

Homework #2

Problem 1

Problem 1.a.

A tensor “temp” from `list(range(9))` was created. The torch member function “.tensor()” was used to do this conversion. This will result in a tensor of type long with size of 9. However, in PyTorch version $\leq 1.8.0$ if the class member function is instantiated as “.Tensor()” this will convert the list to a tensor of type float with a size of 9. The attributes of the tensor “temp” are below in Table 1. Methods for doing such conversion and measuring of attributes is demonstrated in the “hw2.py” provided within the “HW2” folder of the submission package.

Type (Before and After Cast):	Long -> Float
Size:	9
Offset:	0
Stride	(1,)

Table 1: temp Tensor Attributes

Problem 1.b.

The cosine and square-root functions exist within the PyTorch library. These functions exist as the “torch.cos()” and “torch.sqrt()” implementations respectively. Each of these functions will conduct the element-wise function on a given input tensor.

Problem 1.c.

The element-wise cosine and square when applied to the tensor “temp” will return an error when using PyTorch $\leq 1.8.0$ if the list was converted using the “.tensor()” method. This will convert the list to a tensor of type long and the cosine and square-root functions on PyTorch $< 1.8.0$ expect a tensor of type float. Furthermore, if the list was converted to a tensor using the “.Tensor()” function these element-wise operations will not return an error as the tensor type will be returned as float. Lastly, if PyTorch version 1.8.0 is being used the tensor could be converted using either method (long/float) and an error will not be returned as in this version the cosine and square-root functions can handle tensors of type long.

Problem 1.d.

In order to enable functionality of the cosine and square-root functions on the tensor in PyTorch $< v1.8.0$ the tensor of type “long” must be cast to be type “float” using the “type.(torch.FloatTensor)”

functionality. However, depending on how the tensor was converted, or the version being used, the tensor may already be passable into the cosine and square-root functions.





Problem 1.e.

The cosine and square-root functions can be replaced by in-place functions in PyTorch using the “torch.cos_(temp)” or “torch.sqrt_(temp)” functions. These functions will do the same as the original cosine and square-root functions, but will directly perform the calculation (in-place) without copying the tensor.

Problem 2

Problem 2.a.

Several red, green, and blue images were captured to do analysis on this problem. The images were initially opened using the PIL library and converted to be RGB channels. Then, each of the images were transformed into tensors. The images being in tensor form enabled various operations to be performed such as brightness, and mean of RGB channels to identify the color of the image. Table 2 below shows the attributes associated with each image. Methods for doing such conversion and measuring of attributes is demonstrated in the “hw2.py” provided within the “HW2” folder of the submission package.

		Channel Means (RGB):		
Input Image (name):	Tensor Mean (brightness):	Red:	Green:	Blue:
 (r1)	0.3226	0.6609	0.2239	0.0731
 (r2)	0.3335	0.6357	0.220	0.1427
 (r3)	0.3129	0.6236	0.2013	0.1138
 (g1)	0.5654	0.5989	0.7468	0.3506





 (g2)	0.3930	0.4091	0.5700	0.200
 (b1)	0.4187	0.3008	0.4146	0.5407
 (b2)	0.6637	0.3872	0.8295	0.7742
 (b3)	0.2904	0.2302	0.2342	0.4068

Table 2: RGB Images and Attributes.

Problem 2.b.

Each of the individual images were converted into tensors as discussed in Problem 2.a.

Problem 2.c.

As in table 2 above, the mean method was conducted on each image tensor to get an understanding of how bright the image may be. Taking a look at some of these results correctly demonstrates this concept. For example, the mean of image “b3” is 0.2904 and the image is rather dim; however, the mean of tensor image “b2” is 0.6637 and the image is substantially brighter. This applies to all of the other input images as well.

Problem 2.d.

The mean of each of the channels (RGB) of the image tensors was calculated as in Table 2 above in section 2.a. Examining these results, the greatest value of the three channels typically correlates to the color of the image. However, in some cases this is not entirely accurate. For example, in image “b2” which is a very light-teal color, the green channel has the highest value of 0.8295 and the blue channel has the second highest with a value of 0.7742. Furthermore, this average is simply showing the saturation (heatmap) of each of the different channel hues so both the green and blue channel exhibiting high values for a teal image is still valid. Likewise, on images that are the most dim, the mean for each of the channels

are typically lower. Overall, nearly all of the images will have the largest value in the RGB channel for which color it may be if it is nearly a true red, green, or blue image.