Assignment 3

Question 1

Given a 32x32 pixels, 3 channels input. Fill the pixel values with torch.randn(. . .)

Foreach pytorch functions in the list,

1. Initialise the weights with uniform random numbers r
2. Call the functions and get the output tensors - torch_out
3. Implement these functions from scratch, without using any neural network libraries. Use linear algebra libraries in python is ok. Output your tensors as — my_out
4. Compare and show that torch_out and my_out are equal up to small numerical errors

1. torch.nn.MaxPool2d(kernel_size=2, stride=1, padding=0, dilation=1, return_indices=False, ceil_mode=False)

2. torch.nn.AvgPool2d(kernel_size=2, stride=1, padding=0, ceil_mode=False, count_include_pad=True, divisor_override=None)

3. torch.nn.Conv2d(in_channels=3, out_channels=6, kernel_size=3, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros')

4. torch.nn.Conv2d(in_channels=3, out_channels=6, kernel_size=5, stride=2, padding=0, dilation=2, groups=1, bias=True, padding_mode='zeros')

5. torch.nn.ConvTranspose2d(in_channels=3, out_channels=4, kernel_size=3, stride=1, padding=0, output_padding=0, groups=1, bias=True, dilation=1, padding_mode='zeros')

1. torch.flatten(input, start_dim=0, end_dim=-1)

2. torch.sigmoid(input, *, out=None)

3. torchvision.ops.roi_pool(input: torch.Tensor, boxes: torch.Tensor, output_size: None, spatial_scale: float = 1.0)

4. torch.nn.functional.batch_norm(input, running_mean, running_var, weight=None, bias=None, training=False, momentum=0.1, eps=1e-05)

5. torch.nn.functional.cross_entropy(input, target, weight=None, size_average=None, ignore_index=-100, reduce=None, reduction='mean')

6. torch.nn.functional.mse_loss(input, target, size_average=None, reduce=None, reduction='mean')

Question 2a

Implement a CNN and train for CIFAR10 with these definitions

1. cA-B = Conv2d with input A channels, output B channels - kernel size 3x3, stride (1,1), padding with zeros to keep image size constant, followed by ReLU
2. mp = maxpool2d kernel size 2x2, stride (2,2)
3. bn = batchnorm2d with affine=False (i.e. non learning batch norm)
4. fcA-B = nn.linear with input A nodes, output B nodes
5. aap = adaptive average pooling

Use the definition to make the architecture

c3-16 -> c16-16 -> mp -> c16-32 -> c32-32 -> mp
-> c32-64 -> c64-64 -> mp -> c64-128 -> c128-128
-> aap -> flatten -> fc128-10 -> cross entropy loss

Adjust learning rate, batch size and other hyper parameters to make classification results > 80%

Compute the average absolute values of activations average across all train data and channels for each layer, normalise your averaged activations to make them between [0,1]

Print out picture of average absolute values of activations for each layer, show in your assignment report

Explain your observations and make conclusions

Question 2b

Repeat question 2a by adding batch normalization layers with batch size 500:

c3-16 -> bn -> c16-16 -> bn -> mp -> c16-32 -> bn
-> c32-32 -> bn -> mp -> c32-64 -> bn -> c64-64 -> bn
-> mp -> c64-128 -> bn -> c128-128 -> bn
-> aap -> flatten -> fc128-10 -> cross entropy loss

Print out picture of average absolute values of activations for each layer, show in your assignment report

Explain your observations and make conclusions.

What is the different between observations of Q2a and Q2b and why?