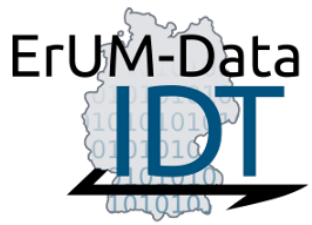
MC filters using ML techniques for ATLAS

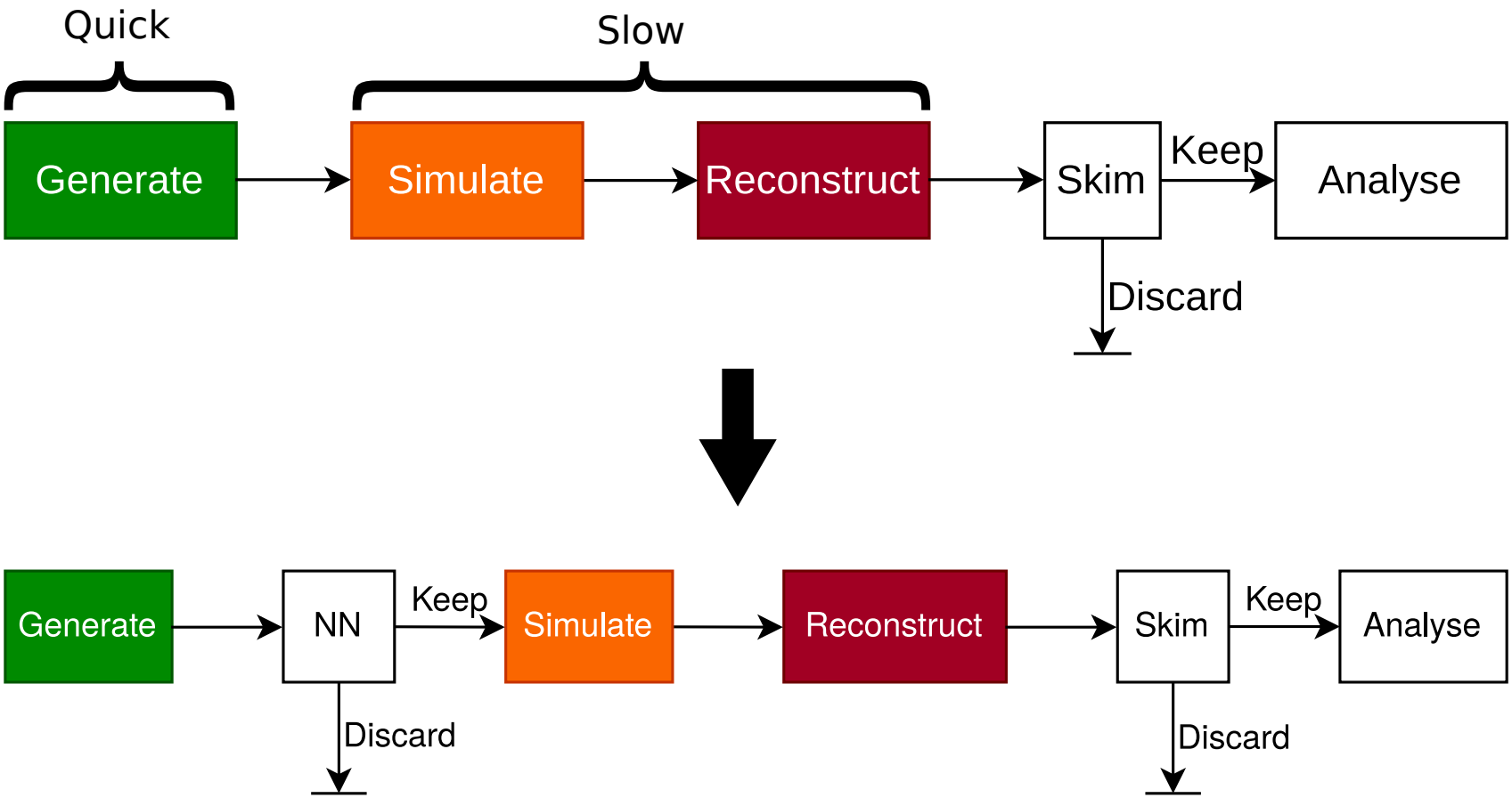
Nikolai Hartmann, Michael Fichtner, Simon Graetz

LMU Munich

February 24, 2020



What this is about



Like any MC filter, but this time with machine learning!

Original study at Belle II

By James Kahn, see talk and thesis

- Trying to predict which events pass skim before detector simulation
  $\rightarrow$ reduce computing effort if $t_{\mathrm{Gen}} << t_{\mathrm{Sim}}$
- Using Neural network, mainly 1D convolutions over particle list and tokenized decay string
  $\rightarrow$ rather low level quanties, requires GPU for training
- Assuming to take only filtered events - reach 50% reduction in simulation time
  (for a particular sample, compared to no filtering at all)
- But this introduces biases, hoped to be dealt with by adjusting training (e.g adversarial setup) or reweighting distributions
  (no studies yet on filtering like at ALTAS - by also simulating events that don't pass filter)

$\rightarrow$ should we try something similar at ATLAS?

Situation at ATLAS different

Filtering is already standard for many samples ... and done on many different variables.

No biases, because we simulate "all slices" and weight accordingly by using the filter efficiency

General problem difficult to define:
Find a filtering scheme that provides for given computing resources the lowest statistical uncertainty for a large number of different analyses?
$\rightarrow$ try to optimize for a specific selection first

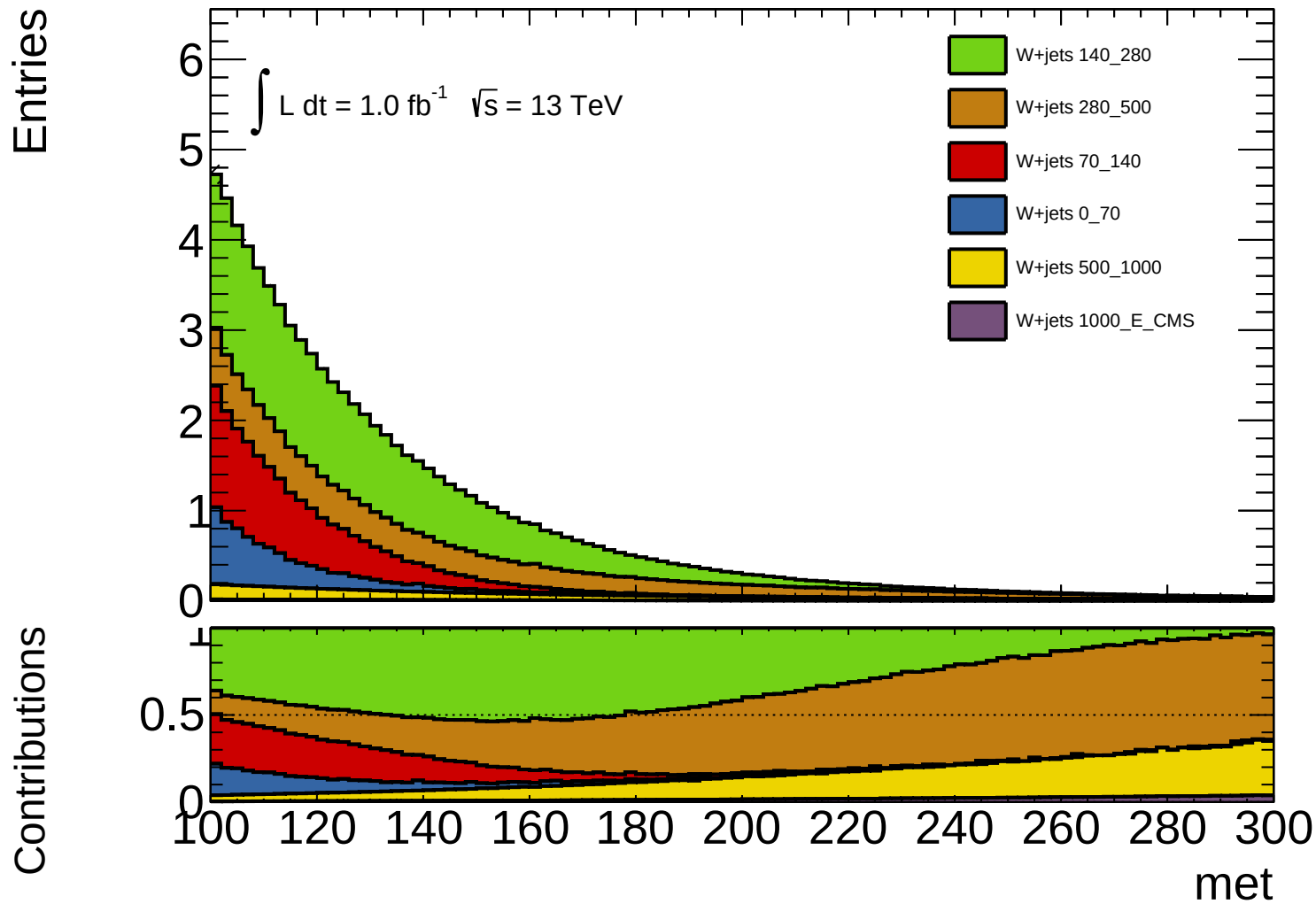
Also: many more particles in pp collisions (1 or 2 orders of magnitude!) (even without pileup)

An Example: W+jets in 1-Lepton searches (based on $\not{E}_T$ and m_{T})

(let's ignore for now that $t_{\mathrm{Gen}} << t_{\mathrm{Sim}}$ doesn't hold so much for current Sherpa samples)

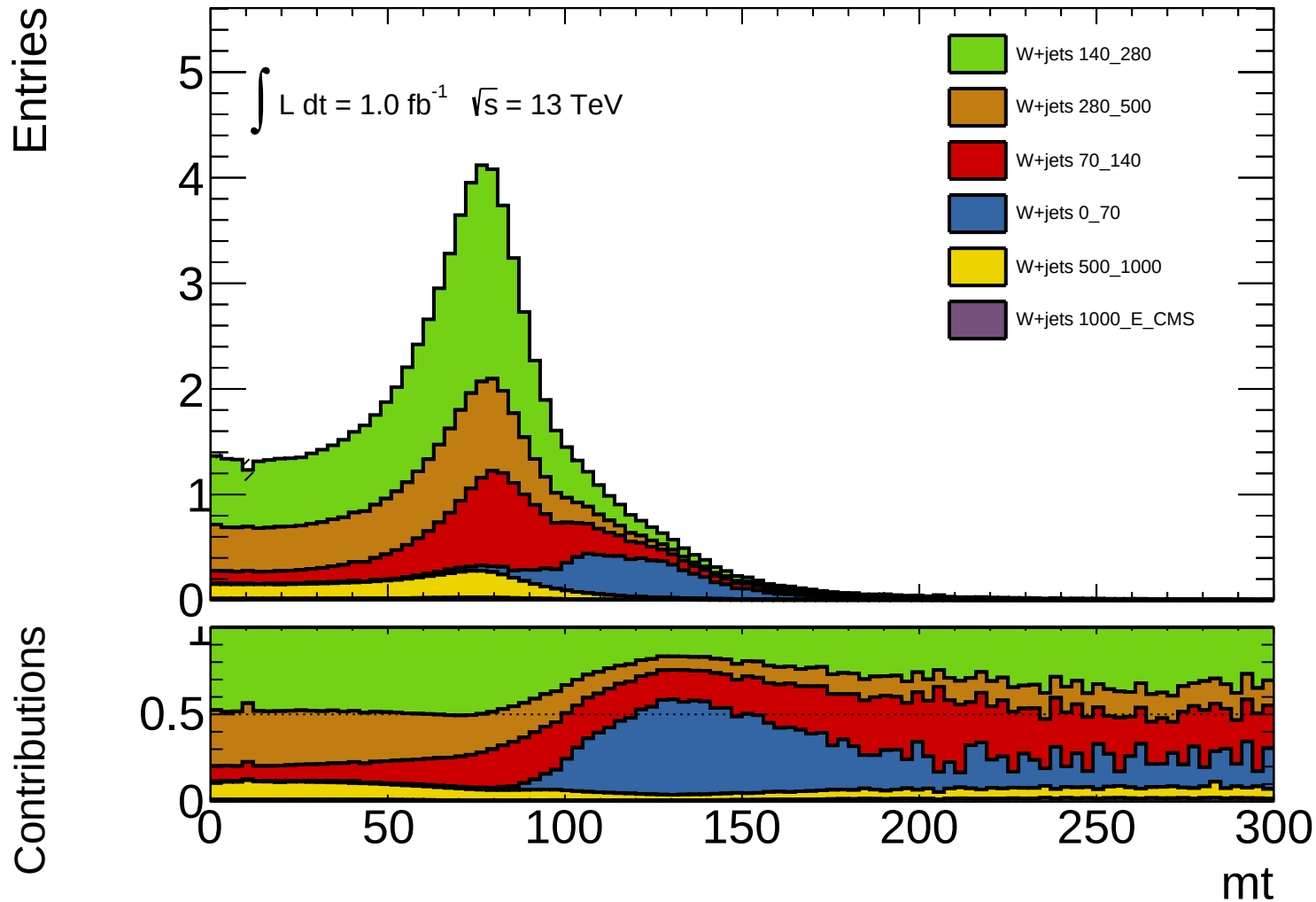
Filters $\approx$ work for $\not{E}_T$

Slicing in $\max(H_{\mathrm{T}}, p_{\mathrm{T}}^{W})$ (+ b/c-filter/veto, $e/\mu/\tau$)



$\rightarrow$ correlation not extremely good, but works

Example of a problem: m_{T}



$\rightarrow$ in region around kinematic edge even enhancement of "low" slices

Study by Michael Fichtner

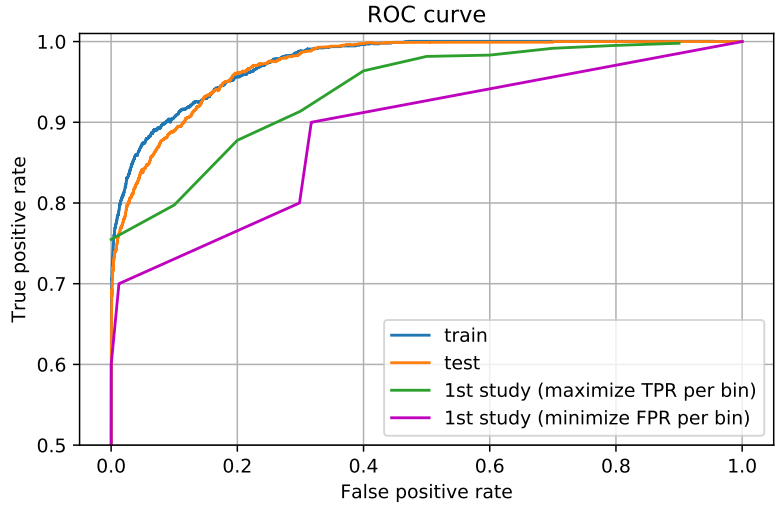
Try to predict which events pass reco $\not{E}_T > 150$ GeV and $m_{\mathrm{T}} > 150$ GeV (+basic acceptance cuts, borrowed from SUSY 1L analysis)

Training dataset: ≈500k pass, ≈2M fail events on generator level

Training procedure:

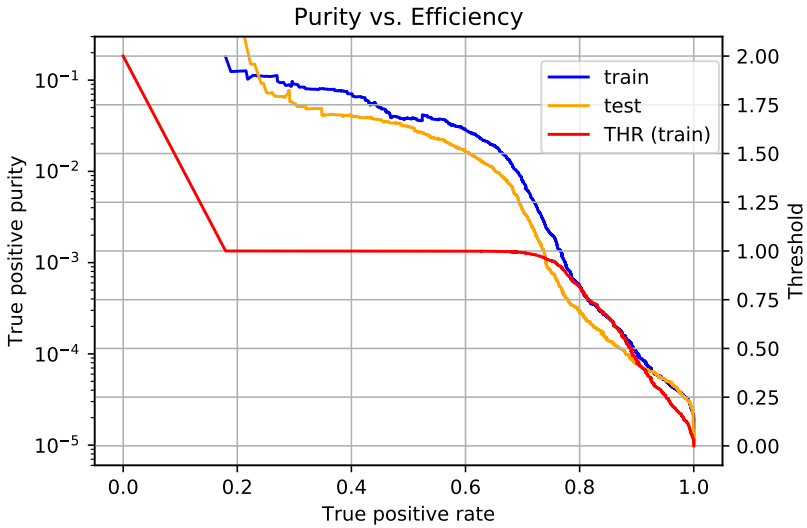
- train with event weights (generator and filter efficiency)
- also use negative weights
- split into train/test sample
- use 30% of train sample for validation during training (Early stopping)

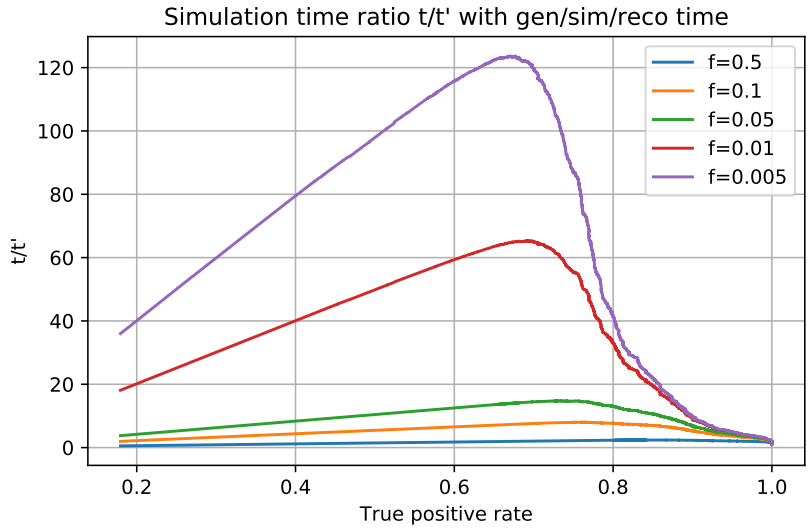
First study: 4-vectors from TRUTH3



- Mimick reco quanties by using TRUTH3 content: lepton and jet 4-vectors and True $\not{E}_T$, m_{T}
- Try simple cut on True $\not{E}_T$ and m_{T} (green and purple line)
- Try training NN on momentum components (+True $\not{E}_T$, m_{T})
- NN clearly more effective of in region of higher True positive rate
- On the other hand: In region of high purity, not much difference

Purity and increase of statistics





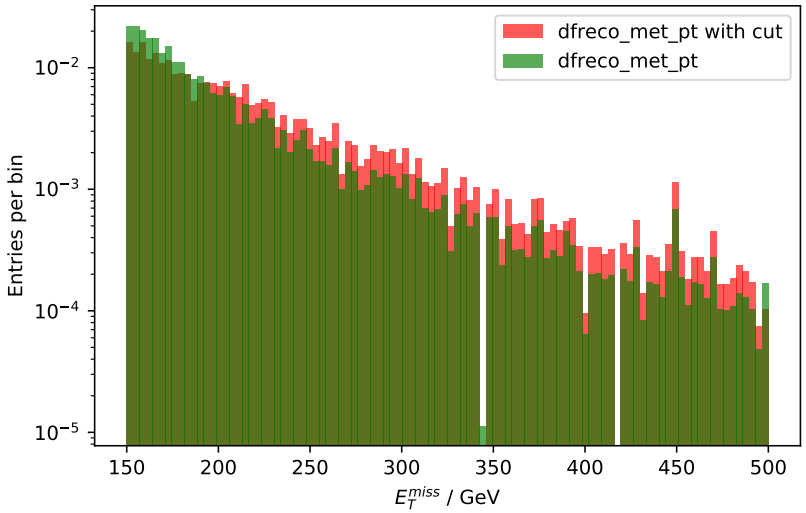
- Initial purity (fraction of *pass* events): $\approx 10^{-5}$
  $\rightarrow$ increased to $\approx 10^{-2} - 10^{-1}$ after applying NN

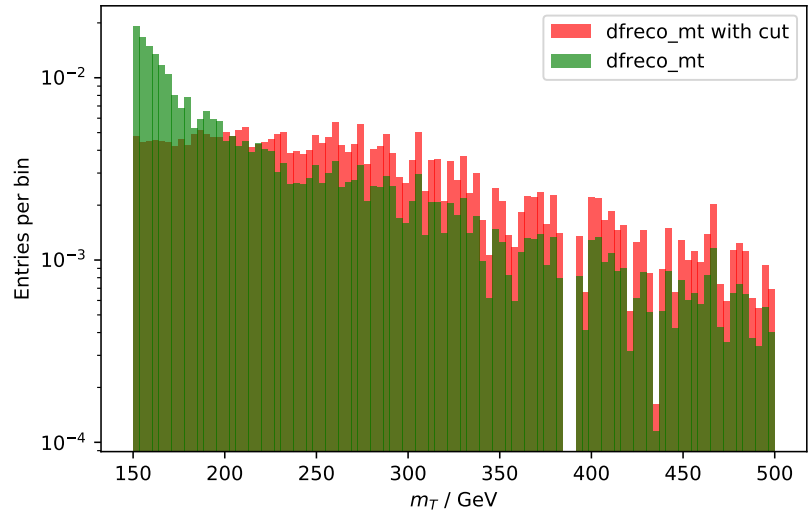
- Increase of statistics (assuming no simulation for rejected events) for spending the same computing time depends on fraction $f = \frac{t_{\text{gen+NN}}}{t_{\text{sim}}}$

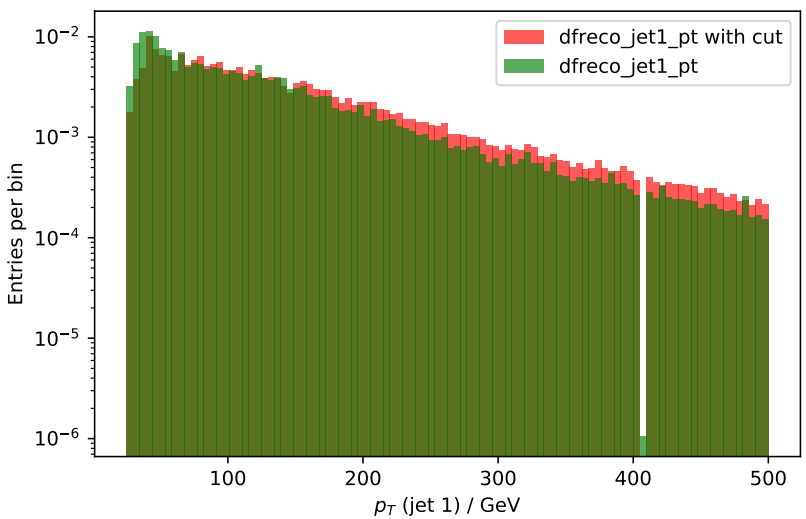
t: Time spend for simulation+reconstruction without applying NN
t': Time spend for simulation+reconstruction with applying NN
(by creating the same number of simulated *pass* events)

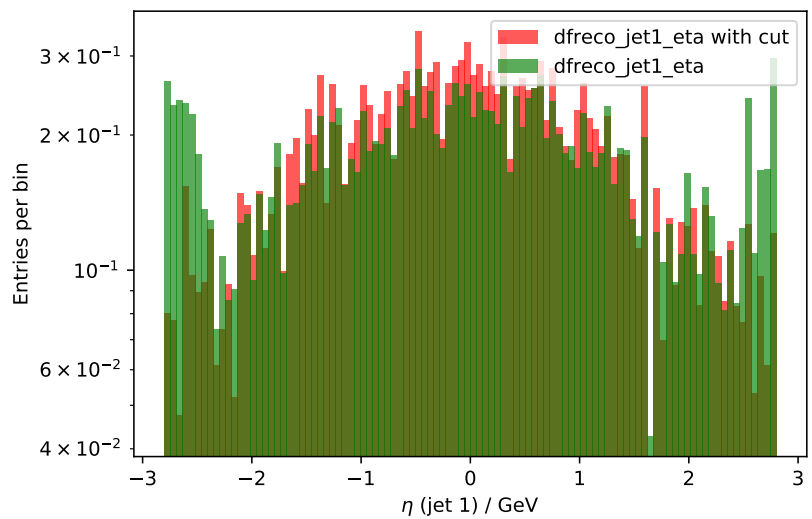
Bias in reco quantities

(For pass events, if not simulating rejected events)









$\rightarrow$ distribution sculpted because of false negatives

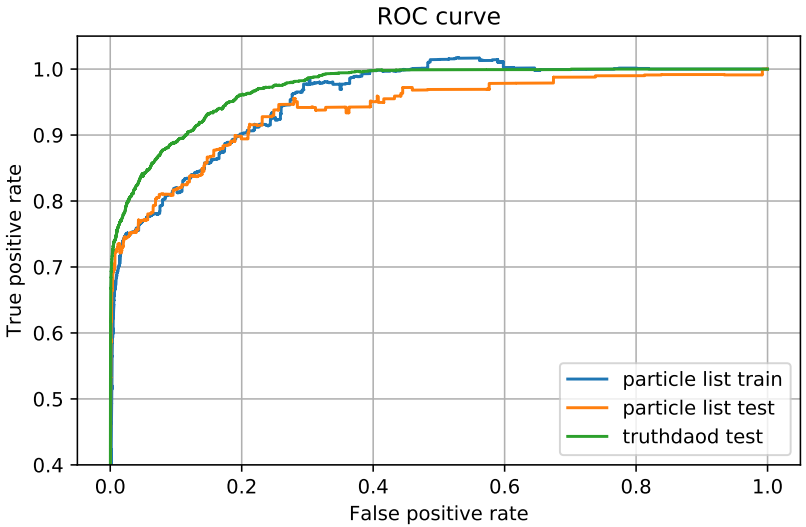
$\rightarrow$ will need to simulate some rejected events as well ("slicing")

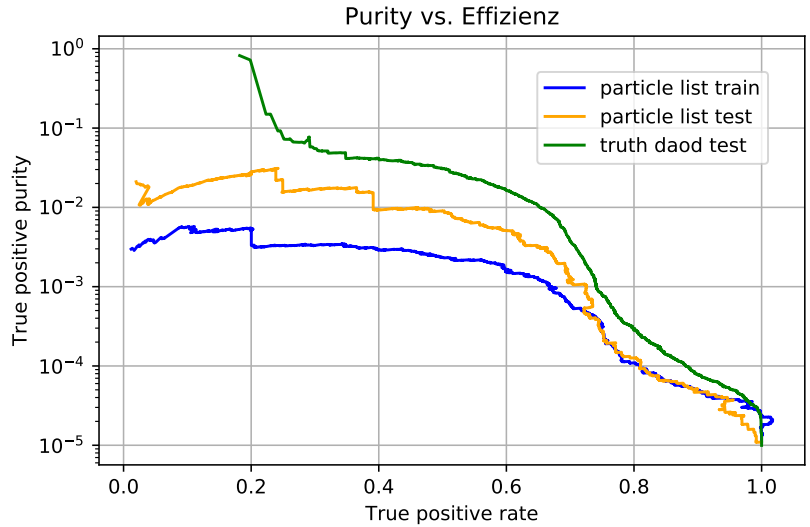
Second study: Train on plain particle list

"Low level" variables

- Inspired by Belle II study: Try 1D convolutions over particle lists (all particles, also intermediate particles)
- Features for each particle in list: pdg id, p_x, p_y, p_z, E, production vertex, decay vertex, status bit (one-hot encoded)
- Feed pdg id through embedding layer (map onto 8 dimensional vector) first
- Take up to 2000 particles per event (compare to Belle II study: 100) $\rightarrow$ fill non existent entries with 0

Sort of works!

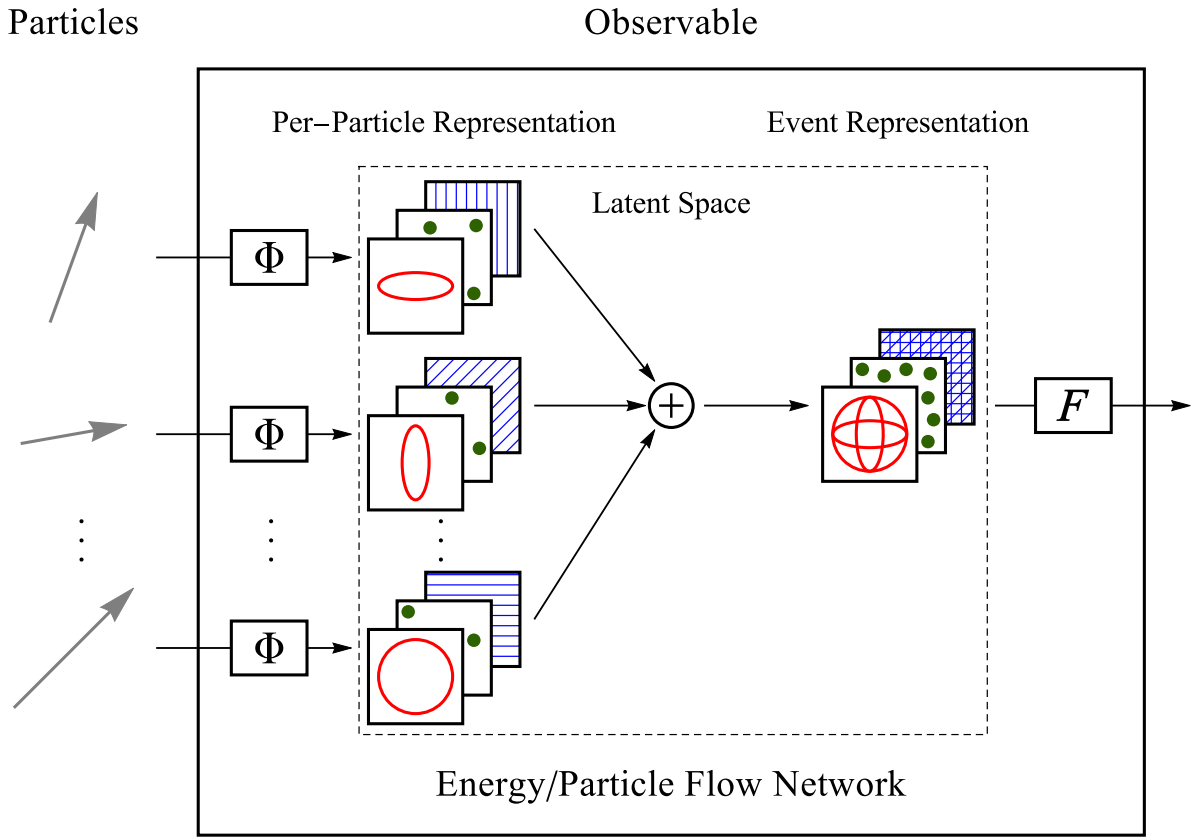




- Surprisingly this works to some extend
- Training takes several hours, on an Nvidia Tesla V100
- Still, doesn't yet reach the performance of "high-level" variables (truth DAOD)
- Note: difference between train/test here might be due to bad "random" selection
  $\rightarrow$ was improved for next study

Particle Flow Networks

arXiv:1810.05165

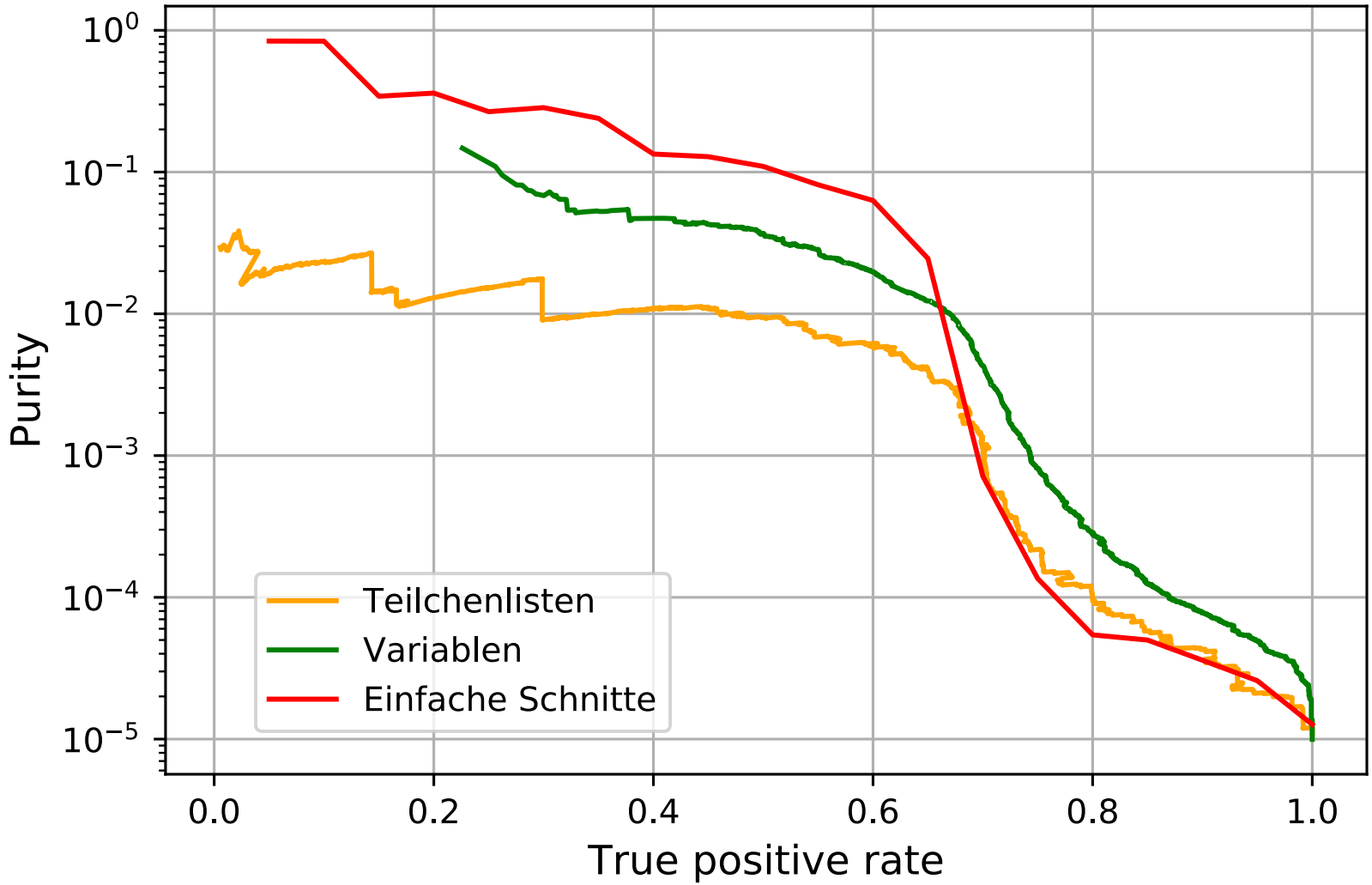


- try this for training on all event-level particles
- use full 4-momentum information ("Particle Flow Network")
  (don't care about soft/collinear safety here since only applied to MC)
- advantage w.r.t RNN: independent of order, better parallelizable

$\rightarrow$ Bachelor thesis Simon Graetz

But first: follow up on this

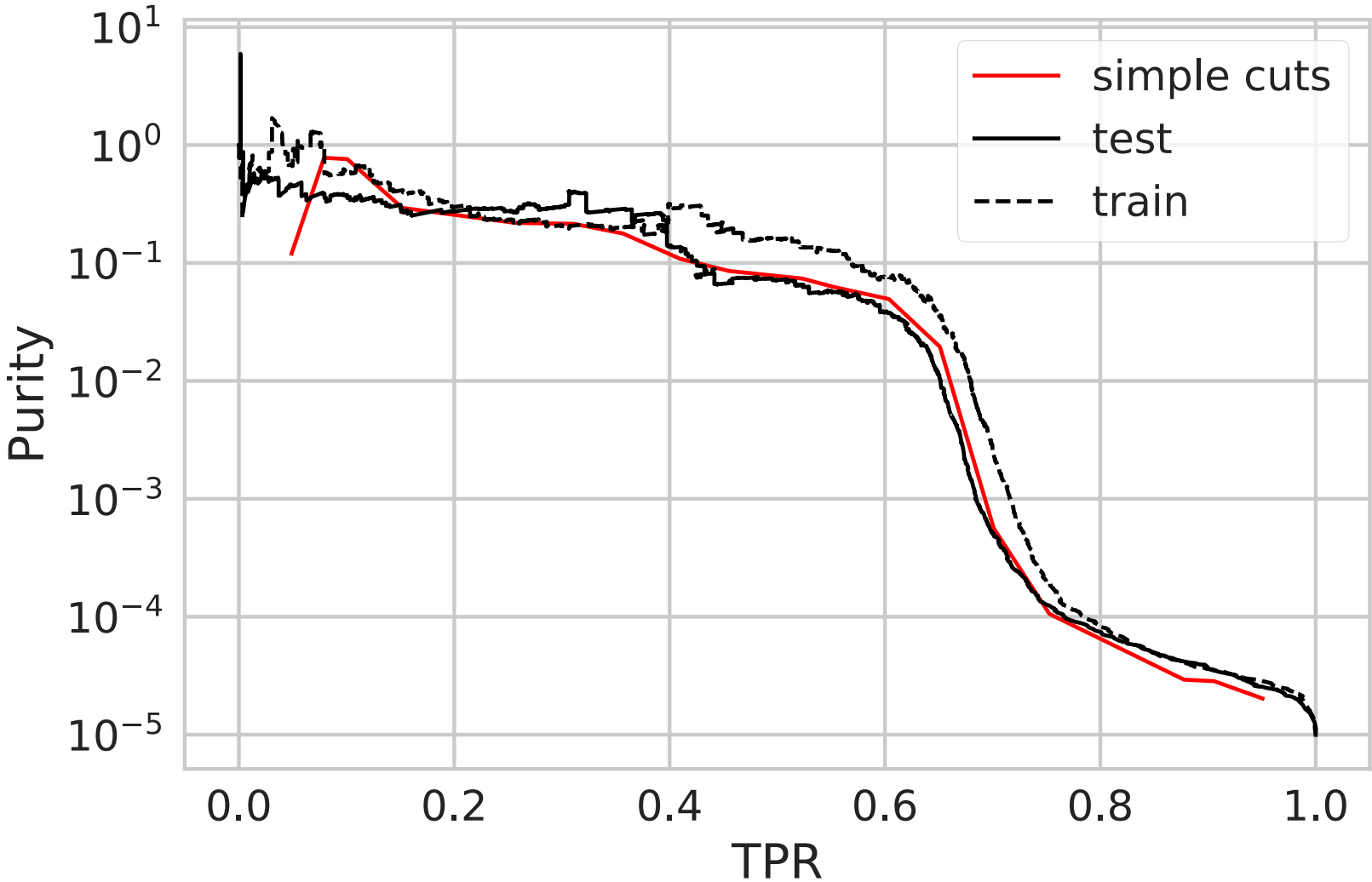
(Plot by Michael Fichtner)



$\rightarrow$ NN trained with more variables worse that cuts on 2 variables in region of high purity (high rejection of false events)

Sanity check - train only on the same 2 variables used for cuts

(Plots by Simon Graetz)

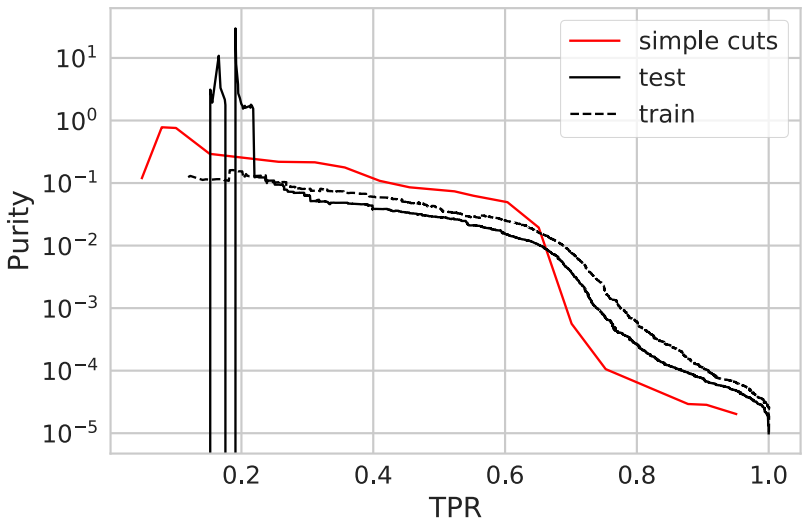


$\rightarrow$ consistent

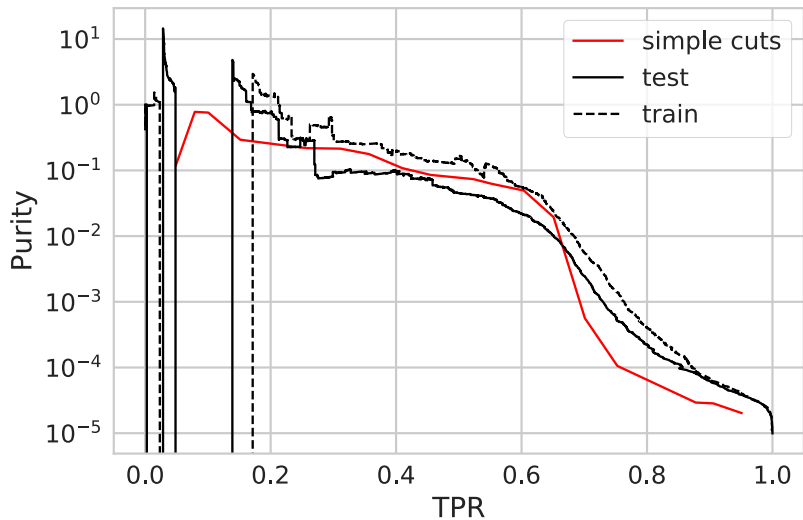
Train on all variables with and without "class weights"

Class weights: apply one weight for true and one for false events such that they have the same sum of weights
$\rightarrow$ sometimes helps with convergence for imbalanced training datasets

with class weights



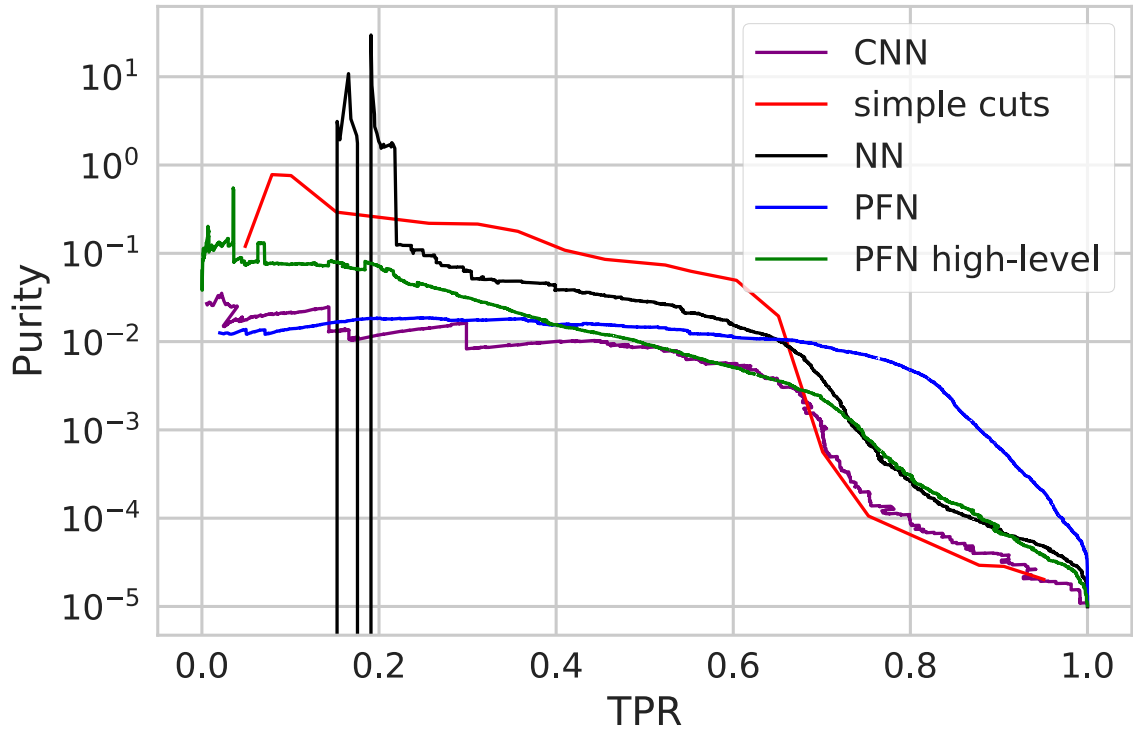
without class weights



(Plots by Simon Graetz)

- **without** class weights the network learns better learns "region of high purity"
- maybe possible to find better loss function?
- but also more overtraining

Training with Particle Flow Networks



Tried to train both on low and high level variables:

High-level:

$p_{\mathrm{T}}, \eta, \phi$ of 1 Lepton, $\not{E}_T$, jets + extra variables after PFN layers

Low-level:

Similar to CNN setup, but this time only final state particles (up to 1000)

$\rightarrow$ promising improvements in region of high TPR

Where to go from here?

- Better metric: Reduction of stat. uncertainty for same computing time?
- Would have an additional parameter: How many rejected events to simulate?
- Could imagine to do this in an "importance sampling" style
  $\rightarrow$ accept events based on predicted probability, reweight accordingly
- Are there other ways to formulate the "general" MC filtering as an optimization problem?

Go back to the ideas of "reducing the bias" (e.g by an adversarial setup) when not simulating rejected events:

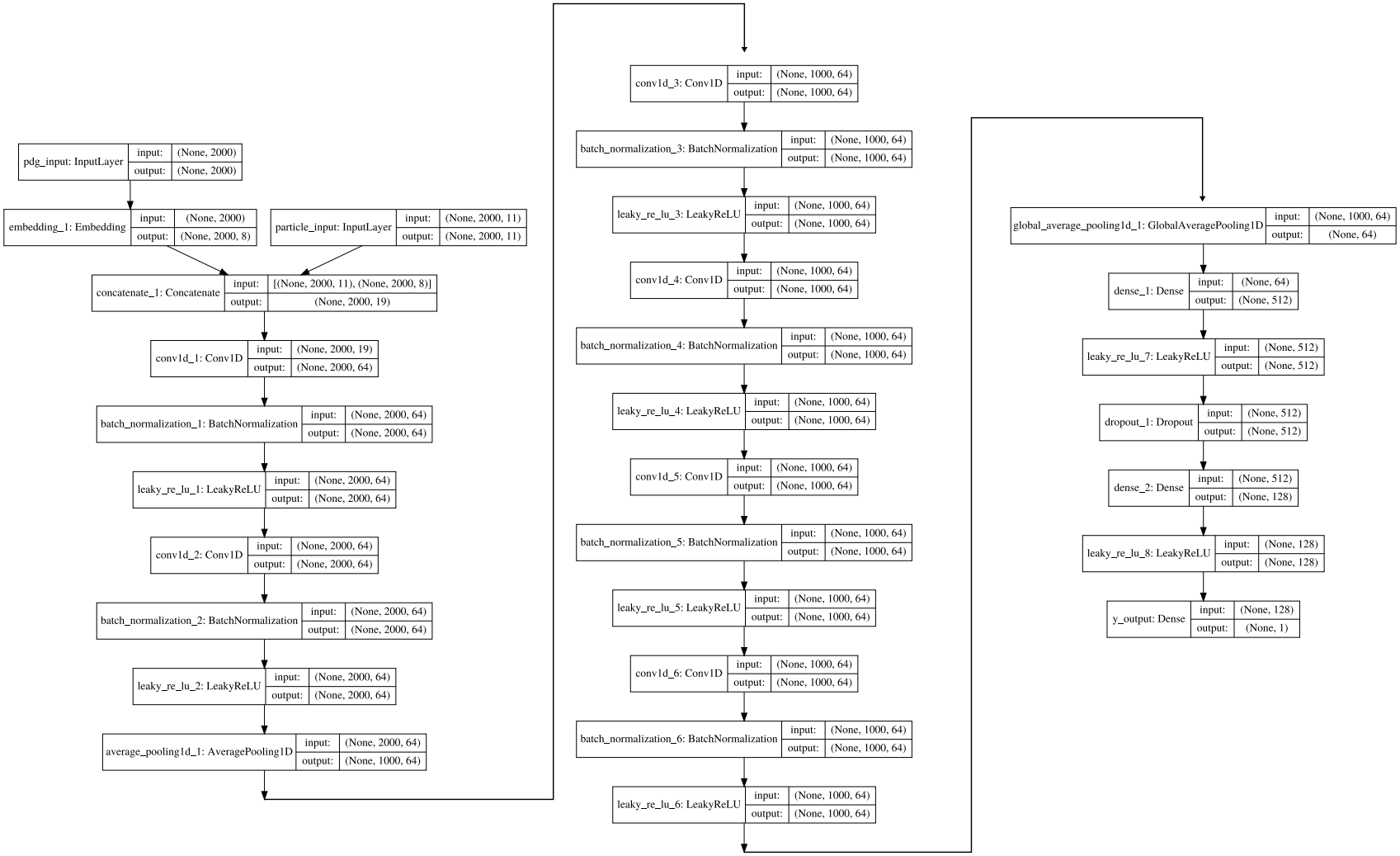
- I don't believe this will be possible to the full extend
- Probably tradeoff between bias and performance
- Introducing a systematic just because of filters would be terrible
- However: A classifier that introduces less bias on the reco quantities we will use in the end will also lead to fewer rejected events (with potentially high filter efficiency weights) end up in the final selection if they are simulated as well

Backup

Some details

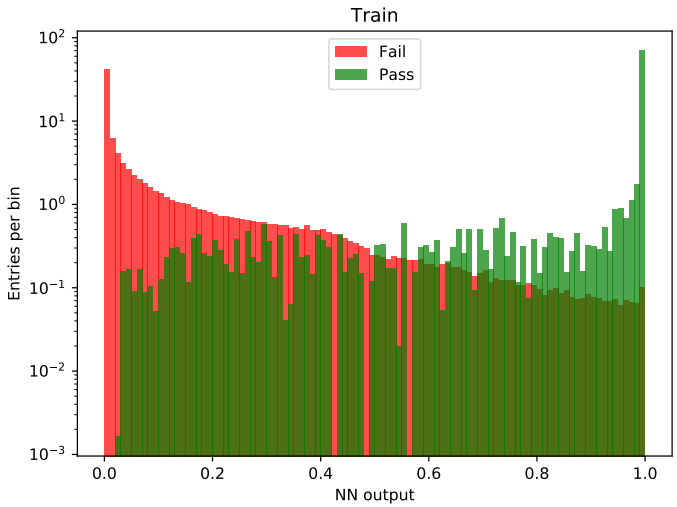
- *fail* training events selected by taking a random selection of a 500th of all files and remove overlap with *pass*
- *pass* events picked by running grid job on **all** EVNT files and picking the right event numbers/dsids

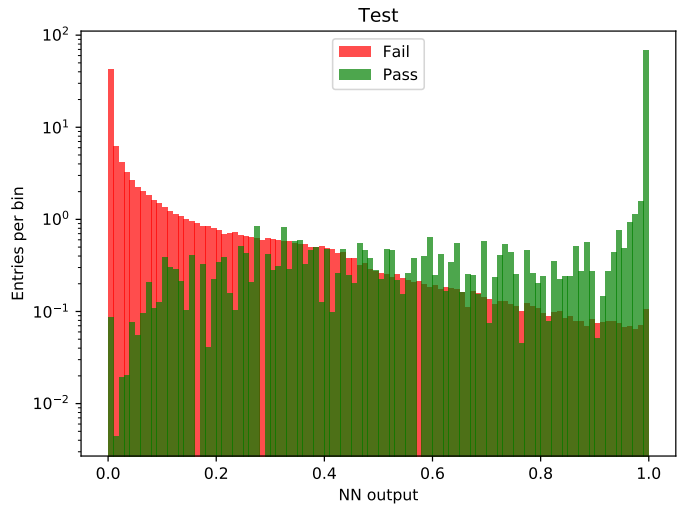
Complete architecture of particle list training



Distribution of output score

TRUTH3 variables:





Particle lists: