‘Searching for Exotic Particles in High-Energy Physics with Deep Learning’ Port to Tensorflow/Keras

Daniel Byrne

11/20/2019

## Introduction

In 2014 p. Baldi et al. published a paper investigating the feasibility of using deep learning models to classifiy the by-products of high energy particle collisions over the then traditional method of “shallow” classifiers popular at the time. Their paper focused on discriminating the Higgs Boson and the related Super Symmertric Bosons through their decay, energy, and momentum measurements collected by the Large Hadron Collider’s detectors to identify specific partical collision cascading signatures.

Their reserach ultimately determined that deep learning models out performed traditional shallow methods, even while excluding the hand engineered features developed by physicists to help their shallow classifieers perform more robustly. The original research used common practices at the time to devise the architecture of and to train a deep neural network. They implemented their code in pylearn2, a python deep learning library.[1,3]

In this paper I focus on investigating their code in the context of more modern approaches to deep learning and porting the code to a more common framework at the time of this writing, Keras with a Tensorflow backend.

### Keras

Keras is a high-level neural networks API written in Python and capable of running on top of TensorFlow or other backends. It was developed to prmote fast experimentation. A complete but complex model can be implemented in just a few lines of code, but full customization of the layer by layer programming approach is also supported.

### Tensorflow

TensorFlow is an open source platform for machine learning. It has a comprehensive library, high level APIs like Keras, strong community support, and state-of-the-art pre-programmed Machine Learning, ML, algorithms for researchers and developers to easily iterate on experimental models and to deploy ML powered applications.

### PlaidML

PlaidML is deep learning compiler for compiling, training, and running deep learning models on a wide variety of laptops, embedded devices, or other devices where the available computing hardware is not traditionally well supported in the deep learning frameworks or the available software contains restrictive licenses.[2](https://github.com/plaidml/plaidml)

PlaidML runs as a backend to Keras, and can accelerate training with autonomously generated Tile code. It is compatible with GPUS and CPUS regardless of the underlying architecture, and it doesn’t require use of CUDA/cuDNN enabled Nvidia hardware, but achieves comparable performance. As such this code can be run variously on MacoS with AMD graphics cards, on custom deep learning rigs with NVIdia GPUS or simply on any CPU or OpenCL compatible GPUs regardless of operating system.

### Dataset

The first 21 features (columns 2-22) are properties measured by the Large Hadron Collider, LHC, detectors. The last seven features are functions of the first 21 features devised by domain expets to help discriminate between the two classes.

## Results

The port was implemented in python as a command line program. The defaults of the program are the original hyperparameters devised by the authors of the original paper. However, these defaults can be changed from the command line by simply including an optional flag followed by the modified parameter’s value

$ python higgs\_original\_tf\_port.py -b 1000 -e 1 -f HIGGS\_22e5.csv -p 1 -a relu

Help can be invoked from the command line by invoking the -h switch.

$ python higgs\_original\_tf\_port.py -h  
usage: higgs\_original\_tf\_port.py [-h] [-f FILEPATH] [-l LR] [-m MOMENTUM]  
 [-d DECAY] [-tt TTSPLIT] [-e EPOCHS]  
 [-b BATCH\_SIZE] [-ft FEATURE\_TYPE]  
 [-p PRETRAIN\_EPOCHS] [-a ACTIVATION]  
  
Builds Higgs Boson Classifier model in Keras with a tensorflow backend on  
PlaidML.  
  
optional arguments:  
 -h, --help show this help message and exit  
 -f FILEPATH, --filepath FILEPATH  
 Filename of training and test data. Original dataset  
 can be obtained from  
 https://archive.ics.uci.edu/ml/datasets/HIGGS.  
 -l LR, --learningrate LR  
 Sets the initial learning rate. Default = .05  
 -m MOMENTUM, --momentum MOMENTUM  
 Sets the initial momentum. Default = .9.  
 -d DECAY, --decay DECAY  
 Sets the learning rate decay rate. Default = 1e-6.  
 -tt TTSPLIT, --ttsplit TTSPLIT  
 Train test split percentage. Default = .045.  
 -e EPOCHS, --epochs EPOCHS  
 Sets the # of epochs. Default = 200.  
 -b BATCH\_SIZE, --batch\_size BATCH\_SIZE  
 Sets the batch size. Default = 100.  
 -ft FEATURE\_TYPE, --feature\_type FEATURE\_TYPE  
 raw - Selects the 22 unadultered features in the  
 dataset. engineered- selects only the hand engineered  
 features. all - Selects the entire set of features.  
 Default 'raw'  
 -p PRETRAIN\_EPOCHS, --pretrain-epochs PRETRAIN\_EPOCHS  
 Number of epochs to pretrain each lahyer on when  
 adding that layer to the complete model. Default = 5  
 -a ACTIVATION, --activation ACTIVATION  
 Sets the activation of each layer not including the  
 final classification layer. Default = tanh.

### Suggested Improvements

THe field of Deep Learning is experiencing an explosive growth curve fueled by increasingly sophisticated and capable hardware and emerging business applications in nearly every field generating real world returns on investments superior to methods deep learning is replacing.

Consequently, there are a number of potential improvements to the original model suggested simply by the performant models proven and establised in research and business settings today. A list of potential improvements to the original model in no particular order are as follows.

* Relu activations to improve training and convergence
* Normalizing the data or using batch normalizing to improve convergence
* Pruning the neural network to remove redundant and non-informative nodes resultingh likelying in a compressed model and improved training times.
* Possibly using a CNN architecture like Resnet to improve the classification performance.
* Promoting localized features maps through gating the inputs or using a variation of dropout to promote localized preferences.
* Increasing the number of layers to extract more features

These are just some of the many hundreds of DNN optimizations that have arisen since the publication of the original paper. I belive it is a worthwhile effort to continuously update this basic model to improve upon the training time, model size, and classificaton performance as a way to evaluate and score new DNN techniques as they are developed.

### Model Features

I attempted to recreate faithfully the architecture of the original paper.

* The model pretrains each layer as in the original paoper to improve convergence.
* Model uses tanh by default, but training speed can be improved by using relu
* The data is not normalized.
* The learning rate decays according to the rate established in the original paper.
* Momentum is not however dynamically altered as in the original paper. This feature is under development.

### Splitcsv

The original dataset consisted of 11 million records. To facilitate fast prototyping, the data was split using the python script splitcsv.py into configurable integer fractional subsets of the original data. Splitcsv can be used to faccilitate this splitting. The following command splits the dataset into 5 equal sized subsets.

$ python splitcsv.py HIGGS.csv 5

## References

## Notes

The higgs\_enhanced\_tf\_port.py file is a work in progress.

## Code Listing

The latest code will be maintained on github at <https://github.com/realdanielbyrne/sussy_higgs_tf_keras_plaidml>