Neural Net Builder

Prof. Dennis Shasha
shasha@cims.nyu.edu

Vikram Mullachery
mv333@nyu.edu

Abstract

Neural Net Builder is a visual tool for designing and building neural networks. This tool intends to make the art of building neural nets less arcane to data scientists, data analysts, research scientists and non-programmers.

Machine learning and specifically deep neural networks have transformed our ability to predict and classify a variety of features and behaviors. However, this requires a well trained computer programmer and a deep learning expert to build neural networks. With this visual tool, we intend to make building a neural net easier, with much less code development time, and much more consistent and robust code for various machine learning libraries, namely Tensorflow and PyTorch.

The key use cases of this tool are: rapid, robust and error-free neural net construction, consistent reproduce-able neural network configurations, and a visual tool for code generation.

Related Work

Work in this field is minimal and the only notable one is by A. Karpathy, ConvNetJS. ConvNetJS is a machine learning framework in javascript; however it falls short of building a visual toolbox and generating code for robust production ready frameworks such as Tensorflow or PyTorch. Further, ConvNetJS is self-described as a toy two layer neural net for educational purposes.

Work Completed

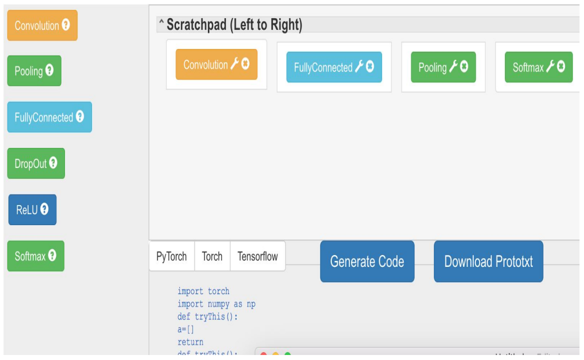
This application is currently ongoing rapid development, and has accomplished a few major feature milestones and program code milestones.

Current features include the availability of drag-and-droppable components of feed forward neural networks - Convolution, Pooling, FullyConnected and Softmax. Also, a feature to download a JSON based model metadata exchange Prototxt file.

Though internal to the programming effort and code design, code and programming framework milestones have numerous long ranging side-effects in terms of agility of layering on features and extensibility. Currently, these include a solid foundation on AngularJS based framework for user interface delivery, and user interaction choreography. Further, the wide use of TypeScript programming language with Jasmine testing framework for unit tests and end to end tests within this effort has facilitated a continuous testing pipeline.

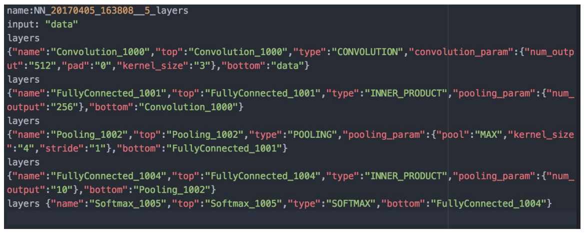
The latest stable release of this application with the above features is made available at the project github pages.

Here is a quick demo of the current user interaction and capabilities. A few screenshots are attached here that showcase some of the development and usages:



```
import torch
import numpy as np
def tryThis():
a=[]
return
```

Figure 1: A simple feed-forward neural network with convolution



```
name:NN_20170405_163808__5_layers
input: "data"
layers
{"name":"Convolution_1000","top":"Convolution_1000","type":"CONVOLUTION","convolution_param":{"num_outp
ut":"512","pad":"0","kernel_size":"3"},"bottom":"data"}
layers
{"name":"FullyConnected_1001","top":"FullyConnected_1001","type":"INNER_PRODUCT","pooling_param":{"num_
output":"256"},"bottom":"Convolution_1000"}
layers
{"name":"Pooling_1002","top":"Pooling_1002","type":"POOLING","pooling_param":{"pool":"MAX","kernel_size
":"4","stride":"1"},"bottom":"FullyConnected_1001"}
layers
{"name":"FullyConnected_1004","top":"FullyConnected_1004","type":"INNER_PRODUCT","pooling_param":{"num_
output":"10"},"bottom":"Pooling_1002"}
layers {"name":"Softmax_1005","top":"Softmax_1005","type":"SOFTMAX","bottom":"FullyConnected_1004"}
```

Figure 2: A sample Prototxt file generated by the builder

Upcoming milestones

In the next few releases we are targeting to enable code generation in PyTorch for feed-forward networks.

Another target that we have identified is the incorporation of well-known public datasets such as MNIST and CIFAR-10 as input choices so that we could recommend the researcher a few starter models.