
(82.031%)
 * Prec 80.690%
best acc: 81.470000
Epoch: [31][0/391]      Time 0.298 (0.298)      Data 0.245 (0.245)      Loss
0.2130 (0.2130)    Prec 93.750% (93.750%)
Epoch: [31][100/391]    Time 0.085 (0.087)      Data 0.002 (0.004)      Loss
0.4575 (0.3508)    Prec 82.812% (88.034%)
Epoch: [31][200/391]    Time 0.088 (0.085)      Data 0.001 (0.003)      Loss
0.2649 (0.3574)    Prec 92.188% (87.885%)
Epoch: [31][300/391]    Time 0.086 (0.085)      Data 0.002 (0.002)      Loss
0.2878 (0.3624)    Prec 89.062% (87.573%)
Validation starts
Test: [0/79]    Time 0.210 (0.210)      Loss 0.4317 (0.4317)    Prec 82.812%
(82.812%)
 * Prec 82.780%
best acc: 82.780000
Epoch: [32][0/391]      Time 0.303 (0.303)      Data 0.237 (0.237)      Loss
0.3235 (0.3235)    Prec 92.188% (92.188%)
Epoch: [32][100/391]    Time 0.088 (0.087)      Data 0.002 (0.004)      Loss
0.2945 (0.3494)    Prec 89.062% (88.475%)
```

```
Epoch: [32][200/391]    Time 0.083 (0.085)      Data 0.002 (0.003)      Loss
0.3845 (0.3524)    Prec 89.062% (88.172%)
Epoch: [32][300/391]    Time 0.084 (0.085)      Data 0.001 (0.003)      Loss
0.2849 (0.3535)    Prec 92.188% (88.097%)
Validation starts
Test: [0/79]    Time 0.196 (0.196)      Loss 0.5222 (0.5222)    Prec 82.031%
(82.031%)
 * Prec 83.850%
best acc: 83.850000
Epoch: [33][0/391]      Time 0.244 (0.244)      Data 0.184 (0.184)      Loss
0.4320 (0.4320)    Prec 85.156% (85.156%)
Epoch: [33][100/391]    Time 0.081 (0.086)      Data 0.002 (0.003)      Loss
0.2519 (0.3312)    Prec 92.969% (88.745%)
Epoch: [33][200/391]    Time 0.084 (0.085)      Data 0.002 (0.003)      Loss
0.4076 (0.3410)    Prec 87.500% (88.351%)
Epoch: [33][300/391]    Time 0.083 (0.085)      Data 0.002 (0.002)      Loss
0.3461 (0.3420)    Prec 86.719% (88.268%)
Validation starts
Test: [0/79]    Time 0.216 (0.216)      Loss 0.4439 (0.4439)    Prec 78.906%
(78.906%)
 * Prec 83.750%
best acc: 83.850000
Epoch: [34][0/391]      Time 0.308 (0.308)      Data 0.256 (0.256)      Loss
0.2288 (0.2288)    Prec 92.969% (92.969%)
Epoch: [34][100/391]    Time 0.085 (0.084)      Data 0.002 (0.004)      Loss
0.4197 (0.3319)    Prec 86.719% (88.420%)
Epoch: [34][200/391]    Time 0.083 (0.084)      Data 0.001 (0.003)      Loss
0.3448 (0.3298)    Prec 88.281% (88.647%)
Epoch: [34][300/391]    Time 0.086 (0.084)      Data 0.001 (0.002)      Loss
0.3279 (0.3310)    Prec 85.938% (88.655%)
Validation starts
Test: [0/79]    Time 0.213 (0.213)      Loss 0.5201 (0.5201)    Prec 83.594%
(83.594%)
 * Prec 84.030%
best acc: 84.030000
Epoch: [35][0/391]      Time 0.275 (0.275)      Data 0.206 (0.206)      Loss
0.2561 (0.2561)    Prec 92.969% (92.969%)
Epoch: [35][100/391]    Time 0.082 (0.085)      Data 0.002 (0.004)      Loss
0.3374 (0.3050)    Prec 89.844% (89.898%)
Epoch: [35][200/391]    Time 0.087 (0.085)      Data 0.002 (0.003)      Loss
0.3992 (0.3147)    Prec 88.281% (89.397%)
Epoch: [35][300/391]    Time 0.096 (0.085)      Data 0.002 (0.002)      Loss
0.2651 (0.3174)    Prec 91.406% (89.343%)
Validation starts
Test: [0/79]    Time 0.204 (0.204)      Loss 0.5272 (0.5272)    Prec 82.812%
(82.812%)
 * Prec 82.320%
best acc: 84.030000
```

```
Epoch: [36][0/391]      Time 0.318 (0.318)      Data 0.263 (0.263)      Loss
0.2699 (0.2699)    Prec 89.844% (89.844%)
Epoch: [36][100/391]    Time 0.086 (0.086)      Data 0.001 (0.004)      Loss
0.2328 (0.3033)    Prec 92.188% (89.782%)
Epoch: [36][200/391]    Time 0.087 (0.085)      Data 0.002 (0.003)      Loss
0.2970 (0.3127)    Prec 89.844% (89.533%)
Epoch: [36][300/391]    Time 0.084 (0.085)      Data 0.002 (0.003)      Loss
0.3068 (0.3175)    Prec 90.625% (89.325%)
Validation starts
Test: [0/79]    Time 0.214 (0.214)      Loss 0.4056 (0.4056)    Prec 86.719%
(86.719%)
 * Prec 84.740%
best acc: 84.740000
Epoch: [37][0/391]      Time 0.488 (0.488)      Data 0.437 (0.437)      Loss
0.2396 (0.2396)    Prec 91.406% (91.406%)
Epoch: [37][100/391]    Time 0.080 (0.087)      Data 0.002 (0.006)      Loss
0.3977 (0.3001)    Prec 88.281% (90.029%)
Epoch: [37][200/391]    Time 0.082 (0.086)      Data 0.002 (0.004)      Loss
0.2895 (0.3039)    Prec 90.625% (89.735%)
Epoch: [37][300/391]    Time 0.088 (0.086)      Data 0.002 (0.003)      Loss
0.1774 (0.3042)    Prec 90.625% (89.634%)
Validation starts
Test: [0/79]    Time 0.197 (0.197)      Loss 0.5233 (0.5233)    Prec 83.594%
(83.594%)
 * Prec 83.630%
best acc: 84.740000
Epoch: [38][0/391]      Time 0.266 (0.266)      Data 0.211 (0.211)      Loss
0.2214 (0.2214)    Prec 91.406% (91.406%)
Epoch: [38][100/391]    Time 0.086 (0.086)      Data 0.002 (0.004)      Loss
0.3786 (0.2863)    Prec 86.719% (90.447%)
Epoch: [38][200/391]    Time 0.094 (0.087)      Data 0.002 (0.003)      Loss
0.2646 (0.2891)    Prec 92.188% (90.213%)
Epoch: [38][300/391]    Time 0.085 (0.087)      Data 0.002 (0.002)      Loss
0.2502 (0.2944)    Prec 89.844% (89.961%)
Validation starts
Test: [0/79]    Time 0.198 (0.198)      Loss 0.4058 (0.4058)    Prec 85.156%
(85.156%)
 * Prec 84.050%
best acc: 84.740000
Epoch: [39][0/391]      Time 0.251 (0.251)      Data 0.203 (0.203)      Loss
0.1995 (0.1995)    Prec 92.188% (92.188%)
Epoch: [39][100/391]    Time 0.078 (0.088)      Data 0.002 (0.004)      Loss
0.1489 (0.2822)    Prec 94.531% (90.200%)
Epoch: [39][200/391]    Time 0.087 (0.086)      Data 0.002 (0.003)      Loss
0.2324 (0.2865)    Prec 90.625% (90.186%)
Epoch: [39][300/391]    Time 0.083 (0.086)      Data 0.002 (0.002)      Loss
0.1789 (0.2875)    Prec 93.750% (90.176%)
Validation starts
```

```
Test: [0/79]    Time 0.199 (0.199)      Loss 0.4611 (0.4611)    Prec 81.250%
(81.250%)
 * Prec 83.750%
best acc: 84.740000
Epoch: [40][0/391]      Time 0.272 (0.272)      Data 0.215 (0.215)      Loss
0.2962 (0.2962)    Prec 89.844% (89.844%)
Epoch: [40][100/391]    Time 0.082 (0.086)      Data 0.002 (0.004)      Loss
0.3583 (0.2710)    Prec 86.719% (90.934%)
Epoch: [40][200/391]    Time 0.087 (0.085)      Data 0.001 (0.003)      Loss
0.2743 (0.2735)    Prec 87.500% (90.831%)
Epoch: [40][300/391]    Time 0.093 (0.085)      Data 0.001 (0.002)      Loss
0.3527 (0.2804)    Prec 87.500% (90.472%)
Validation starts
Test: [0/79]    Time 0.204 (0.204)      Loss 0.3466 (0.3466)    Prec 91.406%
(91.406%)
 * Prec 84.450%
best acc: 84.740000
Epoch: [41][0/391]      Time 0.295 (0.295)      Data 0.233 (0.233)      Loss
0.2086 (0.2086)    Prec 94.531% (94.531%)
Epoch: [41][100/391]    Time 0.091 (0.087)      Data 0.002 (0.004)      Loss
0.3681 (0.2658)    Prec 85.938% (90.903%)
Epoch: [41][200/391]    Time 0.086 (0.086)      Data 0.001 (0.003)      Loss
0.2414 (0.2665)    Prec 91.406% (90.882%)
Epoch: [41][300/391]    Time 0.083 (0.085)      Data 0.002 (0.003)      Loss
0.1594 (0.2738)    Prec 93.750% (90.560%)
Validation starts
Test: [0/79]    Time 0.215 (0.215)      Loss 0.4078 (0.4078)    Prec 85.938%
(85.938%)
 * Prec 82.930%
best acc: 84.740000
Epoch: [42][0/391]      Time 0.278 (0.278)      Data 0.222 (0.222)      Loss
0.2143 (0.2143)    Prec 92.969% (92.969%)
Epoch: [42][100/391]    Time 0.083 (0.086)      Data 0.002 (0.004)      Loss
0.2645 (0.2724)    Prec 89.062% (90.586%)
Epoch: [42][200/391]    Time 0.083 (0.086)      Data 0.002 (0.003)      Loss
0.3462 (0.2672)    Prec 89.062% (90.804%)
Epoch: [42][300/391]    Time 0.086 (0.085)      Data 0.002 (0.002)      Loss
0.2292 (0.2676)    Prec 93.750% (90.908%)
Validation starts
Test: [0/79]    Time 0.223 (0.223)      Loss 0.3845 (0.3845)    Prec 88.281%
(88.281%)
 * Prec 84.530%
best acc: 84.740000
Epoch: [43][0/391]      Time 0.252 (0.252)      Data 0.199 (0.199)      Loss
0.3314 (0.3314)    Prec 88.281% (88.281%)
Epoch: [43][100/391]    Time 0.083 (0.088)      Data 0.002 (0.004)      Loss
0.2841 (0.2399)    Prec 89.844% (91.669%)
Epoch: [43][200/391]    Time 0.081 (0.086)      Data 0.001 (0.003)      Loss
```

```
0.2310 (0.2428)    Prec 89.062% (91.733%)
Epoch: [43][300/391]    Time 0.086 (0.086)      Data 0.002 (0.002)      Loss
0.3501 (0.2495)    Prec 87.500% (91.440%)
Validation starts
Test: [0/79]    Time 0.241 (0.241)      Loss 0.2829 (0.2829)    Prec 91.406%
(91.406%)
 * Prec 84.980%
best acc: 84.980000
Epoch: [44][0/391]      Time 0.300 (0.300)      Data 0.249 (0.249)      Loss
0.3200 (0.3200)    Prec 90.625% (90.625%)
Epoch: [44][100/391]    Time 0.086 (0.087)      Data 0.002 (0.004)      Loss
0.2673 (0.2512)    Prec 93.750% (91.290%)
Epoch: [44][200/391]    Time 0.089 (0.084)      Data 0.001 (0.003)      Loss
0.3070 (0.2516)    Prec 88.281% (91.332%)
Epoch: [44][300/391]    Time 0.099 (0.085)      Data 0.002 (0.003)      Loss
0.4193 (0.2491)    Prec 84.375% (91.323%)
Validation starts
Test: [0/79]    Time 0.207 (0.207)      Loss 0.4378 (0.4378)    Prec 88.281%
(88.281%)
 * Prec 83.860%
best acc: 84.980000
Epoch: [45][0/391]      Time 0.289 (0.289)      Data 0.234 (0.234)      Loss
0.2215 (0.2215)    Prec 92.188% (92.188%)
Epoch: [45][100/391]    Time 0.080 (0.087)      Data 0.002 (0.004)      Loss
0.2329 (0.2284)    Prec 90.625% (91.925%)
Epoch: [45][200/391]    Time 0.084 (0.086)      Data 0.001 (0.003)      Loss
0.1549 (0.2356)    Prec 94.531% (91.721%)
Epoch: [45][300/391]    Time 0.088 (0.086)      Data 0.002 (0.003)      Loss
0.2952 (0.2391)    Prec 92.188% (91.707%)
Validation starts
Test: [0/79]    Time 0.187 (0.187)      Loss 0.3883 (0.3883)    Prec 87.500%
(87.500%)
 * Prec 85.640%
best acc: 85.640000
Epoch: [46][0/391]      Time 0.245 (0.245)      Data 0.184 (0.184)      Loss
0.2375 (0.2375)    Prec 92.969% (92.969%)
Epoch: [46][100/391]    Time 0.082 (0.089)      Data 0.002 (0.004)      Loss
0.2633 (0.2214)    Prec 89.062% (92.396%)
Epoch: [46][200/391]    Time 0.080 (0.087)      Data 0.002 (0.003)      Loss
0.2156 (0.2318)    Prec 92.188% (91.970%)
Epoch: [46][300/391]    Time 0.093 (0.087)      Data 0.002 (0.002)      Loss
0.2348 (0.2356)    Prec 92.188% (91.964%)
Validation starts
Test: [0/79]    Time 0.221 (0.221)      Loss 0.4352 (0.4352)    Prec 85.938%
(85.938%)
 * Prec 84.150%
best acc: 85.640000
Epoch: [47][0/391]      Time 0.255 (0.255)      Data 0.200 (0.200)      Loss
```

```
0.1687 (0.1687)    Prec 94.531% (94.531%)
Epoch: [47][100/391]    Time 0.086 (0.085)      Data 0.002 (0.004)      Loss
0.2659 (0.2223)    Prec 88.281% (92.304%)
Epoch: [47][200/391]    Time 0.083 (0.085)      Data 0.002 (0.003)      Loss
0.1895 (0.2186)    Prec 93.750% (92.355%)
Epoch: [47][300/391]    Time 0.084 (0.085)      Data 0.002 (0.002)      Loss
0.2188 (0.2246)    Prec 89.062% (92.154%)
Validation starts
Test: [0/79]    Time 0.230 (0.230)      Loss 0.2883 (0.2883)    Prec 88.281%
(88.281%)
 * Prec 85.320%
best acc: 85.640000
Epoch: [48][0/391]      Time 0.271 (0.271)      Data 0.213 (0.213)      Loss
0.1736 (0.1736)    Prec 92.969% (92.969%)
Epoch: [48][100/391]    Time 0.088 (0.087)      Data 0.002 (0.004)      Loss
0.2648 (0.2020)    Prec 89.844% (92.938%)
Epoch: [48][200/391]    Time 0.084 (0.086)      Data 0.002 (0.003)      Loss
0.1610 (0.2131)    Prec 94.531% (92.673%)
Epoch: [48][300/391]    Time 0.086 (0.086)      Data 0.002 (0.002)      Loss
0.5157 (0.2177)    Prec 85.156% (92.572%)
Validation starts
Test: [0/79]    Time 0.183 (0.183)      Loss 0.2078 (0.2078)    Prec 92.969%
(92.969%)
 * Prec 85.960%
best acc: 85.960000
Epoch: [49][0/391]      Time 0.291 (0.291)      Data 0.241 (0.241)      Loss
0.2260 (0.2260)    Prec 91.406% (91.406%)
Epoch: [49][100/391]    Time 0.080 (0.088)      Data 0.002 (0.004)      Loss
0.2245 (0.2005)    Prec 94.531% (93.085%)
Epoch: [49][200/391]    Time 0.091 (0.087)      Data 0.001 (0.003)      Loss
0.1663 (0.2090)    Prec 92.969% (92.712%)
Epoch: [49][300/391]    Time 0.085 (0.086)      Data 0.002 (0.003)      Loss
0.1529 (0.2116)    Prec 93.750% (92.709%)
Validation starts
Test: [0/79]    Time 0.217 (0.217)      Loss 0.3850 (0.3850)    Prec 85.938%
(85.938%)
 * Prec 86.560%
best acc: 86.560000
Epoch: [50][0/391]      Time 0.251 (0.251)      Data 0.202 (0.202)      Loss
0.1274 (0.1274)    Prec 95.312% (95.312%)
Epoch: [50][100/391]    Time 0.090 (0.087)      Data 0.001 (0.004)      Loss
0.1647 (0.1916)    Prec 95.312% (93.301%)
Epoch: [50][200/391]    Time 0.082 (0.086)      Data 0.002 (0.003)      Loss
0.1429 (0.1998)    Prec 93.750% (93.008%)
Epoch: [50][300/391]    Time 0.086 (0.085)      Data 0.001 (0.002)      Loss
0.3169 (0.2044)    Prec 91.406% (92.893%)
Validation starts
Test: [0/79]    Time 0.207 (0.207)      Loss 0.3757 (0.3757)    Prec 89.062%
```

```
(89.062%)
 * Prec 86.050%
best acc: 86.560000
Epoch: [51][0/391]      Time 0.297 (0.297)      Data 0.242 (0.242)      Loss
0.2066 (0.2066)    Prec 92.188% (92.188%)
Epoch: [51][100/391]    Time 0.079 (0.088)      Data 0.001 (0.004)      Loss
0.2532 (0.1897)    Prec 92.188% (93.711%)
Epoch: [51][200/391]    Time 0.084 (0.086)      Data 0.002 (0.003)      Loss
0.1934 (0.1969)    Prec 92.969% (93.330%)
Epoch: [51][300/391]    Time 0.071 (0.085)      Data 0.002 (0.003)      Loss
0.1437 (0.2025)    Prec 95.312% (93.065%)
Validation starts
Test: [0/79]    Time 0.229 (0.229)      Loss 0.3423 (0.3423)    Prec 88.281%
(88.281%)
 * Prec 85.680%
best acc: 86.560000
Epoch: [52][0/391]      Time 0.283 (0.283)      Data 0.222 (0.222)      Loss
0.1355 (0.1355)    Prec 96.094% (96.094%)
Epoch: [52][100/391]    Time 0.080 (0.087)      Data 0.002 (0.004)      Loss
0.2304 (0.1949)    Prec 90.625% (93.232%)
Epoch: [52][200/391]    Time 0.084 (0.086)      Data 0.001 (0.003)      Loss
0.2352 (0.1993)    Prec 93.750% (93.132%)
Epoch: [52][300/391]    Time 0.085 (0.085)      Data 0.002 (0.003)      Loss
0.3639 (0.1980)    Prec 89.062% (93.192%)
Validation starts
Test: [0/79]    Time 0.245 (0.245)      Loss 0.2655 (0.2655)    Prec 86.719%
(86.719%)
 * Prec 86.720%
best acc: 86.720000
Epoch: [53][0/391]      Time 0.312 (0.312)      Data 0.248 (0.248)      Loss
0.2155 (0.2155)    Prec 89.844% (89.844%)
Epoch: [53][100/391]    Time 0.085 (0.087)      Data 0.002 (0.004)      Loss
0.2424 (0.1942)    Prec 91.406% (93.100%)
Epoch: [53][200/391]    Time 0.089 (0.086)      Data 0.002 (0.003)      Loss
0.1621 (0.1936)    Prec 91.406% (93.287%)
Epoch: [53][300/391]    Time 0.084 (0.085)      Data 0.001 (0.003)      Loss
0.1808 (0.1926)    Prec 95.312% (93.343%)
Validation starts
Test: [0/79]    Time 0.217 (0.217)      Loss 0.3102 (0.3102)    Prec 90.625%
(90.625%)
 * Prec 86.400%
best acc: 86.720000
Epoch: [54][0/391]      Time 0.256 (0.256)      Data 0.194 (0.194)      Loss
0.1932 (0.1932)    Prec 96.094% (96.094%)
Epoch: [54][100/391]    Time 0.085 (0.087)      Data 0.001 (0.004)      Loss
0.2602 (0.1905)    Prec 89.062% (93.502%)
Epoch: [54][200/391]    Time 0.061 (0.085)      Data 0.002 (0.003)      Loss
0.1179 (0.1908)    Prec 96.875% (93.435%)
```

```
Epoch: [54][300/391]    Time 0.076 (0.085)      Data 0.002 (0.002)      Loss
0.1359 (0.1891)    Prec 93.750% (93.498%)
Validation starts
Test: [0/79]    Time 0.240 (0.240)      Loss 0.3646 (0.3646)    Prec 87.500%
(87.500%)
 * Prec 87.160%
best acc: 87.160000
Epoch: [55][0/391]      Time 0.317 (0.317)      Data 0.258 (0.258)      Loss
0.3616 (0.3616)    Prec 87.500% (87.500%)
Epoch: [55][100/391]    Time 0.084 (0.087)      Data 0.001 (0.004)      Loss
0.1689 (0.1724)    Prec 94.531% (93.905%)
Epoch: [55][200/391]    Time 0.082 (0.084)      Data 0.002 (0.003)      Loss
0.2497 (0.1813)    Prec 93.750% (93.715%)
Epoch: [55][300/391]    Time 0.091 (0.084)      Data 0.002 (0.003)      Loss
0.0873 (0.1814)    Prec 96.875% (93.667%)
Validation starts
Test: [0/79]    Time 0.234 (0.234)      Loss 0.2730 (0.2730)    Prec 89.844%
(89.844%)
 * Prec 86.740%
best acc: 87.160000
Epoch: [56][0/391]      Time 0.312 (0.312)      Data 0.258 (0.258)      Loss
0.2027 (0.2027)    Prec 92.188% (92.188%)
Epoch: [56][100/391]    Time 0.089 (0.088)      Data 0.002 (0.004)      Loss
0.1953 (0.1612)    Prec 93.750% (94.593%)
Epoch: [56][200/391]    Time 0.084 (0.086)      Data 0.002 (0.003)      Loss
0.1783 (0.1723)    Prec 92.969% (94.232%)
Epoch: [56][300/391]    Time 0.083 (0.085)      Data 0.002 (0.003)      Loss
0.2307 (0.1757)    Prec 92.969% (94.025%)
Validation starts
Test: [0/79]    Time 0.222 (0.222)      Loss 0.2335 (0.2335)    Prec 91.406%
(91.406%)
 * Prec 86.070%
best acc: 87.160000
Epoch: [57][0/391]      Time 0.325 (0.325)      Data 0.271 (0.271)      Loss
0.1701 (0.1701)    Prec 95.312% (95.312%)
Epoch: [57][100/391]    Time 0.101 (0.086)      Data 0.002 (0.004)      Loss
0.1117 (0.1719)    Prec 95.312% (93.982%)
Epoch: [57][200/391]    Time 0.081 (0.085)      Data 0.001 (0.003)      Loss
0.2463 (0.1778)    Prec 91.406% (93.847%)
Epoch: [57][300/391]    Time 0.088 (0.085)      Data 0.002 (0.003)      Loss
0.1384 (0.1736)    Prec 94.531% (94.077%)
Validation starts
Test: [0/79]    Time 0.198 (0.198)      Loss 0.3165 (0.3165)    Prec 90.625%
(90.625%)
 * Prec 87.020%
best acc: 87.160000
Epoch: [58][0/391]      Time 0.276 (0.276)      Data 0.217 (0.217)      Loss
0.0873 (0.0873)    Prec 96.875% (96.875%)
```

```
Epoch: [58][100/391]    Time 0.085 (0.086)      Data 0.001 (0.004)      Loss
0.1454 (0.1604)    Prec 96.094% (94.485%)
Epoch: [58][200/391]    Time 0.091 (0.086)      Data 0.001 (0.003)      Loss
0.1078 (0.1633)    Prec 95.312% (94.376%)
Epoch: [58][300/391]    Time 0.086 (0.086)      Data 0.001 (0.002)      Loss
0.1460 (0.1675)    Prec 94.531% (94.235%)
Validation starts
Test: [0/79]    Time 0.214 (0.214)      Loss 0.3369 (0.3369)    Prec 91.406%
(91.406%)
 * Prec 86.580%
best acc: 87.160000
Epoch: [59][0/391]      Time 0.277 (0.277)      Data 0.222 (0.222)      Loss
0.1499 (0.1499)    Prec 94.531% (94.531%)
Epoch: [59][100/391]    Time 0.085 (0.086)      Data 0.002 (0.004)      Loss
0.1002 (0.1557)    Prec 96.875% (94.392%)
Epoch: [59][200/391]    Time 0.088 (0.085)      Data 0.002 (0.003)      Loss
0.1691 (0.1596)    Prec 94.531% (94.356%)
Epoch: [59][300/391]    Time 0.085 (0.085)      Data 0.001 (0.002)      Loss
0.1787 (0.1653)    Prec 92.969% (94.129%)
Validation starts
Test: [0/79]    Time 0.222 (0.222)      Loss 0.3773 (0.3773)    Prec 86.719%
(86.719%)
 * Prec 86.730%
best acc: 87.160000
Epoch: [60][0/391]      Time 0.266 (0.266)      Data 0.219 (0.219)      Loss
0.2148 (0.2148)    Prec 91.406% (91.406%)
Epoch: [60][100/391]    Time 0.086 (0.085)      Data 0.002 (0.004)      Loss
0.1294 (0.1230)    Prec 95.312% (95.746%)
Epoch: [60][200/391]    Time 0.081 (0.085)      Data 0.002 (0.003)      Loss
0.0632 (0.1114)    Prec 98.438% (96.199%)
Epoch: [60][300/391]    Time 0.082 (0.085)      Data 0.002 (0.002)      Loss
0.0733 (0.1062)    Prec 97.656% (96.377%)
Validation starts
Test: [0/79]    Time 0.252 (0.252)      Loss 0.2644 (0.2644)    Prec 91.406%
(91.406%)
 * Prec 89.740%
best acc: 89.740000
Epoch: [61][0/391]      Time 0.382 (0.382)      Data 0.326 (0.326)      Loss
0.1032 (0.1032)    Prec 96.875% (96.875%)
Epoch: [61][100/391]    Time 0.087 (0.089)      Data 0.002 (0.005)      Loss
0.1212 (0.0823)    Prec 92.969% (97.200%)
Epoch: [61][200/391]    Time 0.083 (0.087)      Data 0.002 (0.003)      Loss
0.0938 (0.0836)    Prec 96.094% (97.256%)
Epoch: [61][300/391]    Time 0.084 (0.086)      Data 0.002 (0.003)      Loss
0.0575 (0.0821)    Prec 97.656% (97.282%)
Validation starts
Test: [0/79]    Time 0.192 (0.192)      Loss 0.1908 (0.1908)    Prec 94.531%
(94.531%)
```

```
 * Prec 89.920%
best acc: 89.920000
Epoch: [62][0/391]      Time 0.311 (0.311)      Data 0.252 (0.252)      Loss
0.0636 (0.0636)    Prec 97.656% (97.656%)
Epoch: [62][100/391]    Time 0.081 (0.087)      Data 0.001 (0.004)      Loss
0.0622 (0.0701)    Prec 97.656% (97.726%)
Epoch: [62][200/391]    Time 0.083 (0.086)      Data 0.002 (0.003)      Loss
0.0758 (0.0756)    Prec 96.875% (97.520%)
Epoch: [62][300/391]    Time 0.083 (0.086)      Data 0.001 (0.002)      Loss
0.0882 (0.0751)    Prec 97.656% (97.498%)
Validation starts
Test: [0/79]    Time 0.217 (0.217)      Loss 0.2207 (0.2207)    Prec 93.750%
(93.750%)
 * Prec 90.030%
best acc: 90.030000
Epoch: [63][0/391]      Time 0.265 (0.265)      Data 0.215 (0.215)      Loss
0.1108 (0.1108)    Prec 95.312% (95.312%)
Epoch: [63][100/391]    Time 0.078 (0.087)      Data 0.002 (0.004)      Loss
0.0668 (0.0713)    Prec 98.438% (97.486%)
Epoch: [63][200/391]    Time 0.083 (0.086)      Data 0.002 (0.003)      Loss
0.1139 (0.0679)    Prec 96.094% (97.617%)
Epoch: [63][300/391]    Time 0.088 (0.086)      Data 0.002 (0.002)      Loss
0.1039 (0.0685)    Prec 96.094% (97.674%)
Validation starts
Test: [0/79]    Time 0.223 (0.223)      Loss 0.2767 (0.2767)    Prec 92.969%
(92.969%)
 * Prec 89.890%
best acc: 90.030000
Epoch: [64][0/391]      Time 0.258 (0.258)      Data 0.197 (0.197)      Loss
0.0223 (0.0223)    Prec 100.000% (100.000%)
Epoch: [64][100/391]    Time 0.084 (0.087)      Data 0.002 (0.004)      Loss
0.0921 (0.0614)    Prec 97.656% (98.012%)
Epoch: [64][200/391]    Time 0.091 (0.086)      Data 0.002 (0.003)      Loss
0.0845 (0.0639)    Prec 98.438% (97.921%)
Epoch: [64][300/391]    Time 0.087 (0.085)      Data 0.002 (0.002)      Loss
0.1252 (0.0626)    Prec 95.312% (97.937%)
Validation starts
Test: [0/79]    Time 0.203 (0.203)      Loss 0.2374 (0.2374)    Prec 92.969%
(92.969%)
 * Prec 90.190%
best acc: 90.190000
Epoch: [65][0/391]      Time 0.260 (0.260)      Data 0.204 (0.204)      Loss
0.1281 (0.1281)    Prec 96.875% (96.875%)
Epoch: [65][100/391]    Time 0.078 (0.086)      Data 0.001 (0.004)      Loss
0.0550 (0.0551)    Prec 96.875% (98.028%)
Epoch: [65][200/391]    Time 0.083 (0.085)      Data 0.002 (0.003)      Loss
0.0596 (0.0572)    Prec 98.438% (98.018%)
Epoch: [65][300/391]    Time 0.087 (0.084)      Data 0.001 (0.002)      Loss
```

```
0.0709 (0.0583)    Prec 98.438% (97.999%)
Validation starts
Test: [0/79]    Time 0.208 (0.208)      Loss 0.2432 (0.2432)    Prec 93.750%
(93.750%)
 * Prec 90.320%
best acc: 90.320000
Epoch: [66][0/391]      Time 0.257 (0.257)      Data 0.194 (0.194)      Loss
0.0366 (0.0366)    Prec 99.219% (99.219%)
Epoch: [66][100/391]    Time 0.079 (0.087)      Data 0.001 (0.004)      Loss
0.0628 (0.0557)    Prec 97.656% (98.051%)
Epoch: [66][200/391]    Time 0.085 (0.086)      Data 0.002 (0.003)      Loss
0.0366 (0.0568)    Prec 99.219% (98.084%)
Epoch: [66][300/391]    Time 0.085 (0.085)      Data 0.002 (0.002)      Loss
0.1543 (0.0571)    Prec 95.312% (98.030%)
Validation starts
Test: [0/79]    Time 0.197 (0.197)      Loss 0.2487 (0.2487)    Prec 92.969%
(92.969%)
 * Prec 90.230%
best acc: 90.320000
Epoch: [67][0/391]      Time 0.327 (0.327)      Data 0.262 (0.262)      Loss
0.0245 (0.0245)    Prec 99.219% (99.219%)
Epoch: [67][100/391]    Time 0.083 (0.087)      Data 0.002 (0.004)      Loss
0.0596 (0.0498)    Prec 97.656% (98.345%)
Epoch: [67][200/391]    Time 0.087 (0.085)      Data 0.003 (0.003)      Loss
0.0697 (0.0531)    Prec 96.094% (98.173%)
Epoch: [67][300/391]    Time 0.083 (0.085)      Data 0.002 (0.003)      Loss
0.1739 (0.0548)    Prec 92.969% (98.105%)
Validation starts
Test: [0/79]    Time 0.260 (0.260)      Loss 0.2932 (0.2932)    Prec 90.625%
(90.625%)
 * Prec 90.140%
best acc: 90.320000
Epoch: [68][0/391]      Time 0.267 (0.267)      Data 0.210 (0.210)      Loss
0.0750 (0.0750)    Prec 98.438% (98.438%)
Epoch: [68][100/391]    Time 0.085 (0.087)      Data 0.001 (0.004)      Loss
0.0912 (0.0481)    Prec 96.094% (98.407%)
Epoch: [68][200/391]    Time 0.087 (0.085)      Data 0.002 (0.003)      Loss
0.0345 (0.0487)    Prec 99.219% (98.321%)
Epoch: [68][300/391]    Time 0.081 (0.085)      Data 0.002 (0.002)      Loss
0.0598 (0.0517)    Prec 97.656% (98.269%)
Validation starts
Test: [0/79]    Time 0.206 (0.206)      Loss 0.2167 (0.2167)    Prec 92.188%
(92.188%)
 * Prec 90.320%
best acc: 90.320000
Epoch: [69][0/391]      Time 0.270 (0.270)      Data 0.207 (0.207)      Loss
0.0402 (0.0402)    Prec 98.438% (98.438%)
Epoch: [69][100/391]    Time 0.083 (0.086)      Data 0.002 (0.004)      Loss
```

```
0.0168 (0.0467)    Prec 99.219% (98.445%)
Epoch: [69][200/391]    Time 0.085 (0.085)      Data 0.002 (0.003)      Loss
0.0418 (0.0449)    Prec 98.438% (98.492%)
Epoch: [69][300/391]    Time 0.084 (0.085)      Data 0.001 (0.002)      Loss
0.0356 (0.0461)    Prec 98.438% (98.419%)
Validation starts
Test: [0/79]    Time 0.217 (0.217)      Loss 0.2400 (0.2400)    Prec 93.750%
(93.750%)
 * Prec 90.170%
best acc: 90.320000
Epoch: [70][0/391]      Time 0.297 (0.297)      Data 0.237 (0.237)      Loss
0.1162 (0.1162)    Prec 97.656% (97.656%)
Epoch: [70][100/391]    Time 0.057 (0.086)      Data 0.002 (0.004)      Loss
0.0809 (0.0477)    Prec 97.656% (98.368%)
Epoch: [70][200/391]    Time 0.083 (0.085)      Data 0.002 (0.003)      Loss
0.0158 (0.0476)    Prec 99.219% (98.403%)
Epoch: [70][300/391]    Time 0.082 (0.085)      Data 0.002 (0.003)      Loss
0.0632 (0.0483)    Prec 96.875% (98.409%)
Validation starts
Test: [0/79]    Time 0.213 (0.213)      Loss 0.2203 (0.2203)    Prec 92.188%
(92.188%)
 * Prec 90.280%
best acc: 90.320000
Epoch: [71][0/391]      Time 0.252 (0.252)      Data 0.196 (0.196)      Loss
0.0408 (0.0408)    Prec 98.438% (98.438%)
Epoch: [71][100/391]    Time 0.084 (0.085)      Data 0.002 (0.004)      Loss
0.0838 (0.0436)    Prec 95.312% (98.422%)
Epoch: [71][200/391]    Time 0.087 (0.085)      Data 0.002 (0.003)      Loss
0.0175 (0.0432)    Prec 100.000% (98.438%)
Epoch: [71][300/391]    Time 0.086 (0.085)      Data 0.002 (0.002)      Loss
0.0518 (0.0455)    Prec 97.656% (98.383%)
Validation starts
Test: [0/79]    Time 0.221 (0.221)      Loss 0.3480 (0.3480)    Prec 91.406%
(91.406%)
 * Prec 90.110%
best acc: 90.320000
Epoch: [72][0/391]      Time 0.239 (0.239)      Data 0.183 (0.183)      Loss
0.1151 (0.1151)    Prec 96.094% (96.094%)
Epoch: [72][100/391]    Time 0.083 (0.086)      Data 0.002 (0.004)      Loss
0.0951 (0.0467)    Prec 96.875% (98.360%)
Epoch: [72][200/391]    Time 0.083 (0.086)      Data 0.002 (0.003)      Loss
0.0630 (0.0468)    Prec 97.656% (98.344%)
Epoch: [72][300/391]    Time 0.085 (0.085)      Data 0.002 (0.002)      Loss
0.0246 (0.0468)    Prec 100.000% (98.375%)
Validation starts
Test: [0/79]    Time 0.219 (0.219)      Loss 0.3025 (0.3025)    Prec 90.625%
(90.625%)
 * Prec 90.070%
```

```
best acc: 90.320000
Epoch: [73][0/391]      Time 0.281 (0.281)      Data 0.238 (0.238)      Loss
0.0349 (0.0349)    Prec 98.438% (98.438%)
Epoch: [73][100/391]    Time 0.086 (0.086)      Data 0.002 (0.004)      Loss
0.0392 (0.0400)    Prec 98.438% (98.554%)
Epoch: [73][200/391]    Time 0.086 (0.086)      Data 0.002 (0.003)      Loss
0.0178 (0.0447)    Prec 99.219% (98.438%)
Epoch: [73][300/391]    Time 0.087 (0.085)      Data 0.001 (0.002)      Loss
0.0293 (0.0447)    Prec 100.000% (98.448%)
Validation starts
Test: [0/79]    Time 0.219 (0.219)      Loss 0.2425 (0.2425)    Prec 94.531%
(94.531%)
 * Prec 90.020%
best acc: 90.320000
Epoch: [74][0/391]      Time 0.395 (0.395)      Data 0.326 (0.326)      Loss
0.0420 (0.0420)    Prec 98.438% (98.438%)
Epoch: [74][100/391]    Time 0.084 (0.088)      Data 0.001 (0.005)      Loss
0.0732 (0.0404)    Prec 97.656% (98.623%)
Epoch: [74][200/391]    Time 0.084 (0.086)      Data 0.002 (0.003)      Loss
0.0163 (0.0439)    Prec 100.000% (98.535%)
Epoch: [74][300/391]    Time 0.078 (0.086)      Data 0.002 (0.003)      Loss
0.0306 (0.0443)    Prec 98.438% (98.484%)
Validation starts
Test: [0/79]    Time 0.212 (0.212)      Loss 0.2496 (0.2496)    Prec 94.531%
(94.531%)
 * Prec 90.080%
best acc: 90.320000
Epoch: [75][0/391]      Time 0.267 (0.267)      Data 0.209 (0.209)      Loss
0.0444 (0.0444)    Prec 98.438% (98.438%)
Epoch: [75][100/391]    Time 0.092 (0.086)      Data 0.002 (0.004)      Loss
0.0597 (0.0407)    Prec 99.219% (98.523%)
Epoch: [75][200/391]    Time 0.085 (0.085)      Data 0.002 (0.003)      Loss
0.0124 (0.0383)    Prec 100.000% (98.686%)
Epoch: [75][300/391]    Time 0.089 (0.086)      Data 0.001 (0.002)      Loss
0.0579 (0.0404)    Prec 97.656% (98.614%)
Validation starts
Test: [0/79]    Time 0.227 (0.227)      Loss 0.2113 (0.2113)    Prec 92.969%
(92.969%)
 * Prec 90.230%
best acc: 90.320000
Epoch: [76][0/391]      Time 0.264 (0.264)      Data 0.203 (0.203)      Loss
0.0345 (0.0345)    Prec 98.438% (98.438%)
Epoch: [76][100/391]    Time 0.082 (0.087)      Data 0.002 (0.004)      Loss
0.0505 (0.0362)    Prec 97.656% (98.708%)
Epoch: [76][200/391]    Time 0.088 (0.085)      Data 0.001 (0.003)      Loss
0.0440 (0.0388)    Prec 98.438% (98.644%)
Epoch: [76][300/391]    Time 0.085 (0.085)      Data 0.001 (0.002)      Loss
0.0112 (0.0387)    Prec 100.000% (98.624%)
```

```
Validation starts
Test: [0/79]    Time 0.229 (0.229)      Loss 0.2904 (0.2904)    Prec 91.406%
(91.406%)
 * Prec 90.140%
best acc: 90.320000
Epoch: [77][0/391]      Time 0.270 (0.270)      Data 0.206 (0.206)      Loss
0.0074 (0.0074)    Prec 100.000% (100.000%)
Epoch: [77][100/391]    Time 0.086 (0.087)      Data 0.002 (0.004)      Loss
0.0686 (0.0370)    Prec 96.875% (98.801%)
Epoch: [77][200/391]    Time 0.085 (0.086)      Data 0.002 (0.003)      Loss
0.0173 (0.0374)    Prec 99.219% (98.795%)
Epoch: [77][300/391]    Time 0.084 (0.085)      Data 0.002 (0.002)      Loss
0.0822 (0.0363)    Prec 98.438% (98.785%)
Validation starts
Test: [0/79]    Time 0.211 (0.211)      Loss 0.2566 (0.2566)    Prec 94.531%
(94.531%)
 * Prec 90.270%
best acc: 90.320000
Epoch: [78][0/391]      Time 0.278 (0.278)      Data 0.213 (0.213)      Loss
0.0929 (0.0929)    Prec 97.656% (97.656%)
Epoch: [78][100/391]    Time 0.087 (0.086)      Data 0.002 (0.004)      Loss
0.0845 (0.0359)    Prec 98.438% (98.700%)
Epoch: [78][200/391]    Time 0.088 (0.086)      Data 0.002 (0.003)      Loss
0.0302 (0.0370)    Prec 99.219% (98.717%)
Epoch: [78][300/391]    Time 0.096 (0.085)      Data 0.001 (0.002)      Loss
0.0652 (0.0365)    Prec 98.438% (98.728%)
Validation starts
Test: [0/79]    Time 0.226 (0.226)      Loss 0.2699 (0.2699)    Prec 91.406%
(91.406%)
 * Prec 90.070%
best acc: 90.320000
Epoch: [79][0/391]      Time 0.292 (0.292)      Data 0.235 (0.235)      Loss
0.0166 (0.0166)    Prec 99.219% (99.219%)
Epoch: [79][100/391]    Time 0.083 (0.086)      Data 0.002 (0.004)      Loss
0.0162 (0.0347)    Prec 99.219% (98.762%)
Epoch: [79][200/391]    Time 0.083 (0.085)      Data 0.002 (0.003)      Loss
0.0355 (0.0352)    Prec 99.219% (98.725%)
Epoch: [79][300/391]    Time 0.085 (0.085)      Data 0.001 (0.003)      Loss
0.0223 (0.0346)    Prec 99.219% (98.757%)
Validation starts
Test: [0/79]    Time 0.234 (0.234)      Loss 0.2710 (0.2710)    Prec 92.188%
(92.188%)
 * Prec 90.410%
best acc: 90.410000
Epoch: [80][0/391]      Time 0.273 (0.273)      Data 0.210 (0.210)      Loss
0.0228 (0.0228)    Prec 99.219% (99.219%)
Epoch: [80][100/391]    Time 0.081 (0.086)      Data 0.002 (0.004)      Loss
0.0115 (0.0310)    Prec 100.000% (98.871%)
```

```
Epoch: [80][200/391]    Time 0.082 (0.085)      Data 0.002 (0.003)      Loss
0.0290 (0.0326)    Prec 98.438% (98.884%)
Epoch: [80][300/391]    Time 0.084 (0.085)      Data 0.002 (0.002)      Loss
0.0138 (0.0329)    Prec 100.000% (98.868%)
Validation starts
Test: [0/79]    Time 0.225 (0.225)      Loss 0.2549 (0.2549)    Prec 92.969%
(92.969%)
 * Prec 90.270%
best acc: 90.410000
Epoch: [81][0/391]      Time 0.305 (0.305)      Data 0.255 (0.255)      Loss
0.0387 (0.0387)    Prec 98.438% (98.438%)
Epoch: [81][100/391]    Time 0.088 (0.090)      Data 0.002 (0.004)      Loss
0.0239 (0.0337)    Prec 98.438% (98.786%)
Epoch: [81][200/391]    Time 0.100 (0.089)      Data 0.001 (0.003)      Loss
0.0118 (0.0345)    Prec 100.000% (98.826%)
Epoch: [81][300/391]    Time 0.084 (0.087)      Data 0.002 (0.003)      Loss
0.0043 (0.0332)    Prec 100.000% (98.881%)
Validation starts
Test: [0/79]    Time 0.186 (0.186)      Loss 0.2302 (0.2302)    Prec 93.750%
(93.750%)
 * Prec 90.630%
best acc: 90.630000
Epoch: [82][0/391]      Time 0.261 (0.261)      Data 0.199 (0.199)      Loss
0.0085 (0.0085)    Prec 100.000% (100.000%)
Epoch: [82][100/391]    Time 0.082 (0.089)      Data 0.002 (0.004)      Loss
0.0084 (0.0299)    Prec 100.000% (98.925%)
Epoch: [82][200/391]    Time 0.090 (0.087)      Data 0.002 (0.003)      Loss
0.0206 (0.0306)    Prec 99.219% (98.954%)
Epoch: [82][300/391]    Time 0.089 (0.087)      Data 0.002 (0.002)      Loss
0.0186 (0.0320)    Prec 100.000% (98.889%)
Validation starts
Test: [0/79]    Time 0.206 (0.206)      Loss 0.2792 (0.2792)    Prec 93.750%
(93.750%)
 * Prec 90.320%
best acc: 90.630000
Epoch: [83][0/391]      Time 0.253 (0.253)      Data 0.195 (0.195)      Loss
0.0309 (0.0309)    Prec 99.219% (99.219%)
Epoch: [83][100/391]    Time 0.053 (0.086)      Data 0.002 (0.004)      Loss
0.0452 (0.0323)    Prec 97.656% (98.933%)
Epoch: [83][200/391]    Time 0.087 (0.087)      Data 0.004 (0.003)      Loss
0.0107 (0.0311)    Prec 100.000% (98.958%)
Epoch: [83][300/391]    Time 0.087 (0.086)      Data 0.001 (0.002)      Loss
0.0429 (0.0311)    Prec 96.875% (98.944%)
Validation starts
Test: [0/79]    Time 0.252 (0.252)      Loss 0.3212 (0.3212)    Prec 92.188%
(92.188%)
 * Prec 90.270%
best acc: 90.630000
```

```
Epoch: [84][0/391]      Time 0.253 (0.253)      Data 0.202 (0.202)      Loss
0.0138 (0.0138)    Prec 99.219% (99.219%)
Epoch: [84][100/391]    Time 0.082 (0.086)      Data 0.002 (0.004)      Loss
0.0740 (0.0318)    Prec 97.656% (98.925%)
Epoch: [84][200/391]    Time 0.083 (0.086)      Data 0.002 (0.003)      Loss
0.0414 (0.0320)    Prec 98.438% (98.877%)
Epoch: [84][300/391]    Time 0.099 (0.086)      Data 0.001 (0.002)      Loss
0.0144 (0.0325)    Prec 99.219% (98.879%)
Validation starts
Test: [0/79]    Time 0.190 (0.190)      Loss 0.2530 (0.2530)    Prec 93.750%
(93.750%)
 * Prec 90.230%
best acc: 90.630000
Epoch: [85][0/391]      Time 0.252 (0.252)      Data 0.196 (0.196)      Loss
0.0331 (0.0331)    Prec 98.438% (98.438%)
Epoch: [85][100/391]    Time 0.087 (0.087)      Data 0.001 (0.004)      Loss
0.0287 (0.0312)    Prec 98.438% (98.933%)
Epoch: [85][200/391]    Time 0.090 (0.087)      Data 0.002 (0.003)      Loss
0.0431 (0.0302)    Prec 97.656% (98.919%)
Epoch: [85][300/391]    Time 0.085 (0.087)      Data 0.002 (0.002)      Loss
0.0314 (0.0309)    Prec 98.438% (98.905%)
Validation starts
Test: [0/79]    Time 0.194 (0.194)      Loss 0.2681 (0.2681)    Prec 91.406%
(91.406%)
 * Prec 90.240%
best acc: 90.630000
Epoch: [86][0/391]      Time 0.278 (0.278)      Data 0.224 (0.224)      Loss
0.0562 (0.0562)    Prec 96.875% (96.875%)
Epoch: [86][100/391]    Time 0.088 (0.090)      Data 0.001 (0.004)      Loss
0.0188 (0.0297)    Prec 99.219% (98.925%)
Epoch: [86][200/391]    Time 0.084 (0.088)      Data 0.002 (0.003)      Loss
0.0193 (0.0306)    Prec 100.000% (98.958%)
Epoch: [86][300/391]    Time 0.086 (0.087)      Data 0.001 (0.002)      Loss
0.0427 (0.0312)    Prec 98.438% (98.918%)
Validation starts
Test: [0/79]    Time 0.214 (0.214)      Loss 0.3141 (0.3141)    Prec 92.969%
(92.969%)
 * Prec 90.320%
best acc: 90.630000
Epoch: [87][0/391]      Time 0.215 (0.215)      Data 0.170 (0.170)      Loss
0.0422 (0.0422)    Prec 97.656% (97.656%)
Epoch: [87][100/391]    Time 0.085 (0.087)      Data 0.002 (0.003)      Loss
0.0065 (0.0295)    Prec 100.000% (98.987%)
Epoch: [87][200/391]    Time 0.084 (0.086)      Data 0.002 (0.003)      Loss
0.0202 (0.0287)    Prec 99.219% (99.071%)
Epoch: [87][300/391]    Time 0.051 (0.086)      Data 0.001 (0.002)      Loss
0.0072 (0.0280)    Prec 100.000% (99.089%)
Validation starts
```

```
Test: [0/79]    Time 0.239 (0.239)      Loss 0.2899 (0.2899)    Prec 92.969%
(92.969%)
 * Prec 90.330%
best acc: 90.630000
Epoch: [88][0/391]      Time 0.267 (0.267)      Data 0.214 (0.214)      Loss
0.0932 (0.0932)    Prec 96.094% (96.094%)
Epoch: [88][100/391]    Time 0.082 (0.088)      Data 0.002 (0.004)      Loss
0.0306 (0.0305)    Prec 98.438% (98.940%)
Epoch: [88][200/391]    Time 0.086 (0.087)      Data 0.002 (0.003)      Loss
0.0115 (0.0298)    Prec 100.000% (98.943%)
Epoch: [88][300/391]    Time 0.085 (0.085)      Data 0.002 (0.002)      Loss
0.0110 (0.0302)    Prec 100.000% (98.967%)
Validation starts
Test: [0/79]    Time 0.206 (0.206)      Loss 0.3248 (0.3248)    Prec 93.750%
(93.750%)
 * Prec 90.210%
best acc: 90.630000
Epoch: [89][0/391]      Time 0.252 (0.252)      Data 0.195 (0.195)      Loss
0.0213 (0.0213)    Prec 99.219% (99.219%)
Epoch: [89][100/391]    Time 0.085 (0.086)      Data 0.001 (0.004)      Loss
0.0143 (0.0288)    Prec 100.000% (99.056%)
Epoch: [89][200/391]    Time 0.086 (0.085)      Data 0.002 (0.003)      Loss
0.0525 (0.0299)    Prec 98.438% (99.017%)
Epoch: [89][300/391]    Time 0.087 (0.085)      Data 0.002 (0.002)      Loss
0.0024 (0.0287)    Prec 100.000% (99.019%)
Validation starts
Test: [0/79]    Time 0.244 (0.244)      Loss 0.2329 (0.2329)    Prec 92.969%
(92.969%)
 * Prec 90.340%
best acc: 90.630000
Epoch: [90][0/391]      Time 0.316 (0.316)      Data 0.259 (0.259)      Loss
0.0220 (0.0220)    Prec 99.219% (99.219%)
Epoch: [90][100/391]    Time 0.083 (0.087)      Data 0.003 (0.004)      Loss
0.0208 (0.0283)    Prec 99.219% (99.064%)
Epoch: [90][200/391]    Time 0.071 (0.086)      Data 0.001 (0.003)      Loss
0.0152 (0.0294)    Prec 100.000% (98.993%)
Epoch: [90][300/391]    Time 0.080 (0.086)      Data 0.002 (0.003)      Loss
0.0189 (0.0294)    Prec 99.219% (98.996%)
Validation starts
Test: [0/79]    Time 0.235 (0.235)      Loss 0.3423 (0.3423)    Prec 92.188%
(92.188%)
 * Prec 90.460%
best acc: 90.630000
Epoch: [91][0/391]      Time 0.258 (0.258)      Data 0.190 (0.190)      Loss
0.0256 (0.0256)    Prec 99.219% (99.219%)
Epoch: [91][100/391]    Time 0.092 (0.088)      Data 0.002 (0.004)      Loss
0.0307 (0.0308)    Prec 99.219% (98.925%)
Epoch: [91][200/391]    Time 0.091 (0.086)      Data 0.002 (0.003)      Loss
```

```
0.0268 (0.0284)    Prec 99.219% (99.024%)
Epoch: [91][300/391]    Time 0.085 (0.085)      Data 0.002 (0.003)      Loss
0.0310 (0.0280)    Prec 98.438% (99.068%)
Validation starts
Test: [0/79]    Time 0.248 (0.248)      Loss 0.3416 (0.3416)    Prec 92.969%
(92.969%)
 * Prec 90.410%
best acc: 90.630000
Epoch: [92][0/391]      Time 0.281 (0.281)      Data 0.223 (0.223)      Loss
0.0077 (0.0077)    Prec 100.000% (100.000%)
Epoch: [92][100/391]    Time 0.088 (0.087)      Data 0.002 (0.004)      Loss
0.0732 (0.0275)    Prec 96.875% (99.002%)
Epoch: [92][200/391]    Time 0.097 (0.087)      Data 0.002 (0.003)      Loss
0.0237 (0.0268)    Prec 99.219% (99.052%)
Epoch: [92][300/391]    Time 0.086 (0.087)      Data 0.002 (0.003)      Loss
0.0730 (0.0265)    Prec 98.438% (99.073%)
Validation starts
Test: [0/79]    Time 0.217 (0.217)      Loss 0.2975 (0.2975)    Prec 91.406%
(91.406%)
 * Prec 90.350%
best acc: 90.630000
Epoch: [93][0/391]      Time 0.273 (0.273)      Data 0.214 (0.214)      Loss
0.0247 (0.0247)    Prec 99.219% (99.219%)
Epoch: [93][100/391]    Time 0.084 (0.086)      Data 0.001 (0.004)      Loss
0.0045 (0.0277)    Prec 100.000% (99.049%)
Epoch: [93][200/391]    Time 0.088 (0.086)      Data 0.002 (0.003)      Loss
0.0142 (0.0280)    Prec 100.000% (99.056%)
Epoch: [93][300/391]    Time 0.082 (0.086)      Data 0.002 (0.003)      Loss
0.0339 (0.0275)    Prec 97.656% (99.084%)
Validation starts
Test: [0/79]    Time 0.257 (0.257)      Loss 0.2701 (0.2701)    Prec 93.750%
(93.750%)
 * Prec 90.440%
best acc: 90.630000
Epoch: [94][0/391]      Time 0.287 (0.287)      Data 0.221 (0.221)      Loss
0.0192 (0.0192)    Prec 99.219% (99.219%)
Epoch: [94][100/391]    Time 0.087 (0.085)      Data 0.002 (0.004)      Loss
0.0322 (0.0265)    Prec 97.656% (99.072%)
Epoch: [94][200/391]    Time 0.082 (0.085)      Data 0.002 (0.003)      Loss
0.0246 (0.0291)    Prec 99.219% (98.974%)
Epoch: [94][300/391]    Time 0.090 (0.085)      Data 0.003 (0.003)      Loss
0.0515 (0.0281)    Prec 97.656% (99.016%)
Validation starts
Test: [0/79]    Time 0.220 (0.220)      Loss 0.3040 (0.3040)    Prec 92.969%
(92.969%)
 * Prec 90.330%
best acc: 90.630000
Epoch: [95][0/391]      Time 0.329 (0.329)      Data 0.262 (0.262)      Loss
```

```
0.0049 (0.0049)    Prec 100.000% (100.000%)
Epoch: [95][100/391]    Time 0.085 (0.087)      Data 0.002 (0.004)      Loss
0.0301 (0.0278)    Prec 98.438% (98.987%)
Epoch: [95][200/391]    Time 0.086 (0.087)      Data 0.002 (0.003)      Loss
0.0181 (0.0264)    Prec 99.219% (99.052%)
Epoch: [95][300/391]    Time 0.082 (0.086)      Data 0.002 (0.003)      Loss
0.0072 (0.0269)    Prec 100.000% (99.058%)
Validation starts
Test: [0/79]    Time 0.209 (0.209)      Loss 0.2925 (0.2925)    Prec 92.188%
(92.188%)
 * Prec 90.420%
best acc: 90.630000
Epoch: [96][0/391]      Time 0.300 (0.300)      Data 0.239 (0.239)      Loss
0.0654 (0.0654)    Prec 97.656% (97.656%)
Epoch: [96][100/391]    Time 0.085 (0.089)      Data 0.003 (0.004)      Loss
0.0768 (0.0276)    Prec 97.656% (99.049%)
Epoch: [96][200/391]    Time 0.082 (0.087)      Data 0.002 (0.003)      Loss
0.0398 (0.0280)    Prec 98.438% (99.028%)
Epoch: [96][300/391]    Time 0.085 (0.087)      Data 0.002 (0.003)      Loss
0.0836 (0.0271)    Prec 97.656% (99.089%)
Validation starts
Test: [0/79]    Time 0.213 (0.213)      Loss 0.3079 (0.3079)    Prec 92.969%
(92.969%)
 * Prec 90.450%
best acc: 90.630000
Epoch: [97][0/391]      Time 0.310 (0.310)      Data 0.256 (0.256)      Loss
0.0271 (0.0271)    Prec 99.219% (99.219%)
Epoch: [97][100/391]    Time 0.090 (0.089)      Data 0.002 (0.004)      Loss
0.0195 (0.0292)    Prec 99.219% (99.002%)
Epoch: [97][200/391]    Time 0.087 (0.088)      Data 0.002 (0.003)      Loss
0.0063 (0.0286)    Prec 100.000% (99.009%)
Epoch: [97][300/391]    Time 0.079 (0.087)      Data 0.002 (0.003)      Loss
0.0222 (0.0277)    Prec 99.219% (99.053%)
Validation starts
Test: [0/79]    Time 0.192 (0.192)      Loss 0.3301 (0.3301)    Prec 92.188%
(92.188%)
 * Prec 90.330%
best acc: 90.630000
Epoch: [98][0/391]      Time 0.297 (0.297)      Data 0.240 (0.240)      Loss
0.0148 (0.0148)    Prec 99.219% (99.219%)
Epoch: [98][100/391]    Time 0.089 (0.088)      Data 0.002 (0.004)      Loss
0.0060 (0.0260)    Prec 100.000% (99.118%)
Epoch: [98][200/391]    Time 0.086 (0.087)      Data 0.002 (0.003)      Loss
0.0444 (0.0245)    Prec 98.438% (99.192%)
Epoch: [98][300/391]    Time 0.100 (0.086)      Data 0.002 (0.003)      Loss
0.0268 (0.0253)    Prec 98.438% (99.167%)
Validation starts
Test: [0/79]    Time 0.238 (0.238)      Loss 0.3132 (0.3132)    Prec 92.188%
```

```
(92.188%)
 * Prec 90.380%
best acc: 90.630000
Epoch: [99][0/391]      Time 0.295 (0.295)      Data 0.244 (0.244)      Loss
0.0407 (0.0407)    Prec 97.656% (97.656%)
Epoch: [99][100/391]    Time 0.085 (0.089)      Data 0.002 (0.004)      Loss
0.0024 (0.0235)    Prec 100.000% (99.304%)
Epoch: [99][200/391]    Time 0.082 (0.087)      Data 0.002 (0.003)      Loss
0.0477 (0.0249)    Prec 98.438% (99.227%)
Epoch: [99][300/391]    Time 0.083 (0.086)      Data 0.002 (0.003)      Loss
0.0063 (0.0247)    Prec 100.000% (99.180%)
Validation starts
Test: [0/79]    Time 0.266 (0.266)      Loss 0.2783 (0.2783)    Prec 93.750%
(93.750%)
 * Prec 90.190%
best acc: 90.630000
```

[10]:
```python
class SaveOutput:
    def __init__(self):
        self.outputs = []
    def __call__(self, module, module_in):
        self.outputs.append(module_in)
    def clear(self):
        self.outputs = []

######### Save inputs from selected layer ##########
save_output = SaveOutput()
device = torch.device("cuda" if use_gpu else "cpu")
counter =0
for layer in model.modules():
    if isinstance(layer, torch.nn.Conv2d):
        print("prehooked")
        counter += 1
        print(layer, counter)
        layer.register_forward_pre_hook(save_output)       ## Input for the␣
 ↪module will be grapped
####################################################

dataiter = iter(trainloader)
images, labels = dataiter.next()
images = images.to(device)
out = model(images)
```

```
prehooked
QuantConv2d(
  3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
  (weight_quant): weight_quantize_fn()
) 1
```

```
prehooked
QuantConv2d(
  64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
  (weight_quant): weight_quantize_fn()
) 2
prehooked
QuantConv2d(
  64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
  (weight_quant): weight_quantize_fn()
) 3
prehooked
QuantConv2d(
  128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
  (weight_quant): weight_quantize_fn()
) 4
prehooked
QuantConv2d(
  128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
  (weight_quant): weight_quantize_fn()
) 5
prehooked
QuantConv2d(
  256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
  (weight_quant): weight_quantize_fn()
) 6
prehooked
QuantConv2d(
  256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
  (weight_quant): weight_quantize_fn()
) 7
prehooked
QuantConv2d(
  256, 8, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
  (weight_quant): weight_quantize_fn()
) 8
prehooked
QuantConv2d(
  8, 8, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
  (weight_quant): weight_quantize_fn()
) 9
prehooked
QuantConv2d(
  8, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
  (weight_quant): weight_quantize_fn()
) 10
prehooked
QuantConv2d(
  512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
```

```
  (weight_quant): weight_quantize_fn()
) 11
prehooked
QuantConv2d(
  512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
  (weight_quant): weight_quantize_fn()
) 12
prehooked
QuantConv2d(
  512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
  (weight_quant): weight_quantize_fn()
) 13
```

[ ]:
```
# HW

#  1. Train with 4 bits for both weight and activation to achieve >90% accuracy
#  2. Find x_int and w_int for the 2nd convolution layer
#  3. Check the recovered psum has similar value to the un-quantized original␣
 ↪psum
#     (such as example 1 in W3S2)
```

[3]:
```
PATH = "result/VGG16_quant4bit/model_best.pth.tar"
checkpoint = torch.load(PATH)
model.load_state_dict(checkpoint['state_dict'])
device = torch.device("cuda")

model.cuda()
model.eval()

test_loss = 0
correct = 0

with torch.no_grad():
    for data, target in testloader:
        data, target = data.to(device), target.to(device) # loading to GPU
        output = model(data)
        pred = output.argmax(dim=1, keepdim=True)
        correct += pred.eq(target.view_as(pred)).sum().item()

test_loss /= len(testloader.dataset)

print('\nTest set: Accuracy: {}/{} ({:.0f}%)\n'.format(
        correct, len(testloader.dataset),
        100. * correct / len(testloader.dataset)))
```

```
/opt/conda/lib/python3.9/site-packages/torch/nn/functional.py:718: UserWarning:
Named tensors and all their associated APIs are an experimental feature and
subject to change. Please do not use them for anything important until they are
```

```
released as stable. (Triggered internally at
/pytorch/c10/core/TensorImpl.h:1156.)
  return torch.max_pool2d(input, kernel_size, stride, padding, dilation,
ceil_mode)


Test set: Accuracy: 9062/10000 (91%)
```

[ ]:
```
print(len(save_output.outputs))
```

[ ]:
```
len(save_output.outputs[0][0])
```

[66]:
```
w_bit = 4
weight_q = model.features[27].weight_q # quantized value is stored during the␣
 ↪training
w_alpha = model.features[27].weight_quant.wgt_alpha
w_delta = w_alpha/(2**(w_bit-1)-1)
weight_int = weight_q/w_delta
print(weight_int) # you should see clean integer numbers
```

```
tensor([[[[ 7.0000, -7.0000,  7.0000],
          [-2.0000,  7.0000, -7.0000],
          [-7.0000,  7.0000, -7.0000]],

         [[ 7.0000, -7.0000, -7.0000],
          [ 7.0000,  7.0000,  7.0000],
          [ 7.0000,  7.0000,  7.0000]],

         [[ 7.0000,  7.0000,  7.0000],
          [ 7.0000,  7.0000,  7.0000],
          [-7.0000, -7.0000, -7.0000]],

         [[ 7.0000, -7.0000,  7.0000],
          [-7.0000, -7.0000, -7.0000],
          [ 7.0000, -7.0000,  7.0000]],

         [[-7.0000, -7.0000,  7.0000],
          [ 7.0000,  7.0000, -7.0000],
          [ 7.0000, -7.0000, -7.0000]],

         [[ 7.0000,  7.0000,  7.0000],
          [ 7.0000,  7.0000, -7.0000],
          [ 7.0000, -7.0000, -7.0000]],

         [[-7.0000, -7.0000, -7.0000],
          [ 7.0000,  7.0000, -7.0000],
          [-7.0000, -7.0000,  7.0000]],
```

```
 [[-7.0000,  7.0000,  7.0000],
  [-7.0000,  7.0000,  7.0000],
  [ 7.0000,  7.0000,  7.0000]]],


[[[ 7.0000,  7.0000,  7.0000],
  [ 1.0000,  7.0000,  7.0000],
  [-7.0000,  7.0000,  7.0000]],

 [[-7.0000,  7.0000,  7.0000],
  [-7.0000, -7.0000, -7.0000],
  [-7.0000,  7.0000,  7.0000]],

 [[ 7.0000,  7.0000,  7.0000],
  [-7.0000, -7.0000, -7.0000],
  [ 7.0000,  7.0000,  7.0000]],

 [[-7.0000, -7.0000, -5.0000],
  [-7.0000, -7.0000, -2.0000],
  [ 7.0000,  7.0000,  7.0000]],

 [[-7.0000, -7.0000, -7.0000],
  [ 7.0000,  7.0000,  7.0000],
  [ 7.0000,  7.0000,  7.0000]],

 [[-7.0000, -7.0000,  7.0000],
  [-7.0000,  7.0000,  7.0000],
  [ 7.0000, -7.0000,  7.0000]],

 [[-7.0000, -7.0000, -7.0000],
  [-7.0000, -7.0000, -7.0000],
  [-7.0000, -7.0000, -7.0000]],

 [[-7.0000, -7.0000, -7.0000],
  [ 7.0000, -7.0000, -7.0000],
  [ 7.0000, -7.0000, -7.0000]]],


[[[ 7.0000,  7.0000, -7.0000],
  [ 7.0000,  7.0000, -7.0000],
  [-7.0000, -7.0000,  7.0000]],

 [[ 7.0000, -7.0000, -7.0000],
  [ 7.0000,  7.0000,  7.0000],
  [ 7.0000,  7.0000,  7.0000]],

 [[ 7.0000, -7.0000,  7.0000],
```

```
  [ 7.0000, -7.0000, -7.0000],
  [ 7.0000,  7.0000,  7.0000]],

 [[ 7.0000,  7.0000,  7.0000],
  [ 7.0000,  7.0000,  7.0000],
  [-7.0000, -7.0000, -7.0000]],

 [[-7.0000, -7.0000,  7.0000],
  [-7.0000, -7.0000,  7.0000],
  [-7.0000,  7.0000,  7.0000]],

 [[-7.0000,  7.0000,  7.0000],
  [-7.0000, -7.0000, -7.0000],
  [ 7.0000, -7.0000, -7.0000]],

 [[ 7.0000,  7.0000, -7.0000],
  [ 7.0000, -7.0000,  7.0000],
  [ 7.0000,  7.0000,  7.0000]],

 [[-7.0000,  7.0000,  7.0000],
  [ 7.0000, -7.0000,  7.0000],
  [-7.0000, -7.0000, -7.0000]]],


[[[ 7.0000,  7.0000,  7.0000],
  [-7.0000,  7.0000,  7.0000],
  [-7.0000, -7.0000,  7.0000]],

 [[ 7.0000, -7.0000, -7.0000],
  [ 7.0000,  7.0000,  7.0000],
  [ 7.0000,  7.0000, -7.0000]],

 [[-7.0000, -7.0000, -7.0000],
  [-7.0000, -7.0000, -7.0000],
  [ 7.0000,  7.0000, -7.0000]],

 [[-7.0000, -7.0000,  7.0000],
  [ 7.0000,  7.0000, -7.0000],
  [ 7.0000,  7.0000, -7.0000]],

 [[ 7.0000,  7.0000,  7.0000],
  [ 7.0000,  7.0000, -7.0000],
  [ 7.0000,  7.0000, -7.0000]],

 [[ 7.0000, -7.0000, -7.0000],
  [ 7.0000,  7.0000, -7.0000],
  [ 7.0000,  7.0000, -7.0000]],
```

```
 [[ 7.0000,  7.0000, -7.0000],
  [-7.0000, -7.0000, -7.0000],
  [-7.0000, -7.0000, -7.0000]],

 [[-7.0000, -7.0000, -7.0000],
  [-7.0000, -7.0000,  7.0000],
  [-7.0000, -7.0000,  7.0000]]],


[[[ 7.0000,  7.0000, -7.0000],
  [ 7.0000,  7.0000, -7.0000],
  [ 7.0000,  7.0000, -7.0000]],

 [[-7.0000,  7.0000,  7.0000],
  [-7.0000,  7.0000,  7.0000],
  [ 7.0000, -7.0000, -7.0000]],

 [[ 7.0000, -7.0000, -7.0000],
  [-7.0000, -7.0000, -7.0000],
  [-7.0000, -7.0000, -7.0000]],

 [[-7.0000,  7.0000,  7.0000],
  [-7.0000, -7.0000,  7.0000],
  [-7.0000, -7.0000, -7.0000]],

 [[ 7.0000, -7.0000,  7.0000],
  [ 7.0000,  7.0000,  7.0000],
  [ 7.0000,  7.0000,  7.0000]],

 [[-7.0000,  7.0000,  7.0000],
  [-7.0000,  7.0000,  7.0000],
  [-7.0000,  7.0000, -7.0000]],

 [[-7.0000,  7.0000,  7.0000],
  [ 7.0000,  7.0000,  7.0000],
  [-7.0000,  7.0000, -7.0000]],

 [[ 7.0000,  7.0000,  7.0000],
  [ 7.0000,  7.0000,  7.0000],
  [ 7.0000,  7.0000,  7.0000]]],


[[[ 7.0000,  7.0000,  7.0000],
  [ 7.0000,  7.0000,  7.0000],
  [ 7.0000,  7.0000,  7.0000]],

 [[-7.0000,  7.0000,  7.0000],
  [-7.0000, -7.0000,  7.0000],
```

```
  [-7.0000,  7.0000, -7.0000]],

 [[ 7.0000, -7.0000, -7.0000],
  [ 7.0000,  7.0000,  7.0000],
  [-7.0000, -7.0000,  7.0000]],

 [[ 7.0000,  7.0000,  7.0000],
  [ 7.0000,  7.0000, -7.0000],
  [-7.0000, -7.0000, -7.0000]],

 [[-7.0000,  7.0000,  7.0000],
  [-7.0000, -7.0000,  7.0000],
  [-1.0000, -7.0000,  7.0000]],

 [[ 7.0000, -7.0000, -7.0000],
  [ 7.0000, -7.0000, -7.0000],
  [-7.0000, -7.0000, -7.0000]],

 [[-7.0000, -7.0000, -7.0000],
  [-7.0000, -7.0000, -7.0000],
  [ 7.0000, -7.0000,  4.0000]],

 [[-7.0000,  7.0000,  7.0000],
  [-7.0000, -7.0000, -7.0000],
  [ 7.0000,  1.0000, -7.0000]]],


[[[ 7.0000,  7.0000,  7.0000],
  [ 7.0000,  7.0000,  7.0000],
  [ 7.0000, -7.0000,  7.0000]],

 [[-7.0000,  7.0000,  7.0000],
  [-7.0000, -7.0000,  7.0000],
  [ 7.0000,  7.0000, -7.0000]],

 [[-7.0000, -7.0000, -7.0000],
  [-7.0000,  7.0000,  7.0000],
  [ 7.0000,  7.0000,  7.0000]],

 [[ 7.0000, -7.0000, -7.0000],
  [ 7.0000, -7.0000, -7.0000],
  [-7.0000,  7.0000, -7.0000]],

 [[-7.0000, -7.0000,  7.0000],
  [ 7.0000,  7.0000,  7.0000],
  [ 7.0000,  7.0000,  7.0000]],

 [[-7.0000, -7.0000, -7.0000],
```

```
          [-7.0000, -7.0000, -7.0000],
          [ 7.0000, -7.0000, -7.0000]],

         [[-7.0000, -7.0000, -7.0000],
          [-7.0000, -7.0000, -7.0000],
          [ 7.0000,  7.0000, -7.0000]],

         [[-7.0000,  7.0000,  7.0000],
          [ 7.0000,  7.0000,  7.0000],
          [ 7.0000,  7.0000, -7.0000]]],


        [[[-7.0000,  7.0000,  7.0000],
          [-7.0000, -7.0000, -7.0000],
          [ 7.0000,  7.0000,  7.0000]],

         [[-7.0000,  7.0000,  7.0000],
          [-7.0000, -7.0000,  7.0000],
          [-7.0000,  7.0000, -7.0000]],

         [[ 7.0000,  7.0000,  7.0000],
          [-7.0000, -7.0000, -7.0000],
          [ 7.0000,  7.0000,  7.0000]],

         [[ 7.0000,  7.0000, -7.0000],
          [ 7.0000,  7.0000,  7.0000],
          [-7.0000,  7.0000, -7.0000]],

         [[ 7.0000,  7.0000,  7.0000],
          [ 7.0000,  7.0000, -7.0000],
          [ 7.0000,  7.0000,  7.0000]],

         [[-7.0000, -7.0000, -7.0000],
          [-7.0000,  7.0000,  7.0000],
          [-7.0000, -7.0000, -7.0000]],

         [[-7.0000, -7.0000,  7.0000],
          [ 7.0000,  7.0000,  7.0000],
          [ 7.0000, -7.0000, -7.0000]],

         [[-7.0000,  7.0000, -7.0000],
          [-7.0000,  7.0000,  7.0000],
          [ 7.0000,  7.0000,  7.0000]]]], device='cuda:0',
       grad_fn=<DivBackward0>)
```

[67]:
```
x_bit = 4
x = save_output.outputs[8][0]  # input of the 8th conv layer
```

```
x_alpha  = model.features[27].act_alpha
x_delta = x_alpha/(2**x_bit-1)

act_quant_fn = act_quantization(x_bit) # define the quantization function
x_q = act_quant_fn(x, x_alpha)         # create the quantized value for x

x_int = x_q/x_delta
print(x_int) # you should see clean integer numbers
```

```
tensor([[[[ 0.0000,  0.0000,  0.0000,  0.0000],
          [ 0.0000,  0.0000,  0.0000,  0.0000],
          [ 0.0000,  0.0000,  0.0000,  0.0000],
          [ 0.0000,  0.0000,  0.0000,  0.0000]],

         [[ 5.0000,  2.0000,  2.0000,  1.0000],
          [ 5.0000,  0.0000,  2.0000,  0.0000],
          [ 4.0000,  1.0000,  4.0000,  2.0000],
          [ 4.0000,  2.0000,  4.0000,  3.0000]],

         [[ 2.0000,  1.0000,  1.0000,  1.0000],
          [ 1.0000,  0.0000,  0.0000,  1.0000],
          [ 2.0000,  3.0000,  4.0000,  3.0000],
          [ 2.0000,  3.0000,  4.0000,  3.0000]],

         …,

         [[ 1.0000,  0.0000,  0.0000,  0.0000],
          [ 1.0000,  1.0000,  0.0000,  2.0000],
          [ 2.0000,  1.0000,  0.0000,  1.0000],
          [ 1.0000,  1.0000,  1.0000,  1.0000]],

         [[ 1.0000,  2.0000,  2.0000,  1.0000],
          [ 2.0000,  3.0000,  4.0000,  3.0000],
          [ 1.0000,  2.0000,  3.0000,  2.0000],
          [ 2.0000,  2.0000,  1.0000,  0.0000]],

         [[ 0.0000,  0.0000,  1.0000,  1.0000],
          [ 0.0000,  1.0000,  3.0000,  1.0000],
          [ 0.0000,  0.0000,  1.0000,  0.0000],
          [ 0.0000,  0.0000,  0.0000,  0.0000]]],


        [[[ 0.0000,  0.0000,  0.0000,  0.0000],
          [ 1.0000,  0.0000,  0.0000,  0.0000],
          [ 0.0000,  0.0000,  0.0000,  0.0000],
          [ 0.0000,  0.0000,  0.0000,  0.0000]],
```

```
 [[ 4.0000,  3.0000,  5.0000,  4.0000],
  [ 6.0000,  0.0000,  1.0000,  0.0000],
  [13.0000, 15.0000, 14.0000,  1.0000],
  [ 8.0000, 11.0000, 11.0000,  3.0000]],

 [[ 2.0000,  3.0000,  1.0000,  4.0000],
  [ 3.0000,  9.0000, 12.0000, 11.0000],
  [ 0.0000,  5.0000, 15.0000, 12.0000],
  [ 1.0000,  3.0000,  6.0000,  5.0000]],

 …,

 [[ 2.0000,  0.0000,  0.0000,  0.0000],
  [ 6.0000,  2.0000,  0.0000,  0.0000],
  [ 9.0000, 11.0000,  4.0000,  0.0000],
  [ 5.0000,  5.0000,  2.0000,  0.0000]],

 [[ 0.0000,  0.0000,  0.0000,  0.0000],
  [ 0.0000,  0.0000,  0.0000,  0.0000],
  [ 7.0000,  9.0000,  5.0000,  1.0000],
  [ 3.0000,  3.0000,  1.0000,  0.0000]],

 [[ 2.0000,  6.0000, 10.0000,  9.0000],
  [ 7.0000, 13.0000, 12.0000,  9.0000],
  [ 4.0000,  8.0000,  6.0000,  0.0000],
  [ 0.0000,  0.0000,  0.0000,  0.0000]]],


[[[ 0.0000,  0.0000,  0.0000,  0.0000],
  [ 0.0000,  0.0000,  0.0000,  0.0000],
  [ 0.0000,  0.0000,  0.0000,  0.0000],
  [ 0.0000,  0.0000,  0.0000,  0.0000]],

 [[ 5.0000,  0.0000,  0.0000,  0.0000],
  [ 7.0000,  0.0000,  0.0000,  0.0000],
  [ 4.0000,  0.0000,  0.0000,  0.0000],
  [ 2.0000,  0.0000,  0.0000,  0.0000]],

 [[ 1.0000,  6.0000,  7.0000,  6.0000],
  [ 1.0000,  5.0000,  3.0000,  0.0000],
  [ 1.0000,  1.0000,  0.0000,  0.0000],
  [ 2.0000,  2.0000,  0.0000,  0.0000]],

 …,

 [[ 4.0000, 10.0000, 10.0000,  8.0000],
  [ 4.0000,  3.0000,  1.0000,  0.0000],
  [ 1.0000,  0.0000,  0.0000,  0.0000],
```

```
  [ 1.0000,  0.0000,  0.0000,  0.0000]],

 [[ 1.0000,  1.0000,  3.0000,  3.0000],
  [ 0.0000,  0.0000,  0.0000,  0.0000],
  [ 1.0000,  0.0000,  0.0000,  0.0000],
  [ 1.0000,  0.0000,  0.0000,  0.0000]],

 [[ 2.0000,  2.0000,  8.0000, 11.0000],
  [ 2.0000,  0.0000,  5.0000, 11.0000],
  [ 1.0000,  0.0000,  0.0000,  2.0000],
  [ 0.0000,  0.0000,  0.0000,  0.0000]]],


…,


[[[ 0.0000,  0.0000,  0.0000,  0.0000],
  [ 0.0000,  0.0000,  0.0000,  0.0000],
  [ 0.0000,  0.0000,  0.0000,  0.0000],
  [ 0.0000,  0.0000,  0.0000,  0.0000]],

 [[ 6.0000,  9.0000, 10.0000,  5.0000],
  [ 6.0000,  7.0000,  7.0000,  6.0000],
  [ 7.0000,  7.0000,  8.0000,  6.0000],
  [ 6.0000,  3.0000,  3.0000,  1.0000]],

 [[ 3.0000,  3.0000,  0.0000,  0.0000],
  [ 4.0000,  2.0000,  0.0000,  0.0000],
  [ 2.0000,  0.0000,  0.0000,  0.0000],
  [ 3.0000,  2.0000,  2.0000,  2.0000]],

 …,

 [[ 2.0000,  0.0000,  0.0000,  0.0000],
  [ 2.0000,  0.0000,  0.0000,  0.0000],
  [ 2.0000,  0.0000,  0.0000,  0.0000],
  [ 0.0000,  0.0000,  0.0000,  0.0000]],

 [[ 1.0000,  0.0000,  0.0000,  0.0000],
  [ 0.0000,  0.0000,  3.0000,  3.0000],
  [ 1.0000,  8.0000,  8.0000,  5.0000],
  [ 2.0000,  2.0000,  2.0000,  2.0000]],

 [[ 0.0000,  0.0000,  0.0000,  0.0000],
  [ 0.0000,  0.0000,  0.0000,  0.0000],
  [ 0.0000,  0.0000,  0.0000,  0.0000],
  [ 0.0000,  0.0000,  0.0000,  0.0000]]],
```

```
[[[ 0.0000,  0.0000,  0.0000,  0.0000],
  [ 0.0000,  0.0000,  0.0000,  0.0000],
  [ 0.0000,  0.0000,  0.0000,  0.0000],
  [ 0.0000,  0.0000,  0.0000,  0.0000]],

 [[ 6.0000,  5.0000, 10.0000,  5.0000],
  [ 8.0000,  6.0000, 11.0000,  1.0000],
  [10.0000,  7.0000,  8.0000,  0.0000],
  [ 7.0000,  3.0000,  4.0000,  1.0000]],

 [[ 3.0000,  0.0000,  2.0000,  2.0000],
  [ 2.0000,  0.0000,  0.0000,  1.0000],
  [ 2.0000,  1.0000,  1.0000,  2.0000],
  [ 1.0000,  1.0000,  2.0000,  2.0000]],

 …,

 [[ 0.0000,  0.0000,  0.0000,  0.0000],
  [ 0.0000,  0.0000,  0.0000,  0.0000],
  [ 1.0000,  0.0000,  0.0000,  2.0000],
  [ 0.0000,  0.0000,  0.0000,  1.0000]],

 [[ 0.0000,  2.0000,  4.0000,  0.0000],
  [ 1.0000,  2.0000,  3.0000,  0.0000],
  [ 3.0000,  3.0000,  4.0000,  1.0000],
  [ 2.0000,  2.0000,  2.0000,  0.0000]],

 [[ 0.0000,  2.0000,  1.0000,  1.0000],
  [ 0.0000,  2.0000,  0.0000,  0.0000],
  [ 0.0000,  1.0000,  0.0000,  0.0000],
  [ 0.0000,  0.0000,  0.0000,  0.0000]]],


[[[ 0.0000,  0.0000,  0.0000,  0.0000],
  [ 0.0000,  0.0000,  0.0000,  0.0000],
  [ 0.0000,  0.0000,  0.0000,  0.0000],
  [ 0.0000,  0.0000,  0.0000,  0.0000]],

 [[ 3.0000,  1.0000,  2.0000,  3.0000],
  [ 4.0000,  0.0000,  0.0000,  0.0000],
  [ 0.0000,  0.0000,  0.0000,  0.0000],
  [ 0.0000,  0.0000,  0.0000,  0.0000]],

 [[ 2.0000,  1.0000,  1.0000,  1.0000],
  [ 0.0000,  0.0000,  0.0000,  0.0000],
  [ 0.0000,  0.0000,  0.0000,  0.0000],
  [ 0.0000,  0.0000,  0.0000,  0.0000]],
```

```
        …,

        [[ 1.0000,  0.0000,  0.0000,  0.0000],
         [ 3.0000,  0.0000,  0.0000,  1.0000],
         [ 4.0000,  0.0000,  0.0000,  0.0000],
         [ 1.0000,  0.0000,  0.0000,  0.0000]],

        [[ 2.0000,  2.0000,  2.0000,  1.0000],
         [ 6.0000,  6.0000,  6.0000,  2.0000],
         [ 4.0000,  5.0000,  5.0000,  2.0000],
         [ 1.0000,  1.0000,  1.0000,  0.0000]],

        [[ 0.0000,  0.0000,  0.0000,  0.0000],
         [ 3.0000,  2.0000,  2.0000,  0.0000],
         [ 3.0000,  2.0000,  2.0000,  0.0000],
         [ 0.0000,  0.0000,  0.0000,  0.0000]]]], device='cuda:0',
       grad_fn=<DivBackward0>)
```

[92]:
```
x_int.size()
```

[92]:
```
torch.Size([128, 8, 4, 4])
```

[93]:
```
conv_int = torch.nn.Conv2d(in_channels = 8, out_channels=8, kernel_size = 3,␣
 ↪padding=1, bias = False)
conv_int.weight = torch.nn.parameter.Parameter(weight_int)
relu = nn.ReLU()
output_int = conv_int(x_int)
output_recovered = output_int*w_delta*x_delta
output_recovered = relu(output_recovered)
```

[94]:
```
difference = abs(save_output.outputs[9][0] - output_recovered )
print(difference.mean())  ## It should be small, e.g.,2.3 in my trainned model
```

```
tensor(2.2645e-07, device='cuda:0', grad_fn=<MeanBackward0>)
```

[116]:
```
x_pad = torch.zeros(128, 8, 6, 6).cuda()
```

[117]:
```
x_pad[ : ,  :, 1:5, 1:5] = x_int.cuda()
```

[118]:
```
X = x_pad[0]
X = torch.reshape(X, (X.size(0), -1))
```

[119]:
```
X.size()
```

[119]:
```
torch.Size([8, 36])
```

[120]:
```
tile_id = 0
nij = 200 # just a random number
#X = a_tile[tile_id,:,nij:nij+64]  # [tile_num, array row num, time_steps]

bit_precision = 4
file = open('activation.txt', 'w') #write to file
file.write('#time0row7[msb-lsb],time0row6[msb-lst],....,time0row0[msb-lst]#\n')
file.write('#time1row7[msb-lsb],time1row6[msb-lst],....,time1row0[msb-lst]#\n')
file.write('#................#\n')

for i in range(X.size(1)):  # time step
    for j in range(X.size(0)): # row #
        X_bin = '{0:04b}'.format(int(X[7-j,i].item()+0.001))
        for k in range(bit_precision):
            file.write(X_bin[k])
        #file.write(' ')  # for visibility with blank between words, you can use
    file.write('\n')
file.close() #close file
```

[121]:
```
X[:,7]
```

[121]:
```
tensor([0., 5., 2., 2., 1., 1., 1., 0.], device='cuda:0',
       grad_fn=<SelectBackward>)
```

[122]:
```
weight_int.size()
```

[122]:
```
torch.Size([8, 8, 3, 3])
```

[123]:
```
weight_int.size() # 8, 8 , 3, 3
W = torch.reshape(weight_int, (weight_int.size(0), weight_int.size(1), -1))
W.size() # 8, 8, 9
```

[123]:
```
torch.Size([8, 8, 9])
```

[124]:
```
bit_precision = 4
file = open('weight.txt', 'w') #write to file
file.write('#col0row7[msb-lsb],col0row6[msb-lst],....,col0row0[msb-lst]#\n')
file.write('#col1row7[msb-lsb],col1row6[msb-lst],....,col1row0[msb-lst]#\n')
file.write('#................#\n')
for kij in range(9):
    for i in range(W.size(0)):
        for j in range(W.size(1)):
            if (W[i, 7-j, kij].item()<0):
                W_bin = '{0:04b}'.format(int(W[i,7-j,kij].
 ↪item()+2**bit_precision+0.001))
            else:
                W_bin = '{0:04b}'.format(int(W[i,7-j,kij].item()+0.001))
```

```
            for k in range(bit_precision):
                file.write(W_bin[k])
            #file.write(' ')  # for visibility with blank between words, you␣
 ↪can use
        file.write('\n')
file.close() #close file
```

[125]:
```
W[0,:,0]
```

[125]:
```
tensor([ 7.0000,  7.0000,  7.0000,  7.0000, -7.0000,  7.0000, -7.0000, -7.0000],
       device='cuda:0', grad_fn=<SelectBackward>)
```

[126]:
```
p_nijg = range(X.size(1)) ## psum nij group

psum = torch.zeros(8, len(p_nijg), 9).cuda()
```

[127]:
```
psum.size()
```

[127]:
```
torch.Size([8, 36, 9])
```

[128]:
```
for kij in range(9):
    for nij in p_nijg:        # time domain, sequentially given input
        m = nn.Linear(8, 8, bias=False)
        m.weight = torch.nn.Parameter(W[:,:,kij])
        psum[:, nij, kij] = m(X[:,nij]).cuda()
```

[129]:
```
bit_precision = 16
file = open('psum.txt', 'w') #write to file
file.write('#time0col7[msb-lsb],time0col6[msb-lst],....,time0col0[msb-lst]#\n')
file.write('#time1col7[msb-lsb],time1col6[msb-lst],....,time1col0[msb-lst]#\n')
file.write('#................#\n')
for kij in range(9):
    for i in range(psum.size(1)):
        for j in range(psum.size(0)):
            if (psum[7-j,i,kij].item()<0):
                P_bin = '{0:016b}'.format(int(psum[7-j,i,kij].
 ↪item()+2**bit_precision+0.001))
            else:
                P_bin = '{0:016b}'.format(int(psum[7-j,i,kij].item()+0.001))
            for k in range(bit_precision):
                file.write(P_bin[k])
            #file.write(' ')  # for visibility with blank between words, you␣
 ↪can use
        file.write('\n')
file.close()
```

[130]:
```
psum[:,8,0]
```

[130]:
```
tensor([ 14.0000, -28.0000,  42.0000,  14.0000, -28.0000, -14.0000, -28.0000,
        -14.0000], device='cuda:0', grad_fn=<SelectBackward>)
```

[6]:
```
address = torch.zeros(16, 9).cuda()

for o_nij in range(16):
    for kij in range(9):
        address[o_nij, kij] = int(o_nij/4)*6 + o_nij%4 + int(kij/3)*6 + kij%3
        #print(address[o_nij, kij])
```

```
tensor(0., device='cuda:0')
tensor(1., device='cuda:0')
tensor(2., device='cuda:0')
tensor(6., device='cuda:0')
tensor(7., device='cuda:0')
tensor(8., device='cuda:0')
tensor(12., device='cuda:0')
tensor(13., device='cuda:0')
tensor(14., device='cuda:0')
tensor(1., device='cuda:0')
tensor(2., device='cuda:0')
tensor(3., device='cuda:0')
tensor(7., device='cuda:0')
tensor(8., device='cuda:0')
tensor(9., device='cuda:0')
tensor(13., device='cuda:0')
tensor(14., device='cuda:0')
tensor(15., device='cuda:0')
tensor(2., device='cuda:0')
tensor(3., device='cuda:0')
tensor(4., device='cuda:0')
tensor(8., device='cuda:0')
tensor(9., device='cuda:0')
tensor(10., device='cuda:0')
tensor(14., device='cuda:0')
tensor(15., device='cuda:0')
tensor(16., device='cuda:0')
tensor(3., device='cuda:0')
tensor(4., device='cuda:0')
tensor(5., device='cuda:0')
tensor(9., device='cuda:0')
tensor(10., device='cuda:0')
tensor(11., device='cuda:0')
tensor(15., device='cuda:0')
tensor(16., device='cuda:0')
tensor(17., device='cuda:0')
tensor(6., device='cuda:0')
```

```
tensor(7., device='cuda:0')
tensor(8., device='cuda:0')
tensor(12., device='cuda:0')
tensor(13., device='cuda:0')
tensor(14., device='cuda:0')
tensor(18., device='cuda:0')
tensor(19., device='cuda:0')
tensor(20., device='cuda:0')
tensor(7., device='cuda:0')
tensor(8., device='cuda:0')
tensor(9., device='cuda:0')
tensor(13., device='cuda:0')
tensor(14., device='cuda:0')
tensor(15., device='cuda:0')
tensor(19., device='cuda:0')
tensor(20., device='cuda:0')
tensor(21., device='cuda:0')
tensor(8., device='cuda:0')
tensor(9., device='cuda:0')
tensor(10., device='cuda:0')
tensor(14., device='cuda:0')
tensor(15., device='cuda:0')
tensor(16., device='cuda:0')
tensor(20., device='cuda:0')
tensor(21., device='cuda:0')
tensor(22., device='cuda:0')
tensor(9., device='cuda:0')
tensor(10., device='cuda:0')
tensor(11., device='cuda:0')
tensor(15., device='cuda:0')
tensor(16., device='cuda:0')
tensor(17., device='cuda:0')
tensor(21., device='cuda:0')
tensor(22., device='cuda:0')
tensor(23., device='cuda:0')
tensor(12., device='cuda:0')
tensor(13., device='cuda:0')
tensor(14., device='cuda:0')
tensor(18., device='cuda:0')
tensor(19., device='cuda:0')
tensor(20., device='cuda:0')
tensor(24., device='cuda:0')
tensor(25., device='cuda:0')
tensor(26., device='cuda:0')
tensor(13., device='cuda:0')
tensor(14., device='cuda:0')
tensor(15., device='cuda:0')
tensor(19., device='cuda:0')
```

```
tensor(20., device='cuda:0')
tensor(21., device='cuda:0')
tensor(25., device='cuda:0')
tensor(26., device='cuda:0')
tensor(27., device='cuda:0')
tensor(14., device='cuda:0')
tensor(15., device='cuda:0')
tensor(16., device='cuda:0')
tensor(20., device='cuda:0')
tensor(21., device='cuda:0')
tensor(22., device='cuda:0')
tensor(26., device='cuda:0')
tensor(27., device='cuda:0')
tensor(28., device='cuda:0')
tensor(15., device='cuda:0')
tensor(16., device='cuda:0')
tensor(17., device='cuda:0')
tensor(21., device='cuda:0')
tensor(22., device='cuda:0')
tensor(23., device='cuda:0')
tensor(27., device='cuda:0')
tensor(28., device='cuda:0')
tensor(29., device='cuda:0')
tensor(18., device='cuda:0')
tensor(19., device='cuda:0')
tensor(20., device='cuda:0')
tensor(24., device='cuda:0')
tensor(25., device='cuda:0')
tensor(26., device='cuda:0')
tensor(30., device='cuda:0')
tensor(31., device='cuda:0')
tensor(32., device='cuda:0')
tensor(19., device='cuda:0')
tensor(20., device='cuda:0')
tensor(21., device='cuda:0')
tensor(25., device='cuda:0')
tensor(26., device='cuda:0')
tensor(27., device='cuda:0')
tensor(31., device='cuda:0')
tensor(32., device='cuda:0')
tensor(33., device='cuda:0')
tensor(20., device='cuda:0')
tensor(21., device='cuda:0')
tensor(22., device='cuda:0')
tensor(26., device='cuda:0')
tensor(27., device='cuda:0')
tensor(28., device='cuda:0')
tensor(32., device='cuda:0')
```

```
tensor(33., device='cuda:0')
tensor(34., device='cuda:0')
tensor(21., device='cuda:0')
tensor(22., device='cuda:0')
tensor(23., device='cuda:0')
tensor(27., device='cuda:0')
tensor(28., device='cuda:0')
tensor(29., device='cuda:0')
tensor(33., device='cuda:0')
tensor(34., device='cuda:0')
tensor(35., device='cuda:0')
```

[5]:
```
address.size()
```

[5]:
```
torch.Size([16, 9])
```

[8]:
```
file = open('acc_address.txt', 'w') #write to file
file.write('#1st address#\n')
file.write('#2st address#\n')
file.write('#.............#\n')
bit_precision = 11

for i in range(address.size(0)):
    for j in range(address.size(1)):
        a_bin = '{0:011b}'.format(int(address[i, j]))
        for k in range(bit_precision):
            file.write(a_bin[k])
        file.write('\n')
file.close()
```

[131]:
```
output_int.size()
```

[131]:
```
torch.Size([128, 8, 4, 4])
```

[132]:
```
out = output_int[0]
out.size()
```

[132]:
```
torch.Size([8, 4, 4])
```

[133]:
```
out = torch.reshape(out, (out.size(0), -1))
```

[134]:
```
out.size()
```

[134]:
```
torch.Size([8, 16])
```

[135]:
```
bit_precision = 16
file = open('output.txt', 'w') #write to file
```

```
file.write('#time0col7[msb-lsb],time0col6[msb-lst],....,time0col0[msb-lst]#\n')
file.write('#time1col7[msb-lsb],time1col6[msb-lst],....,time1col0[msb-lst]#\n')
file.write('#...............#\n')

for i in range(out.size(1)):
    for j in range(out.size(0)):
        if (out[7-j,i].item()<0):
            O_bin = '{0:016b}'.format(int(out[7-j,i].item()+2**bit_precision+0.
 ↪001))
        else:
            O_bin = '{0:016b}'.format(int(out[7-j,i].item()+0.001))
        for k in range(bit_precision):
            file.write(O_bin[k])
        #file.write(' ')  # for visibility with blank between words, you can use
    file.write('\n')
file.close()
```

[136]:
```
out[:,4]
```

[136]:
```
tensor([ -56.0000,  -20.0000,   84.0000, -112.0000,   84.0000,  -76.0000,
        -126.0000,  140.0000], device='cuda:0', grad_fn=<SelectBackward>)
```

[ ]:

[ ]:

[ ]:

[ ]: