Adversarial Tuning of Perturbative Parameters

in Non-Differentiable Physics Simulators

Michela Paganini
Ph.D. Student, Yale University
Affiliate, Lawrence Berkeley National Lab

@WonderMicky
@mickypaganini
michela.paganini@yale.edu
mickypaganini.github.io

with **Luke de Oliveira**, **Steve Mrenna**,
Ben Nachman, **Chase Shimmin**, **Paul Tipton**

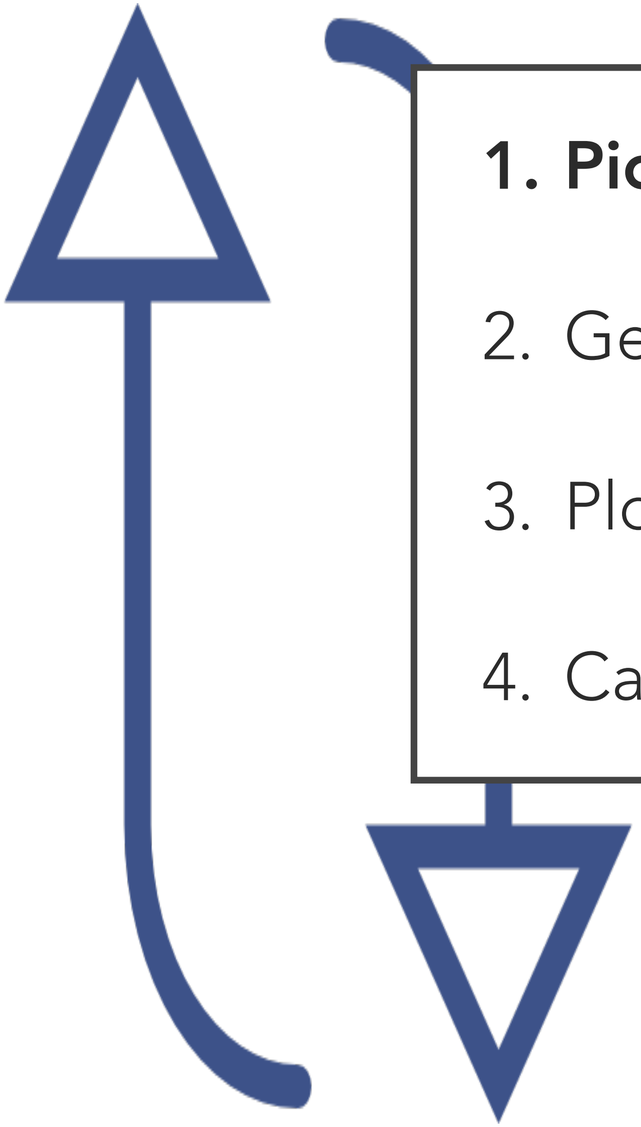
What is MC tuning?

What is MC tuning?

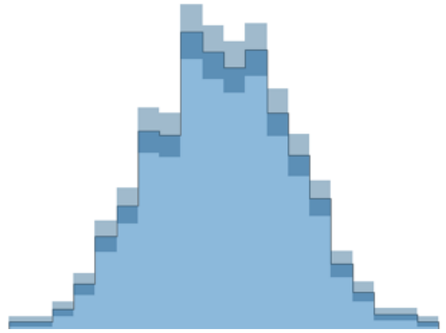
- MC is based on theoretical models
- There are free parameters in the theory

Constraining these parameters to take on values that are consistent with existing data is call “MC tuning”.

Basic MC tuning recipe



1. **Pick set of parameter values**



2. Generate MC events

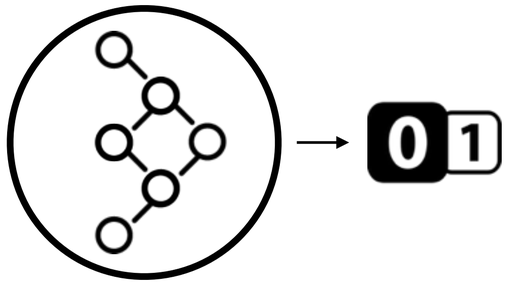
3. Plot 1D observables from MC and reference data events

4. Calculate $\chi^2(\boldsymbol{p}) = \sum_{\mathcal{O}} \sum_{b \in \mathcal{O}} w_b \frac{(f^{(b)}(\boldsymbol{p}) - \mathcal{R}_b)^2}{\Delta_b^2}$

NN-based high-dimensional data-MC comparison.

Fundamental Approach

- Question: does $p_{\text{generator}}(x|\theta)$ match $p_{\text{data}}(x)$?
- Instead of checking 1D histograms, let a NN do this for you in high-dimensions (accounts for correlations)



☑ Can still plot 1D distributions and calculate χ^2 to cross-check NN performance vs. your physics intuition

Experimental Setup

- Dijet events at in *pp* collisions at 13 TeV, produced with `Pythia 8.230` via the `numpythia` package in scikit-hep (500 GeV < p_T < 1000 GeV)
- Jet clustering with `FastJet` via the `pyjet` package in scikit-hep (anti-k_T R=0.4; jet p_T > 10 GeV; select 2 leading jets)
- Specify μ_{FSR} (multiplier on renormalization scale for final-state radiation) variations
- Write out to HDF5
- Train binary classifiers using PyTorch for each variation vs baseline
- Method sensitivity in terms of binary ROC AUC

Automatic Pythia Reweighting

No need to regenerate x′ for each value of θ

Automated Variations of Shower Parameters for Uncertainty Bands

While a number of different central "tunes" of the Pythia parameters are provided, it is often desired to study how event properties change when some of the parameters (such as those describing the parton showers) are varied. Pythia8 now has the ability to provide a series of weights to reflect the change in probability for a particular final state to occur when a subset of parton-shower parameters are varied. Details on the implementation and interpretation of these weights can be found in [Mre16]. Currently, the list of available automated variations (see full list below) includes:

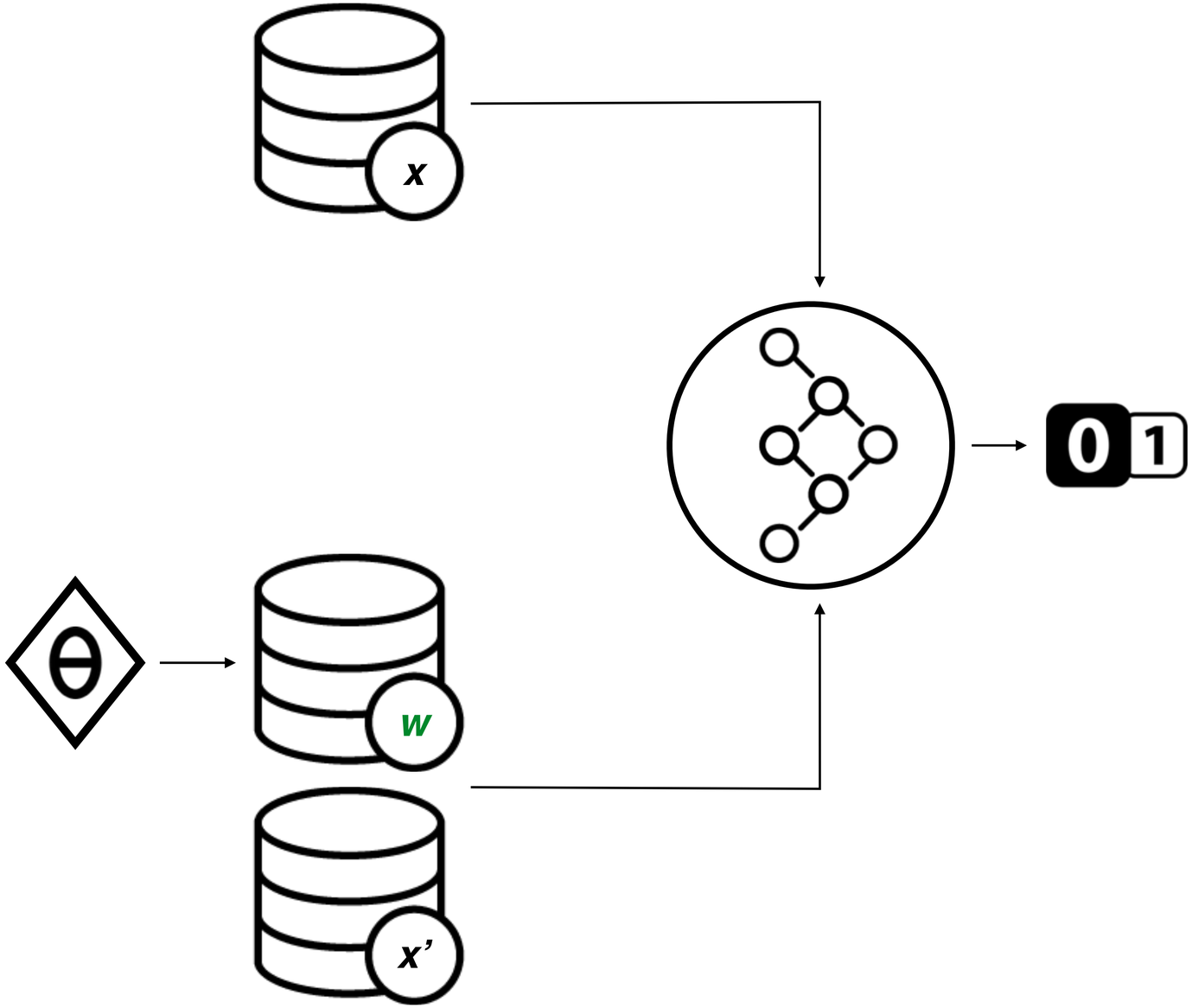
- The renormalization scale for QCD emissions in FSR;
- The renormalization scale for QCD emissions in ISR;
- The inclusion of non-singular terms in QCD emissions in FSR;
- The inclusion of non-singular terms in QCD emissions in ISR.
- The PDF members of a PDF family in LHAPDF6.
- Individual PDF members of a PDF family in LHAPDF6.

arXiv.org > hep-ph > arXiv:1605.08352

High Energy Physics - Phenomenology

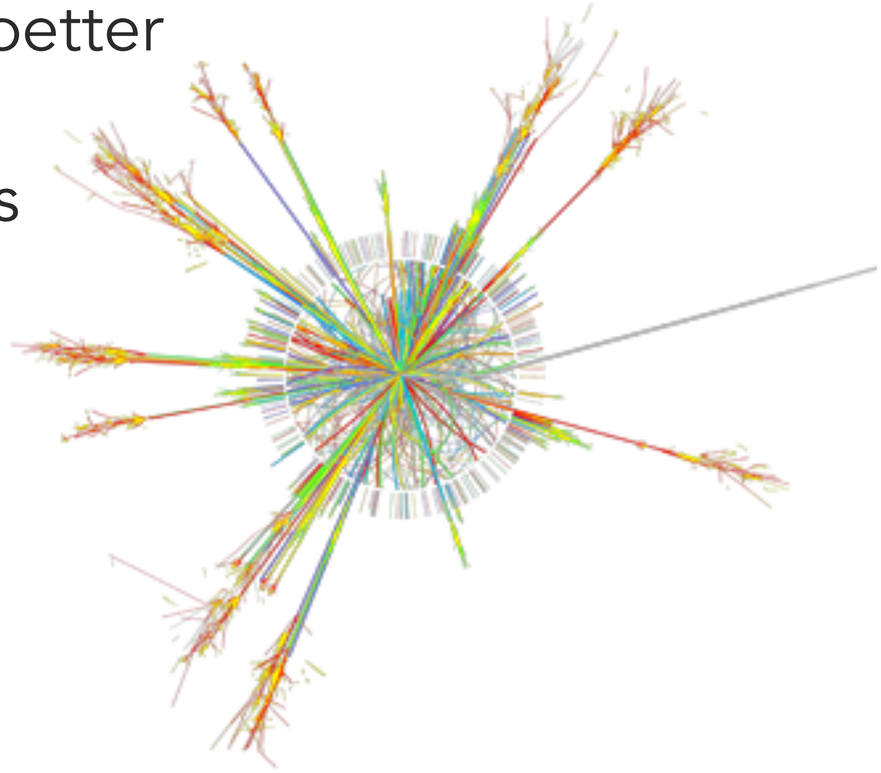
Automated Parton-Shower Variations in Pythia 8

S. Mrenna (1), P. Skands (2) ((1) Computing Division, Fermilab, Batavia, IL USA (2) School of University, Victoria, Australia)

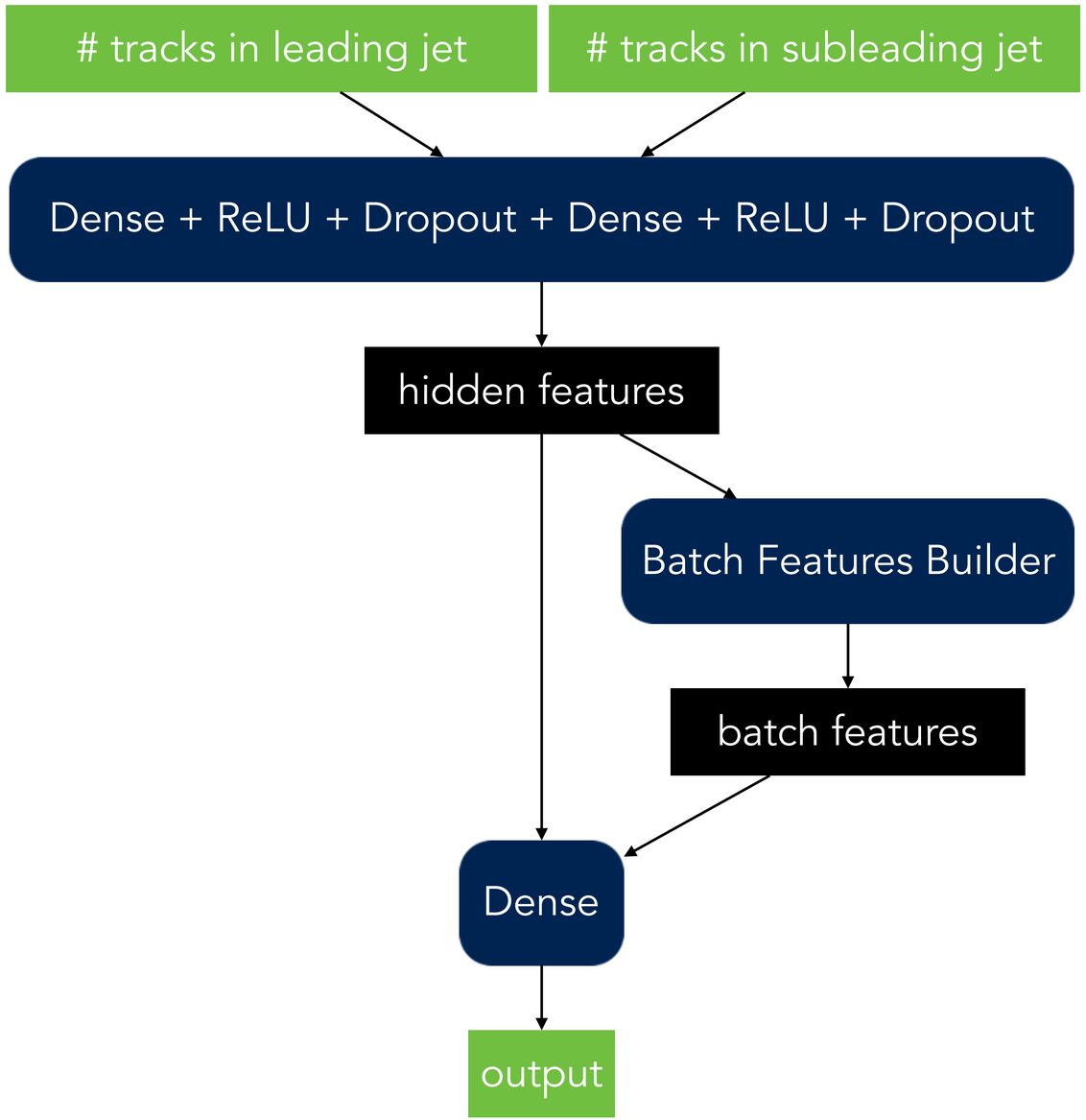


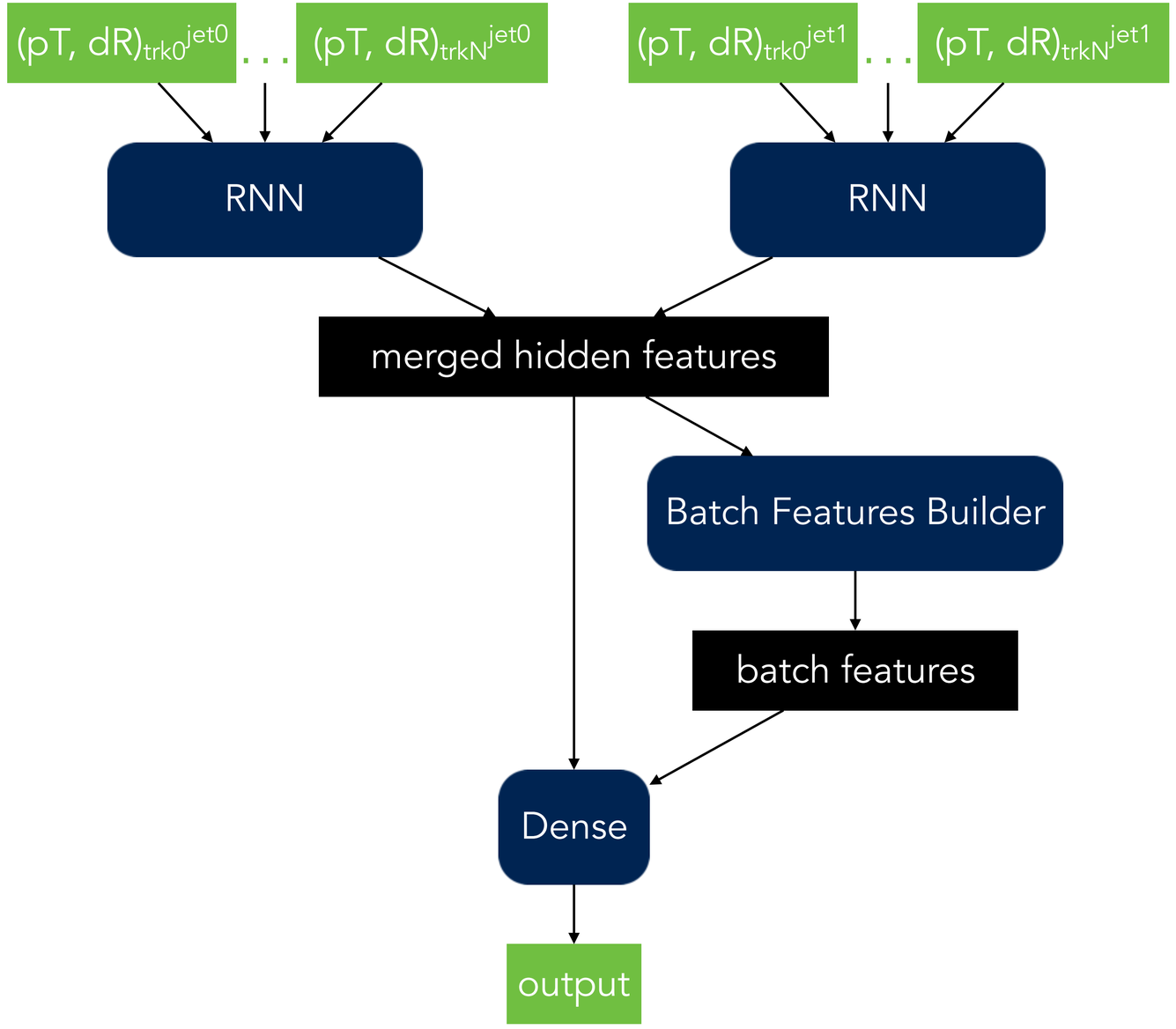
Use more complex input features (full radiation inside jets)

- Letting a NN compare distributions scales better
- Can use much more complex inputs such as properties of all particles in the event



Network Architecture





Batch Features

- Classifiers sees **pure batches** from either dataset
- Prior on how to classify an event should be updated based on batch characteristics
- Could use mini *batch discrimination* (standard in GAN discriminators) but intra-batch distance is not informative here
- Instead, calculate batch summary stats like mean and std of hidden features

```
batch_mean = batch_weights.mm(hidden).expand(batch_size, hidden.shape[-1])
batch_second_moment = batch_weights.mm(torch.pow(hidden, 2)).expand(batch_size, hidden.shape[-1])
batch_std = batch_second_moment - torch.pow(batch_mean, 2)
all_features = torch.cat([batch_mean, batch_std, hidden], 1)
```

Account for uncertainties.

Uncertainties in the AUC calculation

- Each NN is trained on the same train set, validated on the same validation set, and tested on the same test set

- *Statistical uncertainty*: due to the finite size of the test set

from literature:
The meaning and use of the area under a ROC curve,
The area above the ordinal dominance graph and the area below the receiver operating characteristic graph, …

The formula for SE(AUC) was given by Hanley and McNeil (1982) is

$$SE(AUC) = \sqrt{\frac{AUC(1-AUC) + (N_1 - 1)(Q_1 - AUC^2) + (N_2 - 1)(Q_2 - AUC^2)}{N_1 N_2}}$$

where

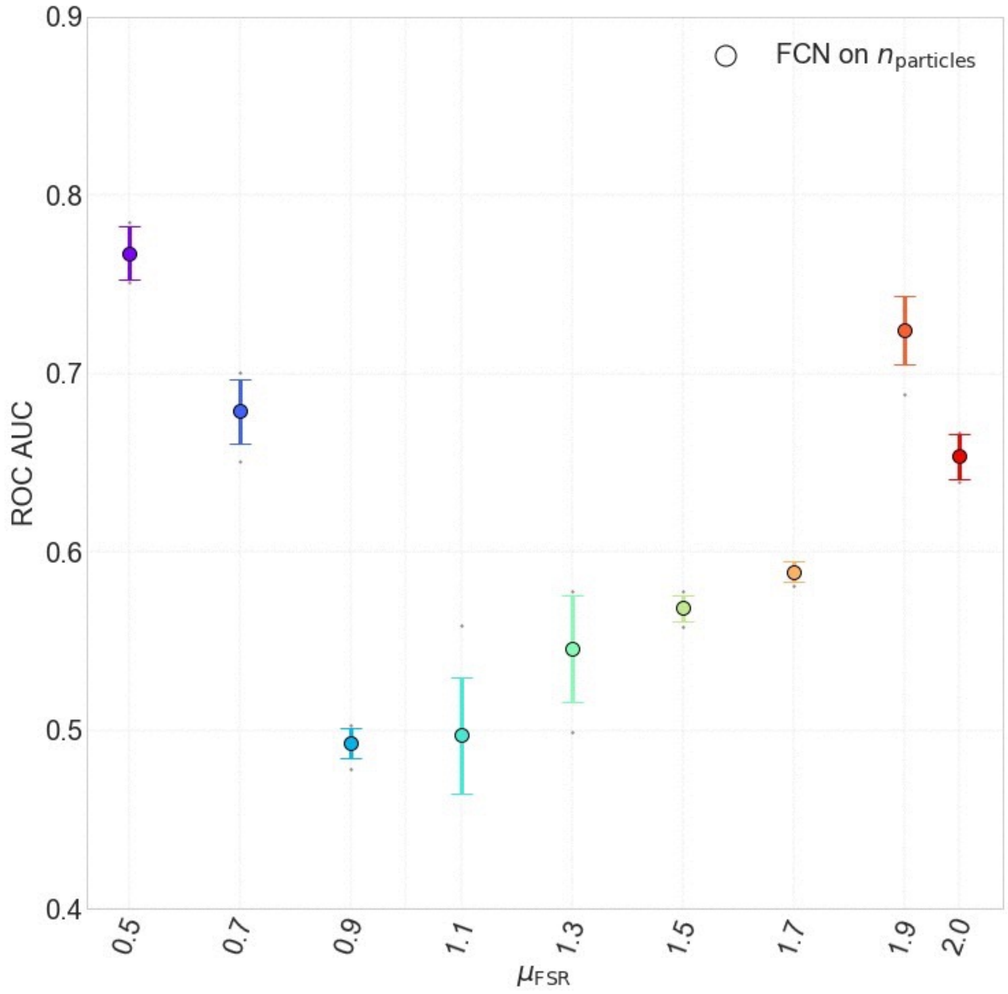
$$Q_1 = \frac{AUC}{2 - AUC}$$

$$Q_2 = \frac{2AUC^2}{1 + AUC}$$

- *Systematic uncertainty*: due to the randomness in building batches —> test on 10 different reshufflings

Binary Classification Tasks vs. μ_{FSR}=1.0 Baseline

Start by training a classifier per mu value:

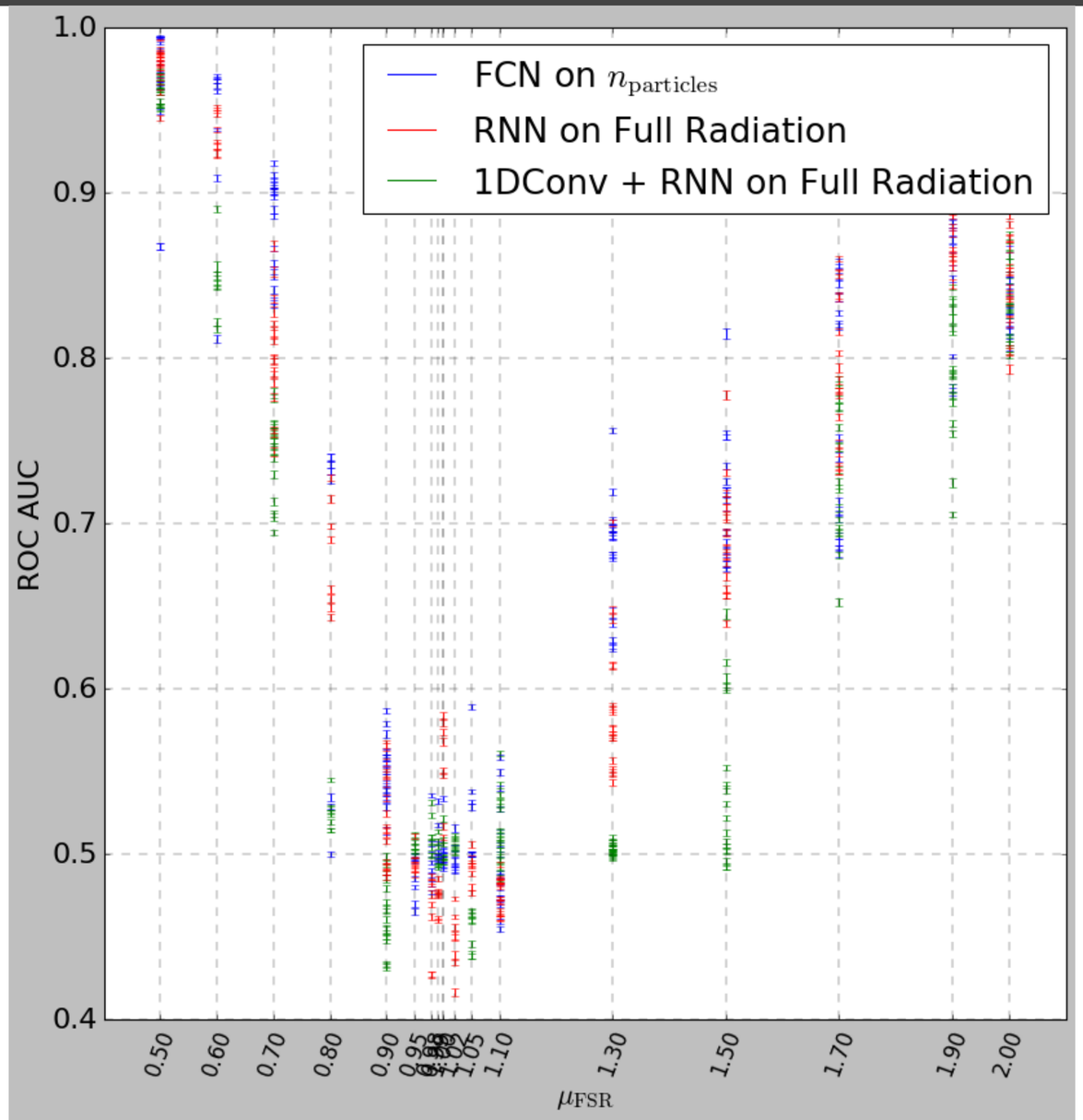


How reliable is it?

If I train another time, will I get the same performance? (unlikely)

the majority of variation comes from the randomness of the training process

Binary Classification Tasks vs. μ_{FSR}=1.0 Baseline



Procedure

Cannot assume the AUCs are normally distributed for each mu value. So:

- Smear individual AUCs to represent their stat. and syst. uncertainties
- Bootstrap the mean AUC per mu value & assign an error bar that corresponds to the std of the bootstrapped distributions
- Fit a curve, find its minimum, propagate uncertainties from the covariance metric of the best-fit parameters to the minimum

Need to examine how choice of fit function and fit range affect the goodness of the fit and the estimate of the minimum.

Then: Tune Multiple Parameters at Once

```yaml
# Example configuration for Pythia in YAML.

Beams:eCM: 13000
Next:numberShowInfo: 0
Next:numberShowEvent: 0
Next:numberShowLHA: 0
Next:numberShowProcess: 0
Next:numberCount: 0
HardQCD:all: on
PhaseSpace:pTHatMin: 500
PhaseSpace:pTHatMax: 1000
UncertaintyBands:doVariations: on
Proposals:
  fsr:muRfac:
    dist: scipy.stats.uniform
    kwargs:
      loc: 0.5
      scale: 1.5
    min: 0
    max: 130
  isr:muRfac:
    values:
      - 0.1
      - 0.2
      - 0.9
NumSamples: 4
Mode: sample # or grid
```

Automatically generates Pythia config file with all the possible variations

parameter names

values drawn from distribution

individual values selection

Goal

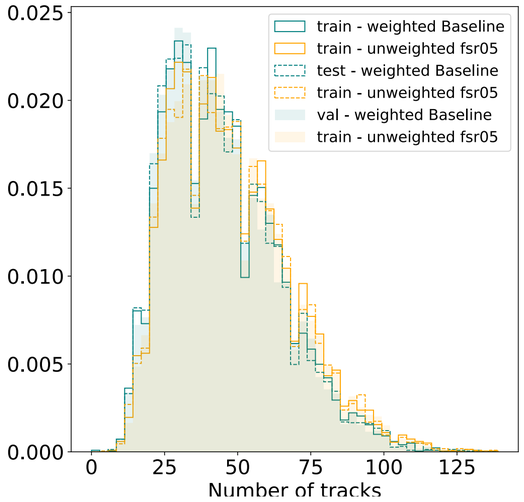
- Obtain a new tune for these Pythia8 parameters which is hopefully more precise than previous methods

Thank you!

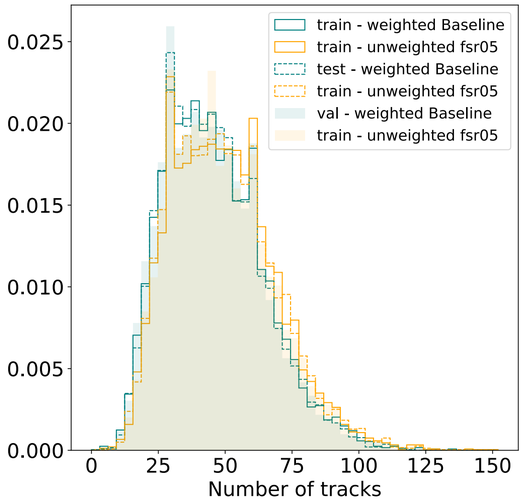
Questions?

You can find me at: michela.paganini@yale.edu

Baseline versus mu = 0.5

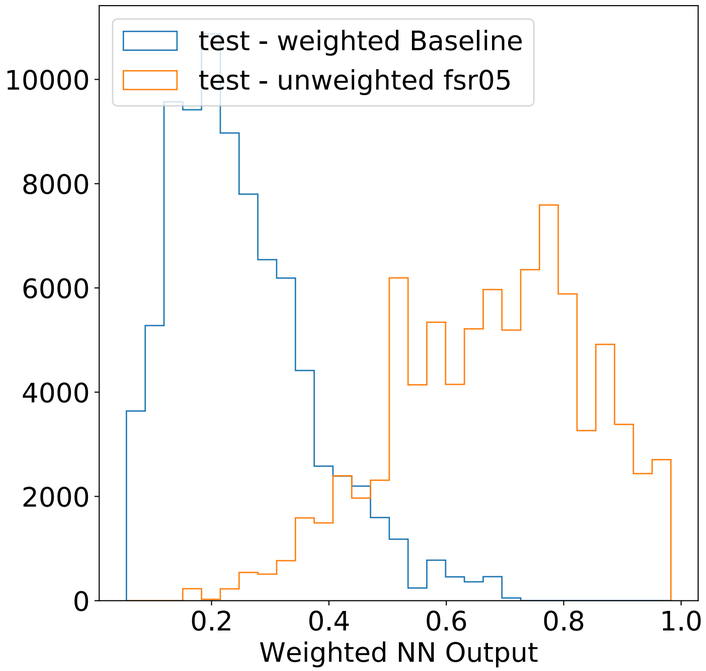


in leading jet



in subleading jet

Output of fully connected network on the two ntrack variables



Output of the RNN on the track level variables