VGG16_Quantization_Aware_Training

December 13, 2025

[3]:

```
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 = 256
model_name = "VGG16_quant_project_part2_90_prec"   #"Resnet20_quant"
model = VGG16_quant_project_part2()

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 = [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…
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, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
      (weight_quant): weight_quantize_fn()
    )
    (25): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (26): ReLU(inplace=True)
    (27): QuantConv2d(
      16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
      (weight_quant): weight_quantize_fn()
    )
    (28): ReLU(inplace=True)
    (29): QuantConv2d(
      16, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
      (weight_quant): weight_quantize_fn()
    )
    (30): ReLU(inplace=True)
    (31): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (32): QuantConv2d(
      512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
      (weight_quant): weight_quantize_fn()
    )
    (33): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (34): ReLU(inplace=True)
    (35): QuantConv2d(
      512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
      (weight_quant): weight_quantize_fn()
    )
    (36): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (37): ReLU(inplace=True)
    (38): QuantConv2d(
      512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
      (weight_quant): weight_quantize_fn()
    )
    (39): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (40): ReLU(inplace=True)
    (41): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (42): AvgPool2d(kernel_size=1, stride=1, padding=0)
  )
  (classifier): Linear(in_features=512, out_features=10, bias=True)
)
```

[ ]:

```
lr = 1e-3
weight_decay = 1e-6
epochs = 1000
best_prec = 0
# momentum = 0.9
#model = nn.DataParallel(model).cuda()
model.cuda()
criterion = nn.CrossEntropyLoss(label_smoothing=0.1).cuda()
# optimizer = torch.optim.SGD(model.parameters(), lr=lr, momentum=0.9,␣
 ↪weight_decay=weight_decay)
```

```
optimizer = torch.optim.Adam(model.parameters(), lr=lr, weight_decay=weight_decay)
#cudnn.benchmark = True

from torch.optim.lr_scheduler import CosineAnnealingLR
scheduler = CosineAnnealingLR(
    optimizer,
    T_max=epochs,  # 500
    eta_min=1e-10,  # final LR
)



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):

    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)
    scheduler.step()
```

[4]:

```
fdir = 'result/'+str(model_name)+'/model_best.pth.tar'

checkpoint = torch.load(fdir)
model.load_state_dict(checkpoint['state_dict'])
device = torch.device("cuda")

model.cuda()
```

```python
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: 9010/10000 (90%)
```

[86]:

```python
# Pre-hook to save inputs
class SaveOutput:
    def __init__(self):
        self.outputs = []   # list of (name, module_in) for pre-hooks
    def clear(self):
        self.outputs = []

save_output = SaveOutput()
hook_map = []   # keeps the module name for each saved output

def make_pre_hook(name):
    def hook(module, module_in, module_out=None):
        # store (module_name, module_in tensor)
        save_output.outputs.append((name, module_in))
        hook_map.append(name)
    return hook

# register named pre-hooks only for relevant layer types
for name, module in model.named_modules():
    if isinstance(module, (torch.nn.Conv2d, torch.nn.MaxPool2d, torch.nn.ReLU)):
        module.register_forward_pre_hook(make_pre_hook(name))

# run a single batch to populate save_output
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)
```

```python
save_output.clear()
images, labels = next(iter(testloader))
images = images.to(device)
_ = model(images)

# print mapping of saved outputs
print("Saved-hook index -> module name (in order):")
for idx, nm in enumerate(hook_map):
    print(idx, ":", nm)

# show modules that have weight_q (quantized convs) and find their hook indices
print("\nModules that expose weight_q (quantized conv layers):")
quant_names = []
for name, m in model.named_modules():
    if hasattr(m, 'weight_q'):
        print("  -", name)
        quant_names.append(name)

print("\nHook indices for quantized modules (if present in hook_map):")
for qn in quant_names:
    indices = [i for i, nm in enumerate(hook_map) if nm == qn]
    print(qn, "-> hook indices:", indices)
```

```
Saved-hook index -> module name (in order):
0 : conv1
1 : relu
2 : layer1.0.conv1
3 : layer1.0.relu
4 : layer1.0.conv2
5 : layer1.0.relu
6 : layer1.1.conv1
7 : layer1.1.relu
8 : layer1.1.conv2
9 : layer1.1.relu
10 : layer1.2.conv1
11 : layer1.2.relu
12 : layer1.2.conv2
13 : layer1.2.relu
14 : layer2.0.conv1
15 : layer2.0.relu
16 : layer2.0.conv2
17 : layer2.0.downsample.0
18 : layer2.0.relu
19 : layer2.1.conv1
20 : layer2.1.relu
21 : layer2.1.conv2
22 : layer2.1.relu
23 : layer2.2.conv1
```

```
24 : layer2.2.relu
25 : layer2.2.conv2
26 : layer2.2.relu
27 : layer3.0.conv1
28 : layer3.0.relu
29 : layer3.0.conv2
30 : layer3.0.downsample.0
31 : layer3.0.relu
32 : layer3.1.conv1
33 : layer3.1.relu
34 : layer3.1.conv2
35 : layer3.1.relu
36 : layer3.2.conv1
37 : layer3.2.relu
38 : layer3.2.conv2
39 : layer3.2.relu

Modules that expose weight_q (quantized conv layers):
  - layer1.0.conv1
  - layer1.0.conv1.weight_quant
  - layer1.0.conv2
  - layer1.0.conv2.weight_quant
  - layer1.1.conv1
  - layer1.1.conv1.weight_quant
  - layer1.1.conv2
  - layer1.1.conv2.weight_quant
  - layer1.2.conv1
  - layer1.2.conv1.weight_quant
  - layer1.2.conv2
  - layer1.2.conv2.weight_quant
  - layer2.0.conv1
  - layer2.0.conv1.weight_quant
  - layer2.0.conv2
  - layer2.0.conv2.weight_quant
  - layer2.0.downsample.0
  - layer2.0.downsample.0.weight_quant
  - layer2.1.conv1
  - layer2.1.conv1.weight_quant
  - layer2.1.conv2
  - layer2.1.conv2.weight_quant
  - layer2.2.conv1
  - layer2.2.conv1.weight_quant
  - layer2.2.conv2
  - layer2.2.conv2.weight_quant
  - layer3.0.conv1
  - layer3.0.conv1.weight_quant
  - layer3.0.conv2
  - layer3.0.conv2.weight_quant
  - layer3.0.downsample.0
  - layer3.0.downsample.0.weight_quant
  - layer3.1.conv1
  - layer3.1.conv1.weight_quant
  - layer3.1.conv2
  - layer3.1.conv2.weight_quant
  - layer3.2.conv1
  - layer3.2.conv1.weight_quant
  - layer3.2.conv2
  - layer3.2.conv2.weight_quant

Hook indices for quantized modules (if present in hook_map):
layer1.0.conv1 -> hook indices: [2]
layer1.0.conv1.weight_quant -> hook indices: []
layer1.0.conv2 -> hook indices: [4]
layer1.0.conv2.weight_quant -> hook indices: []
layer1.1.conv1 -> hook indices: [6]
layer1.1.conv1.weight_quant -> hook indices: []
layer1.1.conv2 -> hook indices: [8]
layer1.1.conv2.weight_quant -> hook indices: []
layer1.2.conv1 -> hook indices: [10]
layer1.2.conv1.weight_quant -> hook indices: []
layer1.2.conv2 -> hook indices: [12]
layer1.2.conv2.weight_quant -> hook indices: []
layer2.0.conv1 -> hook indices: [14]
layer2.0.conv1.weight_quant -> hook indices: []
layer2.0.conv2 -> hook indices: [16]
layer2.0.conv2.weight_quant -> hook indices: []
layer2.0.downsample.0 -> hook indices: [17]
layer2.0.downsample.0.weight_quant -> hook indices: []
layer2.1.conv1 -> hook indices: [19]
layer2.1.conv1.weight_quant -> hook indices: []
layer2.1.conv2 -> hook indices: [21]
layer2.1.conv2.weight_quant -> hook indices: []
layer2.2.conv1 -> hook indices: [23]
layer2.2.conv1.weight_quant -> hook indices: []
layer2.2.conv2 -> hook indices: [25]
layer2.2.conv2.weight_quant -> hook indices: []
layer3.0.conv1 -> hook indices: [27]
layer3.0.conv1.weight_quant -> hook indices: []
layer3.0.conv2 -> hook indices: [29]
layer3.0.conv2.weight_quant -> hook indices: []
layer3.0.downsample.0 -> hook indices: [30]
layer3.0.downsample.0.weight_quant -> hook indices: []
layer3.1.conv1 -> hook indices: [32]
layer3.1.conv1.weight_quant -> hook indices: []
layer3.1.conv2 -> hook indices: [34]
layer3.1.conv2.weight_quant -> hook indices: []
```

```
layer3.2.conv1 -> hook indices: [36]
layer3.2.conv1.weight_quant -> hook indices: []
layer3.2.conv2 -> hook indices: [38]
layer3.2.conv2.weight_quant -> hook indices: []
```

[ ]:
```
# 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)
```

[ ]:
```
#send an input and grap the value by using prehook like HW3
```

[87]:
```
# Find x_int and w_int for the 2nd convolution layer, which is the first␣
 ↪quantized layer
mod = dict(model.named_modules())['layer1.0.conv1']
mod.show_params()
w_bit = 4
weight_q = mod.weight_q.detach().cuda() # quantized value is stored during the␣
 ↪training
w_alpha = 2.087000 # alpha is defined in model already. bring it out here,␣
 ↪should be 8 according to design
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, it should be␣
 ↪extremely close to integers
```

```
clipping threshold weight alpha: 2.087000, activation alpha: 3.201000
tensor([[[[ 2.9995, -3.9993,  1.9996],
          [-0.9998, -5.9989,  3.9993],
          [-6.9987, -3.9993, -2.9995]],

         [[ 0.0000,  0.0000,  1.9996],
          [-1.9996, -1.9996, -0.0000],
          [-1.9996, -3.9993, -0.9998]],

         [[ 0.0000, -0.9998,  0.9998],
          [ 0.0000, -3.9993, -0.9998],
          [ 1.9996, -2.9995, -1.9996]],

         …,

         [[ 2.9995,  0.0000, -0.0000],
          [-0.0000, -3.9993, -2.9995],
  [ 0.9998, -2.9995, -2.9995]],

 [[ 1.9996,  0.0000,  1.9996],
  [ 0.9998, -0.9998, -3.9993],
  [ 2.9995,  4.9991,  0.0000]],

 [[ 1.9996, -2.9995,  0.9998],
  [ 5.9989,  4.9991,  3.9993],
  [-5.9989, -1.9996,  0.0000]]],


[[[ 0.9998, -0.9998,  0.0000],
  [ 0.9998, -1.9996, -0.0000],
  [-0.9998, -0.9998, -0.9998]],

 [[-0.0000,  0.9998,  0.9998],
  [ 0.9998,  0.9998,  1.9996],
  [ 0.0000,  0.0000,  0.9998]],

 [[-0.0000, -0.0000, -0.0000],
  [ 0.0000,  0.0000,  0.0000],
  [-0.9998,  0.0000,  0.9998]],

 …,

 [[-0.9998,  0.0000,  0.9998],
  [-2.9995, -0.9998, -0.0000],
  [-5.9989, -3.9993, -2.9995]],

 [[-0.9998,  0.0000, -0.0000],
  [-0.9998,  0.0000, -0.9998],
  [ 0.9998,  0.9998, -0.9998]],

 [[ 0.9998,  0.0000,  0.9998],
  [ 0.9998,  0.0000,  0.9998],
  [ 0.0000,  0.0000,  0.0000]]],


[[[ 2.9995,  5.9989, -0.9998],
  [ 2.9995,  6.9987, -3.9993],
  [-0.9998,  1.9996, -6.9987]],

 [[-0.0000, -1.9996, -2.9995],
  [ 0.0000, -0.9998, -0.9998],
  [-1.9996, -1.9996, -0.9998]],

 [[-6.9987,  0.0000,  2.9995],
  [-4.9991,  4.9991,  6.9987],
  [-1.9996,  3.9993,  4.9991]],

 …,

 [[-6.9987, -3.9993,  1.9996],
  [-6.9987,  0.9998,  6.9987],
  [-6.9987,  0.9998,  5.9989]],

 [[-1.9996, -0.0000,  0.9998],
  [-1.9996,  0.0000, -1.9996],
  [-2.9995, -0.0000, -4.9991]],

 [[ 0.0000,  2.9995, -0.9998],
  [ 0.9998,  6.9987,  5.9989],
  [ 0.0000, -0.9998, -0.9998]]],


…,


[[[-3.9993,  5.9989, -2.9995],
  [-6.9987,  6.9987,  0.9998],
  [ 2.9995, -6.9987, -0.9998]],

 [[ 0.0000,  2.9995,  2.9995],
  [-3.9993, -0.9998, -1.9996],
  [-3.9993, -2.9995, -2.9995]],

 [[-2.9995, -0.9998, -0.9998],
  [-0.9998,  2.9995, -0.9998],
  [-0.0000,  0.9998, -0.9998]],

 …,

 [[-0.9998, -0.0000, -0.0000],
  [ 0.9998,  0.9998,  1.9996],
  [-0.9998,  0.0000, -1.9996]],

 [[ 0.9998,  0.9998, -5.9989],
  [ 1.9996,  2.9995, -4.9991],
  [-3.9993, -0.0000,  1.9996]],

 [[-4.9991,  1.9996, -2.9995],
  [-3.9993,  3.9993,  0.9998],
  [-1.9996, -6.9987, -3.9993]]],


[[[ 0.9998,  3.9993, -1.9996],
  [ 6.9987, -6.9987, -3.9993],
  [-0.0000, -5.9989, -0.0000]],

 [[-5.9989, -5.9989, -4.9991],
  [-1.9996, -0.9998, -0.0000],
  [ 0.9998,  1.9996,  0.9998]],

 [[ 4.9991,  4.9991, -4.9991],
  [ 4.9991,  3.9993, -2.9995],
  [-0.0000,  0.0000,  1.9996]],

 …,

 [[-0.9998,  1.9996, -1.9996],
  [-0.0000,  2.9995, -3.9993],
  [-6.9987, -3.9993, -6.9987]],

 [[-1.9996,  0.0000,  0.0000],
  [ 0.0000,  0.9998,  3.9993],
  [-1.9996, -1.9996,  1.9996]],

 [[ 0.9998,  5.9989, -2.9995],
  [-2.9995, -5.9989, -1.9996],
  [ 1.9996,  0.9998,  0.9998]]],


[[[ 1.9996, -3.9993,  2.9995],
  [-3.9993,  6.9987, -6.9987],
  [ 0.9998,  1.9996, -3.9993]],

 [[ 1.9996, -0.0000, -1.9996],
  [-0.9998, -0.0000, -1.9996],
  [-0.0000,  1.9996,  0.9998]],

 [[-6.9987, -5.9989, -2.9995],
  [-0.0000, -0.9998, -0.9998],
  [ 4.9991,  2.9995,  0.0000]],

 …,

 [[-6.9987, -6.9987, -5.9989],
  [-2.9995,  0.9998,  0.9998],
  [ 4.9991,  4.9991,  5.9989]],

 [[ 0.9998,  1.9996,  3.9993],
  [ 2.9995,  0.9998, -1.9996],
  [ 3.9993,  4.9991, -2.9995]],
          [[-1.9996,  0.9998,  0.0000],
           [-1.9996,  6.9987, -0.9998],
           [-3.9993, -2.9995,  3.9993]]]], device='cuda:0')
```

[94]:
```python
x_bit = 4
# input of the 2nd conv layer
saved_entry = save_output.outputs[2]
module_name, module_in = saved_entry
print ("Module name for saved input:", module_name)
x = module_in[0].detach().clone()
x_alpha  = 3.201000
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  # x_int can be calculated by x_q and x_delta
print(x_int) # you should see clean integer numbers
```

```
Module name for saved input: layer1.0.conv1
tensor([[[[ 3.0000,  6.0000,  7.0000,  …,  8.0000,  8.0000,  6.0000],
          [ 4.0000,  7.0000,  8.0000,  …,  9.0000,  8.0000, 12.0000],
          [ 3.0000,  7.0000,  8.0000,  …,  9.0000,  9.0000, 11.0000],
          …,
          [13.0000,  6.0000,  5.0000,  …,  7.0000,  3.0000,  0.0000],
          [ 9.0000,  7.0000,  5.0000,  …,  2.0000, 11.0000,  0.0000],
          [ 5.0000,  5.0000,  4.0000,  …,  2.0000,  5.0000,  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,  1.0000,  1.0000,  …,  1.0000,  2.0000,  2.0000]],

         [[ 0.0000,  0.0000,  0.0000,  …,  0.0000,  0.0000,  0.0000],
          [ 0.0000,  0.0000,  0.0000,  …,  0.0000,  0.0000,  0.0000],
          [ 0.0000,  0.0000,  0.0000,  …,  0.0000,  0.0000,  0.0000],
          …,
          [12.0000,  9.0000,  9.0000,  …,  9.0000,  7.0000,  4.0000],
          [12.0000,  8.0000,  8.0000,  …,  8.0000,  8.0000,  3.0000],
          [ 6.0000,  4.0000,  4.0000,  …,  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,  0.0000,  0.0000],
  [ 0.0000,  0.0000,  0.0000,  …,  0.0000,  0.0000,  0.0000],
  …,
  [15.0000, 15.0000, 15.0000,  …, 15.0000, 13.0000,  8.0000],
  [14.0000, 14.0000, 14.0000,  …, 15.0000, 14.0000,  8.0000],
  [10.0000,  9.0000,  9.0000,  …, 10.0000,  9.0000,  6.0000]],

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

 [[ 5.0000,  6.0000,  6.0000,  …,  3.0000,  2.0000,  2.0000],
  [ 5.0000,  7.0000,  7.0000,  …,  7.0000,  7.0000,  8.0000],
  [ 5.0000,  7.0000,  7.0000,  …,  8.0000,  8.0000,  8.0000],
  …,
  [10.0000,  6.0000,  4.0000,  …,  6.0000,  5.0000,  3.0000],
  [ 7.0000,  5.0000,  5.0000,  …,  3.0000,  7.0000,  0.0000],
  [ 4.0000,  3.0000,  2.0000,  …,  0.0000,  0.0000,  1.0000]]],


[[[15.0000,  5.0000,  6.0000,  …,  6.0000,  6.0000, 15.0000],
  [15.0000,  6.0000,  6.0000,  …,  6.0000,  6.0000, 10.0000],
  [15.0000,  6.0000,  6.0000,  …,  6.0000,  6.0000, 10.0000],
  …,
  [ 8.0000,  3.0000,  7.0000,  …,  8.0000,  8.0000,  8.0000],
  [ 6.0000,  4.0000,  7.0000,  …,  8.0000,  8.0000,  8.0000],
  [ 6.0000,  6.0000,  7.0000,  …,  8.0000,  8.0000, 12.0000]],

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

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

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

 [[ 6.0000,  0.0000,  0.0000,  …,  0.0000,  0.0000,  0.0000],
  [12.0000,  2.0000,  2.0000,  …,  2.0000,  2.0000,  0.0000],
  [12.0000,  2.0000,  2.0000,  …,  2.0000,  2.0000,  0.0000],
  …,
  [ 0.0000,  0.0000,  0.0000,  …,  5.0000,  4.0000,  0.0000],
  [ 0.0000,  0.0000,  0.0000,  …,  5.0000,  4.0000,  0.0000],
  [ 0.0000,  0.0000,  1.0000,  …,  5.0000,  5.0000,  0.0000]],

 [[15.0000, 15.0000, 15.0000,  …, 15.0000, 15.0000, 15.0000],
  [ 8.0000,  9.0000,  9.0000,  …,  9.0000,  9.0000,  9.0000],
  [ 8.0000,  8.0000,  8.0000,  …,  8.0000,  8.0000,  8.0000],
  …,
  [ 6.0000,  3.0000,  5.0000,  …,  8.0000,  8.0000,  7.0000],
  [ 5.0000,  4.0000,  6.0000,  …,  9.0000,  8.0000,  7.0000],
  [ 4.0000,  3.0000,  3.0000,  …, 12.0000, 13.0000, 12.0000]]],


[[[15.0000,  8.0000,  5.0000,  …,  4.0000,  6.0000, 15.0000],
  [14.0000,  7.0000,  5.0000,  …,  5.0000,  7.0000, 10.0000],
  [13.0000,  6.0000,  5.0000,  …,  4.0000,  8.0000,  9.0000],
  …,
  [ 3.0000,  8.0000,  8.0000,  …,  9.0000,  5.0000,  2.0000],
  [ 2.0000,  8.0000,  7.0000,  …,  5.0000,  7.0000,  3.0000],
  [ 2.0000,  5.0000,  5.0000,  …,  2.0000,  4.0000,  0.0000]],

 [[ 5.0000,  7.0000,  5.0000,  …, 10.0000, 11.0000,  9.0000],
  [ 3.0000,  3.0000,  2.0000,  …,  4.0000,  5.0000,  4.0000],
  [ 2.0000,  3.0000,  2.0000,  …,  4.0000,  4.0000,  4.0000],
  …,
  [ 0.0000,  0.0000,  0.0000,  …,  0.0000,  0.0000,  0.0000],
  [ 0.0000,  0.0000,  0.0000,  …,  0.0000,  0.0000,  0.0000],
  [ 2.0000,  4.0000,  4.0000,  …,  4.0000,  5.0000,  4.0000]],

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

 …,

 [[ 8.0000, 10.0000, 10.0000,  …,  5.0000,  4.0000,  4.0000],
  [ 8.0000, 10.0000, 10.0000,  …,  5.0000,  5.0000,  5.0000],
  [ 7.0000,  9.0000,  9.0000,  …,  5.0000,  5.0000,  5.0000],
  …,
  [ 8.0000,  5.0000,  5.0000,  …,  4.0000,  4.0000,  3.0000],
  [ 8.0000,  5.0000,  5.0000,  …,  4.0000,  4.0000,  3.0000],
  [ 7.0000,  5.0000,  5.0000,  …,  5.0000,  4.0000,  2.0000]],

 [[ 5.0000,  0.0000,  0.0000,  …,  0.0000,  0.0000,  0.0000],
  [ 9.0000,  0.0000,  0.0000,  …,  3.0000,  3.0000,  0.0000],
  [ 9.0000,  0.0000,  0.0000,  …,  3.0000,  3.0000,  0.0000],
  …,
  [ 0.0000,  3.0000,  3.0000,  …,  1.0000,  1.0000, 15.0000],
  [ 0.0000,  3.0000,  3.0000,  …,  1.0000,  2.0000, 15.0000],
  [ 0.0000,  2.0000,  2.0000,  …,  0.0000,  2.0000, 11.0000]],

 [[15.0000, 15.0000, 12.0000,  …, 15.0000, 15.0000, 15.0000],
  [ 9.0000,  8.0000,  8.0000,  …,  9.0000,  9.0000,  9.0000],
  [ 8.0000,  7.0000,  7.0000,  …,  8.0000,  9.0000,  8.0000],
  …,
  [ 6.0000,  5.0000,  5.0000,  …,  5.0000,  3.0000,  3.0000],
  [ 5.0000,  4.0000,  4.0000,  …,  3.0000,  4.0000,  4.0000],
  [ 1.0000,  0.0000,  0.0000,  …,  0.0000,  0.0000,  0.0000]]],


…,


[[[11.0000,  6.0000,  8.0000,  …,  9.0000, 14.0000,  1.0000],
  [ 9.0000,  8.0000,  8.0000,  …, 10.0000, 13.0000,  8.0000],
  [ 9.0000,  7.0000,  9.0000,  …, 13.0000, 11.0000,  7.0000],
  …,
  [ 6.0000,  8.0000, 12.0000,  …, 15.0000,  6.0000,  5.0000],
  [ 5.0000,  7.0000, 12.0000,  …,  7.0000, 14.0000,  6.0000],
  [ 7.0000,  8.0000, 12.0000,  …,  3.0000,  9.0000,  5.0000]],

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

 …,

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

 [[ 3.0000,  0.0000,  1.0000,  …,  0.0000,  0.0000,  7.0000],
  [ 5.0000,  1.0000,  2.0000,  …,  0.0000,  0.0000,  6.0000],
  [ 5.0000,  1.0000,  2.0000,  …,  0.0000,  0.0000,  7.0000],
  …,
  [ 9.0000,  5.0000,  1.0000,  …,  1.0000,  0.0000,  5.0000],
  [ 8.0000,  7.0000,  2.0000,  …, 11.0000,  4.0000,  5.0000],
  [ 6.0000,  6.0000,  4.0000,  …,  8.0000,  6.0000,  5.0000]],

 [[13.0000, 13.0000, 13.0000,  …,  8.0000,  4.0000,  0.0000],
  [ 7.0000,  9.0000,  8.0000,  …, 10.0000, 10.0000,  9.0000],
  [ 5.0000,  7.0000,  7.0000,  …, 11.0000,  8.0000,  7.0000],
  …,
  [ 5.0000,  9.0000, 10.0000,  …, 15.0000,  8.0000,  6.0000],
  [ 4.0000,  7.0000,  9.0000,  …,  0.0000,  7.0000,  6.0000],
  [ 6.0000,  8.0000,  9.0000,  …,  0.0000,  2.0000,  5.0000]]],


[[[14.0000,  7.0000,  7.0000,  …,  7.0000,  7.0000,  9.0000],
  [12.0000,  7.0000,  7.0000,  …,  7.0000,  7.0000,  5.0000],
  [12.0000,  7.0000,  7.0000,  …,  7.0000,  7.0000,  6.0000],
  …,
  [10.0000,  7.0000,  7.0000,  …,  8.0000,  6.0000,  5.0000],
  [ 9.0000,  7.0000,  7.0000,  …,  7.0000,  7.0000,  6.0000],
  [ 9.0000,  7.0000,  6.0000,  …,  5.0000,  6.0000,  2.0000]],

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

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

…,

[[ 7.0000,  8.0000,  8.0000,  …,  8.0000,  8.0000,  6.0000],
 [ 7.0000,  8.0000,  8.0000,  …,  8.0000,  8.0000,  6.0000],
 [ 7.0000,  8.0000,  8.0000,  …,  8.0000,  8.0000,  6.0000],
 …,
 [ 7.0000,  7.0000,  7.0000,  …,  5.0000,  5.0000,  4.0000],
 [ 7.0000,  7.0000,  7.0000,  …,  6.0000,  6.0000,  4.0000],
 [ 5.0000,  5.0000,  5.0000,  …,  5.0000,  5.0000,  4.0000]],

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

[[14.0000, 14.0000, 14.0000,  …, 13.0000, 13.0000, 13.0000],
 [ 8.0000,  7.0000,  7.0000,  …,  7.0000,  7.0000,  6.0000],
 [ 8.0000,  7.0000,  7.0000,  …,  7.0000,  7.0000,  7.0000],
 …,
 [ 7.0000,  6.0000,  7.0000,  …,  6.0000,  5.0000,  6.0000],
 [ 7.0000,  7.0000,  7.0000,  …,  4.0000,  6.0000,  6.0000],
 [ 8.0000,  7.0000,  7.0000,  …,  2.0000,  3.0000,  3.0000]]],


[[[ 0.0000,  9.0000,  7.0000,  …,  8.0000,  8.0000,  0.0000],
 [ 4.0000,  6.0000,  7.0000,  …,  7.0000,  7.0000,  3.0000],
 [ 5.0000,  6.0000,  8.0000,  …,  7.0000,  7.0000,  2.0000],
 …,
 [10.0000,  7.0000,  8.0000,  …,  7.0000,  8.0000,  2.0000],
 [ 6.0000,  5.0000,  7.0000,  …,  7.0000,  8.0000,  2.0000],
 [ 9.0000, 10.0000,  5.0000,  …,  4.0000,  4.0000,  0.0000]],

[[ 0.0000,  0.0000,  0.0000,  …,  0.0000,  0.0000,  0.0000],
 [ 0.0000,  0.0000,  0.0000,  …,  0.0000,  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,  …,  5.0000,  5.0000,  4.0000]],

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


        …,


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

        [[ 1.0000,  5.0000,  5.0000,  …,  7.0000,  8.0000, 15.0000],
         [ 0.0000,  2.0000,  4.0000,  …,  2.0000,  3.0000, 15.0000],
         [ 0.0000,  2.0000,  5.0000,  …,  1.0000,  3.0000, 15.0000],
         …,
         [ 7.0000,  4.0000,  3.0000,  …,  1.0000,  3.0000, 15.0000],
         [ 6.0000,  4.0000,  4.0000,  …,  2.0000,  3.0000, 15.0000],
         [ 5.0000,  3.0000,  4.0000,  …,  1.0000,  2.0000, 12.0000]],

        [[ 0.0000,  0.0000,  0.0000,  …,  0.0000,  0.0000,  0.0000],
         [ 8.0000,  7.0000,  7.0000,  …,  5.0000,  5.0000,  4.0000],
         [ 7.0000,  5.0000,  6.0000,  …,  5.0000,  5.0000,  4.0000],
         …,
         [ 6.0000,  5.0000,  4.0000,  …,  5.0000,  5.0000,  4.0000],
         [ 3.0000,  6.0000,  9.0000,  …,  5.0000,  5.0000,  4.0000],
         [ 9.0000,  9.0000,  4.0000,  …,  0.0000,  0.0000,  0.0000]]]],
      device='cuda:0')
```

[95]:

```
conv_int = torch.nn.Conv2d(in_channels = 64, out_channels=64, 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 * x_delta * w_delta  # recover with x_delta and␣
 ↪w_delta
print(output_recovered)
```

```
tensor([[[[-1.1450e+00, -1.1450e+01, -9.7327e+00,  …, -4.1984e+00,
           -6.1068e+00, -1.8320e+01],
          [-1.5903e+00, -1.2913e+01, -9.9235e+00,  …, -2.6717e+00,
           -3.6259e+00, -1.4376e+01],
          [-3.4987e+00, -1.2086e+01, -6.3612e+00,  …, -6.9973e-01,
            3.8167e-01, -1.1450e+01],
          …,
          [-2.9134e+01, -2.6844e+01, -2.6399e+01,  …, -2.2773e+01,
           -3.2697e+01, -2.9516e+01],
          [-2.2773e+01, -2.2328e+01, -2.5063e+01,  …, -1.7875e+01,
           -2.7290e+01, -2.3155e+01],
          [-2.1819e+01, -1.5012e+01, -1.9211e+01,  …, -1.5203e+01,
           -1.3295e+01, -2.4045e+01]],

         [[ 5.4070e+01,  5.2798e+01,  5.3371e+01,  …,  4.8345e+01,
            4.5992e+01,  4.1666e+01],
          [ 4.9490e+01,  4.8600e+01,  5.0063e+01,  …,  4.3574e+01,
            4.1793e+01,  3.8358e+01],
          [ 4.6437e+01,  4.6500e+01,  4.8472e+01,  …,  4.4719e+01,
            4.3129e+01,  3.8803e+01],
          …,
          [ 1.8766e+01,  2.1565e+01,  2.4300e+01,  …,  2.1628e+01,
            2.4618e+01,  3.6704e+01],
          [ 2.1755e+01,  2.4936e+01,  2.5000e+01,  …,  2.3536e+01,
            2.5699e+01,  3.4160e+01],
          [ 3.7722e+01,  4.1602e+01,  3.9567e+01,  …,  3.8994e+01,
            4.2493e+01,  4.2811e+01]],

         [[-2.3155e+01, -1.6603e+01, -1.6221e+01,  …, -2.1310e+01,
           -1.5649e+01, -6.1068e+00],
          [-1.2150e+01, -7.6335e+00, -8.9693e+00,  …, -1.1895e+01,
           -6.3612e+00, -1.2722e+00],
          [-1.7939e+01, -1.2913e+01, -1.2086e+01,  …, -1.0369e+01,
           -5.8523e+00, -2.8625e+00],
          …,
          [-7.3154e+00, -9.6054e+00, -1.2850e+01,  …, -1.9974e+01,
           -1.9974e+01, -2.8180e+01],
          [-5.8523e+00, -1.0114e+01, -1.2277e+01,  …, -8.9693e+00,
           -1.4440e+01, -2.2646e+01],
          [-5.3434e+00, -8.3332e+00, -8.9057e+00,  …, -4.5165e+00,
           -1.2468e+01, -1.7557e+01]],


         …,
 [[-8.3332e+00, -7.8243e+00, -6.8065e+00,  …, -1.3040e+01,
   -1.4949e+01, -2.0229e+01],
  [-8.7785e+00, -8.7785e+00, -8.7149e+00,  …, -1.2722e+01,
   -1.1323e+01, -1.9529e+01],
  [-1.7557e+01, -1.9529e+01, -1.1450e+01,  …, -1.4694e+01,
   -1.2468e+01, -1.3740e+01],
  …,
  [-1.5585e+01, -8.8421e+00, -1.4694e+01,  …, -8.5240e+00,
   -1.6921e+01, -2.6017e+01],
  [-1.7557e+01, -8.7149e+00, -7.5698e+00,  …, -6.9973e+00,
   -9.1601e+00, -3.5241e+01],
  [-1.1832e+01, -8.3332e+00,  3.7531e+00,  …,  1.3995e+00,
   -5.6615e+00, -8.9693e+00]],

 [[-2.7099e+01, -2.1437e+01, -1.9402e+01,  …, -3.1297e+01,
   -2.9007e+01, -2.3155e+01],
  [-1.5394e+01, -7.5062e+00, -4.7073e+00,  …, -1.6348e+01,
   -1.3231e+01, -1.0814e+01],
  [-2.9262e+01, -1.7875e+01, -1.5839e+01,  …, -1.6094e+01,
   -1.0941e+01, -1.0114e+01],
  …,
  [ 7.3790e+00,  1.8448e+00,  2.7353e+00,  …,  4.7073e+00,
    1.1387e+01,  2.0165e+01],
  [ 5.0890e+00,  2.3536e+00,  1.9720e+00,  …, -2.8625e+00,
    1.5903e+00,  5.7251e+00],
  [ 2.4554e+01,  1.7048e+01,  1.7939e+01,  …,  1.6666e+01,
    2.3982e+01,  2.1374e+01]],

 [[-2.6272e+01, -2.3664e+01, -2.4872e+01,  …, -2.6781e+01,
   -2.3727e+01, -2.3409e+01],
  [-1.8066e+01, -1.8511e+01, -1.9974e+01,  …, -1.4440e+01,
   -1.2977e+01, -1.3295e+01],
  [-1.9084e+01, -2.3218e+01, -2.1883e+01,  …, -8.9693e+00,
   -1.2341e+01, -1.4567e+01],
  …,
  [-2.5063e+01, -2.0101e+01, -1.7557e+01,  …, -3.6068e+01,
   -3.8167e+01, -4.2366e+01],
  [-2.3028e+01, -3.0788e+01, -2.0419e+01,  …, -3.1488e+01,
   -2.6399e+01, -1.9974e+01],
  [-2.7099e+01, -3.3269e+01, -3.1552e+01,  …, -3.0597e+01,
   -3.8994e+01, -1.3040e+01]]],


[[[-5.2162e+00, -9.2874e+00, -9.0329e+00,  …, -8.7149e+00,
   -8.5876e+00, -1.2723e-01],
  [-9.7327e+00, -1.2977e+01, -1.2595e+01,  …, -1.2150e+01,
   -1.2404e+01, -4.4529e+00],
  [-9.5418e+00, -1.2722e+01, -1.2659e+01,  …, -1.2850e+01,
  -1.3168e+01, -4.5165e+00],
 …,
 [-5.4070e+00, -1.6539e+00, -1.2659e+01,  …, -1.8066e+01,
  -1.7748e+01, -1.0878e+01],
 [-6.5520e+00, -6.7429e+00, -1.1768e+01,  …, -1.5521e+01,
  -1.4122e+01, -1.0178e+01],
 [-5.4706e+00, -1.1259e+01, -9.5418e+00,  …, -1.4694e+01,
  -1.5076e+01, -1.3295e+01]],

[[ 4.9299e+01,  4.7137e+01,  4.6819e+01,  …,  4.7009e+01,
   4.7009e+01,  5.2734e+01],
 [ 5.0063e+01,  4.9109e+01,  4.8663e+01,  …,  4.8727e+01,
   4.8218e+01,  5.4325e+01],
 [ 4.9935e+01,  4.9109e+01,  4.8663e+01,  …,  4.8727e+01,
   4.8218e+01,  5.4643e+01],
 …,
 [ 3.9376e+01,  4.2811e+01,  4.0775e+01,  …,  3.1234e+01,
   3.2379e+01,  4.0712e+01],
 [ 3.7404e+01,  3.7340e+01,  3.6259e+01,  …,  3.2951e+01,
   3.4223e+01,  4.2111e+01],
 [ 2.9452e+01,  2.7417e+01,  2.7862e+01,  …,  4.4210e+01,
   4.6564e+01,  5.4452e+01]],

[[-1.3422e+01, -1.5012e+01, -1.4567e+01,  …, -1.4758e+01,
  -1.5203e+01, -1.9084e+01],
 [-9.8599e+00, -1.3613e+01, -1.3168e+01,  …, -1.3295e+01,
  -1.3549e+01, -1.6476e+01],
 [-9.4782e+00, -1.3549e+01, -1.3677e+01,  …, -1.3677e+01,
  -1.3422e+01, -1.6794e+01],
 …,
 [-1.0305e+01, -1.5394e+01, -1.8638e+01,  …, -8.7785e+00,
  -1.2532e+01, -1.2532e+01],
 [-1.7430e+01, -1.4376e+01, -1.6285e+01,  …, -7.6335e+00,
  -1.1577e+01, -1.1895e+01],
 [-1.4694e+01, -1.1069e+01, -1.0814e+01,  …, -5.1526e+00,
  -7.4426e+00, -8.2696e+00]],


…,


[[-9.2874e+00, -2.4809e+00, -2.8625e+00,  …, -3.1806e+00,
  -2.2900e+00,  9.5418e-01],
 [-2.0229e+01, -1.1959e+01, -1.1895e+01,  …, -1.2214e+01,
  -1.0496e+01, -5.5979e+00],
 [-1.9147e+01, -1.1768e+01, -1.0496e+01,  …, -1.2086e+01,
  -1.0687e+01, -5.6615e+00],
 …,
 [ 2.2264e+00, -1.7939e+01, -1.4631e+00,  …, -1.8129e+01,
  -1.6794e+01, -6.8701e+00],
  [-9.6690e+00, -1.2850e+01, -6.3612e+00,  …, -1.3867e+01,
   -1.6094e+01, -5.0890e+00],
  [-1.6666e+01, -7.3154e+00, -7.4426e+00,  …, -8.6512e+00,
   -5.4070e+00,  2.5445e-01]],

 [[-5.5343e+00, -7.5062e+00, -7.8879e+00,  …, -7.3790e+00,
   -8.0151e+00, -1.1514e+01],
  [-3.3078e+00, -5.3434e+00, -6.0432e+00,  …, -6.4248e+00,
   -6.8065e+00, -1.1387e+01],
  [-4.5165e+00, -6.6793e+00, -6.9973e+00,  …, -6.4884e+00,
   -6.6157e+00, -1.1259e+01],
  …,
  [-2.2264e+00, -8.3332e+00, -1.0242e+01,  …, -2.4809e+00,
   -1.7811e+00, -7.4426e+00],
  [ 5.5979e+00, -1.0496e+01, -8.2060e+00,  …, -2.9262e+00,
   -5.6615e+00, -8.3332e+00],
  [-2.0356e+00, -1.1387e+01, -1.1259e+01,  …,  3.4987e+00,
    1.0814e+00, -1.0178e+00]],

 [[-7.9515e+00, -1.2086e+01, -1.2023e+01,  …, -1.2786e+01,
   -1.2277e+01, -1.3359e+01],
  [-1.2086e+01, -1.8129e+01, -1.7684e+01,  …, -1.6984e+01,
   -1.7493e+01, -1.9338e+01],
  [-1.1577e+01, -1.6857e+01, -1.7112e+01,  …, -1.7112e+01,
   -1.7875e+01, -1.9720e+01],
  …,
  [-1.7302e+01, -1.8002e+01, -2.4491e+01,  …, -1.3168e+01,
   -1.6794e+01, -1.5903e+01],
  [-2.5063e+01, -9.9235e+00, -1.9847e+01,  …, -1.1768e+01,
   -1.3104e+01, -1.5140e+01],
  [-1.2341e+01, -5.4070e+00, -8.1424e+00,  …, -1.3040e+01,
   -1.0623e+01, -1.3486e+01]]],


[[[-1.3422e+01, -1.7239e+01, -1.4440e+01,  …, -1.2277e+01,
   -9.6054e+00, -4.4528e-01],
  [-1.2086e+01, -1.9084e+01, -1.4694e+01,  …, -1.4822e+01,
   -1.1196e+01, -4.9617e+00],
  [-1.1832e+01, -2.0292e+01, -1.5267e+01,  …, -1.5331e+01,
   -1.0051e+01, -7.0609e+00],
  …,
  [-1.9274e+01, -1.2532e+01, -1.4249e+01,  …, -6.6793e+00,
   -4.8345e+00, -1.7366e+01],
  [-1.6476e+01, -1.0814e+01, -1.3677e+01,  …, -4.1984e+00,
   -7.4426e+00, -1.5331e+01],
  [-1.7811e+01, -1.3168e+01, -1.3867e+01,  …, -9.4782e+00,
   -7.6971e+00, -1.5458e+01]],
[[ 3.6068e+01,  3.4605e+01,  3.9312e+01,  …,  4.8472e+01,
   4.9808e+01,  5.5915e+01],
 [ 3.9249e+01,  3.8931e+01,  4.4528e+01,  …,  5.0635e+01,
   5.3243e+01,  5.7505e+01],
 [ 3.9630e+01,  3.8803e+01,  4.4974e+01,  …,  5.1844e+01,
   5.3943e+01,  5.7505e+01],
 …,
 [ 4.3447e+01,  4.4783e+01,  4.1793e+01,  …,  4.4465e+01,
   4.8154e+01,  4.8409e+01],
 [ 3.8040e+01,  3.8549e+01,  3.6386e+01,  …,  4.7645e+01,
   4.9045e+01,  4.8472e+01],
 [ 3.2060e+01,  3.2569e+01,  3.1488e+01,  …,  3.8994e+01,
   4.1157e+01,  3.9439e+01]],

[[-4.5165e+00, -1.8447e+00, -1.3359e+01,  …, -1.2722e+01,
  -1.7939e+01, -2.0292e+01],
 [-1.1577e+01, -6.0432e+00, -1.5140e+01,  …, -1.0369e+01,
  -1.7302e+01, -1.9720e+01],
 [-1.2023e+01, -5.5979e+00, -1.6412e+01,  …, -9.3510e+00,
  -1.6921e+01, -2.0928e+01],
 …,
 [-1.7493e+01, -1.6285e+01, -1.4186e+01,  …, -1.8129e+01,
  -1.7239e+01, -1.8193e+01],
 [-1.6603e+01, -1.6857e+01, -1.3613e+01,  …, -1.9211e+01,
  -2.2010e+01, -1.9211e+01],
 [-1.2659e+01, -1.0750e+01, -8.2060e+00,  …, -1.6857e+01,
  -2.1819e+01, -1.6730e+01]],


…,

[[-2.0992e+00, -8.6512e+00, -1.2214e+01,  …, -4.3892e+00,
  -3.3078e+00,  1.8448e+00],
 [-8.0787e+00, -1.5140e+01, -2.0674e+01,  …, -1.3931e+01,
  -1.4122e+01,  9.5418e-01],
 [-1.3677e+01, -1.6603e+01, -1.9720e+01,  …, -1.2595e+01,
  -1.8829e+01,  2.6081e+00],
 …,
 [-6.1068e+00, -9.9871e+00, -7.1246e+00,  …, -4.8981e+00,
   3.5623e+00, -1.4631e+00],
 [-2.1628e+00, -6.9973e+00, -7.3790e+00,  …, -1.1450e+00,
   1.0750e+01, -1.4631e+01],
 [-8.2696e-01, -7.9515e+00, -5.0254e+00,  …,  5.5343e+00,
   3.3372e-06, -1.7875e+01]],

[[ 9.1601e+00,  7.8879e+00,  1.0750e+01,  …, -3.5623e+00,
  -3.5623e+00, -1.2532e+01],
 [ 2.7989e+00,  6.9973e-01,  3.6259e+00,  …, -7.1882e+00,
  -4.0712e+00, -1.4885e+01],
  [ 1.0814e+00, -2.1628e+00, -2.5445e-01,  …, -5.9159e+00,
   -6.4884e+00, -1.2850e+01],
  …,
  [-1.4249e+01, -1.5776e+01, -1.7557e+01,  …, -6.8065e+00,
   -1.0432e+01,  5.3434e+00],
  [-1.4376e+01, -1.3549e+01, -1.4122e+01,  …, -1.0178e+01,
   -6.6793e+00,  2.2900e+00],
  [-6.9337e+00, -4.0076e+00, -6.4248e+00,  …, -6.2340e+00,
   -1.9084e-01, -5.7251e+00]],

 [[-1.0496e+01, -1.3740e+01, -1.8575e+01,  …, -1.7302e+01,
   -1.6794e+01, -1.3422e+01],
  [-2.2391e+01, -2.1755e+01, -2.5572e+01,  …, -1.9020e+01,
   -2.3727e+01, -1.9656e+01],
  [-2.2073e+01, -2.2328e+01, -2.5190e+01,  …, -1.9465e+01,
   -1.9211e+01, -2.2773e+01],
  …,
  [-3.1106e+01, -3.2124e+01, -2.8625e+01,  …, -2.5127e+01,
   -2.7798e+01, -3.5623e+01],
  [-2.3664e+01, -2.3409e+01, -2.2391e+01,  …, -2.4363e+01,
   -3.2379e+01, -3.1870e+01],
  [-1.5967e+01, -1.7239e+01, -1.5458e+01,  …, -2.6463e+01,
   -3.9630e+01, -1.9274e+01]]],


…,


[[[ 5.5979e+00, -9.6054e+00, -7.5698e+00,  …, -1.3359e+01,
   -1.2723e-01, -1.8257e+01],
  [-2.1628e+00, -1.8320e+01, -1.5712e+01,  …, -1.9402e+01,
   -1.2913e+01, -1.8638e+01],
  [ 6.9973e-01, -2.0101e+01, -1.2722e+01,  …, -2.4173e+01,
   -1.7112e+01, -2.4681e+01],
  …,
  [ 9.7327e+00, -4.9617e+00, -1.3549e+01,  …,  3.1170e+00,
   -1.9720e+00, -2.4872e+01],
  [ 6.3612e+00, -4.0712e+00, -1.2659e+01,  …, -1.4313e+01,
   -1.0814e+00, -5.8523e+00],
  [ 6.9973e-01, -5.4070e+00, -6.8065e+00,  …, -3.0025e+01,
   -2.9898e+01, -1.8638e+01]],

 [[ 5.4706e+01,  5.0444e+01,  4.6819e+01,  …,  5.8778e+01,
    5.7633e+01,  4.6564e+01],
  [ 5.2098e+01,  4.7645e+01,  4.6310e+01,  …,  5.6169e+01,
    5.2862e+01,  4.3638e+01],
  [ 5.1081e+01,  4.5673e+01,  4.8154e+01,  …,  5.3816e+01,
    4.9490e+01,  4.0012e+01],
 …,
 [ 5.7251e+01,  5.1271e+01,  4.7391e+01,  …,  4.9745e+01,
   4.5737e+01,  3.8613e+01],
 [ 5.6233e+01,  5.0444e+01,  4.6946e+01,  …,  5.1971e+01,
   5.2353e+01,  4.7455e+01],
 [ 5.6615e+01,  5.1526e+01,  4.8854e+01,  …,  3.5241e+01,
   3.7531e+01,  3.8676e+01]],

[[-1.4376e+01, -1.2086e+01, -8.0787e+00,  …, -2.3282e+01,
  -1.0878e+01,  1.8002e+01],
 [-5.5979e+00, -1.5903e+00, -2.2264e+00,  …, -8.2696e+00,
  -6.9974e-01,  1.6921e+01],
 [-2.4809e+00,  1.4631e+00, -3.8803e+00,  …,  5.0890e-01,
   5.0254e+00,  2.0356e+01],
 …,
 [-1.7811e+01, -1.8448e+00,  6.3612e+00,  …,  1.8893e+01,
   1.4758e+01, -2.7989e+00],
 [-1.7875e+01, -1.4631e+00,  9.4782e+00,  …,  6.6157e+00,
   1.1196e+01, -3.2442e+00],
 [-2.6208e+01, -7.6971e+00,  4.8981e+00,  …, -2.0038e+01,
  -9.0965e+00,  1.7175e+00]],


…,


[[-9.4146e+00, -8.6512e+00, -1.0242e+01,  …, -1.5458e+01,
  -2.4363e+01, -1.7112e+01],
 [-1.5076e+01, -1.0814e+01, -1.4058e+01,  …, -2.3409e+01,
  -1.2913e+01, -2.0356e+01],
 [-1.5649e+01, -9.6054e+00, -1.2150e+01,  …, -1.8257e+01,
  -1.6157e+01, -1.9593e+01],
 …,
 [-1.0941e+01, -8.9693e+00, -1.3677e+01,  …, -1.5903e+00,
  -3.8358e+01, -9.4782e+00],
 [-5.7251e+00, -4.8981e+00, -1.3931e+01,  …,  2.5954e+01,
   4.3256e+00, -2.9389e+01],
 [-1.1895e+01, -6.8701e+00, -2.4173e+01,  …, -1.6857e+01,
   3.6259e+00, -1.4822e+01]],

[[-3.3905e+01, -2.0738e+01, -1.9911e+01,  …, -3.2442e+01,
  -5.1399e+01, -3.2760e+01],
 [-1.4949e+01, -3.4351e+00, -8.2696e-01,  …, -1.4758e+01,
  -3.0788e+01, -1.8702e+01],
 [-1.8829e+01,  2.5445e-01,  1.9084e-01,  …, -1.8575e+01,
  -2.1692e+01, -1.7811e+01],
 …,
 [-3.3078e+01, -2.3982e+01, -1.5203e+01,  …, -4.0521e+01,
  -1.0496e+01,  1.4058e+01],
 [-2.9007e+01, -2.2900e+01, -1.0178e+01,  …, -1.5712e+01,
   -2.9834e+01, -4.8345e+00],
  [-3.0089e+01, -3.0407e+01, -1.8575e+01,  …,  2.5445e-01,
    3.6259e+00, -9.0329e+00]],

 [[-2.5190e+01, -2.2137e+01, -1.6984e+01,  …, -3.3396e+01,
   -2.3473e+01, -2.2073e+01],
  [-1.7939e+01, -1.2786e+01, -1.1387e+01,  …, -1.8638e+01,
   -8.2060e+00, -1.2532e+01],
  [-1.3295e+01, -1.5203e+01, -1.5967e+01,  …, -1.0814e+01,
   -1.3231e+01,  4.4528e-01],
  …,
  [-1.5331e+01, -9.6054e+00, -9.9871e+00,  …,  2.2010e+01,
   -8.0787e+00, -2.9198e+01],
  [-9.3510e+00, -6.6793e+00, -1.0750e+01,  …, -2.6590e+01,
    1.0305e+01, -1.7430e+01],
  [-2.2710e+01, -1.6539e+01, -1.6857e+01,  …, -3.7404e+01,
   -3.6513e+01, -8.9057e-01]]],


[[[-1.6476e+01, -1.5585e+01, -1.5394e+01,  …, -1.6094e+01,
   -1.5839e+01, -9.9235e+00],
  [-1.8829e+01, -1.7112e+01, -1.6857e+01,  …, -1.5903e+01,
   -1.6285e+01, -1.1959e+01],
  [-1.8066e+01, -1.7621e+01, -1.6984e+01,  …, -1.7048e+01,
   -1.7112e+01, -1.2722e+01],
  …,
  [-1.6857e+01, -1.4249e+01, -1.3995e+01,  …, -1.4567e+01,
   -1.0178e+01, -1.4504e+01],
  [-1.8384e+01, -1.5521e+01, -1.3804e+01,  …, -2.0165e+01,
   -1.4249e+01, -1.5458e+01],
  [-1.8893e+01, -1.6539e+01, -1.5394e+01,  …, -2.0674e+01,
   -1.9211e+01, -1.6794e+01]],

 [[ 2.6717e+01,  2.8180e+01,  2.8244e+01,  …,  2.8880e+01,
    2.9071e+01,  3.6068e+01],
  [ 2.8435e+01,  3.0152e+01,  3.0534e+01,  …,  3.2188e+01,
    3.1997e+01,  3.8040e+01],
  [ 2.8689e+01,  3.0216e+01,  3.0597e+01,  …,  3.5368e+01,
    3.5114e+01,  4.0330e+01],
  …,
  [ 3.1870e+01,  3.3778e+01,  3.4223e+01,  …,  4.0457e+01,
    4.2620e+01,  4.2556e+01],
  [ 3.2569e+01,  3.4605e+01,  3.4732e+01,  …,  3.9567e+01,
    4.1730e+01,  4.2811e+01],
  [ 4.0775e+01,  4.3129e+01,  4.3320e+01,  …,  3.8613e+01,
    4.0457e+01,  4.1857e+01]],

 [[-7.0609e+00, -6.6793e+00, -7.1246e+00,  …, -5.4070e+00,
  -7.6971e+00, -1.2404e+01],
 [-6.7429e+00, -7.5062e+00, -8.5876e+00,  …, -8.2696e+00,
  -8.4604e+00, -1.2532e+01],
 [-6.4884e+00, -7.3154e+00, -8.9693e+00,  …, -5.9795e+00,
  -5.2798e+00, -1.0941e+01],
 …,
 [-7.0609e+00, -7.8243e+00, -7.8879e+00,  …, -1.0878e+01,
  -1.0560e+01, -1.1196e+01],
 [-8.2060e+00, -8.8421e+00, -9.4782e+00,  …, -1.2214e+01,
  -1.2341e+01, -1.4440e+01],
 [-3.4987e+00, -5.9159e+00, -6.2976e+00,  …, -1.1387e+01,
  -1.1259e+01, -1.0941e+01]],

…,

[[-8.8421e+00, -5.7887e+00, -4.9617e+00,  …, -4.2620e+00,
  -4.5165e+00, -2.9262e+00],
 [-1.8638e+01, -1.3359e+01, -1.3422e+01,  …, -1.2977e+01,
  -1.3613e+01, -8.5876e+00],
 [-1.8575e+01, -1.2659e+01, -1.3613e+01,  …, -1.0369e+01,
  -1.1069e+01, -7.5062e+00],
 …,
 [-1.2722e+01, -1.3104e+01, -1.4694e+01,  …, -1.2214e+01,
  -1.2913e+01, -9.8599e+00],
 [-1.5521e+01, -1.2532e+01, -1.2277e+01,  …, -5.8523e+00,
  -9.2238e+00, -1.5649e+01],
 [-1.1577e+01, -5.5343e+00, -7.1882e+00,  …, -5.1526e+00,
  -7.5062e+00, -1.1895e+01]],

[[ 3.6259e+00,  1.2723e-01,  5.7251e-01,  …,  3.6895e+00,
   3.4987e+00,  5.1526e+00],
 [-3.1806e-01, -5.4070e+00, -4.3256e+00,  …, -4.5801e+00,
  -4.7073e+00, -1.7811e+00],
 [-5.7250e-01, -3.9439e+00, -4.1984e+00,  …, -1.8447e+00,
  -2.1628e+00,  7.6335e-01],
 …,
 [-1.6539e+00, -5.5979e+00, -5.2162e+00,  …, -1.2786e+01,
  -1.2468e+01, -2.7989e+00],
 [-1.9083e-01, -5.1526e+00, -6.6793e+00,  …, -5.2798e+00,
  -5.9159e+00,  1.5903e+00],
 [ 1.0687e+01,  7.9515e+00,  6.8065e+00,  …,  1.2023e+01,
   8.5240e+00,  7.9515e+00]],

[[-8.0151e+00, -1.1895e+01, -1.1832e+01,  …, -1.2468e+01,
  -1.3040e+01, -1.3231e+01],
 [-1.5903e+01, -1.7175e+01, -1.9147e+01,  …, -1.9720e+01,
  -1.9338e+01, -2.0292e+01],
 [-1.4949e+01, -1.8257e+01, -1.8066e+01,  …, -1.6730e+01,
   -1.5712e+01, -1.8129e+01],
  …,
  [-1.9147e+01, -2.0292e+01, -2.0674e+01,  …, -1.8448e+01,
   -2.0292e+01, -2.3028e+01],
  [-1.9402e+01, -1.8002e+01, -1.8320e+01,  …, -1.7048e+01,
   -1.6412e+01, -1.9020e+01],
  [-1.5331e+01, -1.5903e+01, -1.5331e+01,  …, -1.9529e+01,
   -1.6348e+01, -1.4504e+01]]],


[[[-1.7175e+01, -1.0941e+01, -1.3295e+01,  …, -1.3359e+01,
   -1.5776e+01, -2.5890e+01],
  [-9.6054e+00, -1.0687e+01, -1.2468e+01,  …, -1.3359e+00,
   -1.0432e+01, -1.9084e+01],
  [-1.2977e+01, -1.5712e+01, -1.5203e+01,  …,  2.4809e+00,
   -7.0609e+00, -1.9147e+01],
  …,
  [ 6.3612e-01, -2.9262e+00, -1.8448e+00,  …, -2.5445e-01,
   -8.3332e+00, -2.0483e+01],
  [-6.6793e+00, -1.6857e+01, -1.3804e+01,  …, -9.9235e+00,
   -1.1005e+01, -2.0229e+01],
  [-1.1768e+01, -1.0496e+01, -2.8625e+00,  …, -1.0623e+01,
   -1.1005e+01, -1.9211e+01]],

 [[ 4.9935e+01,  5.1526e+01,  5.1462e+01,  …,  4.6946e+01,
    4.7200e+01,  4.6373e+01],
  [ 4.7836e+01,  4.7455e+01,  4.6373e+01,  …,  4.4147e+01,
    4.2493e+01,  4.1793e+01],
  [ 4.6819e+01,  4.6119e+01,  4.5673e+01,  …,  4.6055e+01,
    4.3638e+01,  4.1602e+01],
  …,
  [ 6.1131e+01,  5.9605e+01,  5.9795e+01,  …,  4.8600e+01,
    4.7900e+01,  4.4401e+01],
  [ 6.0813e+01,  6.1258e+01,  6.2658e+01,  …,  4.7391e+01,
    4.7327e+01,  4.4338e+01],
  [ 5.7633e+01,  5.9032e+01,  6.0177e+01,  …,  3.8485e+01,
    3.8676e+01,  3.7022e+01]],

 [[-1.3931e+01, -1.0051e+01, -7.6335e+00,  …, -6.8065e+00,
   -1.1005e+01, -9.7327e+00],
  [-1.3867e+01, -1.2086e+01, -1.2214e+01,  …, -8.3968e+00,
   -1.3677e+01, -1.3295e+01],
  [-1.1514e+01, -1.1768e+01, -1.1895e+01,  …, -6.8701e+00,
   -1.3613e+01, -1.3995e+01],
  …,
  [-8.9693e+00, -8.4604e+00, -1.4440e+01,  …, -1.0242e+01,
   -1.1768e+01, -1.3040e+01],
  [-1.7621e+01, -1.6984e+01, -1.7939e+01,  …, -1.1387e+01,
    -1.1196e+01, -1.3867e+01],
   [-1.3931e+01, -1.2977e+01, -9.6690e+00,  …, -8.0787e+00,
    -9.4146e+00, -1.5140e+01]],

  …,

  [[-1.1832e+01, -1.7811e+01, -1.3995e+01,  …, -2.1628e+01,
    -1.8257e+01, -2.2328e+01],
   [-7.7607e+00, -3.1170e+00, -5.0890e-01,  …, -9.7963e+00,
    -2.0356e+00, -1.0242e+01],
   [-1.1259e+01, -9.8599e+00, -1.0178e+01,  …, -9.5418e+00,
    -4.6437e+00, -1.0305e+01],
   …,
   [ 2.2264e+00,  4.5801e+00,  6.2340e+00,  …, -3.5623e+00,
    -6.4248e+00, -1.0878e+01],
   [-1.1895e+01, -1.1959e+01, -2.1374e+01,  …, -3.5623e+00,
    -5.9795e+00, -9.9235e+00],
   [-1.8638e+01,  3.8167e-01, -5.9795e+00,  …, -3.2442e+00,
    -5.9795e+00, -1.0432e+01]],

  [[-1.1514e+01, -1.6984e+01, -1.7302e+01,  …, -1.4822e+01,
    -1.8384e+01, -1.0114e+01],
   [-1.9084e+00, -5.2798e+00, -3.2442e+00,  …, -6.8065e+00,
    -1.0242e+01, -4.1348e+00],
   [-4.8345e+00,  5.7251e-01,  1.1450e+00,  …, -8.3332e+00,
    -9.0329e+00, -4.5165e+00],
   …,
   [-9.4782e+00, -1.0941e+01, -2.9262e+00,  …, -1.1705e+01,
    -1.1450e+01, -2.9262e+00],
   [-4.6437e+00, -3.8167e-01,  2.7989e+00,  …, -1.1196e+01,
    -1.1959e+01, -2.7353e+00],
   [-8.2696e+00, -1.0941e+01, -2.0992e+01,  …, -9.0329e+00,
    -1.0178e+01, -1.9084e+00]],

  [[-2.6335e+01, -2.1119e+01, -1.8384e+01,  …, -3.1933e+01,
    -2.9007e+01, -2.7353e+01],
   [-2.0928e+01, -1.7748e+01, -1.7811e+01,  …, -2.4300e+01,
    -2.3918e+01, -1.9593e+01],
   [-1.6984e+01, -1.9402e+01, -1.9274e+01,  …, -2.0292e+01,
    -2.5318e+01, -2.0229e+01],
   …,
   [-1.8066e+01, -1.1514e+01, -2.1119e+01,  …, -1.9974e+01,
    -2.3473e+01, -1.8829e+01],
   [-2.2900e+01, -2.4045e+01, -2.6781e+01,  …, -2.0928e+01,
    -2.2455e+01, -1.9147e+01],
   [-2.5827e+01, -2.8625e+01, -1.8066e+01,  …, -2.0483e+01,
    -2.0229e+01, -1.8702e+01]]]], device='cuda:0',
grad_fn=<MulBackward0>)
```

[98]:

```
#### input floating number / weight quantized version

conv_ref = torch.nn.Conv2d(in_channels = 64, out_channels=64, kernel_size = 3,␣
 ↪bias = False)
conv_ref.weight = mod.weight_q

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

```
tensor([[[[-1.1532e+00, -1.1383e+01, -8.9341e+00,  ..., -4.1374e+00,
           -6.3392e+00, -1.8288e+01],
          [-1.5302e+00, -1.1977e+01, -9.7487e+00,  ..., -2.4339e+00,
           -3.5266e+00, -1.4036e+01],
          [-3.7124e+00, -1.0970e+01, -6.7664e+00,  ..., -4.6223e-01,
            1.9162e-01, -9.9417e+00],
          ...,
          [-2.9828e+01, -2.7429e+01, -2.6654e+01,  ..., -2.2617e+01,
           -3.2656e+01, -3.0112e+01],
          [-2.2900e+01, -2.3543e+01, -2.4849e+01,  ..., -1.7114e+01,
           -2.7560e+01, -2.4075e+01],
          [-2.0955e+01, -1.5818e+01, -2.0526e+01,  ..., -1.5482e+01,
           -1.4689e+01, -2.4806e+01]],

         [[ 5.4460e+01,  5.2927e+01,  5.3573e+01,  ...,  4.8487e+01,
            4.6296e+01,  4.1812e+01],
          [ 4.9419e+01,  4.8506e+01,  4.9826e+01,  ...,  4.3696e+01,
            4.1687e+01,  3.8765e+01],
          [ 4.6172e+01,  4.6188e+01,  4.8053e+01,  ...,  4.4726e+01,
            4.3018e+01,  3.9337e+01],
          ...,
          [ 1.8488e+01,  2.1270e+01,  2.4814e+01,  ...,  2.0849e+01,
            2.3595e+01,  3.6509e+01],
          [ 2.1689e+01,  2.5151e+01,  2.5828e+01,  ...,  2.3214e+01,
            2.5076e+01,  3.4072e+01],
          [ 3.8680e+01,  4.2587e+01,  4.0627e+01,  ...,  4.0558e+01,
            4.2953e+01,  4.3332e+01]],

         [[-2.3009e+01, -1.7066e+01, -1.6490e+01,  ..., -2.1039e+01,
           -1.5653e+01, -6.4787e+00],
          [-1.2231e+01, -8.6141e+00, -9.2969e+00,  ..., -1.1914e+01,
           -6.6982e+00, -1.7504e+00],
          [-1.7285e+01, -1.3111e+01, -1.2255e+01,  ..., -1.0953e+01,
           -6.8802e+00, -2.8655e+00],
          ...,
          [-7.6587e+00, -8.0670e+00, -1.1911e+01,  ..., -1.9087e+01,
           -1.9084e+01, -2.6995e+01],
          [-6.4928e+00, -8.7858e+00, -1.1183e+01,  ..., -8.8336e+00,
           -1.4774e+01, -2.1759e+01],
 [-4.6505e+00, -8.2087e+00, -8.2431e+00,  …, -3.9656e+00,
  -1.2255e+01, -1.6246e+01]],

…,

[[-9.6424e+00, -7.4006e+00, -6.3023e+00,  …, -1.2116e+01,
  -1.4307e+01, -1.9656e+01],
 [-9.6805e+00, -8.5806e+00, -8.2823e+00,  …, -1.3656e+01,
  -1.1233e+01, -1.8912e+01],
 [-1.7619e+01, -2.0028e+01, -1.1105e+01,  …, -1.4806e+01,
  -1.2874e+01, -1.4234e+01],
 …,
 [-1.5329e+01, -9.4759e+00, -1.5131e+01,  …, -9.1591e+00,
  -1.7010e+01, -2.5982e+01],
 [-1.8753e+01, -8.6944e+00, -8.9221e+00,  …, -7.0152e+00,
  -9.1050e+00, -3.4675e+01],
 [-1.2081e+01, -8.2478e+00,  4.5750e+00,  …,  2.2418e+00,
  -5.2299e+00, -9.2272e+00]],

[[-2.7771e+01, -2.1281e+01, -2.0585e+01,  …, -3.1422e+01,
  -2.8783e+01, -2.3643e+01],
 [-1.6388e+01, -7.3199e+00, -5.2890e+00,  …, -1.5411e+01,
  -1.3000e+01, -1.0657e+01],
 [-3.0019e+01, -1.8618e+01, -1.6453e+01,  …, -1.6380e+01,
  -1.2056e+01, -1.0766e+01],
 …,
 [ 8.5207e+00,  1.7030e+00,  3.3654e+00,  …,  4.7938e+00,
   1.0109e+01,  2.0161e+01],
 [ 5.8340e+00,  2.2285e+00,  5.7730e-01,  …, -2.4045e+00,
   9.9913e-01,  6.1446e+00],
 [ 2.5095e+01,  1.7442e+01,  1.8733e+01,  …,  1.8556e+01,
   2.4919e+01,  2.1521e+01]],

[[-2.6774e+01, -2.4700e+01, -2.4973e+01,  …, -2.6824e+01,
  -2.4495e+01, -2.3209e+01],
 [-1.8035e+01, -1.9997e+01, -1.9878e+01,  …, -1.4765e+01,
  -1.3544e+01, -1.3683e+01],
 [-1.8369e+01, -2.2634e+01, -2.2531e+01,  …, -8.9709e+00,
  -1.2887e+01, -1.5720e+01],
 …,
 [-2.6275e+01, -1.9800e+01, -1.8166e+01,  …, -3.5836e+01,
  -3.6879e+01, -4.1227e+01],
 [-2.4117e+01, -2.9955e+01, -1.9769e+01,  …, -3.0290e+01,
  -2.6149e+01, -1.9918e+01],
 [-2.5457e+01, -3.3674e+01, -3.1294e+01,  …, -2.9766e+01,
  -3.8810e+01, -1.3125e+01]]],
[[[-3.6736e+00, -9.6517e+00, -9.4765e+00,  …, -9.4075e+00,
   -9.4012e+00,  4.6477e-01],
  [-8.7684e+00, -1.3144e+01, -1.3108e+01,  …, -1.2875e+01,
   -1.2757e+01, -4.1404e+00],
  [-8.2009e+00, -1.2683e+01, -1.2742e+01,  …, -1.3273e+01,
   -1.3270e+01, -4.2094e+00],
  …,
  [-6.6386e+00, -1.0370e+00, -1.3099e+01,  …, -1.7351e+01,
   -1.8291e+01, -1.0130e+01],
  [-6.4312e+00, -6.4650e+00, -1.0574e+01,  …, -1.5375e+01,
   -1.4758e+01, -9.2955e+00],
  [-6.1165e+00, -1.1640e+01, -9.4551e+00,  …, -1.5554e+01,
   -1.5314e+01, -1.2390e+01]],

 [[ 5.0381e+01,  4.8219e+01,  4.8265e+01,  …,  4.8528e+01,
    4.8528e+01,  5.6138e+01],
  [ 4.9622e+01,  4.8466e+01,  4.8436e+01,  …,  4.8460e+01,
    4.8463e+01,  5.6015e+01],
  [ 4.9912e+01,  4.8843e+01,  4.8822e+01,  …,  4.8394e+01,
    4.8330e+01,  5.6019e+01],
  …,
  [ 3.9806e+01,  4.3089e+01,  4.1321e+01,  …,  3.1372e+01,
    3.2657e+01,  4.1361e+01],
  [ 3.7752e+01,  3.7408e+01,  3.5885e+01,  …,  3.2859e+01,
    3.4478e+01,  4.2218e+01],
  [ 2.9530e+01,  2.7629e+01,  2.7363e+01,  …,  4.4100e+01,
    4.7379e+01,  5.4435e+01]],

 [[-1.3164e+01, -1.4369e+01, -1.4195e+01,  …, -1.4462e+01,
   -1.4453e+01, -1.8608e+01],
  [-1.1295e+01, -1.3166e+01, -1.3049e+01,  …, -1.3155e+01,
   -1.3145e+01, -1.5765e+01],
  [-1.0909e+01, -1.2703e+01, -1.2621e+01,  …, -1.3029e+01,
   -1.3152e+01, -1.5619e+01],
  …,
  [-9.8155e+00, -1.5803e+01, -1.9222e+01,  …, -9.0824e+00,
   -9.7948e+00, -1.1762e+01],
  [-1.6072e+01, -1.4642e+01, -1.8040e+01,  …, -7.8966e+00,
   -8.9216e+00, -1.1345e+01],
  [-1.3650e+01, -1.0800e+01, -1.1558e+01,  …, -6.2025e+00,
   -6.5836e+00, -8.4847e+00]],


 …,


 [[-1.2290e+01, -6.1051e+00, -6.2835e+00,  …, -6.4561e+00,
   -5.9295e+00, -2.9073e+00],
  [-1.9186e+01, -1.2038e+01, -1.2087e+01,  …, -1.1992e+01,
   -1.1408e+01, -4.3972e+00],
  [-1.8958e+01, -1.2087e+01, -1.2048e+01,  …, -1.3157e+01,
   -1.2332e+01, -5.0023e+00],
  …,
  [ 1.5203e+00, -1.8615e+01, -4.4327e-01,  …, -1.7544e+01,
   -1.6220e+01, -7.5092e+00],
  [-8.9746e+00, -1.3005e+01, -6.5522e+00,  …, -1.4122e+01,
   -1.5366e+01, -6.3654e+00],
  [-1.6808e+01, -7.6401e+00, -7.5754e+00,  …, -9.1872e+00,
   -5.6570e+00, -8.5099e-01]],

 [[-4.8573e+00, -6.6689e+00, -6.7665e+00,  …, -6.7213e+00,
   -6.8918e+00, -1.0884e+01],
  [-4.8771e+00, -7.3553e+00, -7.3852e+00,  …, -7.6063e+00,
   -7.8253e+00, -1.2995e+01],
  [-5.2196e+00, -7.7142e+00, -7.6464e+00,  …, -7.5071e+00,
   -7.6102e+00, -1.2985e+01],
  …,
  [-6.8889e-01, -7.7610e+00, -8.6614e+00,  …, -2.8976e+00,
   -1.3919e+00, -6.4525e+00],
  [ 5.2269e+00, -1.0229e+01, -8.7030e+00,  …, -4.0403e+00,
   -5.6254e+00, -8.4144e+00],
  [-1.8171e+00, -1.0266e+01, -1.1639e+01,  …,  3.1349e+00,
    1.5888e+00, -2.0052e+00]],

 [[-8.5481e+00, -1.3497e+01, -1.3215e+01,  …, -1.3347e+01,
   -1.3288e+01, -1.3727e+01],
  [-1.2163e+01, -1.7644e+01, -1.7582e+01,  …, -1.7613e+01,
   -1.7602e+01, -1.8465e+01],
  [-1.1940e+01, -1.7271e+01, -1.7371e+01,  …, -1.7459e+01,
   -1.7641e+01, -1.8866e+01],
  …,
  [-1.8765e+01, -1.8302e+01, -2.5566e+01,  …, -1.2675e+01,
   -1.5929e+01, -1.6029e+01],
  [-2.5366e+01, -9.7643e+00, -2.0592e+01,  …, -1.1877e+01,
   -1.2809e+01, -1.5467e+01],
  [-1.1917e+01, -5.8761e+00, -8.1199e+00,  …, -1.2604e+01,
   -1.1899e+01, -1.3620e+01]]],


[[[-1.3237e+01, -1.7645e+01, -1.4573e+01,  …, -1.2481e+01,
   -9.4467e+00,  1.3317e+00],
  [-1.2718e+01, -1.9258e+01, -1.4440e+01,  …, -1.4852e+01,
   -1.0851e+01, -3.4800e+00],
  [-1.2370e+01, -2.0355e+01, -1.4939e+01,  …, -1.6184e+01,
   -9.8371e+00, -5.9391e+00],
  …,
  [-1.8859e+01, -1.3347e+01, -1.3964e+01,  …, -7.4969e+00,
   -5.3820e+00, -2.4706e+01],
  [-1.6845e+01, -1.1494e+01, -1.2352e+01,  …, -4.4443e+00,
   -8.0630e+00, -2.2680e+01],
  [-1.8680e+01, -1.3445e+01, -1.3438e+01,  …, -8.6828e+00,
   -8.1996e+00, -2.4191e+01]],

 [[ 3.5706e+01,  3.4226e+01,  3.8829e+01,  …,  4.8712e+01,
    5.0682e+01,  5.9567e+01],
  [ 3.8451e+01,  3.8413e+01,  4.3595e+01,  …,  5.0125e+01,
    5.2419e+01,  5.9402e+01],
  [ 3.9798e+01,  3.9478e+01,  4.5014e+01,  …,  5.1521e+01,
    5.3657e+01,  5.9310e+01],
  …,
  [ 4.2882e+01,  4.3867e+01,  4.1496e+01,  …,  4.4480e+01,
    4.7710e+01,  4.9544e+01],
  [ 3.7766e+01,  3.8168e+01,  3.6150e+01,  …,  4.7781e+01,
    4.8787e+01,  5.0628e+01],
  [ 3.2797e+01,  3.3130e+01,  3.2113e+01,  …,  3.9915e+01,
    4.1073e+01,  4.2150e+01]],

 [[-4.4202e+00, -1.7989e+00, -1.2825e+01,  …, -1.1785e+01,
   -1.7665e+01, -2.0128e+01],
  [-9.5657e+00, -5.1810e+00, -1.4505e+01,  …, -9.3523e+00,
   -1.7640e+01, -1.8793e+01],
  [-1.0515e+01, -5.5206e+00, -1.5533e+01,  …, -9.0945e+00,
   -1.7769e+01, -1.9321e+01],
  …,
  [-1.6995e+01, -1.6411e+01, -1.4857e+01,  …, -1.6707e+01,
   -1.5890e+01, -1.0598e+01],
  [-1.6257e+01, -1.5653e+01, -1.3989e+01,  …, -1.8532e+01,
   -2.0204e+01, -1.2123e+01],
  [-1.2288e+01, -1.0968e+01, -9.3386e+00,  …, -1.6911e+01,
   -2.0349e+01, -9.1426e+00]],


 …,


 [[-1.8981e+00, -8.8793e+00, -1.3286e+01,  …, -8.5992e+00,
   -7.2366e+00, -7.0188e-01],
  [-8.6555e+00, -1.4431e+01, -2.1303e+01,  …, -1.4701e+01,
   -1.3785e+01,  2.9444e+00],
  [-1.3226e+01, -1.6126e+01, -1.9428e+01,  …, -1.2935e+01,
   -1.9811e+01,  3.9657e+00],
  …,
  [-5.7124e+00, -9.7837e+00, -7.8283e+00,  …, -4.4470e+00,
    3.3931e+00, -2.6182e+00],
  [-3.2425e+00, -8.0179e+00, -7.4145e+00,  …, -2.5352e+00,
    1.1452e+01, -1.6159e+01],
  [-7.3584e-01, -7.7589e+00, -5.0201e+00,  …,  5.5419e+00,
   -3.4123e-01, -2.0319e+01]],
  [[ 8.7446e+00,  6.8910e+00,  1.0334e+01,  …, -7.1263e-01,
    -2.4987e+00, -1.1642e+01],
   [ 2.4704e+00,  3.7631e-01,  2.5696e+00,  …, -6.4865e+00,
    -4.1455e+00, -1.6456e+01],
   [ 1.9878e+00, -2.9775e-02,  8.9450e-01,  …, -5.4287e+00,
    -5.9349e+00, -1.4516e+01],
   …,
   [-1.3893e+01, -1.5727e+01, -1.7034e+01,  …, -6.5997e+00,
    -1.0870e+01,  5.5150e+00],
   [-1.5348e+01, -1.5039e+01, -1.5752e+01,  …, -1.0064e+01,
    -7.8087e+00,  2.5018e+00],
   [-6.3355e+00, -4.8055e+00, -5.9314e+00,  …, -6.5545e+00,
    -1.1633e+00, -5.3172e+00]],

  [[-1.1068e+01, -1.2369e+01, -1.8130e+01,  …, -1.6397e+01,
    -1.6436e+01, -1.3308e+01],
   [-2.1377e+01, -2.1065e+01, -2.4402e+01,  …, -1.8288e+01,
    -2.3438e+01, -1.7639e+01],
   [-2.1381e+01, -2.2773e+01, -2.4638e+01,  …, -1.8943e+01,
    -1.9997e+01, -2.1647e+01],
   …,
   [-3.1036e+01, -3.1997e+01, -2.9062e+01,  …, -2.4346e+01,
    -2.6247e+01, -3.2916e+01],
   [-2.1688e+01, -2.2952e+01, -2.1543e+01,  …, -2.5100e+01,
    -3.3847e+01, -3.0669e+01],
   [-1.5444e+01, -1.5817e+01, -1.5428e+01,  …, -2.6550e+01,
    -3.9891e+01, -1.8498e+01]]],


 …,


 [[[ 4.8664e+00, -9.8653e+00, -8.8130e+00,  …, -1.2692e+01,
    -1.0704e-01, -1.7775e+01],
   [-2.4691e+00, -1.7622e+01, -1.6134e+01,  …, -1.8710e+01,
    -1.3566e+01, -1.8660e+01],
   [ 1.2171e+00, -1.9722e+01, -1.2065e+01,  …, -2.3369e+01,
    -1.7763e+01, -2.4732e+01],
   …,
   [ 9.4509e+00, -4.5199e+00, -1.3901e+01,  …,  1.9230e+00,
    -2.4083e+00, -2.9010e+01],
   [ 5.9468e+00, -3.7936e+00, -1.2649e+01,  …, -1.8549e+01,
    -5.7472e-01, -4.9222e+00],
   [ 5.3437e-01, -5.4659e+00, -7.2604e+00,  …, -3.0797e+01,
    -3.2454e+01, -1.5254e+01]],

  [[ 5.5482e+01,  5.1011e+01,  4.7142e+01,  …,  5.9603e+01,
    5.7647e+01,  4.6325e+01],
  [ 5.2027e+01,  4.7747e+01,  4.5814e+01,  …,  5.6569e+01,
    5.2633e+01,  4.3188e+01],
  [ 5.1010e+01,  4.5644e+01,  4.7596e+01,  …,  5.4324e+01,
    4.9439e+01,  4.0104e+01],
  …,
  [ 5.7371e+01,  5.1331e+01,  4.7039e+01,  …,  5.2385e+01,
    4.7372e+01,  3.7743e+01],
  [ 5.6809e+01,  5.1093e+01,  4.7066e+01,  …,  5.2714e+01,
    5.2323e+01,  4.7616e+01],
  [ 5.6385e+01,  5.1663e+01,  4.8749e+01,  …,  3.7707e+01,
    3.7501e+01,  3.8903e+01]],

 [[-1.5166e+01, -1.2766e+01, -8.0497e+00,  …, -2.4526e+01,
   -1.0691e+01,  1.8208e+01],
  [-6.0945e+00, -2.2388e+00, -2.5632e+00,  …, -9.1735e+00,
   -6.9553e-01,  1.7323e+01],
  [-2.8579e+00,  1.6368e+00, -3.8975e+00,  …,  3.9918e-01,
    4.1377e+00,  1.9164e+01],
  …,
  [-1.7594e+01, -2.5420e+00,  7.3334e+00,  …,  2.6387e+01,
    1.9439e+01, -3.6104e+00],
  [-1.8331e+01, -1.2401e+00,  9.2955e+00,  …,  1.2263e+01,
    1.9353e+01, -2.0657e+00],
  [-2.7219e+01, -7.1454e+00,  4.6500e+00,  …, -1.8404e+01,
   -4.7833e+00,  2.3038e+00]],


 …,


 [[-9.0639e+00, -8.8495e+00, -1.0999e+01,  …, -1.5891e+01,
   -2.4261e+01, -1.7938e+01],
  [-1.4741e+01, -1.2311e+01, -1.4513e+01,  …, -2.3650e+01,
   -1.2198e+01, -2.0560e+01],
  [-1.5581e+01, -9.8736e+00, -1.2411e+01,  …, -1.7400e+01,
   -1.6236e+01, -2.1073e+01],
  …,
  [-9.7221e+00, -1.0184e+01, -1.2905e+01,  …,  4.8216e-01,
   -4.9231e+01, -7.9018e+00],
  [-5.1591e+00, -5.8879e+00, -1.4295e+01,  …,  3.1735e+01,
    3.0137e+00, -3.4161e+01],
  [-1.1979e+01, -6.9581e+00, -2.2139e+01,  …, -2.2943e+01,
    5.4093e+00, -1.7782e+01]],

 [[-3.3669e+01, -2.0720e+01, -1.9508e+01,  …, -3.4212e+01,
   -5.4048e+01, -3.4741e+01],
  [-1.6038e+01, -3.6857e+00, -1.2587e+00,  …, -1.4956e+01,
   -3.1271e+01, -1.9693e+01],
  [-1.9701e+01, -6.1451e-01, -8.5138e-02,  …, -1.9113e+01,
    -2.1459e+01, -1.7000e+01],
   …,
   [-3.4272e+01, -2.3932e+01, -1.5103e+01,  …, -4.8462e+01,
    -1.0251e+01,  1.3854e+01],
   [-2.9014e+01, -2.3503e+01, -9.5720e+00,  …, -1.3685e+01,
    -3.3494e+01, -1.6280e+00],
   [-2.9839e+01, -3.0534e+01, -1.8832e+01,  …,  4.5440e-01,
     6.8027e+00, -8.7637e+00]],

  [[-2.5528e+01, -2.2801e+01, -1.7929e+01,  …, -3.5863e+01,
    -2.2475e+01, -2.1563e+01],
   [-1.8211e+01, -1.3732e+01, -1.1558e+01,  …, -1.9134e+01,
    -8.2867e+00, -1.2544e+01],
   [-1.4367e+01, -1.3751e+01, -1.5926e+01,  …, -1.0355e+01,
    -1.3510e+01, -3.7085e-01],
   …,
   [-1.3964e+01, -1.0756e+01, -9.6281e+00,  …,  2.8233e+01,
    -1.0687e+01, -2.9240e+01],
   [-1.0585e+01, -6.5128e+00, -1.0723e+01,  …, -3.4521e+01,
     1.6074e+01, -2.0449e+01],
   [-2.3254e+01, -1.6263e+01, -1.6864e+01,  …, -3.5616e+01,
    -3.7837e+01,  5.8673e-02]]],


 [[[-1.6570e+01, -1.5610e+01, -1.5557e+01,  …, -1.5905e+01,
    -1.5423e+01, -9.7758e+00],
   [-1.8186e+01, -1.6733e+01, -1.7079e+01,  …, -1.5704e+01,
    -1.5749e+01, -1.1391e+01],
   [-1.8049e+01, -1.7611e+01, -1.7214e+01,  …, -1.6127e+01,
    -1.6328e+01, -1.1606e+01],
   …,
   [-1.6946e+01, -1.4462e+01, -1.4942e+01,  …, -1.3642e+01,
    -1.0624e+01, -1.5290e+01],
   [-1.8430e+01, -1.5807e+01, -1.5726e+01,  …, -1.9163e+01,
    -1.4187e+01, -1.5390e+01],
   [-1.8462e+01, -1.6958e+01, -1.6264e+01,  …, -1.9452e+01,
    -1.8837e+01, -1.6801e+01]],

  [[ 2.7900e+01,  2.8586e+01,  2.8560e+01,  …,  2.9335e+01,
     2.9556e+01,  3.6138e+01],
   [ 2.9330e+01,  3.0306e+01,  3.0249e+01,  …,  3.2746e+01,
     3.2753e+01,  3.8644e+01],
   [ 2.9799e+01,  3.0646e+01,  3.0821e+01,  …,  3.5756e+01,
     3.5763e+01,  4.1173e+01],
   …,
   [ 3.1518e+01,  3.2901e+01,  3.2884e+01,  …,  4.1110e+01,
     4.2291e+01,  4.2138e+01],
   [ 3.2023e+01,  3.3331e+01,  3.3353e+01,  …,  3.9863e+01,
    4.1471e+01,  4.2753e+01],
  [ 4.0617e+01,  4.2269e+01,  4.2445e+01,  …,  3.8898e+01,
    4.0193e+01,  4.1817e+01]],

 [[-6.9128e+00, -8.4229e+00, -8.3436e+00,  …, -5.7765e+00,
   -8.4163e+00, -1.1945e+01],
  [-6.4817e+00, -8.1684e+00, -8.1995e+00,  …, -6.5891e+00,
   -8.0824e+00, -1.1487e+01],
  [-5.8093e+00, -7.4298e+00, -7.7454e+00,  …, -5.5931e+00,
   -5.9525e+00, -1.0686e+01],
  …,
  [-6.7900e+00, -7.8891e+00, -7.7685e+00,  …, -1.0973e+01,
   -9.7106e+00, -1.0011e+01],
  [-7.1131e+00, -8.0512e+00, -7.8915e+00,  …, -1.1193e+01,
   -1.1456e+01, -1.2990e+01],
  [-4.1505e+00, -5.5851e+00, -5.6097e+00,  …, -1.1277e+01,
   -1.1870e+01, -1.1447e+01]],


 …,


 [[-8.9605e+00, -4.9582e+00, -4.6190e+00,  …, -3.9830e+00,
   -4.1124e+00, -2.3694e+00],
  [-1.7876e+01, -1.3303e+01, -1.2980e+01,  …, -1.2169e+01,
   -1.2838e+01, -7.7129e+00],
  [-1.7581e+01, -1.2547e+01, -1.3918e+01,  …, -1.0820e+01,
   -1.1233e+01, -7.0429e+00],
  …,
  [-1.3178e+01, -1.2378e+01, -1.2465e+01,  …, -1.2465e+01,
   -1.2276e+01, -9.8019e+00],
  [-1.5147e+01, -1.2120e+01, -1.1723e+01,  …, -6.2584e+00,
   -8.1313e+00, -1.4828e+01],
  [-1.1232e+01, -6.8211e+00, -8.3922e+00,  …, -5.3805e+00,
   -7.8726e+00, -1.2850e+01]],

 [[ 4.4257e+00,  4.3230e-01, -1.6483e-03,  …,  3.3826e+00,
    3.5831e+00,  5.2870e+00],
  [ 1.2463e-01, -5.7575e+00, -4.9302e+00,  …, -4.5204e+00,
   -4.4913e+00, -1.9033e+00],
  [ 9.8620e-01, -4.1616e+00, -3.6813e+00,  …, -1.5205e+00,
   -1.7089e+00,  1.1667e+00],
  …,
  [-1.2083e+00, -5.5590e+00, -5.7039e+00,  …, -1.3854e+01,
   -1.2951e+01, -2.8415e+00],
  [ 4.2960e-01, -5.2092e+00, -4.7671e+00,  …, -6.3888e+00,
   -6.7156e+00,  1.5674e+00],
  [ 1.0691e+01,  7.3953e+00,  7.6648e+00,  …,  1.2149e+01,
    9.6030e+00,  9.4113e+00]],
  [[-8.1731e+00, -1.1768e+01, -1.1019e+01,  …, -1.2754e+01,
    -1.2689e+01, -1.3322e+01],
   [-1.4631e+01, -1.6743e+01, -1.7423e+01,  …, -1.8962e+01,
    -1.8821e+01, -1.9745e+01],
   [-1.4567e+01, -1.6826e+01, -1.7306e+01,  …, -1.8298e+01,
    -1.7865e+01, -1.9665e+01],
   …,
   [-1.8992e+01, -2.0224e+01, -1.9568e+01,  …, -1.7297e+01,
    -1.9414e+01, -2.1615e+01],
   [-1.8129e+01, -1.7586e+01, -1.8417e+01,  …, -1.5513e+01,
    -1.5382e+01, -1.9112e+01],
   [-1.6393e+01, -1.6840e+01, -1.6844e+01,  …, -1.8961e+01,
    -1.6514e+01, -1.5389e+01]]],


 [[[-1.7181e+01, -1.1495e+01, -1.3361e+01,  …, -1.3458e+01,
    -1.8588e+01, -3.7030e+01],
   [-9.2418e+00, -1.0923e+01, -1.2669e+01,  …, -1.5412e+00,
    -1.0133e+01, -2.8257e+01],
   [-1.2556e+01, -1.5635e+01, -1.4876e+01,  …,  2.3824e+00,
    -6.8357e+00, -2.8375e+01],
   …,
   [ 8.1845e-01, -2.7660e+00, -2.1820e+00,  …, -7.9468e-01,
    -7.6689e+00, -2.8525e+01],
   [-7.4144e+00, -1.6962e+01, -1.4349e+01,  …, -1.0820e+01,
    -9.8113e+00, -2.8694e+01],
   [-1.1930e+01, -1.0981e+01, -2.3907e+00,  …, -1.0772e+01,
    -1.0483e+01, -2.8476e+01]],

  [[ 4.9964e+01,  5.1719e+01,  5.1692e+01,  …,  4.8670e+01,
     4.9329e+01,  5.0675e+01],
   [ 4.8156e+01,  4.7678e+01,  4.7261e+01,  …,  4.3559e+01,
     4.2499e+01,  4.4824e+01],
   [ 4.7485e+01,  4.6063e+01,  4.5469e+01,  …,  4.5105e+01,
     4.3242e+01,  4.4564e+01],
   …,
   [ 6.0096e+01,  5.8697e+01,  5.8836e+01,  …,  4.9064e+01,
     4.7421e+01,  4.6826e+01],
   [ 6.0317e+01,  6.1128e+01,  6.2864e+01,  …,  4.8676e+01,
     4.7547e+01,  4.6974e+01],
   [ 5.7295e+01,  5.9092e+01,  6.0623e+01,  …,  3.9548e+01,
     3.9453e+01,  4.0254e+01]],

  [[-1.2198e+01, -8.5129e+00, -6.7295e+00,  …, -6.9108e+00,
    -1.0281e+01, -2.5811e+00],
   [-1.2322e+01, -1.1374e+01, -1.2170e+01,  …, -8.5981e+00,
    -1.4290e+01, -7.0805e+00],
   [-1.2023e+01, -1.3028e+01, -1.2736e+01,  …, -5.6286e+00,
   -1.4252e+01, -7.0322e+00],
  …,
  [-8.8957e+00, -8.3397e+00, -1.5132e+01,  …, -1.1180e+01,
   -1.3260e+01, -6.6273e+00],
  [-1.6498e+01, -1.5570e+01, -1.6762e+01,  …, -1.1539e+01,
   -1.3173e+01, -6.6628e+00],
  [-1.3703e+01, -1.2088e+01, -9.8613e+00,  …, -8.1807e+00,
   -1.1148e+01, -6.4234e+00]],

 …,

 [[-1.1634e+01, -1.7947e+01, -1.3998e+01,  …, -2.4075e+01,
   -2.2304e+01, -2.9151e+01],
  [-8.0444e+00, -2.4382e+00, -3.0666e-01,  …, -1.0596e+01,
   -2.2825e+00, -1.2123e+01],
  [-1.0830e+01, -9.2789e+00, -9.8328e+00,  …, -9.3253e+00,
   -4.4626e+00, -1.2524e+01],
  …,
  [ 2.0886e+00,  4.0671e+00,  5.0235e+00,  …, -4.8568e+00,
   -6.2220e+00, -1.2966e+01],
  [-1.2799e+01, -1.2620e+01, -2.1660e+01,  …, -3.6377e+00,
   -5.4238e+00, -1.2353e+01],
  [-1.9191e+01,  6.6668e-01, -5.8043e+00,  …, -3.6211e+00,
   -5.4488e+00, -1.3307e+01]],

 [[-1.2429e+01, -1.6476e+01, -1.7366e+01,  …, -1.4052e+01,
   -1.6872e+01, -6.9648e+00],
  [-2.6652e+00, -3.7424e+00, -2.0737e+00,  …, -6.8320e+00,
   -9.0062e+00, -3.4612e+00],
  [-4.7795e+00,  3.4263e-01, -6.2372e-02,  …, -8.2427e+00,
   -8.9340e+00, -3.2455e+00],
  …,
  [-9.5807e+00, -1.1098e+01, -3.8983e+00,  …, -1.1881e+01,
   -1.0687e+01, -3.8888e+00],
  [-3.0771e+00,  8.0219e-01,  3.6545e+00,  …, -8.7214e+00,
   -1.0718e+01, -4.0472e+00],
  [-8.3223e+00, -1.2238e+01, -2.0716e+01,  …, -7.8050e+00,
   -9.5916e+00, -3.0118e+00]],

 [[-2.6290e+01, -1.9325e+01, -1.7974e+01,  …, -3.3612e+01,
   -3.1098e+01, -2.6688e+01],
  [-1.9948e+01, -1.8155e+01, -1.8965e+01,  …, -2.2602e+01,
   -2.4519e+01, -1.7805e+01],
  [-1.7397e+01, -2.1173e+01, -2.0372e+01,  …, -1.8445e+01,
   -2.4088e+01, -1.7721e+01],
  …,
  [-1.7656e+01, -1.1209e+01, -2.0453e+01,  …, -1.8953e+01,
   -2.2490e+01, -1.6888e+01],
           [-2.2176e+01, -2.2424e+01, -2.6017e+01,  …, -2.1340e+01,
            -2.2529e+01, -1.7155e+01],
           [-2.5370e+01, -2.7767e+01, -1.8655e+01,  …, -2.1990e+01,
            -2.1100e+01, -1.6523e+01]]]], device='cuda:0',
        grad_fn=<ConvolutionBackward0>)
```

[99]:
```
difference = abs( output_ref - output_recovered )
print(difference.mean())  ## It should be small, e.g.,2.3 in my trainned model
```

```
tensor(0.9827, device='cuda:0', grad_fn=<MeanBackward0>)
```

[100]:
```
#### input floating number / weight floating number version

conv_ref = torch.nn.Conv2d(in_channels = 64, out_channels=64, kernel_size = 3,␣
 ↪bias = False)
weight = mod.weight
mean = weight.data.mean()
std = weight.data.std()
conv_ref.weight = torch.nn.parameter.Parameter(weight.add(-mean).div(std))

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

```
tensor([[[[-8.4574e-01, -1.1445e+01, -9.0954e+00,  …, -4.2734e+00,
           -6.4664e+00, -1.8211e+01],
          [-1.5285e+00, -1.2299e+01, -1.0206e+01,  …, -2.8503e+00,
           -3.9378e+00, -1.4211e+01],
          [-3.7134e+00, -1.1298e+01, -7.2323e+00,  …, -8.3200e-01,
           -2.3559e-01, -1.0102e+01],
          …,
          [-2.9319e+01, -2.6865e+01, -2.6056e+01,  …, -2.2011e+01,
           -3.1824e+01, -2.8804e+01],
          [-2.2433e+01, -2.2940e+01, -2.4392e+01,  …, -1.6752e+01,
           -2.6902e+01, -2.3336e+01],
          [-2.0725e+01, -1.5458e+01, -2.0318e+01,  …, -1.5555e+01,
           -1.4081e+01, -2.4665e+01]],

         [[ 5.4835e+01,  5.3816e+01,  5.4462e+01,  …,  4.9114e+01,
            4.6829e+01,  4.1362e+01],
          [ 4.9281e+01,  4.8989e+01,  5.0309e+01,  …,  4.3573e+01,
            4.1664e+01,  3.8076e+01],
          [ 4.6199e+01,  4.6562e+01,  4.8389e+01,  …,  4.4428e+01,
            4.2862e+01,  3.8709e+01],
          …,
          [ 1.9006e+01,  2.1750e+01,  2.5075e+01,  …,  2.2496e+01,
            2.3953e+01,  3.6592e+01],
          [ 2.2287e+01,  2.5727e+01,  2.6531e+01,  …,  2.4332e+01,
            2.5401e+01,  3.3801e+01],
          [ 3.8846e+01,  4.2817e+01,  4.1112e+01,  …,  4.0972e+01,
   4.2999e+01,  4.2743e+01]],

[[-2.2973e+01, -1.7116e+01, -1.6400e+01,  …, -2.1263e+01,
  -1.5816e+01, -5.9838e+00],
 [-1.1952e+01, -8.4066e+00, -8.8300e+00,  …, -1.1728e+01,
  -6.4373e+00, -8.7303e-01],
 [-1.7061e+01, -1.2962e+01, -1.1897e+01,  …, -1.0848e+01,
  -6.7038e+00, -2.0589e+00],
 …,
 [-8.8360e+00, -9.5014e+00, -1.3353e+01,  …, -2.0319e+01,
  -1.9589e+01, -2.8507e+01],
 [-7.4971e+00, -1.0209e+01, -1.2602e+01,  …, -9.9855e+00,
  -1.5740e+01, -2.3026e+01],
 [-4.9516e+00, -8.9513e+00, -9.0153e+00,  …, -4.2476e+00,
  -1.3157e+01, -1.6525e+01]],


…,


[[-9.5900e+00, -7.6124e+00, -6.4324e+00,  …, -1.2341e+01,
  -1.4466e+01, -1.9827e+01],
 [-1.0229e+01, -9.3336e+00, -8.9305e+00,  …, -1.4396e+01,
  -1.1946e+01, -1.9631e+01],
 [-1.8363e+01, -2.0963e+01, -1.1847e+01,  …, -1.5636e+01,
  -1.3597e+01, -1.4979e+01],
 …,
 [-1.5040e+01, -9.2140e+00, -1.4974e+01,  …, -8.8476e+00,
  -1.6285e+01, -2.5150e+01],
 [-1.8454e+01, -8.2558e+00, -8.6580e+00,  …, -6.7691e+00,
  -8.5354e+00, -3.4203e+01],
 [-1.1604e+01, -7.5255e+00,  5.1243e+00,  …,  2.9097e+00,
  -4.6086e+00, -8.7178e+00]],

[[-2.8178e+01, -2.1833e+01, -2.0841e+01,  …, -3.1814e+01,
  -2.9155e+01, -2.3921e+01],
 [-1.6537e+01, -7.7142e+00, -5.4735e+00,  …, -1.5723e+01,
  -1.3344e+01, -1.1037e+01],
 [-3.0187e+01, -1.9086e+01, -1.6689e+01,  …, -1.6708e+01,
  -1.2349e+01, -1.1088e+01],
 …,
 [ 5.4189e+00, -1.0927e+00,  1.3396e+00,  …,  1.7691e+00,
   7.5555e+00,  1.7869e+01],
 [ 2.8359e+00, -5.1163e-01, -2.0049e+00,  …, -5.0862e+00,
  -1.2824e+00,  3.8718e+00],
 [ 2.2680e+01,  1.4879e+01,  1.6466e+01,  …,  1.5638e+01,
   2.3021e+01,  1.9702e+01]],

[[-2.5696e+01, -2.3931e+01, -2.4246e+01,  …, -2.6101e+01,
  -2.3882e+01, -2.1652e+01],
  [-1.6936e+01, -1.9242e+01, -1.9237e+01,  …, -1.3919e+01,
   -1.2814e+01, -1.1819e+01],
  [-1.7306e+01, -2.1741e+01, -2.1616e+01,  …, -8.0052e+00,
   -1.2104e+01, -1.3823e+01],
  …,
  [-2.6359e+01, -1.9385e+01, -1.7634e+01,  …, -3.6594e+01,
   -3.6940e+01, -4.0288e+01],
  [-2.4136e+01, -2.9724e+01, -1.9442e+01,  …, -3.0733e+01,
   -2.5684e+01, -1.8613e+01],
  [-2.5504e+01, -3.3385e+01, -3.1209e+01,  …, -2.9970e+01,
   -3.8727e+01, -1.1460e+01]]],


[[[-3.1001e+00, -9.2246e+00, -9.0442e+00,  …, -8.9783e+00,
   -8.9816e+00,  3.0594e-01],
  [-8.6490e+00, -1.3275e+01, -1.3236e+01,  …, -1.3008e+01,
   -1.2898e+01, -4.8151e+00],
  [-8.0927e+00, -1.2842e+01, -1.2898e+01,  …, -1.3415e+01,
   -1.3416e+01, -4.8922e+00],
  …,
  [-7.2041e+00, -1.3866e+00, -1.3573e+01,  …, -1.7296e+01,
   -1.8150e+01, -1.0491e+01],
  [-6.4349e+00, -6.7970e+00, -1.0814e+01,  …, -1.5384e+01,
   -1.4739e+01, -9.7259e+00],
  [-5.9619e+00, -1.1822e+01, -9.6229e+00,  …, -1.5677e+01,
   -1.5517e+01, -1.2978e+01]],

 [[ 5.0677e+01,  4.7824e+01,  4.7883e+01,  …,  4.8143e+01,
    4.8155e+01,  5.7078e+01],
  [ 5.0203e+01,  4.8572e+01,  4.8555e+01,  …,  4.8594e+01,
    4.8605e+01,  5.7261e+01],
  [ 5.0522e+01,  4.8965e+01,  4.8952e+01,  …,  4.8530e+01,
    4.8484e+01,  5.7269e+01],
  …,
  [ 3.8956e+01,  4.2086e+01,  4.0596e+01,  …,  3.1719e+01,
    3.3099e+01,  4.2650e+01],
  [ 3.6880e+01,  3.6319e+01,  3.5156e+01,  …,  3.3159e+01,
    3.4913e+01,  4.3481e+01],
  [ 2.8944e+01,  2.6989e+01,  2.7017e+01,  …,  4.4285e+01,
    4.7736e+01,  5.5637e+01]],

 [[-1.3241e+01, -1.4365e+01, -1.4185e+01,  …, -1.4452e+01,
   -1.4459e+01, -1.9649e+01],
  [-1.1813e+01, -1.3672e+01, -1.3559e+01,  …, -1.3690e+01,
   -1.3690e+01, -1.7138e+01],
  [-1.1489e+01, -1.3263e+01, -1.3177e+01,  …, -1.3591e+01,
   -1.3720e+01, -1.7015e+01],
  …,
 [-1.0049e+01, -1.6519e+01, -2.0010e+01,  …, -9.4448e+00,
  -1.0267e+01, -1.3044e+01],
 [-1.6624e+01, -1.5441e+01, -1.8663e+01,  …, -8.2940e+00,
  -9.4014e+00, -1.2576e+01],
 [-1.4603e+01, -1.1958e+01, -1.2558e+01,  …, -6.2042e+00,
  -6.6740e+00, -9.2599e+00]],

…,

[[-1.2393e+01, -6.5031e+00, -6.6665e+00,  …, -6.8383e+00,
  -6.3234e+00, -3.7010e+00],
 [-1.9295e+01, -1.2381e+01, -1.2423e+01,  …, -1.2346e+01,
  -1.1776e+01, -5.0697e+00],
 [-1.9084e+01, -1.2452e+01, -1.2392e+01,  …, -1.3487e+01,
  -1.2663e+01, -5.6627e+00],
 …,
 [ 1.1929e+00, -1.8529e+01, -3.9911e-01,  …, -1.8148e+01,
  -1.6811e+01, -8.2843e+00],
 [-9.0450e+00, -1.3105e+01, -6.5214e+00,  …, -1.4820e+01,
  -1.6066e+01, -7.1537e+00],
 [-1.7017e+01, -7.8616e+00, -7.7670e+00,  …, -9.7378e+00,
  -6.2483e+00, -1.4493e+00]],

[[-5.8658e+00, -7.3946e+00, -7.5009e+00,  …, -7.4734e+00,
  -7.6398e+00, -1.2877e+01],
 [-5.7157e+00, -7.8365e+00, -7.8766e+00,  …, -8.0914e+00,
  -8.3049e+00, -1.4438e+01],
 [-6.0100e+00, -8.1495e+00, -8.0987e+00,  …, -7.9711e+00,
  -8.0756e+00, -1.4388e+01],
 …,
 [-1.2210e+00, -8.2196e+00, -9.0387e+00,  …, -3.7301e+00,
  -2.2072e+00, -7.8982e+00],
 [ 4.8291e+00, -1.0638e+01, -8.9472e+00,  …, -4.7449e+00,
  -6.3919e+00, -9.8643e+00],
 [-2.3336e+00, -1.0685e+01, -1.1881e+01,  …,  1.9283e+00,
   2.3445e-01, -3.9912e+00]],

[[-8.5313e+00, -1.2759e+01, -1.2498e+01,  …, -1.2607e+01,
  -1.2588e+01, -1.3820e+01],
 [-1.2233e+01, -1.6800e+01, -1.6774e+01,  …, -1.6794e+01,
  -1.6821e+01, -1.8816e+01],
 [-1.1994e+01, -1.6435e+01, -1.6549e+01,  …, -1.6633e+01,
  -1.6826e+01, -1.9219e+01],
 …,
 [-1.7918e+01, -1.7195e+01, -2.4496e+01,  …, -1.1556e+01,
  -1.5040e+01, -1.6272e+01],
 [-2.4725e+01, -8.7759e+00, -1.9647e+01,  …, -1.0714e+01,
  -1.1862e+01, -1.5654e+01],
  [-1.1083e+01, -4.9857e+00, -7.1220e+00,  …, -1.1427e+01,
   -1.0831e+01, -1.3386e+01]]],


[[[-1.2767e+01, -1.6818e+01, -1.4064e+01,  …, -1.1787e+01,
   -8.9471e+00,  1.0894e+00],
  [-1.2502e+01, -1.8712e+01, -1.4233e+01,  …, -1.4859e+01,
   -1.0971e+01, -4.2807e+00],
  [-1.2137e+01, -1.9880e+01, -1.4869e+01,  …, -1.6163e+01,
   -9.9957e+00, -6.7256e+00],
  …,
  [-1.9310e+01, -1.3487e+01, -1.4163e+01,  …, -7.7090e+00,
   -6.1198e+00, -2.4579e+01],
  [-1.7296e+01, -1.1642e+01, -1.2477e+01,  …, -5.0591e+00,
   -8.7896e+00, -2.2204e+01],
  [-1.8976e+01, -1.3423e+01, -1.3488e+01,  …, -9.3324e+00,
   -8.1362e+00, -2.3672e+01]],

 [[ 3.6332e+01,  3.3234e+01,  3.8725e+01,  …,  4.8015e+01,
    5.0417e+01,  6.0567e+01],
  [ 3.9580e+01,  3.7990e+01,  4.3984e+01,  …,  4.9842e+01,
    5.2686e+01,  6.0885e+01],
  [ 4.0881e+01,  3.9017e+01,  4.5452e+01,  …,  5.1290e+01,
    5.3885e+01,  6.0746e+01],
  …,
  [ 4.2957e+01,  4.3948e+01,  4.1577e+01,  …,  4.4772e+01,
    4.7809e+01,  4.8044e+01],
  [ 3.7630e+01,  3.8001e+01,  3.6083e+01,  …,  4.8028e+01,
    4.8778e+01,  4.8773e+01],
  [ 3.2438e+01,  3.2874e+01,  3.1996e+01,  …,  4.0081e+01,
    4.0782e+01,  4.0073e+01]],

 [[-5.0192e+00, -1.3981e+00, -1.3279e+01,  …, -1.1450e+01,
   -1.7736e+01, -2.1389e+01],
  [-1.0375e+01, -5.1030e+00, -1.5259e+01,  …, -9.4831e+00,
   -1.8369e+01, -2.0390e+01],
  [-1.1201e+01, -5.4053e+00, -1.6240e+01,  …, -9.2271e+00,
   -1.8486e+01, -2.0855e+01],
  …,
  [-1.7946e+01, -1.7551e+01, -1.6050e+01,  …, -1.7596e+01,
   -1.6464e+01, -1.0302e+01],
  [-1.7206e+01, -1.6619e+01, -1.4917e+01,  …, -1.9354e+01,
   -2.0695e+01, -1.1979e+01],
  [-1.3614e+01, -1.2213e+01, -1.0540e+01,  …, -1.8080e+01,
   -2.1639e+01, -9.8437e+00]],


 …,
 [[-2.2275e+00, -8.9757e+00, -1.3630e+01,  …, -8.7042e+00,
   -7.5470e+00, -1.6412e+00],
  [-8.5953e+00, -1.4241e+01, -2.1309e+01,  …, -1.4766e+01,
   -1.4059e+01,  2.0182e+00],
  [-1.3088e+01, -1.5970e+01, -1.9499e+01,  …, -1.2905e+01,
   -2.0096e+01,  3.0822e+00],
  …,
  [-5.5938e+00, -9.5057e+00, -7.6016e+00,  …, -4.4739e+00,
    3.2766e+00, -2.1359e+00],
  [-3.3303e+00, -7.9095e+00, -7.3536e+00,  …, -2.7266e+00,
    1.1653e+01, -1.5495e+01],
  [-8.2891e-01, -7.6675e+00, -4.9762e+00,  …,  5.7489e+00,
    2.0813e-01, -1.9666e+01]],

 [[ 7.0357e+00,  5.7774e+00,  8.6581e+00,  …, -1.4706e+00,
   -3.3730e+00, -1.3678e+01],
  [ 5.3876e-01, -9.0047e-01,  8.7200e-01,  …, -7.0625e+00,
   -4.7273e+00, -1.7858e+01],
  [ 1.7483e-01, -1.2421e+00, -7.1415e-01,  …, -5.8907e+00,
   -6.5513e+00, -1.5874e+01],
  …,
  [-1.4818e+01, -1.6987e+01, -1.8398e+01,  …, -7.6960e+00,
   -1.2158e+01,  4.8339e+00],
  [-1.6153e+01, -1.6153e+01, -1.6858e+01,  …, -1.1303e+01,
   -9.1230e+00,  2.1037e+00],
  [-6.8852e+00, -5.5238e+00, -6.6375e+00,  …, -7.3489e+00,
   -1.4262e+00, -5.1585e+00]],

 [[-1.1741e+01, -1.1307e+01, -1.6540e+01,  …, -1.5550e+01,
   -1.5345e+01, -1.3481e+01],
  [-2.2263e+01, -2.0048e+01, -2.2838e+01,  …, -1.7206e+01,
   -2.2226e+01, -1.8034e+01],
  [-2.2306e+01, -2.1711e+01, -2.3000e+01,  …, -1.7811e+01,
   -1.8840e+01, -2.2072e+01],
  …,
  [-3.0171e+01, -3.1469e+01, -2.8675e+01,  …, -2.4108e+01,
   -2.5877e+01, -3.0382e+01],
  [-2.0718e+01, -2.2360e+01, -2.1087e+01,  …, -2.4909e+01,
   -3.3266e+01, -2.8157e+01],
  [-1.4499e+01, -1.5297e+01, -1.4929e+01,  …, -2.6348e+01,
   -3.9555e+01, -1.6067e+01]]],


…,


[[[ 5.0967e+00, -1.0084e+01, -9.0327e+00,  …, -1.2470e+01,
   -9.0265e-01, -1.8491e+01],
  [-2.5771e+00, -1.8235e+01, -1.6503e+01,  …, -1.9635e+01,
   -1.5253e+01, -1.9394e+01],
  [ 1.0125e+00, -2.0304e+01, -1.2437e+01,  …, -2.4948e+01,
   -1.9074e+01, -2.5203e+01],
  …,
  [ 9.2470e+00, -5.2917e+00, -1.4631e+01,  …,  8.7168e-01,
   -2.9242e+00, -2.9149e+01],
  [ 5.6859e+00, -4.6550e+00, -1.3282e+01,  …, -1.8928e+01,
   -1.4319e+00, -4.9589e+00],
  [-1.4747e-02, -6.5308e+00, -8.0711e+00,  …, -3.0203e+01,
   -3.1922e+01, -1.5004e+01]],

 [[ 5.5929e+01,  5.1523e+01,  4.7368e+01,  …,  6.0363e+01,
    5.7917e+01,  4.5125e+01],
  [ 5.1991e+01,  4.7833e+01,  4.5937e+01,  …,  5.6373e+01,
    5.2139e+01,  4.2049e+01],
  [ 5.0815e+01,  4.5602e+01,  4.7886e+01,  …,  5.4040e+01,
    4.8849e+01,  3.8883e+01],
  …,
  [ 5.7696e+01,  5.1291e+01,  4.6761e+01,  …,  5.2450e+01,
    4.6649e+01,  3.7973e+01],
  [ 5.7229e+01,  5.1061e+01,  4.6694e+01,  …,  5.2749e+01,
    5.2379e+01,  4.7281e+01],
  [ 5.7012e+01,  5.1767e+01,  4.8398e+01,  …,  3.8286e+01,
    3.7959e+01,  3.8656e+01]],

 [[-1.4909e+01, -1.2581e+01, -7.6178e+00,  …, -2.4108e+01,
   -1.0787e+01,  1.8612e+01],
  [-5.7478e+00, -2.0431e+00, -2.2336e+00,  …, -8.6318e+00,
   -7.6064e-01,  1.7638e+01],
  [-2.4385e+00,  2.0137e+00, -3.8237e+00,  …,  7.3365e-01,
    3.8655e+00,  1.9142e+01],
  …,
  [-1.7579e+01, -2.2986e+00,  7.7666e+00,  …,  2.6573e+01,
    1.9833e+01, -3.6025e+00],
  [-1.8506e+01, -1.0678e+00,  9.6629e+00,  …,  1.0527e+01,
    1.8620e+01, -2.1214e+00],
  [-2.7276e+01, -6.7728e+00,  5.0934e+00,  …, -1.9850e+01,
   -6.0965e+00,  1.9433e+00]],

 …,

 [[-8.8409e+00, -9.4716e+00, -1.1362e+01,  …, -1.6050e+01,
   -2.4740e+01, -1.8146e+01],
  [-1.4881e+01, -1.3322e+01, -1.5152e+01,  …, -2.4864e+01,
   -1.3686e+01, -2.1691e+01],
  [-1.5768e+01, -1.0849e+01, -1.3319e+01,  …, -1.8748e+01,
   -1.7463e+01, -2.2176e+01],
    …,
    [-1.0408e+01, -1.1204e+01, -1.3798e+01,  …, -5.6421e-01,
     -4.9062e+01, -7.7308e+00],
    [-6.2647e+00, -7.0101e+00, -1.4938e+01,  …,  2.9932e+01,
      2.3033e+00, -3.4037e+01],
    [-1.3583e+01, -8.4847e+00, -2.2977e+01,  …, -2.3276e+01,
      4.8608e+00, -1.8097e+01]],

   [[-3.4112e+01, -2.1172e+01, -1.9995e+01,  …, -3.4592e+01,
     -5.4311e+01, -3.4659e+01],
    [-1.6241e+01, -3.8785e+00, -1.3213e+00,  …, -1.5283e+01,
     -3.1825e+01, -1.9941e+01],
    [-1.9945e+01, -7.8436e-01, -4.2616e-02,  …, -1.9750e+01,
     -2.1683e+01, -1.7238e+01],
    …,
    [-3.4479e+01, -2.4218e+01, -1.5496e+01,  …, -5.0123e+01,
     -1.1741e+01,  1.2466e+01],
    [-2.9350e+01, -2.3804e+01, -9.8162e+00,  …, -1.5077e+01,
     -3.4960e+01, -2.4542e+00],
    [-3.0539e+01, -3.1002e+01, -1.9430e+01,  …, -1.6496e+00,
      5.8103e+00, -9.3537e+00]],

   [[-2.4842e+01, -2.1910e+01, -1.6950e+01,  …, -3.5111e+01,
     -2.1316e+01, -1.9670e+01],
    [-1.7725e+01, -1.2618e+01, -1.0401e+01,  …, -1.8454e+01,
     -6.8444e+00, -1.0781e+01],
    [-1.3914e+01, -1.2362e+01, -1.4696e+01,  …, -9.2753e+00,
     -1.2137e+01,  1.4622e+00],
    …,
    [-1.3846e+01, -1.0725e+01, -9.1671e+00,  …,  2.9190e+01,
     -1.0352e+01, -2.7418e+01],
    [-1.0178e+01, -6.4742e+00, -1.0189e+01,  …, -3.5150e+01,
      1.7122e+01, -1.9479e+01],
    [-2.2754e+01, -1.6117e+01, -1.6197e+01,  …, -3.5629e+01,
     -3.8831e+01,  5.8267e-01]]],


  [[[-1.6125e+01, -1.5150e+01, -1.5113e+01,  …, -1.5465e+01,
     -1.4983e+01, -9.5651e+00],
    [-1.7931e+01, -1.6518e+01, -1.6843e+01,  …, -1.5504e+01,
     -1.5545e+01, -1.1409e+01],
    [-1.7795e+01, -1.7382e+01, -1.6945e+01,  …, -1.5921e+01,
     -1.6138e+01, -1.1618e+01],
    …,
    [-1.6822e+01, -1.4257e+01, -1.4780e+01,  …, -1.3750e+01,
     -1.0905e+01, -1.5300e+01],
    [-1.8259e+01, -1.5663e+01, -1.5572e+01,  …, -1.9344e+01,
     -1.4362e+01, -1.5190e+01],
  [-1.8495e+01, -1.6938e+01, -1.6205e+01,  …, -1.9315e+01,
   -1.8616e+01, -1.6530e+01]],

 [[ 2.8253e+01,  2.8393e+01,  2.8378e+01,  …,  2.9242e+01,
    2.9472e+01,  3.6659e+01],
  [ 3.0067e+01,  3.0637e+01,  3.0590e+01,  …,  3.3206e+01,
    3.3199e+01,  3.9605e+01],
  [ 3.0536e+01,  3.0986e+01,  3.1160e+01,  …,  3.6189e+01,
    3.6193e+01,  4.2170e+01],
  …,
  [ 3.2233e+01,  3.3362e+01,  3.3337e+01,  …,  4.1651e+01,
    4.2687e+01,  4.2196e+01],
  [ 3.2656e+01,  3.3699e+01,  3.3723e+01,  …,  4.0188e+01,
    4.1656e+01,  4.2694e+01],
  [ 4.1027e+01,  4.2511e+01,  4.2678e+01,  …,  3.9111e+01,
    4.0209e+01,  4.1538e+01]],

 [[-7.3916e+00, -8.7857e+00, -8.7019e+00,  …, -6.1327e+00,
   -8.7339e+00, -1.2866e+01],
  [-7.2649e+00, -8.8559e+00, -8.8839e+00,  …, -7.2552e+00,
   -8.7175e+00, -1.2612e+01],
  [-6.5689e+00, -8.0857e+00, -8.4147e+00,  …, -6.1601e+00,
   -6.4998e+00, -1.1762e+01],
  …,
  [-7.4031e+00, -8.4286e+00, -8.3045e+00,  …, -1.1512e+01,
   -1.0160e+01, -1.0172e+01],
  [-7.7127e+00, -8.5827e+00, -8.4112e+00,  …, -1.1830e+01,
   -1.1962e+01, -1.3264e+01],
  [-4.3854e+00, -5.7643e+00, -5.7724e+00,  …, -1.2064e+01,
   -1.2504e+01, -1.1882e+01]],


 …,


 [[-9.4482e+00, -5.5238e+00, -5.1837e+00,  …, -4.5249e+00,
   -4.6407e+00, -3.0494e+00],
  [-1.8043e+01, -1.3564e+01, -1.3207e+01,  …, -1.2367e+01,
   -1.3013e+01, -8.0690e+00],
  [-1.7740e+01, -1.2776e+01, -1.4111e+01,  …, -1.0979e+01,
   -1.1374e+01, -7.3419e+00],
  …,
  [-1.3435e+01, -1.2624e+01, -1.2688e+01,  …, -1.2880e+01,
   -1.2605e+01, -9.9429e+00],
  [-1.5409e+01, -1.2423e+01, -1.1987e+01,  …, -6.7415e+00,
   -8.4393e+00, -1.4966e+01],
  [-1.1454e+01, -7.0049e+00, -8.5422e+00,  …, -5.6292e+00,
   -8.0914e+00, -1.2949e+01]],

 [[ 2.6915e+00, -9.8735e-01, -1.4668e+00,  …,  1.9520e+00,
     2.1468e+00,  3.5994e+00],
   [-1.6721e+00, -7.2796e+00, -6.4648e+00,  …, -5.9878e+00,
    -5.9568e+00, -3.5183e+00],
   [-8.3610e-01, -5.6818e+00, -5.1904e+00,  …, -2.9632e+00,
    -3.1473e+00, -4.0742e-01],
   …,
   [-2.8304e+00, -6.9419e+00, -7.1605e+00,  …, -1.4794e+01,
    -1.4044e+01, -3.4584e+00],
   [-1.1226e+00, -6.6186e+00, -6.1750e+00,  …, -7.6376e+00,
    -7.8300e+00,  9.4732e-01],
   [ 9.0099e+00,  5.8943e+00,  6.1622e+00,  …,  1.1254e+01,
     9.0353e+00,  9.0495e+00]],

  [[-8.0897e+00, -1.1245e+01, -1.0481e+01,  …, -1.2307e+01,
    -1.2219e+01, -1.3269e+01],
   [-1.4680e+01, -1.6237e+01, -1.6932e+01,  …, -1.8513e+01,
    -1.8377e+01, -1.9928e+01],
   [-1.4623e+01, -1.6330e+01, -1.6825e+01,  …, -1.7836e+01,
    -1.7378e+01, -1.9828e+01],
   …,
   [-1.8652e+01, -1.9709e+01, -1.9030e+01,  …, -1.6597e+01,
    -1.8575e+01, -2.0245e+01],
   [-1.7795e+01, -1.6987e+01, -1.7832e+01,  …, -1.4618e+01,
    -1.4451e+01, -1.7815e+01],
   [-1.5938e+01, -1.6160e+01, -1.6188e+01,  …, -1.8360e+01,
    -1.5922e+01, -1.4264e+01]]],


 [[[-1.7605e+01, -1.1849e+01, -1.3955e+01,  …, -1.3956e+01,
    -1.9053e+01, -3.6717e+01],
   [-9.5270e+00, -1.1417e+01, -1.3000e+01,  …, -2.1789e+00,
    -1.0761e+01, -2.8075e+01],
   [-1.2975e+01, -1.5795e+01, -1.4902e+01,  …,  1.6842e+00,
    -7.4408e+00, -2.8238e+01],
   …,
   [ 7.0708e-01, -3.2132e+00, -2.3986e+00,  …, -1.5119e+00,
    -8.2266e+00, -2.8414e+01],
   [-7.3614e+00, -1.7161e+01, -1.4477e+01,  …, -1.1510e+01,
    -1.0324e+01, -2.8566e+01],
   [-1.2263e+01, -1.1611e+01, -3.1698e+00,  …, -1.1251e+01,
    -1.0861e+01, -2.8196e+01]],

  [[ 4.9767e+01,  5.1891e+01,  5.1725e+01,  …,  4.8435e+01,
     4.8995e+01,  4.8762e+01],
   [ 4.7599e+01,  4.7469e+01,  4.7248e+01,  …,  4.3121e+01,
     4.2015e+01,  4.2867e+01],
   [ 4.6881e+01,  4.6048e+01,  4.5639e+01,  …,  4.4633e+01,
     4.2732e+01,  4.2554e+01],
  …,
  [ 6.0245e+01,  5.8456e+01,  5.8758e+01,  …,  4.8448e+01,
    4.6545e+01,  4.4698e+01],
  [ 6.0754e+01,  6.1253e+01,  6.2883e+01,  …,  4.7720e+01,
    4.6618e+01,  4.4872e+01],
  [ 5.7577e+01,  5.9588e+01,  6.0929e+01,  …,  3.8692e+01,
    3.8720e+01,  3.8334e+01]],

 [[-1.2734e+01, -9.3974e+00, -7.2698e+00,  …, -7.7586e+00,
   -1.1343e+01, -2.8483e+00],
  [-1.2438e+01, -1.1791e+01, -1.2405e+01,  …, -8.8592e+00,
   -1.4812e+01, -6.9104e+00],
  [-1.2091e+01, -1.3322e+01, -1.3096e+01,  …, -5.8542e+00,
   -1.4768e+01, -6.8460e+00],
  …,
  [-8.8908e+00, -8.2808e+00, -1.5002e+01,  …, -1.1693e+01,
   -1.3974e+01, -6.5787e+00],
  [-1.6386e+01, -1.5254e+01, -1.6293e+01,  …, -1.2177e+01,
   -1.3987e+01, -6.6414e+00],
  [-1.3812e+01, -1.2576e+01, -1.0048e+01,  …, -9.5948e+00,
   -1.2693e+01, -6.9896e+00]],


 …,


 [[-1.1572e+01, -1.8130e+01, -1.4085e+01,  …, -2.3609e+01,
   -2.1832e+01, -2.8408e+01],
  [-8.4045e+00, -3.0028e+00, -6.2749e-01,  …, -1.0710e+01,
   -2.2782e+00, -1.1933e+01],
  [-1.1031e+01, -9.4328e+00, -1.0155e+01,  …, -9.6001e+00,
   -4.4380e+00, -1.2352e+01],
  …,
  [ 1.1593e+00,  2.9679e+00,  4.1871e+00,  …, -5.0473e+00,
   -6.1338e+00, -1.2678e+01],
  [-1.3248e+01, -1.3067e+01, -2.1880e+01,  …, -3.7261e+00,
   -5.3419e+00, -1.2043e+01],
  [-1.9925e+01, -3.7456e-01, -6.9647e+00,  …, -3.3882e+00,
   -5.2048e+00, -1.2702e+01]],

 [[-1.2569e+01, -1.6690e+01, -1.7445e+01,  …, -1.4552e+01,
   -1.7473e+01, -7.2417e+00],
  [-2.6344e+00, -4.1976e+00, -2.1035e+00,  …, -7.2773e+00,
   -9.5098e+00, -3.5523e+00],
  [-4.9395e+00, -3.0297e-03,  9.7538e-02,  …, -8.6445e+00,
   -9.4160e+00, -3.3011e+00],
  …,
  [-9.8185e+00, -1.1415e+01, -4.3601e+00,  …, -1.2335e+01,
   -1.1394e+01, -4.0641e+00],
  [-3.6856e+00,  3.9116e-01,  3.0509e+00,  …, -9.4951e+00,
           -1.1570e+01, -4.2247e+00],
          [-8.8407e+00, -1.2467e+01, -2.0858e+01,  …, -8.1545e+00,
           -9.9456e+00, -2.7867e+00]],

         [[-2.4746e+01, -1.8197e+01, -1.6822e+01,  …, -3.3016e+01,
           -3.0315e+01, -2.4236e+01],
          [-1.8510e+01, -1.6996e+01, -1.7923e+01,  …, -2.1979e+01,
           -2.3610e+01, -1.4913e+01],
          [-1.5786e+01, -1.9865e+01, -1.9384e+01,  …, -1.7829e+01,
           -2.3175e+01, -1.4818e+01],
          …,
          [-1.6785e+01, -1.0010e+01, -1.9299e+01,  …, -1.8206e+01,
           -2.1535e+01, -1.3976e+01],
          [-2.1460e+01, -2.1609e+01, -2.5041e+01,  …, -2.0641e+01,
           -2.1609e+01, -1.4308e+01],
          [-2.4358e+01, -2.6418e+01, -1.7508e+01,  …, -2.1478e+01,
           -2.0400e+01, -1.3858e+01]]]], device='cuda:0',
        grad_fn=<ConvolutionBackward0>)
```

[101]:
```python
difference = abs( output_ref - output_recovered )
print(difference.mean())  ## It should be small, e.g.,2.3 in my trainned model
```

```
tensor(1.3720, device='cuda:0', grad_fn=<MeanBackward0>)
```

[4]:
```python
# Now construct a 2 bit version model
model_name_2bit = "Resnet20_quant_2bit"
model_2bit = resnet20_quant_2bit()
model_2bit.cuda()
print(model_2bit)
```

```
ResNet_Cifar_2bit(
  (conv1): QuantConv2d(
    3, 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)
  (layer1): Sequential(
    (0): BasicBlock_2bit(
      (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_2bit(
      (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_2bit(
      (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_2bit(
      (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_2bit(
      (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_2bit(
      (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_2bit(
      (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_2bit(
   (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_2bit(
   (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)
)
```

[9]:

```
fdir = 'result/'+str(model_name_2bit)+'/model_best.pth.tar'

checkpoint = torch.load(fdir)
model_2bit.load_state_dict(checkpoint['state_dict'])
device = torch.device("cuda")

model_2bit.cuda()
model_2bit.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_2bit(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: 8237/10000 (82%)
```