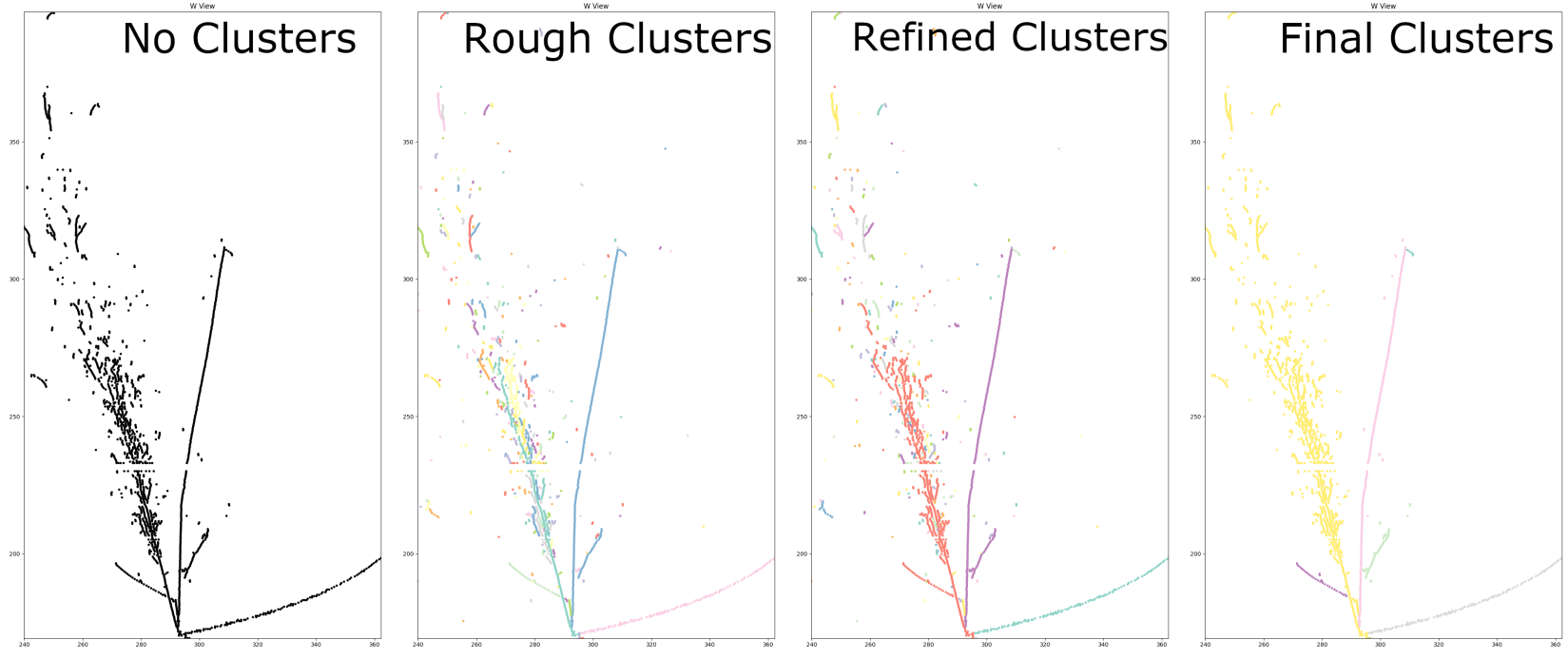
Deep Learning Based Shower Growing

Ryan Cross

Summary

Clustering In Pandora

Here is a very quick overview of some of the stages in the Pandora clustering pipeline. That is, from entirely unclustered hits, through to final PFParticles.



Here, we go from unclustered, to roughly clustered, grow those rough clusters, and finally tidy up and finish the particles. I'm focusing on the middle two steps.

Motivation: Performance and Potential gains

I have performed a basic check of where the ceiling for this specific algorithm lies (i.e. how much can we improve the shower growing, and how much does improving it improve the final output that we all use?), by cheating this algorithm. That is, using MC information on this algorithm alone, to see how the reconstruction improves with "perfect shower growing".

That gave some results like this (full details are available for this in my previous set of slides), where we can see that cheating the shower growing can have a large, visible effect on the final reconstruction outputs.

	Current	**Cheated**
Avg Shower Completness	72.8%	87.0%
Avg Shower Purity	86.8%	87.1%

These values imply that there is scope for improvements in the shower growing, which is where this work comes in.

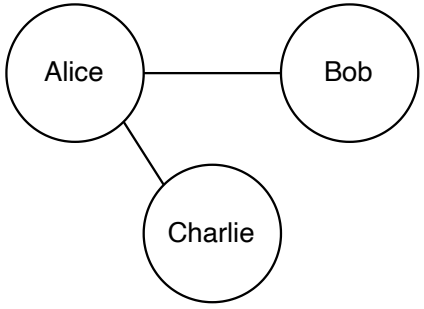
MCC11 10Kt 1x2x6 ν_e, 1000 events

Graph Networks

I am using graph neural networks (GNNs) to improve the shower growing. Graph networks specifically were chosen as they seem to fit the problem at hand fairly well, and easily scale for the differing event sizes.

A graph in this context is a structure made up of nodes which are then connected via edges. These nodes and edges can have arbitrary properties assigned to them.

A simple example would be considering some form of social media. Users would make up the nodes, with properties like name, attached to each node, and edges linking two users if they are friends. A graph network could then be used to generate potential new edges of a given strength, which can be used for a "Recommended Friend" feature. There is a similar usage for "Users who bought this also bought" like features.



Nodes/Points/Vertices are usually used interchangeably.

The Process

Since most Deep Learning based work is done in Python, the workflow is a bit more awkward than normal:

- Run Pandora as normal, but in the Pandora config, set the DL Shower growing to "Training Mode", which outputs CSV files.
- Can now load/process/train over those CSV files in Python land. The end result is a trained GNN, which was trained on the events from the previous step.
- That trained model can be loaded back into Pandora by pointing the DL Shower growing at the file, and changing any required parameters (various thresholds etc.)
- With that, the performance can now be evaluated as normal inside Pandora, as it runs transparently as just another algorithm in the chain.

There is some awkwardness that the final implementation uses multiple passes of the network, which isn't inherently trained for, as well as the efficiency metric being different, but its close enough to be useful.

Results for Neutrino Events

In previous talks, I was working with sample datasets rather than full neutrino events. This was useful from a development point of view, allowing me to add in required features incrementally (iterative running, selecting the best input etc.).

I'm now running over full MCC11 events, which should give a better idea of how the network performs in a real use case. I'm still intending to perform further studies with my test datasets, as I can more easily assess things like angular dependence using those test datasets.

Results for Neutrino Events

In previous talks, I was working with sample datasets rather than full neutrino events. This was useful from a development point of view, allowing me to add in required features incrementally (iterative running, selecting the best input etc.).

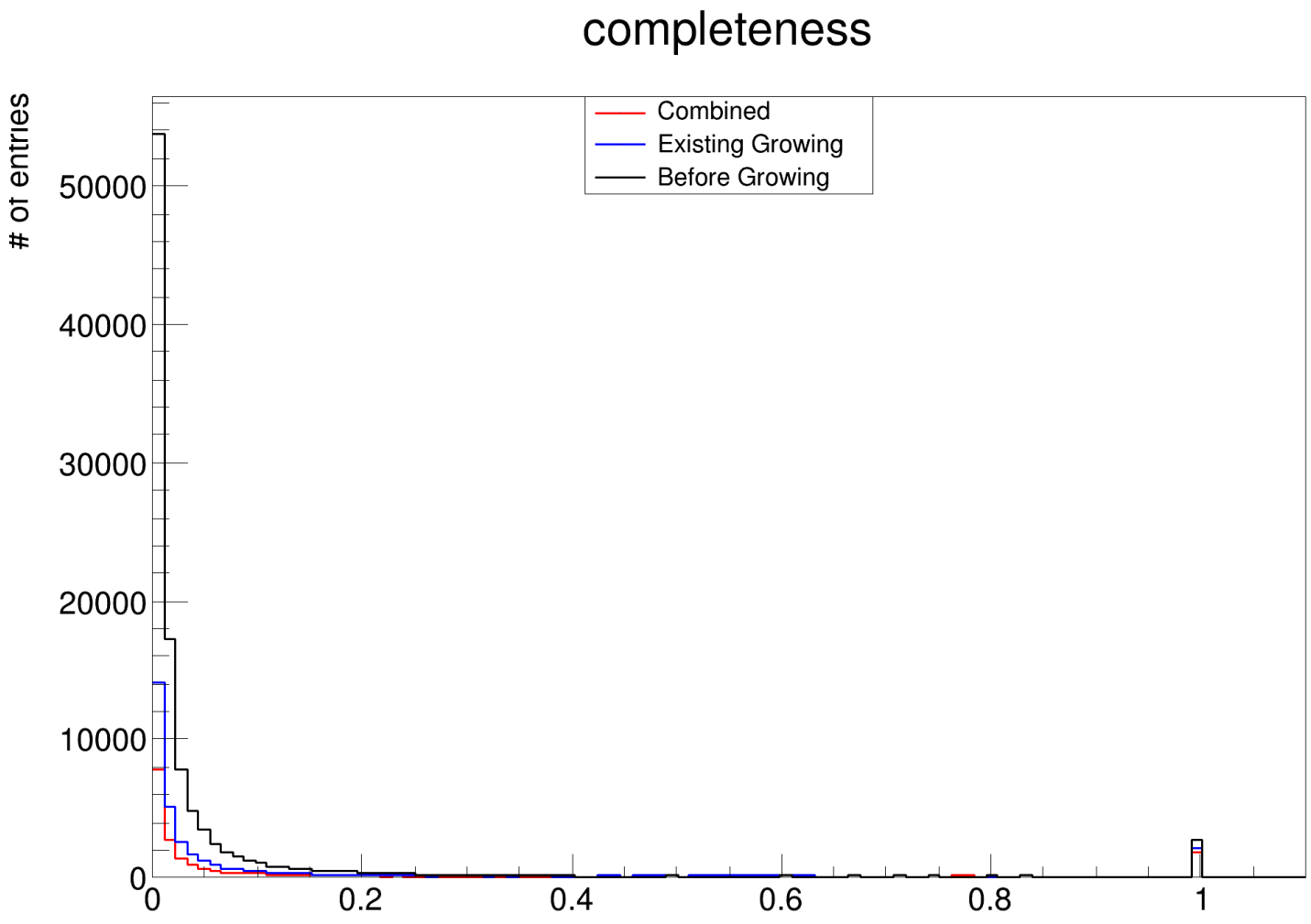
I'm now running over full MCC11 events, which should give a better idea of how the network performs in a real use case. I'm still intending to perform further studies with my test datasets, as I can more easily assess things like angular dependence using those test datasets.

I've split my results into two sections, to help highlight what the algorithm is doing.

First, I'll show directly before and after the shower growing. This gives the best picture of what the shower growing changes are actually doing.

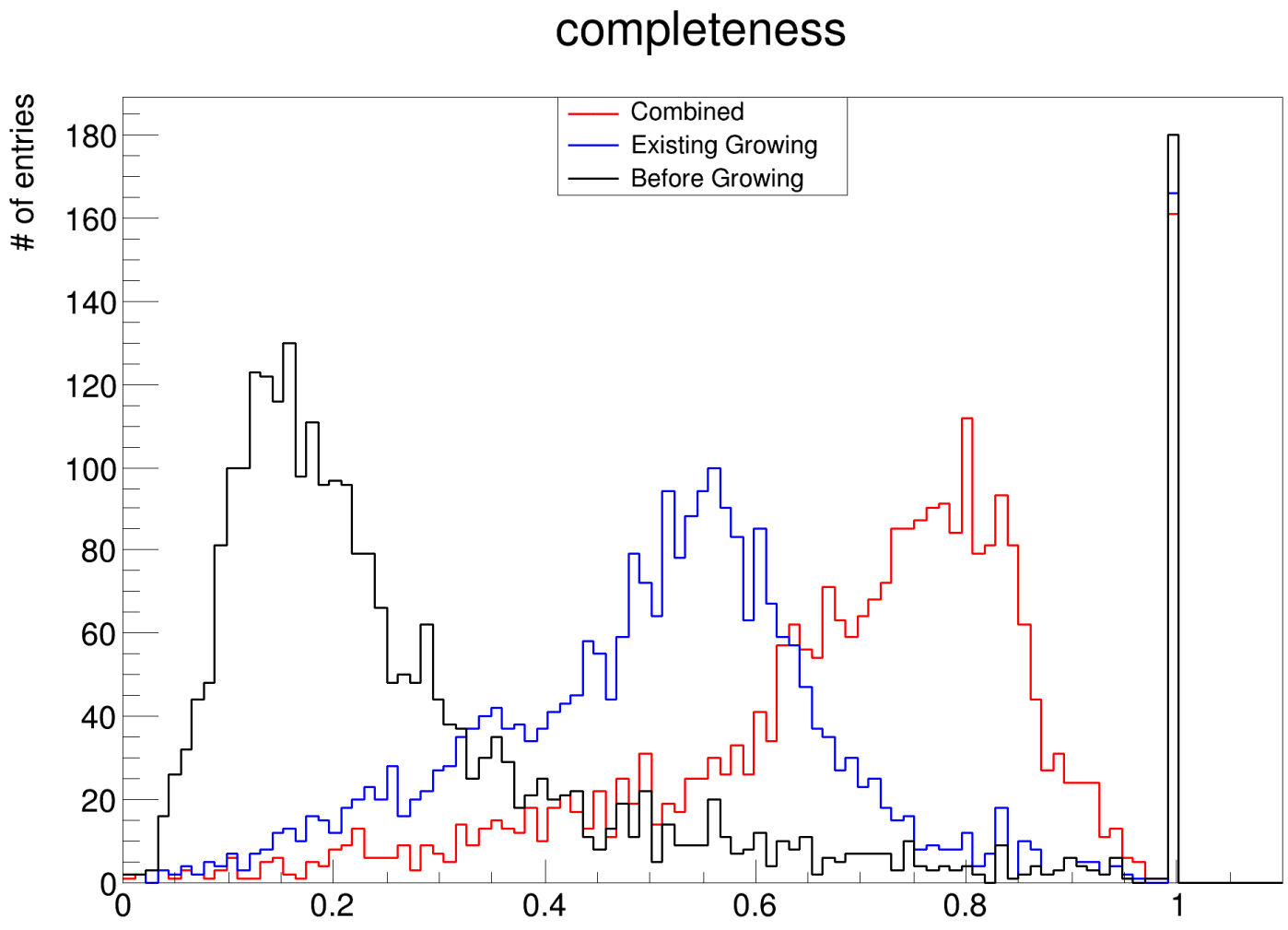
Then, I'll show the results for the end of Pandora, as used by analysers. The performance gap will have closed a lot in these plots, as there is a whole set of follow up algorithms that improve the initial growing.

Completeness - Before / After Growing - All Showers



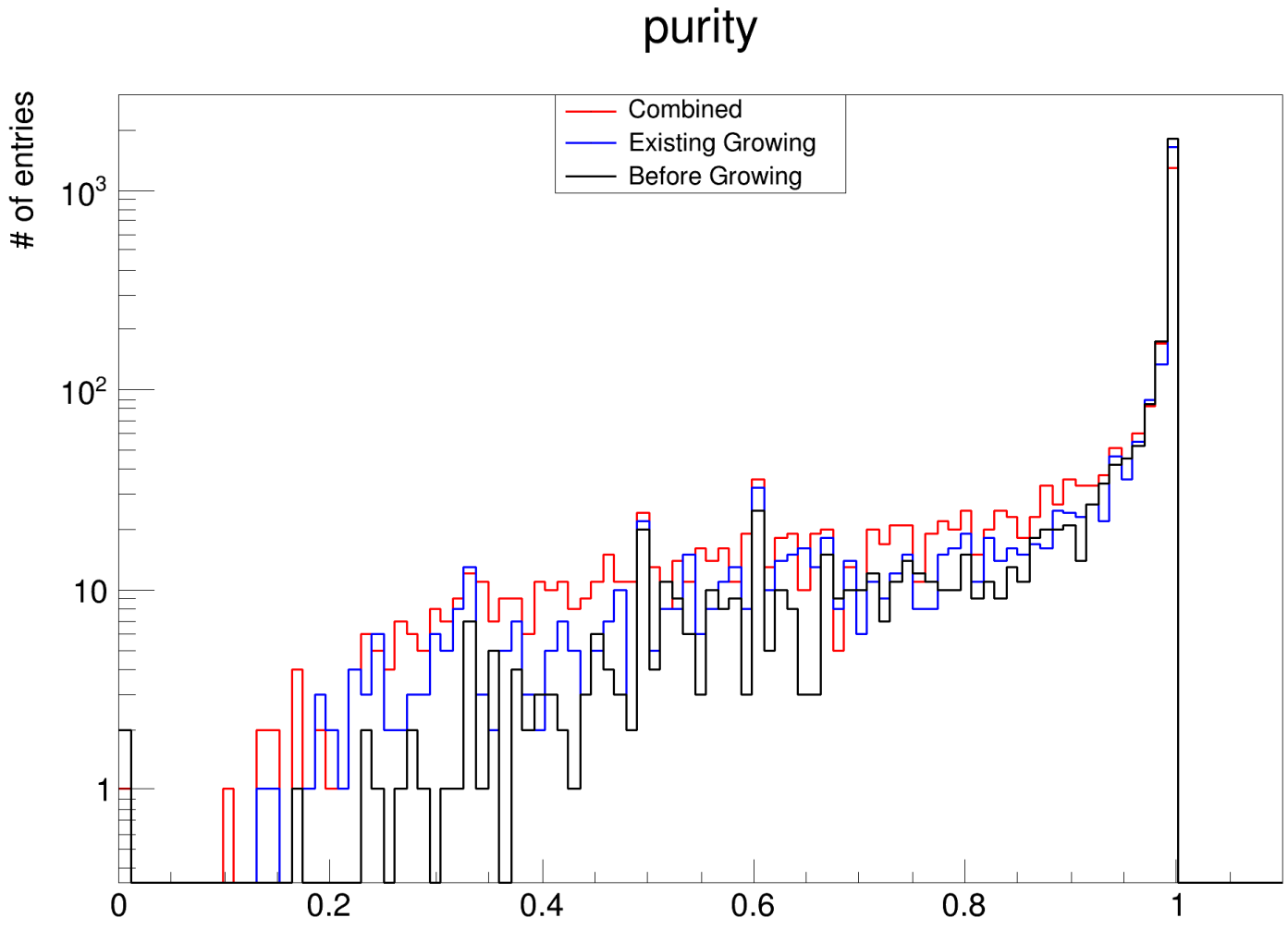
Completeness here is how many hits out of the total hits for that shower. Energy plots in backup.

Completeness - Before / After Growing - Largest Shower Only



Completeness here is how many hits out of the total hits for that shower. Energy plots in backup.

Purity - Before / After Growing - Largest Shower Only



Purity here is how many hits are included that are not part of the current shower. Energy plots in Backup.

Shower Growing Result Remarks

So, the improvements that I was seeing in both my single and two shower samples seems to also be seen in this neutrino sample, at least at this point, **mid way through Pandora**.

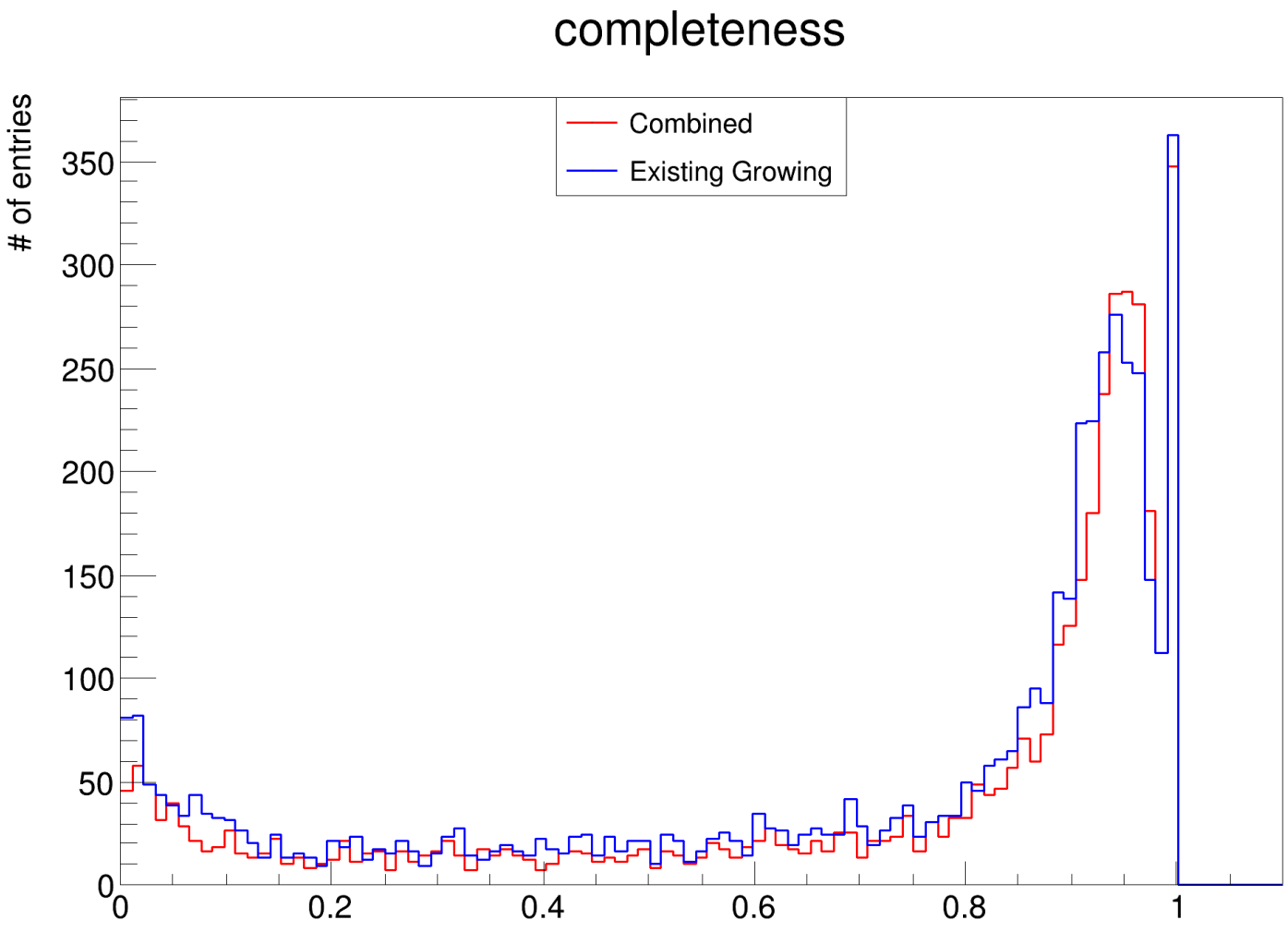
In numbers, the before and after shower growing looks like this. These are the average completeness and purity, with the averages for the largest shower in the event given in parenthesis.

Sample	Completeness	Purity
Before Growing	**7.8%** (30.3%)	**95.4%** (93.6%)
Existing Growing	**17.2%** (53.0%)	**94.4%** (90.5%)
DL Based Growing	**24.6%** (69.7%)	**93.1%** (86.5%)

These results are all reco-object first (i.e. iterate over reconstructed objects, and produce results about them), which could miss some small particles. But the high purity at least shows if this is happening, its not to a large effect.

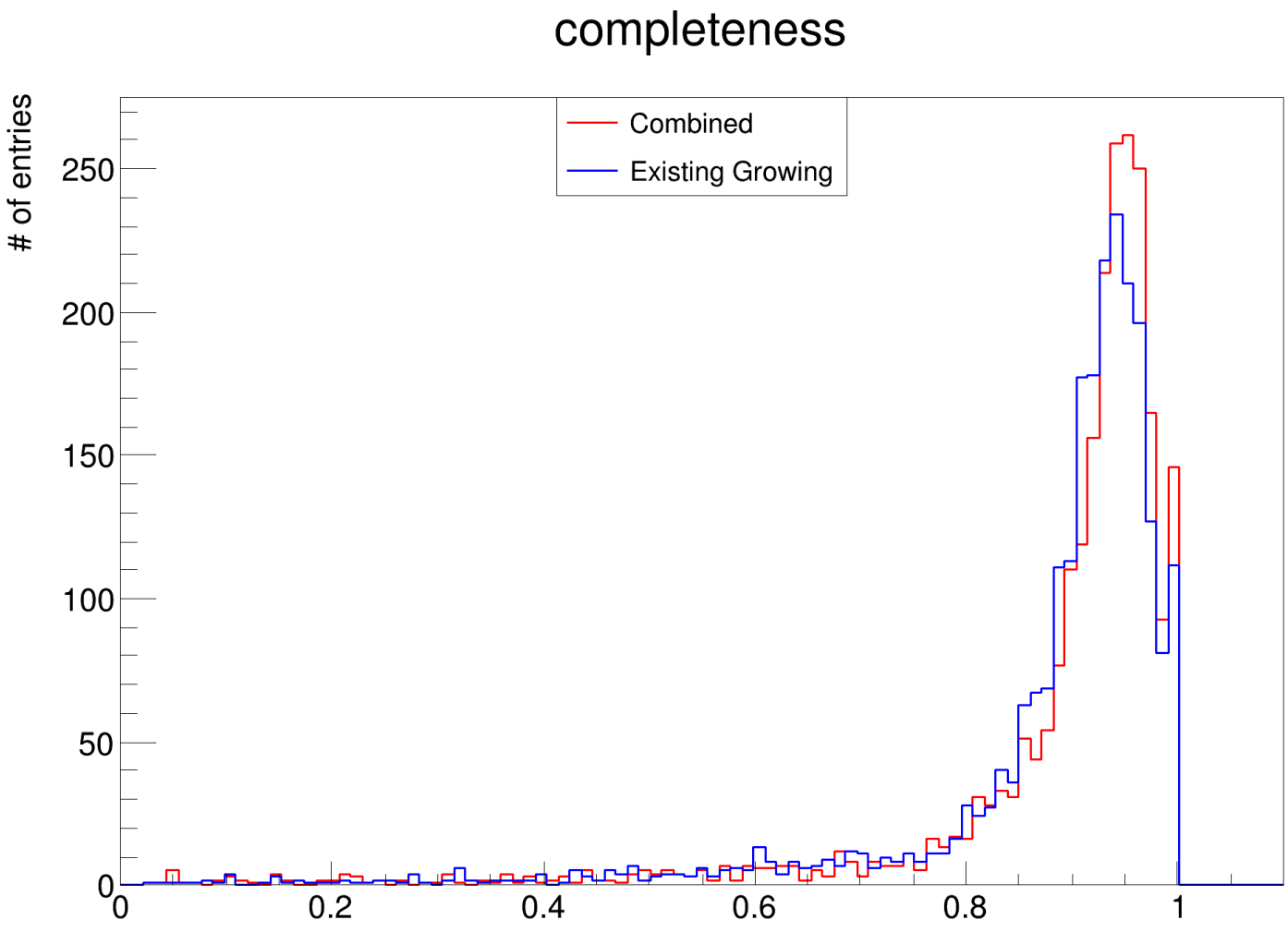
In both the other samples, I saw this large gap shrink considerably when considered in the context of the full Pandora reconstruction chain, so the next results consider the final Pandora performance, where the results should be closer.

Completeness - End Performance - All Showers



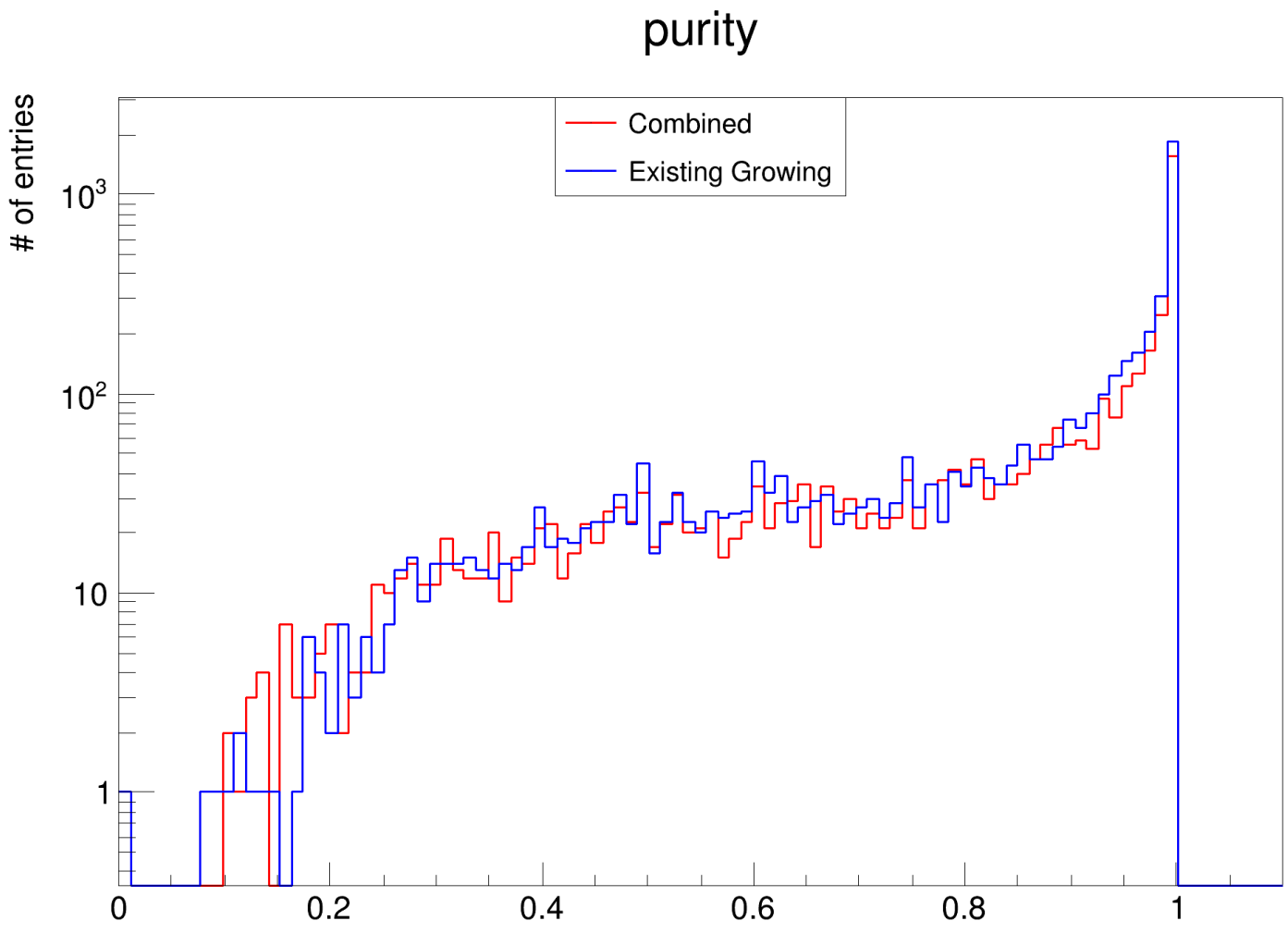
Completeness here is how many hits out of the total hits for that shower. Energy plots in backup.

Completeness - End Performance - Largest Shower Only



Completeness here is how many hits out of the total hits for that shower. Energy plots in backup.

Purity - End Performance - All Showers



Purity here is how many hits are included that are not part of the current shower. Energy plots in Backup.

End Result Remarks

It looks like the performance difference seen in the more basic samples is carrying forward in to the neutrino event sample, even without specific changes being made for the neutrino events.

Again, the averages are presented below.

Sample	Completeness	Purity
Existing Growing	**71.7%** (80.8%)	**85.8%** (85.4%)
DL Based Growing	**75.0%** (83.3%)	**84.6%** (83.8%)

As with the previous set of results, these are reco-first based metrics, so further plots are needed to check the full results. However, the objects that are being reconstructed have higher completeness and similar purity, which is encouraging.

Current Work

There is still a bit of follow up work to complete:

- Continue to optimise and train the network. With a full (and realistic) pipeline in place to run over neutrino events and get plots out, it is now even easier to benchmark models.
- Conduct some further studies on the limitations of the network. One of the test datasets is two showers from a shared vertex with a distribution in the angle between them. I'd like to check if there is any angular dependence where the network breaks down, as well as for energy.
- Understand the small showers that are contributing to the completeness plots.
- Produce equivalent plots that are MC-focused, i.e. iterate every MC in the event and check its completeness and purity, to also help track down the small showers that are appearing in plots.
- Run over a more varied, larger sample, including more than just ν_e events.
- Re-run cheating studies, to get a better idea of the current performance ceiling.

Summary

In conclusion,

- Using GNNs for a deep-learning based shower growing seems to work well!
- The performance seen on test datasets seems to have been fairly representative for the neutrino dataset as well.
- The current results are encouraging and can hopefully be improved further with updates to the underlying model and graph.
- More indepth studies into how the network performs for certain topologies are planned.

Deep Learning Based Shower Growing

Ryan Cross

Backup Slides

Example Shower Growing

To give a more concrete example, I've got some images that show how the process works. We start with raw clusters, here showing only the shower-like ones.

Example Shower Growing

Using these raw clusters, we can build a graph of the event, pick some candidate growing clusters, and then grow them to increase the size of that shower. Here, the red clusters are the selected clusters to grow.

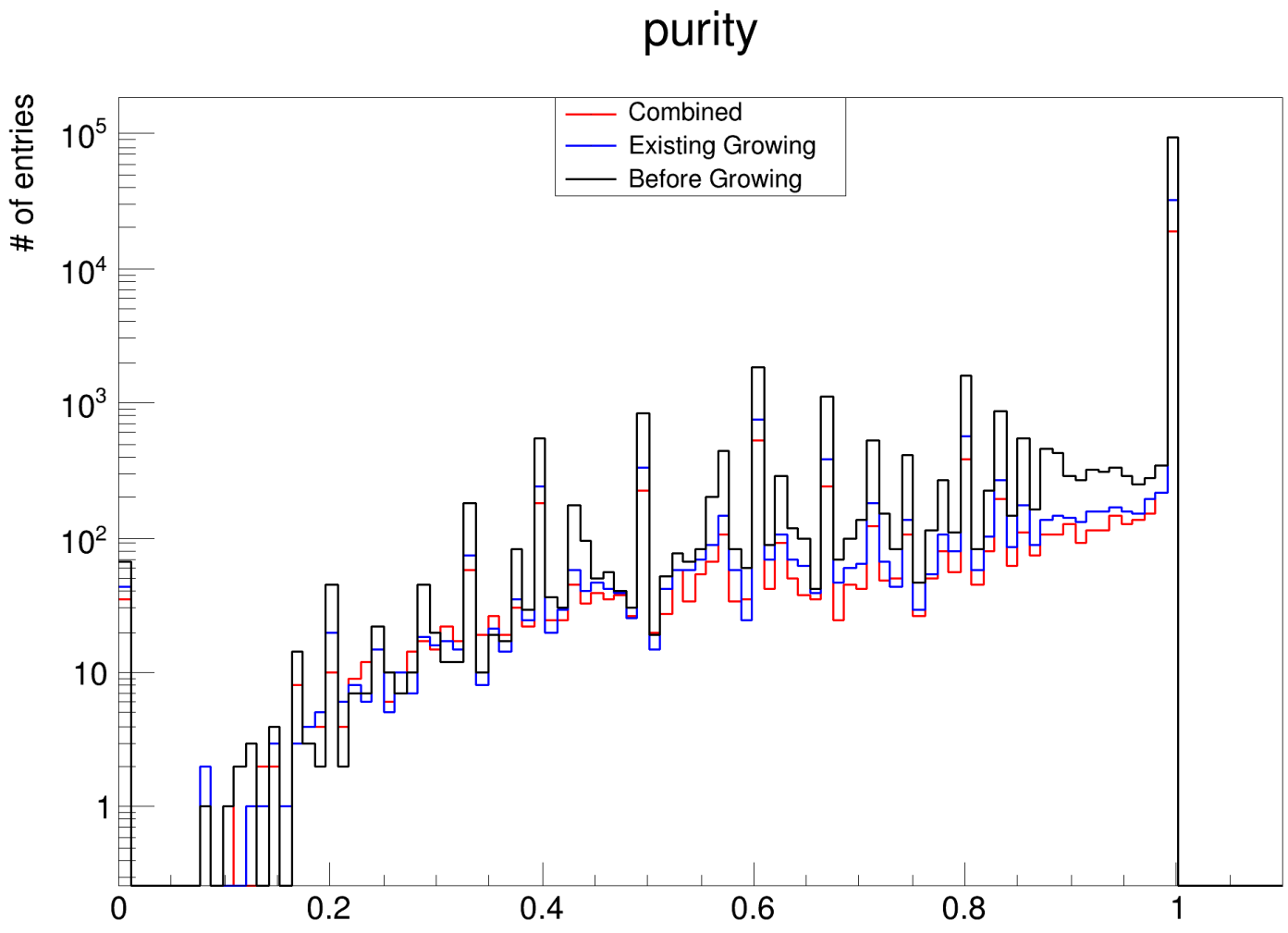
Example Shower Growing

The result of the network is then parsed, so we look at what the network result was and how strong of a merge-decision it was, and use that information to merge clusters into the current input cluster.

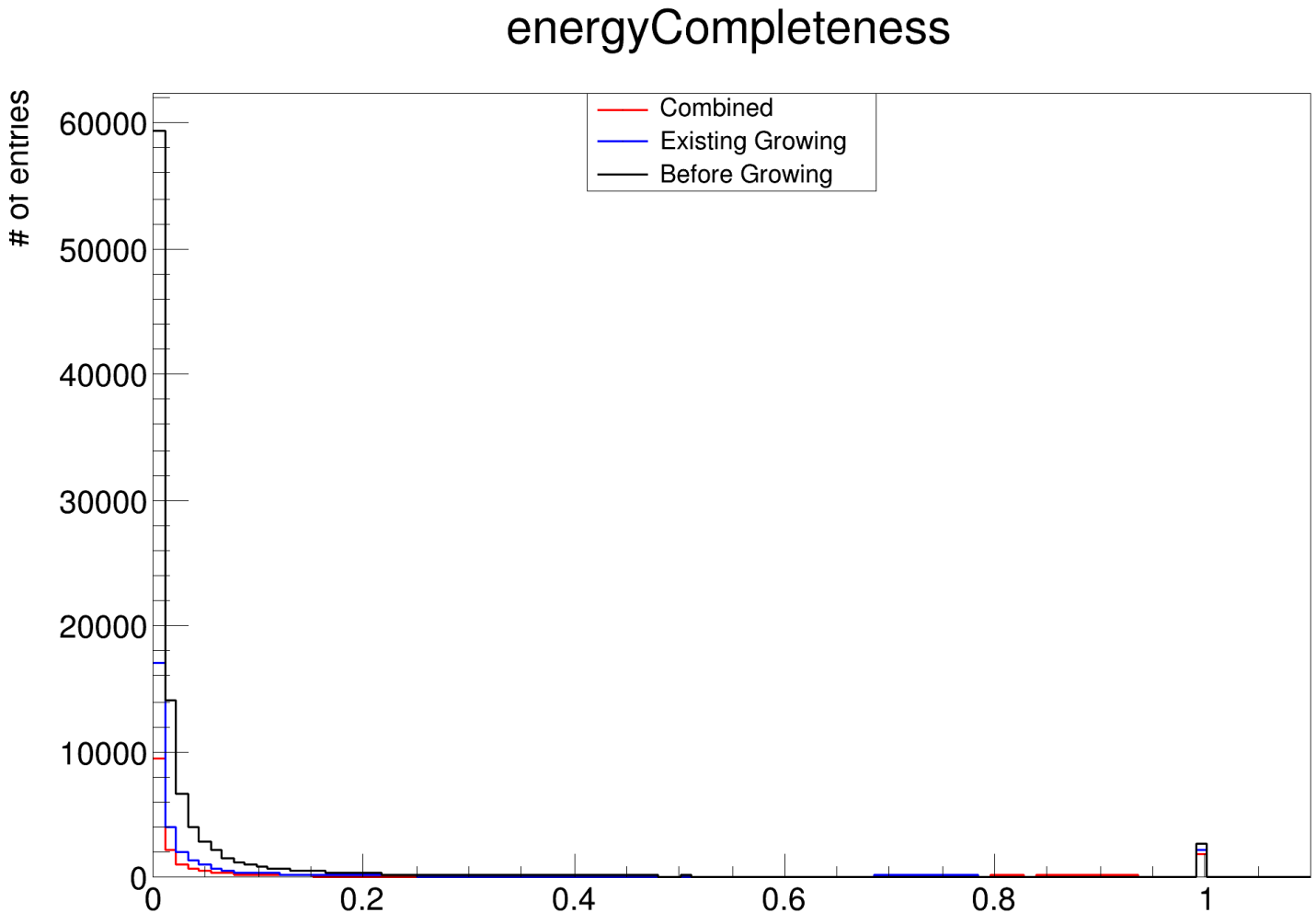
Example Shower Growing

The previous steps showed multiple iterations of the shower growing at once, but usually this is a sequential process, removing clusters as they are merged. For comparison, this is the original shower merging for the same event.

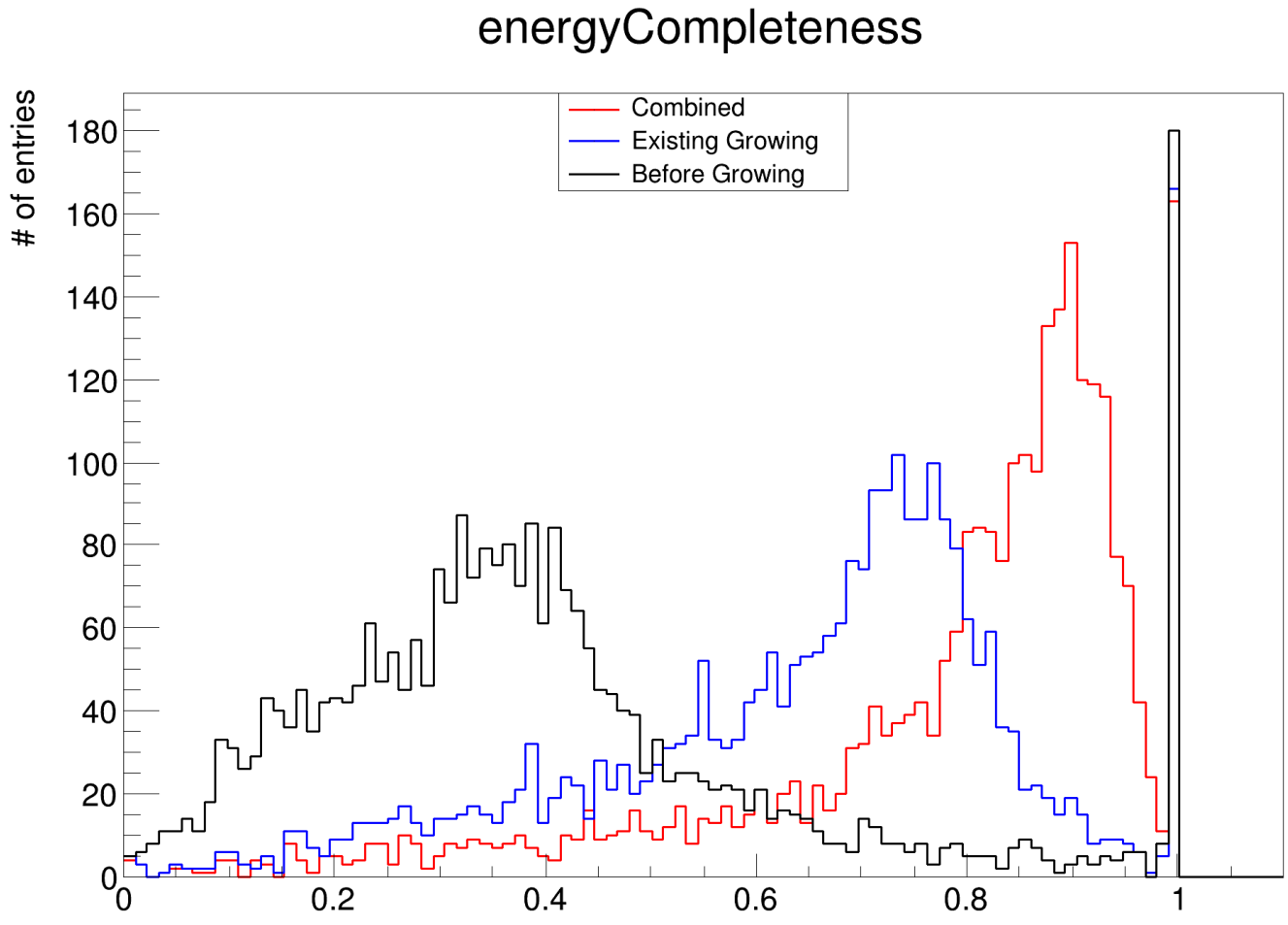
Purity - Before / After Growing - All



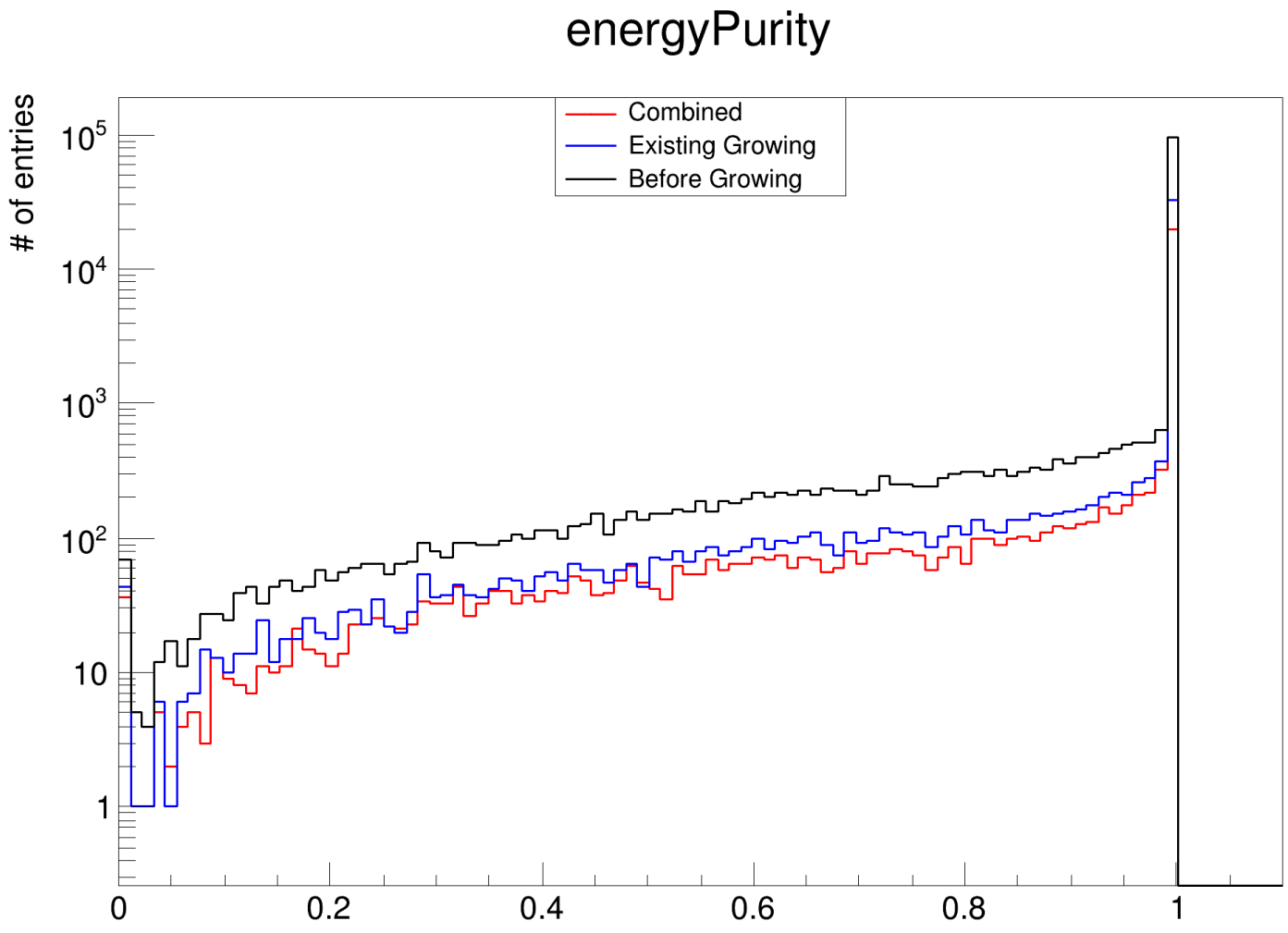
Energy Completeness - Before / After Growing - All



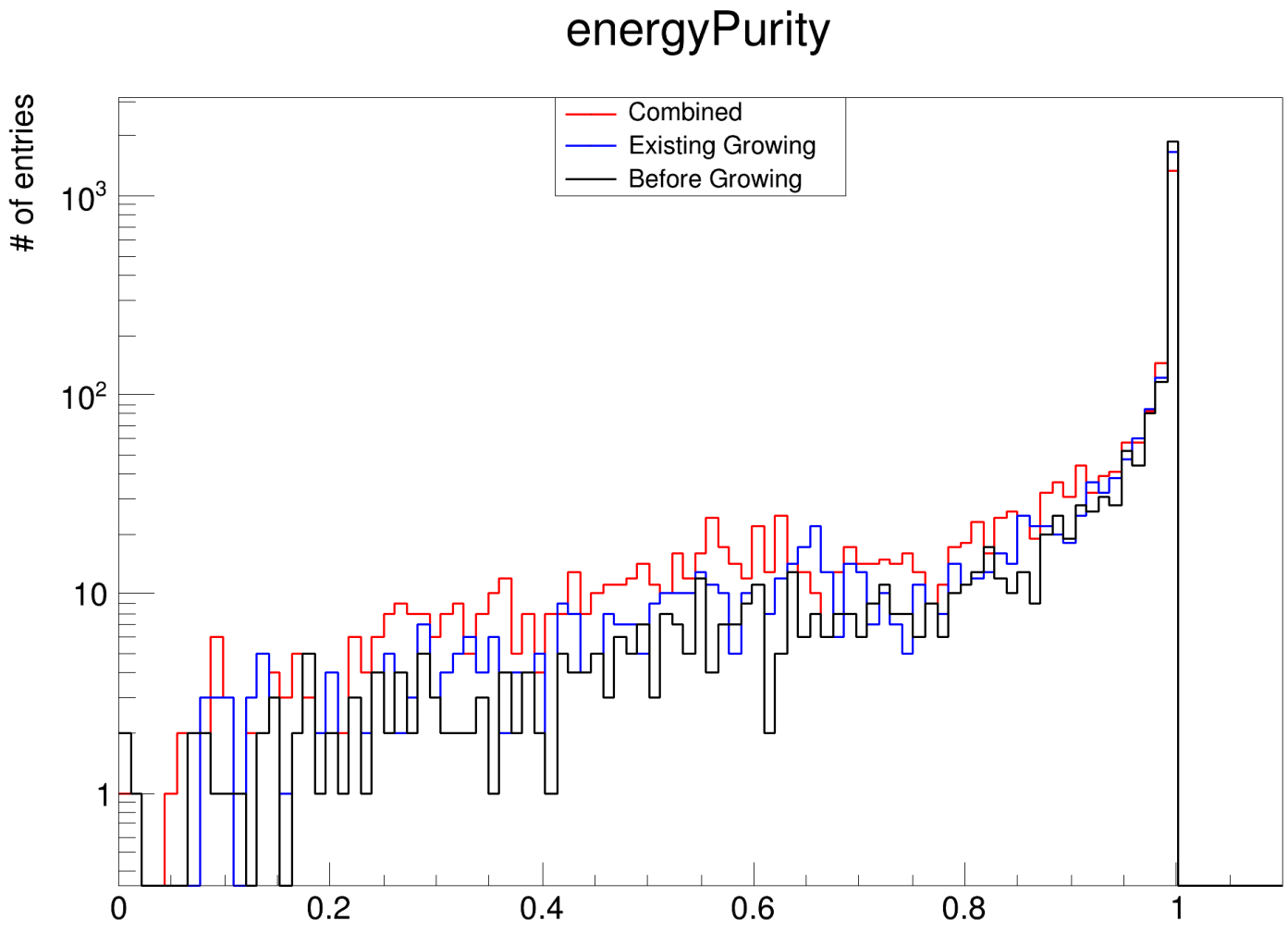
Energy Completeness - Before / After Growing - Largest Shower Only



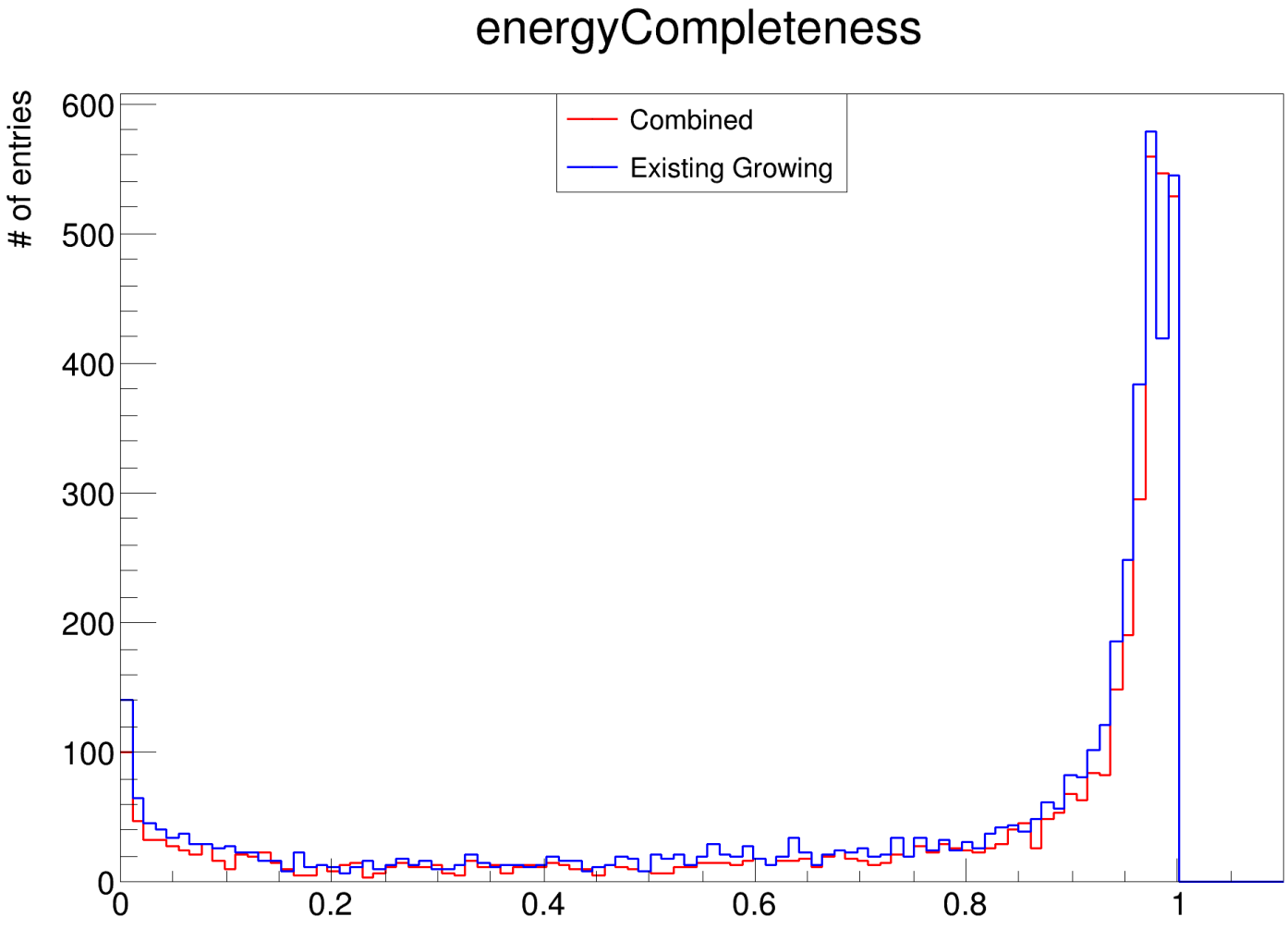
Energy Purity - Before / After Growing - All



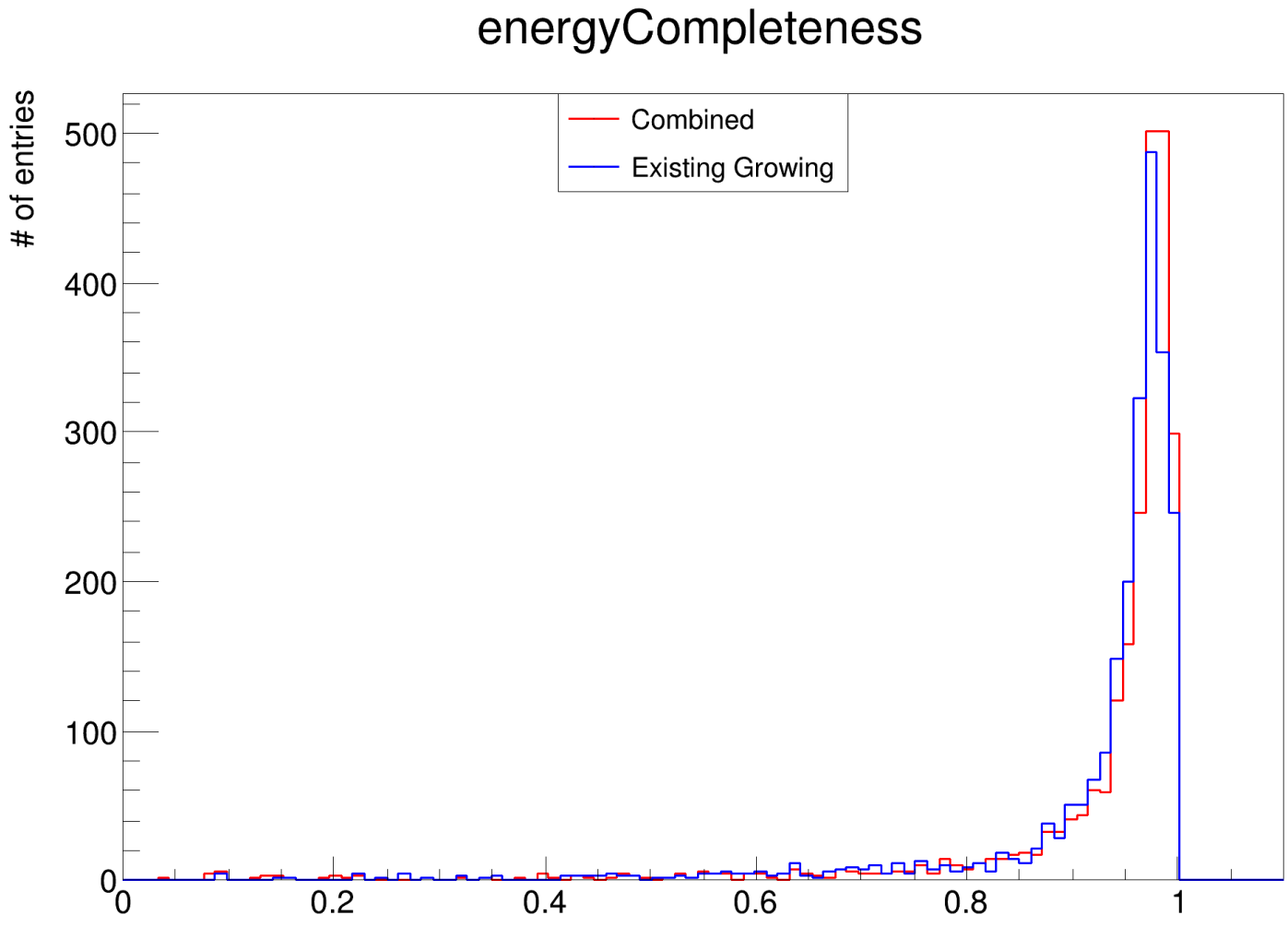
Energy Purity - Before / After Growing - Largest Shower Only



Energy Completeness - End - All



Energy Completeness - End - Largest Shower Only



Energy Purity - End - All