HW5-1 (4-bit)

November 9, 2023

0.1 Resnet20 Model

0.1.1 Quantization-aware training with 4 bits

[1]:
```
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 = "resnet20_quant"
model = resnet20_quant()
print(model)
# print(model.layer1[0].conv1.weight_q.shape) #tuple: (3,3)
# print(model.conv1.kernel_size)

for name, module in model.named_modules():
    if isinstance(module, QuantConv2d):
        module.bit = 4
        print(f"Layer name: {name}, bit: {module.bit}")
```

```
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 = [150, 225]
    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…
ResNet_Cifar(
  (conv1): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
  (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (relu): ReLU(inplace=True)
  (layer1): Sequential(
    (0): BasicBlock(
      (conv1): QuantConv2d(
        16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
```

```
        (weight_quant): weight_quantize_fn()
      )
      (conv2): QuantConv2d(
        16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (bn2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
    (1): BasicBlock(
      (conv1): QuantConv2d(
        16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (conv2): QuantConv2d(
        16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (bn2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
    (2): BasicBlock(
      (conv1): QuantConv2d(
        16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (conv2): QuantConv2d(
        16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (bn2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (layer2): Sequential(
    (0): BasicBlock(
      (conv1): QuantConv2d(
        16, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
```

```
      )
      (conv2): QuantConv2d(
        32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (bn1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (bn2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): QuantConv2d(
          16, 32, kernel_size=(1, 1), stride=(2, 2), bias=False
          (weight_quant): weight_quantize_fn()
        )
        (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): QuantConv2d(
        32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (conv2): QuantConv2d(
        32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (bn1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (bn2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
    (2): BasicBlock(
      (conv1): QuantConv2d(
        32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (conv2): QuantConv2d(
        32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (bn1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (bn2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
```

```
    )
  )
  (layer3): Sequential(
    (0): BasicBlock(
      (conv1): QuantConv2d(
        32, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (conv2): QuantConv2d(
        64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): QuantConv2d(
          32, 64, kernel_size=(1, 1), stride=(2, 2), bias=False
          (weight_quant): weight_quantize_fn()
        )
        (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): QuantConv2d(
        64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (conv2): QuantConv2d(
        64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
    (2): BasicBlock(
      (conv1): QuantConv2d(
        64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (conv2): QuantConv2d(
        64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
```

```
        (weight_quant): weight_quantize_fn()
      )
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (avgpool): AvgPool2d(kernel_size=8, stride=1, padding=0)
  (fc): Linear(in_features=64, out_features=10, bias=True)
)
Layer name: layer1.0.conv1, bit: 4
Layer name: layer1.0.conv2, bit: 4
Layer name: layer1.1.conv1, bit: 4
Layer name: layer1.1.conv2, bit: 4
Layer name: layer1.2.conv1, bit: 4
Layer name: layer1.2.conv2, bit: 4
Layer name: layer2.0.conv1, bit: 4
Layer name: layer2.0.conv2, bit: 4
Layer name: layer2.0.downsample.0, bit: 4
Layer name: layer2.1.conv1, bit: 4
Layer name: layer2.1.conv2, bit: 4
Layer name: layer2.2.conv1, bit: 4
Layer name: layer2.2.conv2, bit: 4
Layer name: layer3.0.conv1, bit: 4
Layer name: layer3.0.conv2, bit: 4
Layer name: layer3.0.downsample.0, bit: 4
Layer name: layer3.1.conv1, bit: 4
Layer name: layer3.1.conv2, bit: 4
Layer name: layer3.2.conv1, bit: 4
Layer name: layer3.2.conv2, bit: 4
Files already downloaded and verified
Files already downloaded and verified
```

[ ]:
```
#training
model_name = "resnet20_quant"
model = resnet20_quant()

lr = 2.4e-2
weight_decay = 9e-5
epochs = 200
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)+str("4-bit")
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)
```

[2]:
```
#testing
model_name = "resnet20_quant"
model = resnet20_quant()

PATH = "result/resnet20_quant4-bit/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)))
```

```
Test set: Accuracy: 9114/10000 (91%)
```

[ ]:

[3]:
```
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()

# Register hooks for the first and second convolutional layers
for layer in model.modules():
    if isinstance(layer, torch.nn.Conv2d):
        print("prehooked")
        layer.register_forward_pre_hook(save_output)
####################################################
use_gpu = torch.cuda.is_available()
print(use_gpu)
device = torch.device("cuda" if use_gpu else "cpu")


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

```
prehooked
prehooked
```

```
prehooked
prehooked
prehooked
prehooked
prehooked
prehooked
prehooked
prehooked
prehooked
prehooked
prehooked
prehooked
prehooked
prehooked
prehooked
prehooked
prehooked
prehooked
True
```

[4]:
```
model
```

[4]:
```
ResNet_Cifar(
  (conv1): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
  (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (relu): ReLU(inplace=True)
  (layer1): Sequential(
    (0): BasicBlock(
      (conv1): QuantConv2d(
        16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (conv2): QuantConv2d(
        16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (bn2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
    (1): BasicBlock(
      (conv1): QuantConv2d(
```

```
        16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (conv2): QuantConv2d(
        16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (bn2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
    (2): BasicBlock(
      (conv1): QuantConv2d(
        16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (conv2): QuantConv2d(
        16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (bn2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (layer2): Sequential(
    (0): BasicBlock(
      (conv1): QuantConv2d(
        16, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (conv2): QuantConv2d(
        32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (bn1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (bn2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): QuantConv2d(
          16, 32, kernel_size=(1, 1), stride=(2, 2), bias=False
```

```
          (weight_quant): weight_quantize_fn()
        )
        (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): QuantConv2d(
        32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (conv2): QuantConv2d(
        32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (bn1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (bn2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
    (2): BasicBlock(
      (conv1): QuantConv2d(
        32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (conv2): QuantConv2d(
        32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (bn1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (bn2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (layer3): Sequential(
    (0): BasicBlock(
      (conv1): QuantConv2d(
        32, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (conv2): QuantConv2d(
        64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
```

```
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): QuantConv2d(
          32, 64, kernel_size=(1, 1), stride=(2, 2), bias=False
          (weight_quant): weight_quantize_fn()
        )
        (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): QuantConv2d(
        64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (conv2): QuantConv2d(
        64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
    (2): BasicBlock(
      (conv1): QuantConv2d(
        64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (conv2): QuantConv2d(
        64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (weight_quant): weight_quantize_fn()
      )
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (avgpool): AvgPool2d(kernel_size=8, stride=1, padding=0)
  (fc): Linear(in_features=64, out_features=10, bias=True)
```

```
)
```

[5]:
```
w_bit = 4
weight_q = model.layer1[0].conv1.weight_q # quantized value is stored during␣
 ↪the training
w_alpha = model.layer1[0].conv1.weight_quant.wgt_alpha   # alpha is defined in␣
 ↪your model already. bring it out here
print(w_alpha)
w_delta = w_alpha / (2 ** (w_bit - 1) - 1)  # delta can be calculated by using␣
 ↪alpha and w_bit
weight_int = weight_q / w_delta # w_int can be calculated by weight_q and␣
 ↪w_delta
print(weight_int) # you should see clean integer numbers
```

```
Parameter containing:
tensor(2.4811, device='cuda:0', requires_grad=True)
tensor([[[[-3.0000,  1.0000,  1.0000],
          [-1.0000,  2.0000,  1.0000],
          [ 1.0000,  1.0000,  2.0000]],

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

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

         …,

         [[-6.0000,  2.0000,  0.0000],
          [ 2.0000, -7.0000,  6.0000],
          [-0.0000, -2.0000,  0.0000]],

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

         [[-3.0000, -3.0000,  3.0000],
          [-3.0000,  2.0000,  3.0000],
          [ 2.0000,  1.0000,  1.0000]]],


        [[[ 0.0000,  1.0000,  1.0000],
          [-0.0000, -1.0000,  2.0000],
          [-0.0000, -1.0000,  4.0000]],
```

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

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

 …,

 [[ 4.0000,  0.0000,  2.0000],
  [ 5.0000, -7.0000, -5.0000],
  [-4.0000, -1.0000,  1.0000]],

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

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


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

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

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

 …,

 [[-2.0000,  3.0000,  5.0000],
  [ 0.0000,  7.0000,  2.0000],
  [-6.0000, -4.0000, -4.0000]],

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

 [[-3.0000,  1.0000,  0.0000],
  [-2.0000,  3.0000, -1.0000],
  [-1.0000,  3.0000, -1.0000]]],
```

```
…,


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

 [[-1.0000,  1.0000,  4.0000],
  [ 1.0000, -1.0000,  1.0000],
  [-2.0000,  2.0000, -2.0000]],

 [[-1.0000,  1.0000,  0.0000],
  [ 6.0000, -2.0000,  2.0000],
  [-2.0000, -7.0000,  0.0000]],

 …,

 [[ 0.0000, -0.0000, -4.0000],
  [-6.0000, -1.0000, -1.0000],
  [ 7.0000,  7.0000,  4.0000]],

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

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


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

 [[-1.0000, -1.0000,  1.0000],
  [-1.0000, -3.0000, -1.0000],
  [ 1.0000,  1.0000,  2.0000]],

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

 …,

 [[-4.0000,  3.0000,  2.0000],
  [-1.0000,  1.0000, -1.0000],
```

```
          [-3.0000, -2.0000, -1.0000]],

         [[-1.0000, -1.0000, -2.0000],
          [ 2.0000,  1.0000, -1.0000],
          [ 1.0000,  2.0000, -1.0000]],

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


        [[[-1.0000, -1.0000, -1.0000],
          [ 1.0000,  5.0000,  1.0000],
          [-2.0000,  4.0000, -1.0000]],

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

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

         …,

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

         [[-1.0000, -2.0000, -2.0000],
          [-1.0000, -3.0000, -2.0000],
          [-1.0000, -2.0000, -1.0000]],

         [[-1.0000,  2.0000,  2.0000],
          [-2.0000,  2.0000,  1.0000],
          [-1.0000,  2.0000,  2.0000]]]], device='cuda:0',
       grad_fn=<DivBackward0>)
```

[6]:
```
x_bit = 4
x = save_output.outputs[1][0]   # input of the 2nd conv layer
x_alpha  = model.layer1[0].conv1.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([[[[ 3.0000,  6.0000,  6.0000,  …,  6.0000,  6.0000,  6.0000],
          [ 3.0000,  4.0000,  4.0000,  …,  4.0000,  4.0000,  5.0000],
          [ 3.0000,  4.0000,  4.0000,  …,  4.0000,  4.0000,  5.0000],
          …,
          [ 0.0000,  0.0000,  0.0000,  …,  1.0000,  0.0000,  2.0000],
          [ 0.0000,  0.0000,  0.0000,  …,  1.0000,  0.0000,  2.0000],
          [ 0.0000,  0.0000,  0.0000,  …,  2.0000,  0.0000,  4.0000]],

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

         [[12.0000, 11.0000, 11.0000,  …, 11.0000, 11.0000, 11.0000],
          [ 1.0000,  4.0000,  4.0000,  …,  4.0000,  4.0000,  2.0000],
          [ 1.0000,  4.0000,  4.0000,  …,  4.0000,  4.0000,  2.0000],
          …,
          [ 4.0000,  4.0000,  4.0000,  …,  7.0000, 11.0000,  0.0000],
          [ 5.0000,  6.0000,  6.0000,  …,  6.0000, 10.0000,  0.0000],
          [ 5.0000,  7.0000,  7.0000,  …, 11.0000, 15.0000,  0.0000]],

         …,

         [[ 0.0000,  0.0000,  0.0000,  …,  0.0000,  0.0000, 10.0000],
          [ 3.0000,  3.0000,  3.0000,  …,  3.0000,  3.0000, 13.0000],
          [ 3.0000,  3.0000,  3.0000,  …,  3.0000,  3.0000, 13.0000],
          …,
          [ 1.0000,  0.0000,  0.0000,  …,  7.0000,  0.0000, 14.0000],
          [ 0.0000,  0.0000,  0.0000,  …,  6.0000,  0.0000, 14.0000],
          [ 0.0000,  0.0000,  0.0000,  …,  5.0000,  0.0000, 15.0000]],

         [[ 2.0000,  7.0000,  7.0000,  …,  7.0000,  7.0000, 10.0000],
          [ 1.0000,  4.0000,  4.0000,  …,  4.0000,  4.0000,  8.0000],
          [ 1.0000,  4.0000,  4.0000,  …,  4.0000,  4.0000,  8.0000],
          …,
          [ 0.0000,  0.0000,  0.0000,  …,  0.0000,  0.0000,  2.0000],
          [ 0.0000,  0.0000,  0.0000,  …,  0.0000,  0.0000,  3.0000],
          [ 0.0000,  0.0000,  0.0000,  …,  0.0000,  0.0000,  2.0000]],

         [[ 9.0000,  4.0000,  4.0000,  …,  4.0000,  4.0000,  1.0000],
          [11.0000,  6.0000,  6.0000,  …,  6.0000,  6.0000,  2.0000],
          [11.0000,  6.0000,  6.0000,  …,  6.0000,  6.0000,  2.0000],
```

```
  …,
   [ 0.0000,  0.0000,  0.0000,  …,  0.0000,  2.0000,  9.0000],
   [ 0.0000,  0.0000,  0.0000,  …,  0.0000,  3.0000,  9.0000],
   [ 0.0000,  0.0000,  0.0000,  …,  0.0000,  6.0000,  7.0000]]],


[[[ 3.0000,  6.0000,  6.0000,  …,  6.0000,  6.0000,  6.0000],
   [ 0.0000,  0.0000,  0.0000,  …,  4.0000,  4.0000,  5.0000],
   [ 0.0000,  3.0000,  3.0000,  …,  4.0000,  4.0000,  5.0000],
   …,
   [ 0.0000,  0.0000,  0.0000,  …,  4.0000,  4.0000,  5.0000],
   [ 0.0000,  0.0000,  0.0000,  …,  4.0000,  4.0000,  5.0000],
   [ 0.0000,  0.0000,  0.0000,  …,  4.0000,  4.0000,  6.0000]],

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

  [[ 8.0000, 12.0000, 12.0000,  …, 11.0000, 11.0000, 11.0000],
   [ 0.0000,  0.0000,  0.0000,  …,  4.0000,  4.0000,  2.0000],
   [11.0000,  7.0000,  8.0000,  …,  4.0000,  4.0000,  2.0000],
   …,
   [ 3.0000,  4.0000,  4.0000,  …,  4.0000,  4.0000,  2.0000],
   [ 4.0000,  4.0000,  4.0000,  …,  4.0000,  4.0000,  2.0000],
   [ 5.0000,  5.0000,  5.0000,  …,  3.0000,  3.0000,  0.0000]],

  …,

  [[ 0.0000,  0.0000,  0.0000,  …,  0.0000,  0.0000, 10.0000],
   [15.0000, 15.0000, 15.0000,  …,  3.0000,  3.0000, 13.0000],
   [ 0.0000,  0.0000,  0.0000,  …,  3.0000,  3.0000, 13.0000],
   …,
   [ 5.0000,  5.0000,  6.0000,  …,  3.0000,  3.0000, 13.0000],
   [ 5.0000,  5.0000,  6.0000,  …,  3.0000,  3.0000, 13.0000],
   [ 5.0000,  5.0000,  5.0000,  …,  4.0000,  4.0000, 15.0000]],

  [[ 6.0000, 13.0000, 13.0000,  …,  7.0000,  7.0000, 10.0000],
   [ 0.0000,  0.0000,  0.0000,  …,  4.0000,  4.0000,  8.0000],
   [ 0.0000,  0.0000,  0.0000,  …,  4.0000,  4.0000,  8.0000],
   …,
   [ 0.0000,  0.0000,  0.0000,  …,  4.0000,  4.0000,  8.0000],
   [ 0.0000,  0.0000,  0.0000,  …,  4.0000,  4.0000,  8.0000],
   [ 0.0000,  0.0000,  0.0000,  …,  7.0000,  7.0000,  7.0000]],
```

```
 [[ 9.0000,  9.0000,  9.0000,  …,  4.0000,  4.0000,  1.0000],
  [ 0.0000,  0.0000,  0.0000,  …,  6.0000,  6.0000,  2.0000],
  [ 0.0000,  0.0000,  0.0000,  …,  6.0000,  6.0000,  2.0000],
  …,
  [ 0.0000,  0.0000,  0.0000,  …,  6.0000,  6.0000,  2.0000],
  [ 0.0000,  0.0000,  0.0000,  …,  6.0000,  6.0000,  2.0000],
  [ 0.0000,  0.0000,  0.0000,  …,  8.0000,  8.0000,  2.0000]]],


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

 [[ 2.0000,  0.0000,  0.0000,  …,  0.0000,  0.0000,  0.0000],
  [ 9.0000,  6.0000,  6.0000,  …,  5.0000,  2.0000,  6.0000],
  [ 9.0000,  6.0000,  6.0000,  …,  6.0000,  6.0000,  6.0000],
  …,
  [ 1.0000,  4.0000,  4.0000,  …,  4.0000,  4.0000,  2.0000],
  [ 1.0000,  4.0000,  4.0000,  …,  4.0000,  4.0000,  2.0000],
  [ 0.0000,  3.0000,  3.0000,  …,  3.0000,  3.0000,  0.0000]],

 …,

 [[ 9.0000,  9.0000, 10.0000,  …, 10.0000, 13.0000,  0.0000],
  [ 6.0000,  6.0000,  6.0000,  …,  5.0000,  4.0000,  0.0000],
  [ 7.0000,  5.0000,  4.0000,  …,  5.0000,  2.0000,  1.0000],
  …,
  [ 3.0000,  3.0000,  3.0000,  …,  3.0000,  3.0000, 13.0000],
  [ 3.0000,  3.0000,  3.0000,  …,  3.0000,  3.0000, 13.0000],
  [ 4.0000,  4.0000,  4.0000,  …,  4.0000,  4.0000, 15.0000]],

 [[ 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],
  …,
  [ 1.0000,  4.0000,  4.0000,  …,  4.0000,  4.0000,  8.0000],
```

```
  [ 1.0000,  4.0000,  4.0000,  …,  4.0000,  4.0000,  8.0000],
  [ 3.0000,  7.0000,  7.0000,  …,  7.0000,  7.0000,  7.0000]],

 [[ 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],
  …,
  [11.0000,  6.0000,  6.0000,  …,  6.0000,  6.0000,  2.0000],
  [11.0000,  6.0000,  6.0000,  …,  6.0000,  6.0000,  2.0000],
  [12.0000,  8.0000,  8.0000,  …,  8.0000,  8.0000,  2.0000]]],


…,


[[[ 0.0000,  0.0000,  0.0000,  …,  6.0000,  6.0000,  6.0000],
  [ 0.0000,  0.0000,  0.0000,  …,  4.0000,  4.0000,  5.0000],
  [ 0.0000,  0.0000,  0.0000,  …,  4.0000,  4.0000,  5.0000],
  …,
  [ 0.0000,  0.0000,  0.0000,  …,  4.0000,  4.0000,  5.0000],
  [ 0.0000,  0.0000,  0.0000,  …,  4.0000,  4.0000,  5.0000],
  [ 1.0000,  3.0000,  3.0000,  …,  4.0000,  4.0000,  6.0000]],

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

 [[ 0.0000,  0.0000,  0.0000,  …, 11.0000, 11.0000, 11.0000],
  [ 3.0000,  3.0000,  4.0000,  …,  4.0000,  4.0000,  2.0000],
  [ 3.0000,  3.0000,  3.0000,  …,  4.0000,  4.0000,  2.0000],
  …,
  [ 7.0000,  2.0000,  1.0000,  …,  4.0000,  4.0000,  2.0000],
  [15.0000, 13.0000, 11.0000,  …,  4.0000,  4.0000,  2.0000],
  [15.0000, 15.0000, 15.0000,  …,  3.0000,  3.0000,  0.0000]],

 …,

 [[14.0000, 14.0000, 15.0000,  …,  0.0000,  0.0000, 10.0000],
  [ 7.0000,  8.0000,  8.0000,  …,  3.0000,  3.0000, 13.0000],
  [ 7.0000,  8.0000,  8.0000,  …,  3.0000,  3.0000, 13.0000],
  …,
  [ 3.0000,  2.0000,  3.0000,  …,  3.0000,  3.0000, 13.0000],
  [ 2.0000,  0.0000,  1.0000,  …,  3.0000,  3.0000, 13.0000],
  [ 0.0000,  0.0000,  0.0000,  …,  4.0000,  4.0000, 15.0000]],
```

```
 [[ 0.0000,  0.0000,  0.0000,  …,  7.0000,  7.0000, 10.0000],
  [ 0.0000,  0.0000,  0.0000,  …,  4.0000,  4.0000,  8.0000],
  [ 0.0000,  0.0000,  0.0000,  …,  4.0000,  4.0000,  8.0000],
  …,
  [ 0.0000,  0.0000,  0.0000,  …,  4.0000,  4.0000,  8.0000],
  [ 0.0000,  0.0000,  0.0000,  …,  4.0000,  4.0000,  8.0000],
  [ 5.0000, 11.0000, 11.0000,  …,  7.0000,  7.0000,  7.0000]],

 [[ 0.0000,  0.0000,  0.0000,  …,  4.0000,  4.0000,  1.0000],
  [ 0.0000,  0.0000,  0.0000,  …,  6.0000,  6.0000,  2.0000],
  [ 0.0000,  0.0000,  0.0000,  …,  6.0000,  6.0000,  2.0000],
  …,
  [ 0.0000,  0.0000,  0.0000,  …,  6.0000,  6.0000,  2.0000],
  [ 0.0000,  0.0000,  0.0000,  …,  6.0000,  6.0000,  2.0000],
  [ 9.0000,  7.0000,  7.0000,  …,  8.0000,  8.0000,  2.0000]]],


[[[ 2.0000,  2.0000,  2.0000,  …,  6.0000,  6.0000,  6.0000],
  [ 2.0000,  3.0000,  3.0000,  …,  4.0000,  4.0000,  5.0000],
  [ 3.0000,  4.0000,  4.0000,  …,  4.0000,  4.0000,  5.0000],
  …,
  [ 2.0000,  7.0000,  7.0000,  …,  4.0000,  4.0000,  5.0000],
  [ 3.0000,  4.0000,  4.0000,  …,  4.0000,  4.0000,  5.0000],
  [ 3.0000,  4.0000,  4.0000,  …,  4.0000,  4.0000,  6.0000]],

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

 [[ 7.0000,  6.0000,  6.0000,  …, 11.0000, 11.0000, 11.0000],
  [ 6.0000,  6.0000,  6.0000,  …,  4.0000,  4.0000,  2.0000],
  [ 5.0000,  6.0000,  5.0000,  …,  4.0000,  4.0000,  2.0000],
  …,
  [15.0000, 15.0000, 15.0000,  …,  4.0000,  4.0000,  2.0000],
  [ 1.0000,  4.0000,  4.0000,  …,  4.0000,  4.0000,  2.0000],
  [ 0.0000,  3.0000,  3.0000,  …,  3.0000,  3.0000,  0.0000]],

 …,

 [[ 3.0000,  3.0000,  3.0000,  …,  0.0000,  0.0000, 10.0000],
  [ 3.0000,  3.0000,  3.0000,  …,  3.0000,  3.0000, 13.0000],
  [ 4.0000,  4.0000,  4.0000,  …,  3.0000,  3.0000, 13.0000],
  …,
```

```
  [ 0.0000,  0.0000,  0.0000,  …,  3.0000,  3.0000, 13.0000],
  [ 3.0000,  3.0000,  3.0000,  …,  3.0000,  3.0000, 13.0000],
  [ 4.0000,  4.0000,  4.0000,  …,  4.0000,  4.0000, 15.0000]],

 [[ 0.0000,  0.0000,  0.0000,  …,  7.0000,  7.0000, 10.0000],
  [ 0.0000,  0.0000,  0.0000,  …,  4.0000,  4.0000,  8.0000],
  [ 0.0000,  1.0000,  0.0000,  …,  4.0000,  4.0000,  8.0000],
  …,
  [ 3.0000, 10.0000, 10.0000,  …,  4.0000,  4.0000,  8.0000],
  [ 1.0000,  4.0000,  4.0000,  …,  4.0000,  4.0000,  8.0000],
  [ 3.0000,  7.0000,  7.0000,  …,  7.0000,  7.0000,  7.0000]],

 [[ 1.0000,  0.0000,  0.0000,  …,  4.0000,  4.0000,  1.0000],
  [ 4.0000,  2.0000,  1.0000,  …,  6.0000,  6.0000,  2.0000],
  [ 6.0000,  3.0000,  2.0000,  …,  6.0000,  6.0000,  2.0000],
  …,
  [ 6.0000,  3.0000,  3.0000,  …,  6.0000,  6.0000,  2.0000],
  [11.0000,  6.0000,  6.0000,  …,  6.0000,  6.0000,  2.0000],
  [12.0000,  8.0000,  8.0000,  …,  8.0000,  8.0000,  2.0000]]],


[[[ 3.0000,  6.0000,  6.0000,  …,  6.0000,  6.0000,  7.0000],
  [ 3.0000,  4.0000,  4.0000,  …,  0.0000,  0.0000,  0.0000],
  [ 3.0000,  4.0000,  4.0000,  …,  0.0000,  0.0000,  0.0000],
  …,
  [ 3.0000,  4.0000,  4.0000,  …,  0.0000,  3.0000,  0.0000],
  [ 3.0000,  4.0000,  4.0000,  …,  1.0000,  1.0000,  0.0000],
  [ 3.0000,  4.0000,  4.0000,  …,  0.0000,  0.0000,  0.0000]],

 [[ 0.0000,  2.0000,  2.0000,  …,  0.0000,  0.0000,  0.0000],
  [ 0.0000,  4.0000,  4.0000,  …,  0.0000,  0.0000,  0.0000],
  [ 0.0000,  4.0000,  4.0000,  …,  0.0000,  0.0000,  0.0000],
  …,
  [ 0.0000,  4.0000,  4.0000,  …,  0.0000,  0.0000,  0.0000],
  [ 0.0000,  4.0000,  4.0000,  …,  0.0000,  0.0000,  0.0000],
  [ 0.0000,  1.0000,  1.0000,  …,  0.0000,  0.0000,  0.0000]],

 [[12.0000, 11.0000, 11.0000,  …, 13.0000, 13.0000,  7.0000],
  [ 1.0000,  4.0000,  4.0000,  …,  0.0000,  0.0000,  0.0000],
  [ 1.0000,  4.0000,  4.0000,  …,  2.0000,  3.0000,  5.0000],
  …,
  [ 1.0000,  4.0000,  4.0000,  …,  7.0000,  1.0000,  7.0000],
  [ 1.0000,  4.0000,  4.0000,  …,  0.0000,  0.0000,  7.0000],
  [ 0.0000,  3.0000,  3.0000,  …,  0.0000,  4.0000,  8.0000]],

 …,

 [[ 0.0000,  0.0000,  0.0000,  …,  0.0000,  0.0000, 15.0000],
```

```
          [ 3.0000,  3.0000,  3.0000,  …, 15.0000, 15.0000,  4.0000],
          [ 3.0000,  3.0000,  3.0000,  …,  8.0000,  8.0000,  0.0000],
          …,
          [ 3.0000,  3.0000,  3.0000,  …,  5.0000, 12.0000,  2.0000],
          [ 3.0000,  3.0000,  3.0000,  …, 14.0000, 11.0000,  0.0000],
          [ 4.0000,  4.0000,  4.0000,  …,  8.0000,  5.0000,  0.0000]],

         [[ 2.0000,  7.0000,  7.0000,  …, 13.0000, 13.0000,  9.0000],
          [ 1.0000,  4.0000,  4.0000,  …,  0.0000,  0.0000,  0.0000],
          [ 1.0000,  4.0000,  4.0000,  …,  0.0000,  0.0000,  0.0000],
          …,
          [ 1.0000,  4.0000,  4.0000,  …,  0.0000,  0.0000,  0.0000],
          [ 1.0000,  4.0000,  4.0000,  …,  0.0000,  0.0000,  0.0000],
          [ 3.0000,  7.0000,  7.0000,  …,  0.0000,  0.0000,  0.0000]],

         [[ 9.0000,  4.0000,  4.0000,  …,  9.0000,  9.0000,  2.0000],
          [11.0000,  6.0000,  6.0000,  …,  0.0000,  0.0000,  0.0000],
          [11.0000,  6.0000,  6.0000,  …,  0.0000,  0.0000,  0.0000],
          …,
          [11.0000,  6.0000,  6.0000,  …,  3.0000,  0.0000,  0.0000],
          [11.0000,  6.0000,  6.0000,  …,  0.0000,  0.0000,  0.0000],
          [12.0000,  8.0000,  8.0000,  …,  0.0000,  0.0000,  0.0000]]]],
       device='cuda:0', grad_fn=<DivBackward0>)
```

[7]:
```
conv_int = torch.nn.Conv2d(in_channels = 16, out_channels=16, kernel_size = 3,␣
 ↪bias = False)
conv_int.weight = torch.nn.parameter.Parameter(weight_int)


output_int = conv_int(x_int)    # output_int can be calculated with conv_int␣
 ↪and x_int
output_recovered = output_int * w_delta * x_delta  # recover with x_delta and␣
 ↪w_delta
print(output_recovered)
```

```
tensor([[[[ 5.1161e+00,  1.4211e+00,  1.4211e+00,  …,  1.4211e+00,
            1.4211e+00,  5.9687e+00],
          [-7.1056e-02, -1.1369e+00, -1.0658e+00,  …,  8.8094e-07,
           -3.6239e+00,  6.5372e+00],
          [-1.9185e+00,  7.1056e-01,  1.4922e+00,  …, -1.4211e+00,
            5.3292e+00,  1.3501e+00],
          …,
          [-5.3292e+00,  1.6343e+00, -7.4609e+00,  …,  1.2080e+00,
           -1.0943e+01,  2.4799e+01],
          [-8.5978e+00, -4.9739e+00,  4.6897e+00,  …,  3.6239e+00,
           -1.1582e+01,  2.4088e+01],
          [-4.5476e+00, -8.6689e+00, -3.6949e+00,  …,  8.5268e-01,
```

```
  -1.0161e+01,  1.7764e+01]],

[[ 1.1369e+00, -2.8423e-01, -2.8423e-01,  …, -2.8423e-01,
  -2.8423e-01, -6.3951e+00],
 [-2.1317e-01,  1.4211e-01,  4.2634e-01,  …,  1.2790e+00,
   4.6897e+00, -7.8162e-01],
 [ 7.9583e+00,  8.4557e+00,  9.4505e+00,  …,  8.8820e+00,
   1.0374e+01, -1.7480e+01],
 …,
 [ 1.4922e+00,  1.0161e+01, -1.9896e+00,  …, -1.4993e+01,
  -3.0554e+00,  2.0535e+01],
 [-5.1871e+00,  1.0516e+01,  7.3188e+00,  …, -1.0943e+01,
  -6.8925e+00,  1.4567e+01],
 [-4.1923e+00, -3.4818e+00, -1.8974e-06,  …, -8.0294e+00,
  -8.1715e+00,  1.0658e+01]],

[[ 1.3501e+00,  6.7503e+00,  6.7503e+00,  …,  6.7503e+00,
   6.7503e+00,  7.8162e-01],
 [-2.0606e+00,  4.3344e+00,  3.9081e+00,  …,  3.3396e+00,
   5.2582e+00, -5.0450e+00],
 [-1.1440e+01, -1.2293e+01, -1.3003e+01,  …, -1.4282e+01,
  -1.1795e+01, -1.6982e+01],
 …,
 [ 8.3136e+00,  3.2686e+00,  4.9029e+00,  …, -6.1108e+00,
  -3.6239e+00, -2.4657e+01],
 [ 6.7503e+00,  7.7451e+00,  4.6897e+00,  …, -5.9687e+00,
  -3.6949e+00, -2.6504e+01],
 [-1.6343e+00,  5.0450e+00,  7.4609e+00,  …, -4.5476e+00,
  -1.7764e+00, -1.6059e+01]],

…,

[[ 5.2582e+00,  4.6897e+00,  4.6897e+00,  …,  4.6897e+00,
   4.6897e+00,  9.3084e+00],
 [ 4.2634e-01, -1.2080e+00, -1.4922e+00,  …,  1.4211e-01,
   4.4055e+00,  4.3344e+00],
 [ 3.3041e+01,  3.2189e+01,  3.1336e+01,  …,  2.8423e+01,
   2.4941e+01,  3.8299e+01],
 …,
 [-3.8370e+00, -9.9479e-01,  4.1923e+00,  …, -1.4638e+01,
  -1.1227e+01, -1.4567e+01],
 [-5.8266e+00, -1.0019e+01,  9.9479e-01,  …, -1.5490e+01,
  -5.2582e+00, -1.4424e+01],
 [-9.8768e+00, -1.0090e+01, -1.1795e+01,  …, -8.8110e+00,
  -3.6239e+00, -7.8872e+00]],

[[-6.8925e+00, -3.6239e+00, -3.6239e+00,  …, -3.6239e+00,
  -3.6239e+00,  2.1317e+00],
```

```
  [-4.4765e+00,  4.9740e-01,  1.1369e+00,  …, -2.4159e+00,
    4.4055e+00,  1.5632e+00],
  [ 4.6897e+00,  7.2477e+00,  8.1004e+00,  …,  6.4661e+00,
    1.7267e+01,  1.1298e+01],
  …,
  [ 6.1819e+00,  1.0161e+01,  1.2719e+01,  …, -4.8318e+00,
   -9.3794e+00, -1.9043e+01],
  [ 2.7712e+00,  3.9081e+00,  1.0943e+01,  …, -2.3449e+00,
   -7.1767e+00, -2.0109e+01],
  [-2.8422e-01,  2.2027e+00,  1.0801e+01,  …, -2.6291e+00,
   -8.7399e+00, -1.8972e+01]],

 [[-1.0801e+01, -9.5215e+00, -9.5215e+00,  …, -9.5215e+00,
   -9.5215e+00, -1.2648e+01],
  [-8.4557e+00, -8.1715e+00, -8.2425e+00,  …, -7.2477e+00,
   -8.6689e+00, -9.6637e+00],
  [-2.0891e+01, -1.8830e+01, -1.8261e+01,  …, -1.7693e+01,
   -1.6698e+01, -1.3998e+01],
  …,
  [-1.4424e+01, -1.3359e+01, -1.4993e+01,  …,  1.3714e+01,
    3.6949e+00, -3.9792e+00],
  [-1.4424e+01, -1.4424e+01, -1.2222e+01,  …,  1.4211e+01,
    2.7712e+00, -6.3951e+00],
  [-1.3643e+01, -1.7196e+01, -1.3714e+01,  …,  1.3714e+01,
    5.2582e+00, -5.1871e+00]]],


[[[-5.9687e+00, -3.1265e+00, -3.5528e+00,  …, -1.4638e+01,
    5.7556e+00,  5.9687e+00],
  [-3.6949e+00, -1.9185e+00, -5.3292e+00,  …, -7.6030e+00,
    2.7712e+00,  6.5372e+00],
  [ 2.2738e+00, -1.6343e+00,  4.1923e+00,  …, -6.2530e+00,
   -1.0658e+00,  6.5372e+00],
  …,
  [-1.9896e+00, -1.7054e+00, -9.9479e-01,  …, -1.0161e+01,
    6.3951e-01,  6.5372e+00],
  [-2.2027e+00, -3.5528e+00, -3.6949e+00,  …, -1.0232e+01,
    8.5268e-01,  6.5372e+00],
  [-5.6845e-01, -2.7712e+00, -5.6845e-01,  …, -1.3145e+01,
   -1.1369e+00,  4.8318e+00]],

 [[-1.0090e+01, -9.4505e+00, -9.6637e+00,  …,  7.1056e+00,
   -1.2080e+00, -6.3951e+00],
  [-2.8423e-01,  7.8162e-01, -2.4159e+00,  …,  1.7125e+01,
    7.9583e+00, -1.4922e+00],
  [-5.8977e+00, -6.5372e+00, -2.4870e+00,  …,  9.0241e+00,
    6.5372e+00, -1.4922e+00],
  …,
```

```
 [-1.4922e+00,  8.5268e-01, -7.8162e-01,  …,  1.4851e+01,
   5.9687e+00, -1.4922e+00],
 [-6.3951e-01, -1.5632e+00, -1.2080e+00,  …,  1.3643e+01,
   6.5372e+00, -1.4922e+00],
 [ 2.7001e+00,  9.2373e-01,  2.3449e+00,  …,  1.0445e+01,
   1.7764e+00, -2.5580e+00]],

[[ 3.3539e+01,  2.8138e+01,  2.8209e+01,  …,  2.4088e+01,
  -1.4211e+00,  7.8162e-01],
 [-2.3164e+01, -2.1175e+01, -1.8830e+01,  …,  1.4922e+01,
  -4.3344e+00, -4.9029e+00],
 [-4.4765e+00, -4.7608e+00, -8.3846e+00,  …,  1.6059e+01,
  -4.6187e+00, -4.9029e+00],
 …,
 [ 1.5632e+00, -3.4107e+00, -3.9081e+00,  …,  2.2525e+01,
  -6.2530e+00, -4.9029e+00],
 [ 1.1369e+00, -4.9739e-01, -9.2373e-01,  …,  2.2383e+01,
  -6.9635e+00, -4.9029e+00],
 [-1.4211e-01, -8.5268e-01, -3.6949e+00,  …,  2.4585e+01,
  -4.6187e+00, -3.6239e+00]],

…,

[[-5.0876e+01, -5.1587e+01, -5.0876e+01,  …,  1.8830e+01,
   4.8318e+00,  9.3084e+00],
 [-9.7347e+00, -1.3998e+01, -1.4495e+01,  …,  1.2932e+01,
   1.9185e+00,  6.3951e+00],
 [ 1.7054e+00,  4.2634e-01, -2.8422e-01,  …,  1.3216e+01,
   1.5632e+00,  6.3951e+00],
 …,
 [-4.4765e+00, -5.3292e+00, -1.8475e+00,  …,  1.5064e+01,
   3.9081e+00,  6.3951e+00],
 [-1.8475e+00, -4.6187e+00, -5.2582e+00,  …,  1.3216e+01,
   2.9133e+00,  6.3951e+00],
 [-5.4003e+00, -3.5528e+00, -5.7556e+00,  …,  1.5775e+01,
   2.0606e+00,  8.3136e+00]],

[[ 7.1056e-01, -8.5268e-01, -4.9739e-01,  …, -3.9081e+00,
  -2.7712e+00,  2.1317e+00],
 [-2.1246e+01, -2.2454e+01, -2.2738e+01,  …, -1.1440e+01,
  -9.4505e+00, -9.2373e-01],
 [-2.4159e+00, -6.6082e+00, -8.0294e+00,  …, -6.1819e+00,
  -9.9479e+00, -9.2373e-01],
 …,
 [ 2.9133e+00, -2.6291e+00, -5.9687e+00,  …, -4.9029e+00,
  -1.1866e+01, -9.2373e-01],
 [-2.4159e+00, -3.4107e+00, -4.2634e+00,  …, -1.9185e+00,
  -1.1085e+01, -9.2373e-01],
```

```
  [-5.0450e+00, -5.6134e+00, -5.9687e+00,  …,  1.1369e+00,
   -4.9739e+00,  7.1767e+00]],

 [[ 1.0658e+00, -2.7712e+00, -3.9792e+00,  …, -1.1795e+01,
   -1.3643e+01, -1.2648e+01],
  [-3.9081e+00, -4.3344e+00, -2.9844e+00,  …, -4.9739e-01,
   -8.5268e+00, -1.1582e+01],
  [-4.9739e+00, -2.2738e+00, -1.7764e+00,  …, -3.2686e+00,
   -7.1056e+00, -1.1582e+01],
  …,
  [-5.7556e+00, -4.0502e+00, -4.8318e+00,  …,  7.1057e-02,
   -7.5320e+00, -1.1582e+01],
  [-5.1871e+00, -3.6949e+00, -2.3449e+00,  …,  1.0658e+00,
   -6.4661e+00, -1.1582e+01],
  [-2.9844e+00, -2.3449e+00, -1.7054e+00,  …, -1.5632e+00,
   -9.4505e+00, -1.2577e+01]]],


[[[-2.9133e+00, -6.5372e+00, -4.4765e+00,  …, -1.9896e+00,
   -5.8266e+00, -6.0398e+00],
  [-4.2634e-01, -9.4870e-07, -1.9185e+00,  …,  1.8475e+00,
   -9.2373e+00,  2.2027e+00],
  [-3.4818e+00, -2.9844e+00, -4.2634e-01,  …, -1.4780e+01,
   -8.5268e-01, -1.1369e+01],
  …,
  [ 6.8214e+00,  4.8318e+00,  2.9844e+00,  …,  1.7053e+00,
    5.0450e+00,  1.9043e+01],
  [ 5.8266e+00,  1.7054e+00,  1.4922e+00,  …,  2.9133e+00,
    2.4159e+00,  7.7451e+00],
  [-6.3951e-01, -1.5632e+00, -1.5632e+00,  …, -1.5632e+00,
   -1.5632e+00,  4.8318e+00]],

 [[-5.8266e+00, -2.1317e+00,  5.3292e+00,  …, -1.0019e+01,
   -7.7451e+00, -7.6741e+00],
  [-1.2790e+00,  3.4818e+00,  5.6134e+00,  …, -2.4443e+01,
   -1.5774e+01,  1.0872e+01],
  [-8.4557e+00, -3.1265e+00,  5.5424e+00,  …, -2.7215e+01,
   -2.2809e+01, -3.6949e+00],
  …,
  [-3.0554e+00, -5.4713e+00, -5.9687e+00,  …, -2.6291e+00,
   -4.6187e+00, -1.4993e+01],
  [ 4.2634e-01, -1.1369e+00, -7.1056e-01,  …,  2.8422e-01,
   -4.2634e-01, -5.3292e+00],
  [ 1.2080e+00,  2.4870e+00,  2.4870e+00,  …,  2.4870e+00,
    2.4870e+00, -2.5580e+00]],

 [[ 4.9029e+00,  5.6845e+00,  1.0658e+00,  …, -7.1056e-02,
    6.9635e+00, -7.4609e+00],
```

```
 [ 1.0658e+00, -4.9739e-01, -2.9844e+00,  …, -8.3846e+00,
   1.2009e+01, -2.8423e+00],
 [-1.9896e+00, -1.7764e+00, -7.8162e-01,  …, -1.4424e+01,
  -1.0658e+01,  2.1317e+00],
 …,
 [-3.2117e+01, -2.7286e+01, -3.0270e+01,  …, -2.0606e+01,
  -2.3022e+01, -4.1213e+01],
 [ 9.2373e-01,  6.6082e+00,  6.6082e+00,  …,  3.4107e+00,
   4.6897e+00, -1.2790e+00],
 [-4.4055e+00,  1.6343e+00,  1.6343e+00,  …,  1.6343e+00,
   1.6343e+00, -3.6239e+00]],

…,

[[-3.9792e+00, -6.1108e+00, -1.0161e+01,  …, -1.1227e+01,
  -4.8318e+00, -4.9740e-01],
 [-1.2577e+01, -5.4713e+00, -1.0658e+00,  …, -2.0251e+01,
  -1.2719e+01, -1.3501e+01],
 [-6.4661e+00, -5.1161e+00, -6.8214e+00,  …, -1.1582e+01,
  -2.3093e+01, -1.7267e+01],
 …,
 [ 2.0322e+01,  1.6698e+01,  1.5277e+01,  …,  2.2809e+01,
   1.9469e+01,  2.3733e+01],
 [ 5.1871e+00,  3.9081e+00,  4.2634e+00,  …,  6.5372e+00,
   5.2582e+00,  8.8820e+00],
 [ 1.9185e+00,  9.2373e-01,  9.2373e-01,  …,  9.2373e-01,
   9.2373e-01,  8.3136e+00]],

[[ 1.5632e+00, -3.5528e-01, -6.5372e+00,  …, -7.8162e+00,
  -9.9479e-01, -5.1161e+00],
 [ 1.2790e+00, -3.3396e+00, -6.2530e+00,  …, -8.2425e+00,
  -2.5580e+00, -6.3951e-01],
 [-4.9740e-01, -5.5424e+00, -2.5580e+00,  …,  3.3396e+00,
  -7.2477e+00, -7.6741e+00],
 …,
 [-9.9479e+00, -3.1975e+00, -5.6134e+00,  …, -1.4567e+01,
  -1.0943e+01, -8.2425e+00],
 [-6.7503e+00, -2.8423e+00, -4.0502e+00,  …, -4.4765e+00,
  -4.1923e+00,  3.0554e+00],
 [-5.1161e+00, -2.1317e-01, -2.1317e-01,  …, -2.1317e-01,
  -2.1317e-01,  7.1767e+00]],

[[ 6.3951e+00,  5.6134e+00,  5.1871e+00,  …,  2.4870e+00,
   2.2738e+00, -3.5528e+00],
 [ 2.4870e+00,  1.2080e+00, -3.5528e-01,  …,  6.1108e+00,
   4.4055e+00,  2.7712e+00],
 [ 7.1056e-01,  1.4211e+00,  9.2373e-01,  …, -3.5528e+00,
   4.9739e+00, -3.5528e-01],
```

```
  …,
   [-1.4709e+01, -1.2790e+01, -1.3359e+01,  …, -1.4495e+01,
    -1.4069e+01, -1.9043e+01],
   [-1.6201e+01, -1.4709e+01, -1.4282e+01,  …, -1.5064e+01,
    -1.4282e+01, -1.6272e+01],
   [-9.1663e+00, -8.7399e+00, -8.7399e+00,  …, -8.7399e+00,
    -8.7399e+00, -1.2577e+01]]],


…,


[[[-4.4055e+00, -4.7608e+00, -2.9844e+00,  …, -1.0801e+01,
     3.6239e+00,  5.9687e+00],
   [-1.9896e+00,  1.2080e+00, -2.1317e-01,  …, -1.3572e+01,
     1.4922e+00,  6.5372e+00],
   [-3.9792e+00, -3.4818e+00, -1.6343e+00,  …, -9.4505e+00,
     1.2080e+00,  6.5372e+00],
   …,
   [-4.4055e+00, -7.4609e+00, -5.2582e+00,  …, -1.0232e+01,
     1.4922e+00,  6.5372e+00],
   [-2.1317e-01, -6.6793e+00, -1.7054e+00,  …, -1.5277e+01,
     1.8475e+00,  6.5372e+00],
   [-4.1923e+00, -1.0090e+01, -9.8768e+00,  …, -6.8214e+00,
    -2.5580e+00,  4.8318e+00]],

  [[-3.5528e-01,  1.8475e+00,  3.1265e+00,  …,  8.7399e+00,
     5.6845e-01, -6.3951e+00],
   [-8.5268e-01, -3.5528e-01, -8.5268e-01,  …,  1.5064e+01,
     6.9635e+00, -1.4922e+00],
   [-1.2790e+00,  3.5528e-01,  3.5528e+00,  …,  1.7409e+01,
     6.5372e+00, -1.4922e+00],
   …,
   [ 1.2790e+01,  3.2686e+00, -1.4922e+00,  …,  1.7693e+01,
     6.9635e+00, -1.4922e+00],
   [ 1.3145e+01,  1.1582e+01,  1.1440e+01,  …,  1.5277e+01,
     7.0346e+00, -1.4922e+00],
   [-6.4661e+00, -7.1056e+00, -2.7712e+00,  …,  1.0943e+01,
     2.2027e+00, -2.5580e+00]],

  [[ 1.0658e+00, -1.9185e+00, -3.1975e+00,  …,  2.2809e+01,
     4.7608e+00,  7.8162e-01],
   [ 4.9029e+00, -1.1369e+00,  8.5268e-01,  …,  1.7764e+01,
    -5.1161e+00, -4.9029e+00],
   [ 6.8214e+00, -7.8162e-01,  2.3449e+00,  …,  1.9398e+01,
    -5.6845e+00, -4.9029e+00],
   …,
   [ 6.1108e+00,  3.9792e+00, -9.9479e-01,  …,  2.2525e+01,
```

```
  -5.4003e+00, -4.9029e+00],
 [ 1.3359e+01,  1.8190e+01,  1.1724e+01,  …,  2.6788e+01,
  -5.4713e+00, -4.9029e+00],
 [ 3.0554e+00,  1.5561e+01,  1.4709e+01,  …,  2.4159e+01,
  -2.1317e-01, -3.6239e+00]],

…,

[[-5.8977e+00, -6.1819e+00, -7.2477e+00,  …,  1.3714e+01,
    6.3240e+00,  9.3084e+00],
 [-3.9081e+00, -1.7764e+00,  4.9740e-01,  …,  6.8925e+00,
    2.4159e+00,  6.3951e+00],
 [-3.8370e+00, -1.2009e+01, -1.7480e+01,  …,  1.5846e+01,
    2.4159e+00,  6.3951e+00],
 …,
 [ 9.4505e+00,  9.2373e+00,  3.2686e+00,  …,  1.1582e+01,
    4.1213e+00,  6.3951e+00],
 [-3.1975e+00,  2.8423e+00,  9.9479e-01,  …,  1.1582e+01,
    1.1369e+00,  6.3951e+00],
 [-3.7233e+01, -3.6807e+01, -4.2350e+01,  …,  2.4443e+01,
    1.7054e+00,  8.3136e+00]],

[[-8.5978e+00, -1.2080e+01, -1.2648e+01,  …, -2.8422e-01,
  -3.4107e+00,  2.1317e+00],
 [-4.2634e+00, -1.0161e+01, -8.1004e+00,  …,  6.4661e+00,
  -6.8214e+00, -9.2373e-01],
 [-5.4713e+00, -1.5064e+01, -1.3643e+01,  …, -3.6239e+00,
  -8.1004e+00, -9.2373e-01],
 …,
 [ 4.3344e+00,  9.5926e+00,  7.9583e+00,  …,  7.0346e+00,
  -1.1085e+01, -9.2373e-01],
 [-4.1923e+00, -1.4211e-01,  2.0606e+00,  …,  1.0161e+01,
  -1.1369e+01, -9.2373e-01],
 [-1.0232e+01, -4.2634e+00, -4.4055e+00,  …, -1.3501e+00,
  -5.0450e+00,  7.1767e+00]],

[[-4.5476e+00, -3.4818e+00, -2.1317e+00,  …, -4.0502e+00,
  -1.0090e+01, -1.2648e+01],
 [-7.8162e-01, -4.2634e-01, -7.1056e-01,  …, -1.8475e+00,
  -4.6187e+00, -1.1582e+01],
 [ 4.2634e-01,  1.2790e+00,  3.8370e+00,  …, -1.7764e+00,
  -9.4505e+00, -1.1582e+01],
 …,
 [-9.9479e+00, -1.1724e+01, -1.1440e+01,  …,  7.8162e-01,
  -6.5372e+00, -1.1582e+01],
 [-1.1369e+00,  3.5528e-01, -1.4211e-01,  …,  2.2738e+00,
  -5.1161e+00, -1.1582e+01],
 [ 3.3396e+00,  2.7712e+00,  3.6949e+00,  …, -2.0606e+00,
```

```
   -1.3856e+01, -1.2577e+01]]],


[[[ 2.7712e+00,  3.1975e+00,  2.5580e+00,  …, -1.3288e+01,
    5.6134e+00,  5.9687e+00],
  [ 2.7712e+00,  4.7608e+00, -4.2634e-01,  …, -1.0729e+01,
    6.3951e-01,  6.5372e+00],
  [-3.4107e+00,  8.8820e+00, -3.5528e-01,  …, -1.0872e+01,
    7.1056e-02,  6.5372e+00],
  …,
  [-8.5268e+00, -7.1767e+00, -7.6030e+00,  …, -5.4713e+00,
   -2.4159e+00,  6.5372e+00],
  [ 1.7764e+00, -7.1056e-01, -9.2373e-01,  …,  1.4211e-01,
   -1.5632e+00,  6.5372e+00],
  [ 6.3240e+00,  2.3449e+00,  2.3449e+00,  …,  1.2790e+00,
   -4.8318e+00,  4.8318e+00]],

 [[-3.1265e+00, -2.7712e+00, -1.2080e+00,  …,  1.5064e+01,
    2.0606e+00, -6.3951e+00],
  [ 1.1369e+00, -1.8475e+00, -3.9081e+00,  …,  1.5988e+01,
    7.0346e+00, -1.4922e+00],
  [-2.7001e+00,  1.4211e+00, -3.2686e+00,  …,  1.5774e+01,
    7.1056e+00, -1.4922e+00],
  …,
  [-1.2080e+00, -5.6845e-01, -1.2080e+00,  …,  1.7267e+01,
    6.8925e+00, -1.4922e+00],
  [-6.2530e+00, -5.3292e+00, -5.4713e+00,  …,  4.2634e+00,
    6.6082e+00, -1.4922e+00],
  [-6.0398e+00, -7.0346e+00, -7.0346e+00,  …, -4.1213e+00,
   -8.5268e-01, -2.5580e+00]],

 [[-7.4609e+00, -8.1004e+00, -6.8214e+00,  …,  2.8565e+01,
    1.4211e-01,  7.8162e-01],
  [-5.9687e+00, -7.1056e+00, -1.2790e+00,  …,  2.3804e+01,
   -7.9583e+00, -4.9029e+00],
  [-2.9133e+00, -9.6637e+00, -2.9133e+00,  …,  2.3804e+01,
   -7.4609e+00, -4.9029e+00],
  …,
  [ 1.3501e+01,  1.8190e+01,  1.7977e+01,  …,  2.1530e+01,
   -3.6239e+00, -4.9029e+00],
  [-3.1762e+01, -2.5722e+01, -2.6078e+01,  …,  5.8266e+00,
   -9.9479e-01, -4.9029e+00],
  [ 5.4713e+00,  9.0952e+00,  9.0952e+00,  …,  5.1871e+00,
   -3.5528e-01, -3.6239e+00]],

 …,

 [[ 9.2373e-01, -3.5528e+00, -4.8318e+00,  …,  1.8261e+01,
```

```
    5.0450e+00,  9.3084e+00],
   [-1.4922e+00, -1.8475e+00, -5.7556e+00,  …,  1.4993e+01,
    2.8423e+00,  6.3951e+00],
   [-6.3951e-01, -7.1767e+00,  8.1004e+00,  …,  1.5206e+01,
    2.5580e+00,  6.3951e+00],
   …,
   [-5.3790e+01, -5.3008e+01, -5.2297e+01,  …,  2.5722e+01,
    1.2080e+00,  6.3951e+00],
   [ 2.4728e+01,  2.3306e+01,  2.3378e+01,  …,  1.4709e+01,
    4.9740e-01,  6.3951e+00],
   [ 5.8266e+00,  5.1871e+00,  5.1871e+00,  …,  6.7503e+00,
    7.1056e-01,  8.3136e+00]],

  [[-1.0587e+01, -1.1653e+01, -1.1014e+01,  …, -4.9029e+00,
    -6.7503e+00,  2.1317e+00],
   [-6.8214e+00, -9.3794e+00, -9.6637e+00,  …, -3.4107e+00,
    -1.0587e+01, -9.2373e-01],
   [-5.2582e+00, -1.0019e+01, -4.2634e+00,  …, -3.6949e+00,
    -1.0303e+01, -9.2373e-01],
   …,
   [-1.4140e+01, -1.3714e+01, -1.3998e+01,  …, -1.7267e+01,
    -1.2222e+01, -9.2373e-01],
   [-1.6840e+01, -1.3998e+01, -1.4495e+01,  …, -1.0374e+01,
    -1.1014e+01, -9.2373e-01],
   [ 1.4211e-01,  3.8370e+00,  3.8370e+00,  …,  1.9896e+00,
    -2.8423e+00,  7.1767e+00]],

  [[-2.7001e+00, -2.1317e+00, -1.4922e+00,  …,  1.5632e+00,
    -1.1582e+01, -1.2648e+01],
   [-3.1975e+00, -1.9185e+00, -1.8475e+00,  …, -1.0842e-06,
    -7.8162e+00, -1.1582e+01],
   [-6.5372e+00, -2.9133e+00, -6.0398e+00,  …,  7.1056e-01,
    -7.6030e+00, -1.1582e+01],
   …,
   [ 1.6556e+01,  1.0801e+01,  1.0658e+01,  …,  9.2373e-01,
    -1.4851e+01, -1.1582e+01],
   [-7.6030e+00, -5.4003e+00, -5.6134e+00,  …, -8.6689e+00,
    -6.8214e+00, -1.1582e+01],
   [-1.7622e+01, -1.7054e+01, -1.7054e+01,  …, -1.2577e+01,
    -8.3136e+00, -1.2577e+01]]],


 [[[ 5.1161e+00,  4.6187e+00, -6.1819e+00,  …, -9.2373e-01,
    -3.8370e+00, -1.7196e+01],
   [ 1.2790e+00,  1.3501e+00, -5.6134e+00,  …, -5.5424e+00,
    -5.8266e+00, -9.3794e+00],
   [ 1.2790e+00,  1.7054e+00, -4.8318e+00,  …, -2.6291e+00,
    -2.4159e+00, -1.3856e+01],
```

```
 …,
 [ 1.2790e+00,  7.3188e+00, -2.2667e+01,  …, -9.3084e+00,
   7.5320e+00, -3.0625e+01],
 [ 1.2790e+00,  6.7503e+00, -2.1388e+01,  …,  1.6627e+01,
  -1.3927e+01, -1.0872e+01],
 [-6.3951e-01,  4.1923e+00, -2.0464e+01,  …,  6.6082e+00,
  -2.0109e+01, -8.3136e+00]],

[[ 1.1369e+00, -3.9081e+00, -1.0303e+01,  …, -7.5320e+00,
  -9.1663e+00,  1.1937e+01],
 [ 6.2530e+00,  3.7660e+00, -4.4765e+00,  …,  5.8266e+00,
   3.3396e+00,  7.5320e+00],
 [ 6.2530e+00,  2.3449e+00, -2.9133e+00,  …,  3.5528e-01,
  -2.8423e+00,  7.8162e-01],
 …,
 [ 6.2530e+00, -2.7001e+00, -3.2544e+01,  …,  5.8977e+00,
   1.1085e+01, -2.0677e+01],
 [ 6.2530e+00, -2.7001e+00, -3.1123e+01,  …,  4.2136e+01,
  -4.2634e+00, -3.7162e+01],
 [ 1.2080e+00, -3.9792e+00, -3.3396e+01,  …,  1.7977e+01,
  -2.6007e+01, -4.6187e+00]],

[[ 1.3501e+00,  5.1871e+00,  1.4353e+01,  …,  1.5846e+01,
   1.8119e+01,  2.5225e+01],
 [-6.3951e+00, -1.1369e+00,  4.2634e+00,  …,  1.2080e+00,
   2.9133e+00,  2.1317e-01],
 [-6.3951e+00, -7.8162e-01,  6.1108e+00,  …,  1.6343e+00,
   1.8974e-06, -3.7660e+00],
 …,
 [-6.3951e+00, -2.2027e+00,  1.8759e+01,  …, -1.2435e+01,
  -2.1956e+01,  1.3359e+01],
 [-6.3951e+00, -2.8423e+00,  1.8261e+01,  …, -2.2241e+01,
   5.4713e+00,  9.2373e-01],
 [-4.4055e+00, -1.1369e+00,  1.8333e+01,  …, -4.9739e+00,
   2.8351e+01,  5.4713e+00]],

…,

[[ 5.2582e+00,  5.6845e+00,  3.3396e+00,  …, -1.5206e+01,
  -1.4424e+01, -1.8261e+01],
 [ 2.7712e+00,  1.9185e+00,  1.9185e+00,  …, -1.2151e+01,
  -1.5348e+01, -1.5846e+01],
 [ 2.7712e+00,  1.1369e+00,  5.6845e+00,  …, -3.3752e+01,
  -3.0128e+01, -2.2951e+01],
 …,
 [ 2.7712e+00,  7.3899e+00,  1.2080e+01,  …, -9.3794e+00,
  -2.1459e+01,  9.8058e+00],
 [ 2.7712e+00,  7.2477e+00,  1.1014e+01,  …, -4.9739e+00,
```

```
          2.7072e+01,  2.4372e+01],
         [ 1.9185e+00,  7.7451e+00,  5.5424e+00,  …,  1.1724e+01,
          3.2757e+01, -1.4353e+01]],

        [[-6.8925e+00, -3.7660e+00,  1.8475e+00,  …,  6.1108e+00,
          6.7503e+00,  1.1298e+01],
         [-1.0943e+01, -5.4003e+00,  1.4211e+00,  …, -1.7125e+01,
          -1.5632e+01, -2.0891e+01],
         [-1.0943e+01, -7.5320e+00,  2.7001e+00,  …, -1.9967e+01,
          -1.9256e+01, -2.1885e+01],
         …,
         [-1.0943e+01,  2.8423e-01,  1.3074e+01,  …, -8.8110e+00,
          -1.1298e+01, -5.8977e+00],
         [-1.0943e+01,  3.5528e-01,  1.4353e+01,  …, -1.9967e+01,
          -5.1161e+00,  9.3084e+00],
         [-5.1161e+00,  8.0294e+00,  1.6059e+01,  …, -1.7764e+01,
          7.8873e+00,  1.1653e+01]],

        [[-1.0801e+01, -9.8768e+00, -1.0587e+01,  …, -1.2506e+01,
          -1.2435e+01, -8.3846e+00],
         [-7.0346e+00, -6.9635e+00, -7.4609e+00,  …, -2.4870e+00,
          -1.4922e+00, -7.6030e+00],
         [-7.0346e+00, -7.0346e+00, -9.4505e+00,  …,  4.6187e+00,
          4.1213e+00, -1.0658e+00],
         …,
         [-7.0346e+00, -1.2222e+01, -5.4713e+00,  …,  8.5268e-01,
          -9.5215e+00, -4.6187e+00],
         [-7.0346e+00, -1.2080e+01, -5.6134e+00,  …,  7.1056e-02,
          -7.0346e+00, -6.1108e+00],
         [-9.1663e+00, -1.2577e+01, -1.9185e+00,  …, -5.8977e+00,
          2.1317e-01,  6.3951e-01]]]], device='cuda:0',
       grad_fn=<MulBackward0>)
```

[8]:
```
#### input floating number / weight quantized version

conv_ref = torch.nn.Conv2d(in_channels = 16, out_channels=16, kernel_size = 3,␣
 ↪bias = False)
conv_ref.weight = model.layer1[0].conv1.weight_q

output_ref = conv_ref(x)
print(output_ref)
```

```
tensor([[[[ 1.6284e+00, -1.1100e+00, -1.1100e+00,  …, -1.1100e+00,
          -1.1100e+00,  6.3102e+00],
         [-1.3505e-01, -8.9356e-01, -1.1729e+00,  …, -1.3454e-02,
          -3.0839e+00,  9.5083e+00],
         [-2.5020e+00,  8.7685e-01,  1.2547e+00,  …, -1.7519e+00,
          2.8567e+00,  3.3267e+00],
```

```
 …,
 [-4.6528e+00,  1.2273e+00, -7.0981e+00,  …,  8.0367e-01,
  -2.2426e+01,  4.5546e+01],
 [-7.8390e+00, -4.7935e+00,  3.8080e+00,  …,  2.3688e+00,
  -2.1897e+01,  4.2395e+01],
 [-4.4119e+00, -1.0232e+01, -2.7468e+00,  …,  1.1647e+00,
  -1.7875e+01,  3.2359e+01]],

[[-9.1647e-01, -3.2873e+00, -3.2873e+00,  …, -3.2873e+00,
  -3.2873e+00, -1.3599e+01],
 [ 4.6317e-01, -8.5273e-02, -7.3866e-01,  …,  7.5950e-01,
   4.1638e+00, -3.7104e+00],
 [ 6.4919e+00,  7.8488e+00,  8.5590e+00,  …,  8.4390e+00,
   1.3104e+01, -1.2397e+01],
 …,
 [ 1.7425e+00,  9.2295e+00, -2.8483e+00,  …, -1.4974e+01,
   1.3999e+01,  6.5554e+01],
 [-5.0529e+00,  1.0100e+01,  6.5383e+00,  …, -1.1523e+01,
   7.1825e+00,  5.5441e+01],
 [-5.6220e+00, -3.6240e+00,  3.1578e-01,  …, -9.7116e+00,
   1.5825e+00,  4.3496e+01]],

[[ 7.9561e+00,  1.2931e+01,  1.2931e+01,  …,  1.2931e+01,
   1.2931e+01,  1.2513e+00],
 [-2.7846e+00,  2.9675e+00,  2.7207e+00,  …,  3.3133e+00,
   4.6245e+00, -8.6171e+00],
 [-1.3024e+01, -1.2413e+01, -1.2687e+01,  …, -1.4714e+01,
  -1.2613e+01, -2.4274e+01],
 …,
 [ 8.2165e+00,  4.3793e+00,  5.1420e+00,  …, -5.3902e+00,
  -5.9339e+00, -5.2328e+01],
 [ 6.7252e+00,  8.7164e+00,  5.4719e+00,  …, -6.5759e+00,
  -6.0453e+00, -5.0588e+01],
 [-2.9569e+00,  5.1055e+00,  6.1494e+00,  …, -3.4946e+00,
  -4.2454e+00, -3.5398e+01]],

…,

[[ 6.7213e+00,  5.1429e+00,  5.1429e+00,  …,  5.1429e+00,
   5.1429e+00,  1.0260e+01],
 [-1.0111e+00, -6.7216e-01, -1.1551e+00,  …,  4.6131e-01,
   5.1416e+00,  5.7422e+00],
 [ 3.2692e+01,  3.1835e+01,  3.1485e+01,  …,  2.9633e+01,
   3.1927e+01,  4.4109e+01],
 …,
 [-3.2682e+00,  2.7962e-02,  4.5027e+00,  …, -1.5944e+01,
  -1.2783e+01, -1.7238e+01],
 [-6.2854e+00, -9.9723e+00,  1.0510e+00,  …, -1.3873e+01,
```

```
  -7.3300e+00, -1.7463e+01],
 [-9.9664e+00, -1.0516e+01, -1.2495e+01,  …, -8.5483e+00,
  -7.7511e+00, -1.0772e+01]],

[[-9.0370e+00, -6.1038e+00, -6.1038e+00,  …, -6.1038e+00,
  -6.1038e+00,  1.7905e+00],
 [-3.3743e+00,  5.3262e-01,  7.4468e-01,  …, -2.3161e+00,
   4.0393e+00,  4.8784e+00],
 [ 5.7699e+00,  7.4000e+00,  6.6774e+00,  …,  7.0583e+00,
   2.0419e+01,  5.1559e+00],
 …,
 [ 5.3553e+00,  1.0221e+01,  1.3084e+01,  …, -4.7231e+00,
   1.4842e-01, -5.8969e+01],
 [ 2.2786e+00,  6.1169e+00,  1.1625e+01,  …, -2.9117e+00,
   1.0576e+00, -5.5377e+01],
 [-7.5908e-01,  3.7970e+00,  1.1074e+01,  …, -2.3084e+00,
  -2.6501e+00, -4.5044e+01]],

[[-5.7716e+00, -4.8791e+00, -4.8791e+00,  …, -4.8791e+00,
  -4.8791e+00, -9.4584e+00],
 [-8.1336e+00, -8.8156e+00, -9.3756e+00,  …, -7.2622e+00,
  -8.9963e+00, -1.0606e+01],
 [-2.1974e+01, -1.8238e+01, -1.8203e+01,  …, -1.7126e+01,
  -1.7662e+01, -1.2321e+01],
 …,
 [-1.4368e+01, -1.3423e+01, -1.4605e+01,  …,  1.4129e+01,
   1.3909e+00, -2.6373e+00],
 [-1.4330e+01, -1.4185e+01, -1.2037e+01,  …,  1.3649e+01,
   9.7855e-01, -4.7089e+00],
 [-1.5158e+01, -1.7222e+01, -1.3611e+01,  …,  1.3860e+01,
   4.1622e+00, -6.0993e+00]]],


[[[-1.1393e+01, -2.6498e+00, -4.8228e+00,  …, -3.4025e+01,
   2.1635e+00,  6.3102e+00],
 [-1.6754e+00, -1.2795e-01, -3.5653e+00,  …, -2.2661e+01,
   3.0573e+00,  1.0141e+01],
 [ 2.3018e-02, -2.6788e+00,  1.6510e+00,  …, -1.5729e+01,
  -5.3459e-01,  1.0141e+01],
 …,
 [-2.3898e+00, -1.8305e+00, -1.6263e+00,  …, -3.0752e+01,
   6.9302e-01,  1.0141e+01],
 [-2.2044e+00, -3.7950e+00, -3.7754e+00,  …, -3.0395e+01,
   8.6241e-01,  1.0141e+01],
 [-9.6453e-01, -2.1798e+00, -8.2954e-01,  …, -3.4304e+01,
  -1.8578e-01,  7.6341e+00]],

[[-1.4510e+01, -1.5937e+01, -1.6385e+01,  …,  1.4620e+01,
```

```
  -4.2085e+00, -1.3599e+01],
 [ 7.9750e-01,  5.4557e-02, -1.9165e+00,  …,  2.3376e+01,
   7.9617e+00, -3.8332e+00],
 [-8.2085e+00, -8.8289e+00, -5.6405e+00,  …,  1.3797e+01,
   6.2806e+00, -3.8332e+00],
 …,
 [-1.5041e+00,  3.1454e-01, -8.5982e-01,  …,  2.5542e+01,
   6.9361e+00, -3.8332e+00],
 [ 2.0644e-01, -1.2578e+00, -1.3820e+00,  …,  2.5202e+01,
   6.7621e+00, -3.8332e+00],
 [ 2.4106e+00,  1.6561e+00,  1.8477e+00,  …,  1.9977e+01,
   1.6163e+00, -4.7189e+00]],

[[ 4.1174e+01,  3.5361e+01,  3.6748e+01,  …,  3.9592e+01,
   5.7698e+00,  1.2513e+00],
 [-3.0747e+01, -2.9178e+01, -2.6979e+01,  …,  2.7610e+01,
  -4.6895e+00, -8.2618e+00],
 [ 4.8143e-01, -8.1720e-01, -5.2741e+00,  …,  2.5656e+01,
  -4.8291e+00, -8.2618e+00],
 …,
 [ 1.7041e+00, -3.7144e+00, -4.5990e+00,  …,  3.9674e+01,
  -6.6918e+00, -8.2618e+00],
 [ 4.5526e-01, -3.8019e-01,  1.2597e-01,  …,  3.8593e+01,
  -7.0457e+00, -8.2618e+00],
 [-1.1270e+00, -9.3761e-01, -2.7907e+00,  …,  4.4186e+01,
  -4.9986e+00, -5.6262e+00]],

…,

[[-6.5664e+01, -6.6544e+01, -6.5177e+01,  …,  2.8634e+01,
   5.7841e+00,  1.0260e+01],
 [-9.6142e+00, -1.2900e+01, -1.4903e+01,  …,  2.9308e+01,
   2.0948e+00,  7.4420e+00],
 [ 3.0076e+00,  9.8079e-01, -4.2698e-01,  …,  2.6526e+01,
   2.3063e+00,  7.4420e+00],
 …,
 [-3.5072e+00, -5.3357e+00, -1.6332e+00,  …,  3.6054e+01,
   3.0641e+00,  7.4420e+00],
 [-2.4401e+00, -4.1997e+00, -5.0975e+00,  …,  3.4404e+01,
   3.2004e+00,  7.4420e+00],
 [-4.1955e+00, -3.7400e+00, -5.0504e+00,  …,  3.3018e+01,
  -3.5838e-01,  9.9246e+00]],

[[-7.1923e+00, -6.3909e+00, -6.3707e+00,  …, -1.5463e+01,
  -5.2120e+00,  1.7905e+00],
 [-2.8166e+01, -2.9152e+01, -3.0142e+01,  …, -1.9232e+01,
  -9.8226e+00,  1.6301e+00],
 [-4.7409e+00, -8.4841e+00, -1.0677e+01,  …, -1.1964e+01,
```

```
   -1.0317e+01,  1.6301e+00],
  …,
  [ 1.8435e+00, -3.2040e+00, -6.1160e+00,  …, -1.4262e+01,
   -1.1434e+01,  1.6301e+00],
  [-2.7296e+00, -3.4881e+00, -4.5771e+00,  …, -1.1656e+01,
   -1.0864e+01,  1.6301e+00],
  [-4.6877e+00, -5.4251e+00, -6.7706e+00,  …, -4.0157e+00,
   -4.9892e+00,  8.7797e+00]],

 [[ 7.4128e+00,  1.8478e+00,  1.2035e+00,  …, -7.6712e+00,
   -8.1715e+00, -9.4584e+00],
  [-6.1898e+00, -7.2696e+00, -6.3170e+00,  …, -3.2255e+00,
   -8.5628e+00, -1.3091e+01],
  [-1.2281e+00,  1.6836e+00,  2.0852e+00,  …, -6.2344e+00,
   -7.2068e+00, -1.3091e+01],
  …,
  [-4.8721e+00, -3.8864e+00, -4.8364e+00,  …, -5.5096e+00,
   -7.6029e+00, -1.3091e+01],
  [-4.8535e+00, -3.7186e+00, -2.5698e+00,  …, -4.4440e+00,
   -7.2164e+00, -1.3091e+01],
  [-3.7043e+00, -2.2814e+00, -1.5915e+00,  …, -5.0115e+00,
   -8.7863e+00, -1.4262e+01]]],


[[[-2.6367e+00, -7.4480e+00, -4.0937e+00,  …, -1.4156e+00,
   -5.2801e+00, -7.6403e+00],
  [-1.1038e+00,  5.7906e-02, -2.6549e+00,  …,  8.1835e-01,
   -9.4071e+00,  2.3492e+00],
  [-3.0957e+00, -3.3148e+00, -1.2777e+00,  …, -1.3692e+01,
   -1.3574e+00, -1.1280e+01],
  …,
  [ 1.5081e+01,  1.6147e+01,  1.5082e+01,  …,  7.7650e+00,
    1.1190e+01,  2.3715e+01],
  [ 2.6208e+00, -1.2502e+00, -2.0525e+00,  …,  4.1947e-01,
   -1.6925e-01,  7.3601e+00],
  [-4.3907e-01, -8.7089e-01, -8.7089e-01,  …, -8.7089e-01,
   -8.7089e-01,  7.6341e+00]],

 [[-6.3395e+00, -2.5407e+00,  6.3170e+00,  …, -8.4603e+00,
   -7.8116e+00, -5.9909e+00],
  [-2.0339e+00,  2.9920e+00,  4.6376e+00,  …, -2.5307e+01,
   -1.5977e+01,  1.0451e+01],
  [-7.2247e+00, -2.4361e+00,  5.0436e+00,  …, -2.6665e+01,
   -2.3452e+01, -3.9882e+00],
  …,
  [-4.4844e+00, -7.7519e+00, -9.9349e+00,  …, -5.5763e+00,
   -6.9774e+00, -1.9426e+01],
  [-6.6584e-03, -2.9150e+00, -3.9098e+00,  …, -5.4628e+00,
```

```
  -3.6057e+00, -1.3089e+01],
 [ 1.5681e+00,  1.4091e+00,  1.4091e+00,  …,  1.4091e+00,
   1.4091e+00, -4.7189e+00]],

[[ 4.2859e+00,  5.3226e+00,  7.2981e-01,  …, -4.4561e-01,
   6.7060e+00, -7.2053e+00],
 [ 2.2681e+00, -6.8178e-01, -2.8947e+00,  …, -8.2959e+00,
   1.1902e+01, -3.1431e+00],
 [-1.3163e+00, -1.3232e+00, -3.9305e-01,  …, -1.4934e+01,
  -1.1268e+01,  1.7197e+00],
 …,
 [-5.0858e+01, -5.2831e+01, -6.2505e+01,  …, -4.1688e+01,
  -4.0931e+01, -5.7672e+01],
 [ 8.9602e+00,  1.6373e+01,  1.8093e+01,  …,  1.1942e+01,
   1.1741e+01, -3.5249e-01],
 [-4.4115e+00,  1.4801e+00,  1.4801e+00,  …,  1.4801e+00,
   1.4801e+00, -5.6262e+00]],

…,

[[-3.6607e+00, -5.6920e+00, -1.1363e+01,  …, -1.2061e+01,
  -4.4299e+00, -2.5358e+00],
 [-1.2316e+01, -5.5427e+00, -1.0680e+00,  …, -1.9211e+01,
  -1.3965e+01, -1.4112e+01],
 [-5.9356e+00, -4.4577e+00, -6.2214e+00,  …, -1.4195e+01,
  -2.2140e+01, -1.8012e+01],
 …,
 [ 2.3230e+01,  2.1078e+01,  2.1815e+01,  …,  2.6492e+01,
   2.2912e+01,  2.7165e+01],
 [ 6.5353e+00,  5.5058e+00,  5.8404e+00,  …,  6.8553e+00,
   5.8128e+00,  1.0745e+01],
 [ 1.3763e-01,  8.3348e-01,  8.3348e-01,  …,  8.3348e-01,
   8.3348e-01,  9.9246e+00]],

[[ 6.5181e-01, -9.8430e-01, -6.9013e+00,  …, -8.8357e+00,
  -1.3418e+00, -4.1702e+00],
 [ 8.0180e-01, -3.5405e+00, -6.6094e+00,  …, -9.2666e+00,
  -2.8659e+00, -4.7316e-01],
 [-8.1489e-01, -5.5243e+00, -1.8058e+00,  …,  4.3714e+00,
  -7.4869e+00, -6.9309e+00],
 …,
 [-2.3003e+01, -2.0661e+01, -2.6330e+01,  …, -3.1702e+01,
  -2.5874e+01, -1.9683e+01],
 [-8.1036e+00, -5.4480e+00, -5.8035e+00,  …, -7.5189e+00,
  -7.0286e+00,  1.1120e+00],
 [-4.3723e+00, -6.1675e-01, -6.1675e-01,  …, -6.1675e-01,
  -6.1675e-01,  8.7797e+00]],
```

```
 [[ 6.3274e+00,  5.4326e+00,  5.0168e+00,  …,  2.4142e+00,
    1.7799e+00, -3.7516e+00],
  [ 3.1090e+00,  1.9954e+00, -1.4417e-01,  …,  6.4243e+00,
    4.4599e+00,  3.5370e+00],
  [ 1.0623e+00,  1.3050e+00,  1.2585e+00,  …, -2.3012e+00,
    4.7328e+00, -7.0811e-01],
  …,
  [-1.8027e+01, -1.7145e+01, -1.8238e+01,  …, -2.0048e+01,
   -1.8943e+01, -2.3297e+01],
  [-1.0281e+01, -8.6026e+00, -7.4590e+00,  …, -1.0168e+01,
   -9.0088e+00, -1.2633e+01],
  [-8.7891e+00, -9.1112e+00, -9.1112e+00,  …, -9.1112e+00,
   -9.1112e+00, -1.4262e+01]]],


…,


[[[-5.2204e+00, -4.9097e+00, -3.4976e+00,  …, -2.3231e+01,
    7.4297e-02,  6.3102e+00],
  [-1.9104e+00,  1.1721e+00, -1.7025e-01,  …, -2.9856e+01,
    1.8347e+00,  1.0141e+01],
  [-4.2916e+00, -1.9157e+00, -1.8370e+00,  …, -2.9418e+01,
    1.8397e+00,  1.0141e+01],
  …,
  [-2.9411e+00, -8.6309e+00, -4.8126e+00,  …, -3.2190e+01,
    1.7297e+00,  1.0141e+01],
  [-5.2113e-01, -6.5579e+00, -2.3934e+00,  …, -3.5756e+01,
    1.8544e+00,  1.0141e+01],
  [-1.8949e+00, -9.5529e+00, -7.7680e+00,  …, -1.8839e+01,
   -2.8251e+00,  7.6341e+00]],

 [[ 6.7494e-01,  1.8429e+00,  3.5464e+00,  …,  1.6984e+01,
   -1.7677e+00, -1.3599e+01],
  [-3.2693e-01, -1.1587e+00, -2.1409e+00,  …,  2.7302e+01,
    6.6629e+00, -3.8332e+00],
  [-1.9645e+00,  5.5673e-01,  1.7773e+00,  …,  2.6432e+01,
    6.5977e+00, -3.8332e+00],
  …,
  [ 1.3825e+01,  2.7881e+00, -2.3437e+00,  …,  2.8342e+01,
    7.0278e+00, -3.8332e+00],
  [ 1.4337e+01,  1.1661e+01,  1.0504e+01,  …,  2.5931e+01,
    7.5407e+00, -3.8332e+00],
  [-5.8910e+00, -6.9678e+00, -2.2807e+00,  …,  1.2160e+01,
    1.2402e+00, -4.7189e+00]],

 [[ 4.2142e-01, -2.3480e+00, -3.1630e+00,  …,  3.3372e+01,
    1.1159e+01,  1.2513e+00],
```

```
 [ 4.7668e+00, -5.8344e-01,  1.2012e+00,  …,  2.7969e+01,
  -5.6494e+00, -8.2618e+00],
 [ 7.0043e+00, -7.8848e-01,  3.5760e+00,  …,  3.5780e+01,
  -5.9788e+00, -8.2618e+00],
 …,
 [ 4.1617e+00,  5.0683e+00, -6.1209e-01,  …,  4.1000e+01,
  -6.0130e+00, -8.2618e+00],
 [ 1.4596e+01,  1.8536e+01,  1.1373e+01,  …,  4.5743e+01,
  -6.5150e+00, -8.2618e+00],
 [ 2.4758e-01,  1.4733e+01,  1.3106e+01,  …,  3.8708e+01,
  -8.1130e-01, -5.6262e+00]],

…,

[[-5.5273e+00, -6.1308e+00, -6.3296e+00,  …,  2.0488e+01,
   7.1478e+00,  1.0260e+01],
 [-5.1267e+00, -1.4784e+00,  7.3792e-02,  …,  1.8086e+01,
   2.8925e+00,  7.4420e+00],
 [-3.0714e+00, -1.1994e+01, -1.7045e+01,  …,  3.5205e+01,
   2.4847e+00,  7.4420e+00],
 …,
 [ 9.5002e+00,  1.0295e+01,  4.1060e+00,  …,  3.4140e+01,
   3.7142e+00,  7.4420e+00],
 [-4.1372e+00,  3.7695e+00,  1.8936e+00,  …,  3.1530e+01,
   3.2336e-01,  7.4420e+00],
 [-5.8170e+01, -5.4841e+01, -5.9972e+01,  …,  4.3765e+01,
   2.0395e+00,  9.9246e+00]],

[[-8.9191e+00, -1.1708e+01, -1.2438e+01,  …, -5.7428e+00,
  -5.5575e+00,  1.7905e+00],
 [-4.6131e+00, -9.9999e+00, -8.5671e+00,  …, -2.3937e-01,
  -6.5546e+00,  1.6301e+00],
 [-5.6811e+00, -1.5478e+01, -1.3254e+01,  …, -1.3342e+01,
  -8.0511e+00,  1.6301e+00],
 …,
 [ 3.7933e+00,  8.8516e+00,  8.1340e+00,  …, -2.4331e+00,
  -1.1729e+01,  1.6301e+00],
 [-2.0291e+00,  6.9682e-01,  1.5833e+00,  …,  2.4327e+00,
  -1.1658e+01,  1.6301e+00],
 [-1.9511e+01, -1.4429e+01, -1.4379e+01,  …, -8.2750e+00,
  -5.3992e+00,  8.7797e+00]],

[[-4.8871e+00, -3.3878e+00, -2.1254e+00,  …,  2.7106e-01,
  -4.8421e+00, -9.4584e+00],
 [-1.6351e+00, -5.4866e-01, -5.0761e-01,  …, -4.0915e+00,
  -5.0855e+00, -1.3091e+01],
 [-2.6448e-02,  1.9286e+00,  3.4728e+00,  …, -5.3377e+00,
  -9.7728e+00, -1.3091e+01],
```

```
  …,
   [-9.0959e+00, -1.1640e+01, -1.1392e+01,  …, -3.7774e+00,
    -7.1051e+00, -1.3091e+01],
   [-1.2526e+00,  4.0602e-01, -4.4962e-01,  …, -2.3988e+00,
    -5.7459e+00, -1.3091e+01],
   [ 4.1283e+00,  1.4498e+00,  2.5739e+00,  …, -7.2588e+00,
    -1.4204e+01, -1.4262e+01]]],


[[[ 2.7199e+00,  3.2330e+00,  3.5324e+00,  …, -5.0114e+01,
     2.6050e+00,  6.3102e+00],
   [ 2.0545e+00,  7.0031e+00, -1.1207e+00,  …, -4.2764e+01,
     6.3462e-01,  1.0141e+01],
   [-4.1616e+00,  9.1621e+00,  4.4113e-01,  …, -4.2125e+01,
     6.6807e-01,  1.0141e+01],
   …,
   [-5.7023e+00, -3.3926e+00, -3.7483e+00,  …, -2.7180e+01,
    -3.6466e+00,  1.0141e+01],
   [ 1.4965e+01,  1.3880e+01,  1.3910e+01,  …, -1.7479e+00,
    -2.3939e+00,  1.0141e+01],
   [ 5.4082e-02, -4.1318e+00, -4.1154e+00,  …,  2.9960e+00,
    -4.4061e+00,  7.6341e+00]],

  [[-4.9226e+00, -2.9382e+00, -6.2386e-01,  …,  3.5524e+01,
     4.1498e-01, -1.3599e+01],
   [-2.3115e-01, -2.0721e+00, -4.7992e+00,  …,  3.1932e+01,
     7.1611e+00, -3.8332e+00],
   [-3.9396e+00,  1.0301e+00, -2.7264e+00,  …,  3.2321e+01,
     7.1970e+00, -3.8332e+00],
   …,
   [-3.4154e+00, -1.6138e+00, -2.1115e+00,  …,  2.4328e+01,
     6.0940e+00, -3.8332e+00],
   [-1.1437e+01, -1.1115e+01, -1.1461e+01,  …, -2.7581e-01,
     7.0220e+00, -3.8332e+00],
   [-1.1881e+01, -1.5355e+01, -1.5380e+01,  …, -1.1119e+01,
    -1.6514e+00, -4.7189e+00]],

  [[-8.2919e+00, -8.2873e+00, -6.8011e+00,  …,  5.8889e+01,
     5.8804e+00,  1.2513e+00],
   [-5.3687e+00, -7.1717e+00, -1.2864e+00,  …,  5.0252e+01,
    -8.4234e+00, -8.2618e+00],
   [-1.6790e+00, -9.1298e+00, -3.9351e+00,  …,  4.9857e+01,
    -8.1983e+00, -8.2618e+00],
   …,
   [ 9.6312e+00,  1.3897e+01,  1.3358e+01,  …,  5.3331e+01,
    -5.3763e+00, -8.2618e+00],
   [-7.2709e+01, -6.8861e+01, -6.9333e+01,  …,  4.4045e+00,
    -7.5286e-01, -8.2618e+00],
```

```
 [ 2.1079e+01,  2.5083e+01,  2.5134e+01,  …,  9.6481e+00,
  -6.3919e-01, -5.6262e+00]],

…,

[[ 3.4758e-01, -4.3957e+00, -5.4890e+00,  …,  4.6182e+01,
   6.0725e+00,  1.0260e+01],
 [-1.7278e+00, -1.4571e+00, -3.6956e+00,  …,  4.6777e+01,
   2.5419e+00,  7.4420e+00],
 [-1.1414e-01, -7.5741e+00,  8.5194e+00,  …,  4.5979e+01,
   2.6151e+00,  7.4420e+00],
 …,
 [-9.4574e+01, -9.5640e+01, -9.5747e+01,  …,  6.2028e+01,
   1.1473e+00,  7.4420e+00],
 [ 3.3064e+01,  3.1240e+01,  3.1396e+01,  …,  3.3060e+01,
  -2.5726e-01,  7.4420e+00],
 [ 5.9742e+00,  6.6437e+00,  6.6705e+00,  …,  5.6413e+00,
   5.5871e-01,  9.9246e+00]],

[[-1.0270e+01, -1.2246e+01, -1.2568e+01,  …, -2.1024e+01,
  -9.3662e+00,  1.7905e+00],
 [-6.8716e+00, -1.0108e+01, -9.8927e+00,  …, -1.7190e+01,
  -1.0741e+01,  1.6301e+00],
 [-5.3720e+00, -1.0581e+01, -3.9099e+00,  …, -1.6360e+01,
  -1.0421e+01,  1.6301e+00],
 …,
 [-3.3740e+01, -3.5881e+01, -3.6110e+01,  …, -3.7142e+01,
  -1.1740e+01,  1.6301e+00],
 [-4.5355e+01, -4.5641e+01, -4.5723e+01,  …, -1.5873e+01,
  -1.1356e+01,  1.6301e+00],
 [-3.5365e+00, -1.5159e+00, -1.5117e+00,  …,  3.2096e+00,
  -3.1349e+00,  8.7797e+00]],

[[-3.1979e+00, -2.2331e+00, -7.4591e-01,  …,  8.1416e-01,
  -6.6423e+00, -9.4584e+00],
 [-3.5334e+00, -2.2618e+00, -1.9908e+00,  …, -7.3434e+00,
  -8.0000e+00, -1.3091e+01],
 [-5.5643e+00, -2.2535e+00, -6.6630e+00,  …, -7.2467e+00,
  -8.0980e+00, -1.3091e+01],
 …,
 [ 1.3816e+01,  7.5939e+00,  7.6490e+00,  …, -9.2757e+00,
  -1.6436e+01, -1.3091e+01],
 [-1.5728e+01, -1.4918e+01, -1.4988e+01,  …, -1.6272e+01,
  -6.6499e+00, -1.3091e+01],
 [-6.0000e+00, -5.1948e+00, -5.2753e+00,  …, -1.3885e+01,
  -8.6886e+00, -1.4262e+01]]],
```

```
[[[ 1.6284e+00,  1.9988e+00, -9.5692e+00,  …, -2.9520e+00,
   -4.7215e+00, -2.1429e+01],
  [ 9.2784e-01,  2.7028e+00, -4.9376e+00,  …, -5.5777e+00,
   -7.1330e+00, -1.4842e+01],
  [ 9.2784e-01,  2.7074e+00, -4.2255e+00,  …, -2.9647e+00,
   -1.8172e+00, -1.5698e+01],
  …,
  [ 9.2784e-01,  1.5331e+01, -3.5167e+01,  …, -9.5787e+00,
    5.4514e+00, -3.2451e+01],
  [ 9.2784e-01,  1.5331e+01, -3.5276e+01,  …,  1.8432e+01,
   -1.4957e+01, -1.4200e+01],
  [-4.3907e-01,  1.1773e+01, -3.3433e+01,  …,  6.7618e+00,
   -2.0505e+01, -9.1558e+00]],

 [[-9.1647e-01, -7.4154e+00, -1.4338e+01,  …, -1.3656e+01,
   -1.4917e+01,  1.6956e+01],
  [ 6.7936e+00,  3.5883e+00, -4.5705e+00,  …,  8.0072e+00,
    5.5767e+00,  1.1236e+01],
  [ 6.7936e+00,  1.8602e+00, -3.5294e+00,  …, -4.8819e-02,
   -2.1295e+00,  2.9719e+00],
  …,
  [ 6.7936e+00, -1.0124e+01, -4.6365e+01,  …,  7.6100e+00,
    1.3308e+01, -1.8445e+01],
  [ 6.7936e+00, -1.0137e+01, -4.6936e+01,  …,  4.4836e+01,
   -2.8615e+00, -3.3945e+01],
  [ 1.5681e+00, -1.0330e+01, -4.7690e+01,  …,  1.8245e+01,
   -2.6771e+01, -1.6451e+00]],

 [[ 7.9561e+00,  1.1296e+01,  2.0663e+01,  …,  2.5564e+01,
    2.6988e+01,  3.3856e+01],
  [-6.0371e+00, -9.8597e-01,  4.2517e+00,  …,  2.6292e+00,
    3.7709e+00,  9.5816e-01],
  [-6.0371e+00, -1.2070e+00,  5.8196e+00,  …,  1.2003e+00,
   -1.1618e+00, -3.9661e+00],
  …,
  [-6.0371e+00, -1.1275e+01,  2.7476e+01,  …, -1.3330e+01,
   -2.2166e+01,  1.2258e+01],
  [-6.0371e+00, -1.1573e+01,  2.8169e+01,  …, -2.4196e+01,
    7.1786e+00,  1.4640e+00],
  [-4.4115e+00, -8.9629e+00,  2.6246e+01,  …, -5.1263e+00,
    2.9842e+01,  4.8269e+00]],

 …,

 [[ 6.7213e+00,  5.9983e+00,  4.7628e+00,  …, -1.9885e+01,
   -1.9543e+01, -2.3276e+01],
  [ 3.0844e+00,  1.6442e+00,  2.4671e+00,  …, -1.3835e+01,
   -1.6433e+01, -1.8987e+01],
```

```
          [ 3.0844e+00,  6.1629e-01,  4.7578e+00,  …, -3.5691e+01,
           -3.2129e+01, -2.6788e+01],
          …,
          [ 3.0844e+00,  1.1032e+01,  1.3253e+01,  …, -8.1039e+00,
           -2.1168e+01,  1.0752e+01],
          [ 3.0844e+00,  1.1083e+01,  1.2773e+01,  …, -7.2916e+00,
            2.8389e+01,  2.5589e+01],
          [ 1.3763e-01,  1.3614e+01,  1.0711e+01,  …,  1.2557e+01,
            3.3317e+01, -1.5404e+01]],

         [[-9.0370e+00, -6.2446e+00, -1.2480e+00,  …,  4.8920e+00,
            5.2277e+00,  1.5300e+01],
          [-1.0071e+01, -5.9393e+00,  5.7891e-01,  …, -1.7539e+01,
           -1.6740e+01, -1.9532e+01],
          [-1.0071e+01, -7.7500e+00,  2.0400e+00,  …, -2.1878e+01,
           -2.0800e+01, -2.1718e+01],
          …,
          [-1.0071e+01,  7.5076e+00,  2.3827e+01,  …, -1.0323e+01,
           -1.1416e+01, -5.3961e+00],
          [-1.0071e+01,  7.6492e+00,  2.4348e+01,  …, -2.0875e+01,
           -6.6673e+00,  1.0668e+01],
          [-4.3723e+00,  1.3902e+01,  2.6040e+01,  …, -1.8574e+01,
            8.3495e+00,  1.2877e+01]],

         [[-5.7716e+00, -5.3314e+00, -6.0386e+00,  …, -5.6954e+00,
           -5.7326e+00, -1.6383e+00],
          [-7.6318e+00, -7.3817e+00, -8.1751e+00,  …, -3.0563e+00,
           -2.2307e+00, -1.0985e+01],
          [-7.6318e+00, -7.6827e+00, -9.4848e+00,  …,  4.6603e+00,
            4.0260e+00, -2.1135e-01],
          …,
          [-7.6318e+00, -1.5814e+01, -4.1477e+00,  …,  1.6992e+00,
           -9.2622e+00, -4.7772e+00],
          [-7.6318e+00, -1.5795e+01, -4.1585e+00,  …,  1.1068e+00,
           -6.3868e+00, -5.9845e+00],
          [-8.7891e+00, -1.7055e+01, -1.3038e+00,  …, -6.4959e+00,
           -4.2390e-01,  4.0332e-01]]]], device='cuda:0',
       grad_fn=<ConvolutionBackward0>)
```

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

```
tensor(1.3869, device='cuda:0', grad_fn=<MeanBackward0>)
```

[ ]:

[ ]: