Welcome!

Welcome! In this lesson, you'll learn how to use PyTorch for building deep learning models. PyTorch was released in early 2017 and has been making a pretty big impact in the deep learning community. It's developed as an open source project by the Facebook AI Research team, but is being adopted by teams everywhere in industry and academia. In my experience, it's the best framework for learning deep learning and just a delight to work with in general. By the end of this lesson, you'll have trained your own deep learning model that can classify images of cats and dogs.

I'll first give you a basic introduction to PyTorch, where we'll cover **tensors** - the main data structure of PyTorch. I'll show you how to create tensors, how to do simple operations, and how tensors interact with NumPy.

Then you'll learn about a module called **autograd** that PyTorch uses to calculate gradients for training neural networks. Autograd, in my opinion, is amazing. It does all the work of backpropagation for you by calculating the gradients at each operation in the network which you can then use to update the network weights.

Next you'll use PyTorch to build a network and run data forward through it. After that, you'll define a loss and an optimization method to train the neural network on a dataset of handwritten digits. You'll also learn how to test that your network is able to generalize through **validation**.

However, you'll find that your network doesn't work too well with more complex images. You'll learn how to use pre-trained networks to improve the performance of your classifier, a technique known as **transfer learning**.

Follow along with the videos and work through the exercises in your own notebooks. If you get stuck, check out my solution videos and notebooks.

Get the notebooks

The notebooks for this lesson will be provided in the classroom, but if you wish to follow along on your local machine, then the instructions below will help you get setup and ready to learn!

All the notebooks for this lesson are available from our deep learning repo on GitHub. Please clone the repo by typing

git clone https://github.com/udacity/deep-learning-v2-pytorch.git

in your terminal. Then navigate to the intro-to-pytorch directory in the repo.

Follow along in your notebooks to complete the exercises. I'll also be providing solutions to the exercises, both in videos and in the notebooks marked (Solution).

Dependencies

These notebooks require PyTorch v0.4 or newer, and torchvision. The easiest way to install PyTorch and torchvision locally is by following the instructions on the PyTorch site. Choose the stable version, your appropriate OS and Python versions, and how you'd like to install it. You'll also need to install numpy

and jupyter notebooks, the newest versions of these should work fine. Using the conda package manager is generally best for this,

conda install numpy jupyter notebook

If you haven't used conda before, please read the documentation to learn how to create environments and install packages. I suggest installing Miniconda instead of the whole Anaconda distribution. The normal package manager pip also works well. If you have a preference, go with that.

The final part of the series has a soft requirement of a GPU used to accelerate network computations. Even if you don't have a GPU available, you'll still be able to run the code and finish the exercises. PyTorch uses a library called CUDA to accelerate operations using the GPU. If you have a GPU that CUDA supports, you'll be able to install all the necessary libraries by installing PyTorch with conda. If you can't use a local GPU, you can use cloud platforms such as AWS, GCP, and FloydHub to train your networks on a GPU.

Feedback

If you have problems with the notebooks, please contact support or create an issue on the repo. We're also happy to incorporate your improvements through pull requests.