**CAP5415**

**Computer Vision**

**Fall 2022**

**Homework 6**

**Questions 1:**

If a tensor defined as T = torch.rand(1,64,32,32,4) is transformed using T.view(1,x,16,16,-1), which of the following is a valid combination (multiple options may be true)

A. T.view(1,128,16,16,8)

B. T.view(1,24,16,16,4)

C. T.view(1,96,16,16,4)

D. T.view(1,32,16,16,32)

Answer: A, D. view() is used to restructure the tensor, while keeping the number of parameters same. If you multiply all values, only A and D keep the number same as original tensor.

T->1\*64\*32\*32\*4     = 262144

A-> 1\*128\*16\*16\*8 = 262144

B-> 1\*24\*16\*16\*4    = 24576

C-> 1\*96\*16\*16\*4    = 98304

D-> 1\*32\*16\*16\*32  = 262144

**Questions 2:**

We have defined these layers for our network,

def **\_\_init\_\_**(self):

super(Net, self).\_\_init\_\_()

# 1 input image channel, 6 output channels, 3x3 square convolution kernel

self.conv1 = nn.Conv2d(3, 32, 3)

self.conv2 = nn.Conv2d(32, 512, 3)

self.conv3 = nn.Conv2d(512, 512, 3)

self.fc1 = nn.Linear(**X**, 1024)

self.fc2 = nn.Linear(1024, 512)

self.fc3 = nn.Linear(512, 10)

Now, we use these layers to define our network,

def **forward**(self, x):

x = F.max\_pool2d(F.relu(self.conv1(x)), (2, 2))

x = F.max\_pool2d(F.relu(self.conv2(x)), 2)

x = x.view(-1,**Y**)

x = F.relu(self.fc1(x))

x = F.relu(self.fc2(x))

x = self.fc3(x)

**return** x

If the input image has shape (3, 224, 224), what are the values for **X and Y?**

**Answer:**

**None of these.**

Input is 3x224x224 tensor, and we use 2 Conv2d (we don’t use self.conv3 in forward pass) and 2 Maxpool\_2d, followed by 3 linear layers. Since we want final output to be of size [1x10], by applying the formula of {{W-K+2P}/{S}}+1 we calculate values of x.

x = F.max\_pool2d(F.relu(self.conv1(x)), (2, 2))

x = F.max\_pool2d(F.relu(self.conv2(x)), 2)

x = x.view(-1, Y)

**Questions 3:**

We need to convert NumPy arrays to PyTorch tensors as they can be used for computation even on a GPU.

True/False

**True**

**All computations on GPU is performed by converting Numpy arrays into tensors**

**Question 4:**

We have a network **net**, which takes **in** as input, and provides **out** as output. We also define a loss function **criterion** which can give us a loss value using **out** and the ground-truth **target**. Now, consider the following code snippet.

out=net(in)  
loss=criterion(out,target)  
loss.backward()

Calling a backward on loss will compute all the gradients we need and, also update the network parameters.

True/False

**False**

The backward function only computes the gradient for the model, but it does not update the parameters. Parameters are updated by the optimizer based on the gradient values in a separate step.

**Question 5:**

In PyTorch, as discussed during the lecture the forward function is used to implement the feedforward step of the network.

True/False

**True:** forward function is used for passing the inputs to the network layers in feedforward step