Week 3 Report

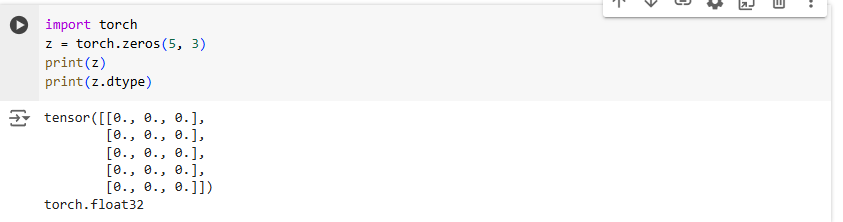
Neural Networks

In this week we discussed the basics of neural networks and their working using pytorch.

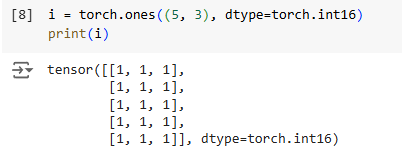
We discussed about:

* Tensors which are the core data abstraction of pytorch.
* Autograd that drives the eager mode computation that makes rapid iteration in the model.
* Building a model using pytorch models
* Efficient loading of data and training models
* Deployment with torch script

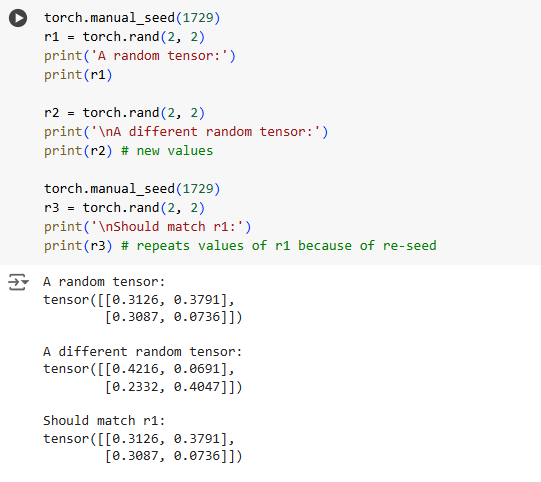
Creating a matrix 5x3 with all zero and find its datatype (float):



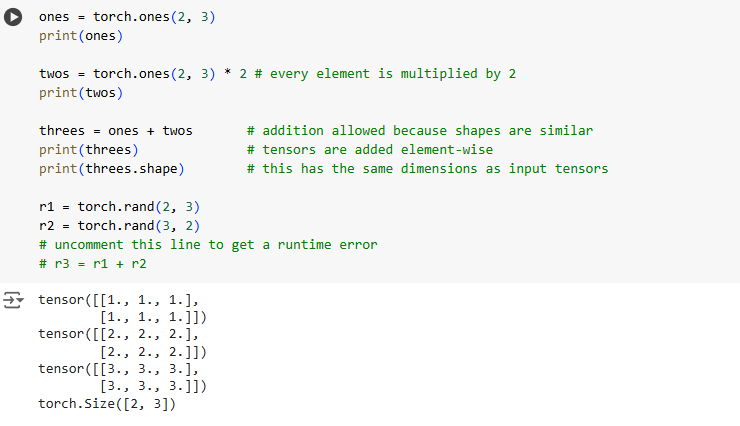
Using integers instead:



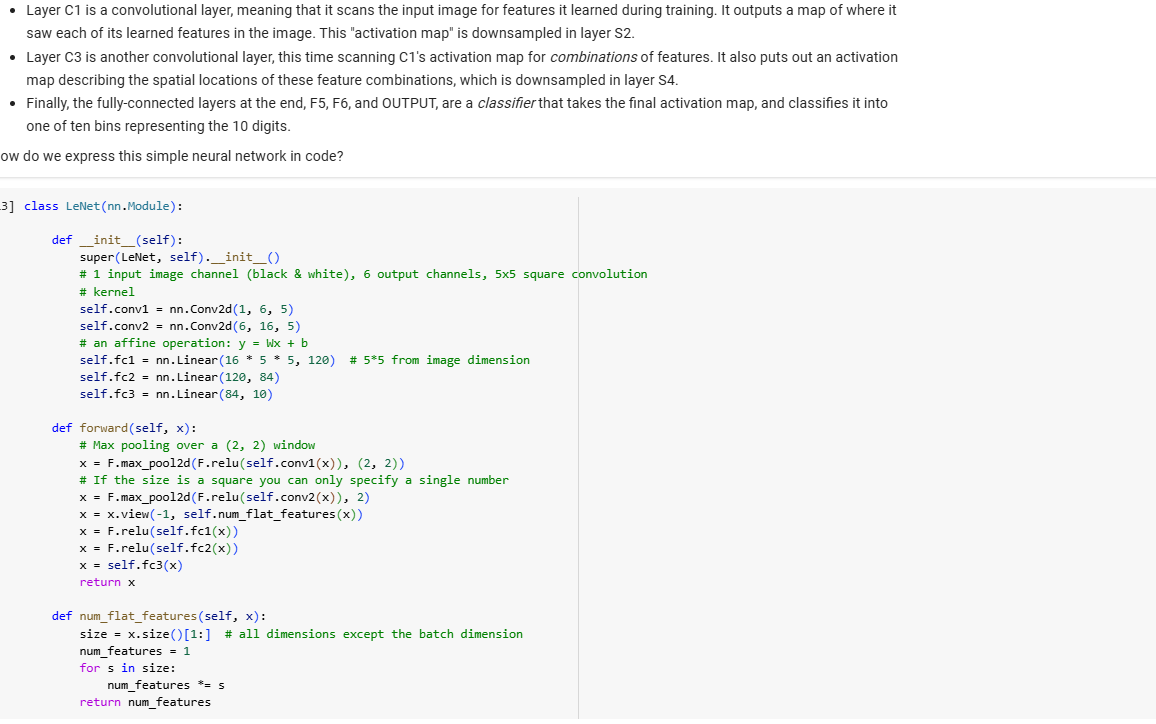
Random tensors generated with the same seed are the same:



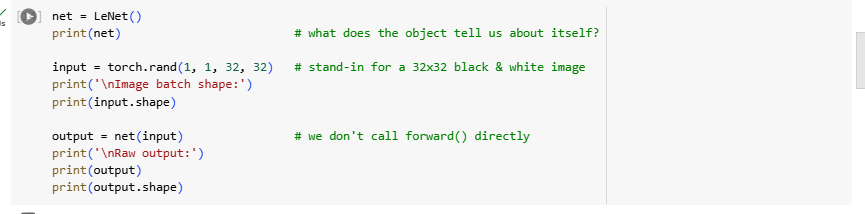
Arithmetic operations can be performed on tensors with the same size:



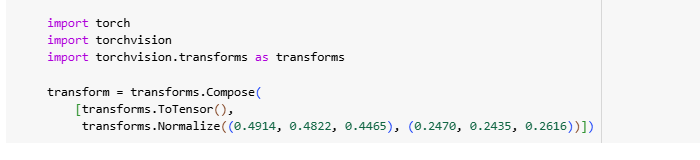
Example of a neural network



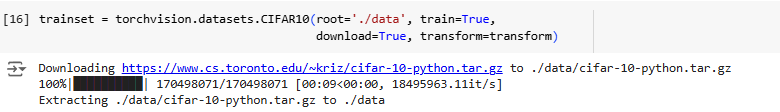
The output of this call represents the model's confidence that the input represents a particular digit.



 Transforming our incoming images into a PyTorch tensor:



 Creating an instance of the CIFAR10 dataset and generating and shuffling 4 images:

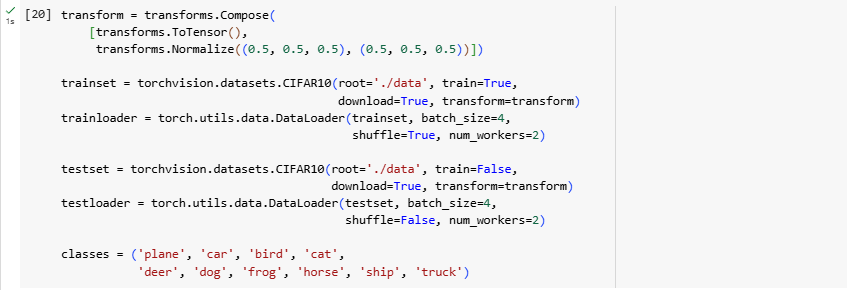


Running the below cell should show you a strip of four images, and the correct label for each.

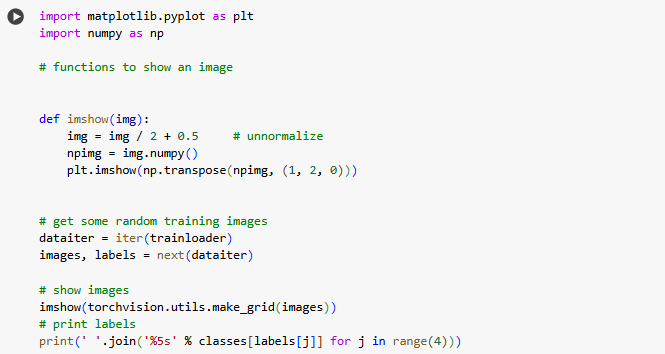


Training your model:

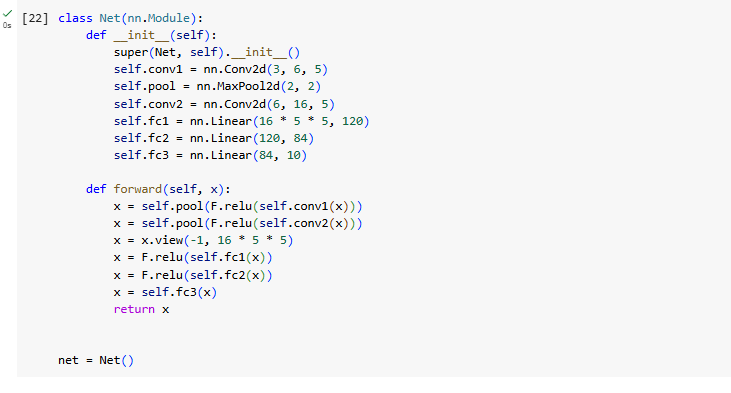
Downloading training and test datasets.



Checking output from the data loader:



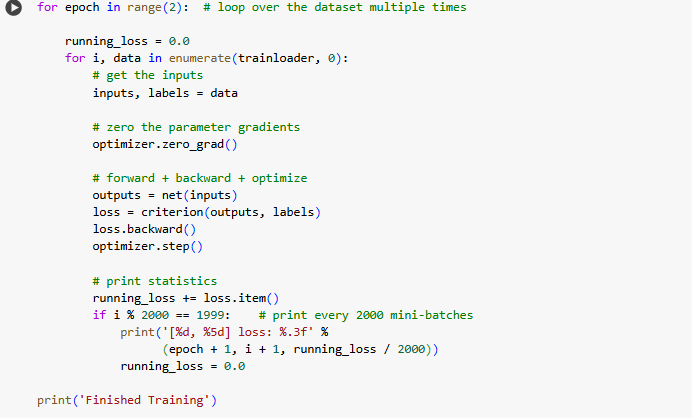
Model to train:



Loss function and optimizer:



Assembling these in the training loop:

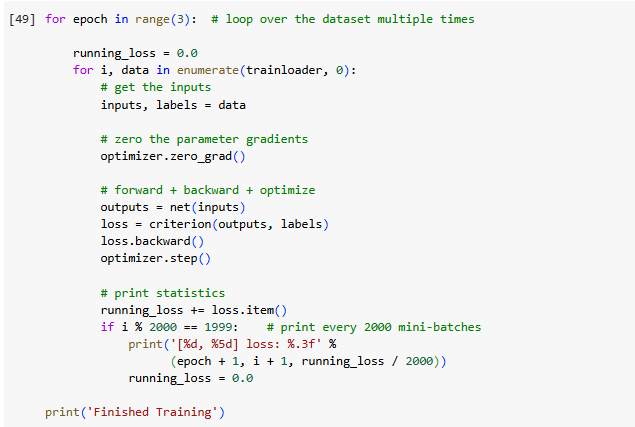


As a final step, we should check that the model is actually doing general learning, and not simply "memorizing" the dataset.



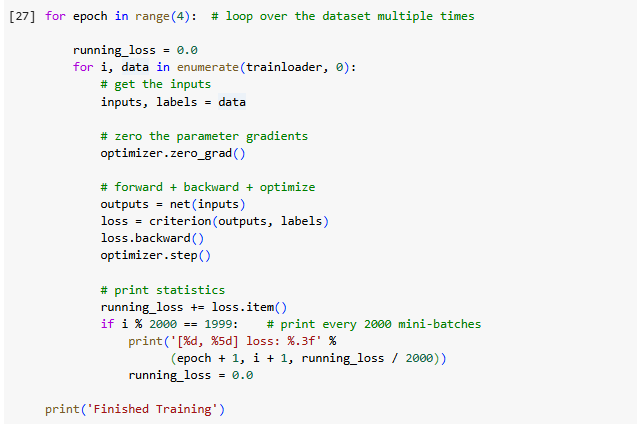
As the number of epochs increase, the loss is better optimised ,and the model is better trained:

Changing epoch to 3 :



Output :



Changing epoch to 4 : 

Output:



Changing the learning rate of the model:

Changing lr to 0.01:



Output:



Changing lr to 0.0001:



Output:



As momentum decreases the accuracy of the model also decreases:

Changing momentum to 0.8:



Output:



Changing momentum to 0.7:



Output:

