Application of Neural Networks to b-jet detection

Stephen Poprocki
REU 2005
Department of Physics, The College of Wooster, Wooster, Ohio 44691

Advisors: Gustaaf Brooijmans, Andy Haas
Nevis Laboratories, Irvington, NY 10533

July 8, 2005

Abstract

A multilayer perceptron (MLP) neural network (NN) was trained with monte carlo data to detect b-jets. A variety of NNs were tested to maximize performance. The best NN was run on data with different reconstruction options. It was found that a simple MLP NN with 6 variables and 6 hidden neurons performed better than using a decay length significance cut for detecting b-jets.

1 Purity & Efficiency

The MC data we use are taggable jets ($\Delta R < 1.0$). Further, we only use MC data with at least 1 vertex and assume 0 vertex jets are non-b-jets. To correct for these 0 vertex jets, we use the following formulas:

$$\text{efficiency} = \frac{\#\text{correctly tagged b-jets}_{\text{nv}>0}}{\#\text{b-jets}_{\text{nv}>0} + \#\text{b-jets}_{\text{nv}=0}},$$

$$\text{purity} = \frac{\#\text{incorrectly tagged non-b-jets}_{\text{nv}>0}}{\#\text{non-b-jets}_{\text{nv}>0} + \#\text{non-b-jets}_{\text{nv}=0}}.$$

The purity versus efficiency plots are obtained by varying a cut on the NN output.

2 Neural Networks

The NNs were implemented in ROOT 4.04/02 with the `TMultiLayerPerceptron` class. A MLP is a simple feed-forward network. Figure 1 shows a simple model of a MLP NN. Each neuron is connected to all neurons on the layer below it. We utilize only one output

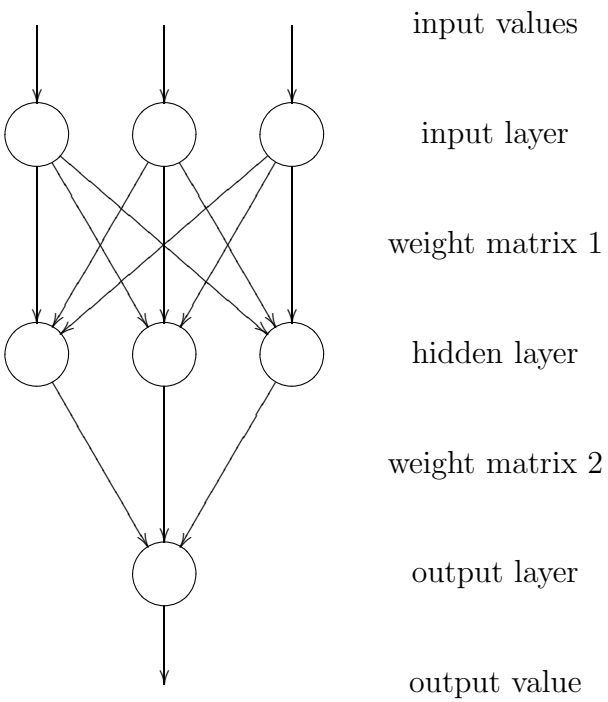

Figure 1: Simple multilayer perceptron neural network.

neuron, namely, the probability of the jet having a b. The output of a neuron is a linear combination of the previous neurons and their weights. Typically the sigmoid function is used to normalize the output between 0 and 1. Multiple hidden layers are possible but typically only complicate matters.

2.1 Training

Training is the process of minimizing the error of the NN output. Out of a total of 816 684 jets, half were used for training and the other half for testing. The `TMultiLayerPerceptron` class implements six different training methods:

1. Stochastic minimization
2. Steepest descent with fixed step size (batch learning)
3. Steepest descent algorithm
4. Conjugate gradients with the Polak-Ribiere updating formula
5. Conjugate gradients with the Fletcher-Reeves updating formula
6. Broyden, Fletcher, Goldfarb, Shanno (BFGS) method

Figure 2 shows the errors as a function of the epoch. Figure 3 on page 6 shows the signal and background outputs of the NN for the different training methods. Figure 4 on page 7 shows the purity versus efficiency for the different training methods.

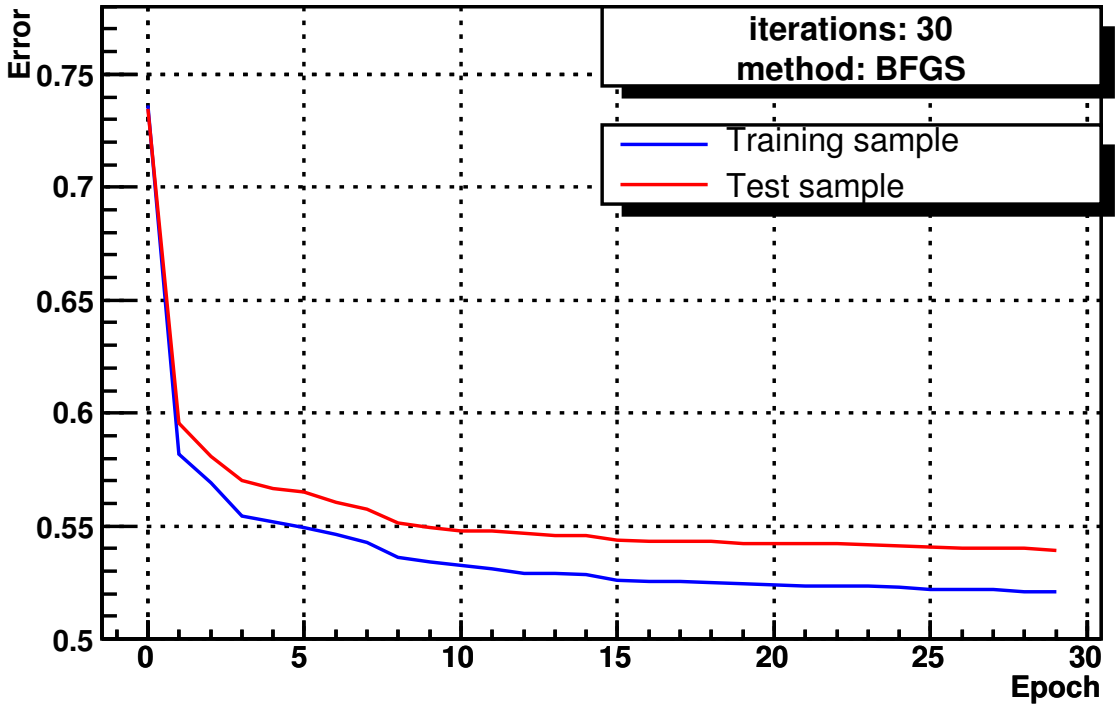

Figure 2: Training error for various training methods as a function of number of epochs.

The training methods adjust the weights based on the error in the output. One weight update is called an epoch, cycle, or iteration. In principle, too few epochs results in an under-trained NN and too many epochs results in an over-trained NN. An over-trained NN is not desirable because it has learned features that are specific to the training data.

2.2 Variables

Initially, the NNs were trained with a variety of combinations of variables. Then two variable sets were chosen for comparison:

"simple":

- 3d decay length significance (sdls3d)
- decay rate (dr)
- mass
- 2d decay length (dl2d)
- chi square (chi2)
- multiplicity (mult)

"fancy": All of "simple" plus:

- number of vertices (nv)
- secondary vertex:
  - 3d decay length significance (sdls3d2)
  - decay rate (dr2)
  - mass (mass2)
  - chi square (chi22)
  - multiplicity (mult2)

The purpose is to see if adding the secondary vertex variables yields an improvement. Figures 5 on page 8 and 6 on page 9 show the result: simple is better.

2.3 Hidden Neurons

The NNs were trained with varying number of hidden neurons. The result was that six hidden neurons for "simple" and 12 hidden neurons for "fancy" were best. That is, the same number of hidden neurons as variables.

3 Reconstruction Options

MC data from a total of five different reconstruction options were compared:

- `cab3`
- `cab_default_sv`
- `cab_default-tj`

- `cab_noadapt`
- `cab_no-tj-merge`

The result is that all but `cab_default_sv` had the same performance, and `cab_default_sv` had poor performance. Figure 7 on page 10 compares their performances.

4 Results

For b-jet detection, NNs provide an improvement over cutting 3d decay length significance (sdls3d). This is because the NN is able to learn the significances of the minor variables. Figure 8 on page 11 compares the NN with the 3d decay length significance cut.

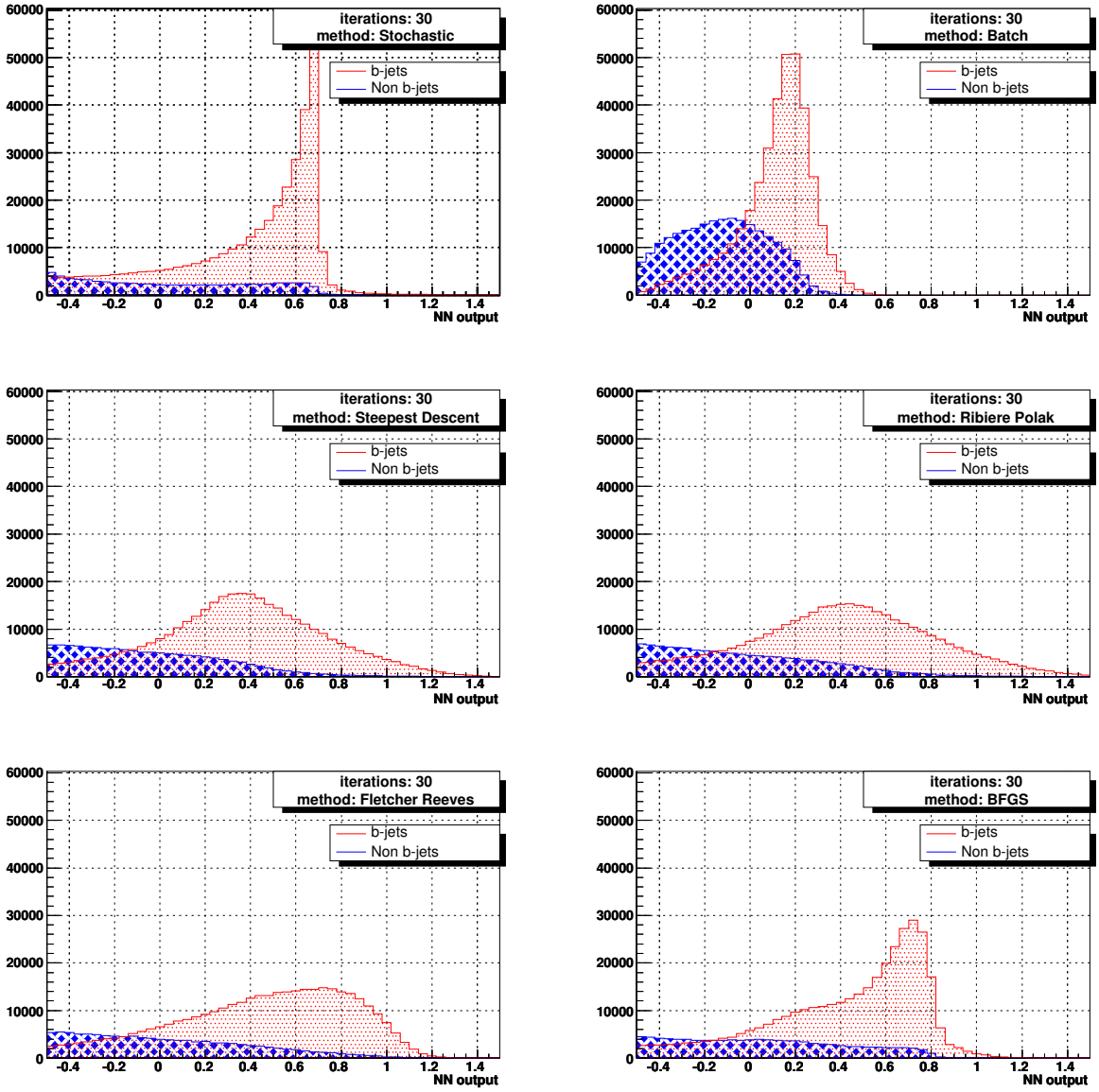

Figure 3: Signal and background outputs of the NN for the different training methods. Note that we only focus on the signal region; there is much more background data out of the bounds of the plot.

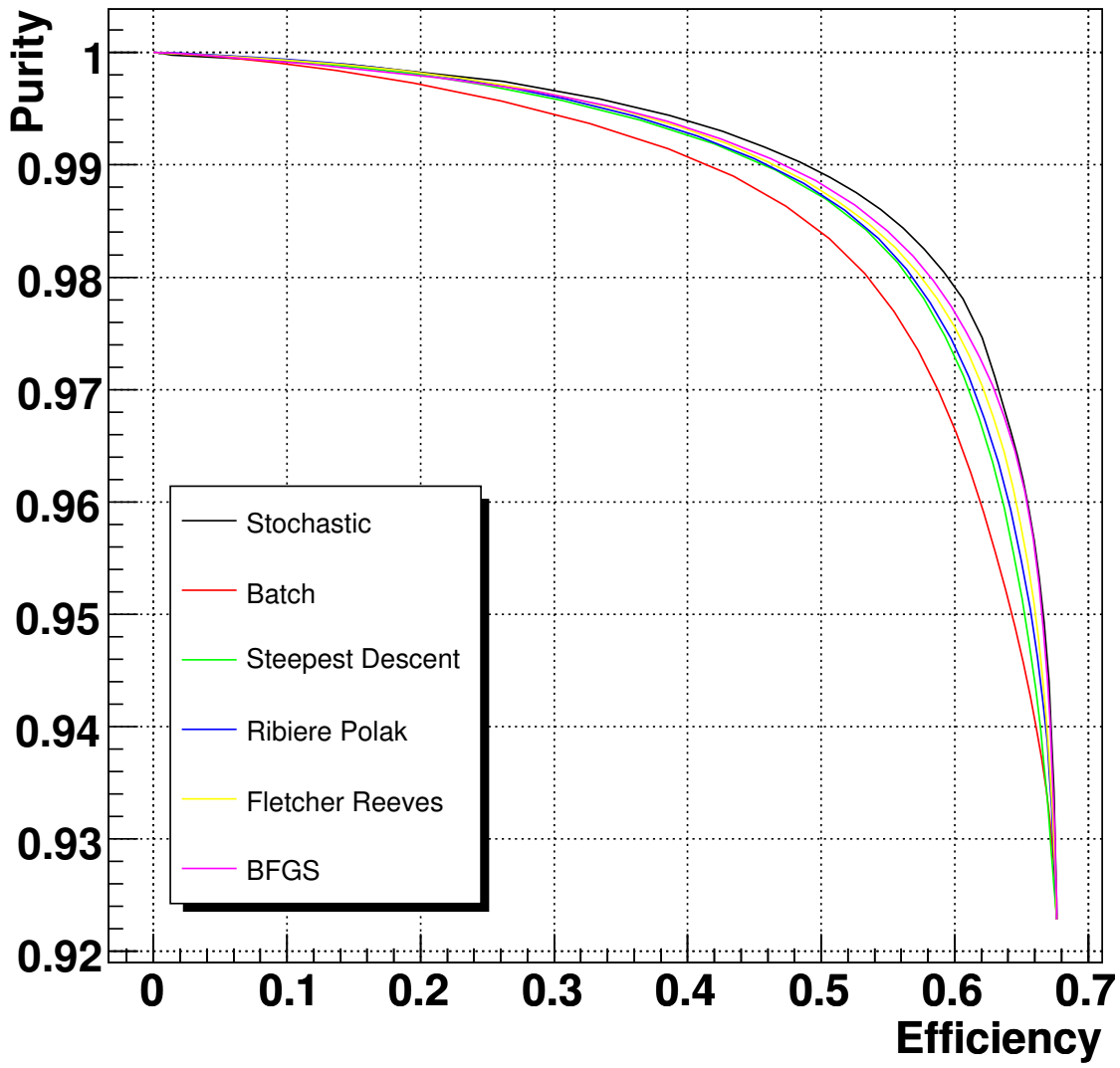

Figure 4: Purity versus efficiency for the different training methods.

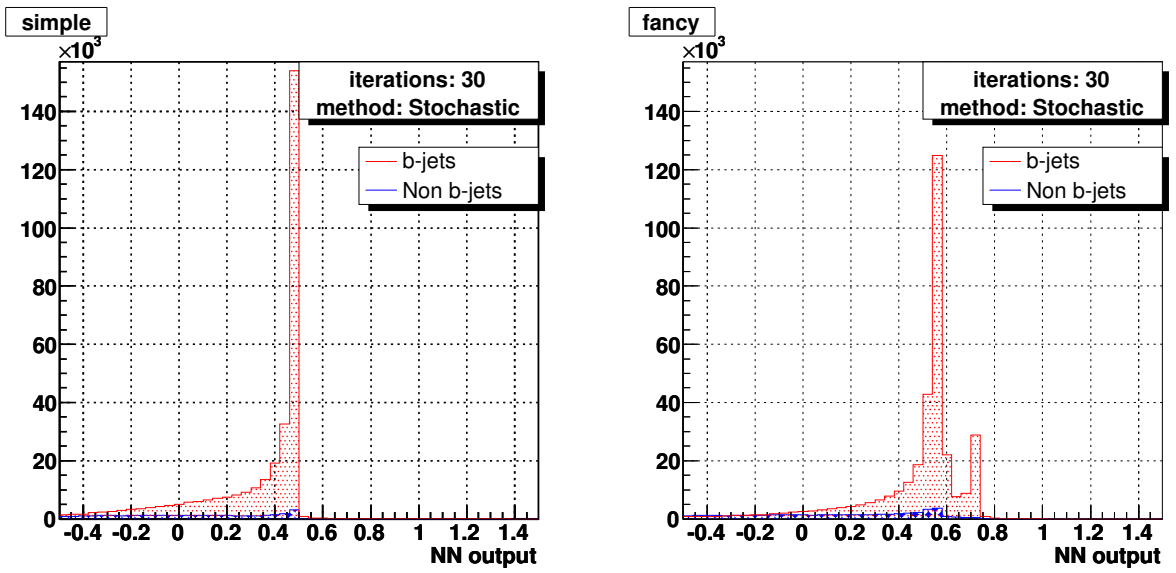

Figure 5: Signal and background outputs of the NN for the different variable sets.

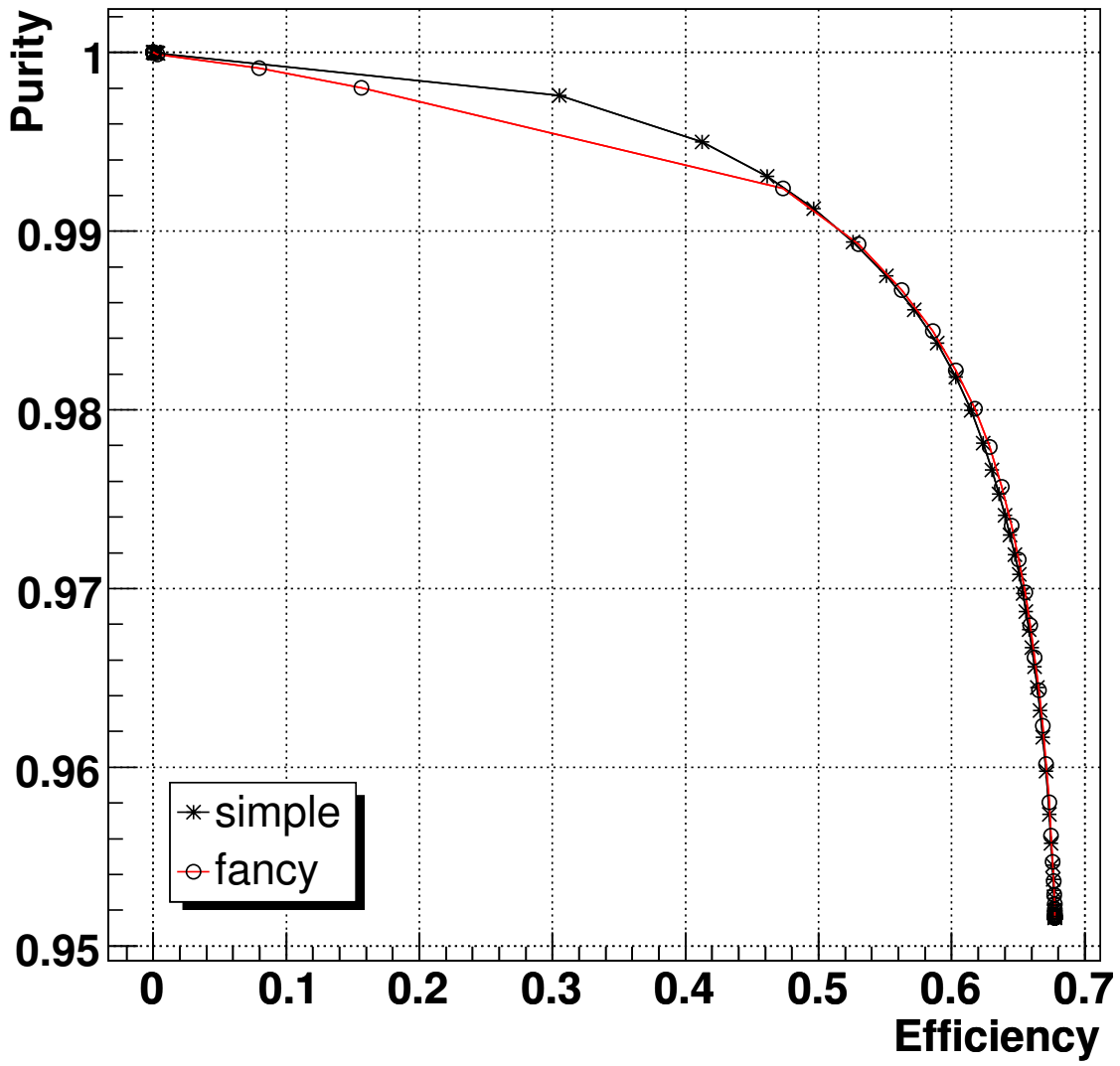

Figure 6: Purity versus efficiency for the different variable sets.

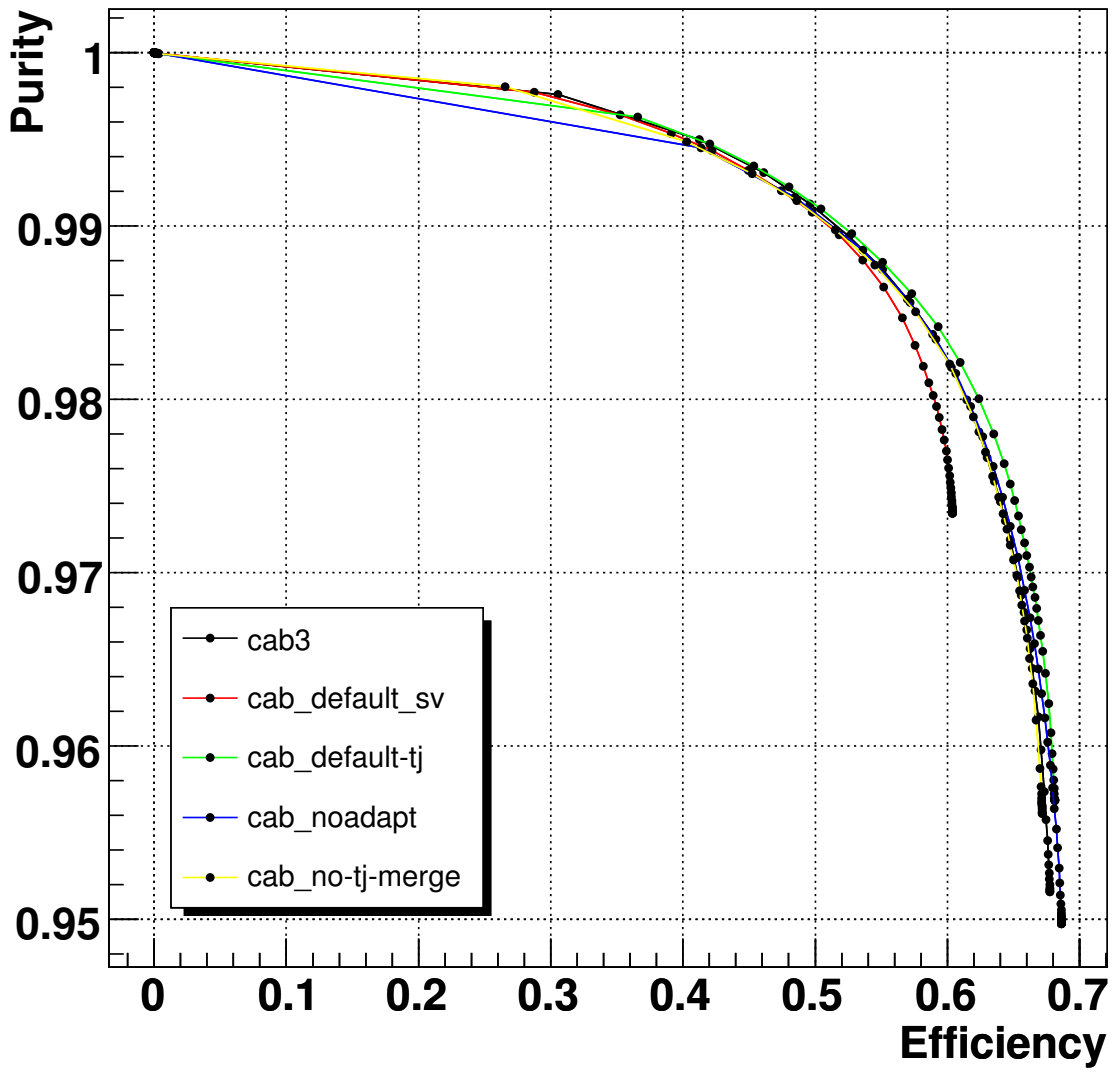

Figure 7: Purity versus efficiency for the different reconstruction options.

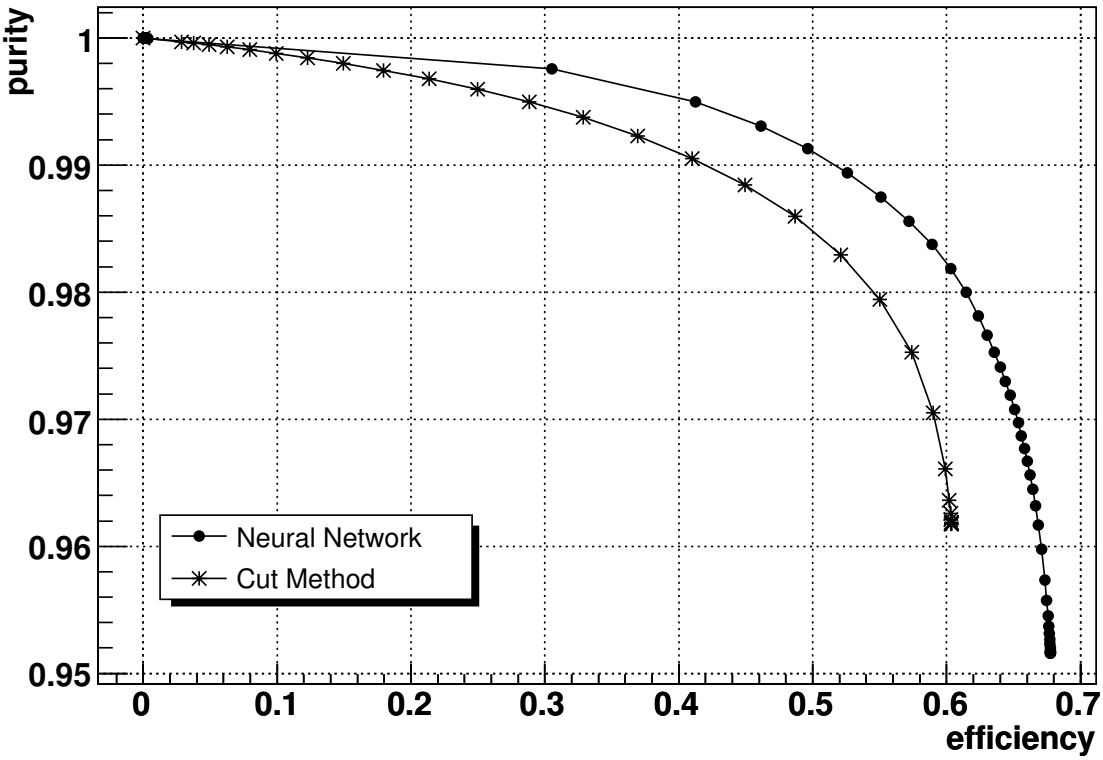

Figure 8: Purity versus efficiency for the neural network and the decay length significance cut method.