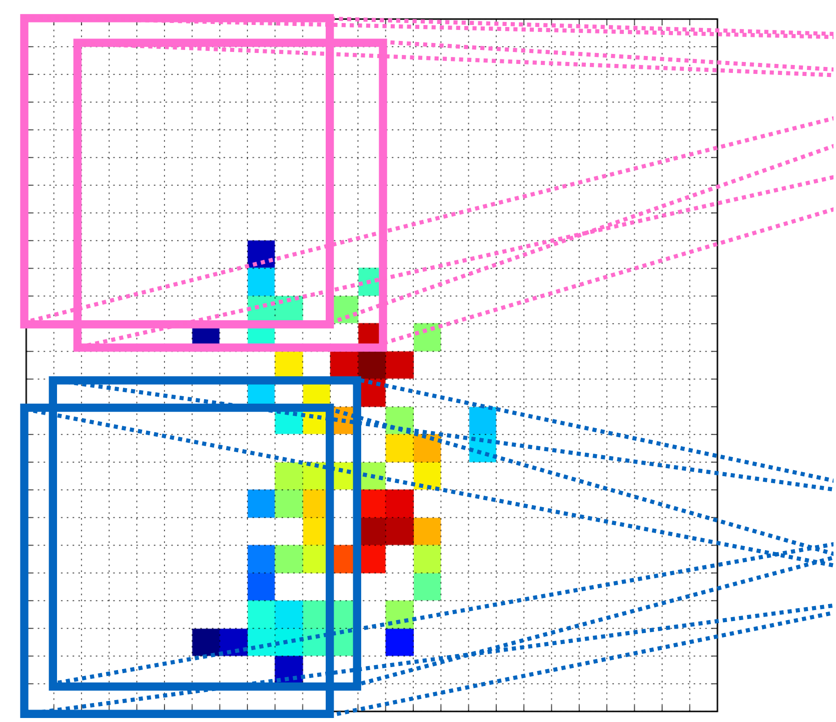
# Reducing our dependence on slow simulators with deep learning

Benjamin Nachman

*Lawrence Berkeley National Laboratory*



BERKELEY LAB

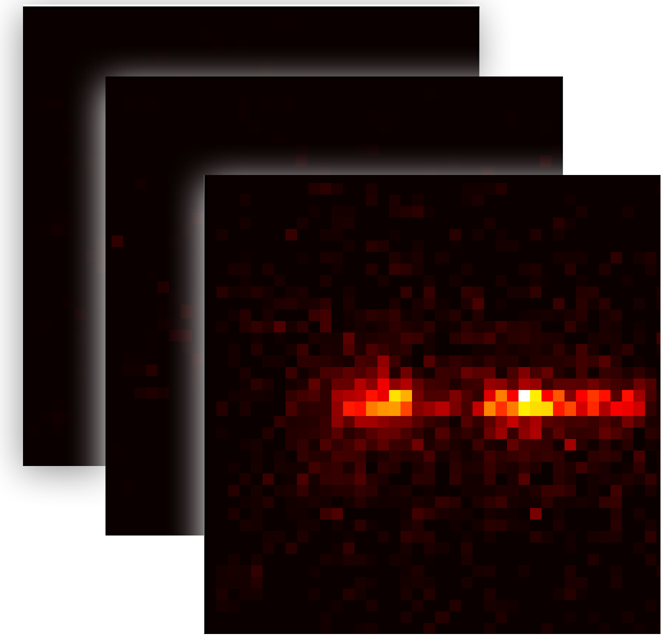
BERKELEY EXPERIMENTAL PARTICLE PHYSICS

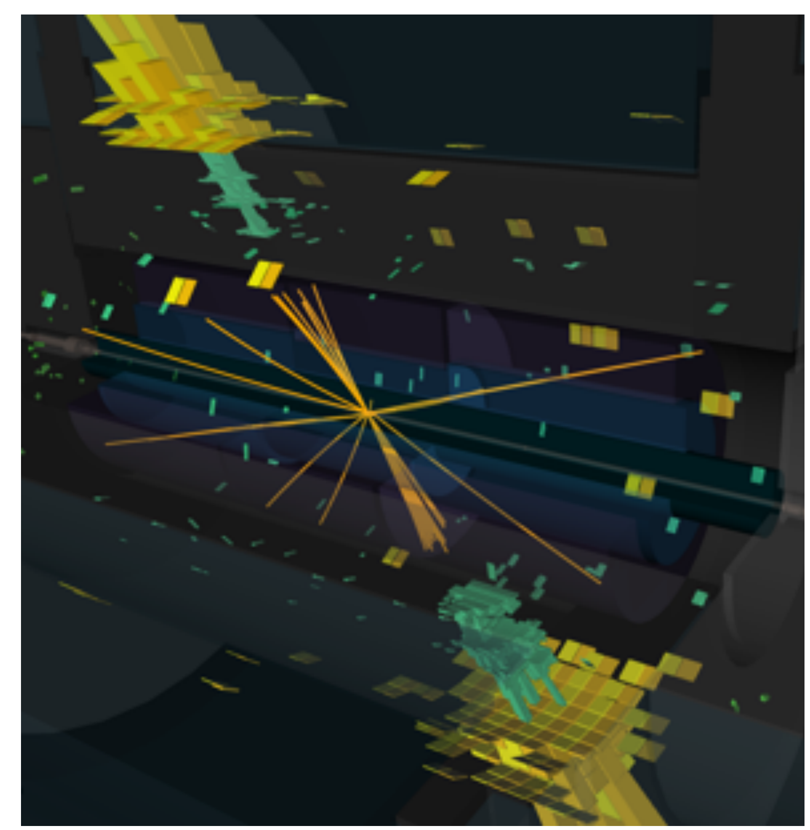
*TRISEP School*

*August 2, 2019*

# ML for classification and beyond

There is a lot more ML can do than classify examples!





**Classification**

provide examples for training

arbitrarily many categories

**Generation** ← **Regression**

map noise to structure

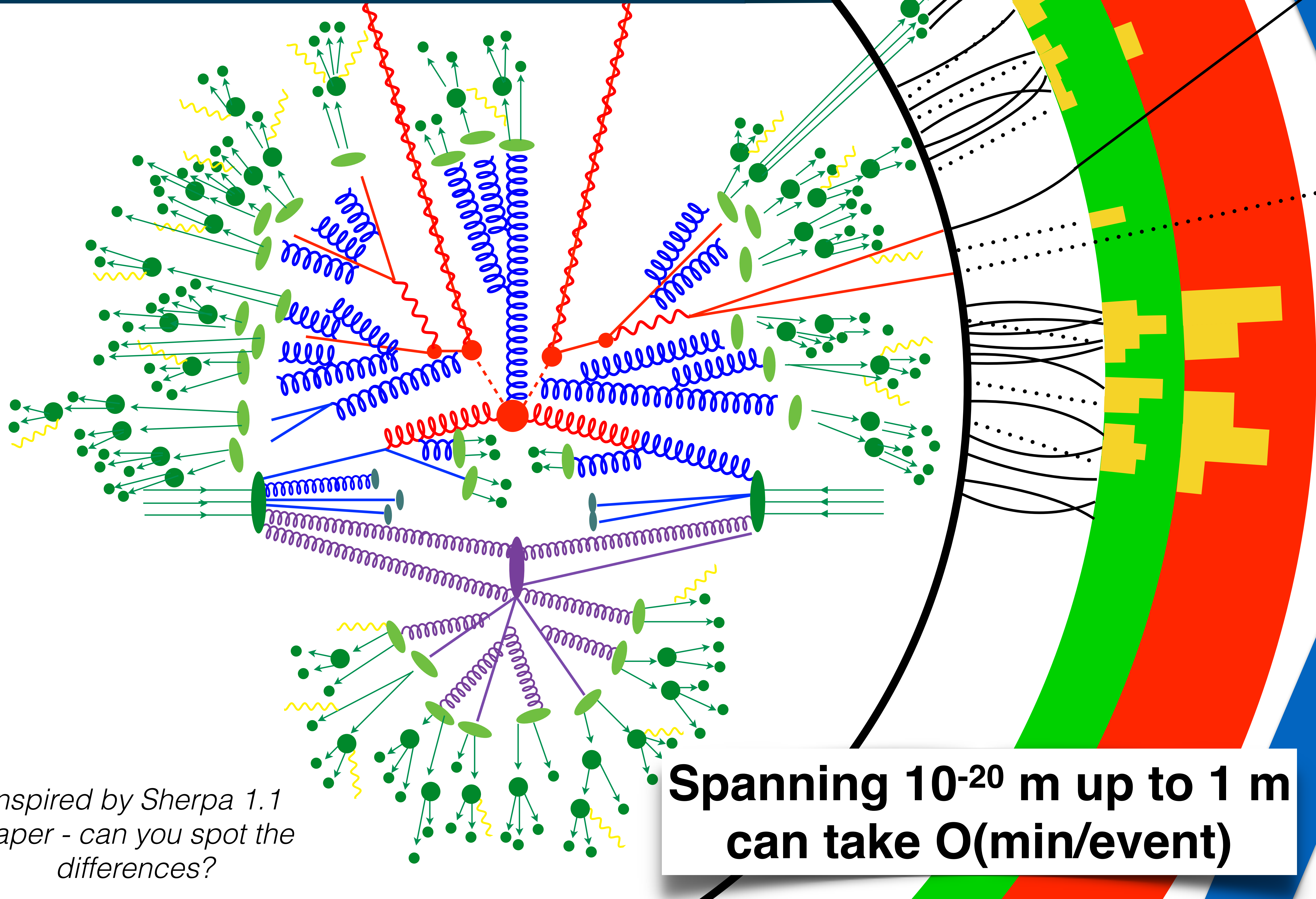
# Outline



- Simulation dependence in collider HEP
- **Generative models**: accelerate simulations with deep learning
- **Generalized numerical inversion**: machine learn with prior independence
- **Weak supervision**: learn directly from (unlabeled) data
- CWoLa hunting: model agnostic **anomaly detection**

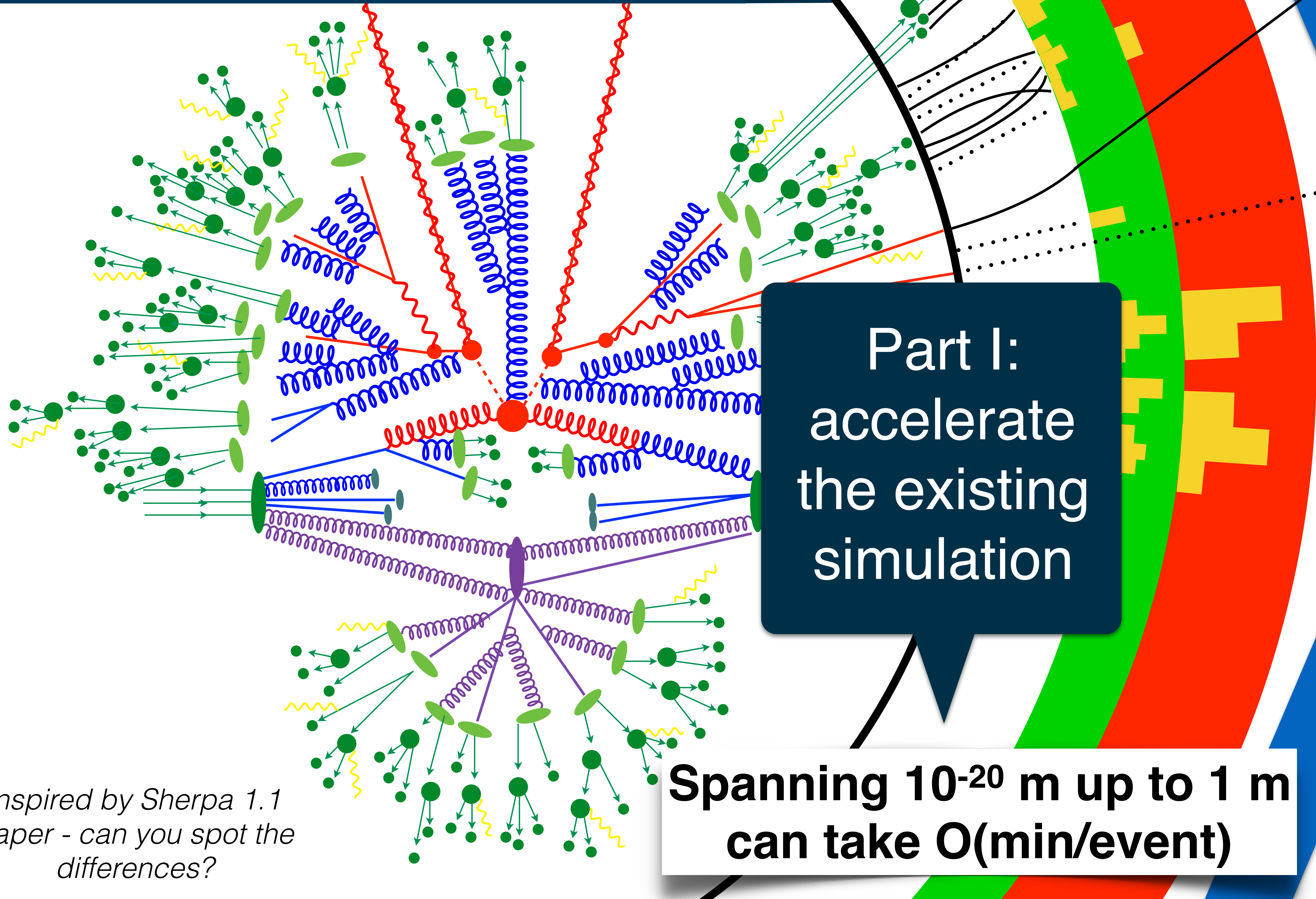
# Simulation at the LHC






*Inspired by Sherpa 1.1 paper - can you spot the differences?*

# Simulation at the LHC



Part I: accelerate the existing simulation

**Spanning $10^{-20}$ m up to 1 m can take O(min/event)**

*Inspired by Sherpa 1.1 paper - can you spot the differences?*

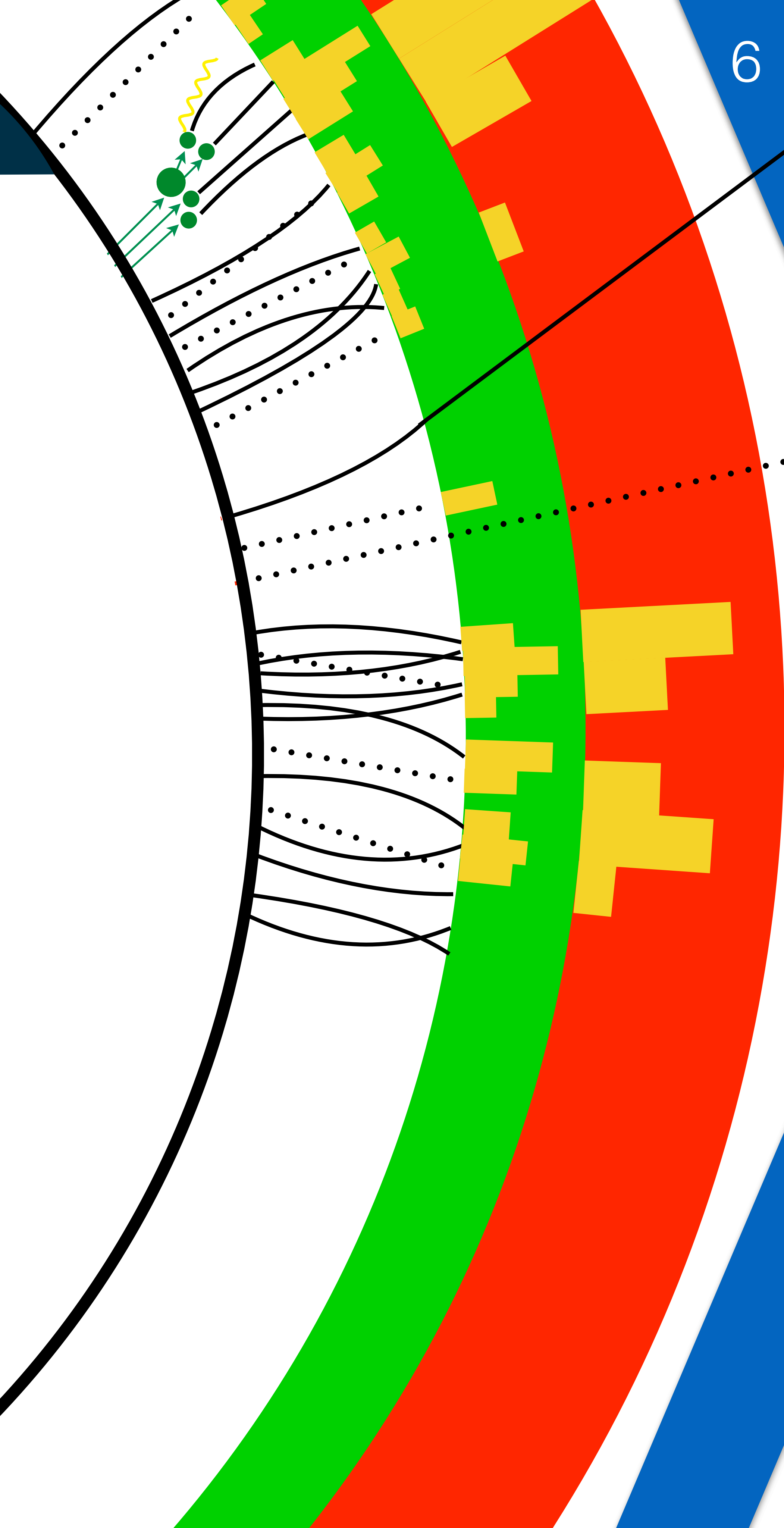
# Material Interactions

State-of-the-art for material interactions is Geant 4.

Includes electromagnetic and hadronic physics with a variety of lists for increasing/decreasing accuracy (at the cost of time)

*This accounts for O(1) fraction of all HEP computing resources!*

Geant 4



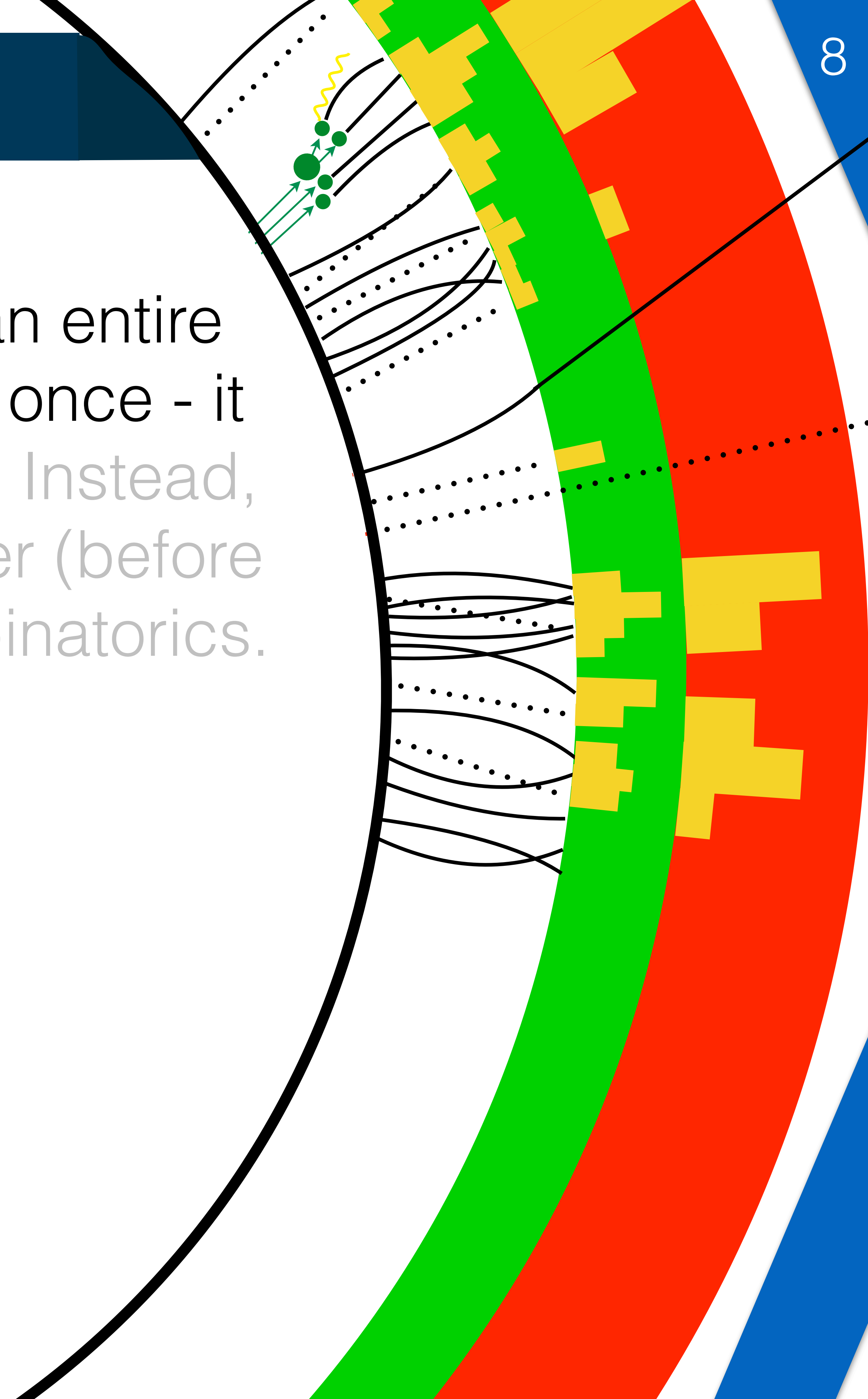
Goal: replace (or augment) simulation steps with a faster, powerful generator based on state-of-the-art machine learning techniques

This work: attack the most important part:
**Calorimeter Simulation**

# Factorization



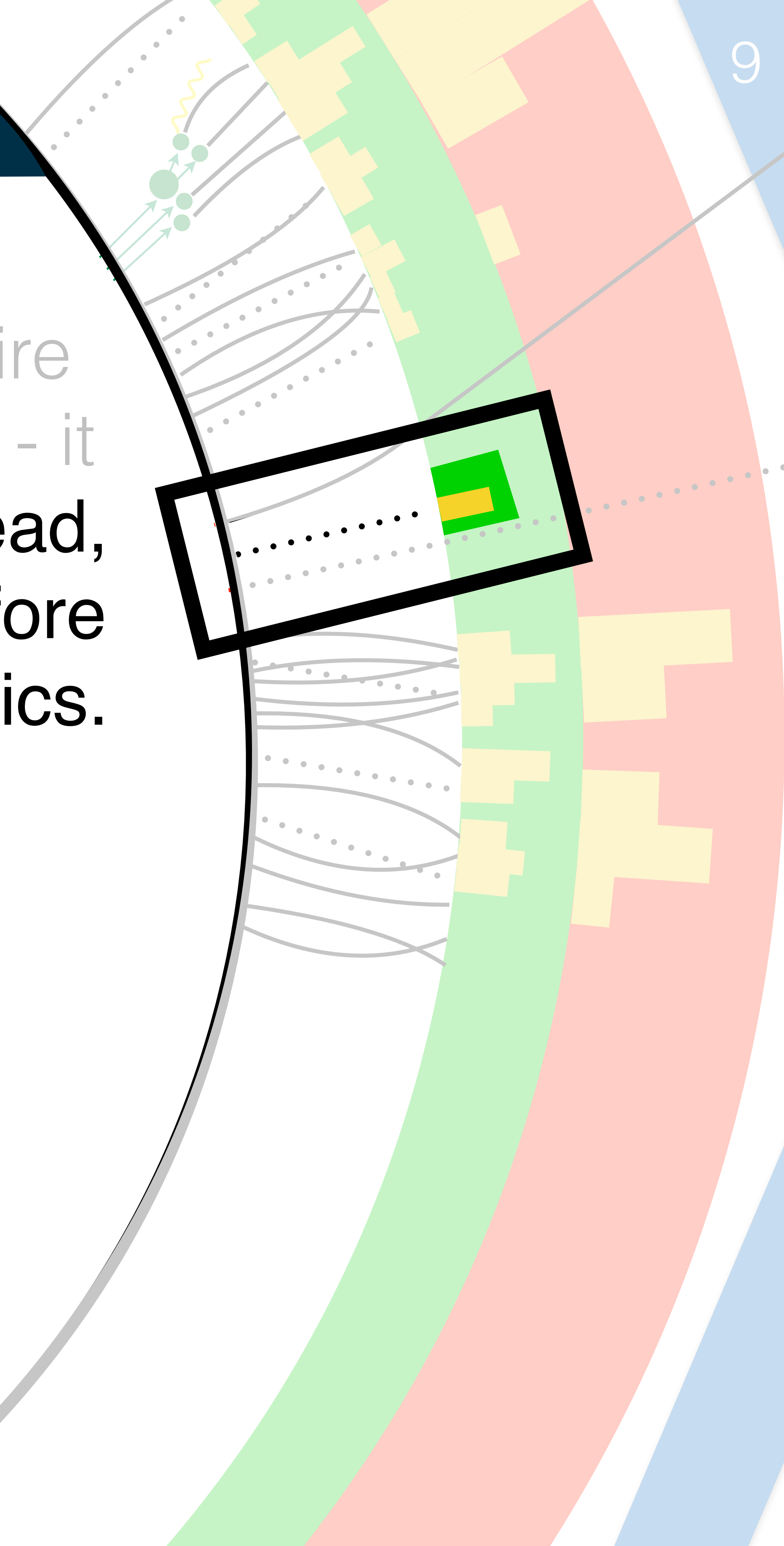
We are not trying to generate an entire event (O(1000) particles)) all at once - it would be **very had to validate!** Instead, generate a single particle shower (before electronics) and appeal to combinatorics.



# Factorization



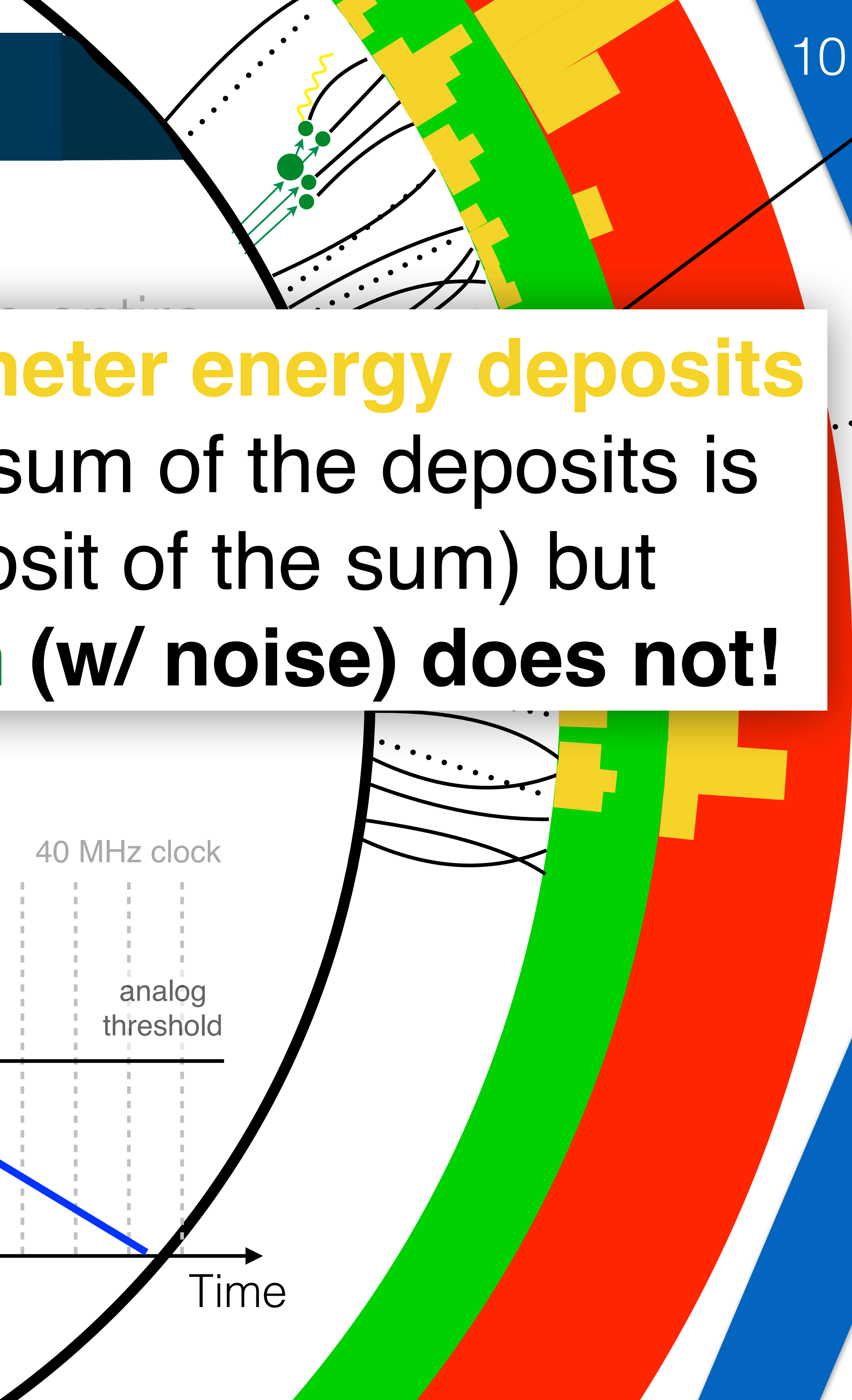
We are not trying to generate an entire event (O(1000) particles)) all at once - it would be **very had to validate!** Instead, generate a single particle shower (before electronics) and appeal to combinatorics.

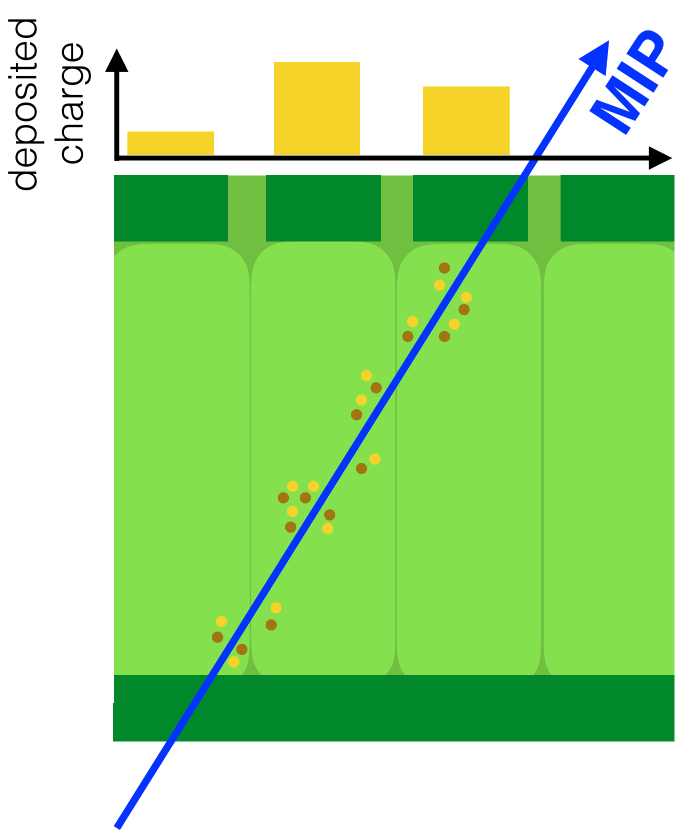


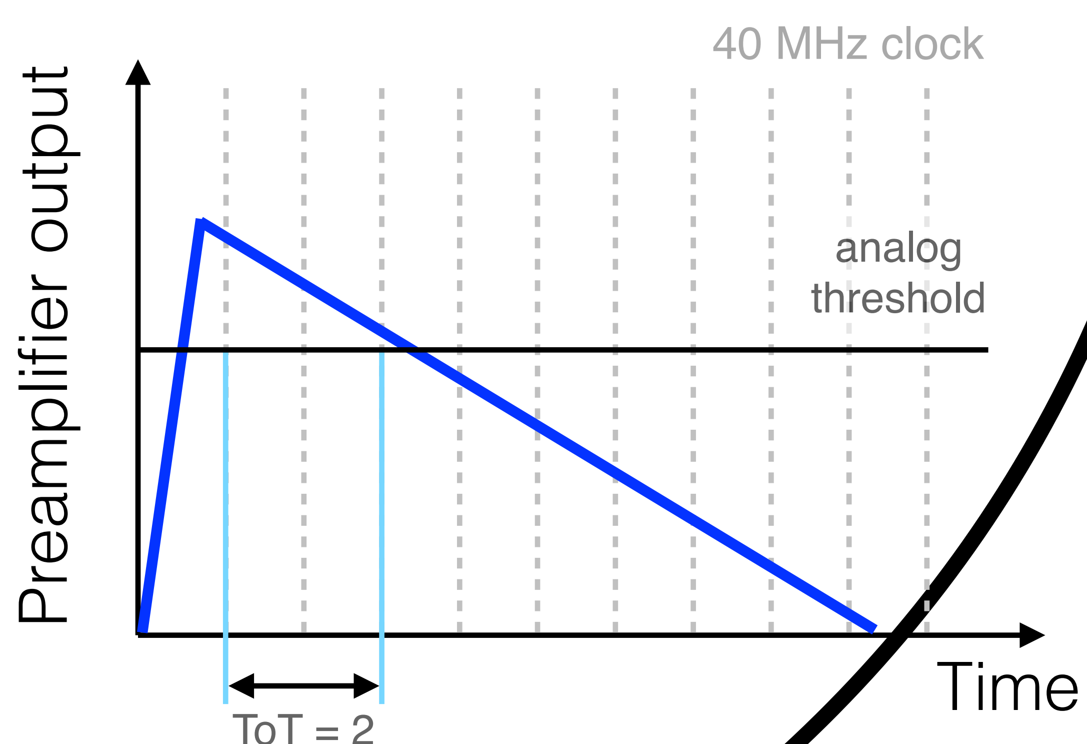
# Factorization

We are not tryin[g to generate an entire]
event (O(1000) p[...]
would be **very ha**[...]
generate a single[...]
electronics) and a[...]

N.B. **calorimeter energy deposits factorize** (sum of the deposits is the deposit of the sum) but **digitization (w/ noise) does not!**



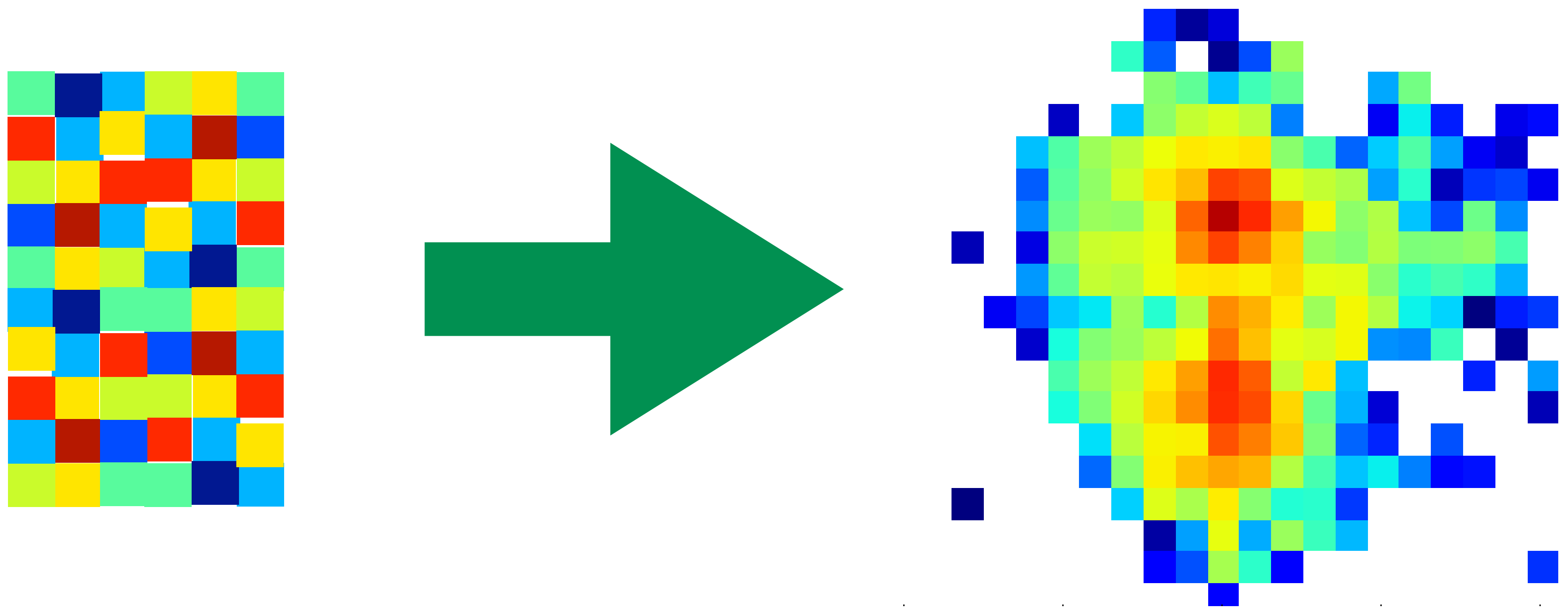


# Now to the machine learning



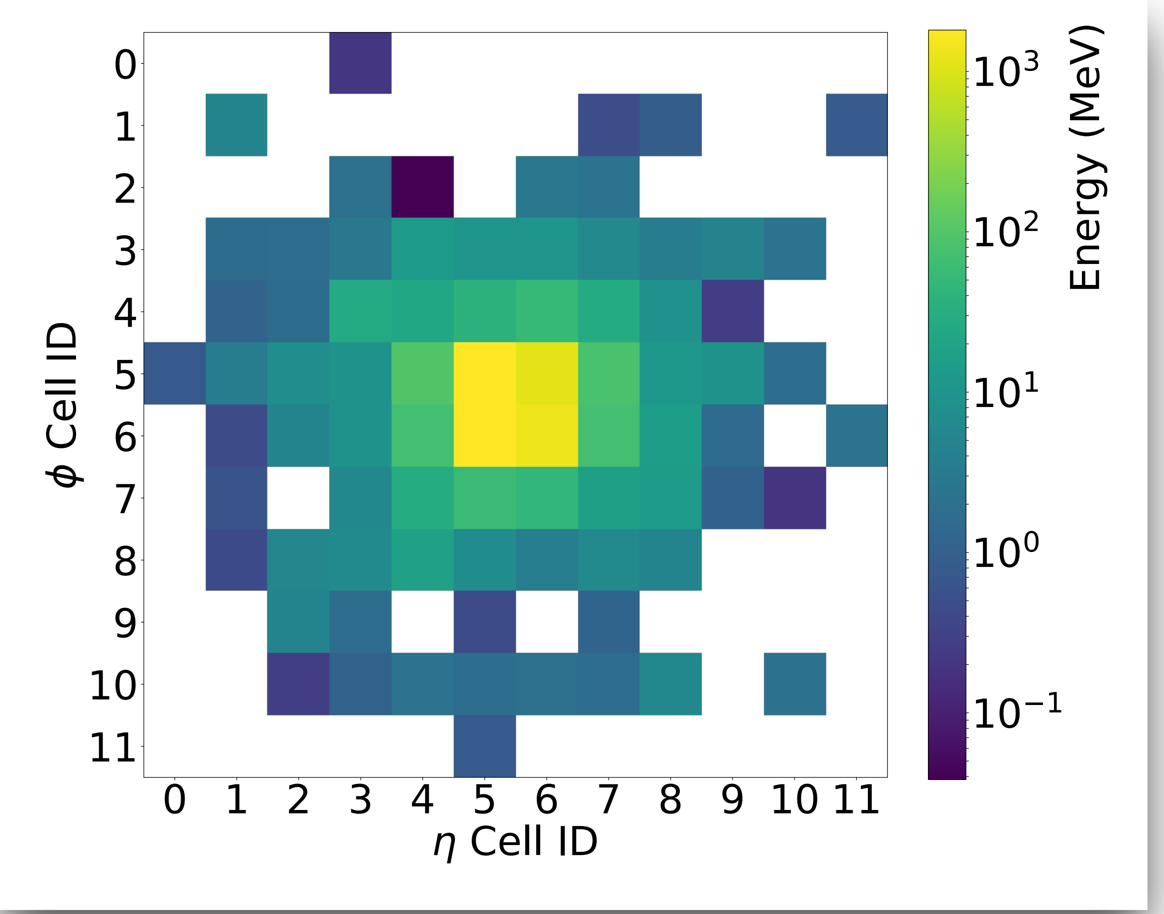
A **generator** is nothing other than a function that maps random numbers to structure.



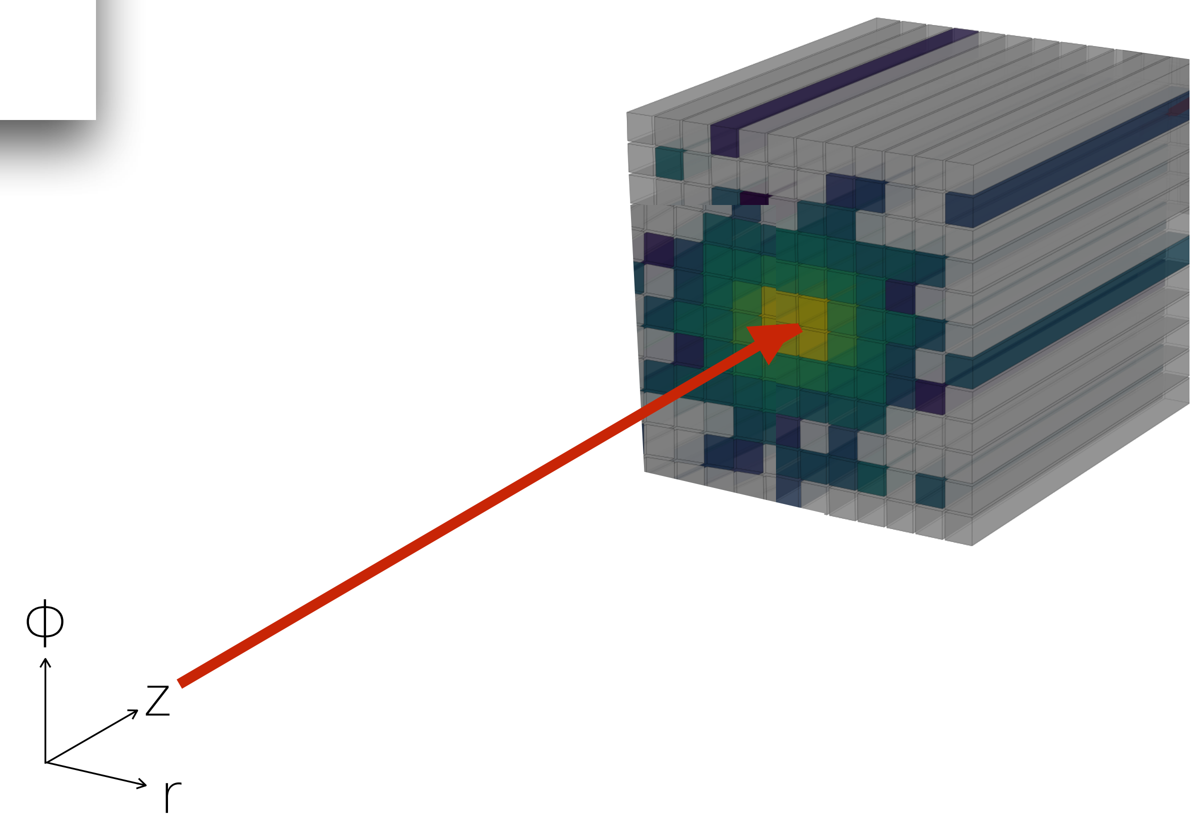
Our structure: **calorimeter images**

# Calorimeter images






Grayscale images:
Pixel intensity =
energy deposited

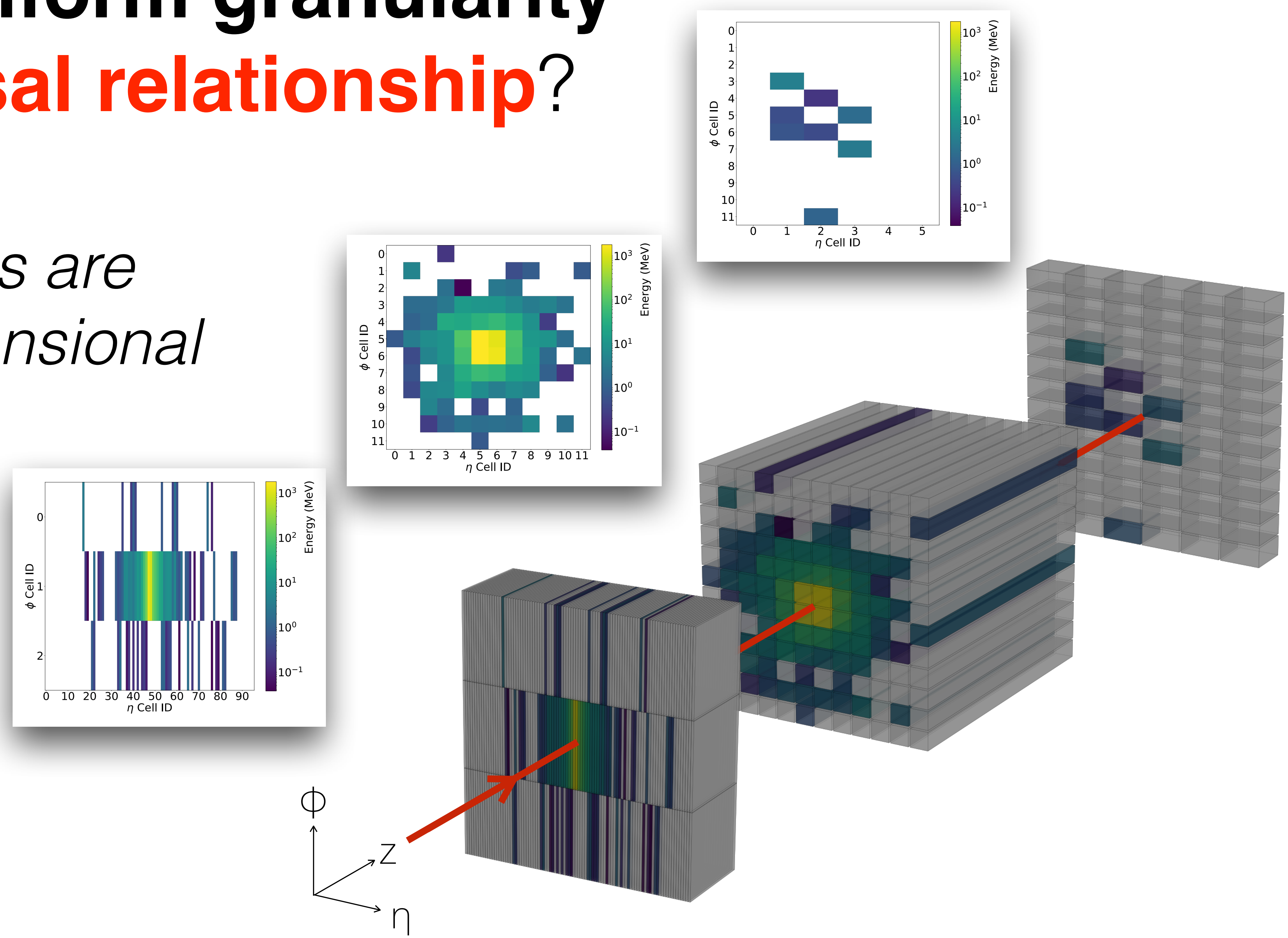


# Calorimeter images



Challenge: **multiple layers** with **non-uniform granularity** and a **causal relationship**?

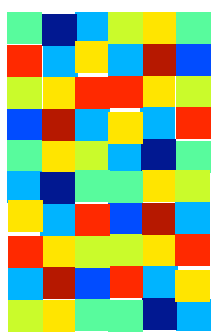
*N.B. images are O(1000) dimensional*



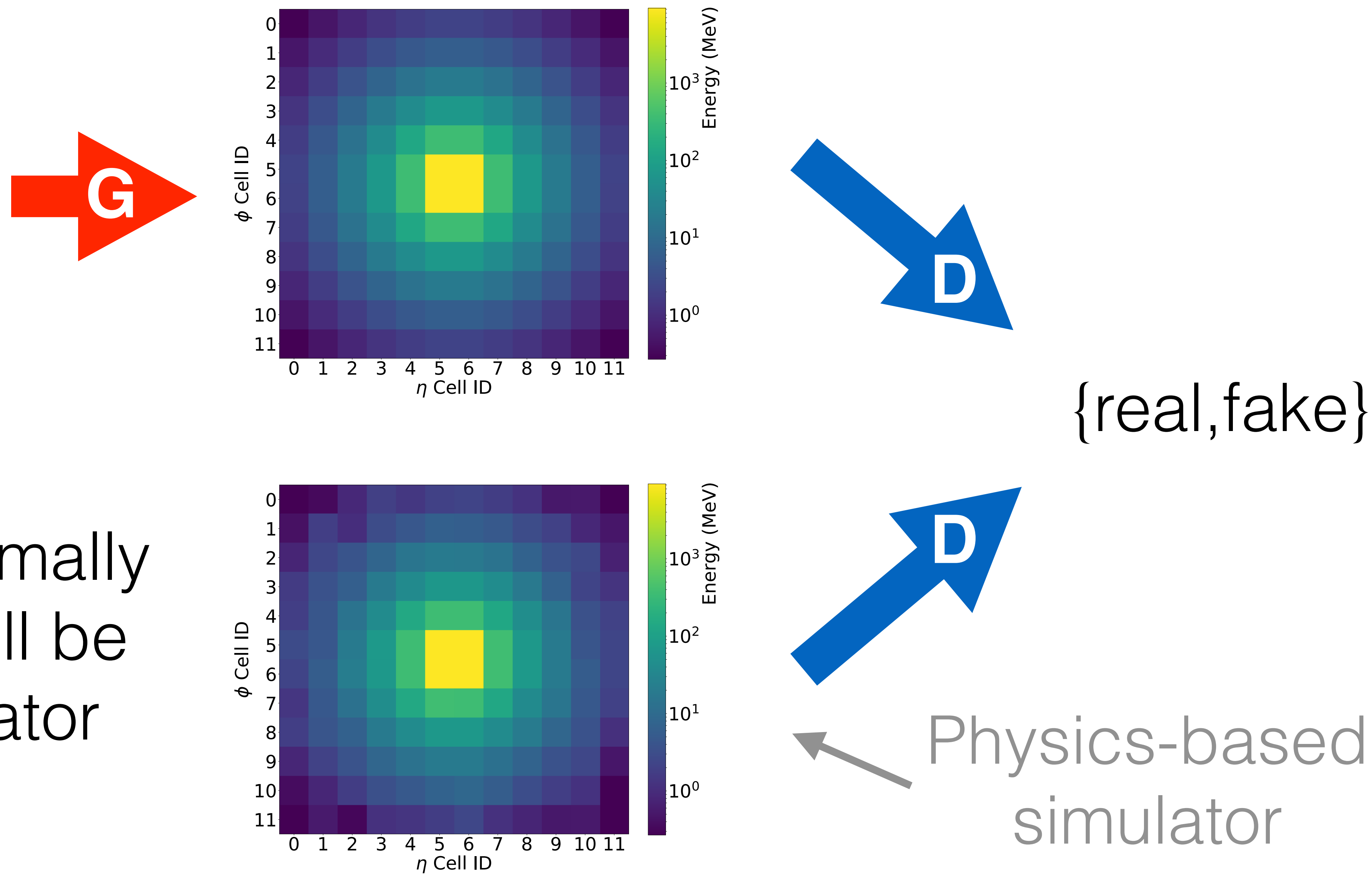
# One possibility: GANs



Generative Adversarial Networks (GAN):
*A two-network game where one **maps noise to images** and one **classifies images as fake or real**.*



noise



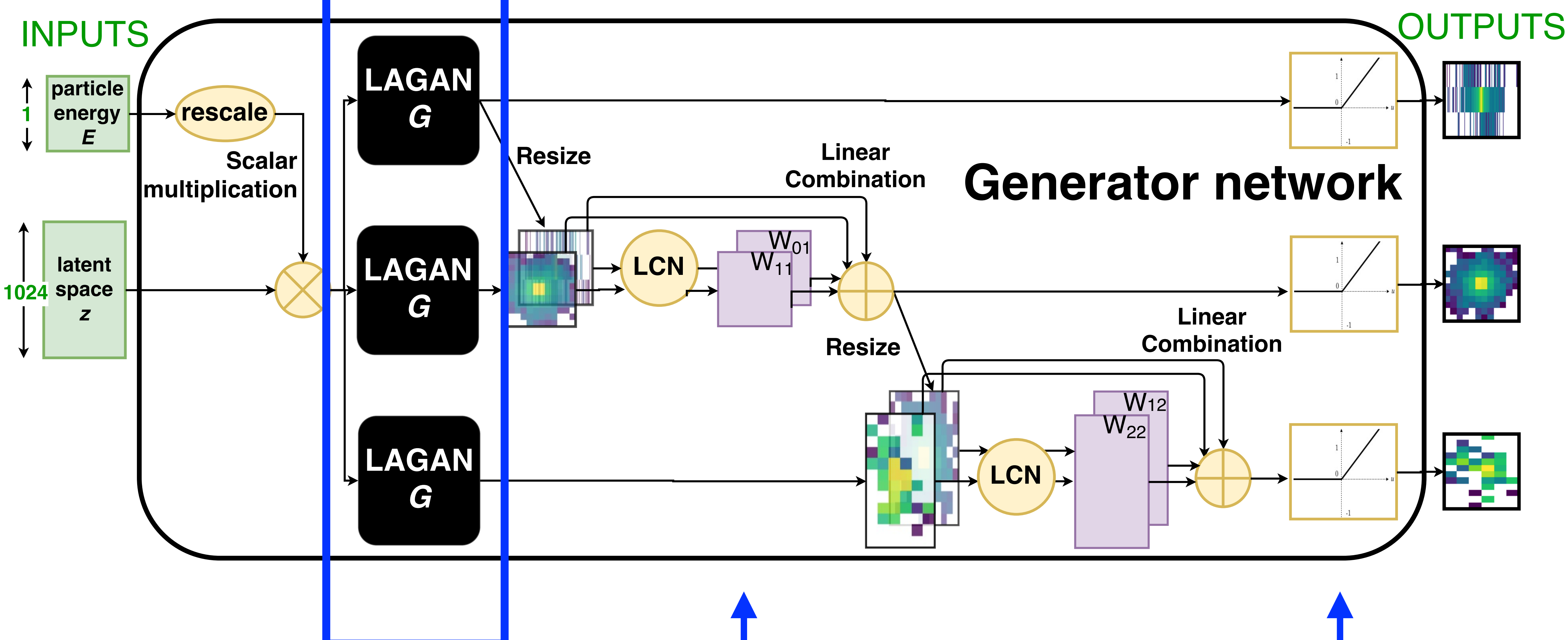

When **D** is maximally confused, **G** will be a good generator

# Introducing CaloGAN



One image per calo layer

One network per particle type; input particle energy




use layer i as input to layer i+1

ReLU to encourage sparsity

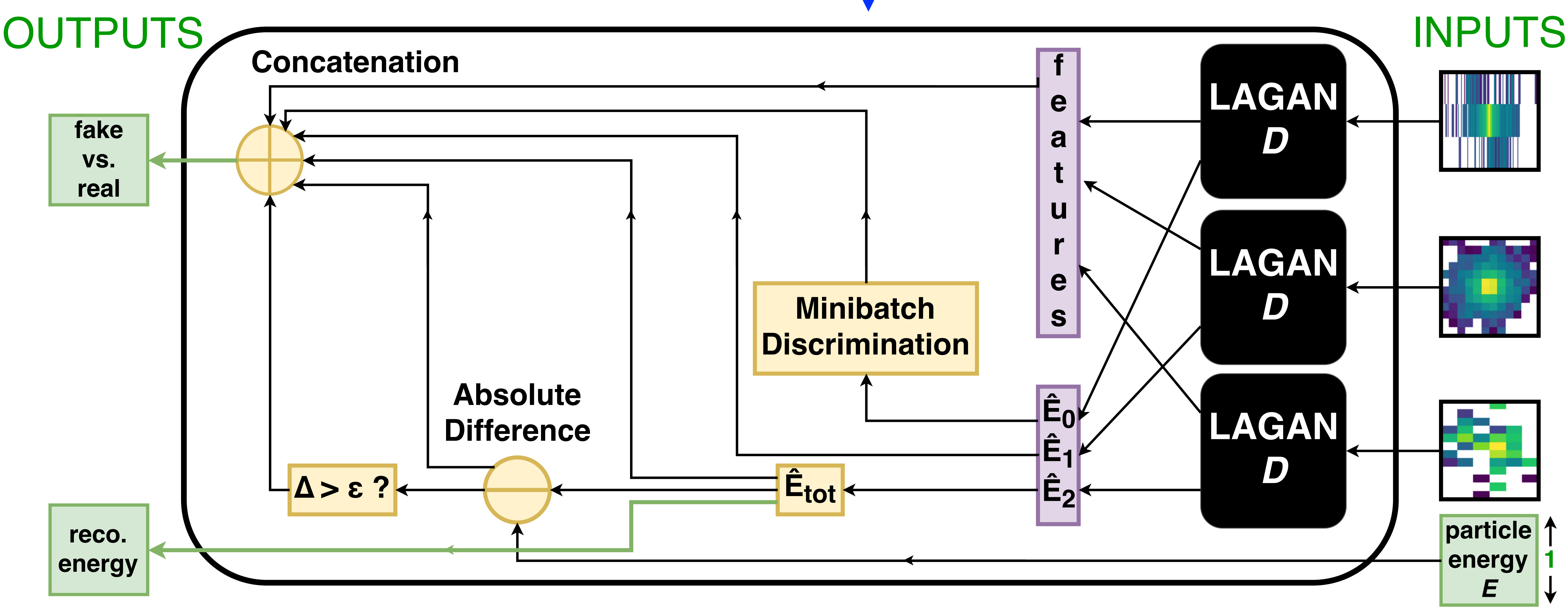
1705.02355

# Introducing CaloGAN



*Mode collapse: learns to generate one part of the distribution well, but leaves out other parts.*

help avoid 'mode collapse'



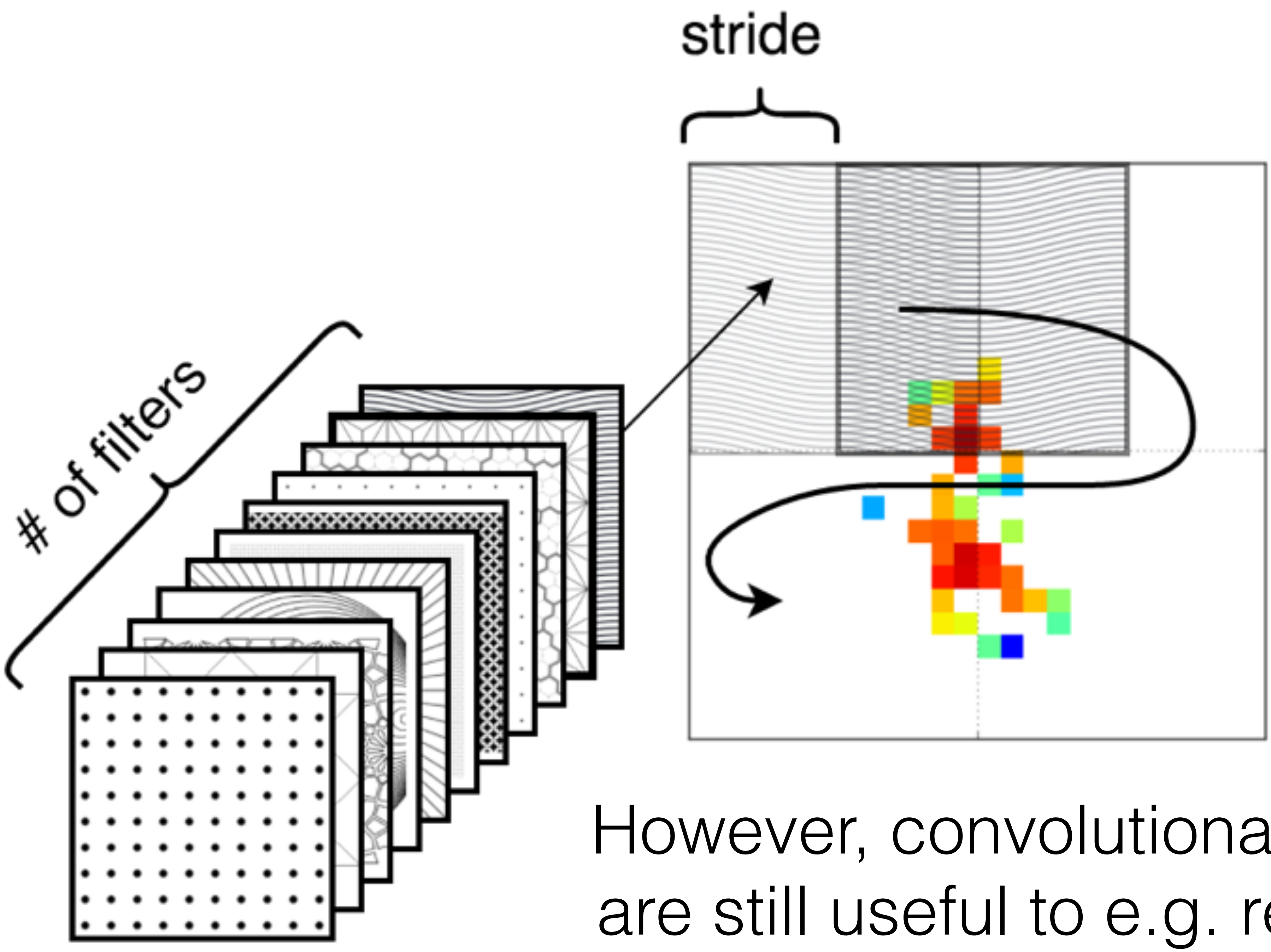

Discriminator network

# Locally connected layers



Due to the structure of the problem, we do not have translation invariance.

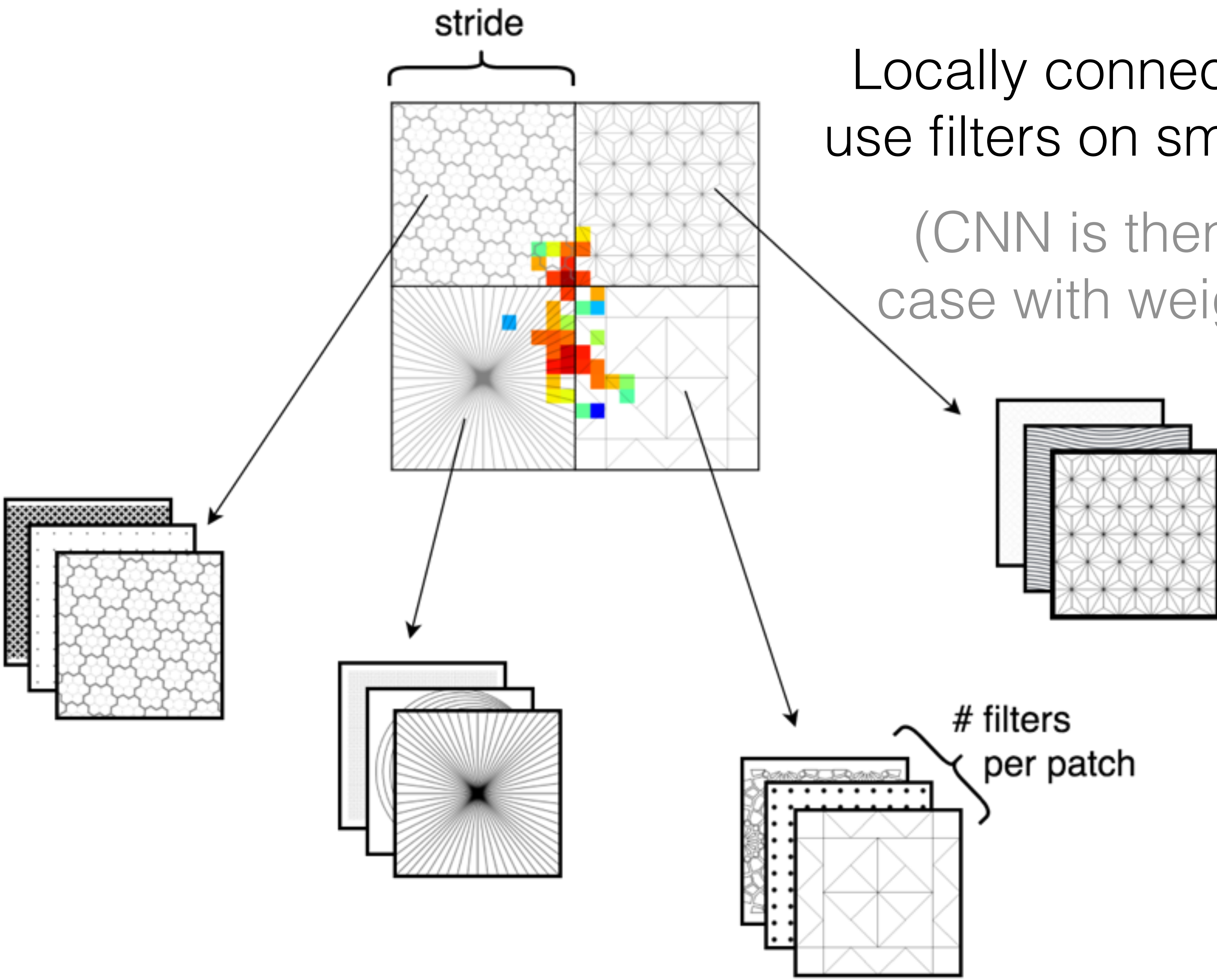
Classification studies found fully connected networks outperformed CNNs




However, convolutional-like architectures are still useful to e.g. reduce parameters

# Locally connected layers



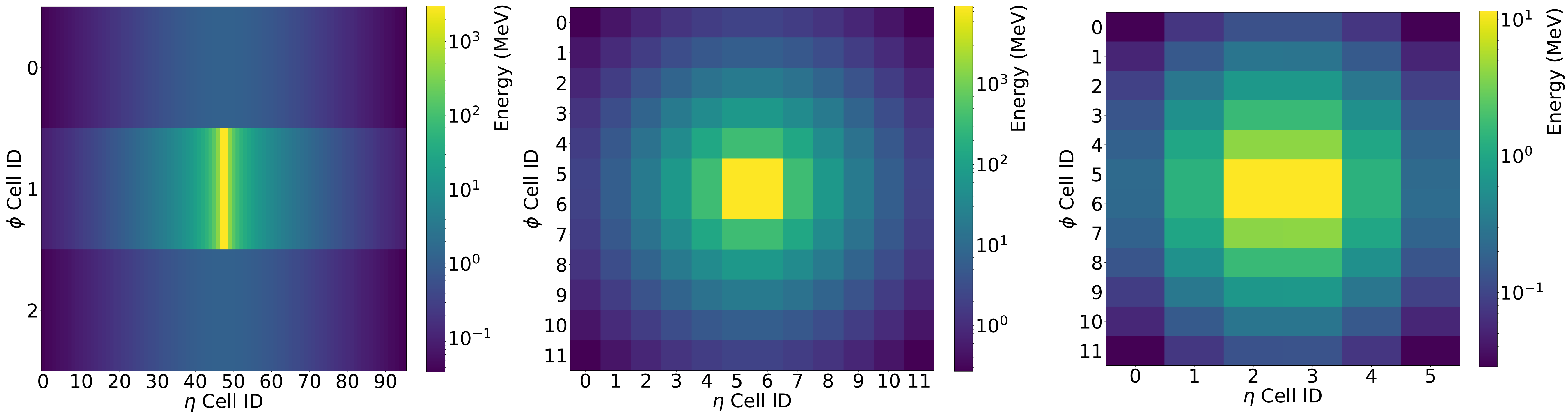


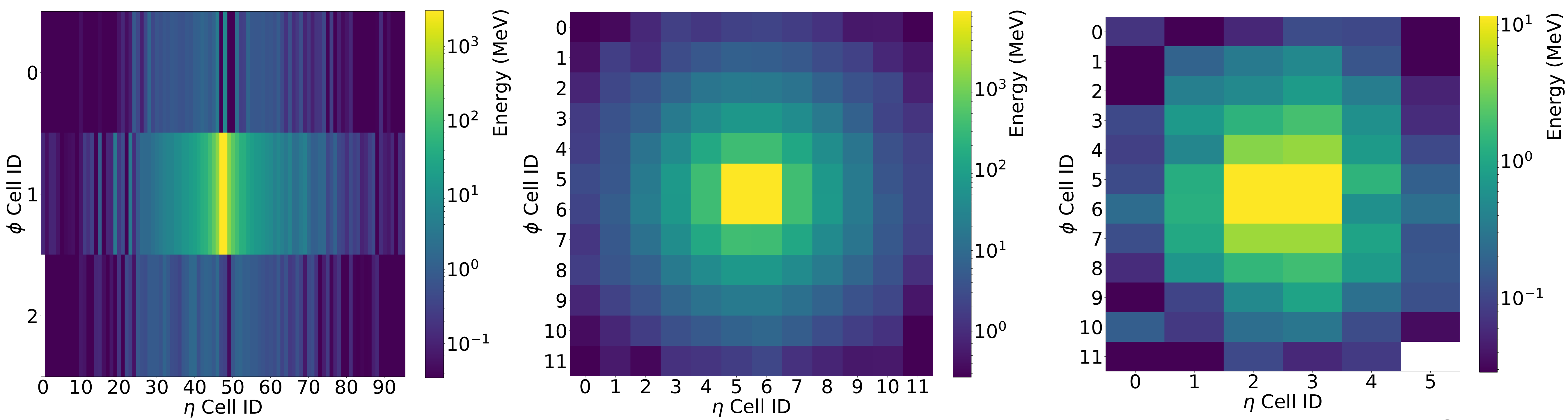

Locally connected layers use filters on small patches

(CNN is then a special case with weight sharing)

# Results: average images

**Geant4**

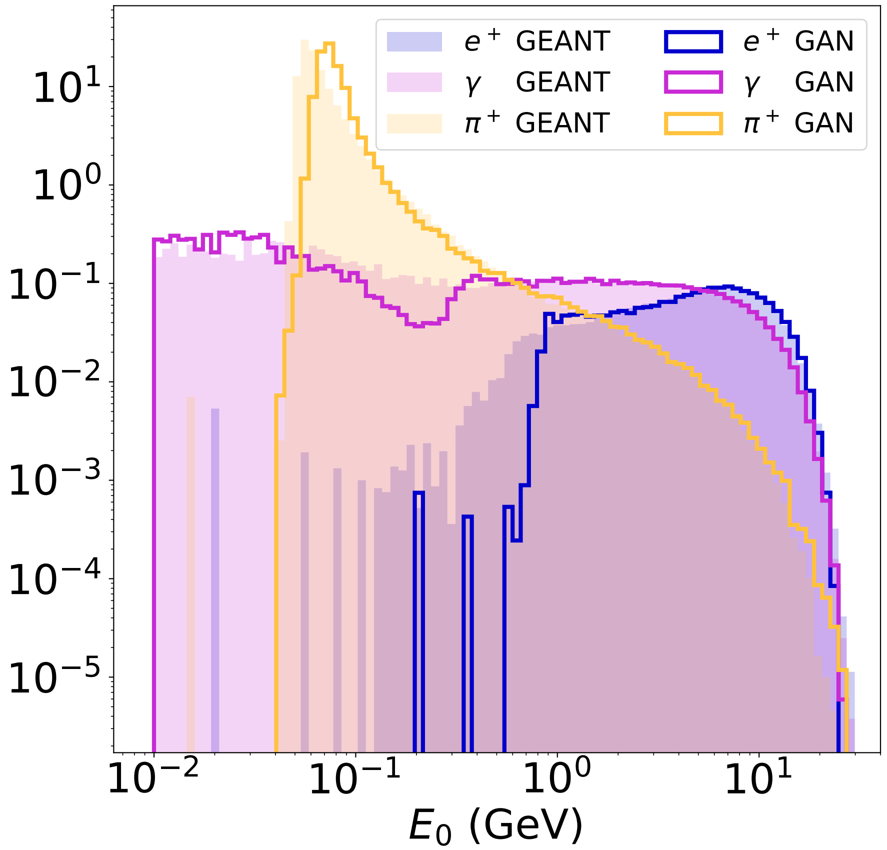


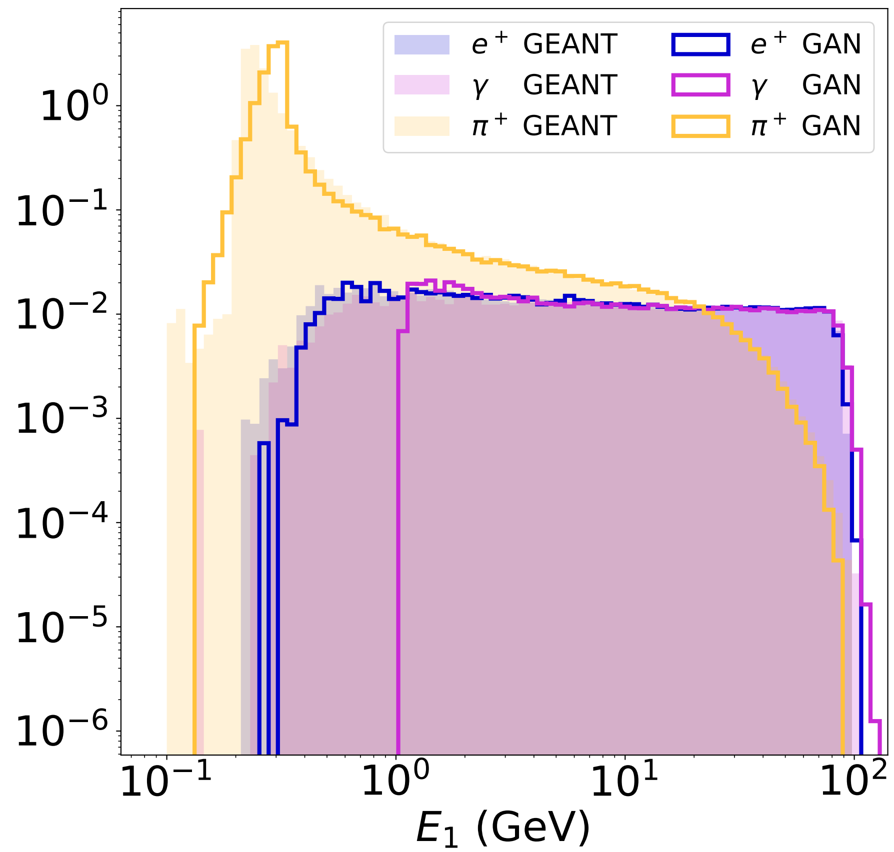

**CaloGAN**

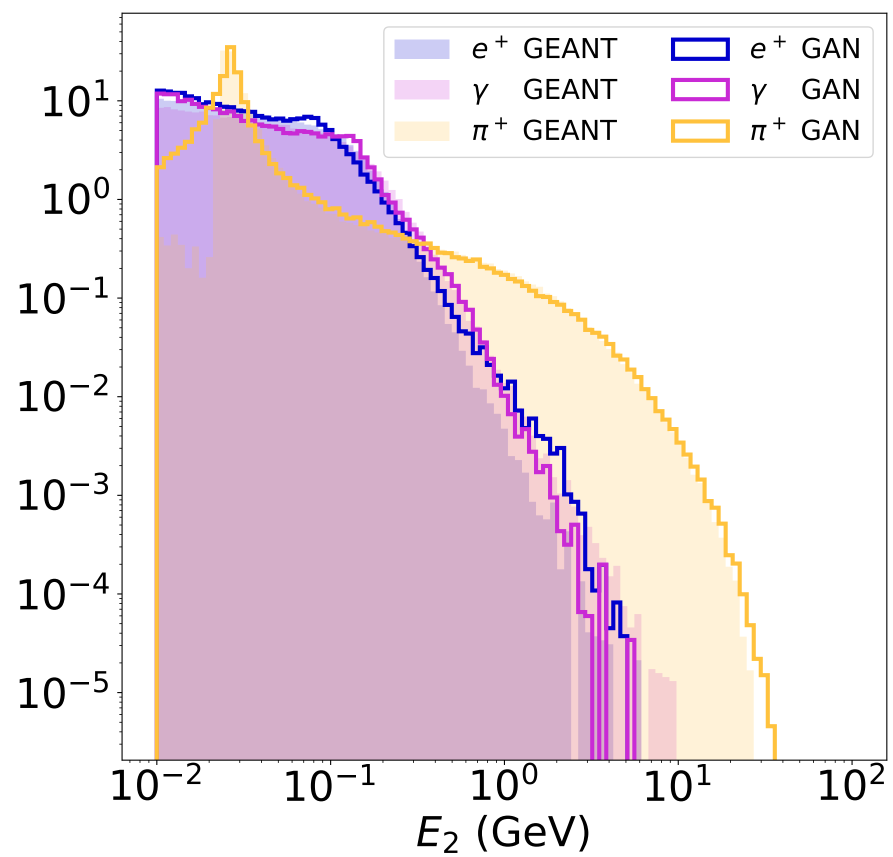
# Energy per layer






Pions deposit much less energy in the first layers; leave the calorimeter with significant energy



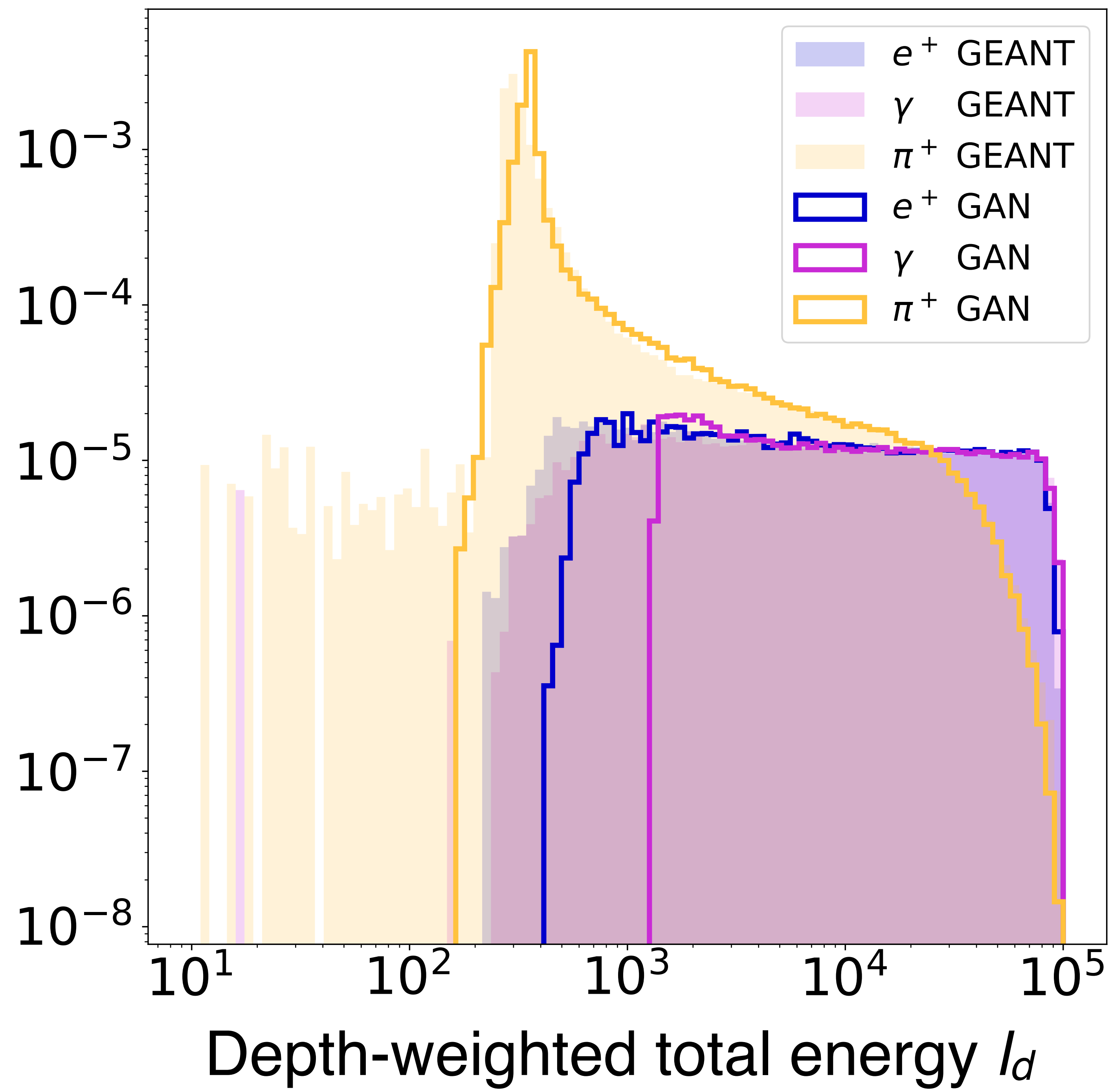

N.B. can always add these (and others) explicitly to the training

# Warning: challenge with GANs

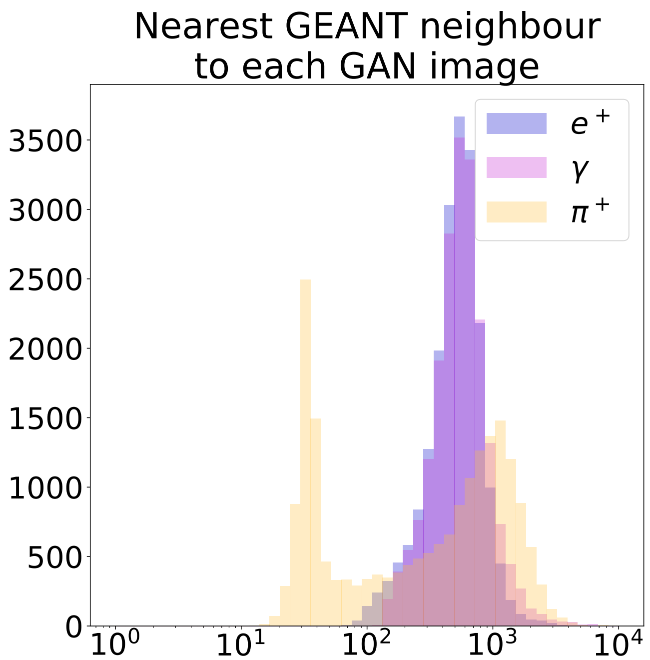


Unlike for classifiers, it is not easy to figure out which GAN is a good GAN - trying to learn a O(1000) generative model and not a single likelihood ratio!

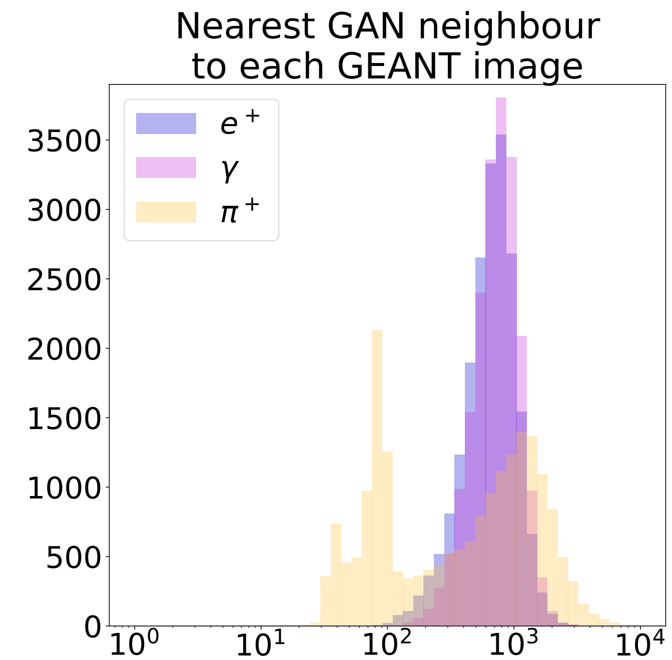
…this is a place where **science applications can make a big impact on ML**.



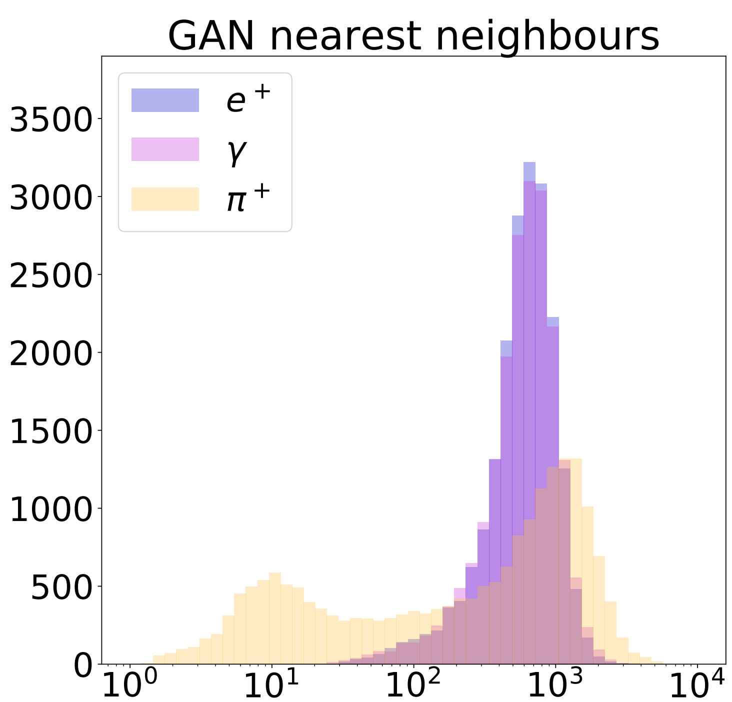
# “Overtraining”



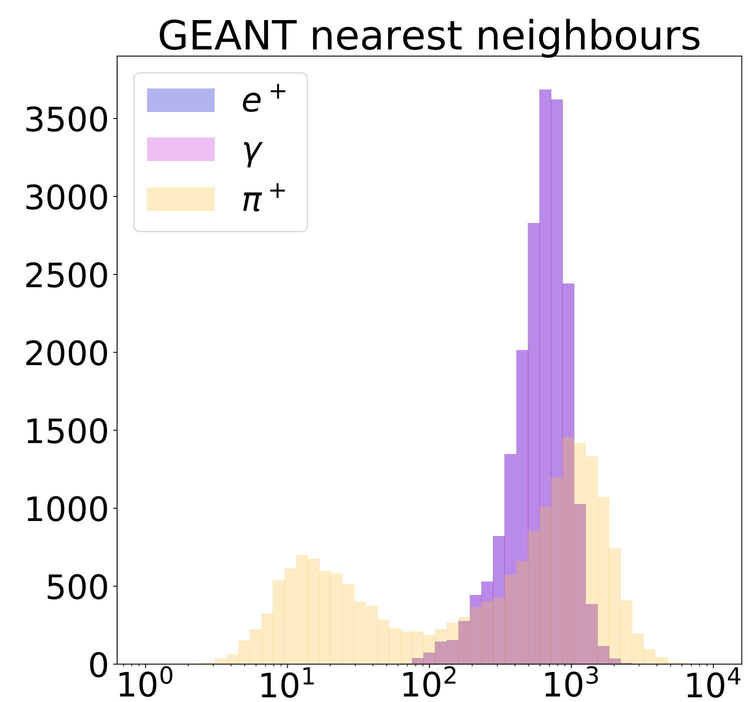

~not memorizing




A key challenge in training GANs is the diversity of generated images. This does not seem to be a (big) problem for CaloGAN.

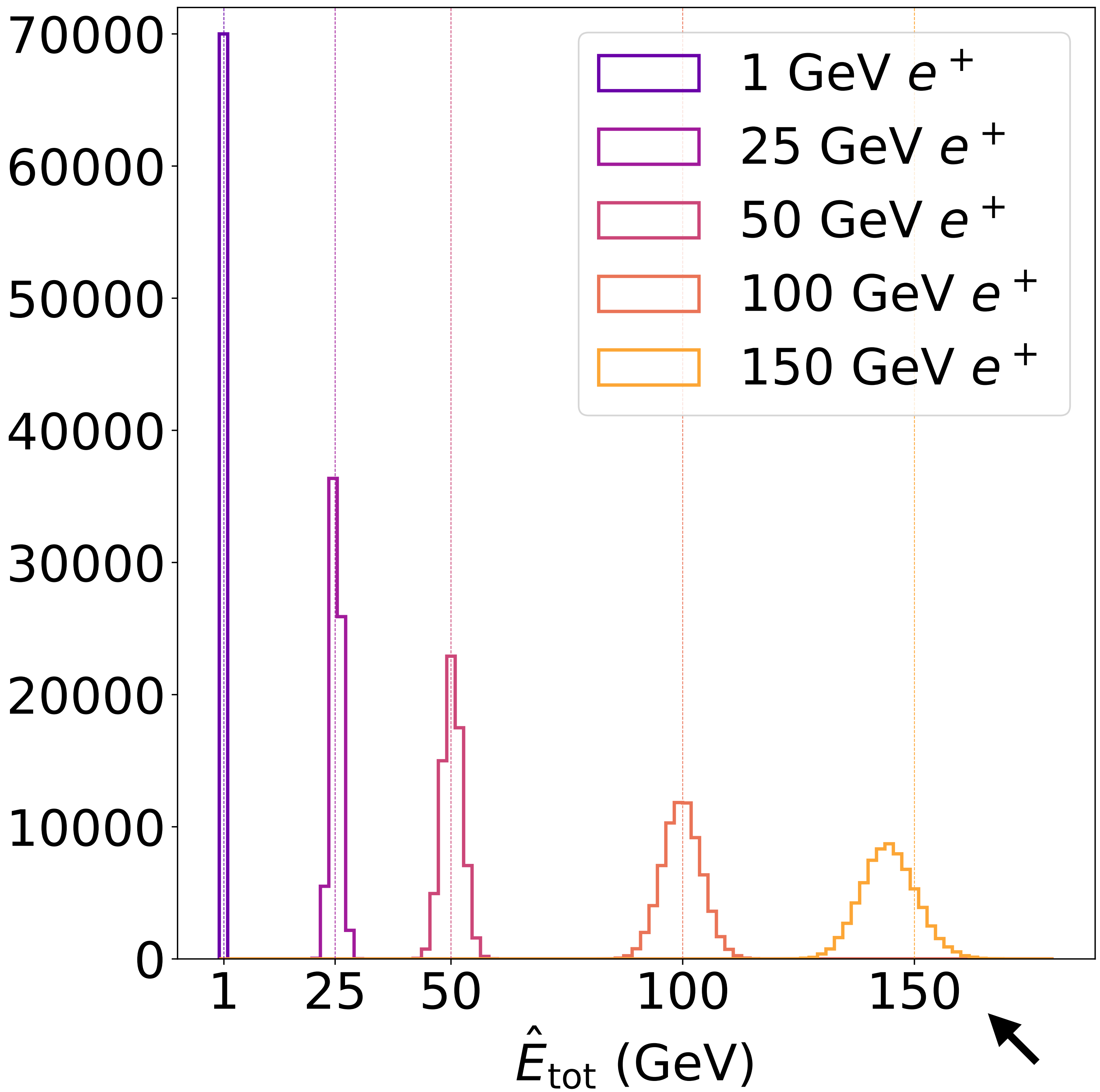



~no mode collapse



# Extrapolating






GANs are not designed to extrapolate, but in some cases, they can smoothly go on!

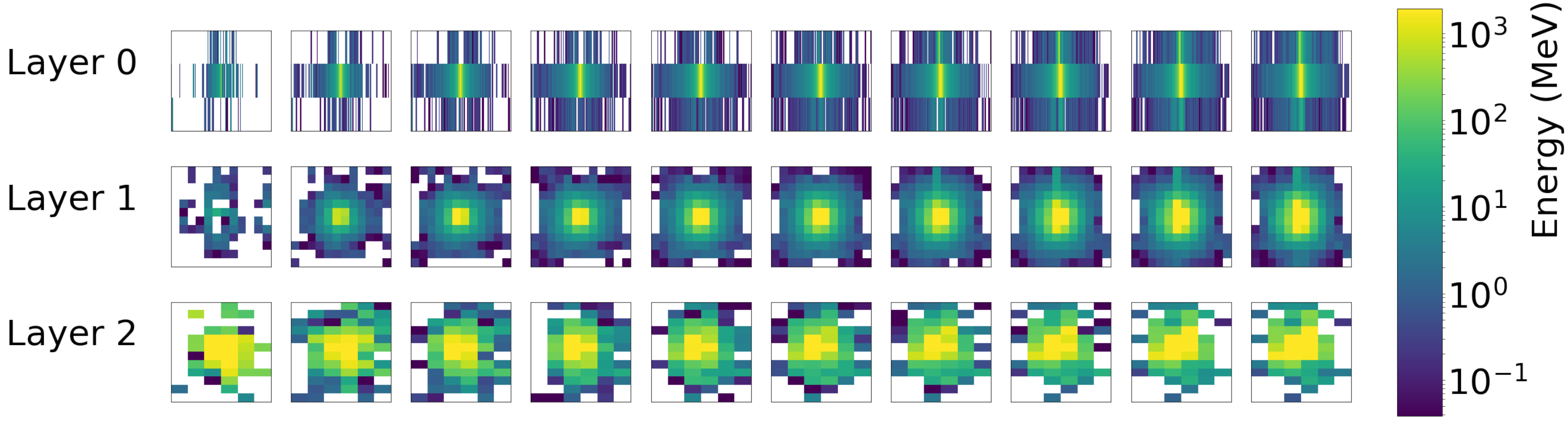
*works here until there is no new physical principles which turn on at some energy*

Beyond our training sample!

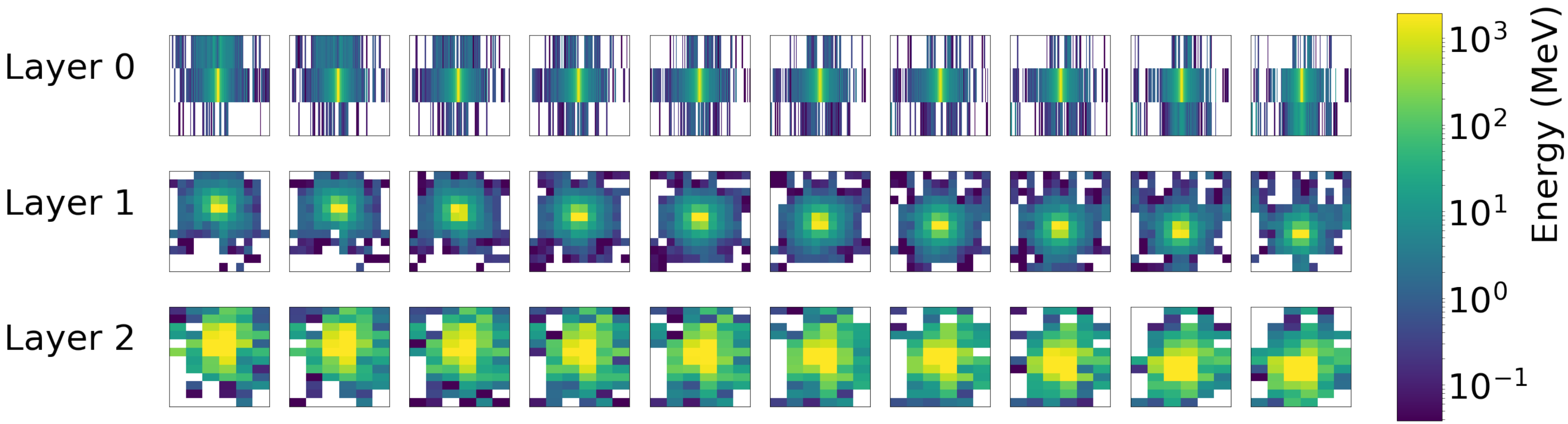
# Conditioning



Fix noise, scan latent variable corresponding to energy




Fix noise, scan latent variable corresponding to x-position



# Timing



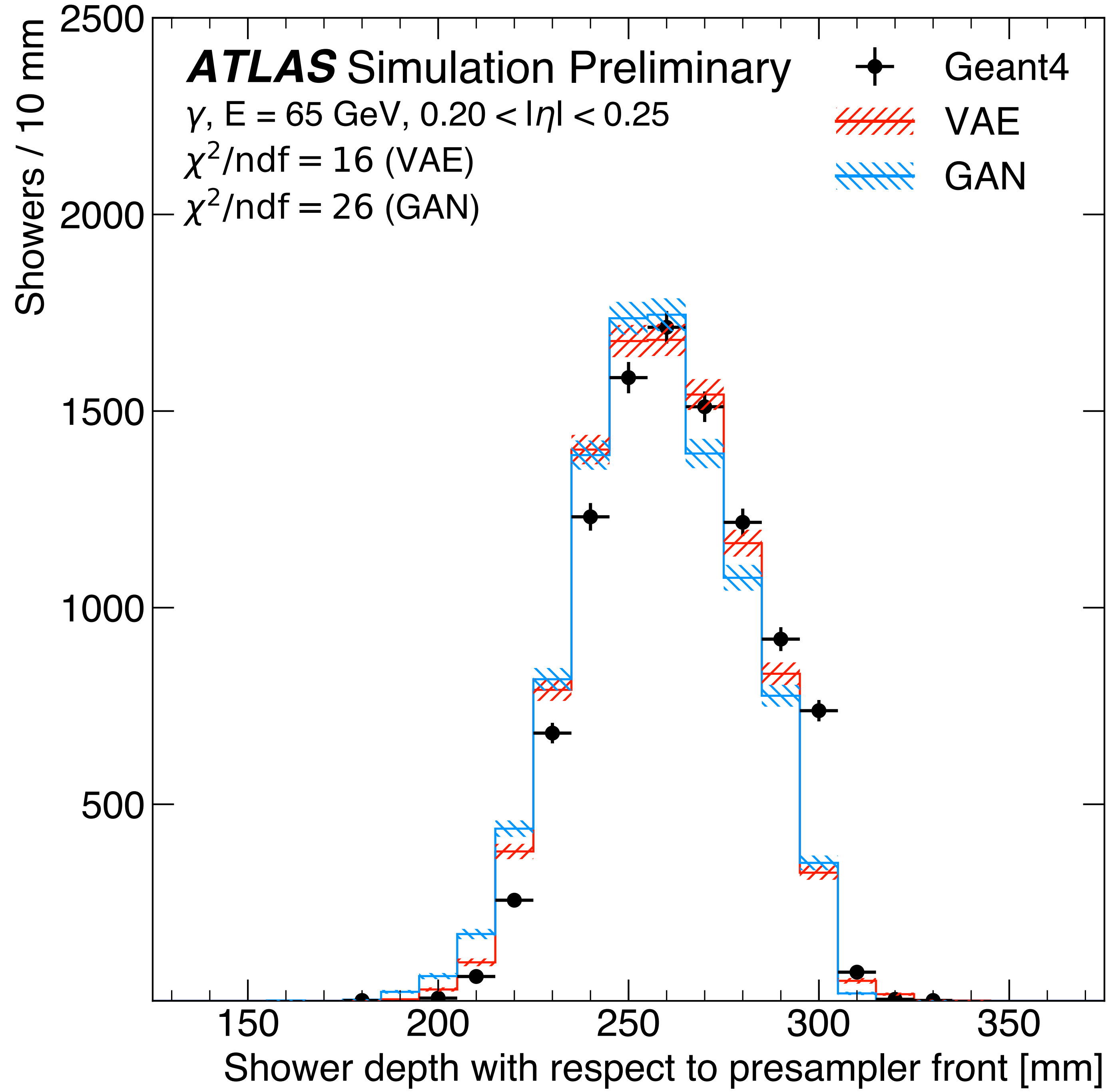
| Generation Method | Hardware | Batch Size | milliseconds/shower |
|---|---|---|---|
| GEANT4 | CPU | N/A | 1772 |
| CALOGAN | CPU *Intel Xeon E5-2670* | 1 | 13.1 |
| | | 10 | 5.11 |
| | | 128 | 2.19 |
| | | 1024 | 2.03 |
| | GPU *NVIDIA K80* | 1 | 14.5 |
| | | 4 | 3.68 |
| | | 128 | 0.021 |
| | | 512 | 0.014 |
| | | 1024 | 0.012 |

(clearly these numbers will change as both technologies improve - this is simply meant to be qualitative and motivating!)

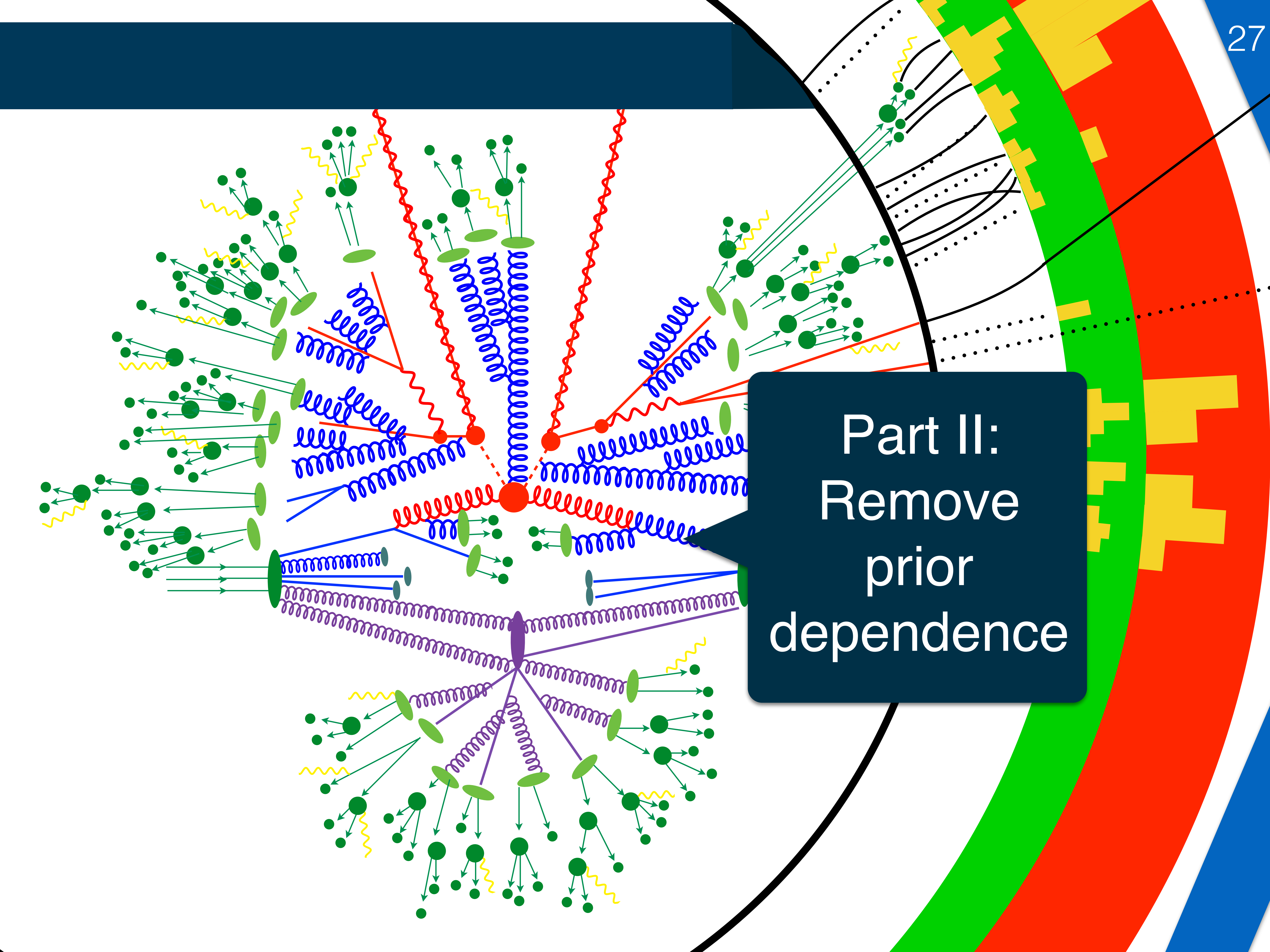
# Collaboration workflow

Integrating these techniques into a full detector simulation is another layer of complication, but is possible and hopefully worth the effort!

ATL-SOFT-PUB-2018-001




GAN studies by Aishik Ghosh and others



Part II:
Remove
prior
dependence

# Simulation dependence + regression

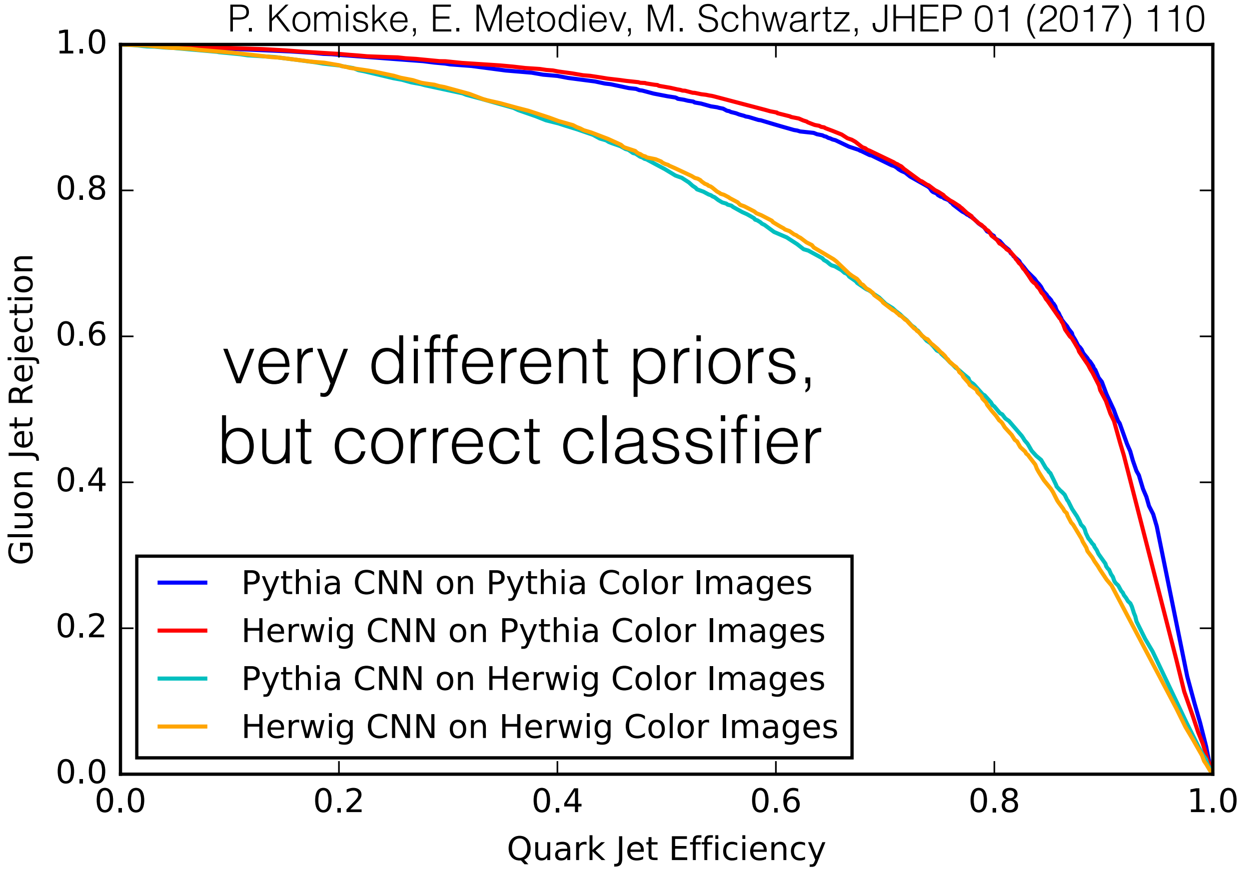


One source of MC dependence is the same as classification:

→ mis-modeling dependencies between features

However, there is a new source:
dependence on the feature priors.

P. Komiske, E. Metodiev, M. Schwartz, JHEP 01 (2017) 110

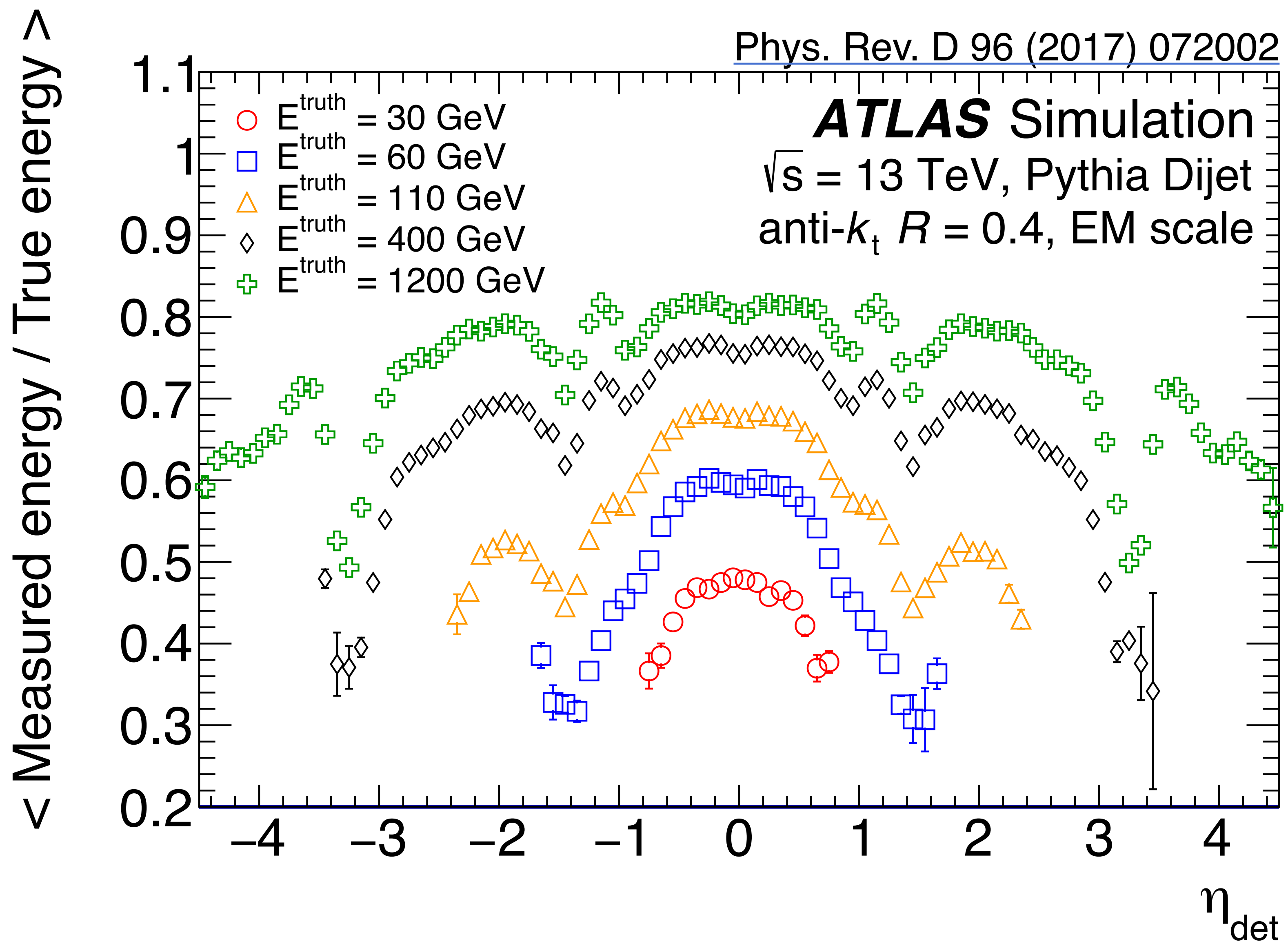



This is really new for regression - for classification, if p(x|signal) and p(x|background) are mis-modeled, you get the optimal answer as long as their ratio is correct.

# Simulation dependence + regression



An example that you can have in mind is energy calibration.




We want to predict the true energy given the measured energy

(and possibly other features - more on that soon)

…however what I'm about to say applied more generally (though the impact is biggest when the resolution is poorest)

# What can go wrong?



Suppose you have some features x and you want to predict y.

*detector energy* *true energy*

One way to do this is to find an f that minimizes the mean squared error:

$$f = \text{argmin}_g \sum_i (g(x_i) - y_i)^2$$

Then, f(x) = E[y|x].

Proof in the lectures from yesterday!

Why is this a problem?

# What can go wrong?



$$f(x) = E[y|x] = \int \mathrm{d}y \, y \, p(y|x)$$

$$E[f(x)|y] = \int \mathrm{d}x \, \mathrm{d}y' \, y' \, p_{\mathrm{train}}(y'|x) \, p_{\mathrm{test}}(x|y)$$

this need not be $y$ even if $p_{train} = p_{test}$ (!)

# One solution: Numerical inversion



ATLAS and CMS use a trick to be prior-independent:

**Numerical inversion** *instead of predicting y from x, predict x from y and then invert the function*

… put another way:

learn $f: y \rightarrow x$ and then for a given x, predict $f^{-1}(x)$

by construction, f is independent of p(y) and thus $f^{-1}$ also does not depend on p(y), as desired.

For math details, see A. Cukierman and B. Nachman, NIMA 858 (2017) 1

# Caveats about numerical inversion



This procedure is independent of the prior p(y) but may not close exactly, i.e. $E[f^{-1}(x)|y]$ may not be y.

…under mild assumptions, it does close for the mean absolute error, but usually has some non-closure for the squared error.

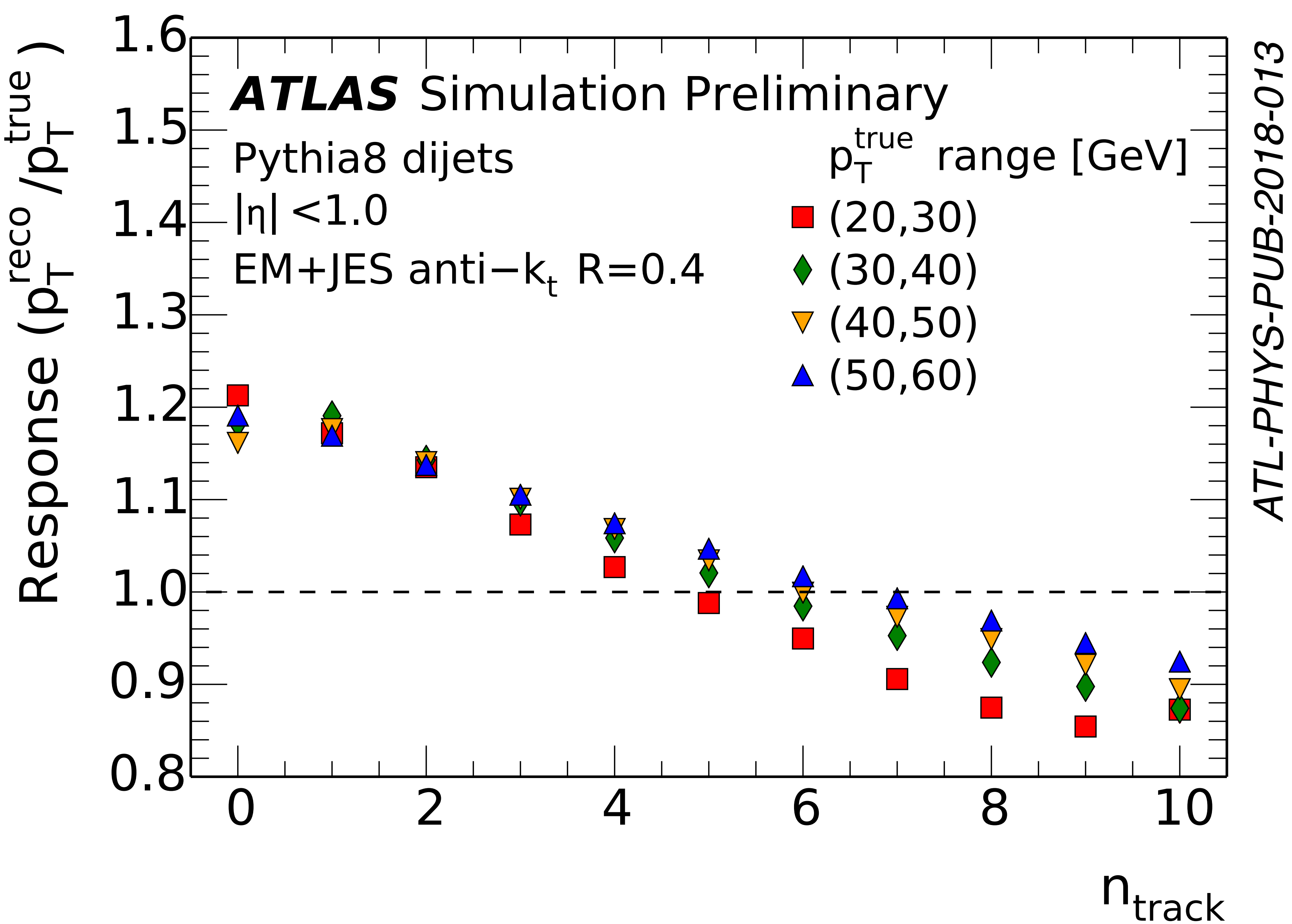
Also, the calibration procedure can distort the underlying distribution, i.e. if you start with a Gaussian, you almost never end up with exactly a Gaussian.

For math details, see A. Cukierman and B. Nachman, NIMA 858 (2017) 1

# + more features

The detector response of jets depends on many properties of the jet. Ideally, the calibration can include this!

E[x|y] / y

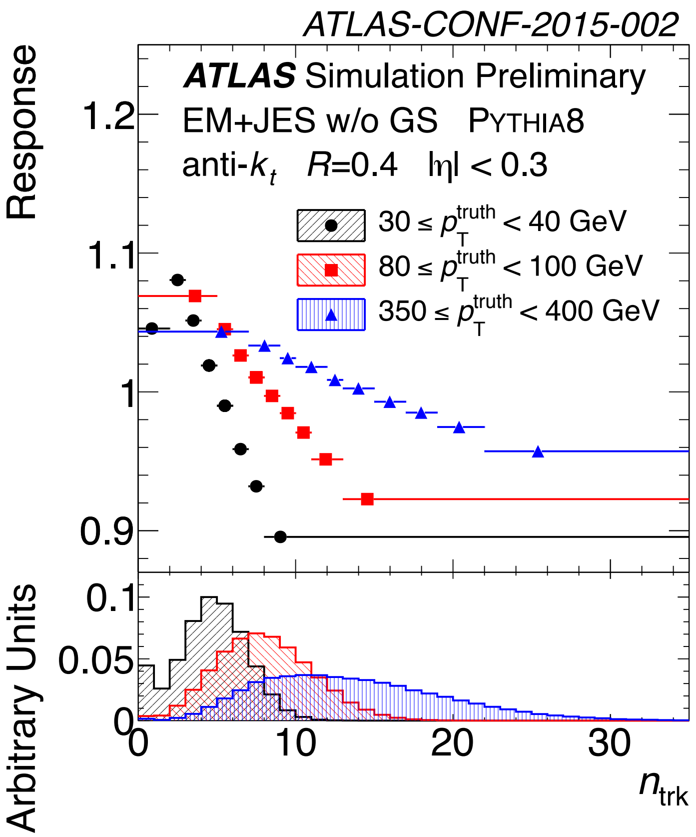


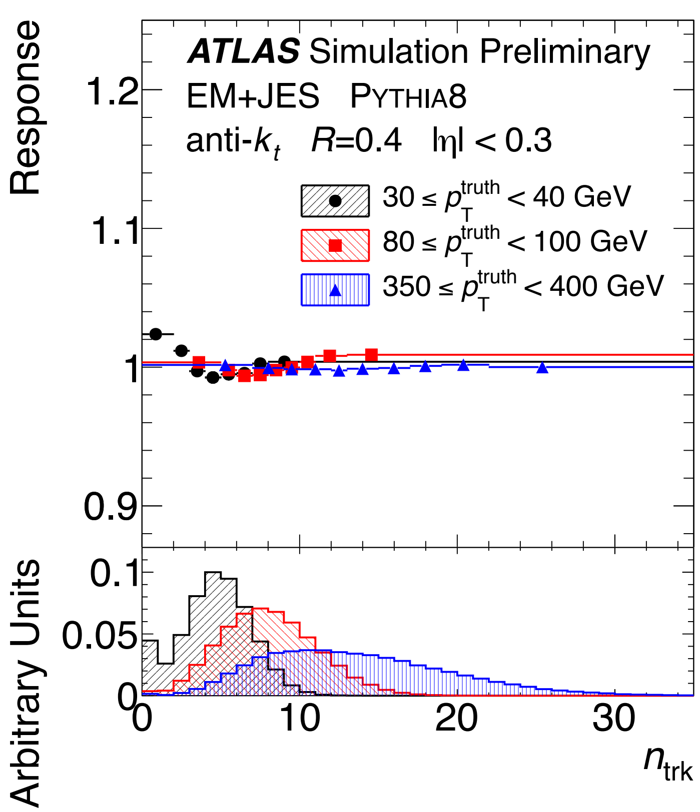
# Global sequential calibration



The current ATLAS approach to including more features is to repeat NI sequentially:

$$p_{\mathrm{T}}^{\mathrm{reco}} \mapsto \hat{p}_{\mathrm{T}}^{\mathrm{reco}} = f_{\theta_n}^{-1}\left(\cdots f_{\theta_2}^{-1}\left(f_{\theta_1}^{-1}\left(p_{\mathrm{T}}^{\mathrm{reco}}\right)\right)\cdots\right)$$




This works well when the jet response is independent of $\theta_i$ given $\theta_j$.

# Machine learning calibration



For reasons discussed earlier, we can't include correlations by learning y given x and all the $\theta$'s.

However, it would still be great to use machine learning to automatically and efficiently make use of correlated information.

We cannot use numerical inversion out-of-the-box because we now have a many-to-one function.

# Generalized numerical inversion



Since we are not (necessarily) interested in calibrating the $\theta$'s, we can generalize NI as follows:

(1) Learn a function f to predict x given y and all the $\theta$'s.

(2) For every combination of $\theta$, invert f.

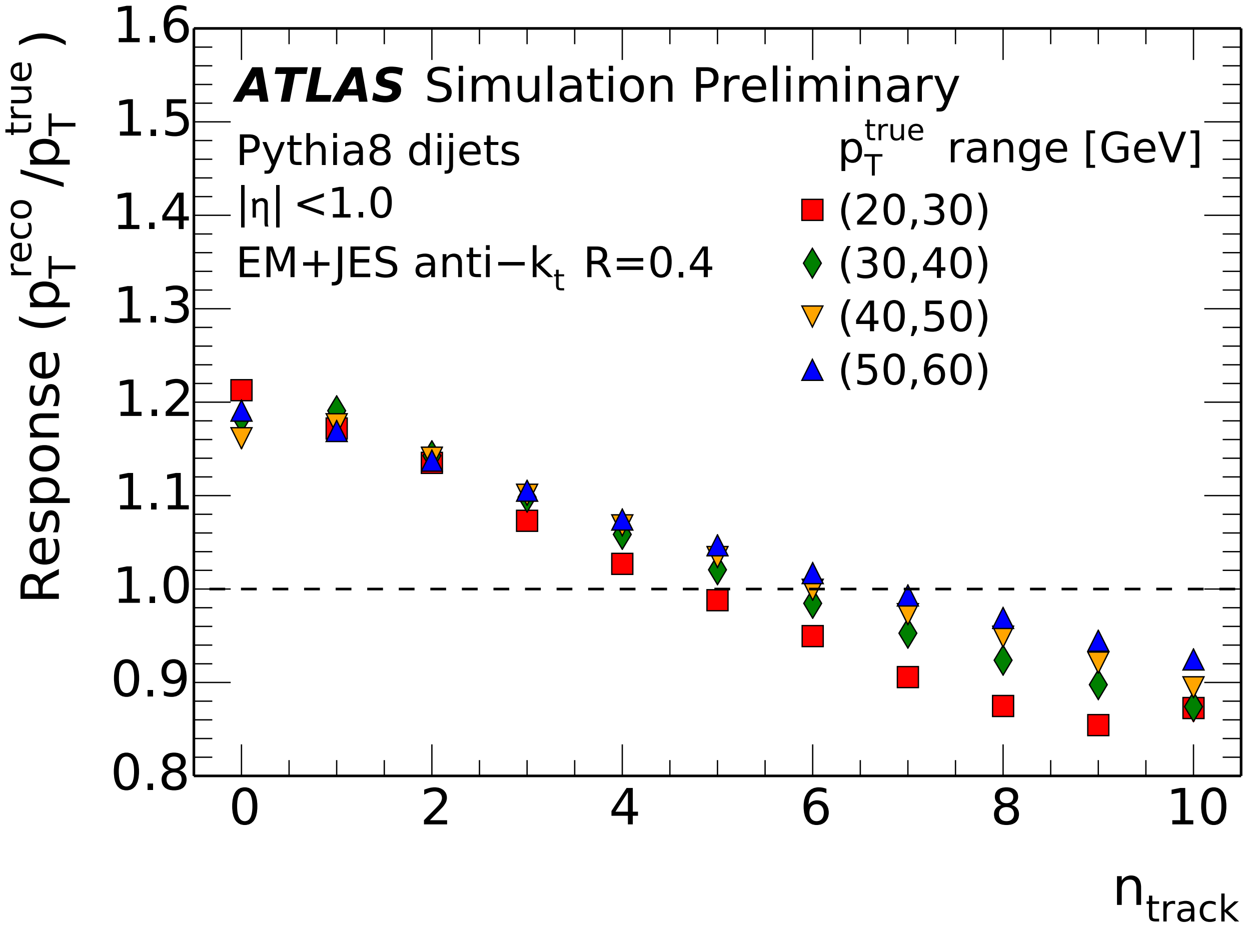
(3) Calibrate via $x \to f_\theta^{-1}(x)$

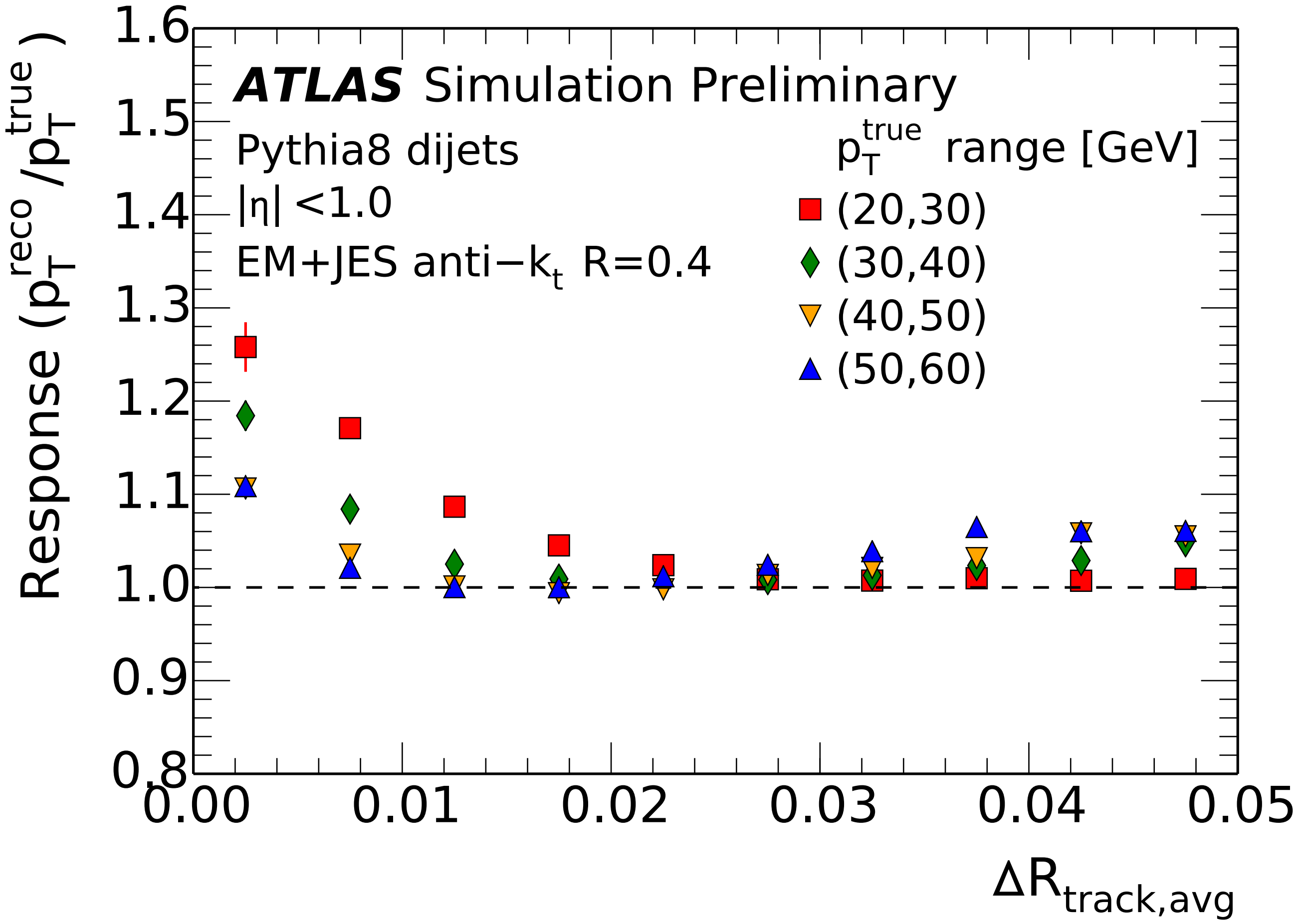
Step (2) is intractable, so replace it with another learning step: predict y given $f(y,\theta)$ and $\theta$.

# GNI in action



Consider two features:



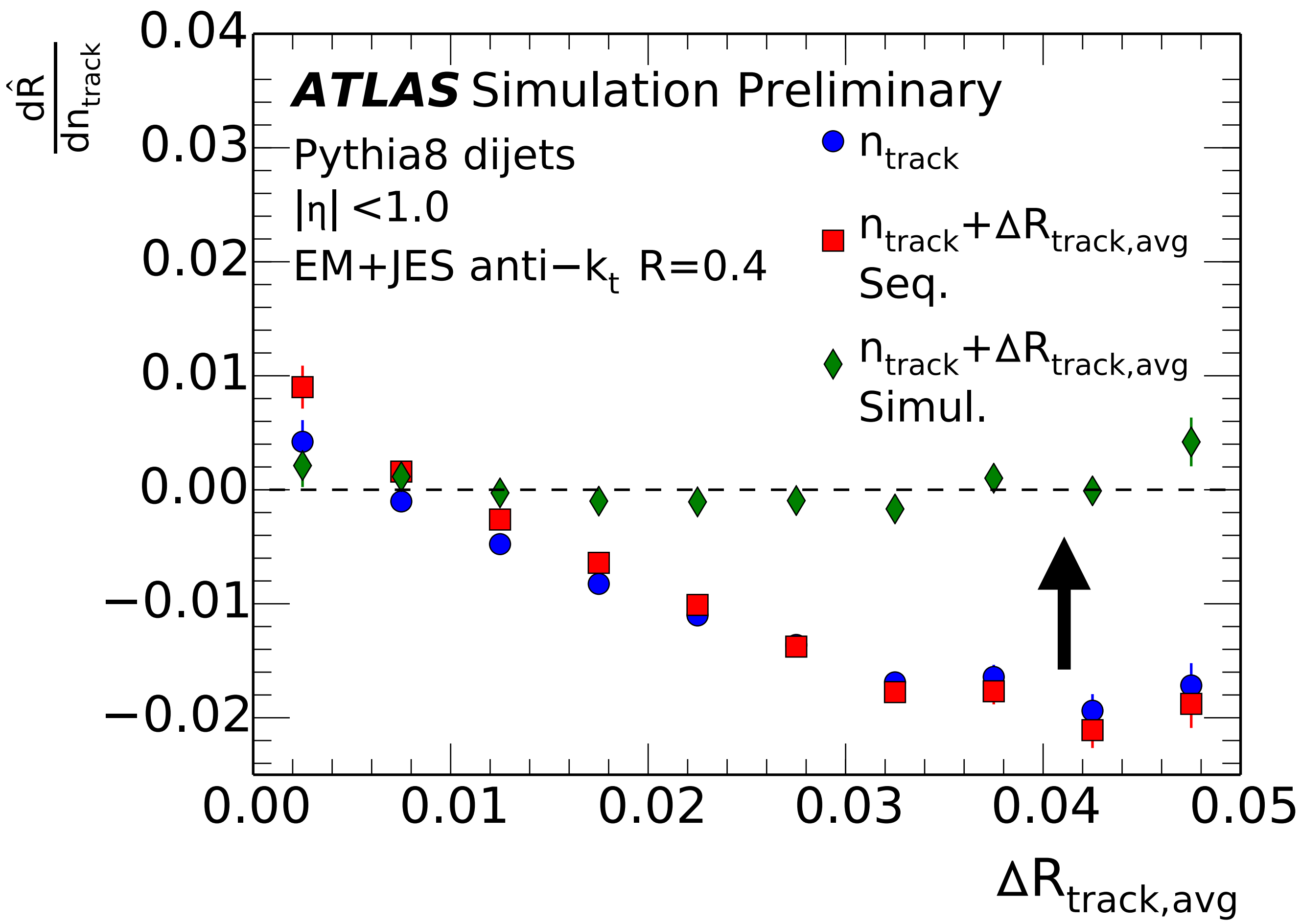



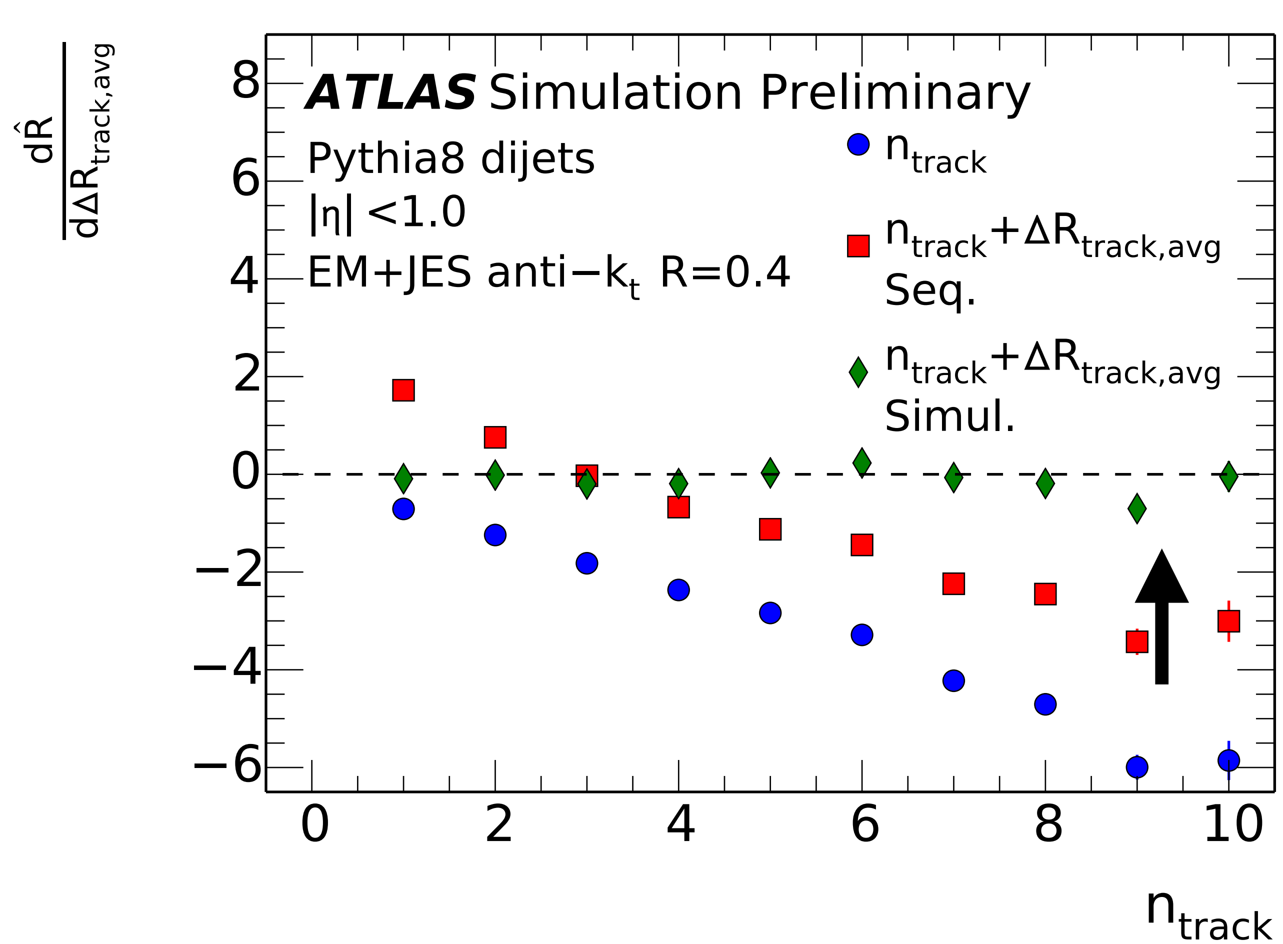

average track $p_T$-weighted distance from jet center (doesn't really matter what it is)

# GNI in action



$\hat{R}$ is the calibrated E[x|y] / y



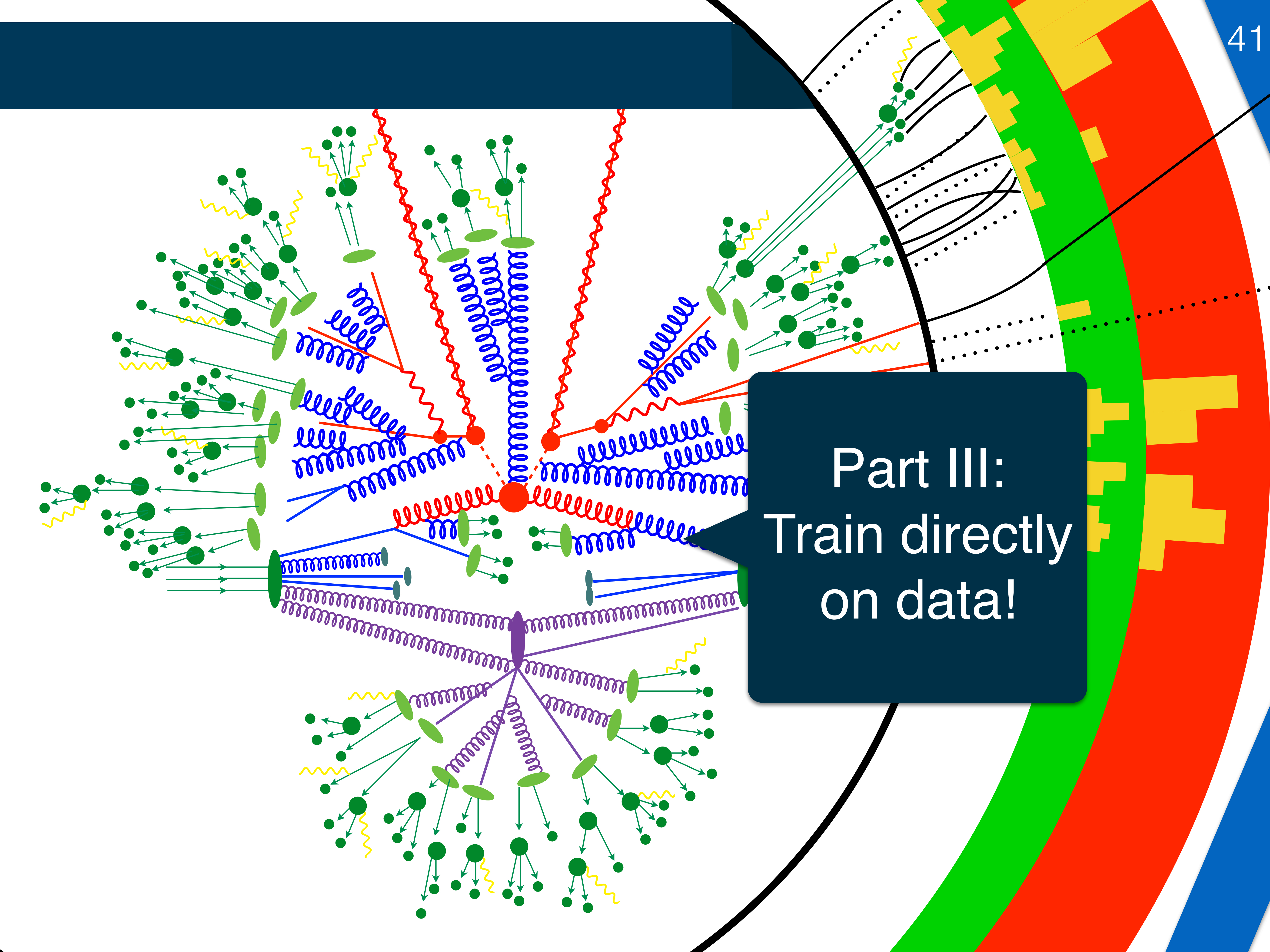

Only the simultaneous approach removes the full residual dependence!

# The future



Adding more features (with more interdependencies) will lead to more dramatic improvements.

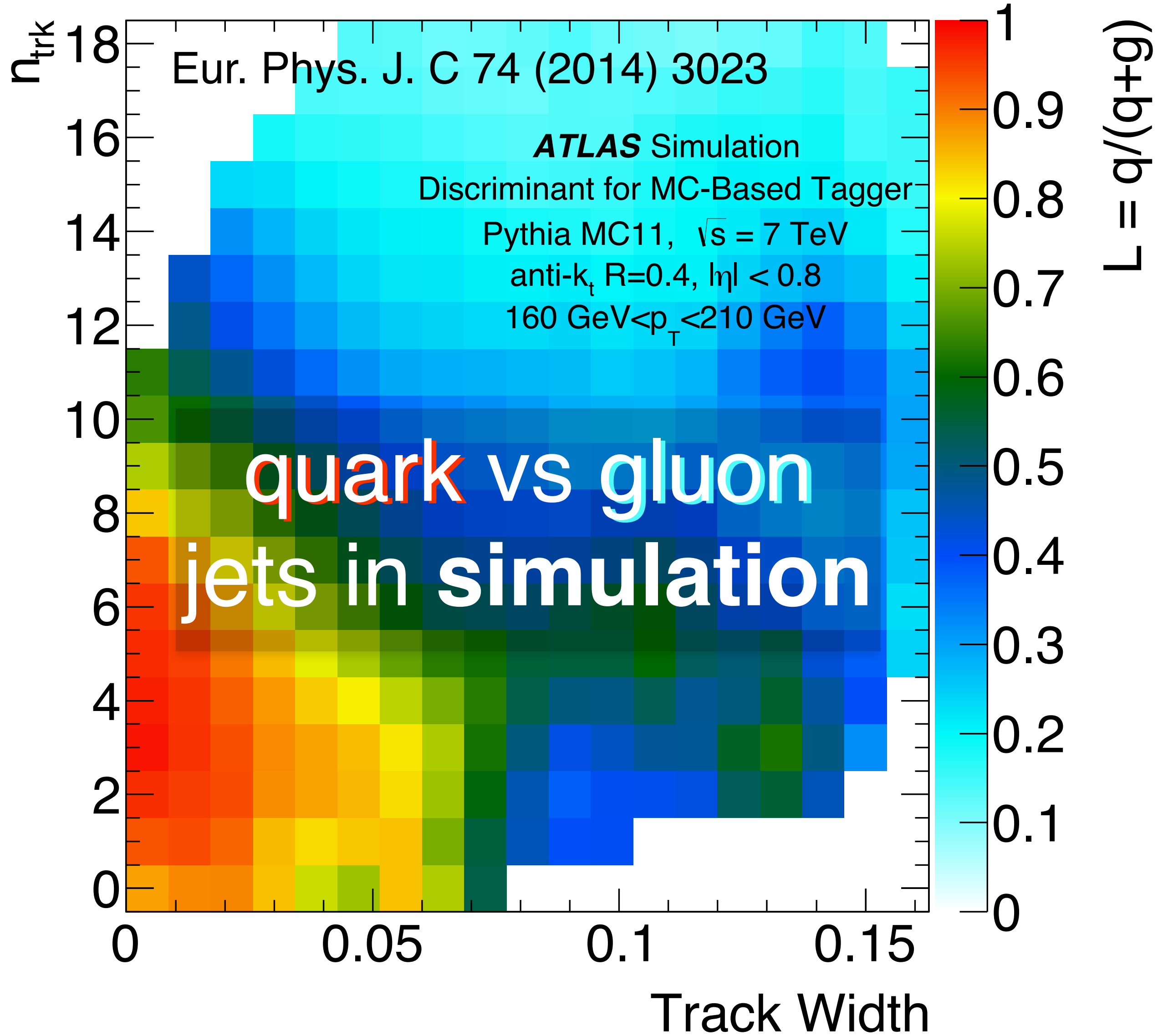
We can also extend this approach to calibrate other observables and even simultaneously calibrate some of the $\theta$'s (even more generalized NI!)

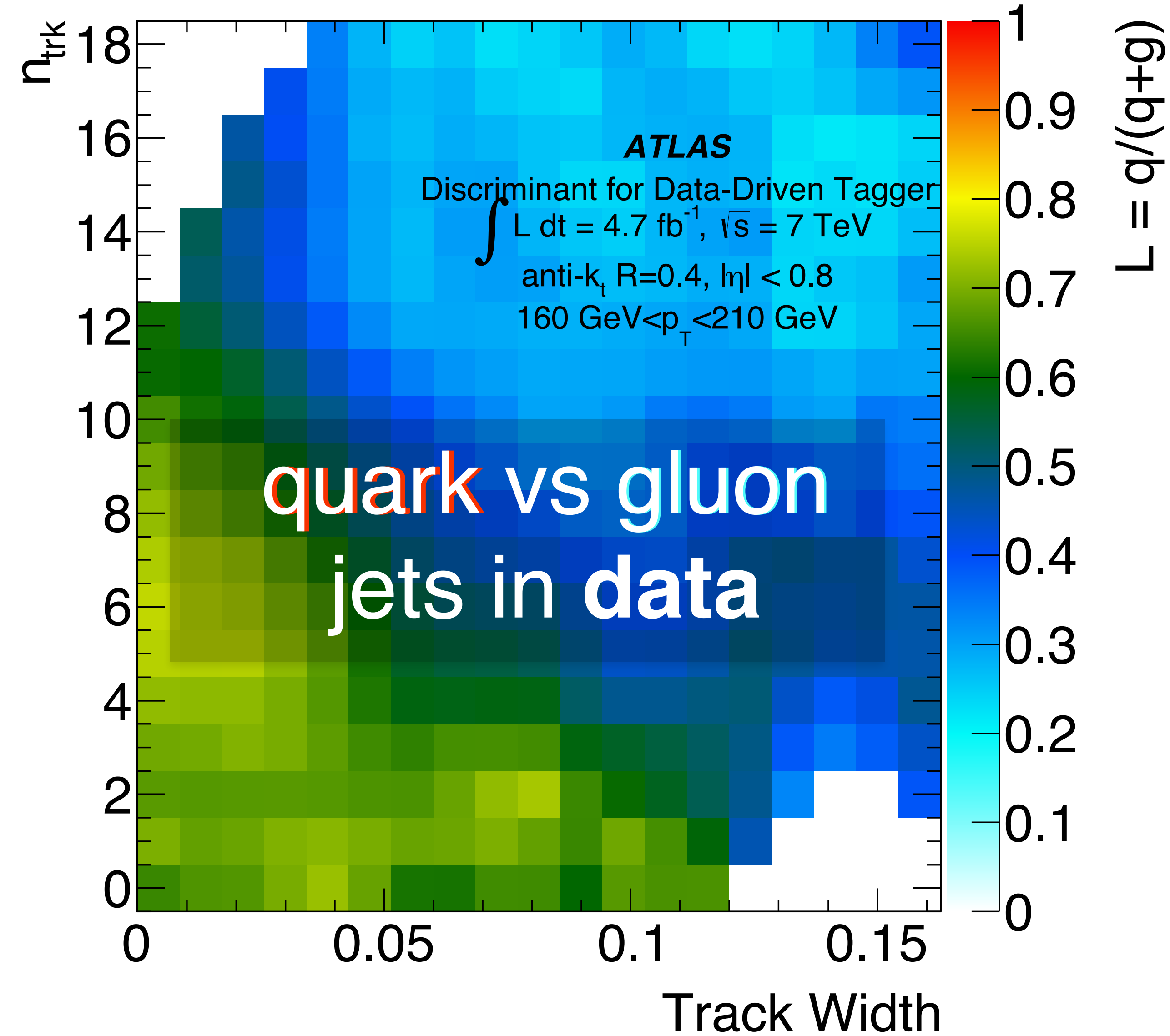

Part III:
Train directly
on data!

# Background and Motivation



Usual paradigm: train in simulation, validate on data, test on data.



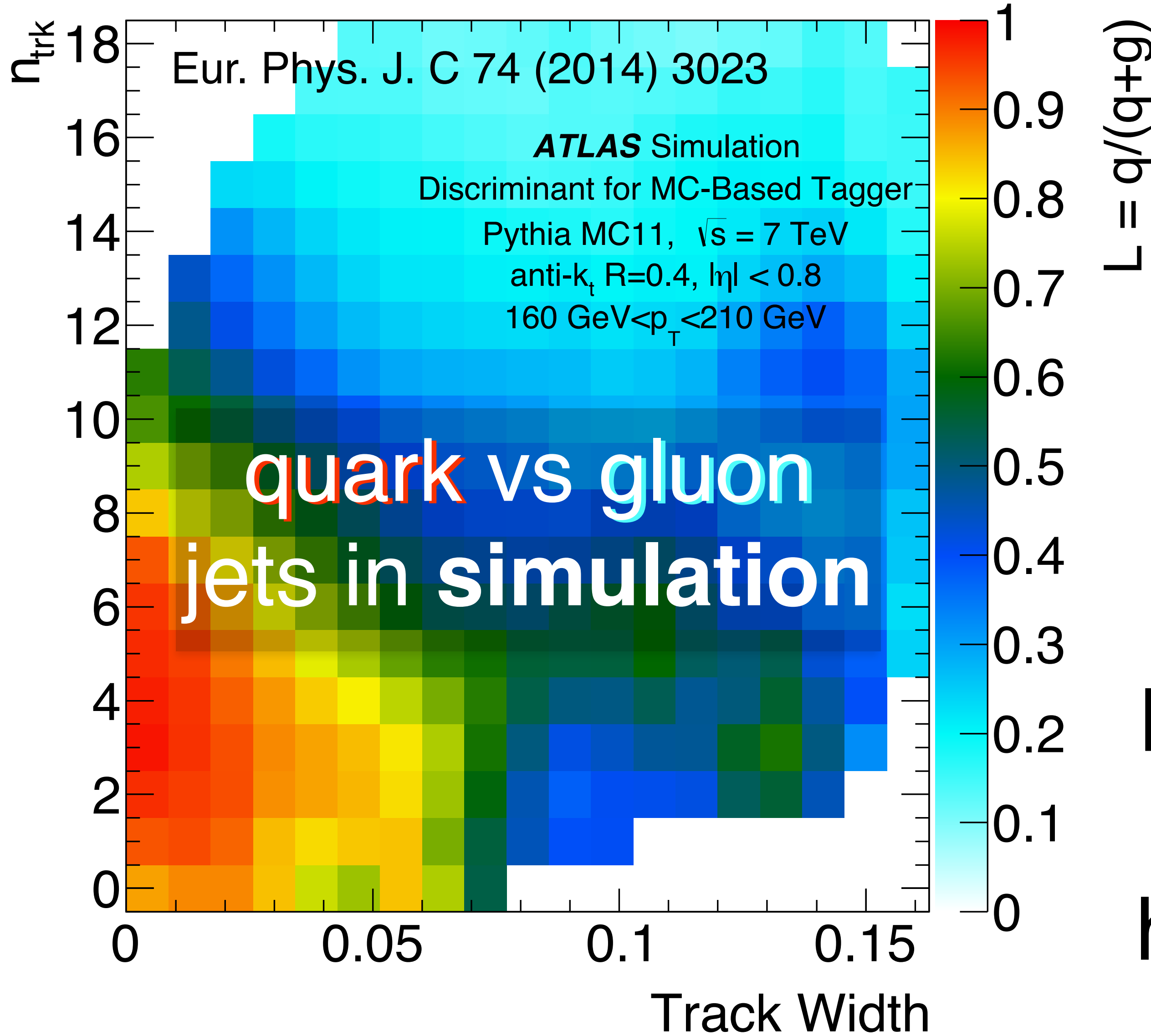




If data and simulation differ, this is sub-optimal!

# Background and Motivation



Usual paradigm: **train in simulation**, validate on data, test on data.




Recall: optimal classifier is a threshold cut on the likelihood ratio.

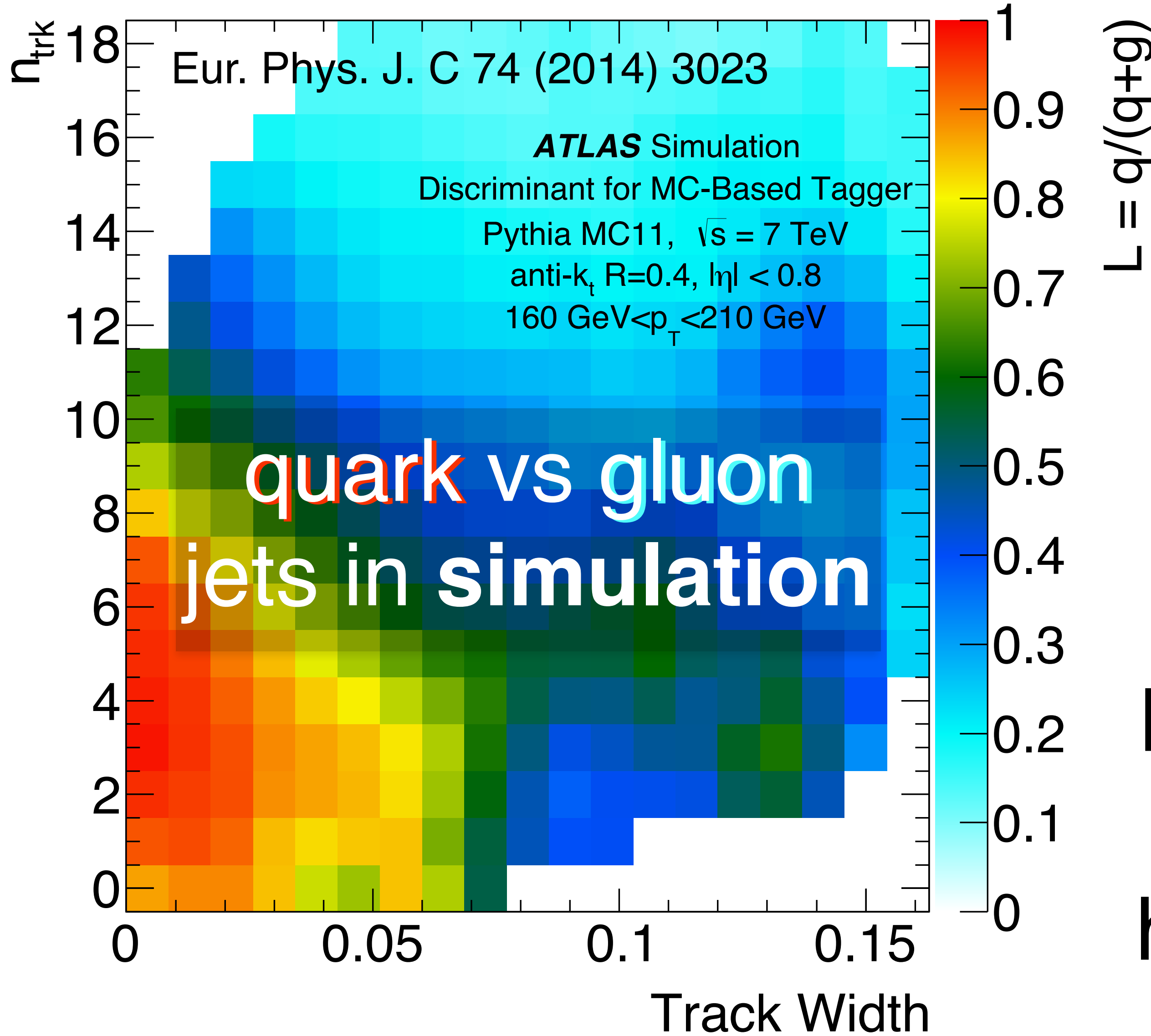
For a 2D feature space, no need for a NN or BDT - can use a histogram to "train" the classifier.

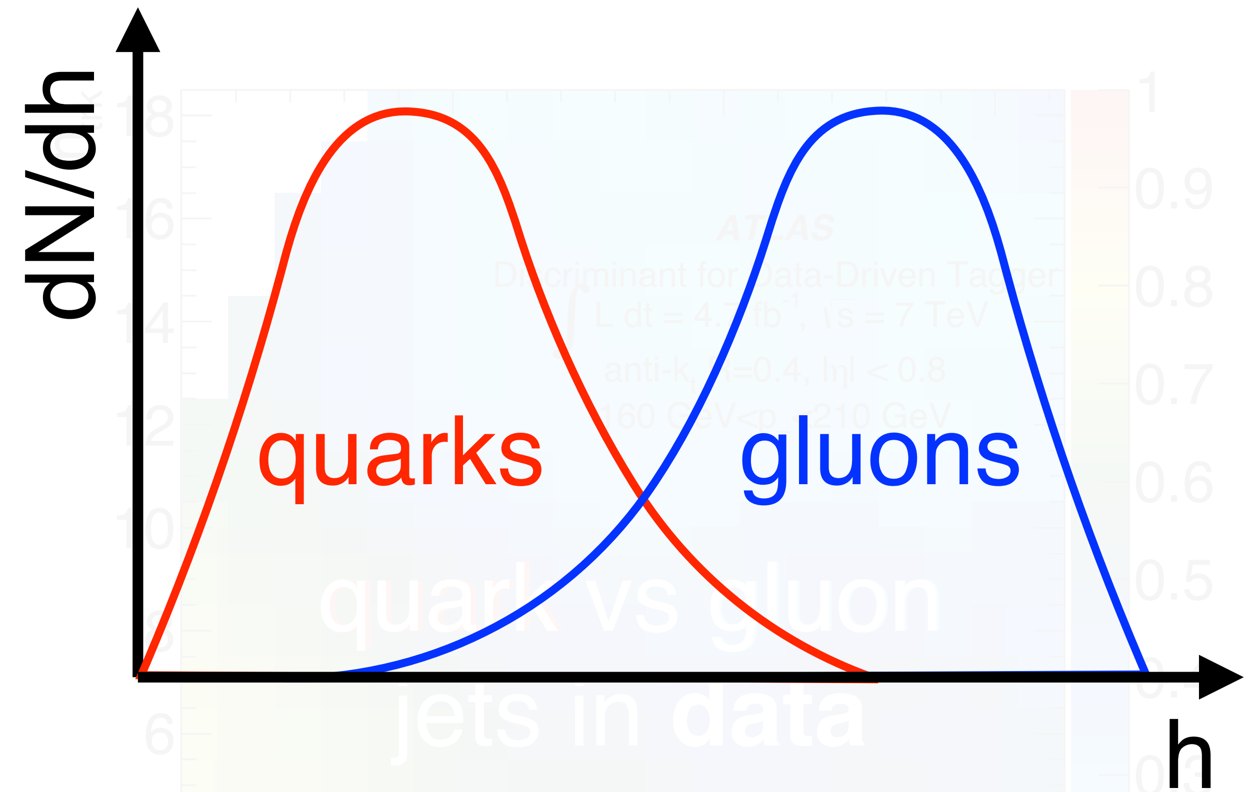
$h(n_{trk}, \text{Track Width}) \rightarrow [0,1]$

# Background and Motivation



Usual paradigm: **train in simulation**, validate on data, test on data.



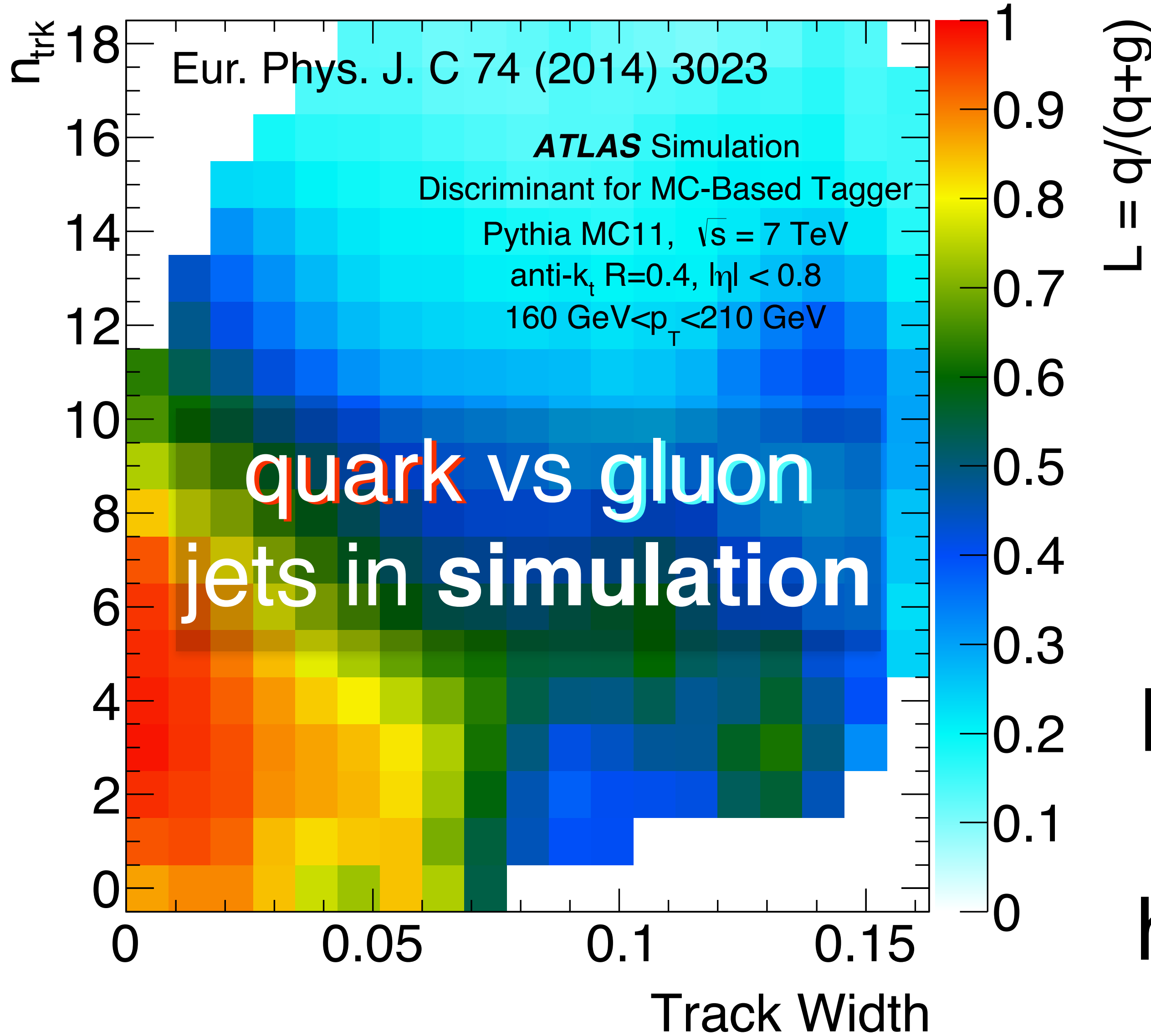

For a 2D feature space, no need for a NN or BDT - can use a histogram to “train” the classifier.

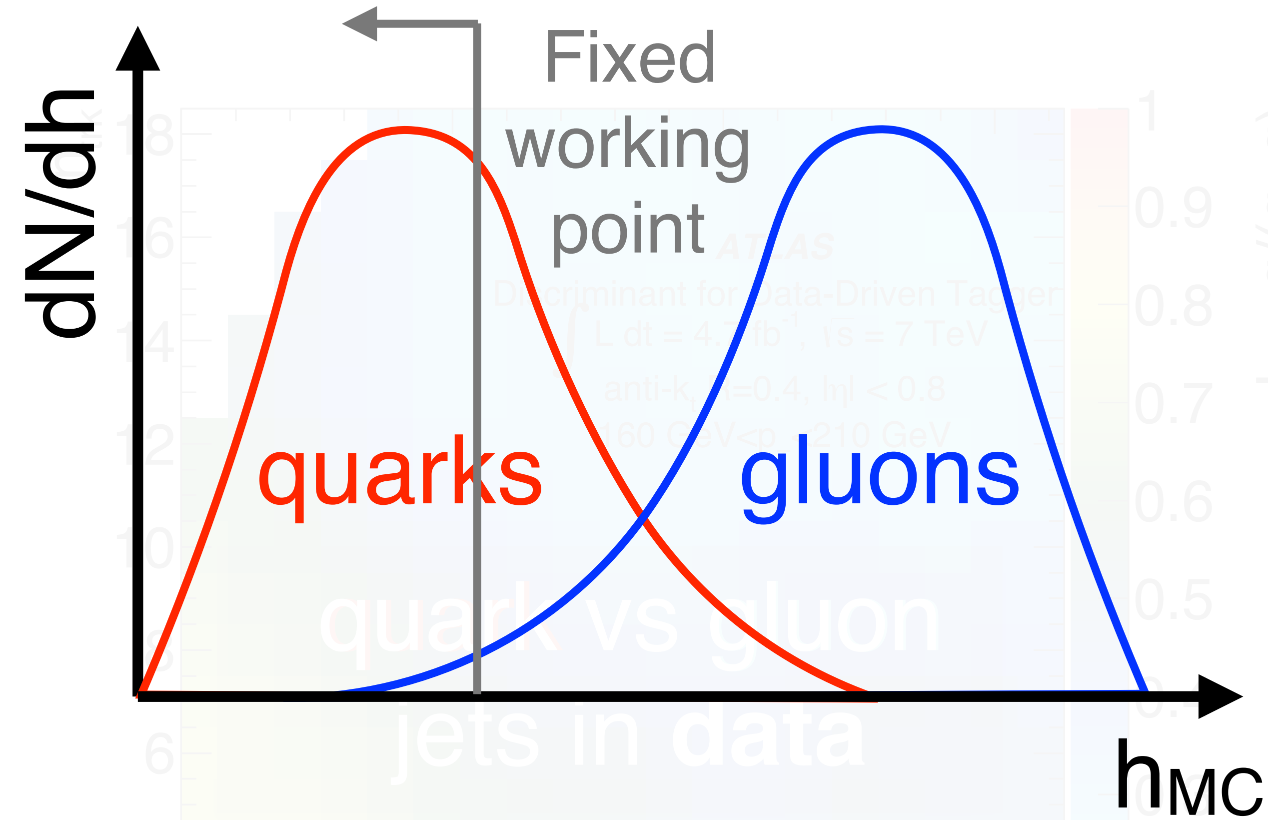
h($n_{trk}$, Track Width) → [0,1]

# Background and Motivation



Usual paradigm: **train in simulation**, validate on data, test on data.



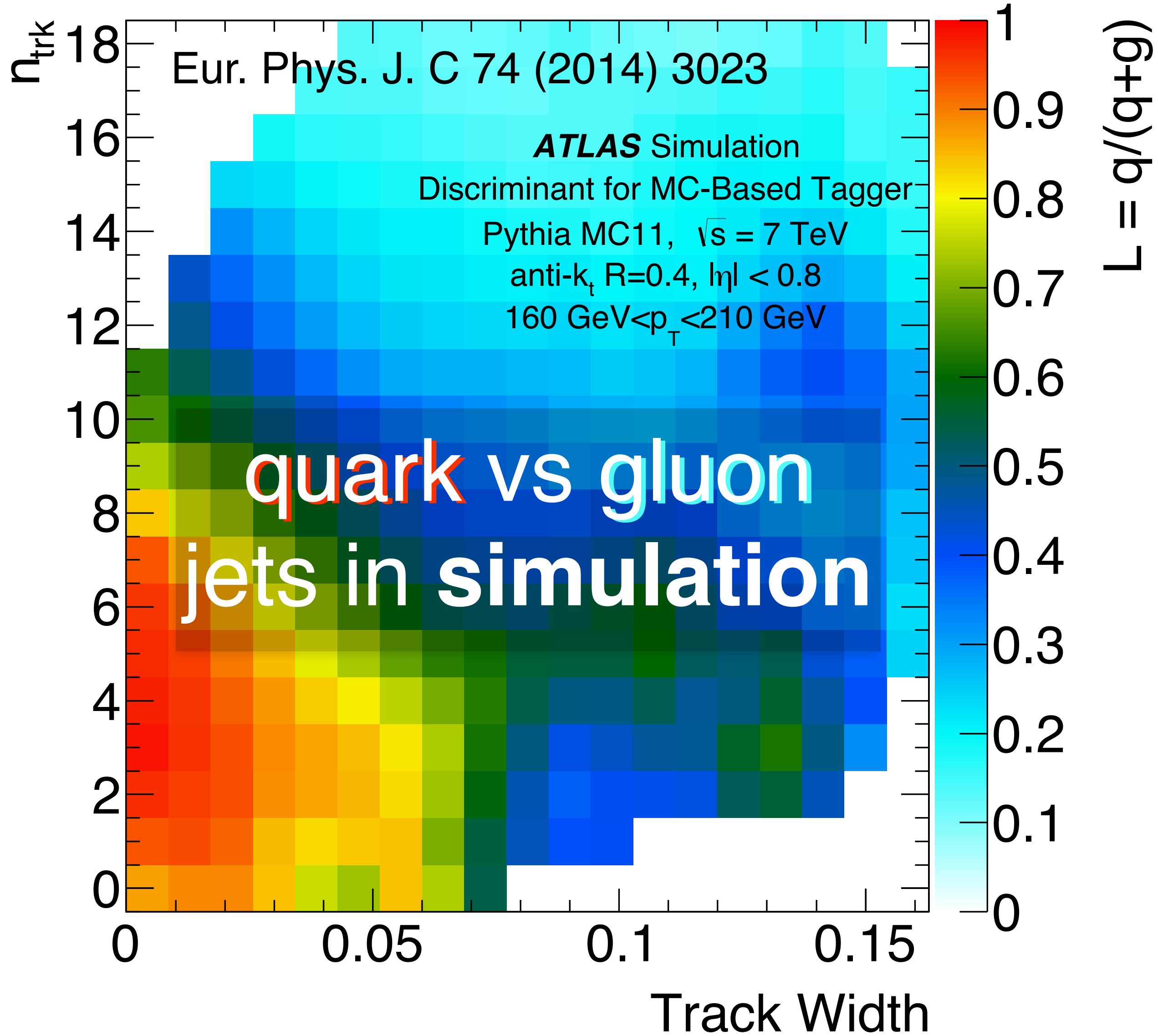

For a 2D feature space, no need for a NN or BDT - can use a histogram to “train” the classifier.

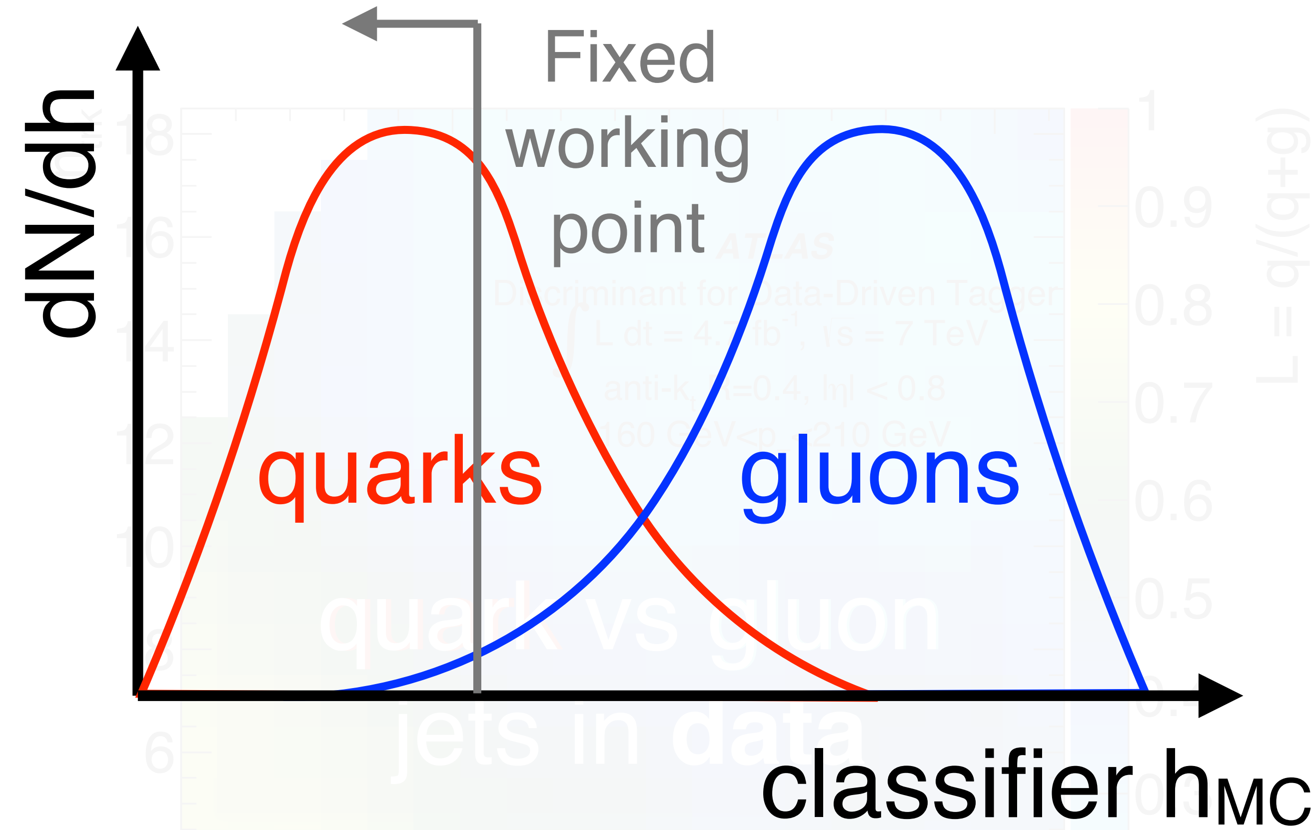
$h_{MC}(n_{trk}, \text{Track Width}) \rightarrow [0,1]$

# Background and Motivation



Usual paradigm: **train in simulation**, validate on data, test on data.




WP in simulation:

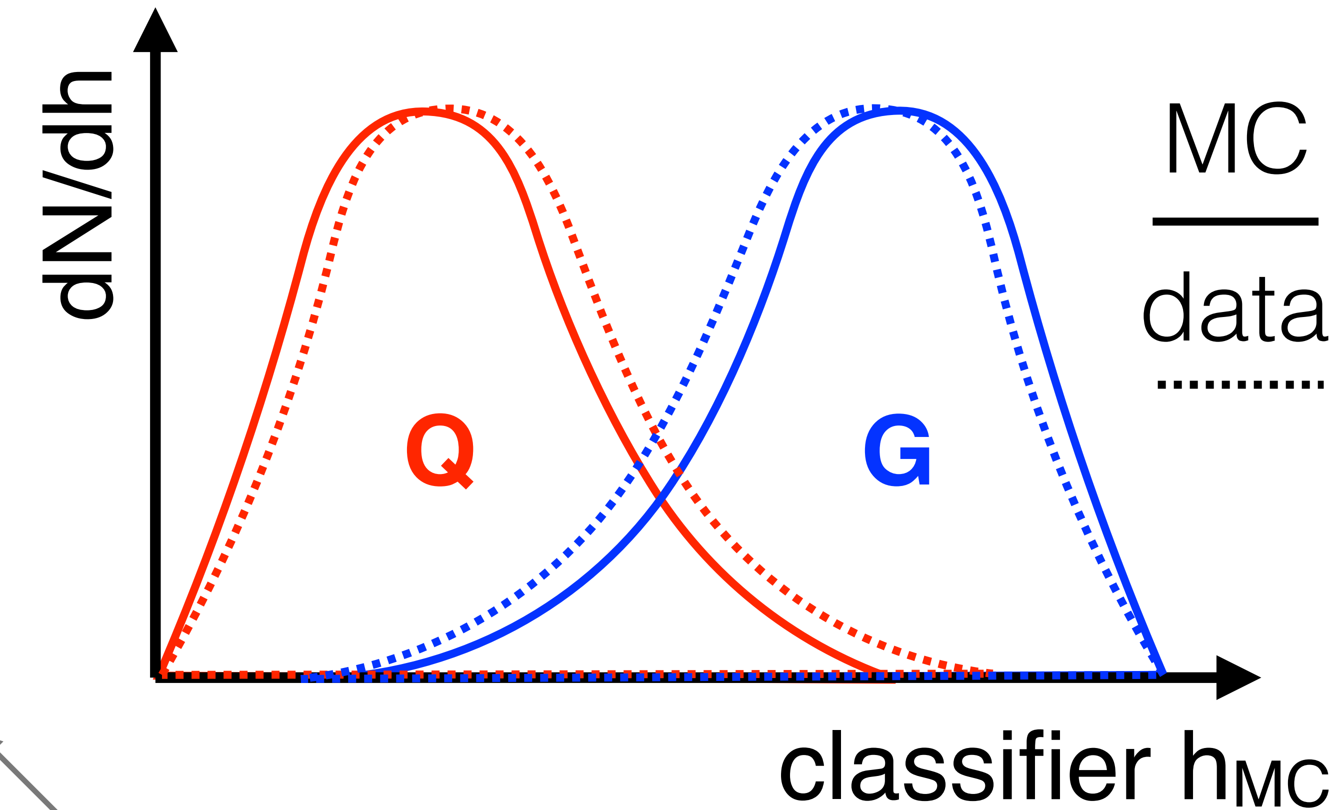
$\varepsilon_{\text{signal,MC}}$, $\varepsilon_{\text{back,MC}}$

# Background and Motivation



Usual paradigm: train in simulation, **validate on data**, test on data.

Determine the performance of the WP in data.




How did we get this?

dijets = $f_q$ x **Q** + ($1-f_q$) x **G**

Z+jets = $g_q$ x **Q** + ($1-g_q$) x **G**

two event samples with different q/g fractions

2 equations, 2 unknowns (**Q**, **G**)

(N.B. f & g from simulation and selection can't bias Q and G - more on that later)

# Background and Motivation



Usual paradigm: train in simulation, **validate on data**, test on data.

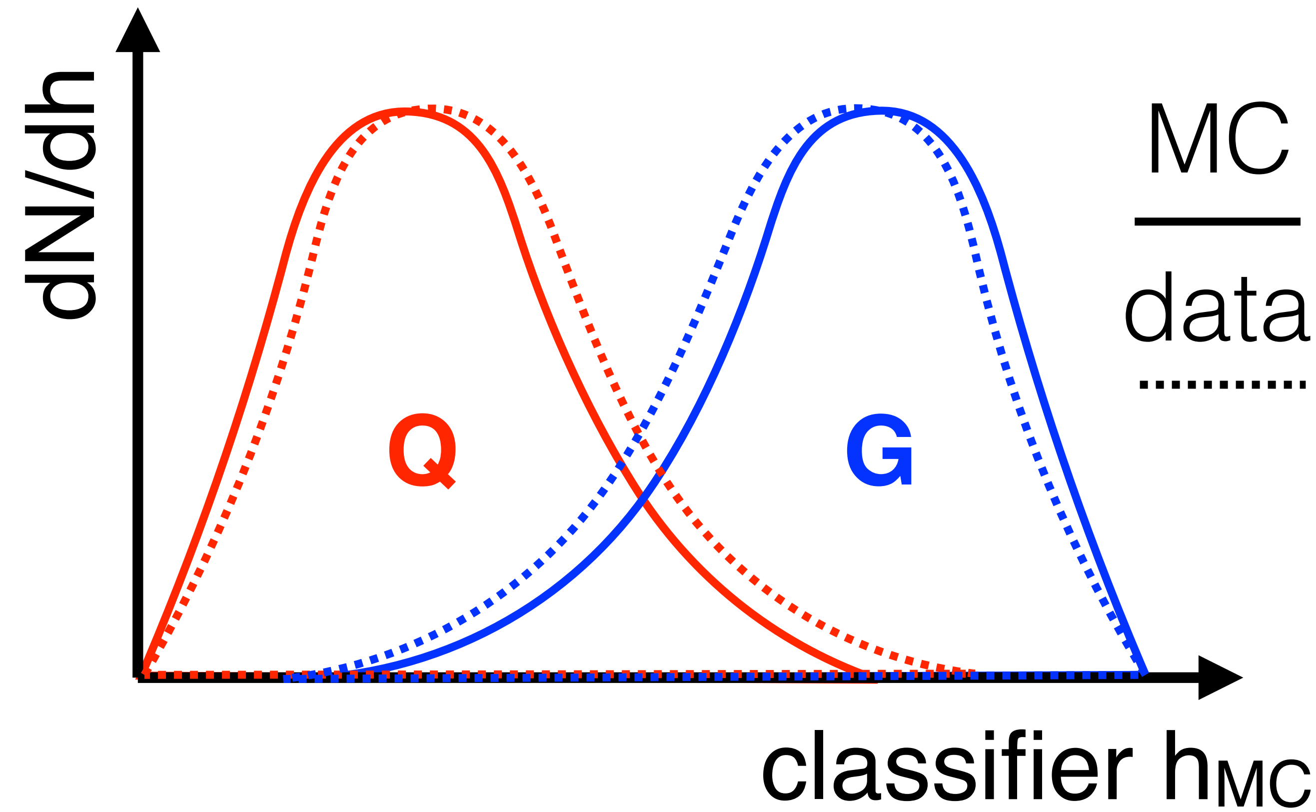
Determine the performance of the WP in data.

How did we get this?

dijets = $f_q$ x **Q** + (1-$f_q$) x **G**

Z+jets = $g_q$ x **Q** + (1-$g_q$) x **G**

2 equations, 2 unknowns (**Q**, **G**)




WP in data:

$\varepsilon_{\text{signal,data}}$, $\varepsilon_{\text{back,data}}$

**Can correct the MC to have the same performance as data.**

# Background and Motivation



Usual paradigm: train in simulation, validate on data, **test on data**.

Once we have scale factors (& their uncertainty), we can ensure that our analysis will be accurate.

…so what is the problem?

*remember my claim from earlier:*

| If data and simulation differ, this is sub-optimal! |
| --- |

This is an accuracy versus precision problem. It is "easy" to achieve accuracy through calibration, but the results may not be the best one possible.

# Background and Motivation



In this 2D feature space, we can actually derive $h_{data}$.

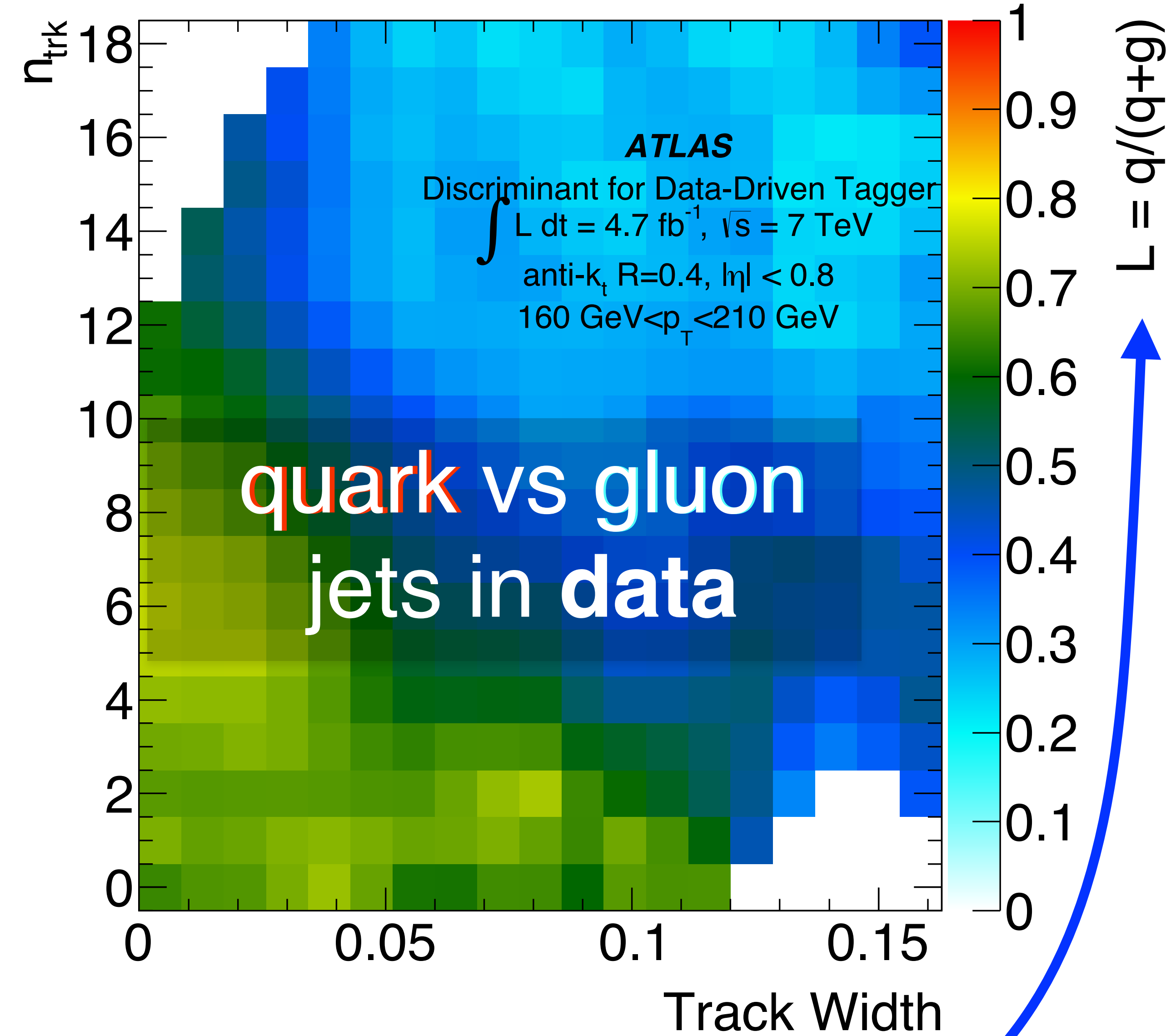
Using the same trick as earlier:

$$\text{dijets} = f_q \times Q + (1-f_q) \times G$$

$$\text{Z+jets} = g_q \times Q + (1-g_q) \times G$$

2 equations, 2 unknowns (Q, G)

(now Q and G are 2D histograms)




*in general:*

$h_{MC}(n_{trk}, \text{Track Width}) \neq h_{data}(n_{trk}, \text{Track Width})$

# Take it to the extreme

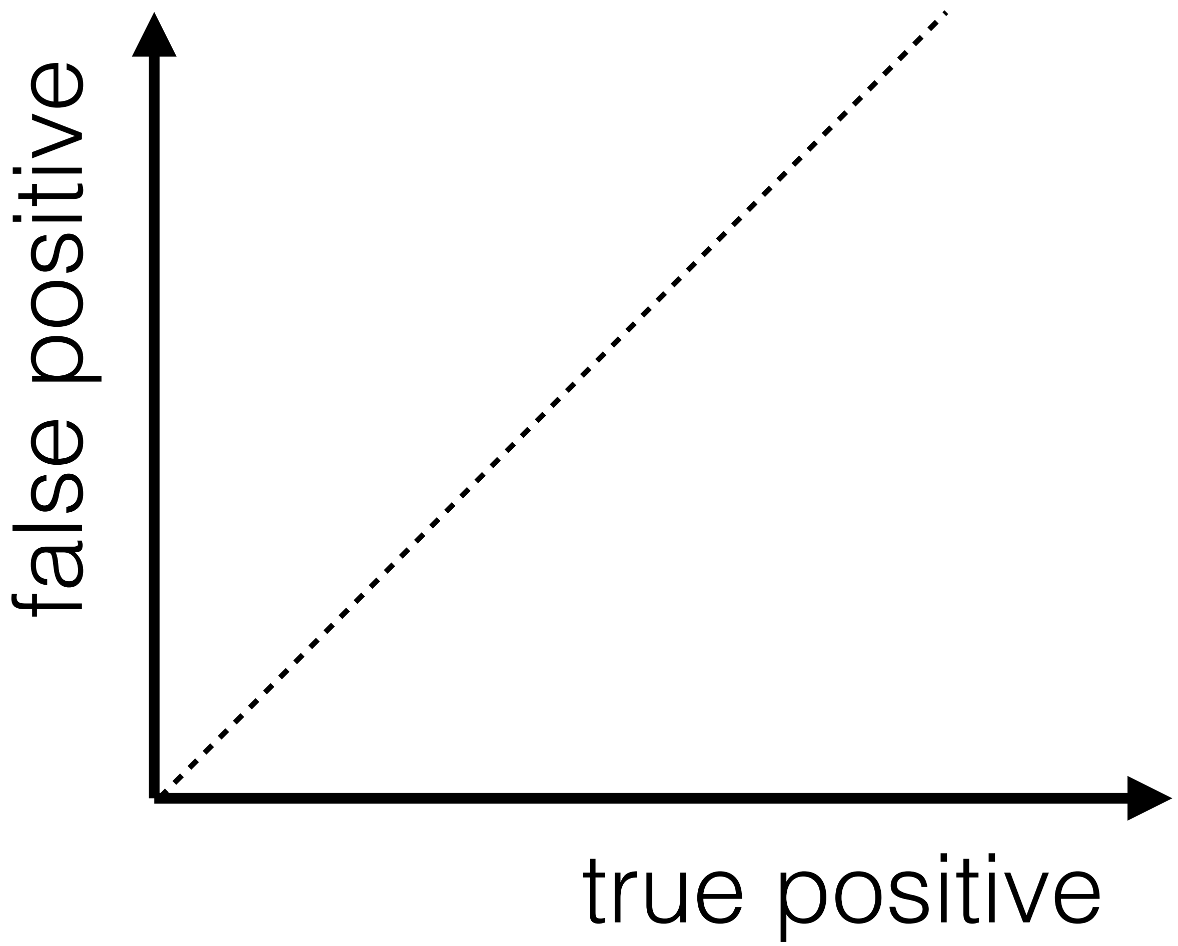


To stress this point, suppose that $h_{MC}$ is the random classifier:

$h_{MC}$ = 0 if you pick a random number x in [0,1] and $x < \varepsilon$

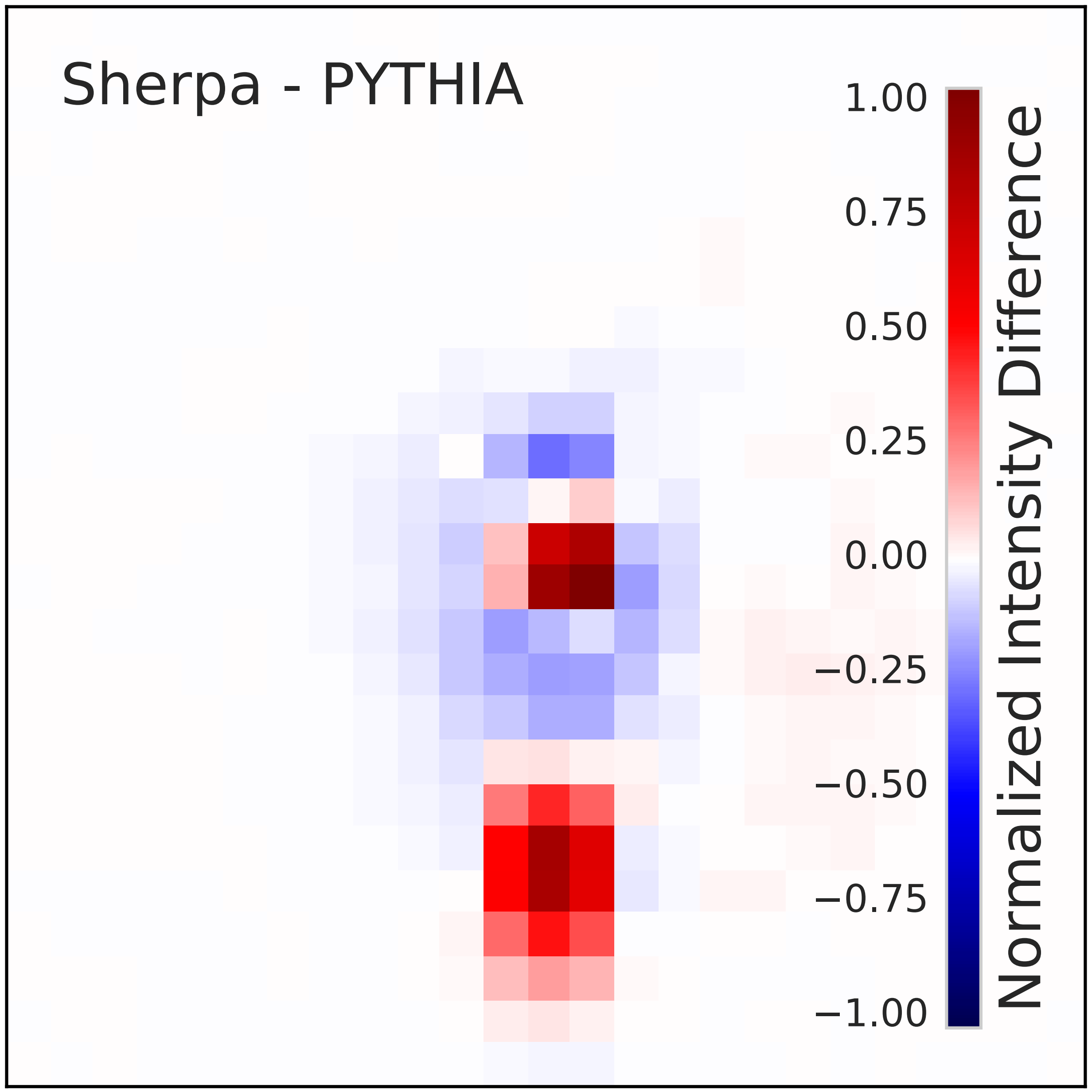
1 otherwise

We can calibrate this classifier in data, but clearly, it is sub-optimal !!



# One more slide about why it matters






J. Barnard, E. Dawe, M. Dolan, N. Rajcic, Phys. Rev. D 95 (2017) 014018

Especially important for **deep learning** using subtle features → hard to model!

*W boson radiation pattern - same physics, different simulators!*

# One solution: Learn from data!



One of the biggest challenges with any MC-based method is that it can't use information that the MC doesn't know about.

**One solution is to train directly on data !**

In general, this is not possible since data are unlabeled. However, in a wide range of cases, it is possible to work with less.

There is an interesting connection between what I'm calling "weak supervision" and the topic of "label noise".

# Weak supervision, caveats up front



The setup: suppose you have (at least) two mixed samples, each composed of two classes (say q and g).

Requirement:

The two classes are well-defined i.e. q in sample 1 is statistically identical to q in sample 2).

# Weak sup. option 1: Use class proportions



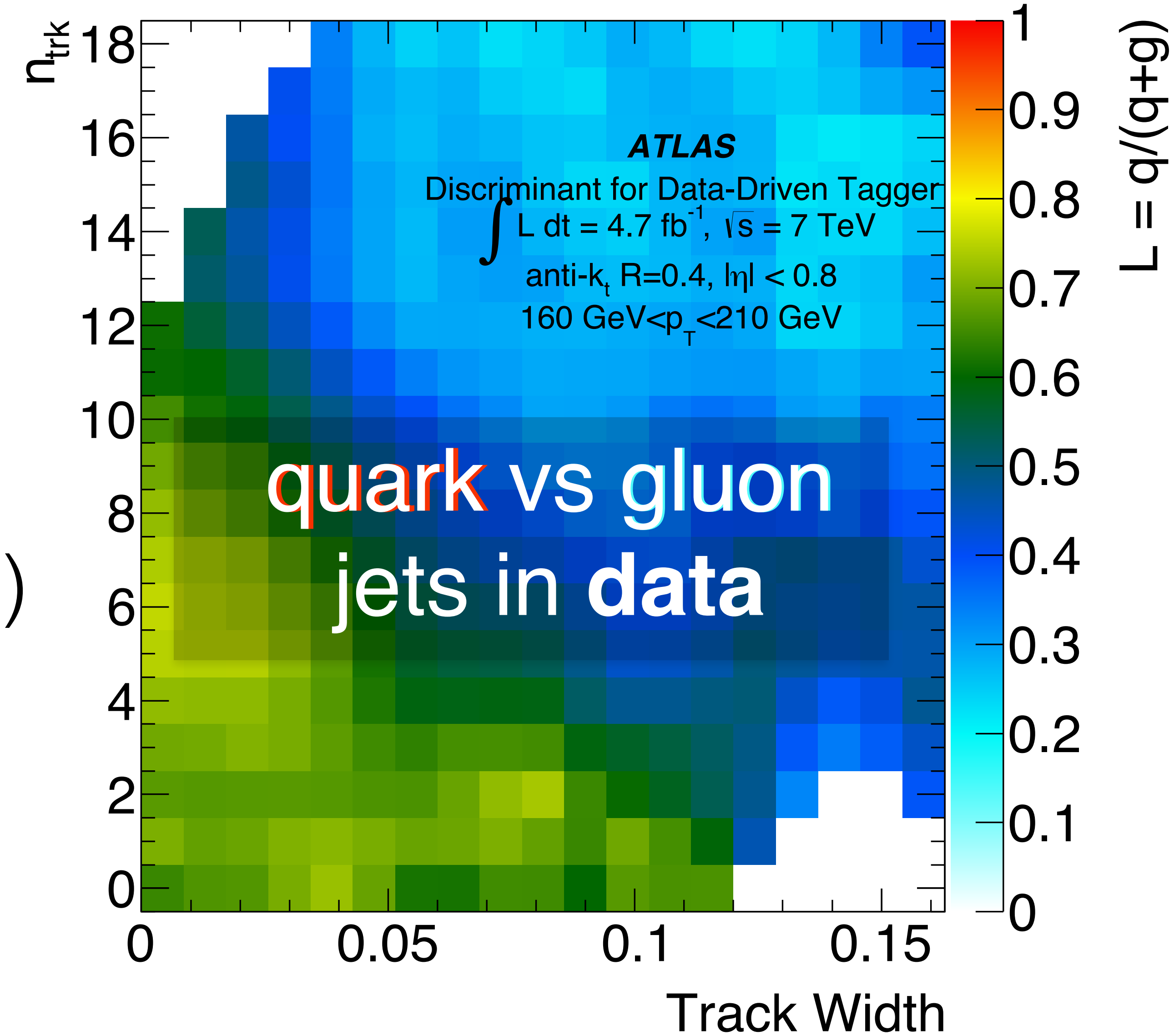
Remember this plot?

dijets = $f_q$ x Q + (1-$f_q$) x G

Z+jets = $g_q$ x Q + (1-$g_q$) x G

two equations, two unknowns (Q, G)

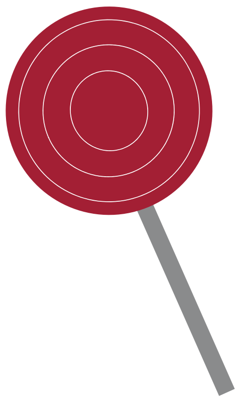
We often know f, g
(from ME + PDF) much better than
full radiation pattern inside jets.




This doesn't work well when you have more than 2 observables because the templates become sparse.

# Method 1: Learn from Proportions





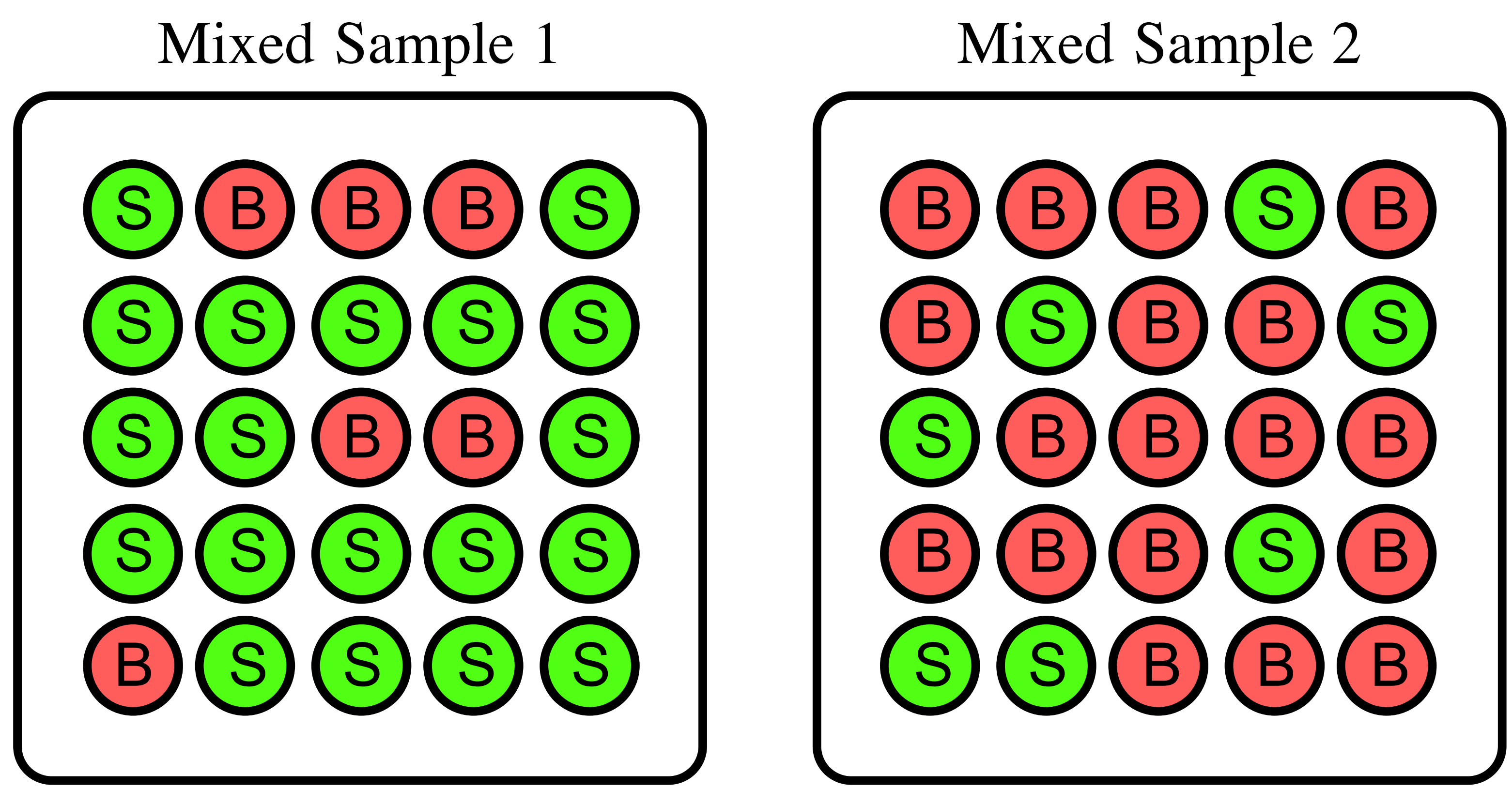
**LoLiProp**

*Learning from Label Proportions*

Solution: Train using class proportions. Work "on average"

loss fcn. labels

$$f_{\text{full}} = \text{argmin}_{f':\mathbb{R}^n \to \{0,1\}} \sum_{i=1}^{N} \ell\left(f'(x_i) - t_i\right)$$




$$f_{\text{weak}} = \text{argmin}_{f':\mathbb{R}^n \to [0,1]} \ell\left(\sum_{i=1}^{N} \frac{f'(x_i)}{N} - y\right)$$

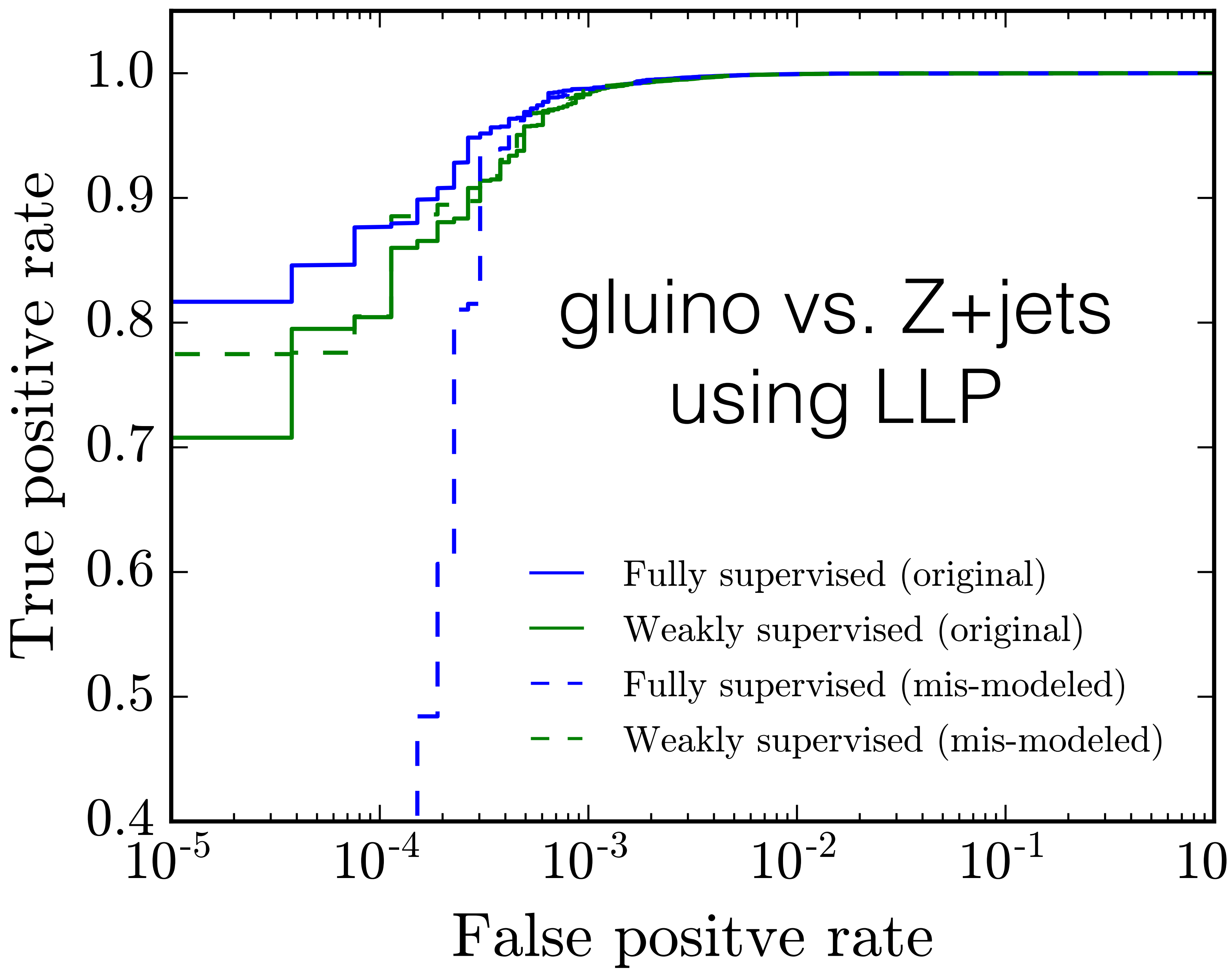
proportions

L. Dery, **BPN**, F. Rubbo, A. Schwartzman, JHEP 05 (2017) 145

# N.B. Don't need 100% fraction accuracy



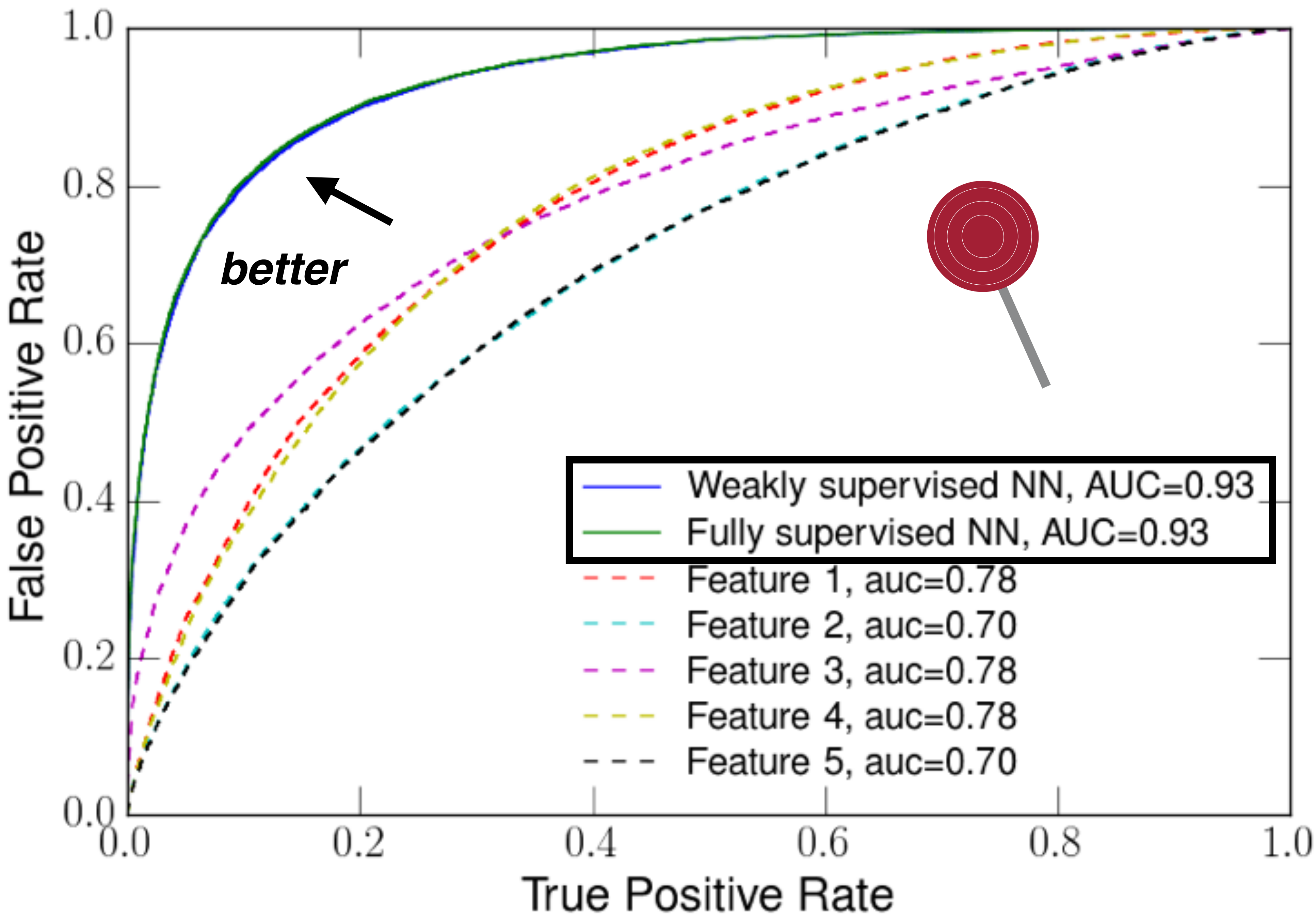
Even though the proportions are required as input, if they are slightly wrong, you can end up with the correct classifier.




T. Cohen, M. Freytsis, B. Ostdiek, JHEP 02 (2018) 034

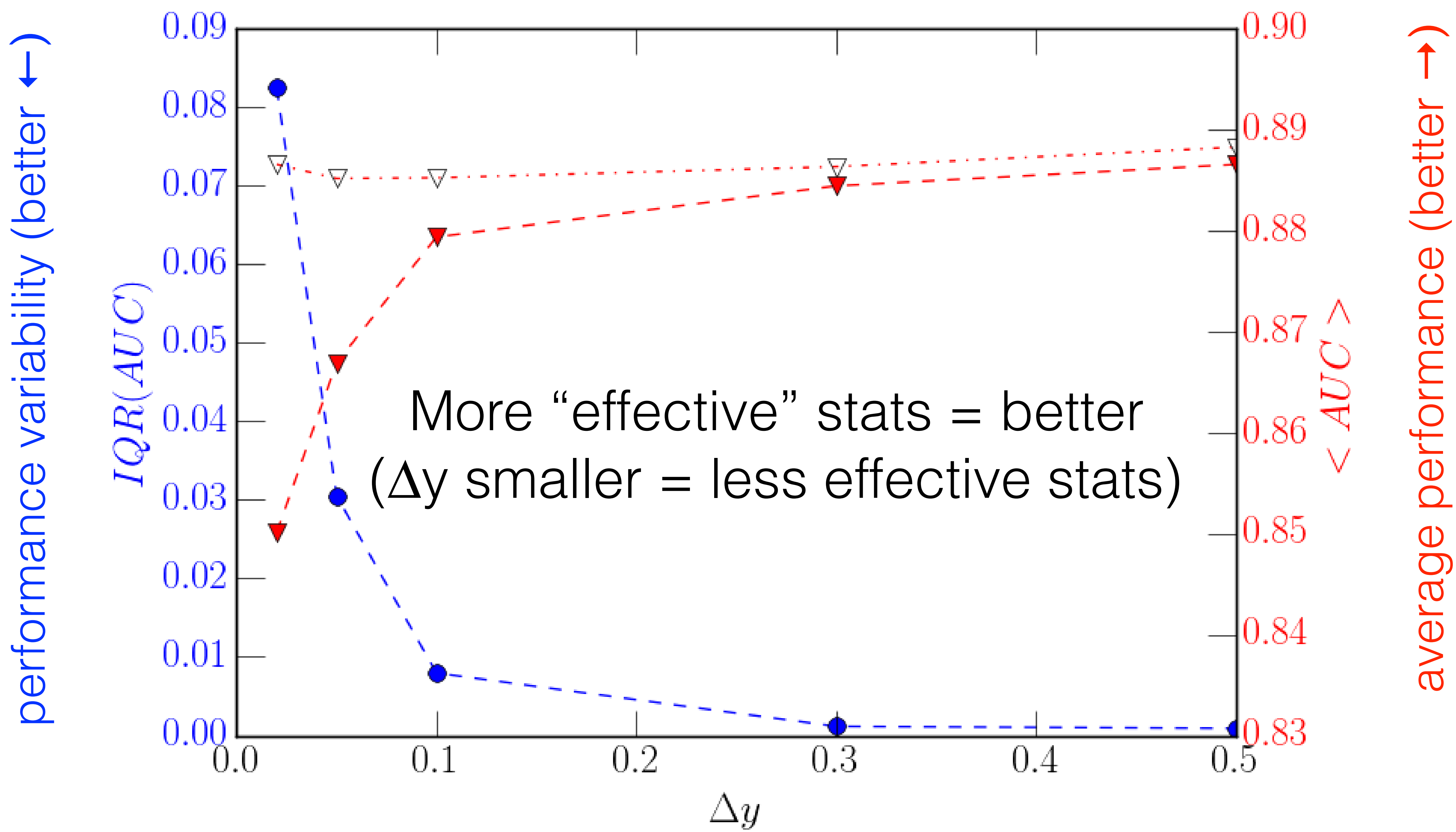
# Works in low-dimensions





# A note about training statistics





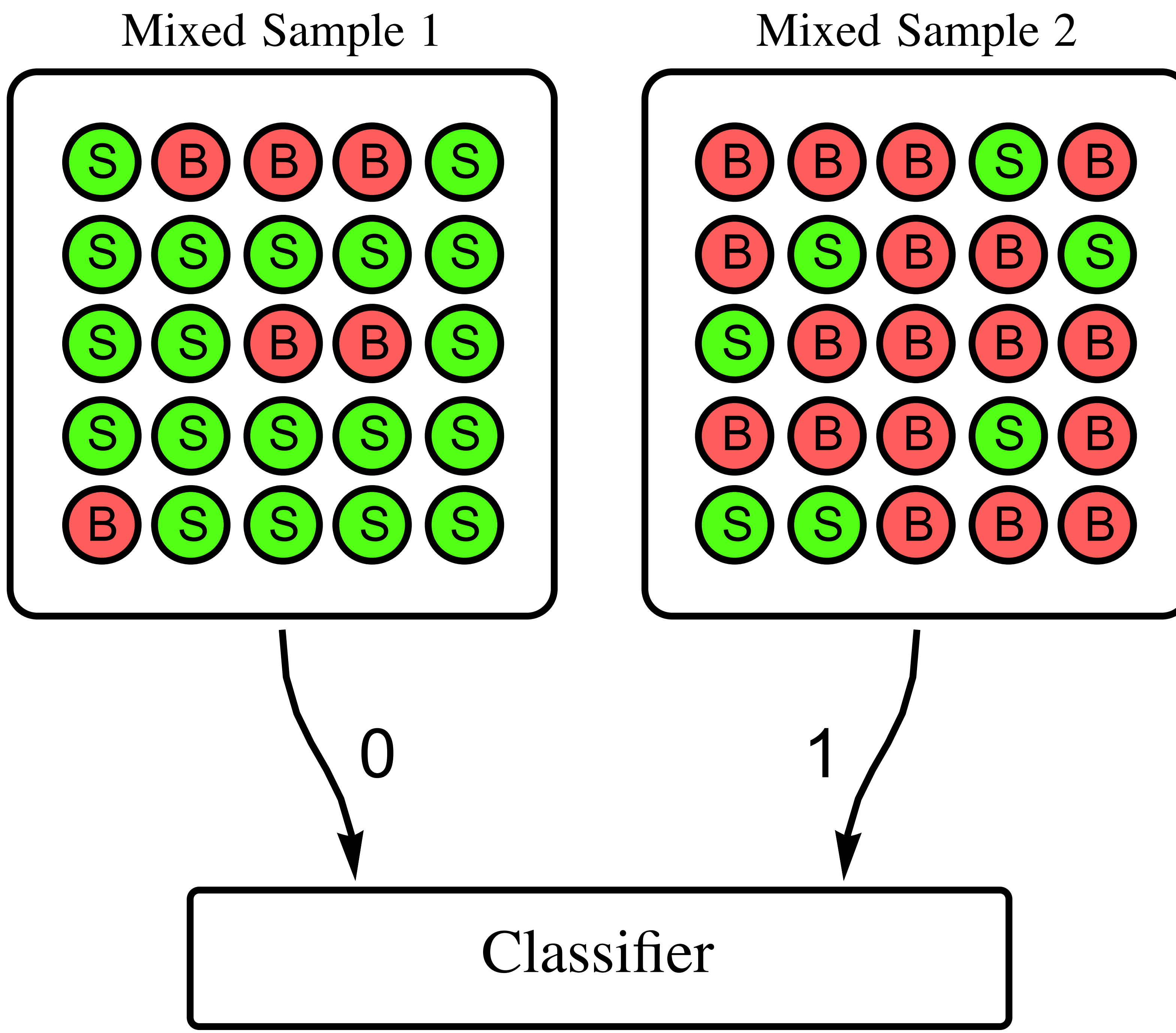

how different are the proportions for the two mixed samples

# Method 2: Learning without Proportions

**CWoLa**

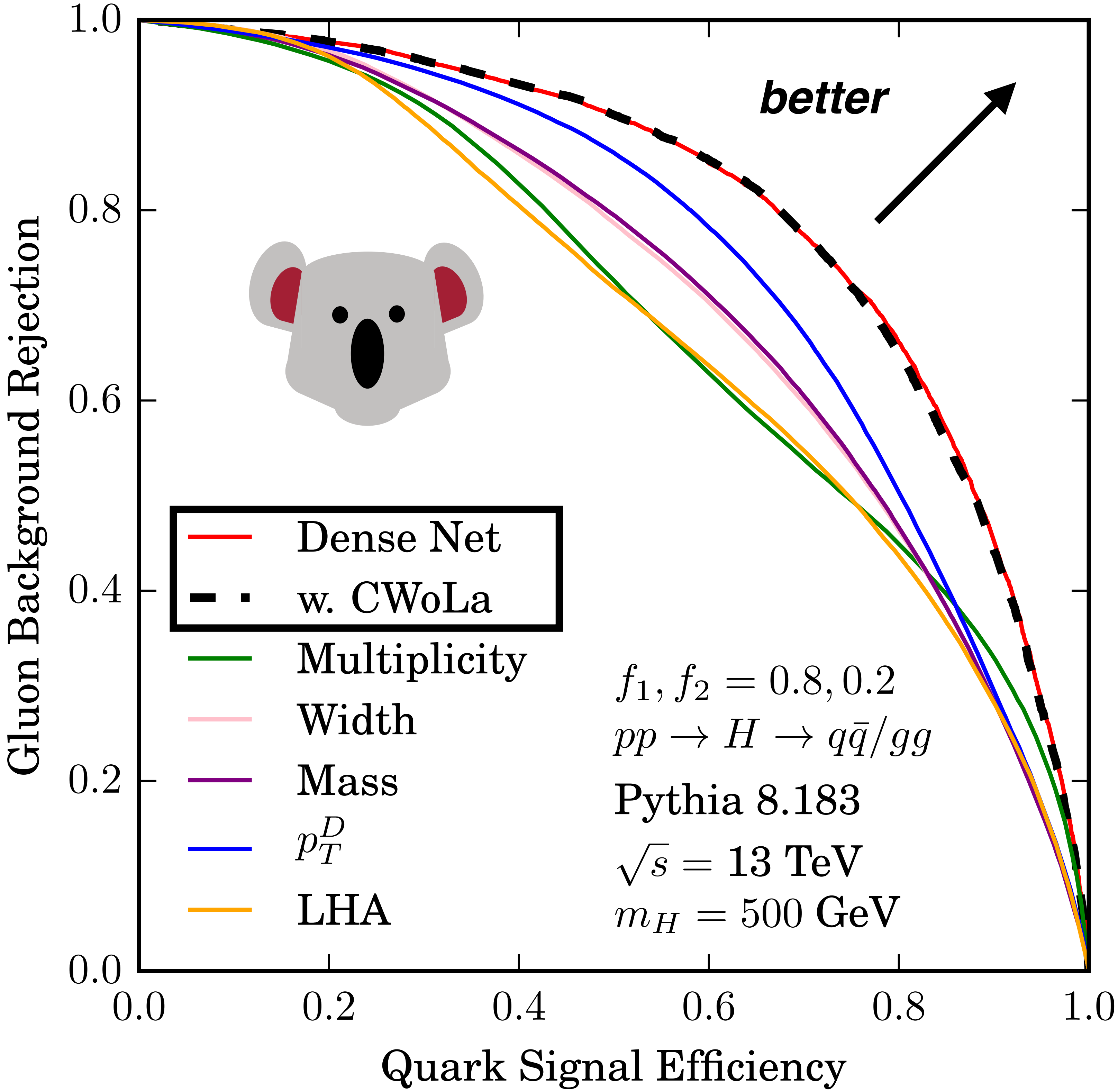
*Classification Without Labels*

Solution: Train **directly on data** using mixed samples



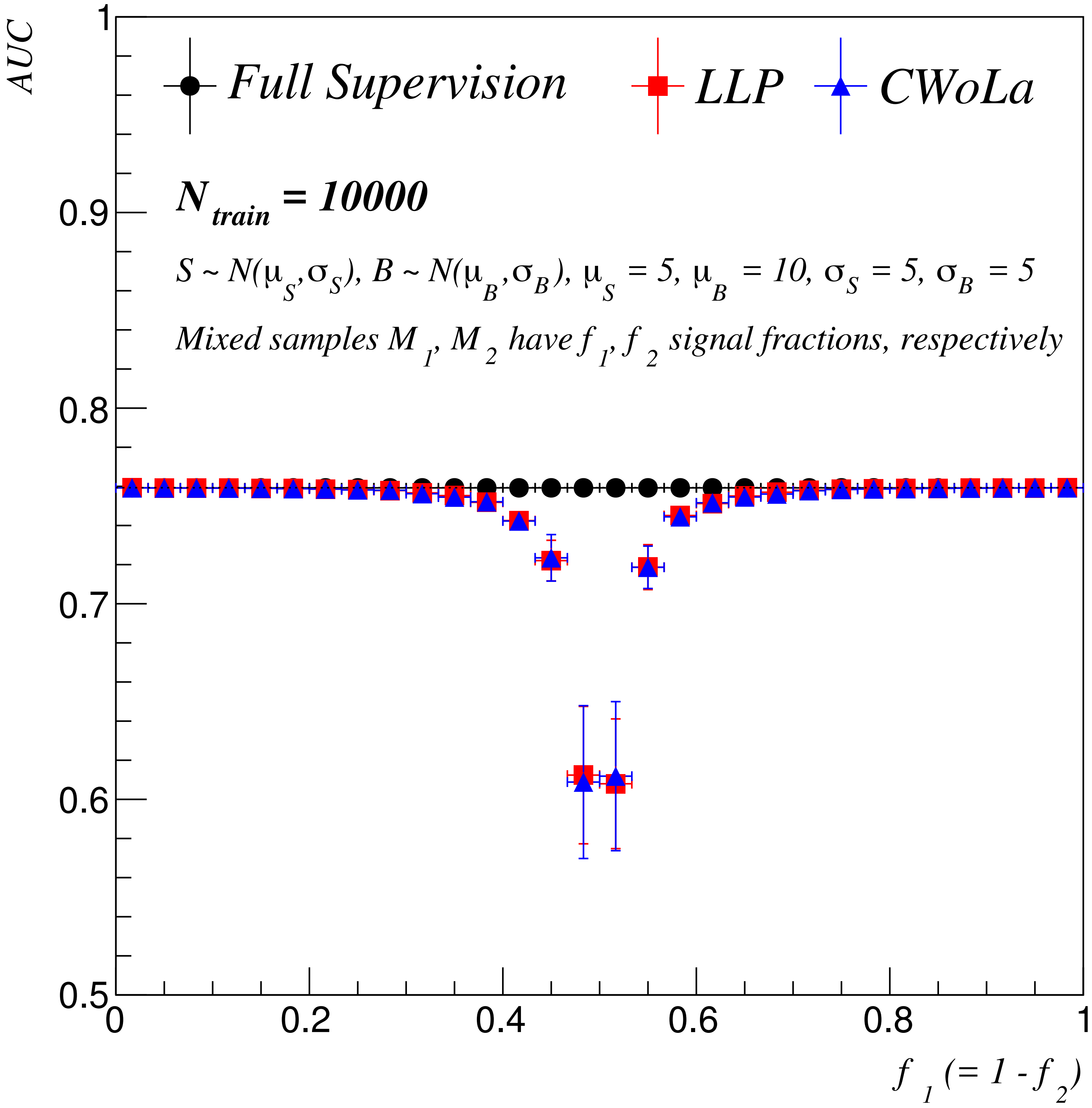

E. Metodiev, **BPN**, J. Thaler, JHEP 10 (2017) 51

# Works in low-dimensions



# A note about training statistics



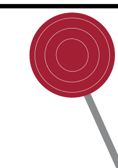
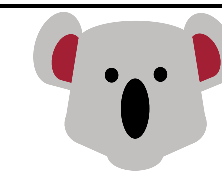



As with LLP, need sufficient effective statistics

Can't learn when the two proportions are the same.

# Methods Overview



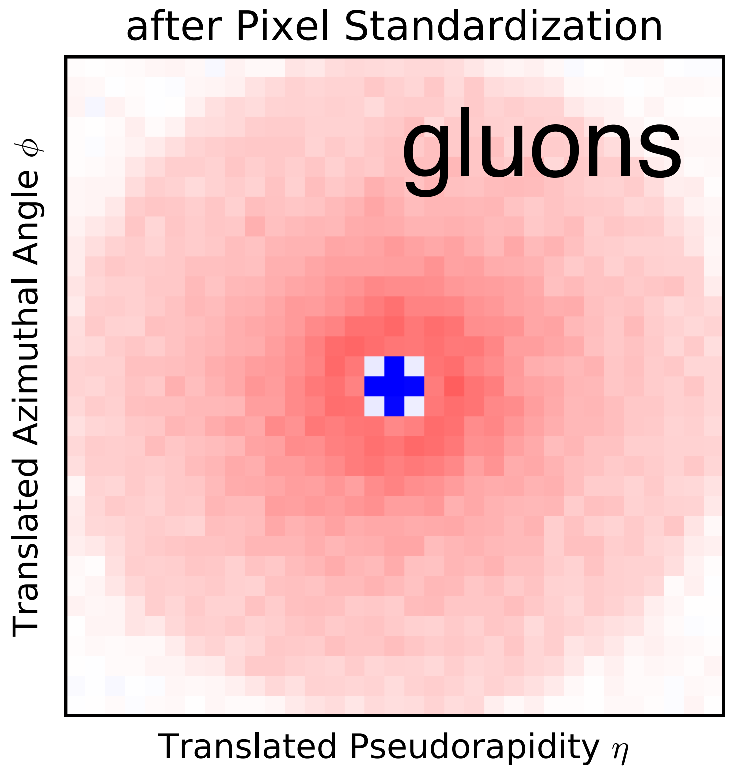
| Property |  LLP |  CWoLa |
|---|---|---|
| Compatible with any trainable model | ✓ | ✓ |
| No training modifications needed | ✗ | ✓ |
| Training does not need fractions | ✗ | ✓ |
| Smooth limit to full supervision | ✗ | ✓ |
| Works for $> 2$ mixed samples | ✓ | ? |

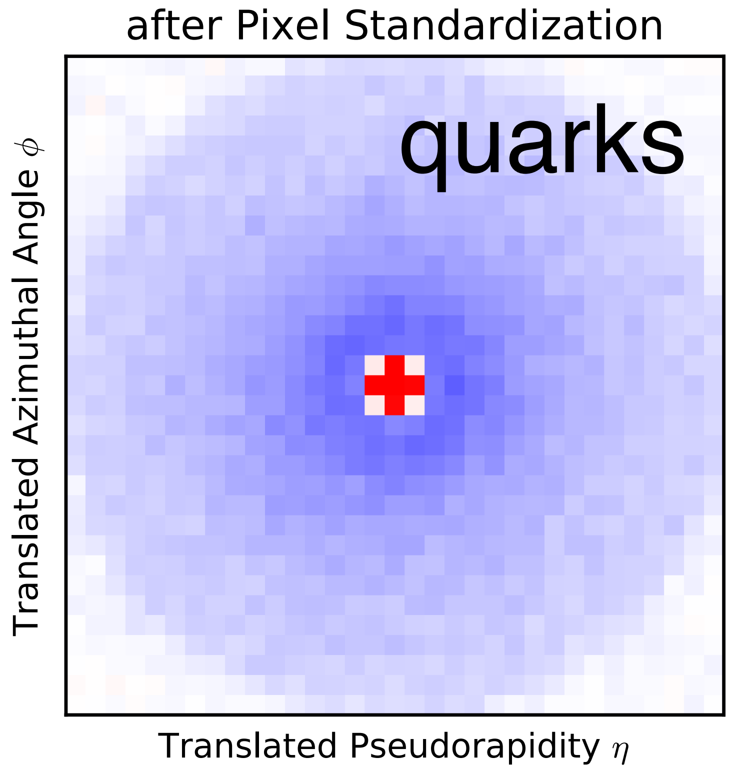
# Next step: what about high dim.?



There are many O(1)-dimensional ML problems for jets, but since the full radiation pattern is higher dimensional, need to go to bigger!

We'll use jet images as a testing ground, still focusing on quarks versus gluons.




P. Komiske, E. Metodiev, M. Schwartz, JHEP 01 (2017) 110

# Some Technical Details



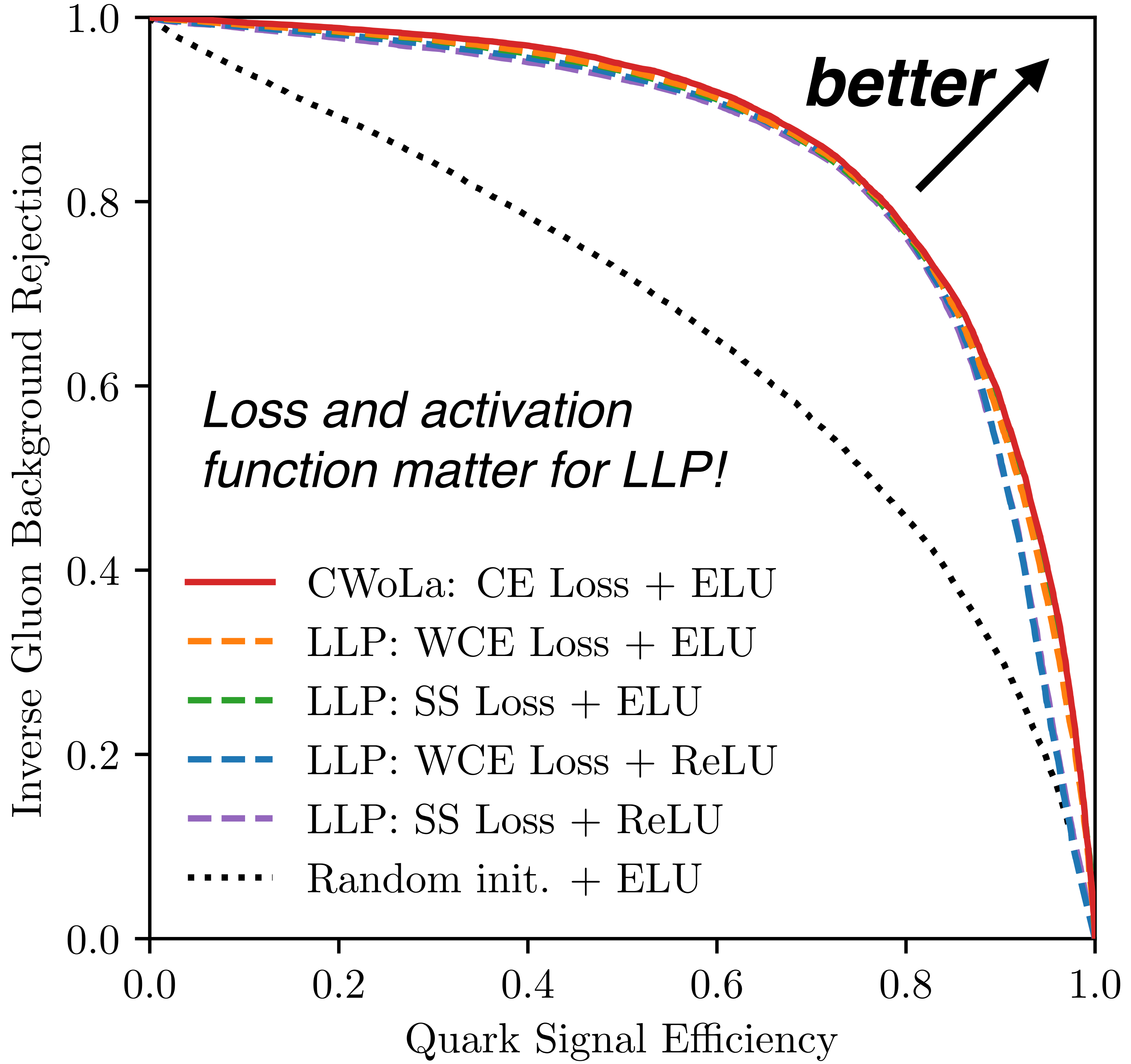
The CWoLa approach works out-of-the box - can use well-tested CNN architecture with usual cross-entropy loss.

On the other hand, LLP requires significant work on the technical implementation / optimization.

$$\ell_{\mathrm{WMSE}} = \sum_a \left( f_a - \frac{1}{N} \sum_{i=1}^{N} h(\mathbf{x}_i) \right)^2 \qquad \ell_{\mathrm{WCE}} = \sum_a \mathrm{CE}\left( f_a, \frac{1}{N} \sum_{i=1}^{N} h(\mathbf{x}_i) \right)$$

# Works in many-dimensions!

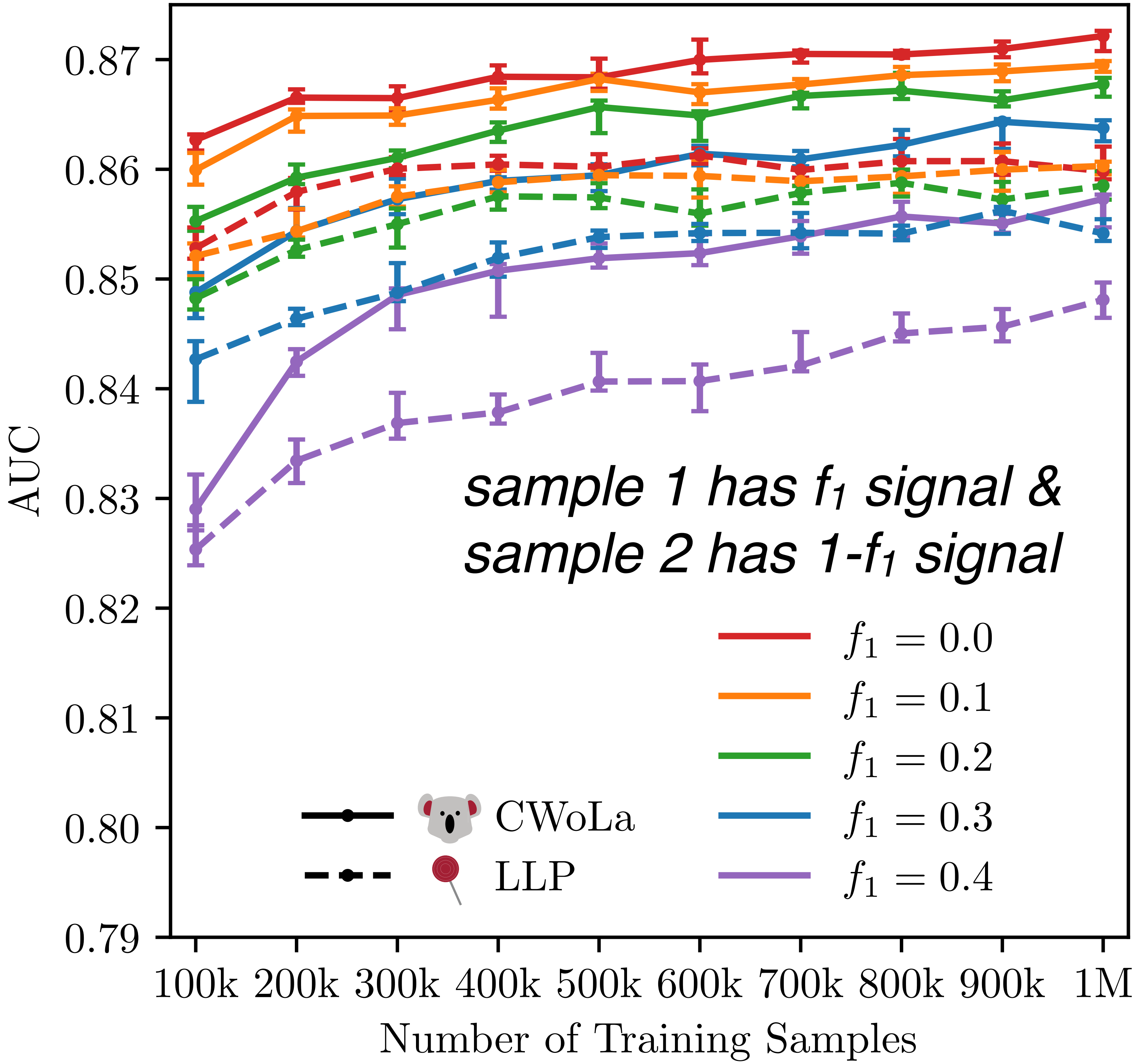





P. Komiske, E. Metodiev, **BPN**, M. Schwartz, Phys. Rev. D 98, 011502(R), arXiv:1801:10158

# A note about training statistics





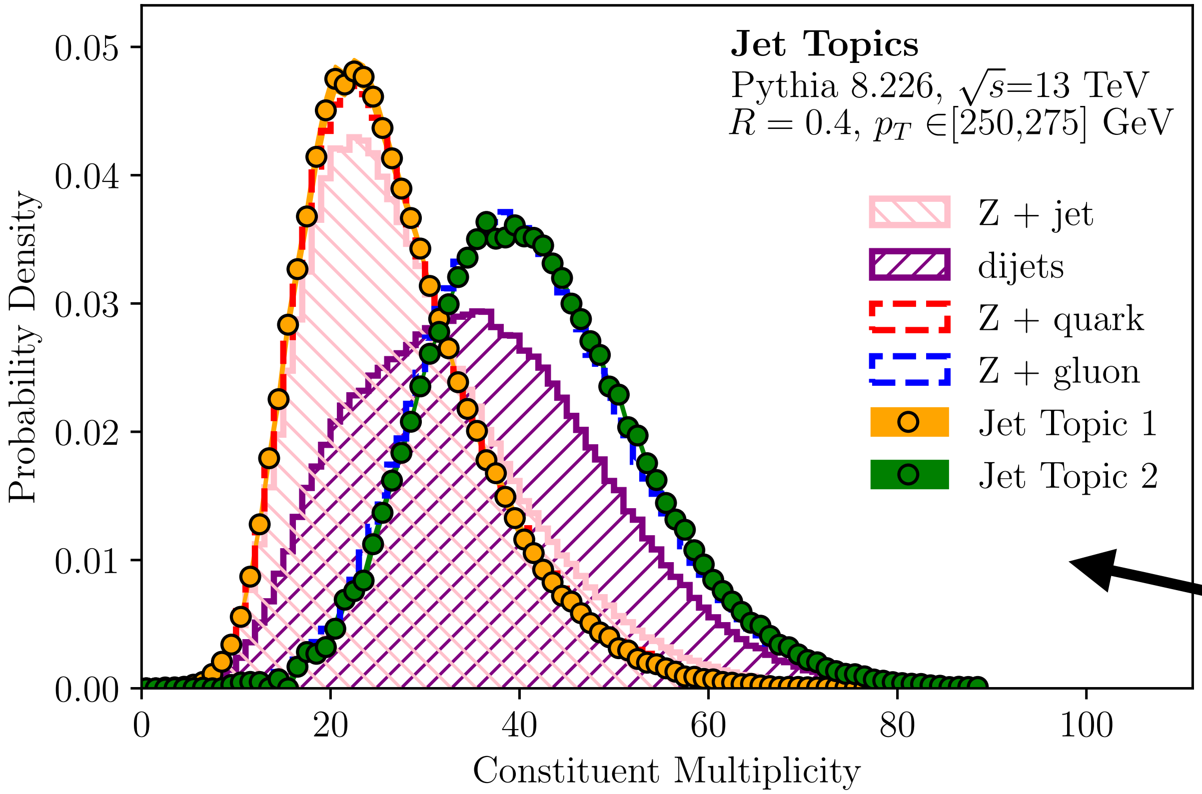
# Hybrid Approaches



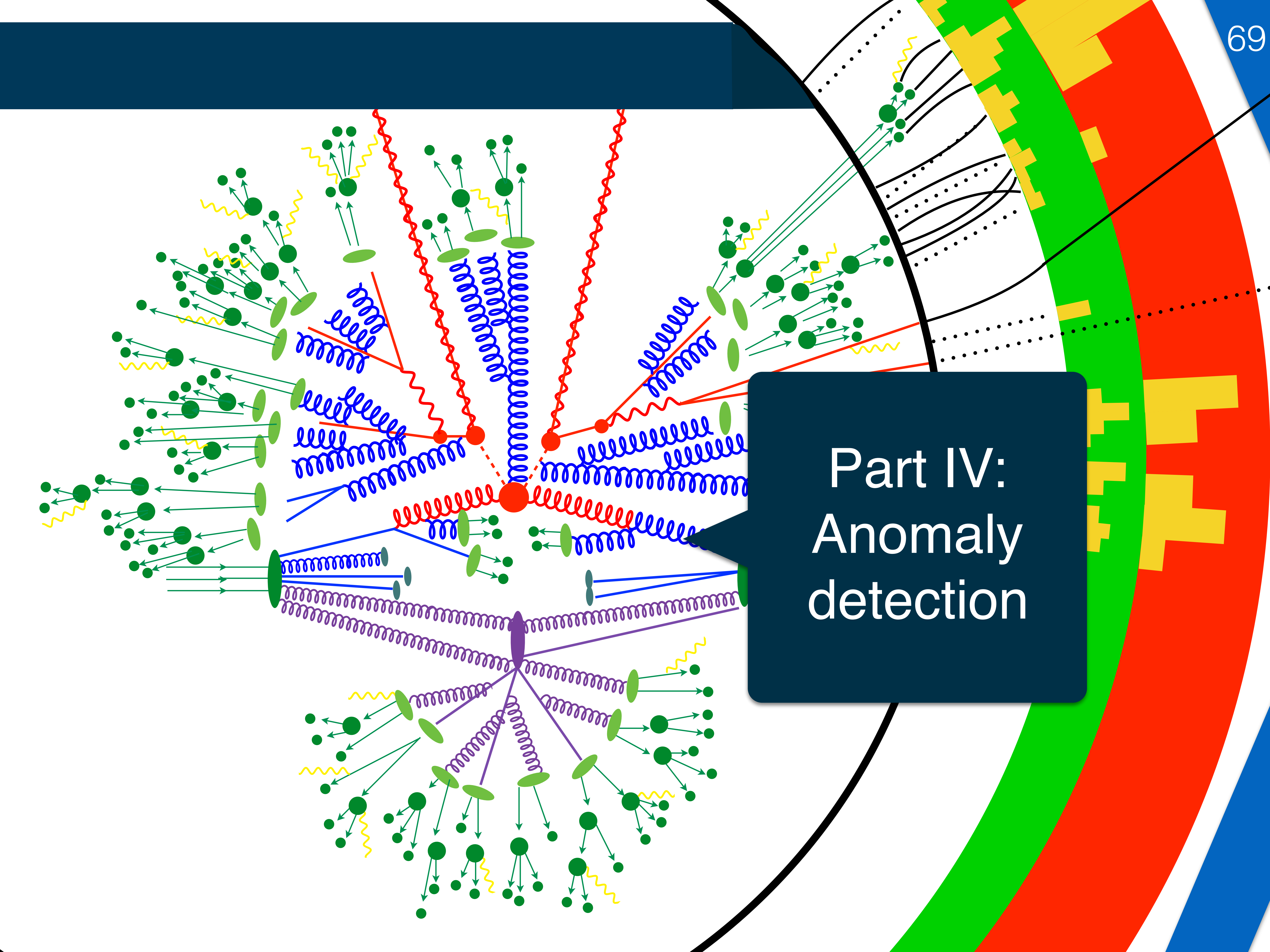
As usual, it is likely that the best approach will use all of the available information, including some input from simulation.

…this could be as simple as pre-training in MC and then running weak supervision

E. Metodiev and J. Thaler, Phys. Rev. Lett. 120 (2018) 241602




Another interesting direction is to push the weak supervision paradigm a step further and **define** the classes so that it works.



Part IV:
Anomaly
detection

# Anomaly Detection



I will leave you with one last, but very exciting topic.

One of the most important goals of HEP is to search for new particles. However, we have not found anything (significantly) unexpected in a while … we need simulation-independent ways of searching for new particles !

anomalies, i.e. something unexpected

N.B. The approach discussed here is not the only one - see also M. Farina, Y. Nakai, D. Shih, 1808.08992 & T. Heimel, G. Kasieczka, T. Phlen, J. Thompson, 1808.08979 for an alternative approach based on auto-encoders.

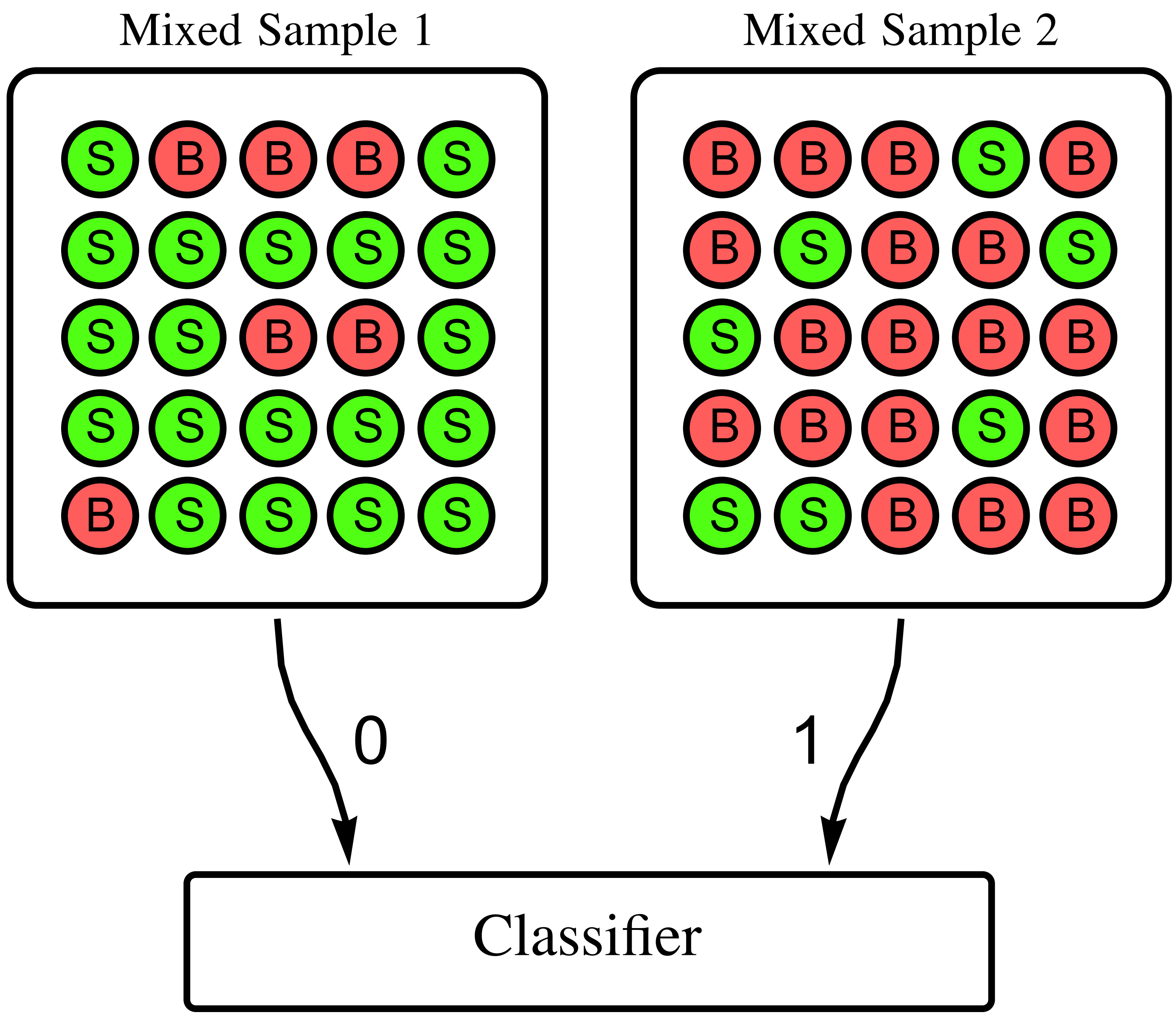
# Remember CWoLa …



**CWoLa**

*Classification Without Labels*

Solution: Train **directly on data** using mixed samples




E. Metodiev, **BPN**, J. Thaler, JHEP 10 (2017) 51

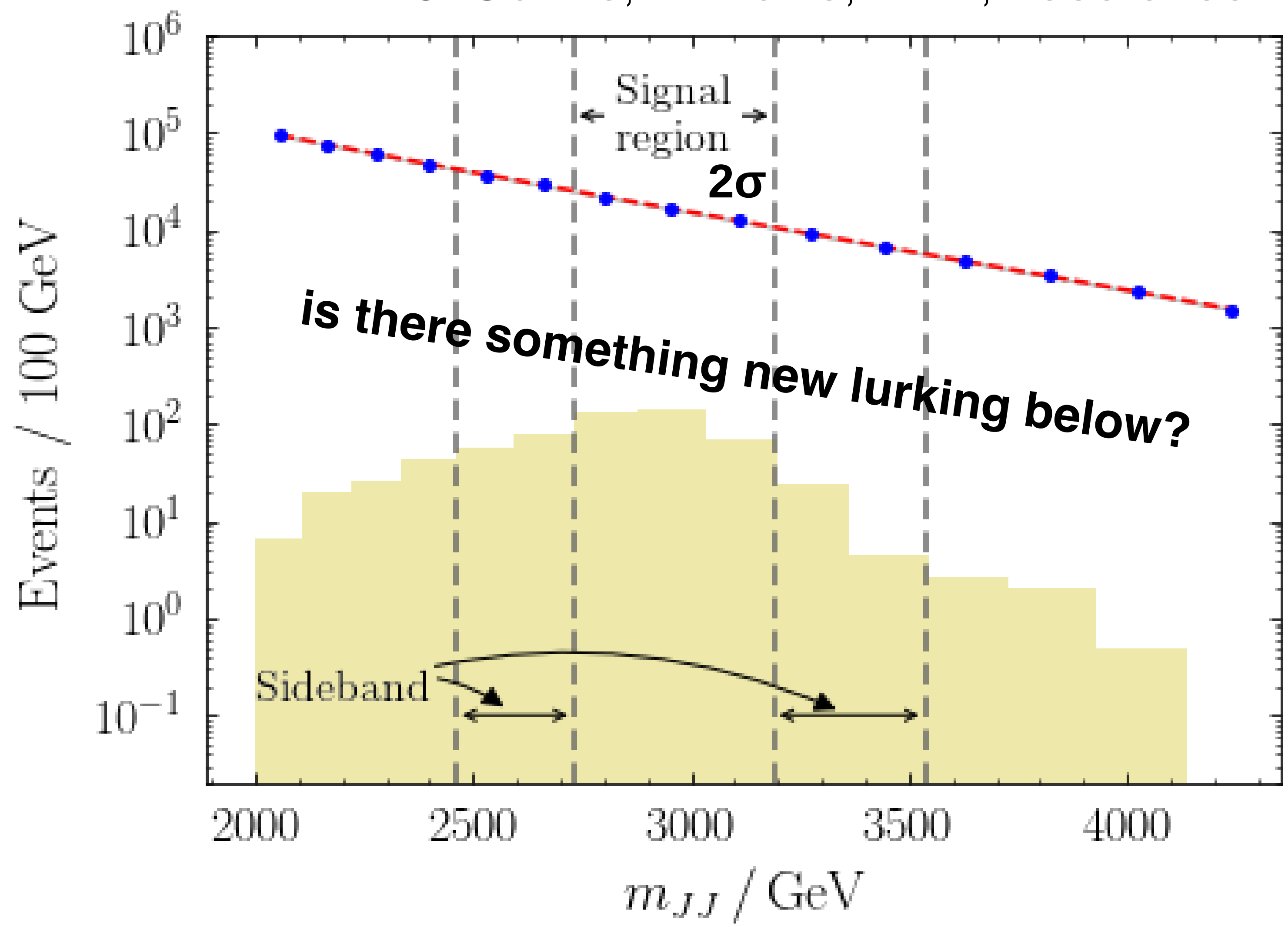
# Weak/unsupervised learning for anomalies



Can we take this idea one step further to look for something unexpected?

= CWoLa Hunting*

J. Collins, K. Howe, **BPN**, 1805.02664




Minimal assumption: if there is a signal, it is localized in one known dimension.

*Image from this article. This Koala is actually being freed - I do not condone violence against these animals!

# Weak/unsupervised learning for anomalies

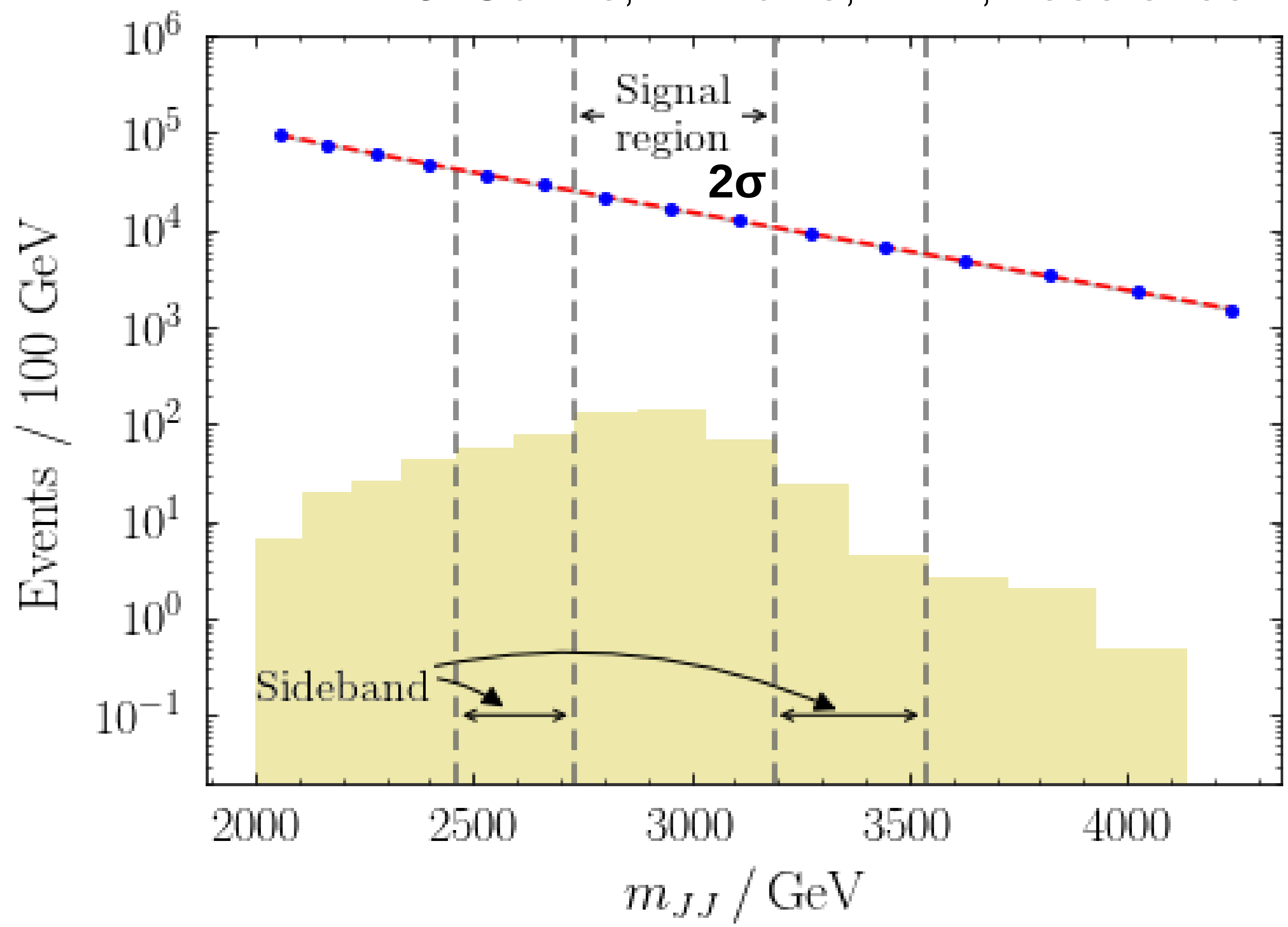


Mixed sample 1:
signal region

Mixed sample 2:
sideband region

Train a classifier to distinguish the two mixed samples.

J. Collins, K. Howe, **BPN**, 1805.02664

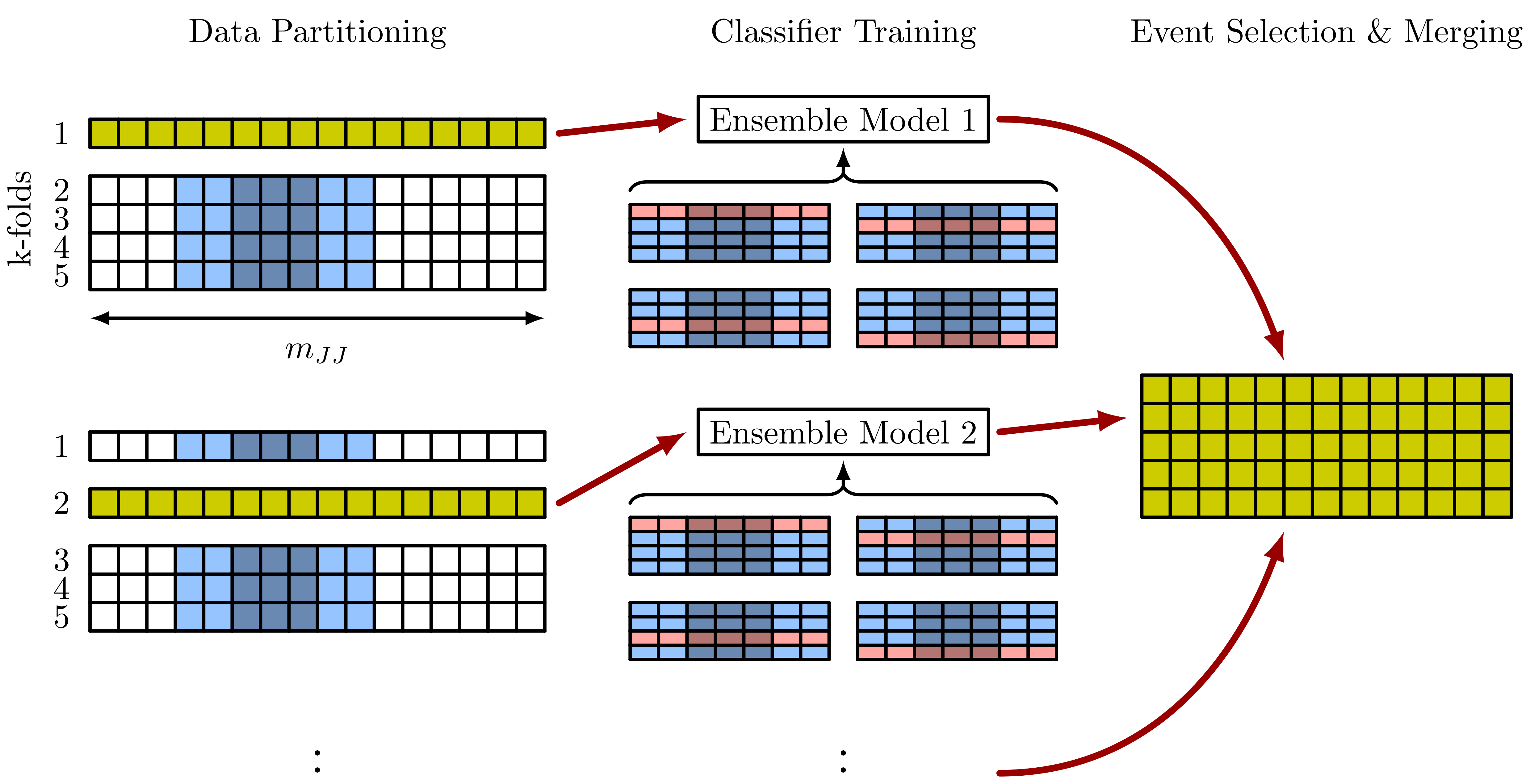



If there is a signal, there will be something to learn and the signal will be enhanced. If no signal, nothing to learn.

# Weak/unsupervised learning for anomalies



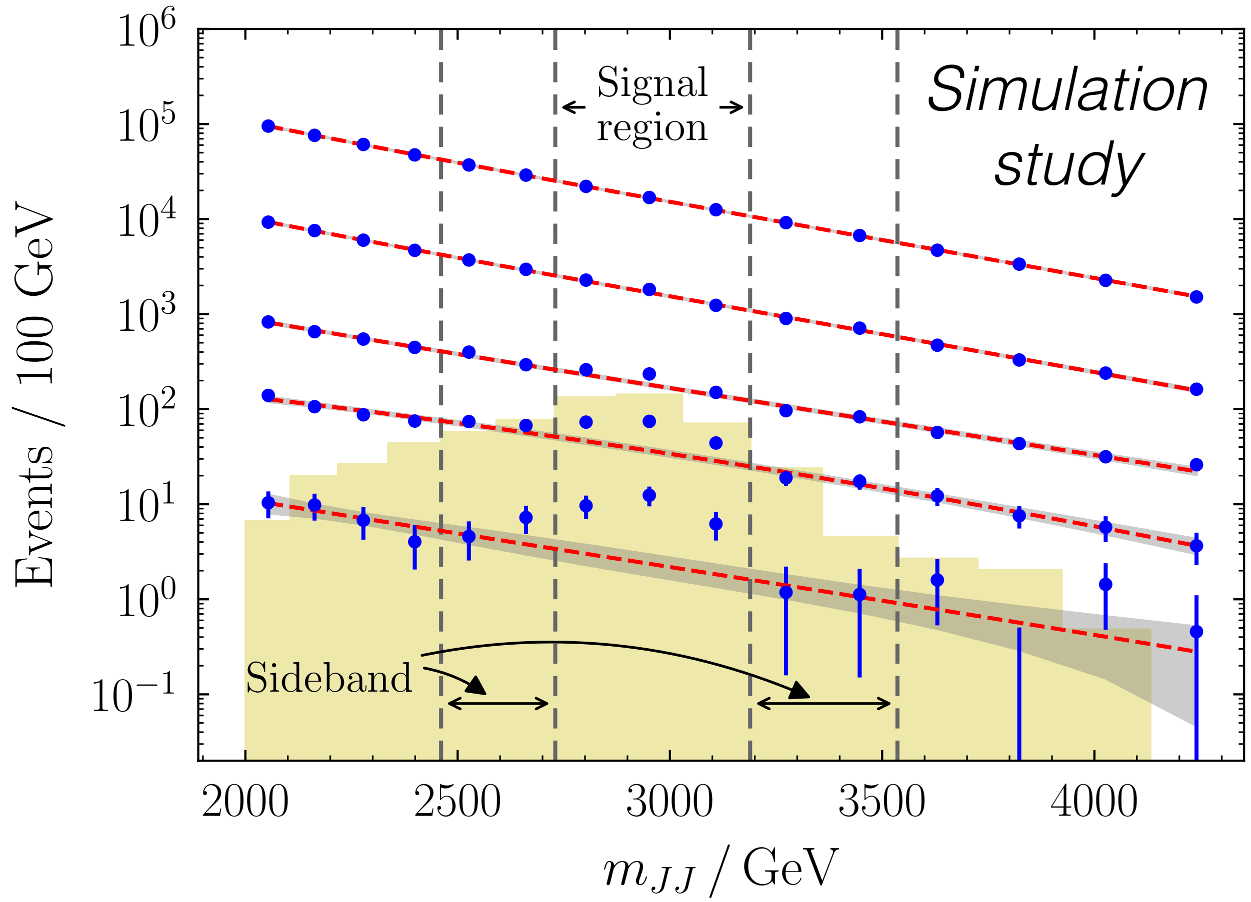
Need to be careful about testing/training on the same data.

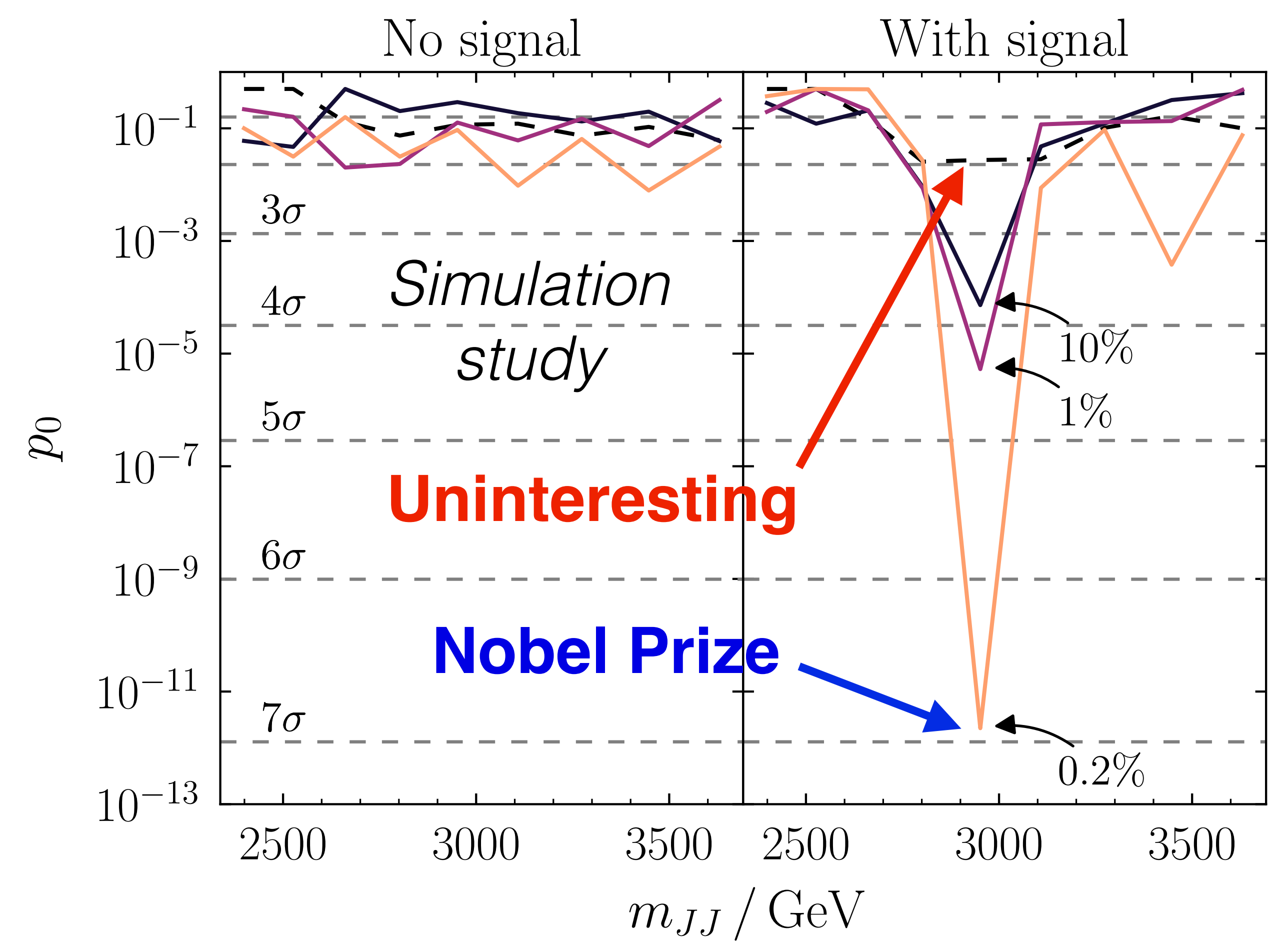


# Weak/unsupervised learning for anomalies



J. Collins, K. Howe, **BPN**, 1805.02664

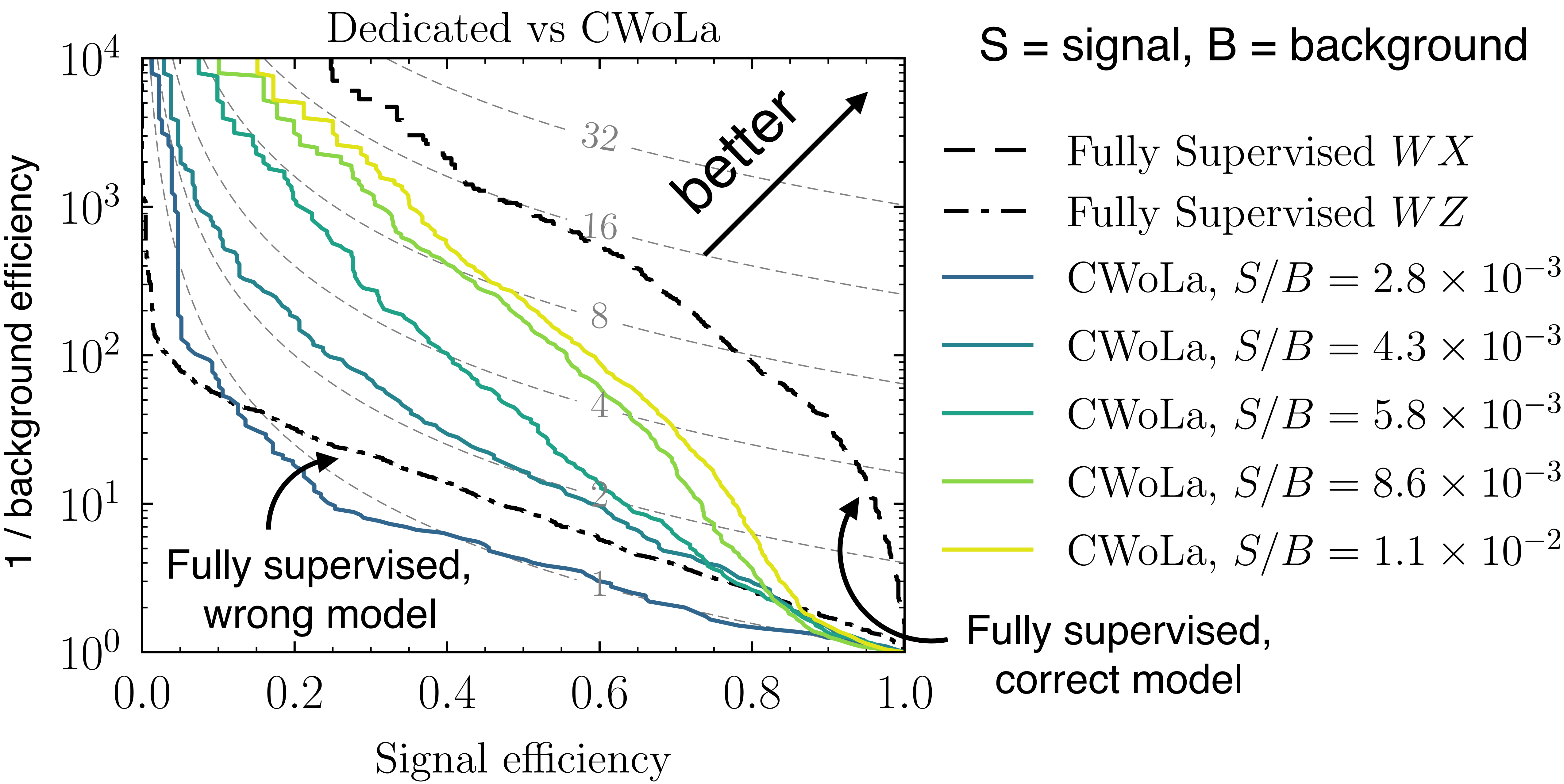



Using a classifier trained to distinguish a signal region from a sideband, make progressively harsher cuts on the NN output

# CWoLa hunting vs. Full Supervision






If you know what you are looking for, you should look for it. If you don't know, then CWoLa hunting may be able to catch it!

# Review of CWoLa hunting



(1) Need an observable X (e.g. $m_{JJ}$) where the signal is localized and the background is not.

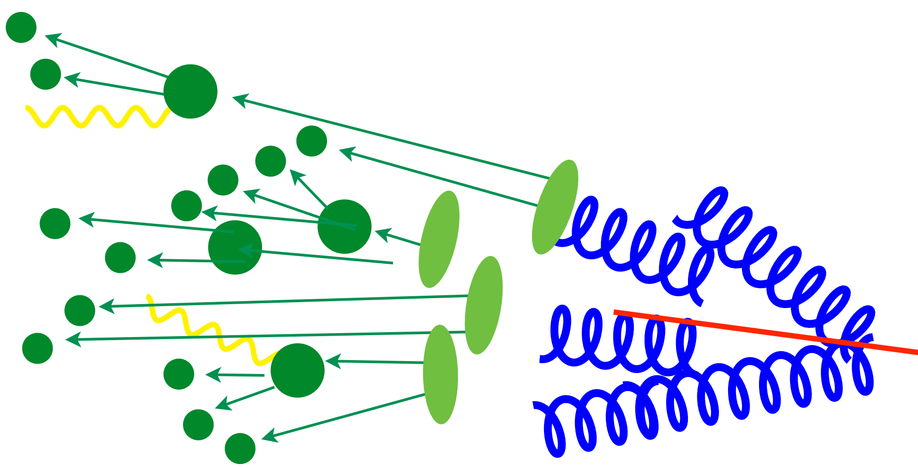
(2) Identify features Y (e.g. jet substructure) that are ~independent of X, but can be useful for identifying a broad range of new particles.

actually, we don't need independence, we just need them to not allow us to sculpt bumps.

# Summary



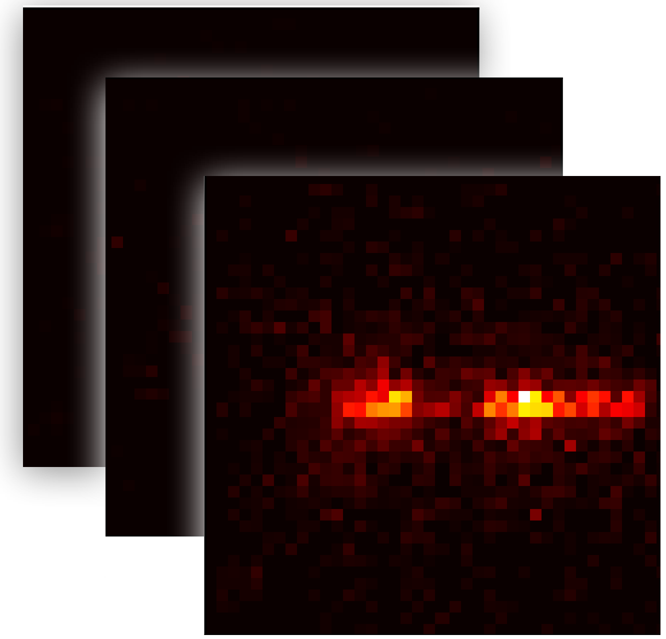
- **Generative models**: accelerate simulations with deep learning
- **Generalized numerical inversion**: machine learn with prior independence
- **Weak supervision**: learn directly from (unlabeled) data
- CWoLa hunting: model agnostic **anomaly detection**



# Conclusions and Outlook



Accelerating simulation/reducing simulation dependence can improve performance and may even help us to understand something new and fundamental about nature!



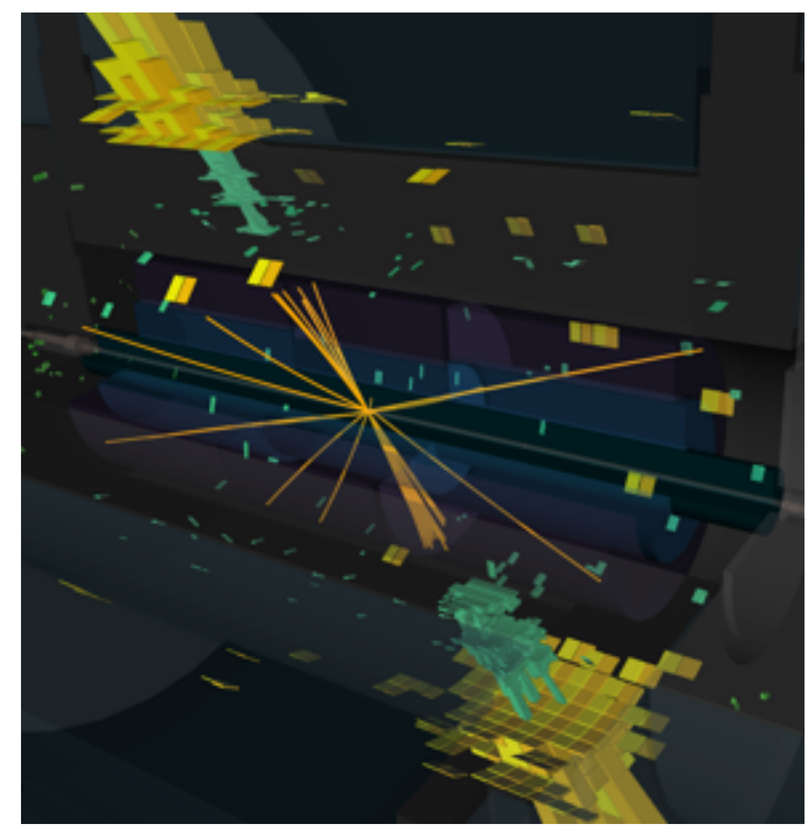
**Classification**

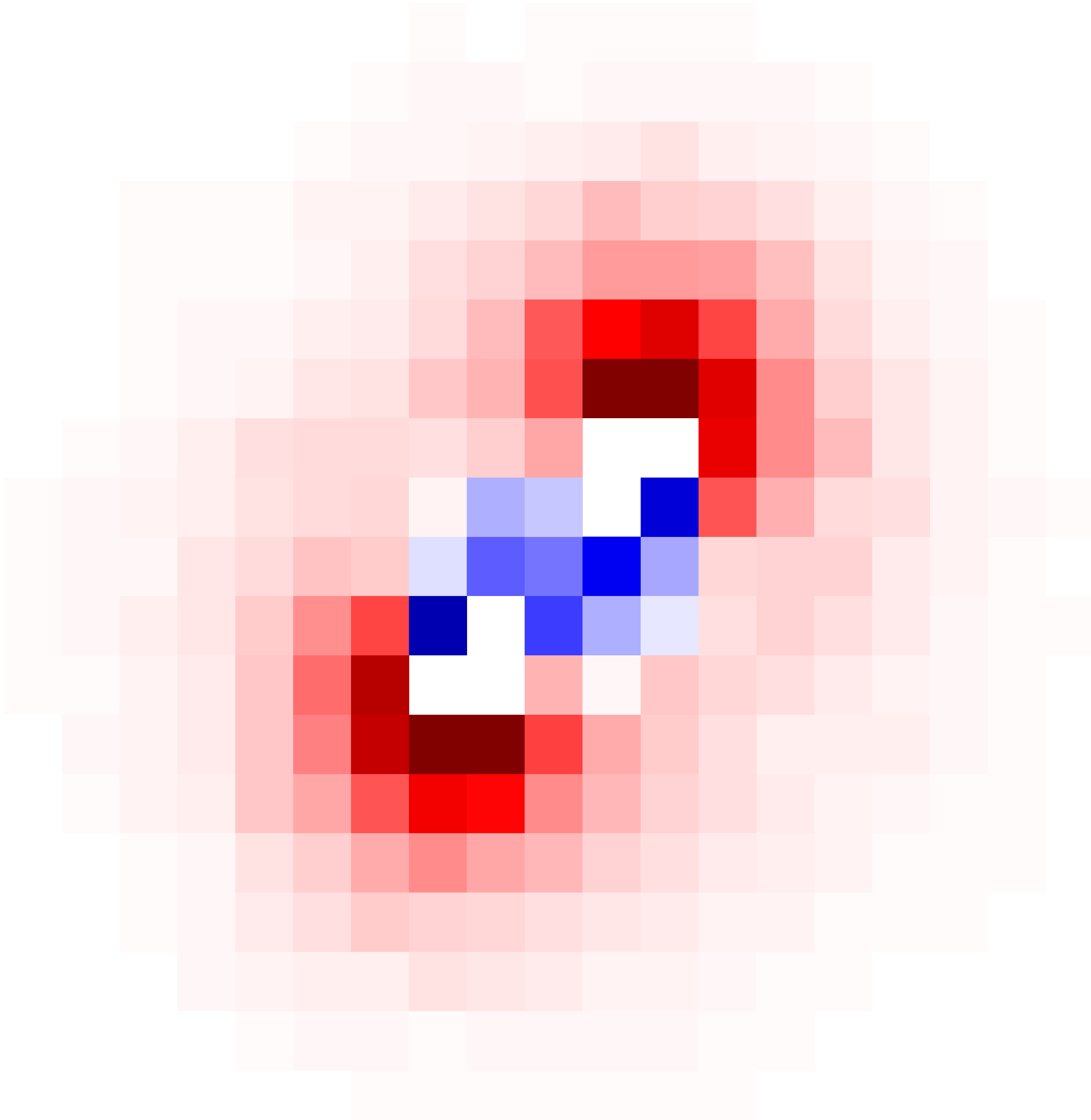
provide examples for training

arbitrarily many categories

**Generation** ← **Regression**

map noise to structure





Fin.