Machine Learning with TensorFlow as an alternative to TMVA

Aidan Dang, Martin Sevior
ARC Center of excellence for Particle Physics at the TeraScale
School of Physics,
University of Melbourne

TensorFlow Introduction

- Open source machine learning library from Google
  - Used by Google, for example, in search ranking and image classification
- Python API which interfaces closely with NumPy
- Easily scalable to distributed, heterogeneous hardware architectures
- CUDA acceleration
- Mainly used for neural network machine learning approaches
  - Automatic differentiation
- Useful for other applications where GPU acceleration on n-dimensional arrays is important

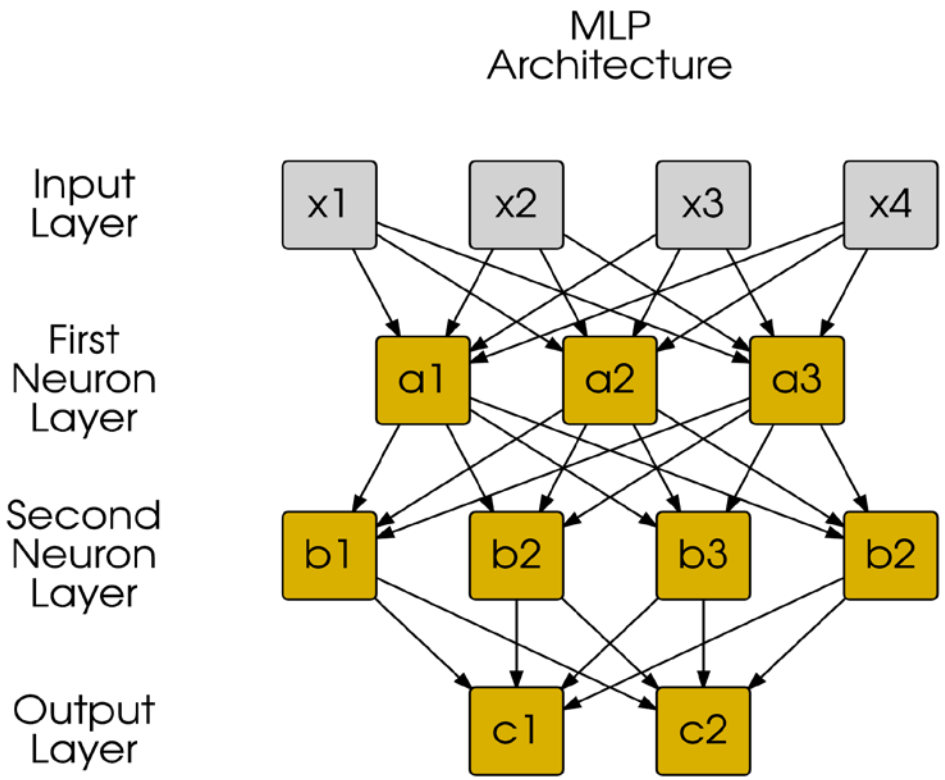
A TensorFlow Workflow for ROOT

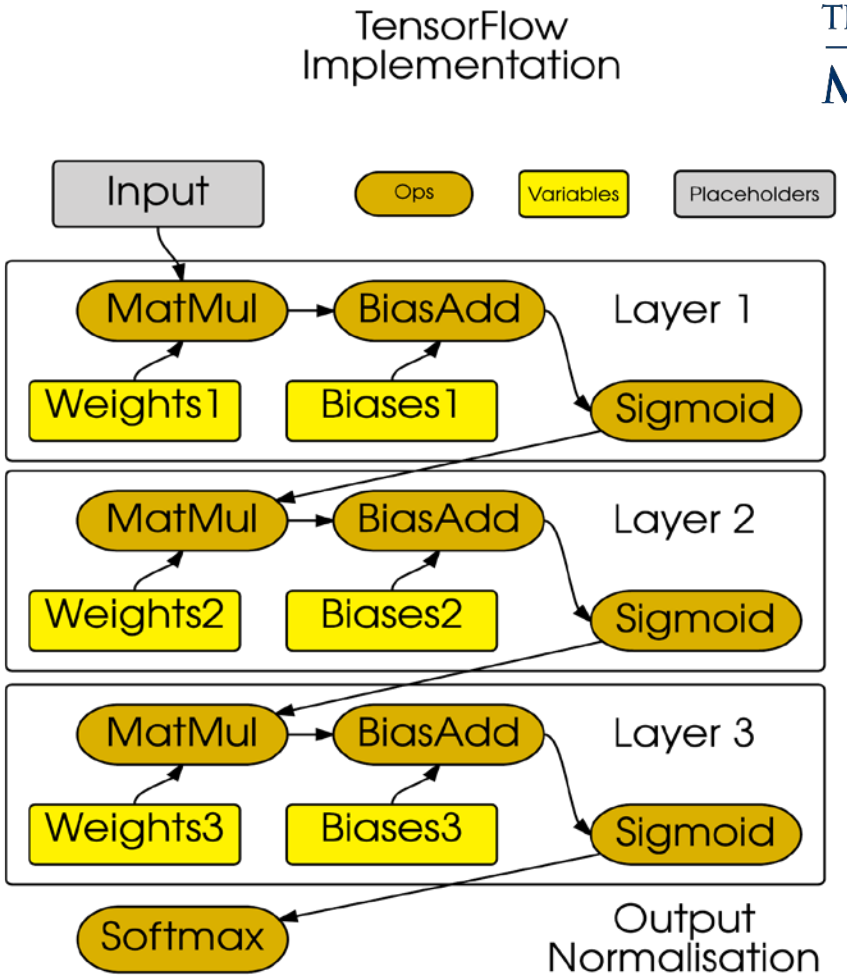
- Required Python libraries: NumPy, root_numpy, and of course, TensorFlow
- Using root_numpy, load data from *.root file into a NumPy array
- Perform any extra pre-processing in NumPy
- For training:
  - Build model in TensorFlow
  - Feed slices of NumPy array into model (batching)
  - Optimise loss (sets weights and biases)
- For classification with a trained model:
  - Feed entire NumPy array into model
  - Append TensorFlow output as new column
  - Use root_numpy to convert back to *.root file

General Approach to Machine Learning with TensorFlow

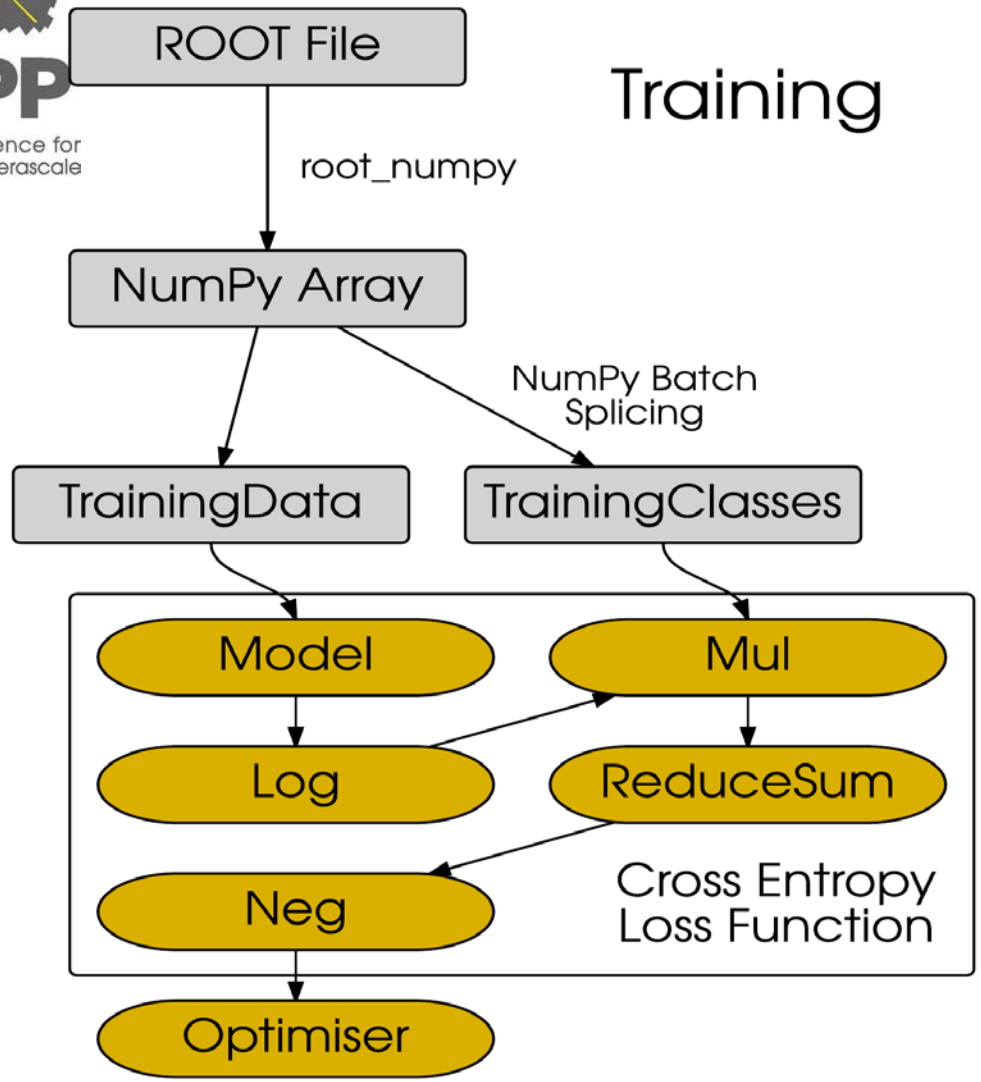
- We'll look at recreating the multi-layer perceptron in TensorFlow
- Build the model from 'tensors' and 'ops'
  - Tensors are the n-dimensional storage arrays. The main types of tensors are placeholder and variable tensors
  - Ops are the operations on tensors (matrix multiplication, addition)
- For each layer in the MLP, we have one set of multiplication with weights, addition of biases and a non-linear activation function
- The output of this layer is fed to another layer as per the MLP, then renormalised at the end
- A loss function showing how far our model is from the actual classification is built
- Run the loss through one of TensorFlow's optimisers, which will update weights and biases
- Feed in data

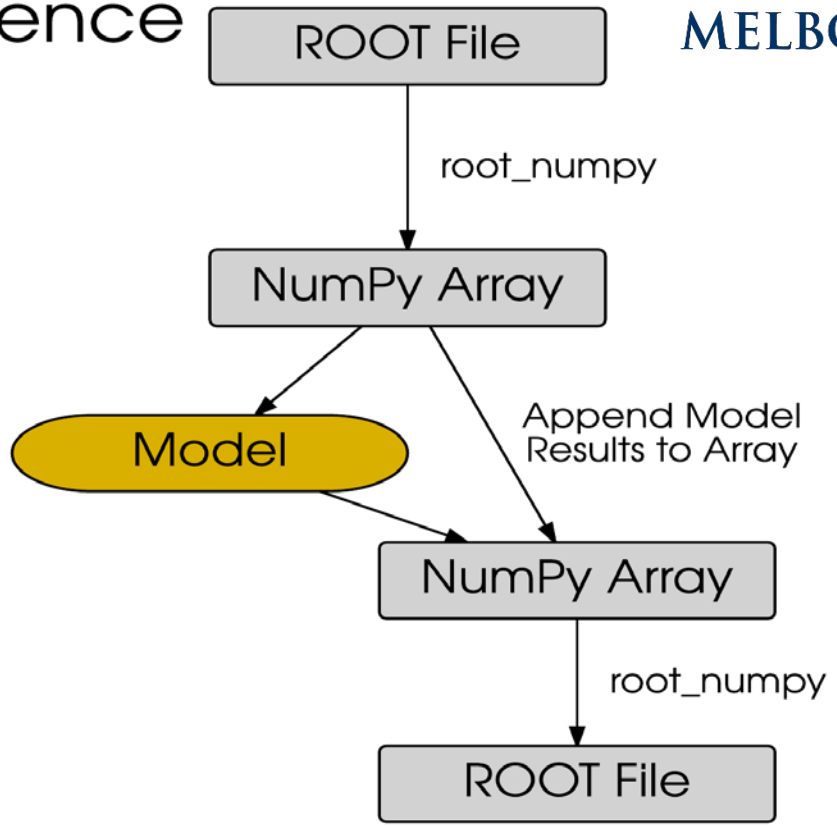
Implementing Multilayer Perceptron (MLP) in Tensor Flow





A TensorFlow Workflow for ROOT

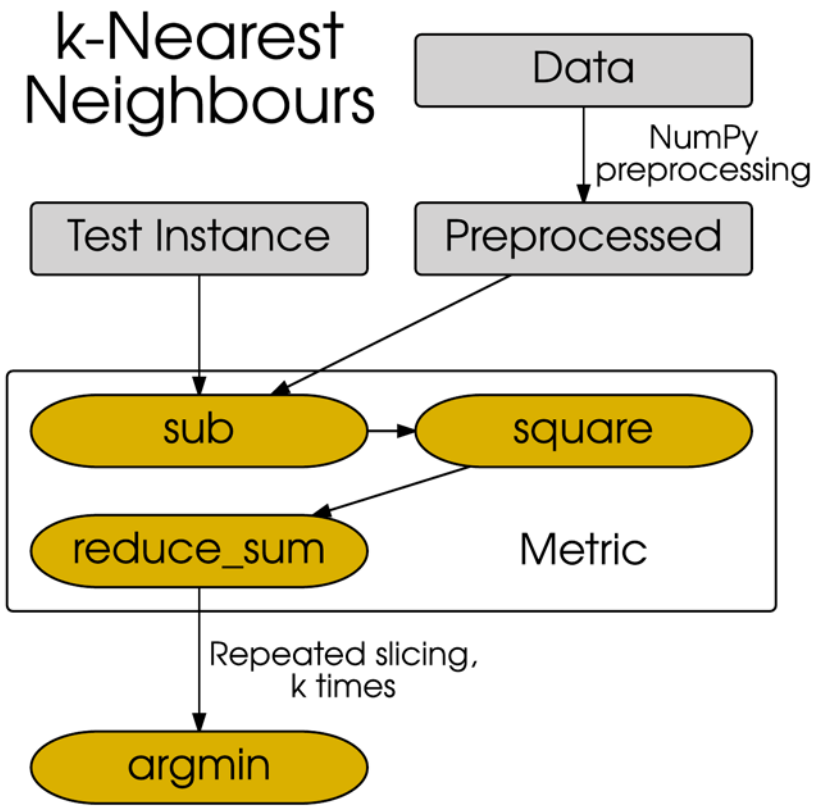




Other Classifiers in TensorFlow

k-Nearest Neighbours

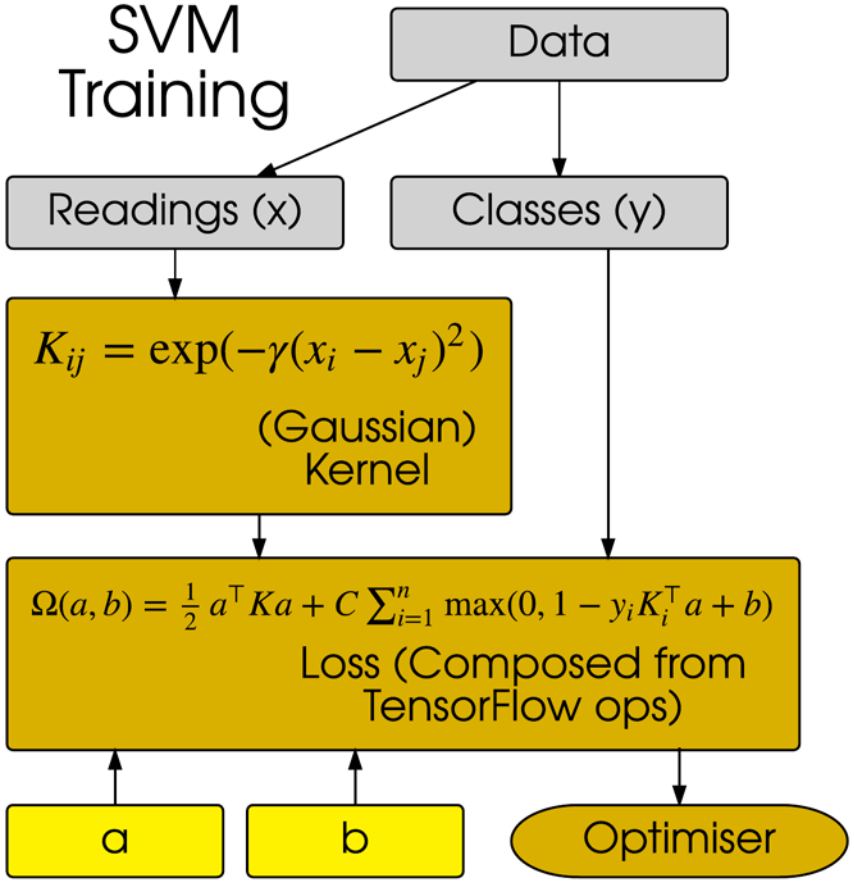
- Classification of a measurement based on the k closest training examples.
- Preprocessing is performed in NumPy:
  - Randomly selects training examples to have the same number per class
  - Scales measurements in each dimension for equal spread.
- Requires a metric to determine the distance between two points.
- TensorFlow is used to find the k training examples closest to the test instance using this metric.
- Classify based on classes of these k examples.

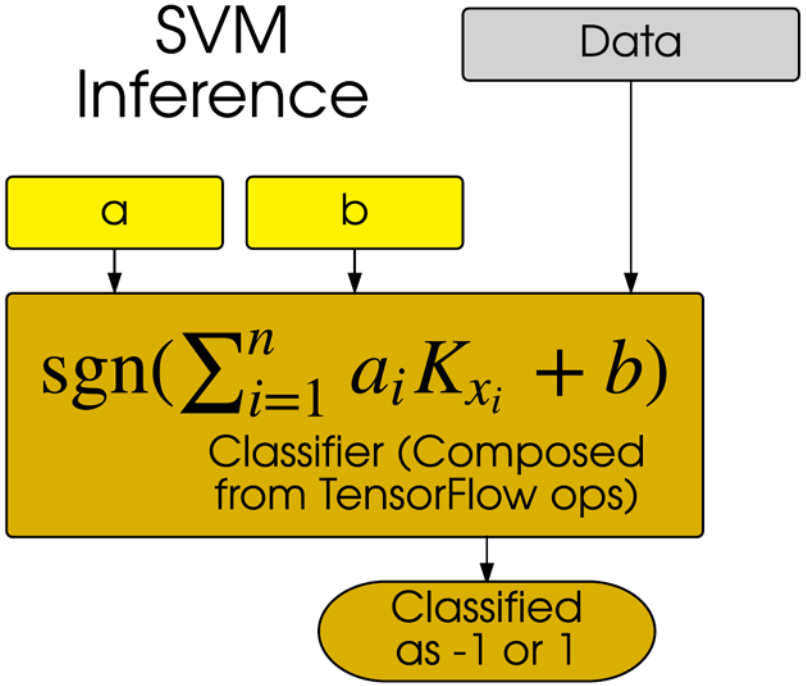


Support Vector Machine

- Binary classification based on a hyperplane which maximally separates examples of the two classes.
- Softness (for non-separable sets) and kernel transformations (for non-linear cuts) can be chosen.
- An optimisation problem:
  - Define the loss function as a composition of ops in TensorFlow.
  - Apply an optimiser on this loss function to find the gradients and offset of the hyperplane
- Classify test points based on which side of the hyperplane they lie.

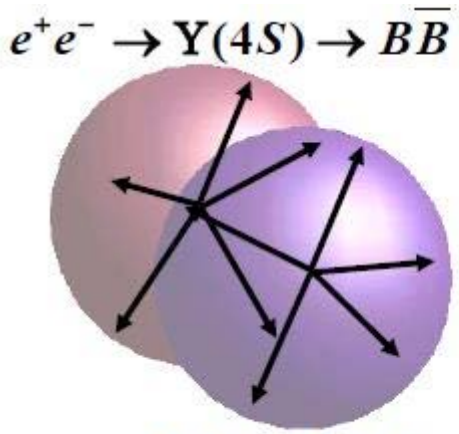
Support Vector Machine



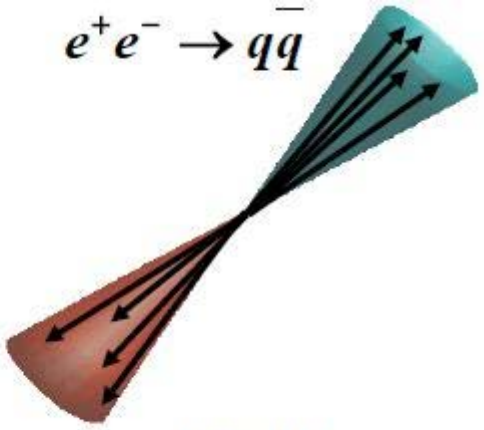


Can we make it work for Particle Physics?

Easy to use examples for standard computer science problems
What about particle physics?
Example from B-Physics analysis: $B^0 \to \pi^0\pi^0$
BR~10^{-6}, large Background from $B^0 \to q\bar{q}$
Mostly use Event Shape Topology to reduce background

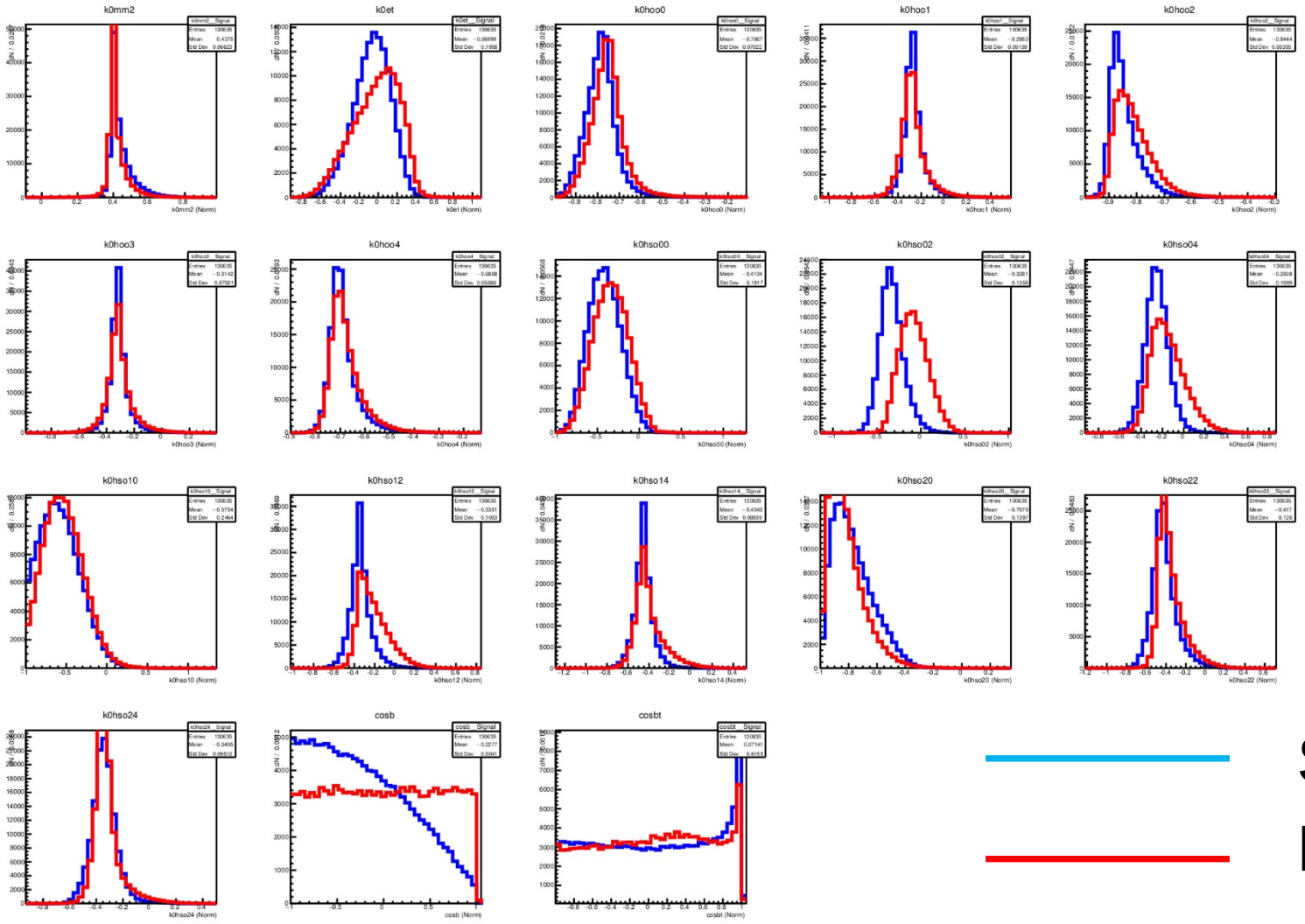


Spherical



Jet-like

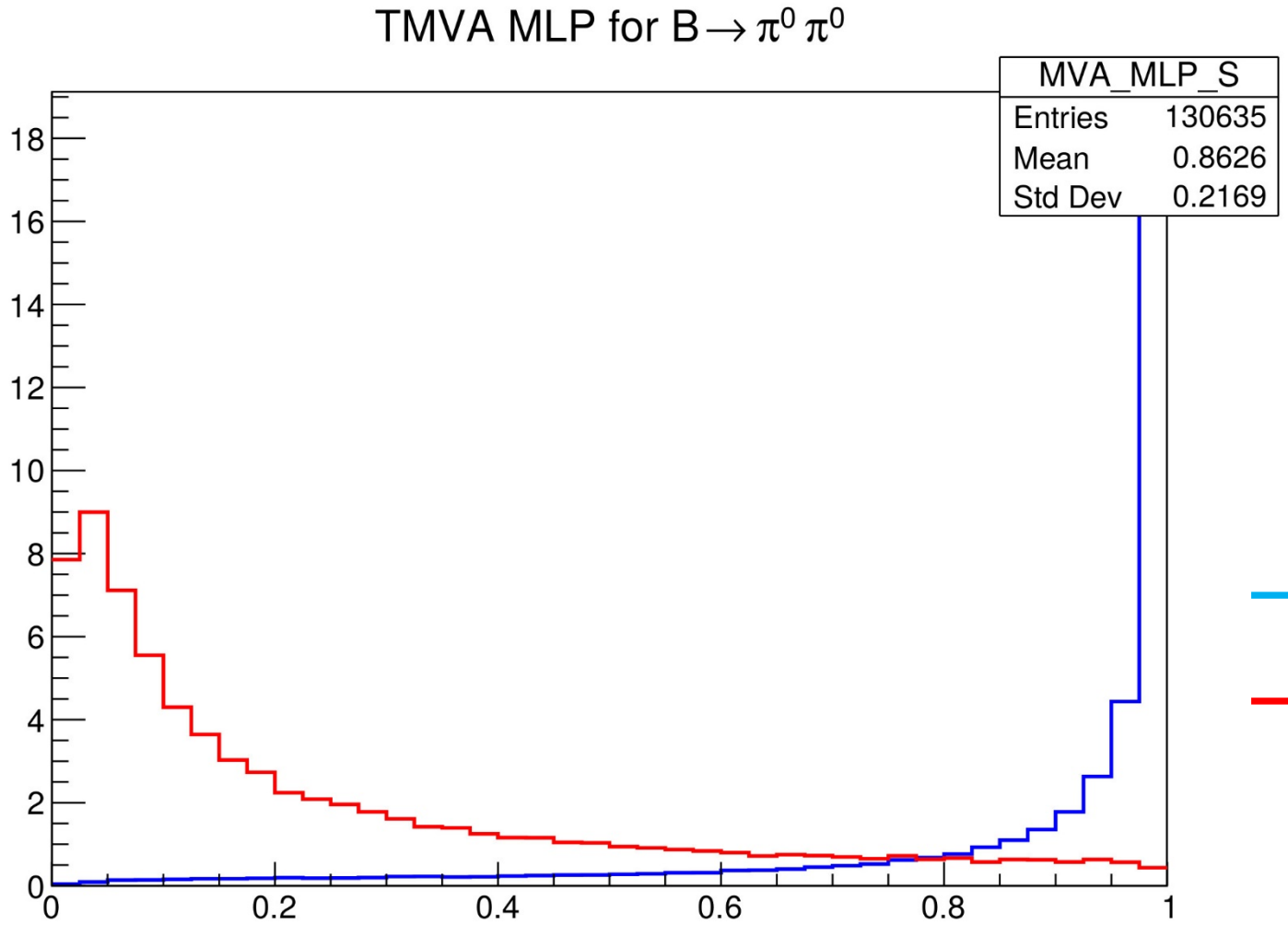
Continuum Fighting Variables



Signal

Background

TMVA MLP (Default settings)

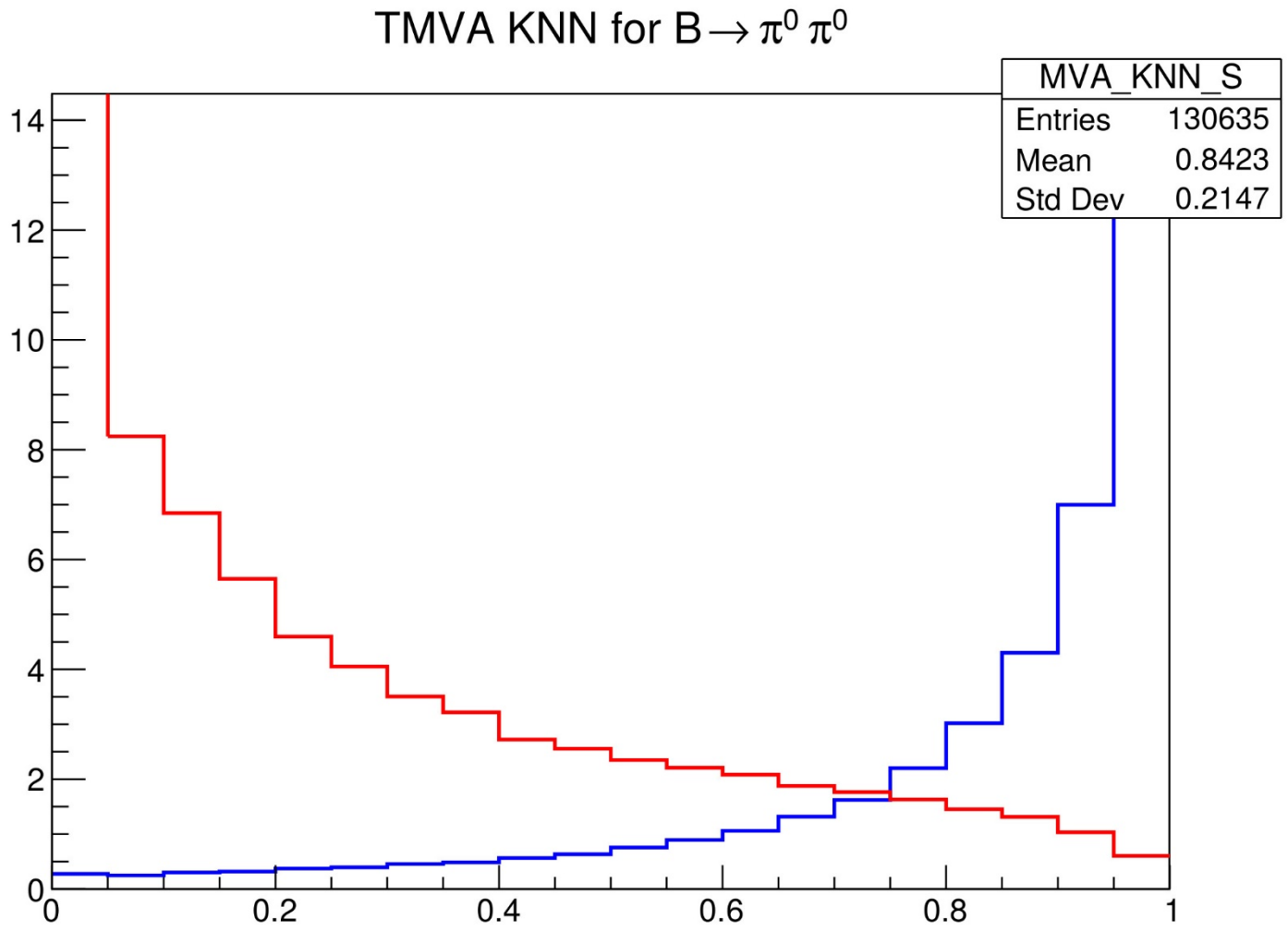


2 hours of processing on my laptop

Signal

Background

TMVA KNN (Default settings)

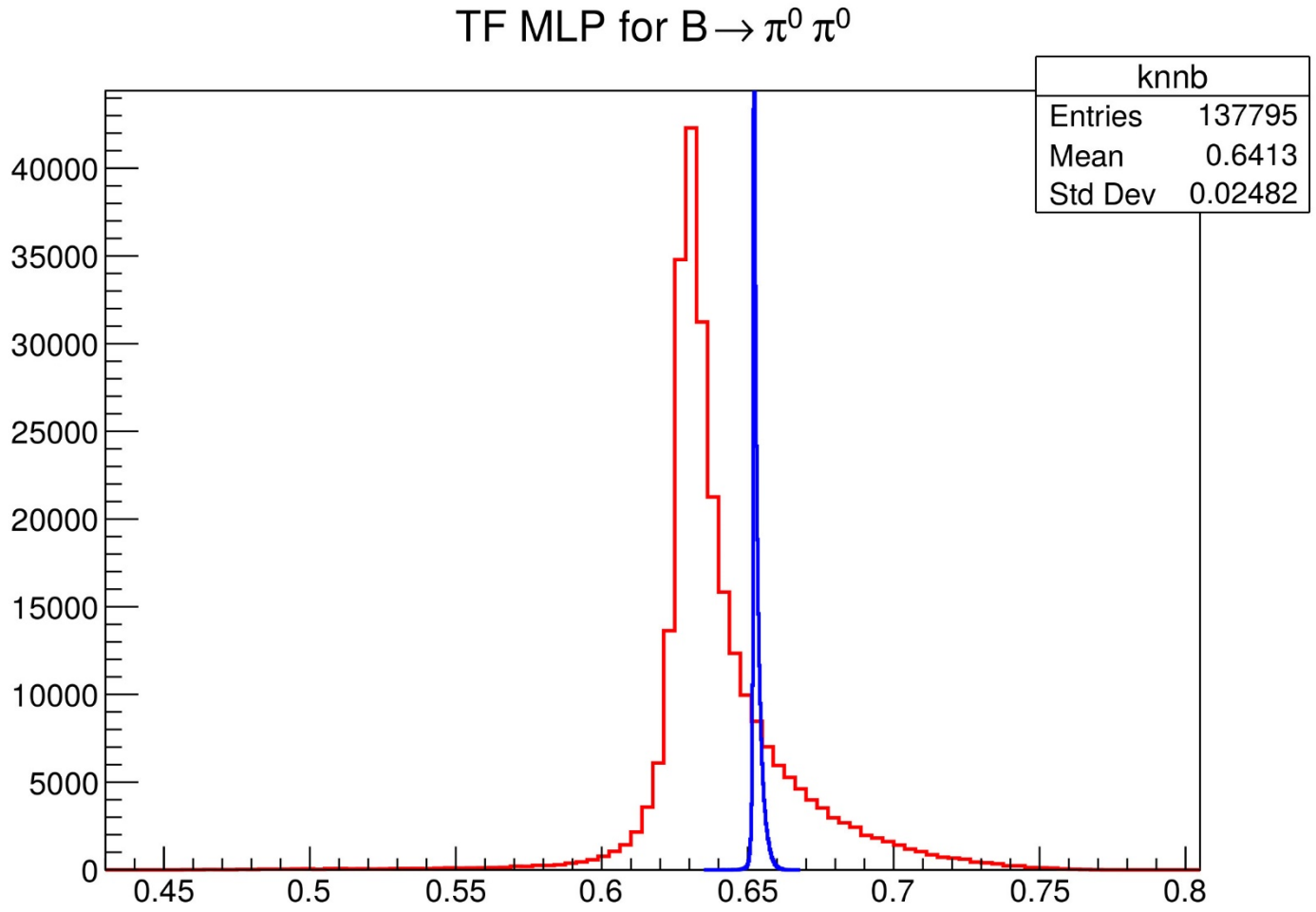


5 hours of processing on my laptop

Signal

Background

MLP with Tensorflow



- Cost = MSE
- 2 Hidden layers
- Neuron=tanh
- Clearly not optimized!
- Far easier to use TMVA

Signal

Background

Summary

- TMVA is nicely designed for Particle Physics
- Tensorflow is a powerful general purpose ML toolkit
- Very expressive
- MLP, KNN and SVM all written in less than 100 lines of python each
- BUT….
- Generalization comes with a big learning curve
- Direct interface to root is awkward using the numpy approach