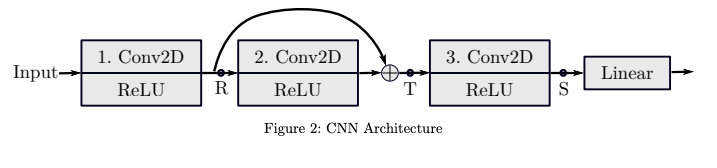
### Coding:

You want to implement the following architecture in PyTorch. It consists of multiple convolutional layers with different strides and padding (specified in the Model constructor below), each followed by a ReLU activation.

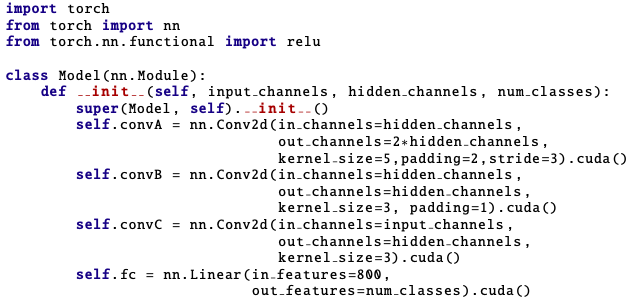


Remark: All convolutional layers in PyTorch are per default in ”valid” mode, which means no padding at the borders is applied (padding value of 0). If the padding value is larger than 0, padding is applied before the convolution in height and width dimension on both sides of the image. The padding value (in the constructor) defines the padding width of one side of the image.

**Question 1:**

**Define the model below in Pytorch**

**Solution 1:**



**Question 2:**

To check the correct handling of the image sizes in your network, fill in the following tables. First, find the correct order of the convolution objects defined in the constructor. Then, enter the shape of the weights for the above-defined convolutions (bias can be neglected) and the size of your images at the locations **R, T, S** for the defined input size. The weights should be defined as ***(number of kernels, channel-dimension, x-dimension, y-dimension)*** while the image sizes should have the shape ***(batchsize, number of channels, x-dimension, y-dimension)***.

Note: in\_channels is set to 3 and hidden channels is set to 4.

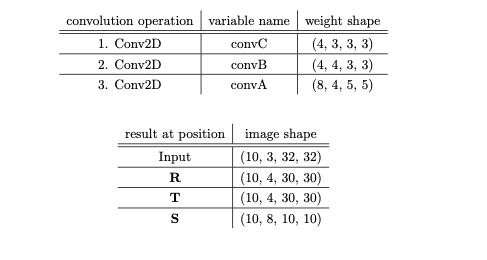
Table-1:

|  |  |  |
| --- | --- | --- |
| **convolution operation** | **variable name** | **weight shape** |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |
| --- | --- |
| **result at position** | **image shape** |
| Input | (10, 3, 32, 32) |
| **R** |  |
| **T** |  |
| **S** |  |

Table-2:

**Solution 2:**



**Question 3:**

Implement the forward pass for the given architecture in python using the PyTorch library. The input x is located on the CPU.

**Solution 3:**

