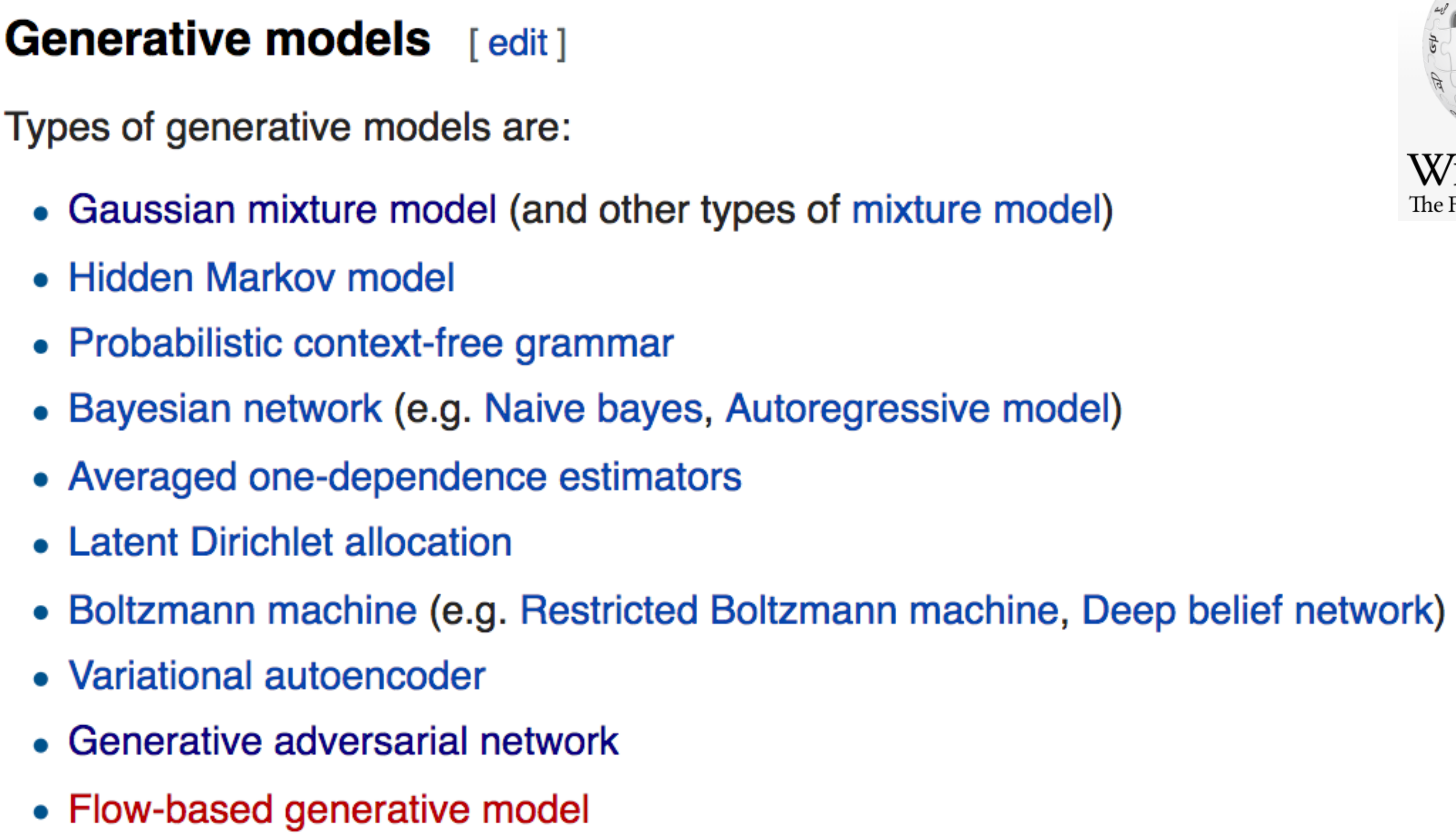
Generative Models for HEP

(Including a success story)

Fedor Ratnikov
NRU Higher School of Economics,
Yandex School of Data Analysis

Approaches



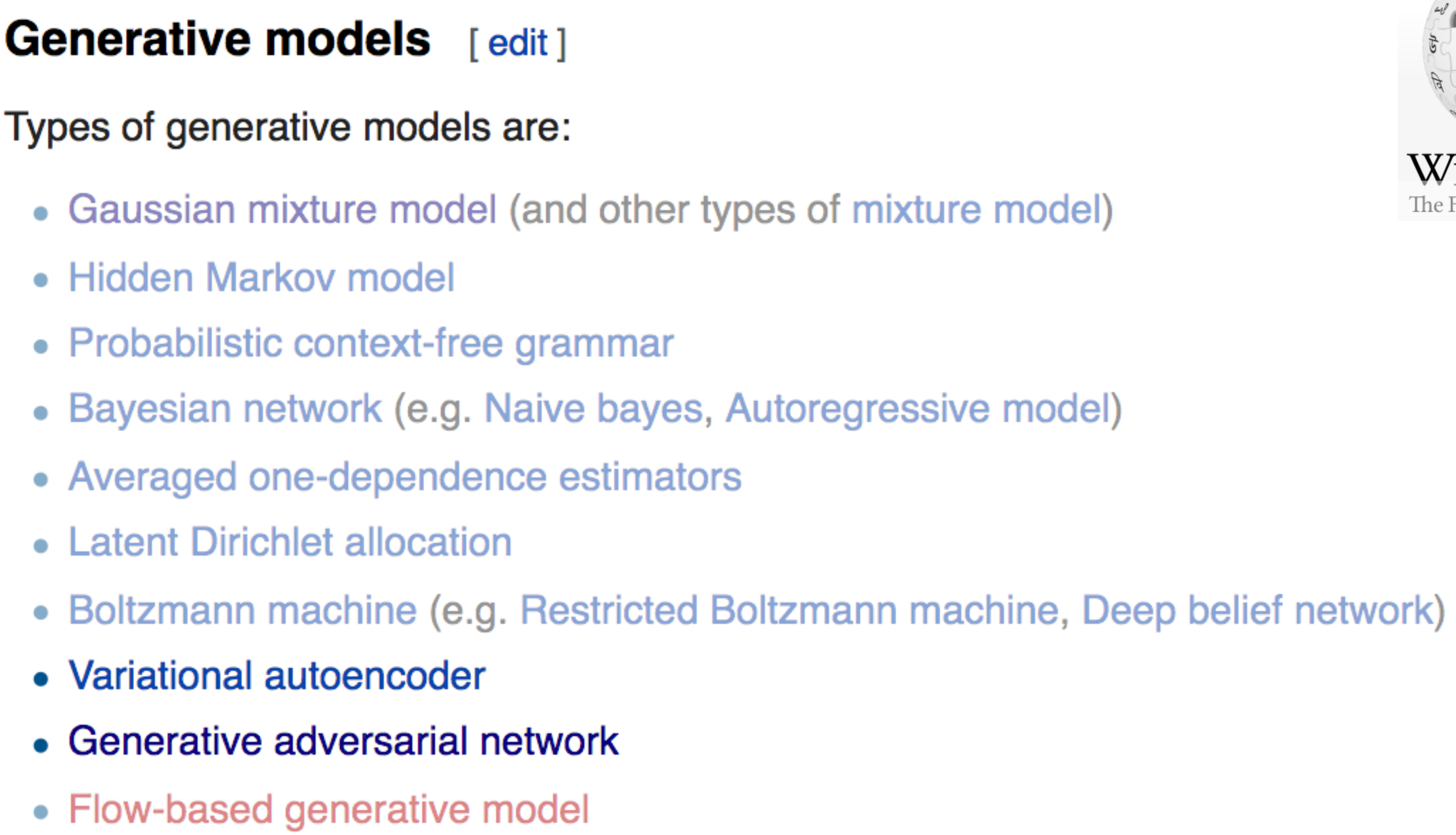
Generative models [edit]

Types of generative models are:

- Gaussian mixture model (and other types of mixture model)
- Hidden Markov model
- Probabilistic context-free grammar
- Bayesian network (e.g. Naive bayes, Autoregressive model)
- Averaged one-dependence estimators
- Latent Dirichlet allocation
- Boltzmann machine (e.g. Restricted Boltzmann machine, Deep belief network)
- Variational autoencoder
- Generative adversarial network
- Flow-based generative model

WIKIPEDIA
The Free Encyclopedia

Approaches



Generative models [edit]

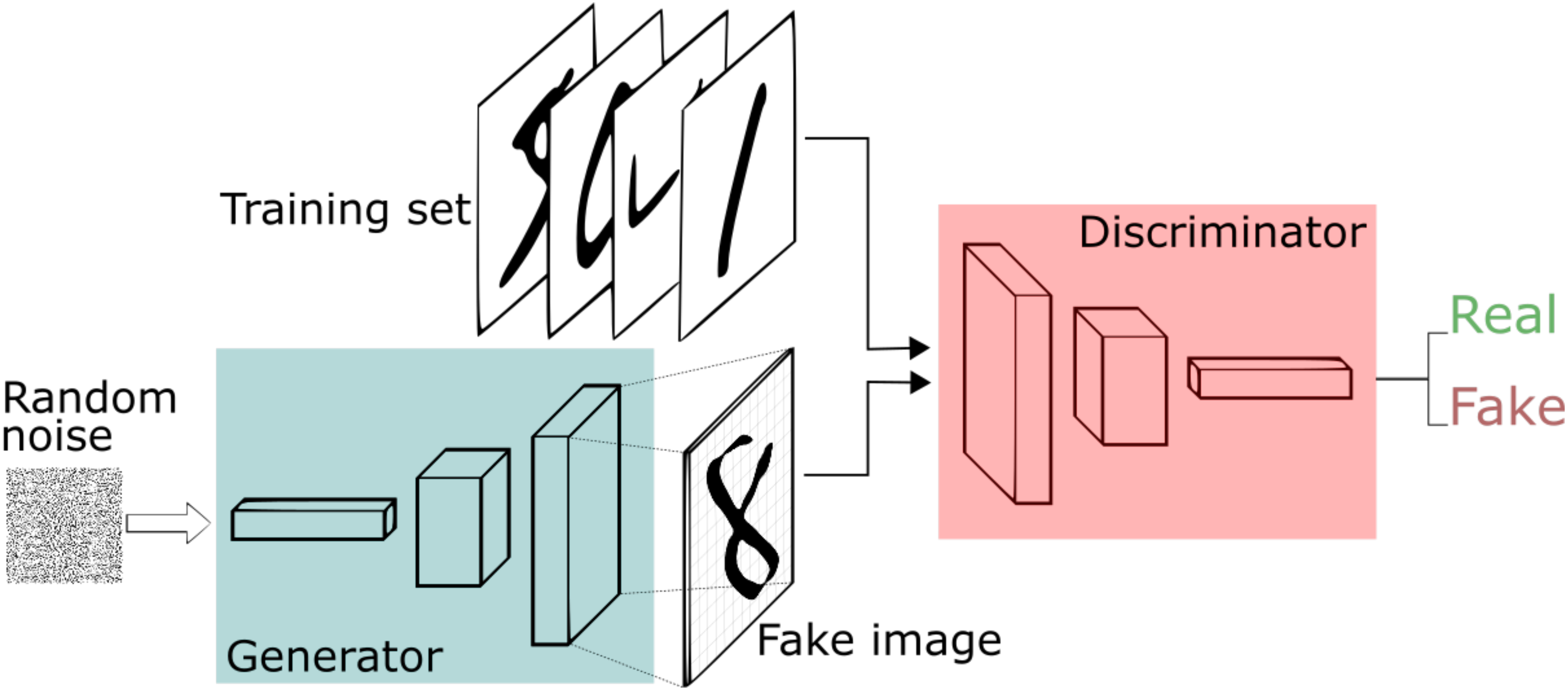
Types of generative models are:

- Gaussian mixture model (and other types of mixture model)
- Hidden Markov model
- Probabilistic context-free grammar
- Bayesian network (e.g. Naive bayes, Autoregressive model)
- Averaged one-dependence estimators
- Latent Dirichlet allocation
- Boltzmann machine (e.g. Restricted Boltzmann machine, Deep belief network)
- Variational autoencoder
- Generative adversarial network
- Flow-based generative model

- GAN and VAE are mostly used nowadays for generating complicated objects

Generative Adversarial Network (GAN)

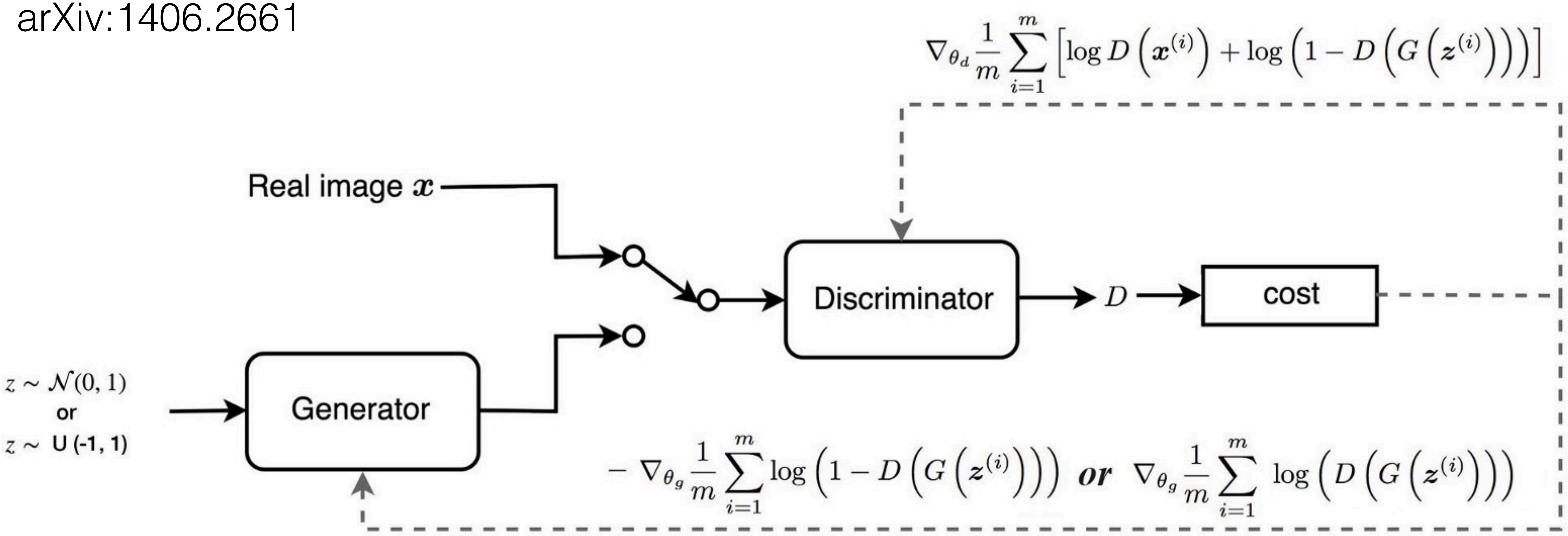
https://medium.freecodecamp.org/an-intuitive-introduction-to-generative-adversarial-networks-gans-7a2264a81394



- Implicit p(xly), sampling only

Classic GAN

arXiv:1406.2661



$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right]$$

$$- \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \quad \boldsymbol{or} \quad \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$$

- Discriminator approaches Jensen–Shannon divergence
    - vanishing gradients for poor generator
    - mode collapse

Illustration: Jonathan Hui

Wasserstein GAN

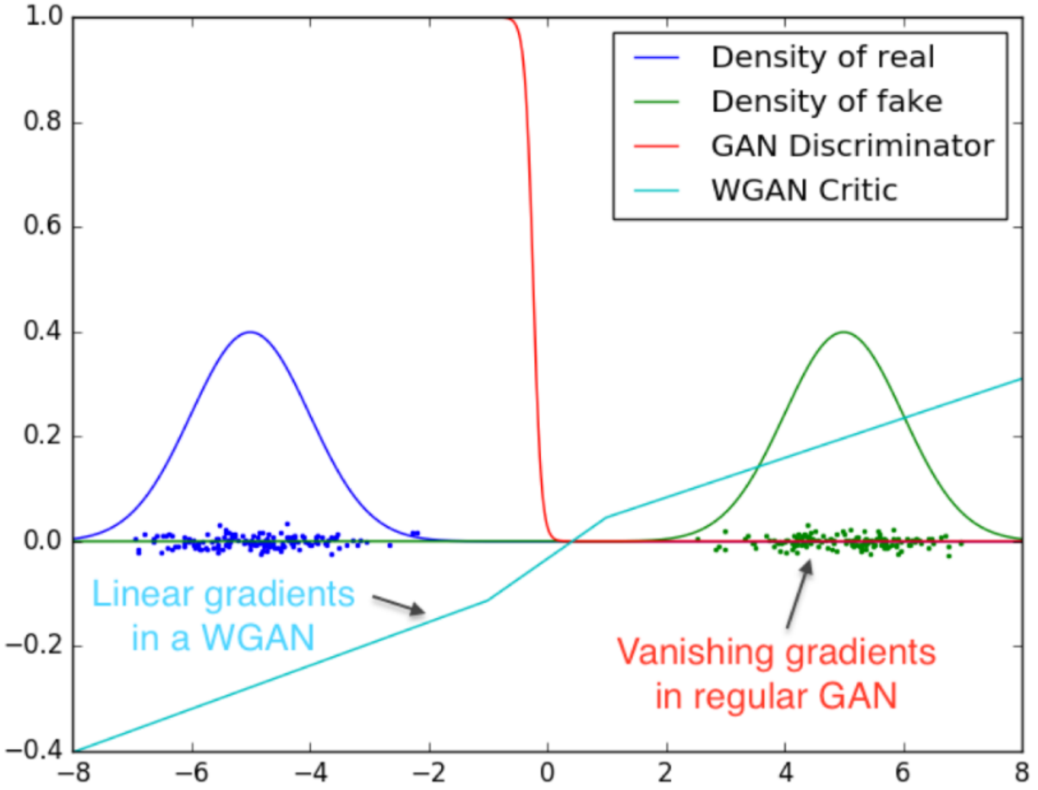
- Uses "earth mover's distance":

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\big[\, \|x - y\| \,\big]$$

  - Kantorovich-Rubinstein duality:

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)]$$

$$|f(x_1) - f(x_2)| \leq |x_1 - x_2|$$

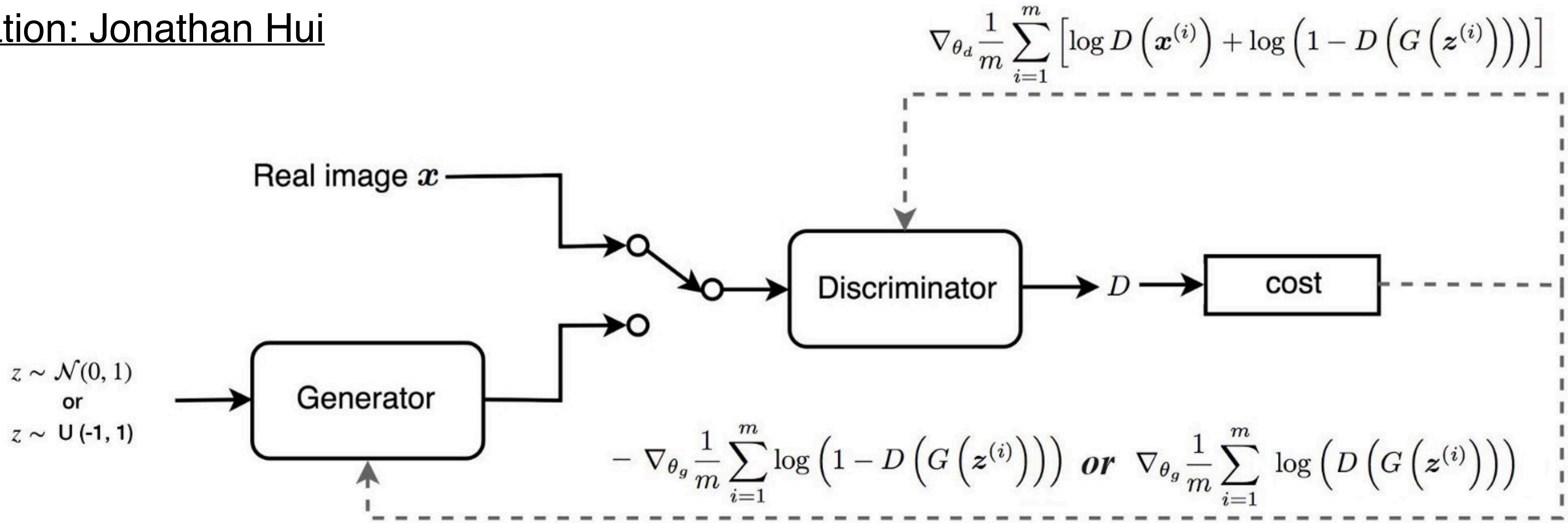


arXiv:1701.07875

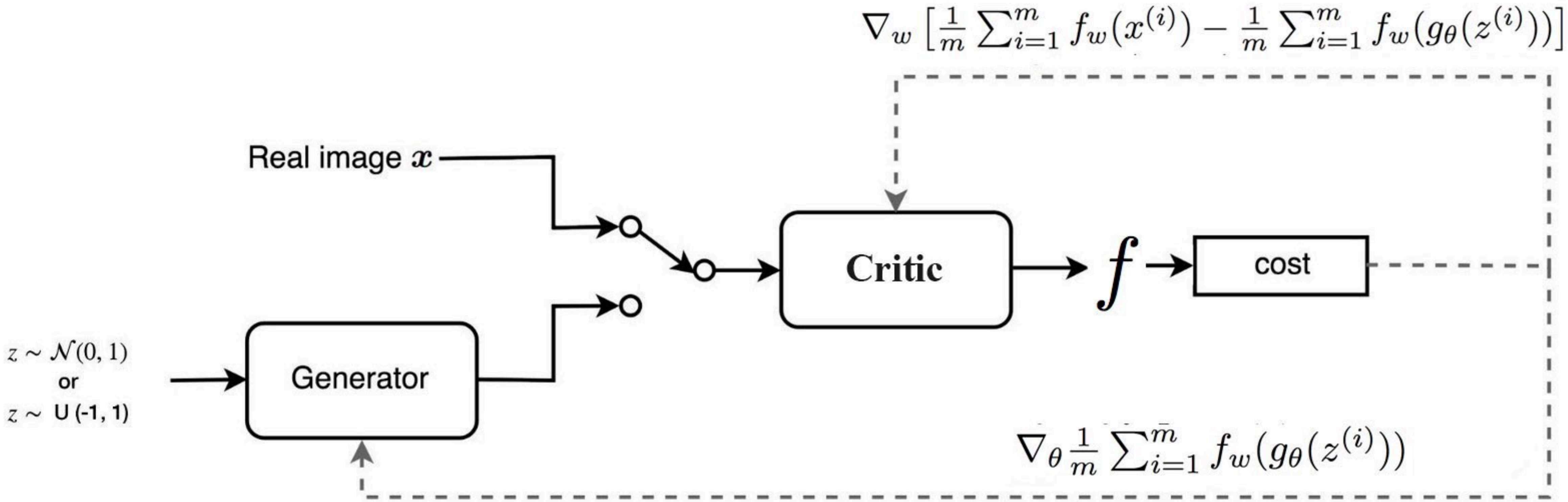
- "Discriminator" function *f(x)* may be approached using deep network
  - output is not probability, but any scalar number
  - need to satisfy 1-Lipschitz condition

WS GAN vs JS GAN

Illustration: Jonathan Hui



$\nabla_{\theta_d} \frac{1}{m}\sum_{i=1}^{m}\left[\log D\left(\boldsymbol{x}^{(i)}\right)+\log\left(1-D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right]$
Real image $\boldsymbol{x}$
Discriminator
D
cost
$z \sim \mathcal{N}(0, 1)$
or
$z \sim$ U (-1, 1)
Generator
$-\nabla_{\theta_g}\frac{1}{m}\sum_{i=1}^{m}\log\left(1-D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$ or $\nabla_{\theta_g}\frac{1}{m}\sum_{i=1}^{m}\log\left(D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$



$\nabla_w\left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$
Real image $\boldsymbol{x}$
Critic
f
cost
$z \sim \mathcal{N}(0, 1)$
or
$z \sim$ U (-1, 1)
Generator
$\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$

CramerGAN

- WGAN produces biased gradients, that makes converging slower, sometimes never reaching optimum

- CramerGAN uses energy distance as a critic (discriminator):

$$\mathcal{E}(X,Y) := 2\,\mathbb{E}\,\|X-Y\|_2 - \mathbb{E}\,\|X-X'\|_2 - \mathbb{E}\,\|Y-Y'\|_2$$

arXiv:1705.10743

  - where X, X', Y, Y' are statistically independent samples from two distributions

  - corresponds to the Cramer distance in 1D case:

$$l_2^2(P,Q) := \int_{-\infty}^{\infty} (F_P(x) - F_Q(x))^2 \mathrm{d}x$$

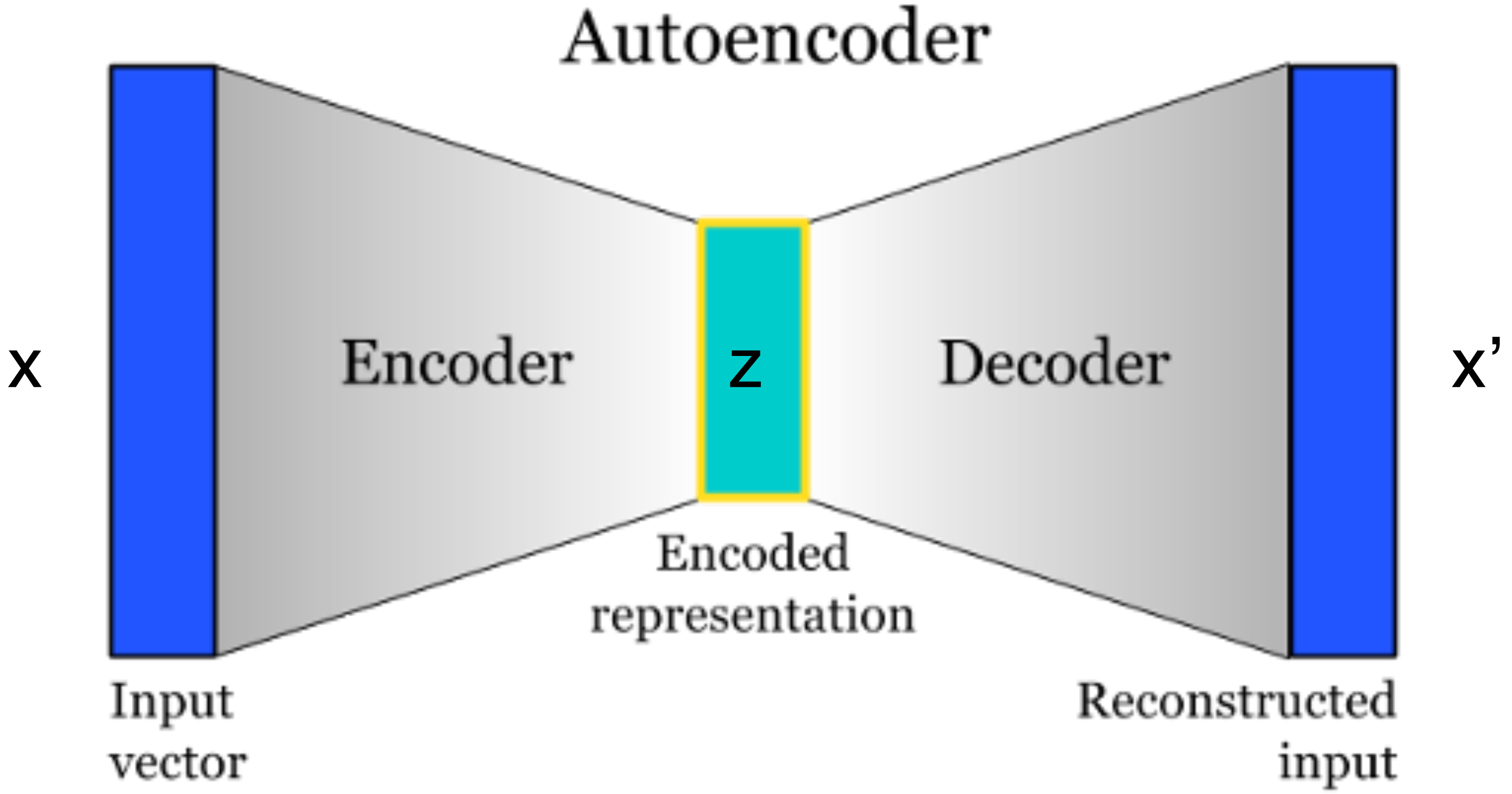
  - generator loss is therefore more complicated:

$$L_g = 2||h(x_r) - h(x_g)||_2 - ||h(x_r) - h(x'_r)||_2 - ||h(x_g) - h(x'_g)||_2$$

  - critic is trained to maximize the energy distance

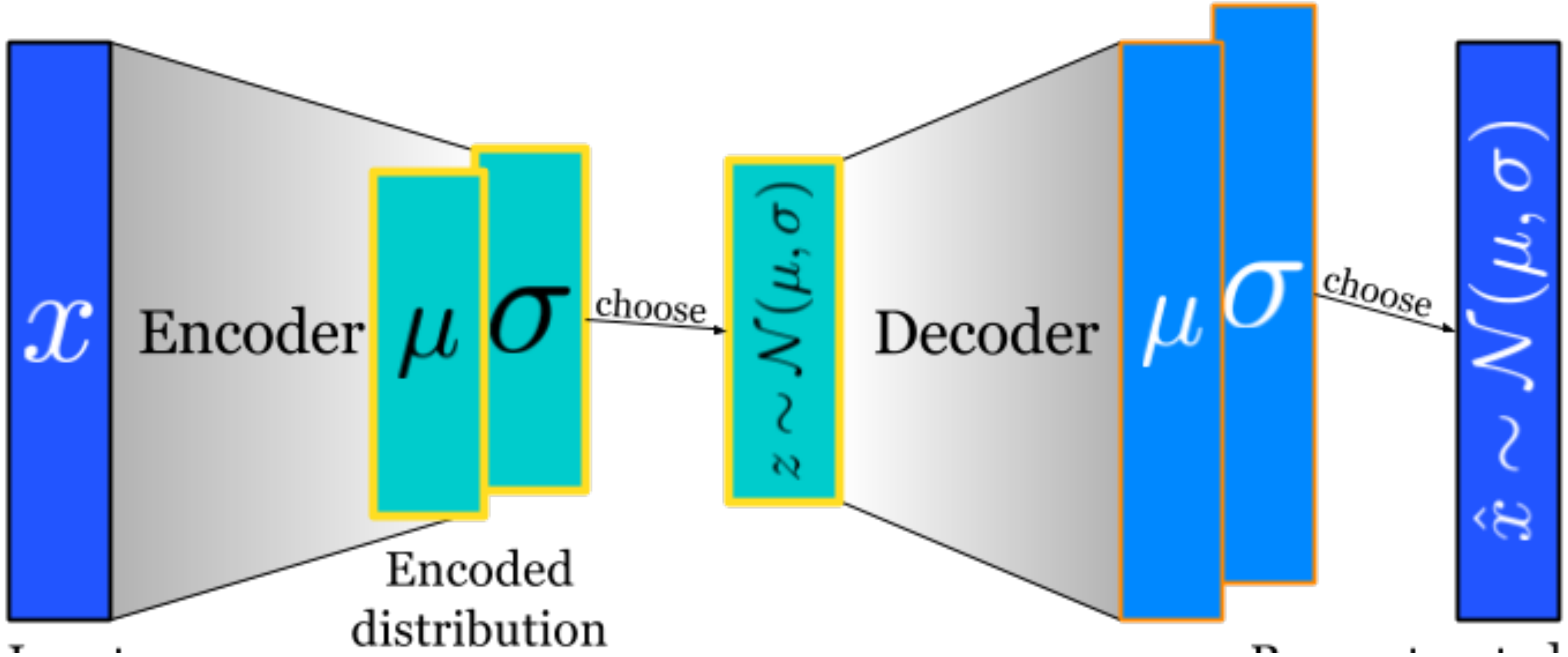
- CramerGAN demonstrates better convergency indeed

Variational Autoencoder



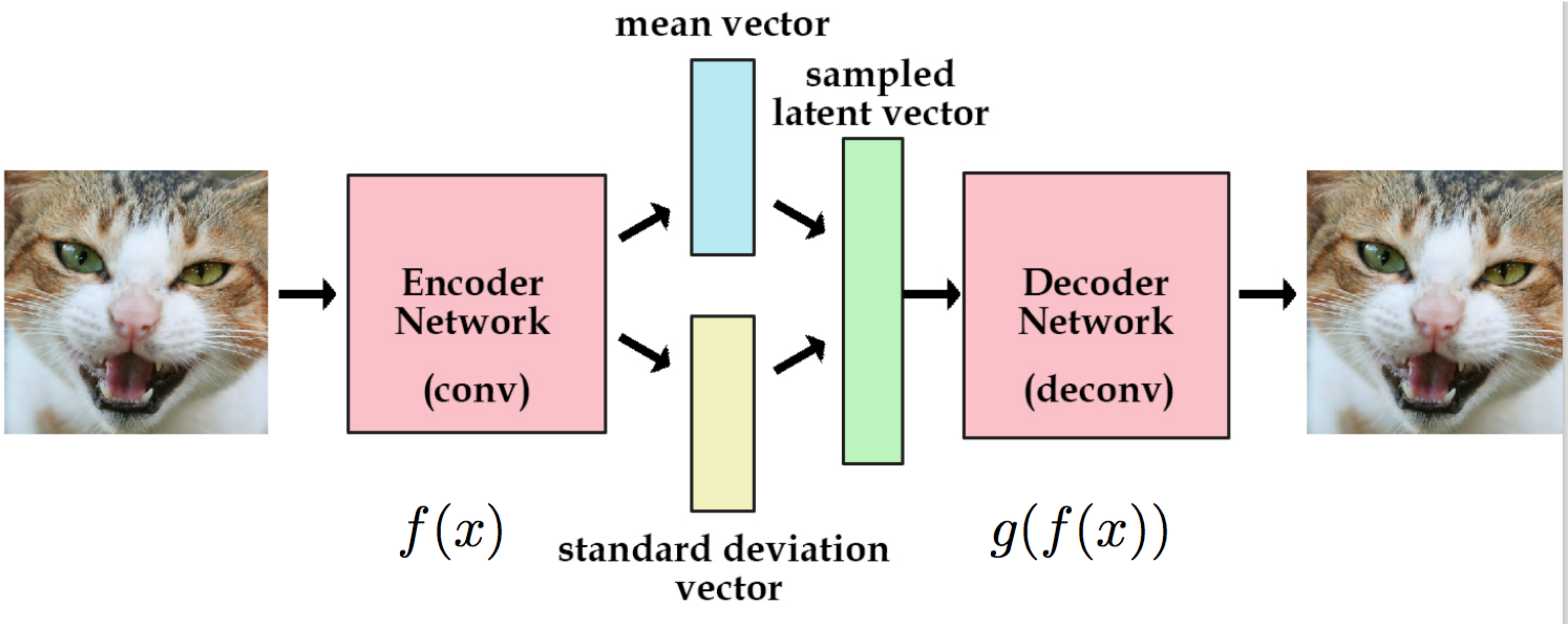
- Autoencoder can be trained to sample realistic objects
- *x* → encoder → *z* → decoder → *x’*
  - require *x’* ~ *x*
- Decoder part of the AE can generate realistic objects
  - … providing correct prior distribution in the latent space *p(z)*

Variational Autoencoder



- Decoder part of the AE can generate realistic objects
  - … providing correct prior distribution in the latent space *p(z)*
- *P*ut extra requirement into the loss
  - latent distribution *p(z|X)* must approach some standard one, e.g. $\mathcal{N}(\mathbf{0},\mathbf{I})$
  - make *z(x)* variational
  - (make *x'(z)* variational)

Variational Autoencoder



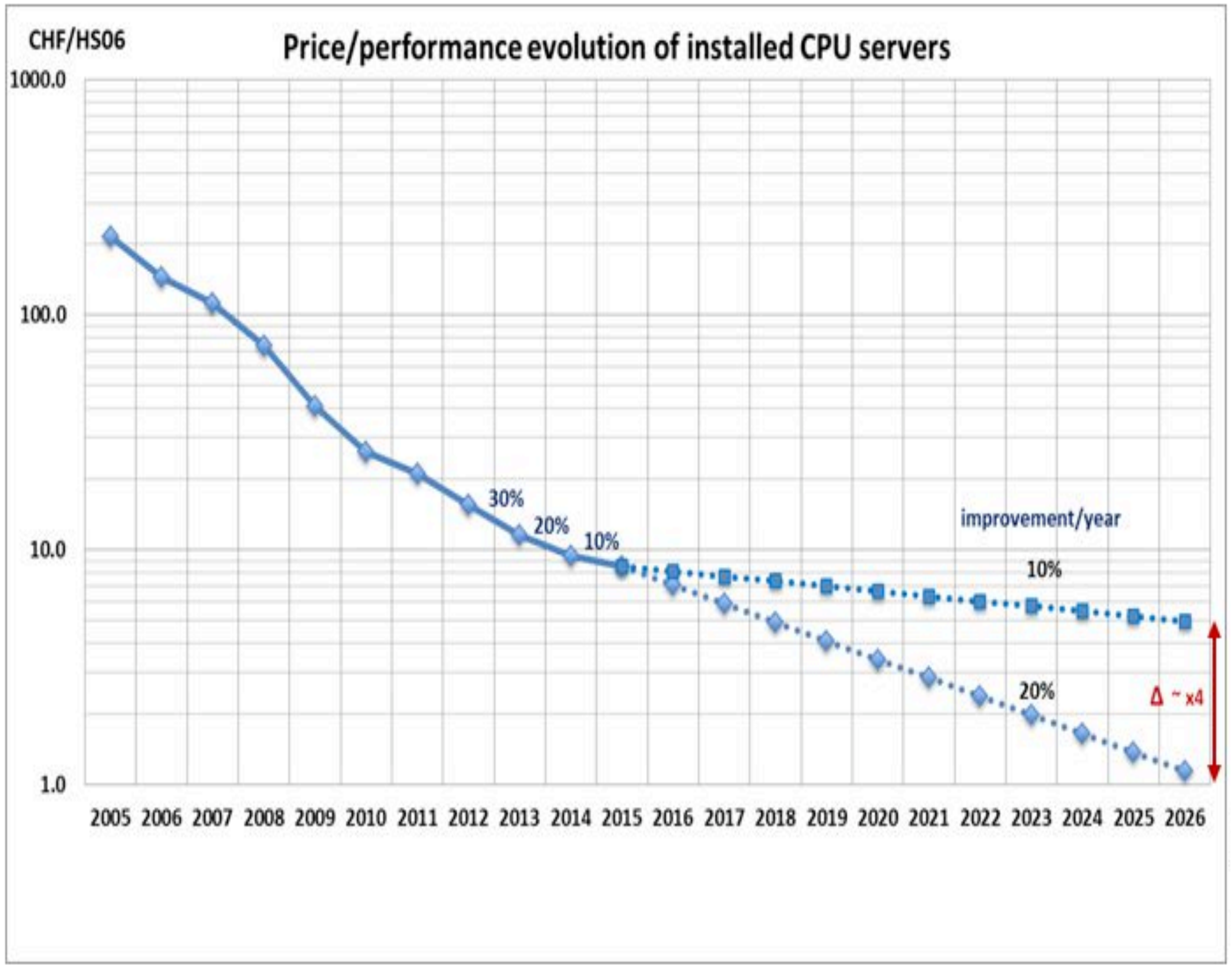
- VAE allows calculating p(xly)
  - NB: GAN only allows sampling from p(xly)
- … but smaller number of dimensions in the latent space
  - blurry objects

Library Approach

- We have train sample for the generative model anyway
  - consistency with this train sample is a figure of merit for the generative model
- Objects of the train sample may be used for generation directly
  - similar to KNN classification algorithm
  - k=1: search for the object with appropriate conditions in the (presumably huge) data library
  - k>1: need to interpolate between objects
    - short distance objects interpolation, more robust than global generation
- NB: library approach by construction uses full information which is contained in the training sample

Generative Models at LHC

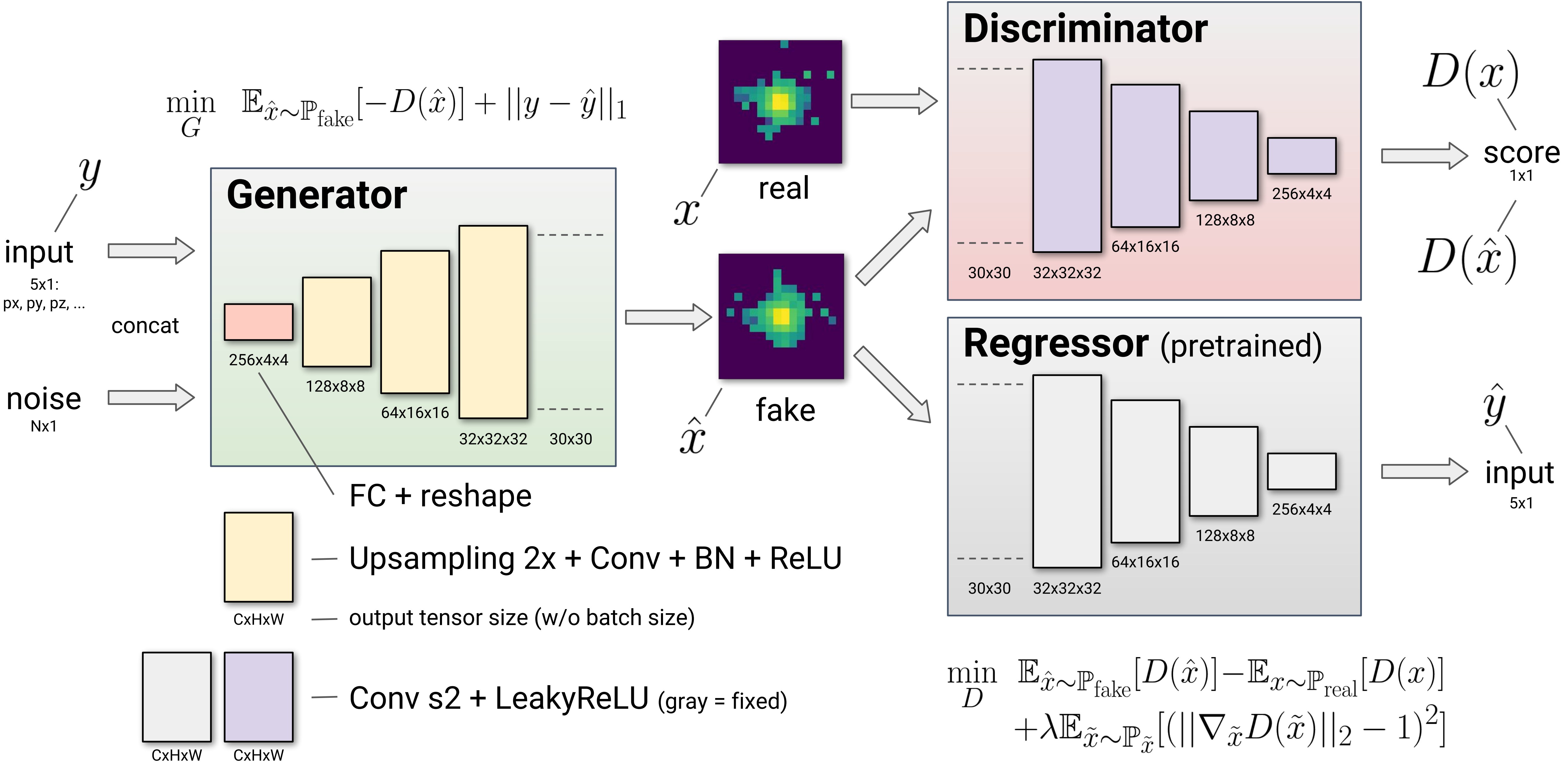
- About 80% of computing resources are used for MC simulation in HEP experiments
  - Calorimeter simulation is one of bottlenecks
  - RICH is the next in the row for LHCb detector
    - > 85% of simulation is taken by these



- Can not expect exponential rise of CPU performance
- Need a work around for Run3 and HL-LHC
- Generative models trained on the detailed GEANT simulation may be a solution

Example: ECAL Conditional Fast Simulation

Training scheme



$$\min_G \; \mathbb{E}_{\hat{x}\sim\mathbb{P}_{\text{fake}}}[-D(\hat{x})] + ||y - \hat{y}||_1$$

FC + reshape

Upsampling 2x + Conv + BN + ReLU

output tensor size (w/o batch size)

Conv s2 + LeakyReLU (gray = fixed)

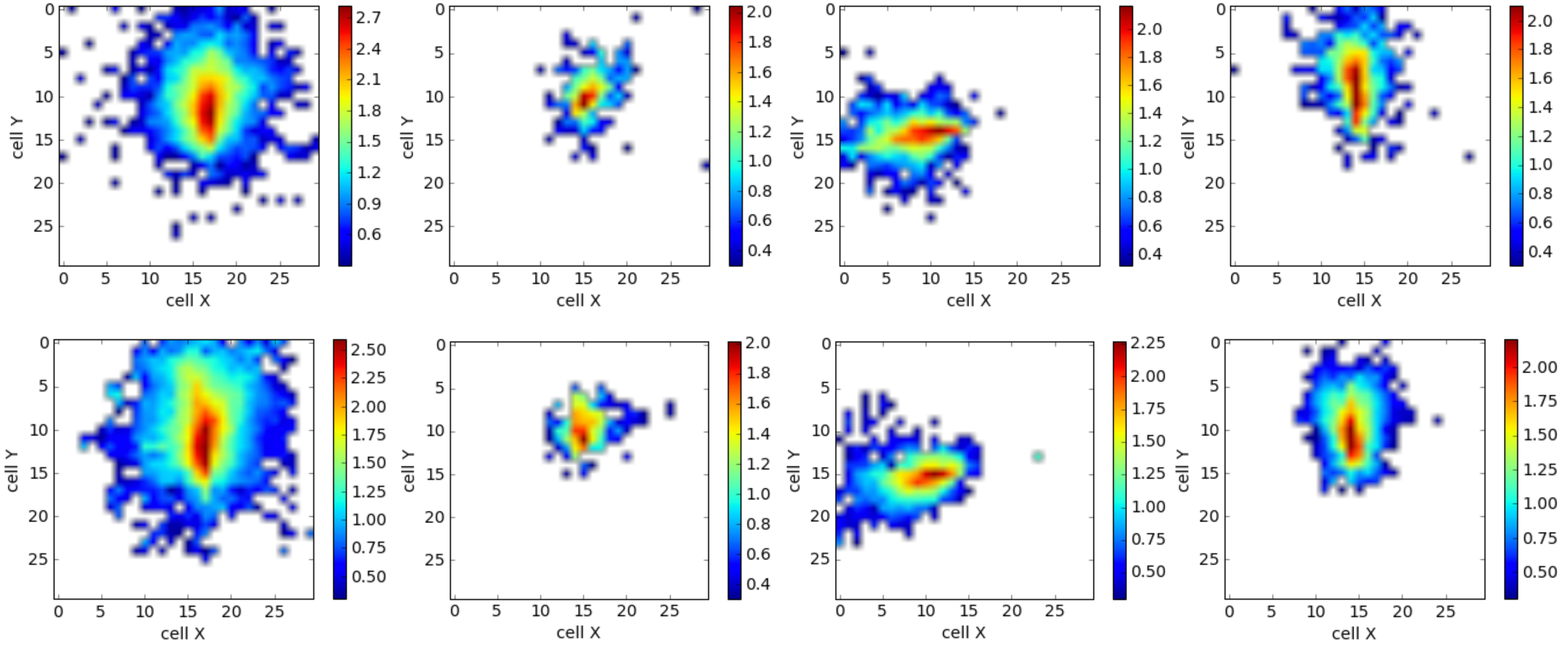
$$\min_D \; \mathbb{E}_{\hat{x}\sim\mathbb{P}_{\text{fake}}}[D(\hat{x})] - \mathbb{E}_{x\sim\mathbb{P}_{\text{real}}}[D(x)] + \lambda\mathbb{E}_{\tilde{x}\sim\mathbb{P}_{\tilde{x}}}[(||\nabla_{\tilde{x}}D(\tilde{x})||_2 - 1)^2]$$

LHCb ECAL Simulation

GEANT Simulated

log$_{10}$(cell energy)

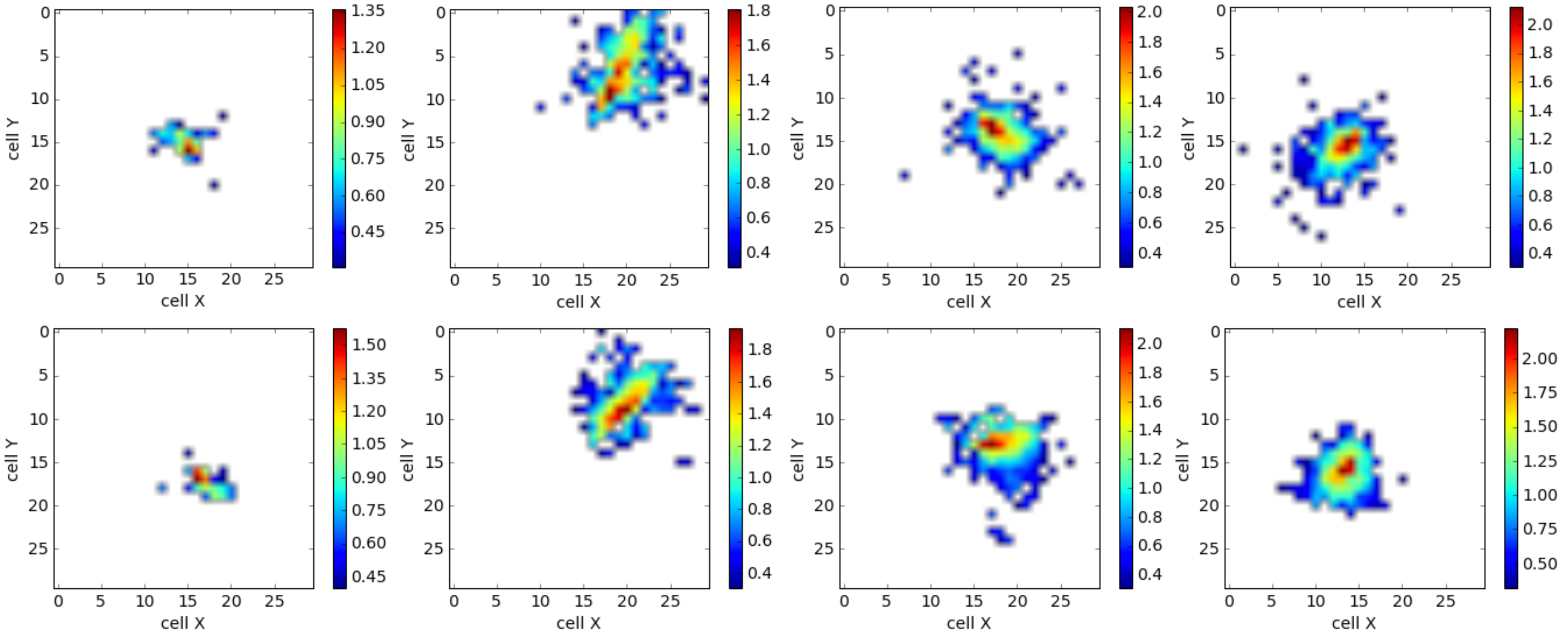
GAN Generated



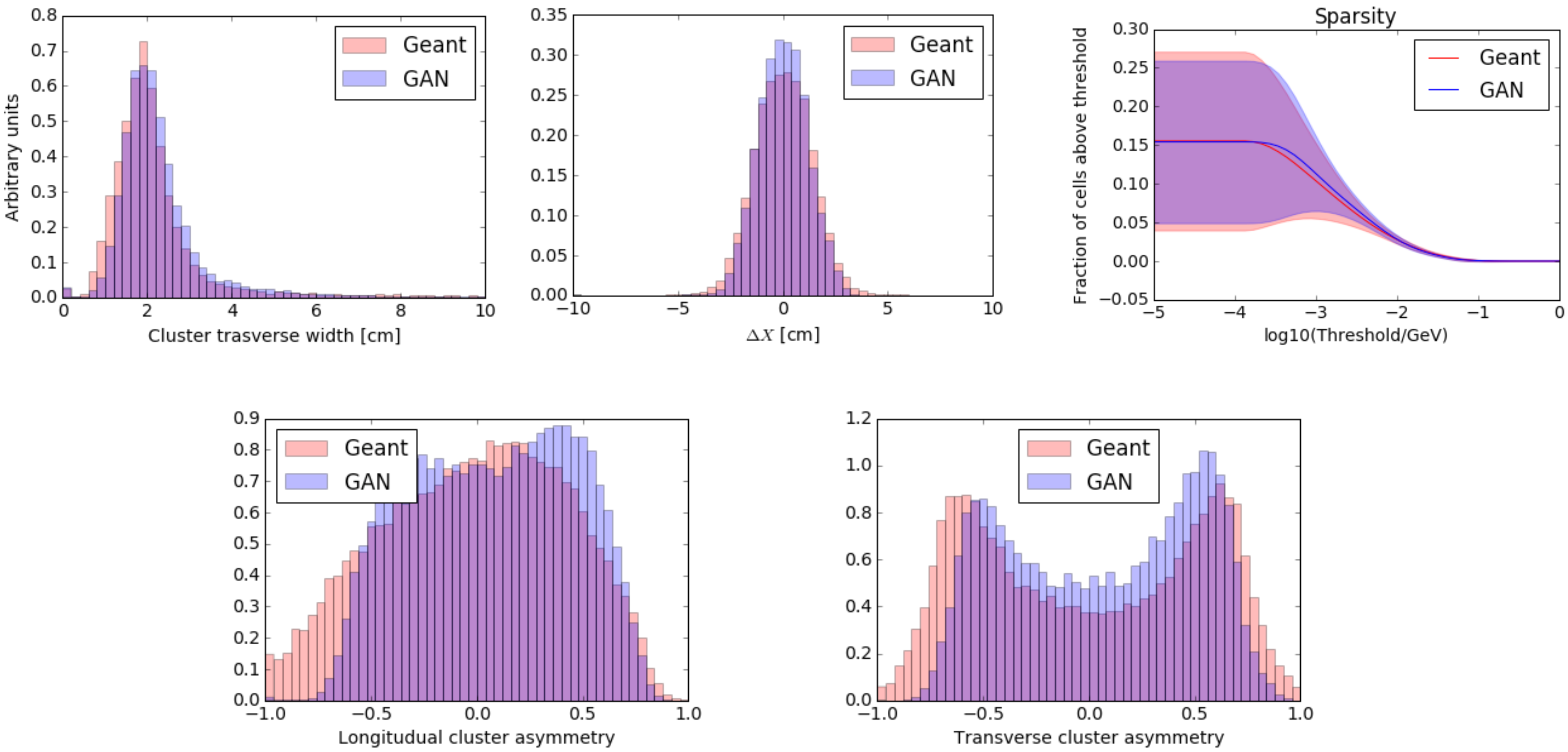
GEANT Simulated

log$_{10}$(cell energy)

GAN Generated



Primary and Marginal Distributions



- Is hard to fit marginal distributions
  - unless the model is aware that those are important for us

Scientific Requirements

- For image generation we are usually happy if the result **looks** like it is desired
- In science we need the result to reasonably well match the given set of requirements. Requirements are driven by scientific considerations closely connected to the ultimate scientific goal

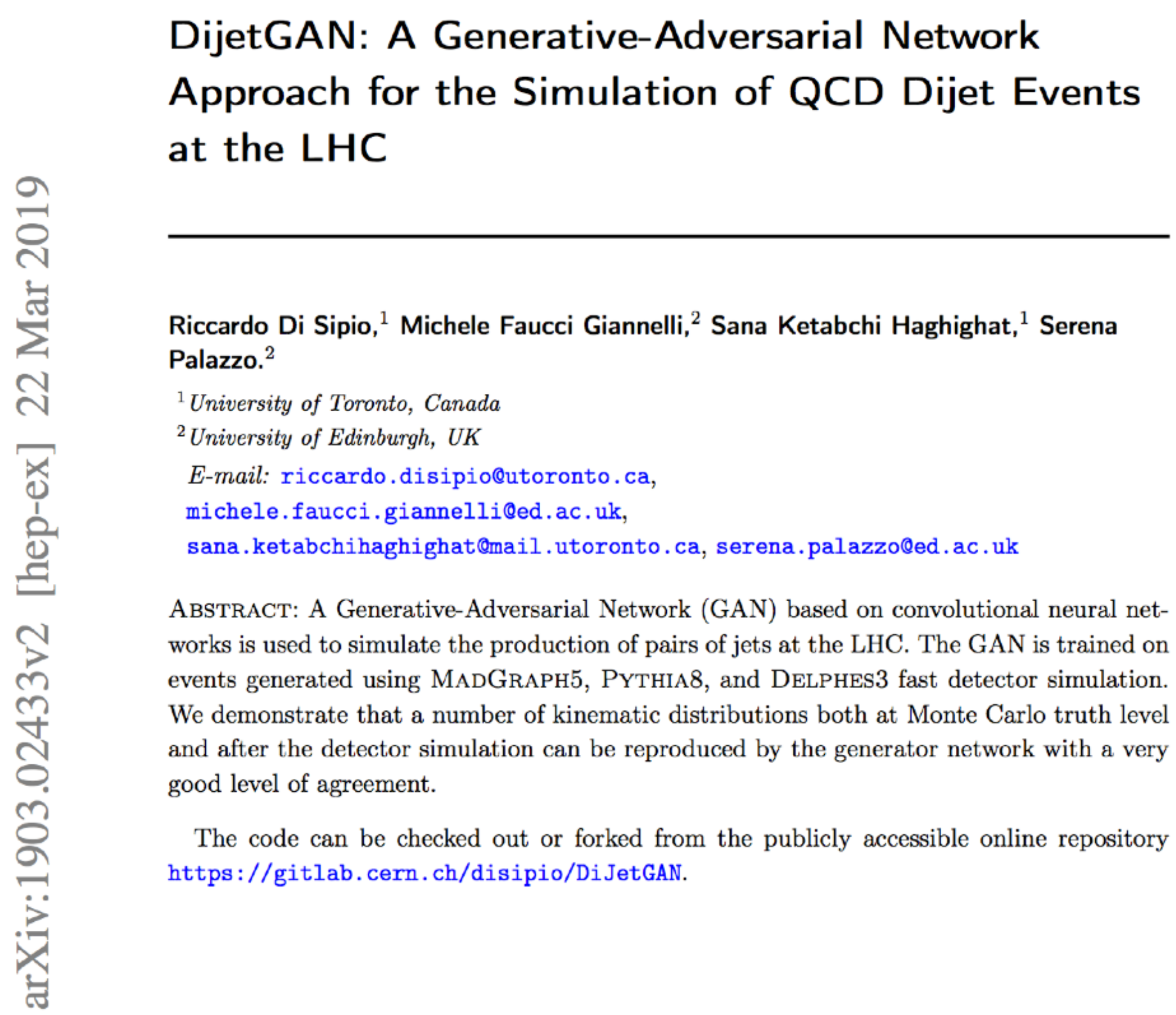
Enforcing Important Statistics

- No generative model is ideal
  - some deviations from the original distribution remain
- Models tend to learn primary statistics of generated objects
- In physics applications, we often need for our model to learn particular statistics which are marginal for the generated object
  - e.g. cluster shape fluctuations for fast calorimeter simulation
- Can enforce these statistics by explicit adding them to the loss
  - can't we?

Enforcing Important Statistics

- Can enforce statistics by explicit adding them to the loss
    - can’t we?
- By adding necessary statistics to the loss we do enforce match for these statistics
    - most likely by the price of overtraining these particular statistics
        - … and we lose handle to validate quality of generator on this statistics
- Still can remove those statistics from loss, and see how far they would deviate
    - this could be a figure of merit for generating this statistics

Generating Tails



DijetGAN: A Generative-Adversarial Network Approach for the Simulation of QCD Dijet Events at the LHC

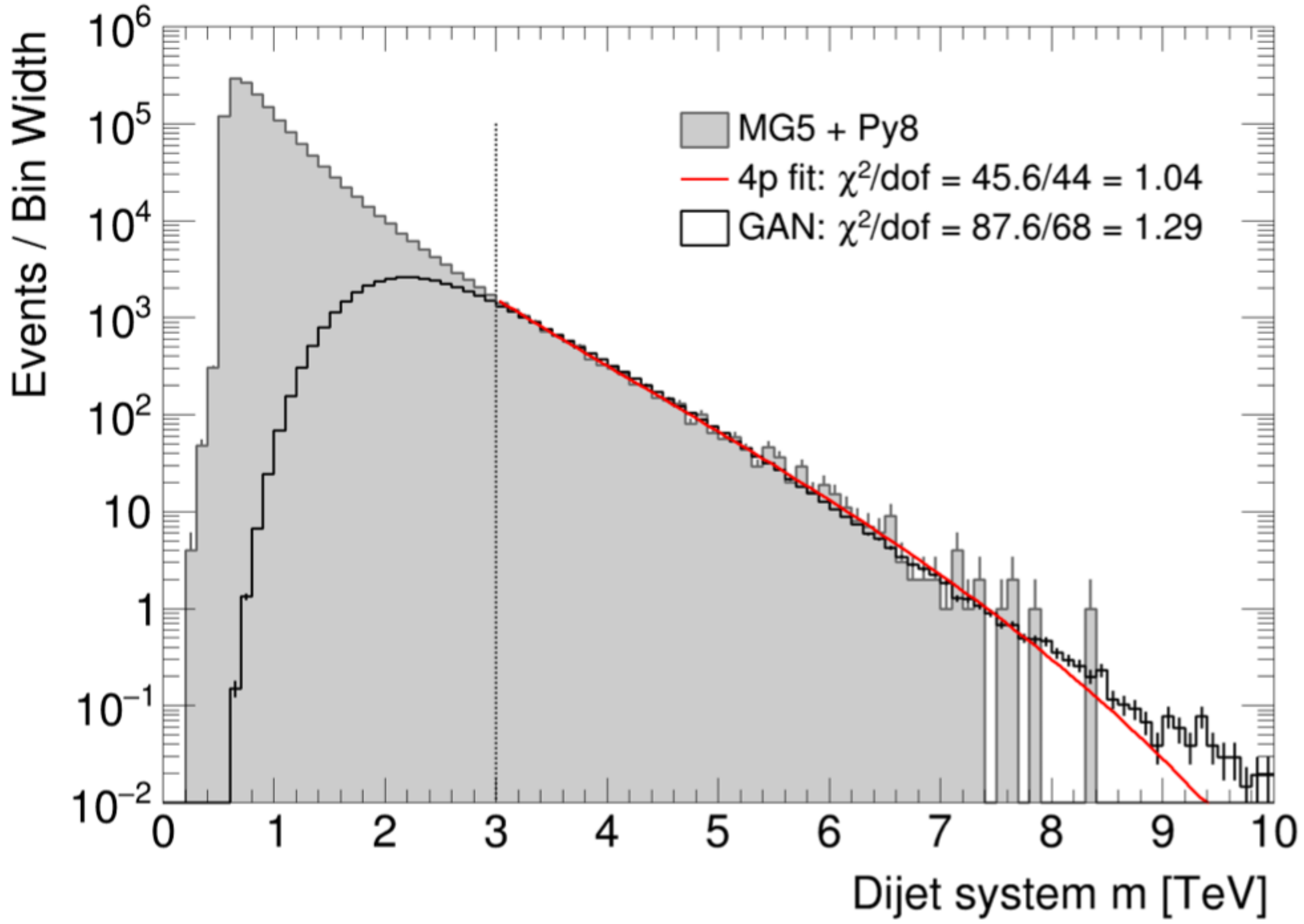
Riccardo Di Sipio,[1] Michele Faucci Giannelli,[2] Sana Ketabchi Haghighat,[1] Serena Palazzo.[2]

[1] *University of Toronto, Canada*
[2] *University of Edinburgh, UK*

E-mail: riccardo.disipio@utoronto.ca, michele.faucci.giannelli@ed.ac.uk, sana.ketabchihaghighat@mail.utoronto.ca, serena.palazzo@ed.ac.uk

Abstract: A Generative-Adversarial Network (GAN) based on convolutional neural networks is used to simulate the production of pairs of jets at the LHC. The GAN is trained on events generated using MadGraph5, Pythia8, and Delphes3 fast detector simulation. We demonstrate that a number of kinematic distributions both at Monte Carlo truth level and after the detector simulation can be reproduced by the generator network with a very good level of agreement.

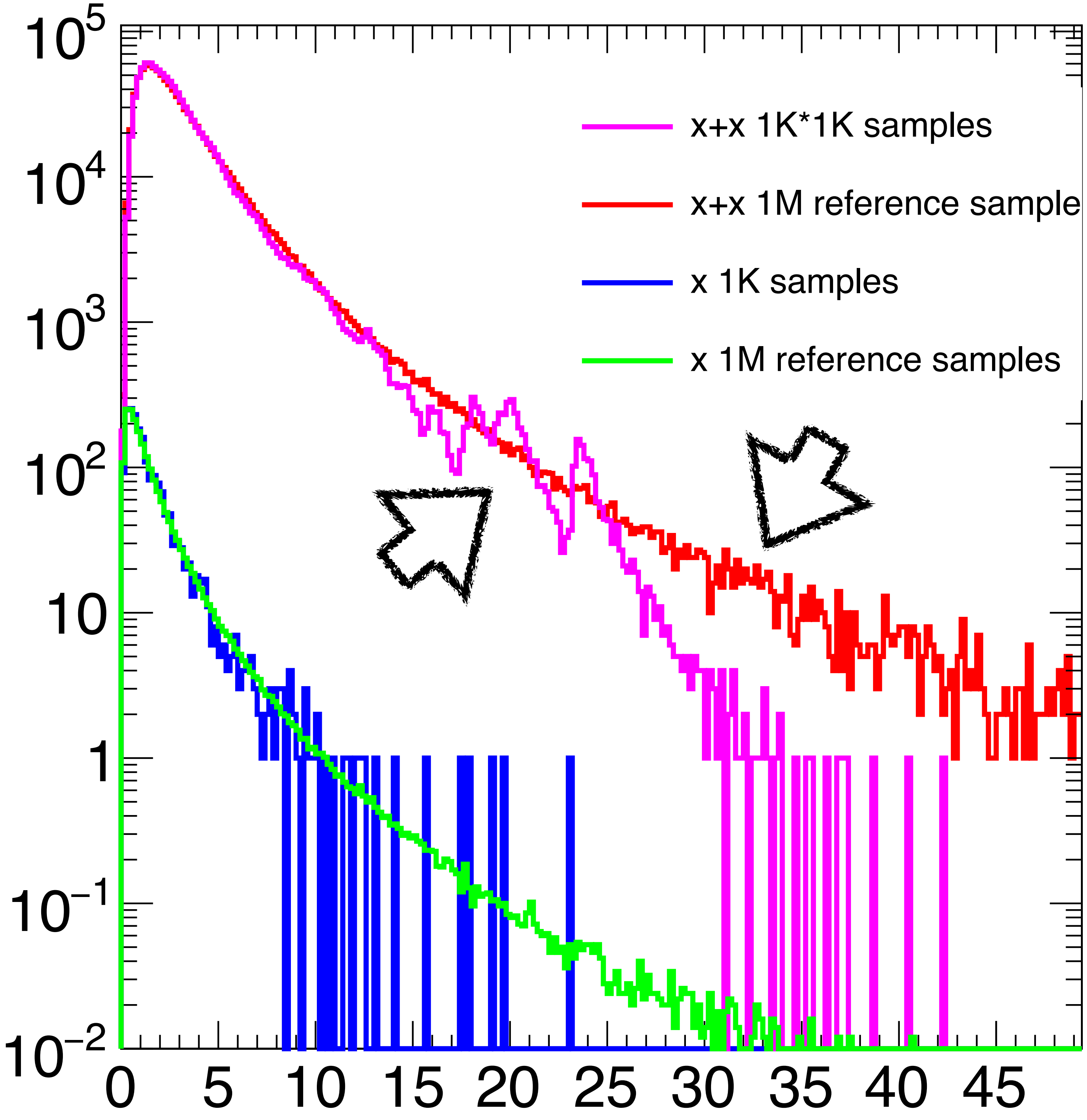
The code can be checked out or forked from the publicly accessible online repository https://gitlab.cern.ch/disipio/DiJetGAN.



- If the model is trained on the limited sample, how reliable are predictions beyond the training domain?

Asymptotic Properties

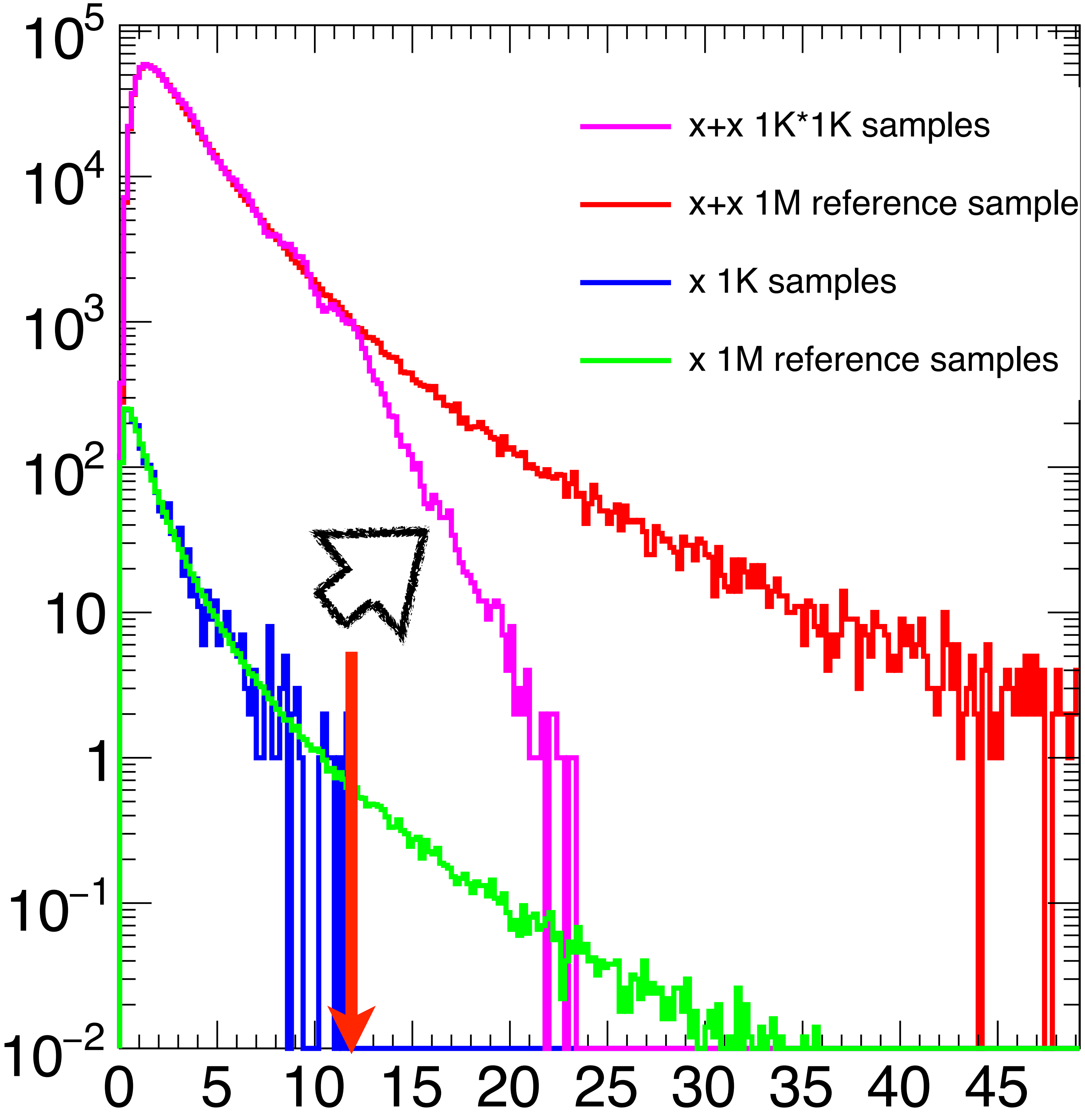
- Toy model
  - two variables distributed LogNormal
  - training sample 1K events
  - target sample 1M events x_1+x_2
  - use 1K samples with permutations



- Systematics due to fluctuation in tails

Asymptotic Properties

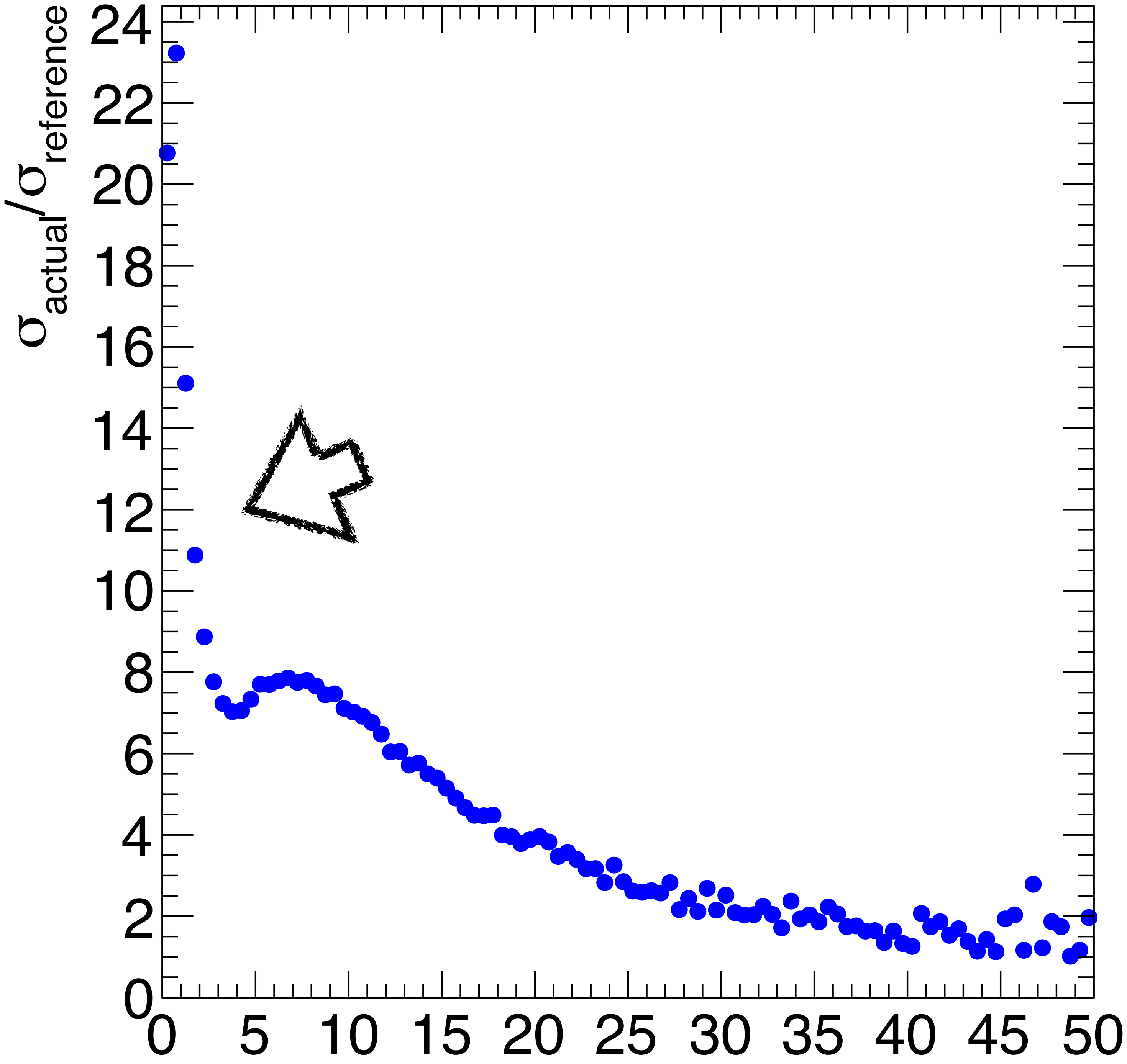
- Toy model
  - two variables distributed LogNormal
  - training sample 1K events (x<12)
  - target sample 1M events x_1+x_2
  - use 1K samples with permutations



- Systematics due to the marginal cut off

Statistics Properties

- Toy model
  - two variables distributed LogNormal
  - training sample 1K events
  - fit LogNormal to 1K samples
  - variation for x_1+x_2
- Systematics from the model systematics
  - driven by the train sample statistics

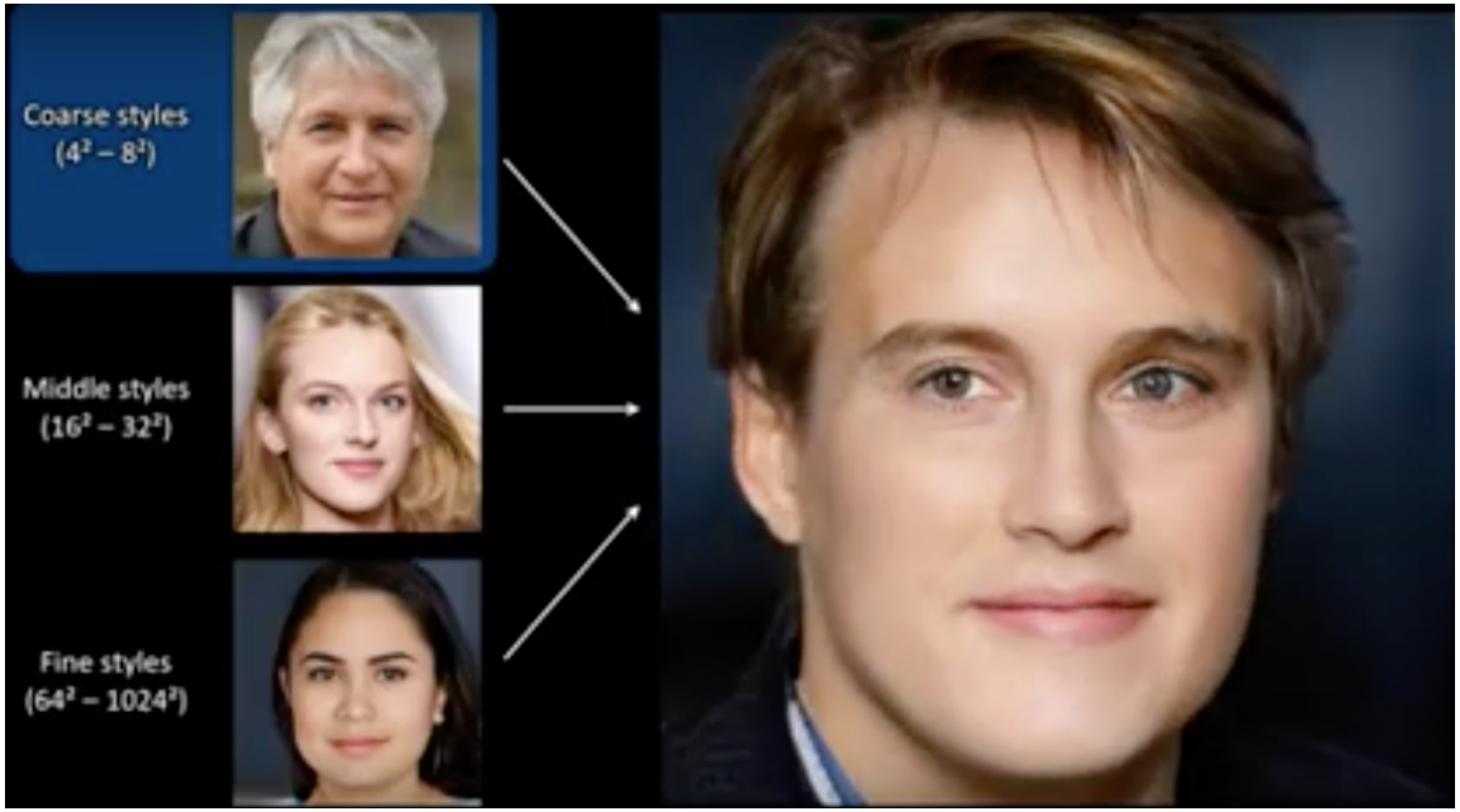


Decomposition

- Quality of the generative models is limited by the size of the train data sample
  - generative models may not give profit for producing statistically correct big data sets
    - no information beyond the train sample is available
    - model systematics corresponds to the train sample statistics

Decomposition

- No information beyond the train sample is available



- Not quite if we can decompose generative model into separate components
  - random combinations of different components may drastically increase variability

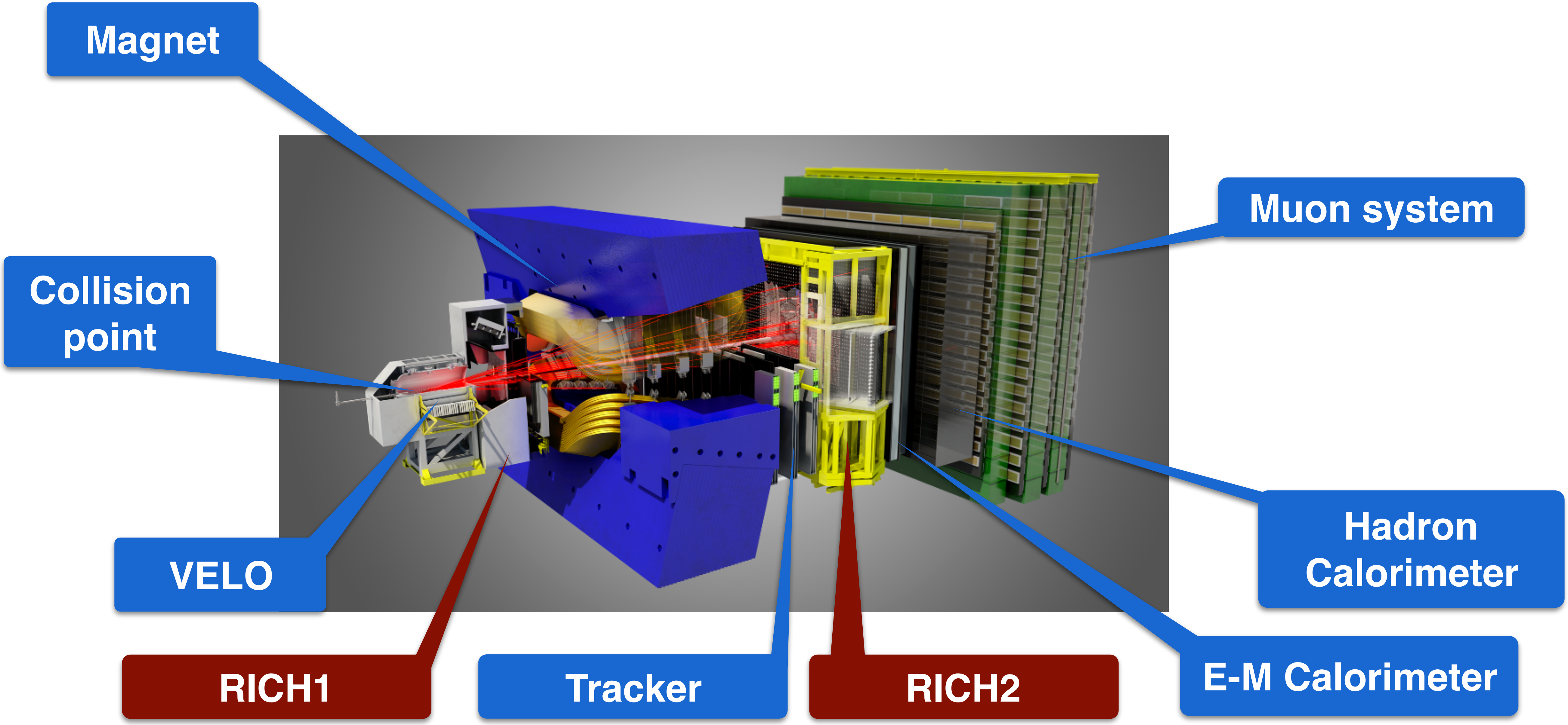
Decomposition

- Quality of the generative models is limited by the size of the train data sample
  - generative models may not give profit for producing statistically correct big data sets
    - no information beyond the train sample is available
- Not quite if we can decompose generative model into separate components
  - random combinations of different components may drastically increase variativity
- E.g. fast simulation of the calorimeter response
  - generator is trained on 10^6 incident particles
  - ~50 particles in the calorimeter per event
  - total variability ~$(10^6)^{50} = 10^{300}$! (NB intrinsic correlation)

Quality Metric

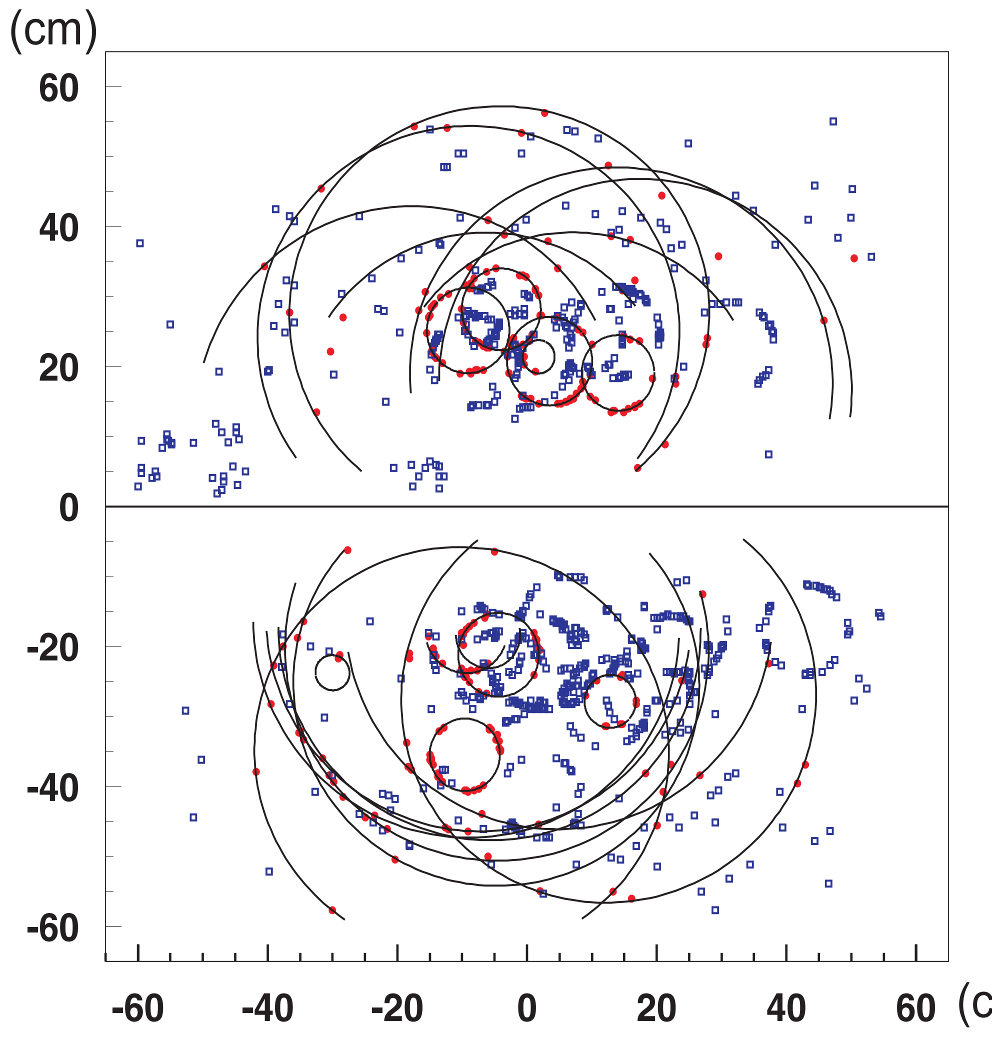
- No generative model is ideal
  - some deviations from the original distribution remain
- Minor deviations are not that important e.g. for image generation
- Minor deviations may be a big deal for generative models in physics
  - e.g. we could want E^2-p^2=m^2 for generated particles to be precise
- Ultimate generative model quality metric is a comparing the final physics result obtained using generative model with the one obtained using the test data
  - accuracy is limited by the size of the test data

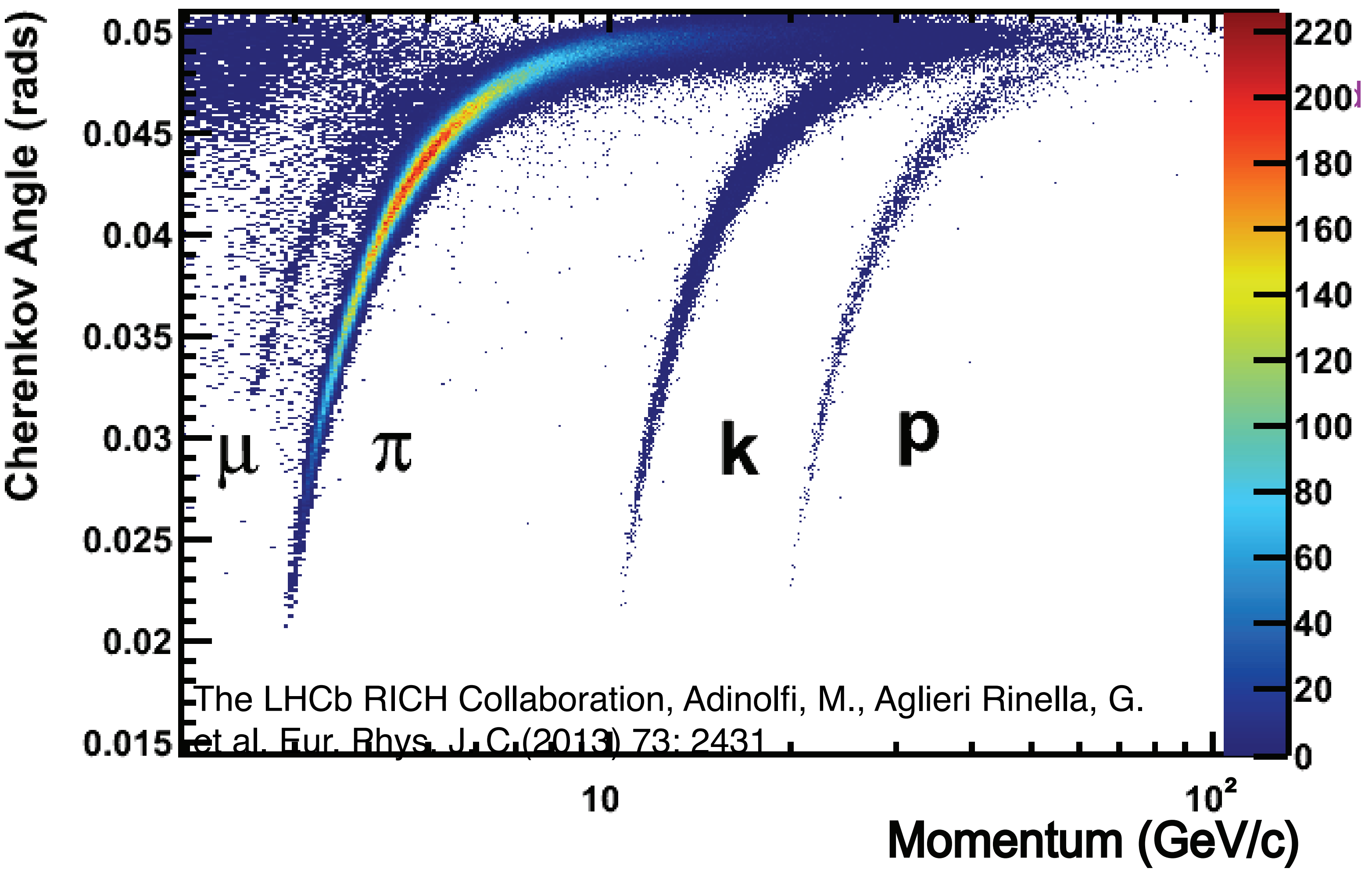
Practical Example: RICH-based particle ID



- Ring Image CHerenkov detector

RICH Basics



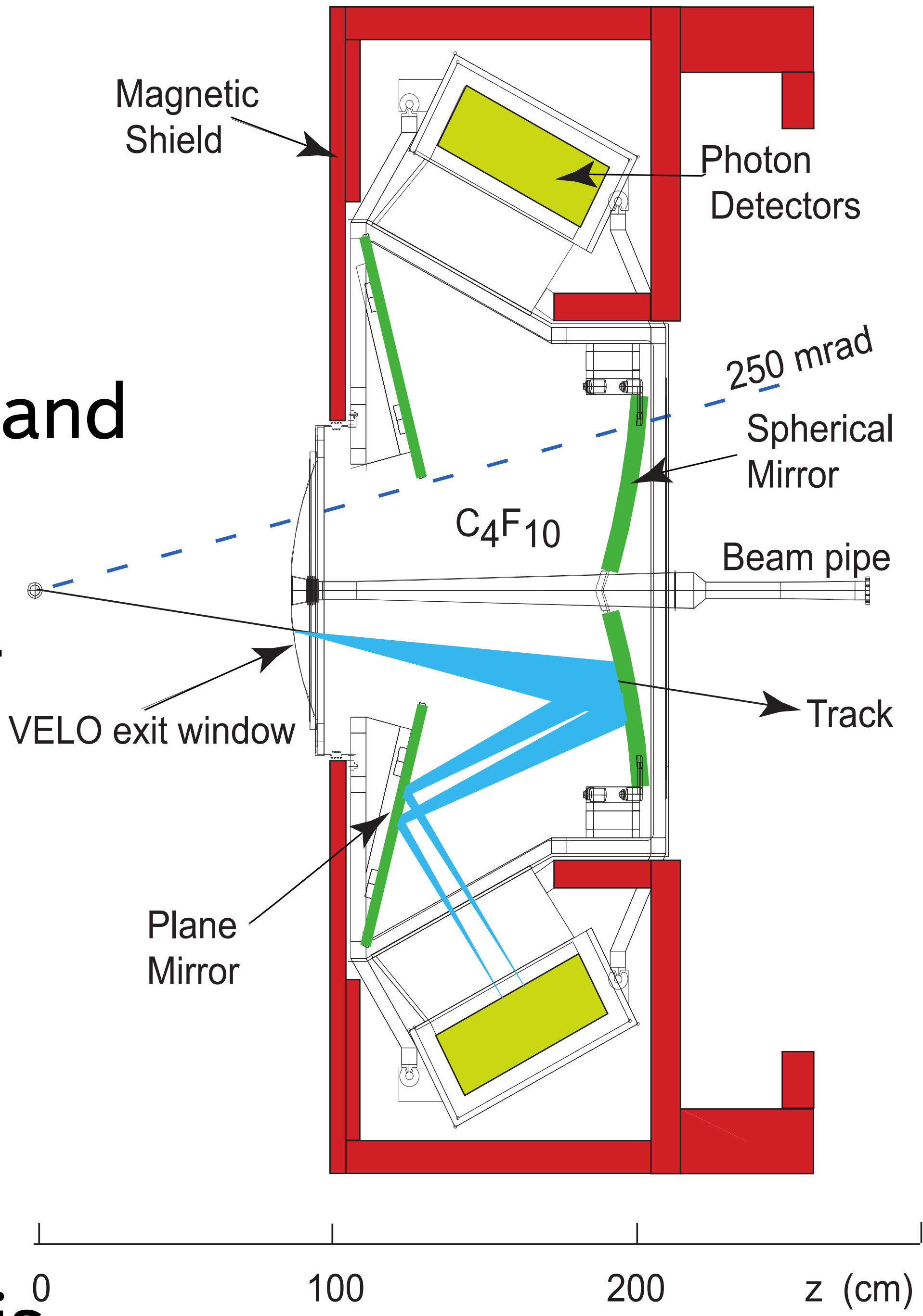


The LHCb RICH Collaboration, Adinolfi, M., Aglieri Rinella, G. et al. Eur. Phys. J. C (2013) 73: 2431

- RICH response is used to identify particles
  - e.g. separate pions and kaons

RICH ID Simulation

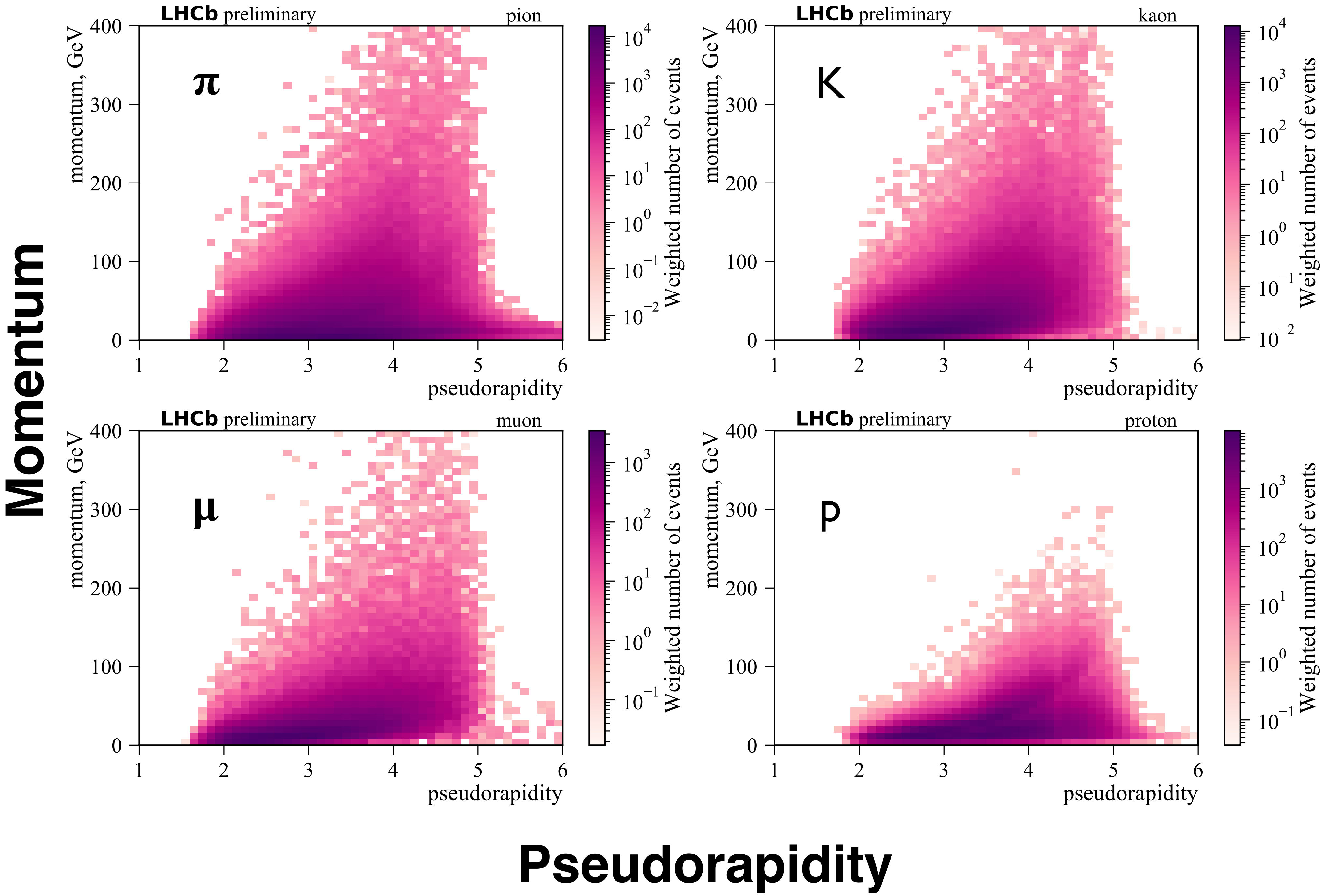
- Accurate RICH simulation involves:
  - tracing the particles through the radiators
  - Cherenkov light generation
  - photon propagation, reflection, refraction and scattering
  - Hybrid Photon Detector (photo-cathode + silicon pixel) simulation
- These require significant computing resources
- Besides:
  - quality of obtained simulated ID variables is not satisfactory when comparing to calibration data samples



RICH ID Simulation

- Accurate RICH simulation involves:
  - tracing the particles through the radiators
  - Cherenkov light generation
  - photon propagation, reflection, refraction and scattering
  - Hybrid Photon Detector (photo-cathode + silicon pixel) simulation
- These require significant computing resources
- Besides:
  - quality of obtained simulated ID variables is not satisfactory when comparing to calibration data samples
- Let’s use ML:
  - train generative model to directly convert track kinematics into ID variables
    - can train directly on calibration data samples

Calibration Samples

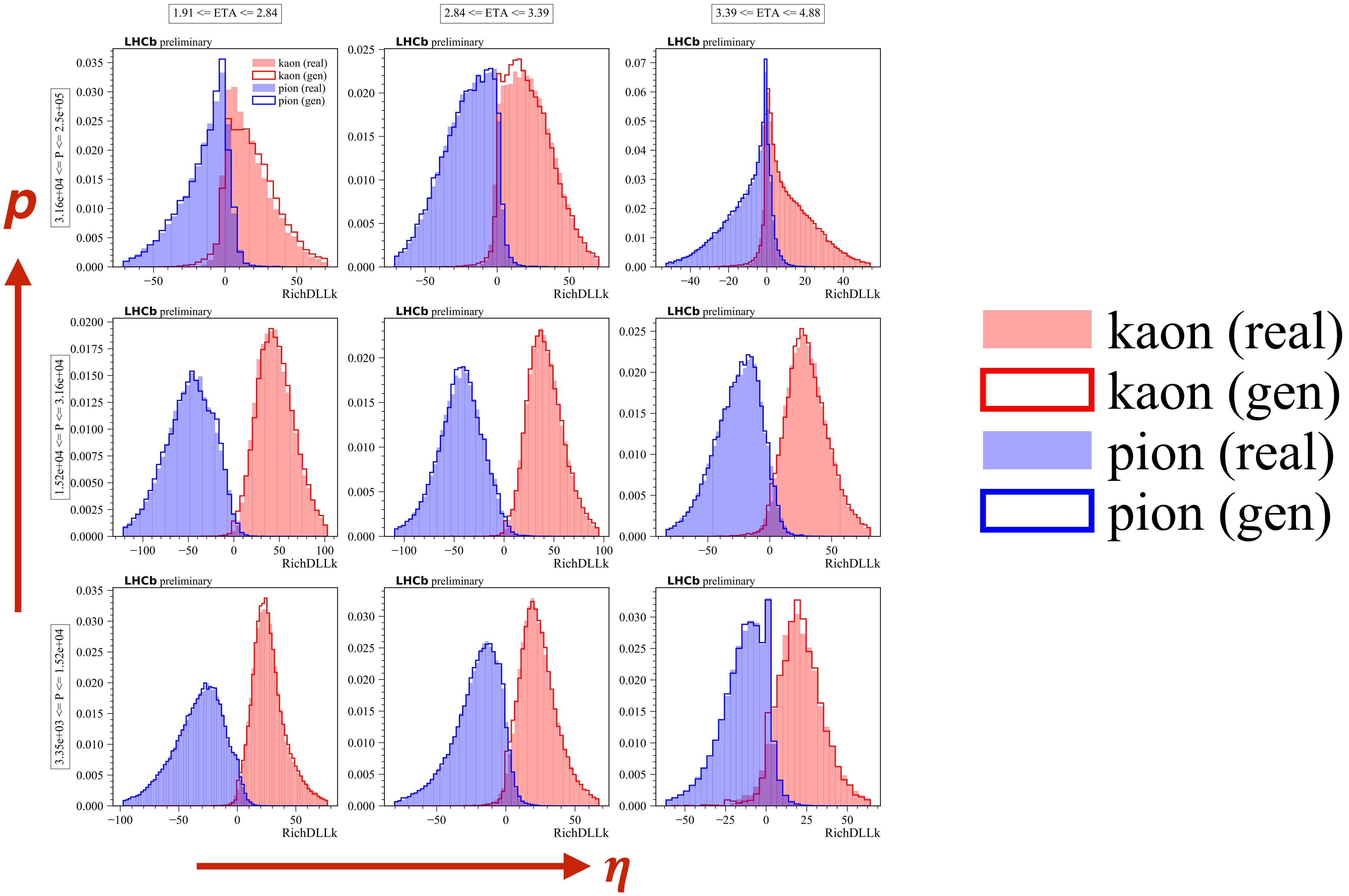


◇ Minor problem: different phase space for different particles

Technical Details

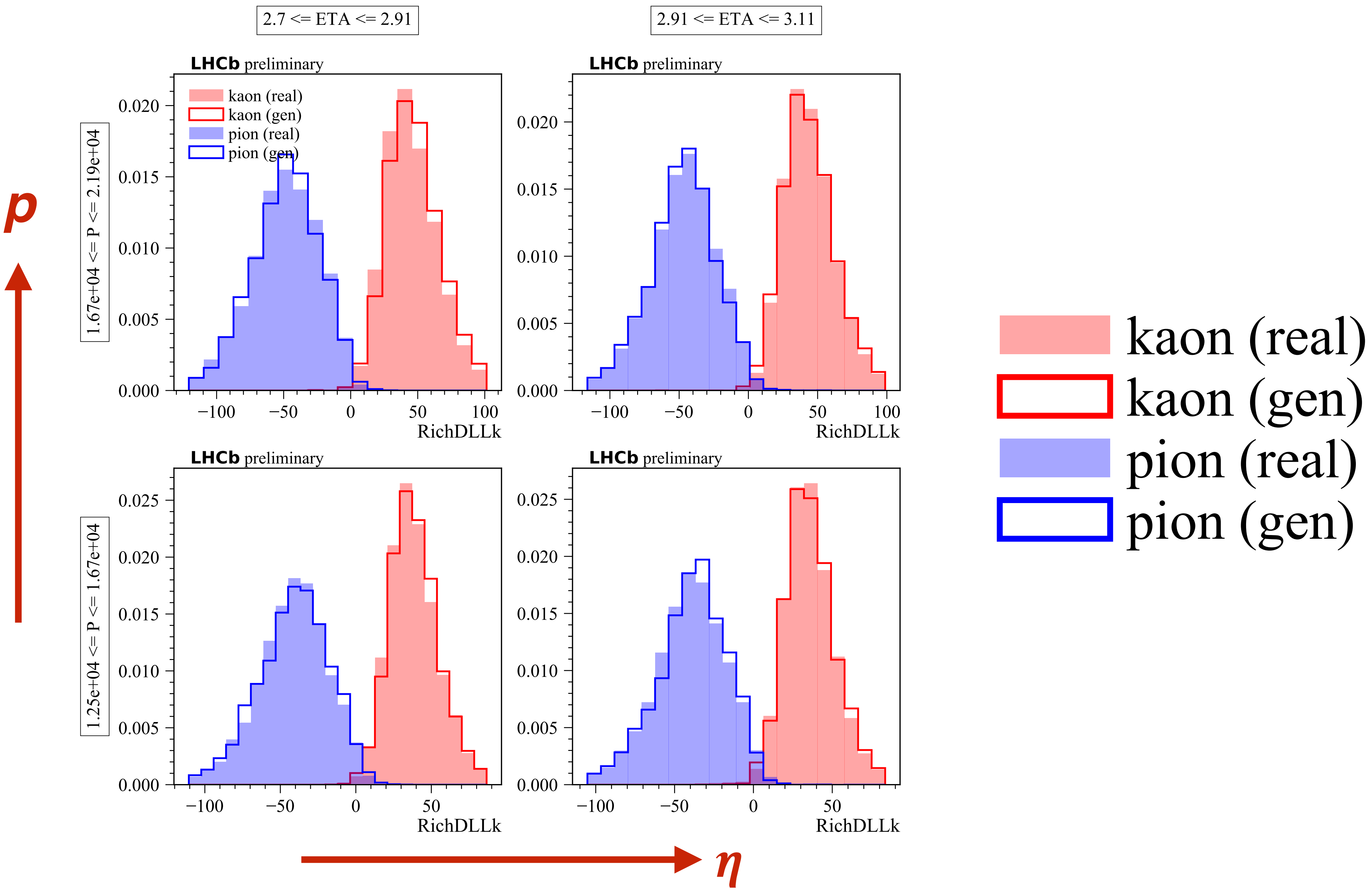
- 10 hidden fully-connected layers for both generator and discriminator
  - 128 neurons each
  - ReLU activation
- 64-dimensional latent space (noise shape)
- 256-dimensional discriminator output
- 15 discriminator updates per 1 generator update
- RMSProp optimizer, exp decaying learning rate

RICH ID Separation



- ◇ Is this generation quality good or bad?
  - ◇ depends on what it is used for…

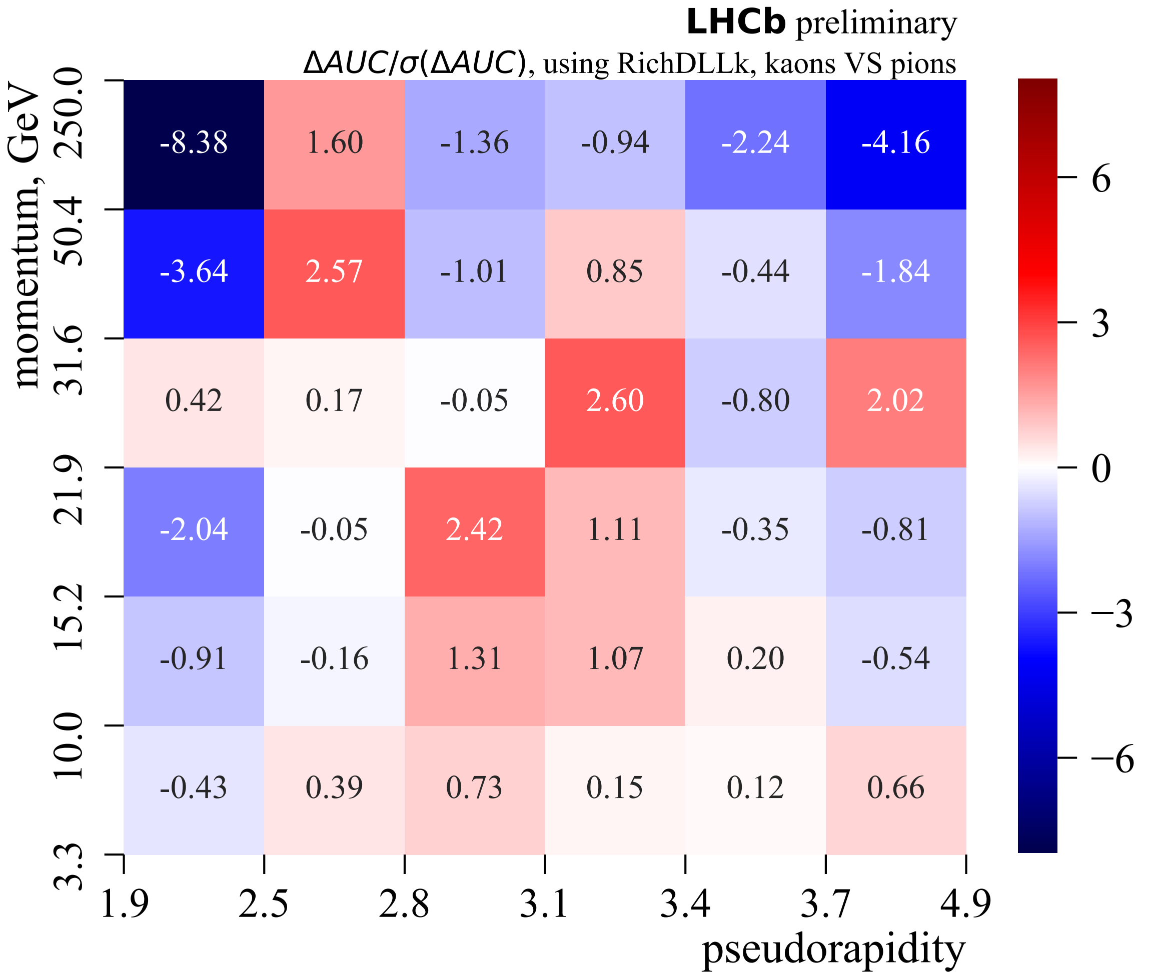
RICH ID Separation (Zoomed)



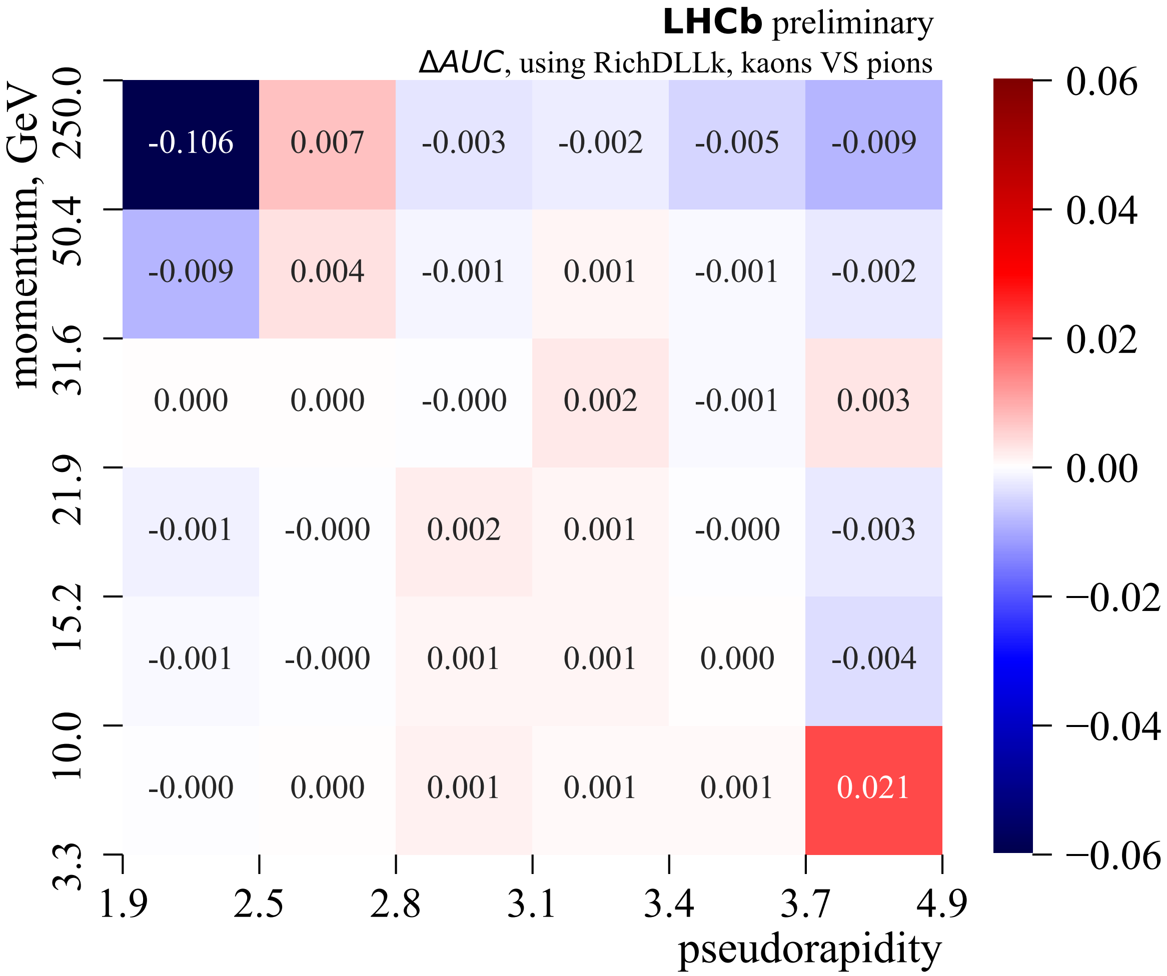
- Is this generation quality good or bad?
  - depends on what it is used for…

Comparing Separation Power

$(AUC_{REAL}-AUC_{GEN})/\sigma_{AUC}$



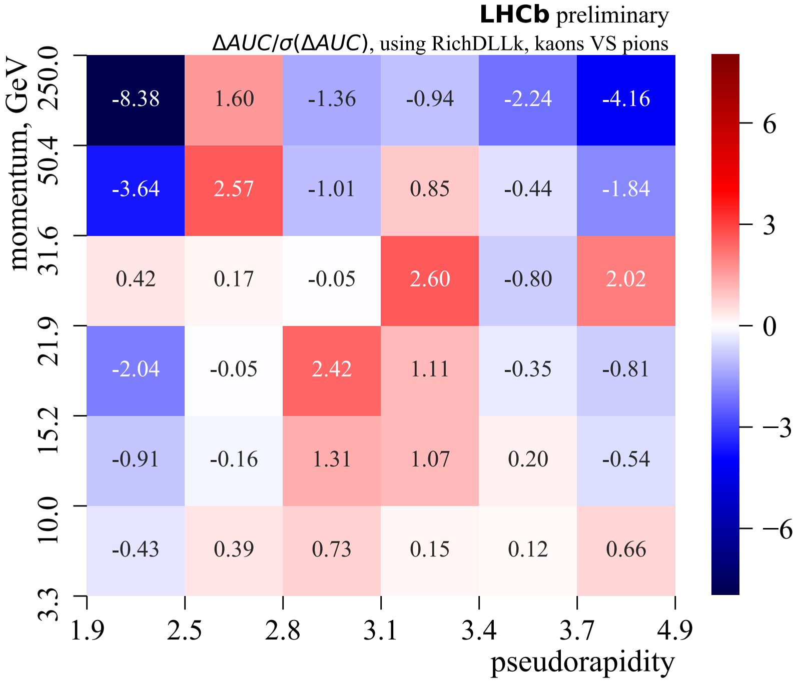
$AUC_{REAL}-AUC_{GEN}$

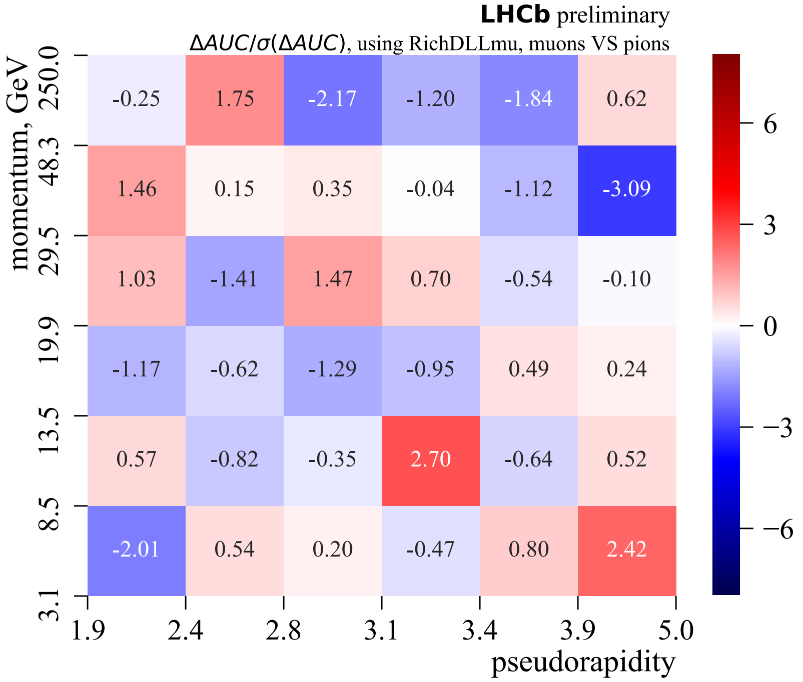


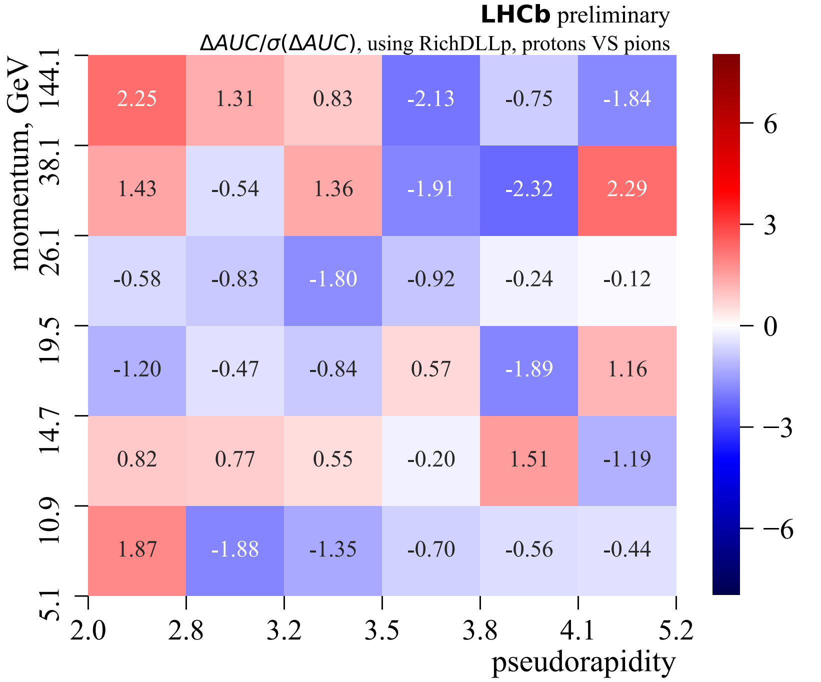
- The final figure of merit is driven by particular physics analysis using this generated ID
  - FOM is as accurate as can be evaluated from available calibration statistics
    - fundamental limitation

Comparing Separation Power

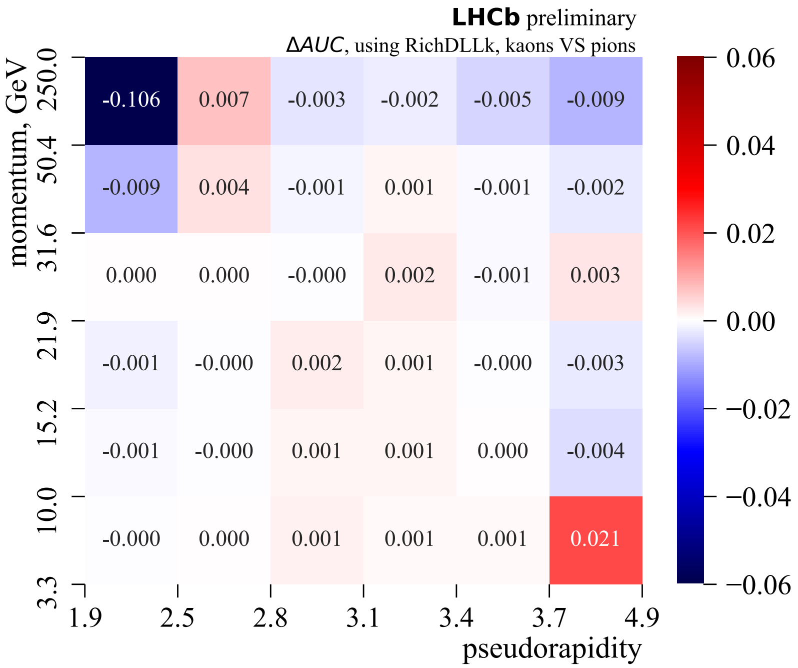
$(AUC_{REAL}-AUC_{GEN})/\sigma_{AUC}$

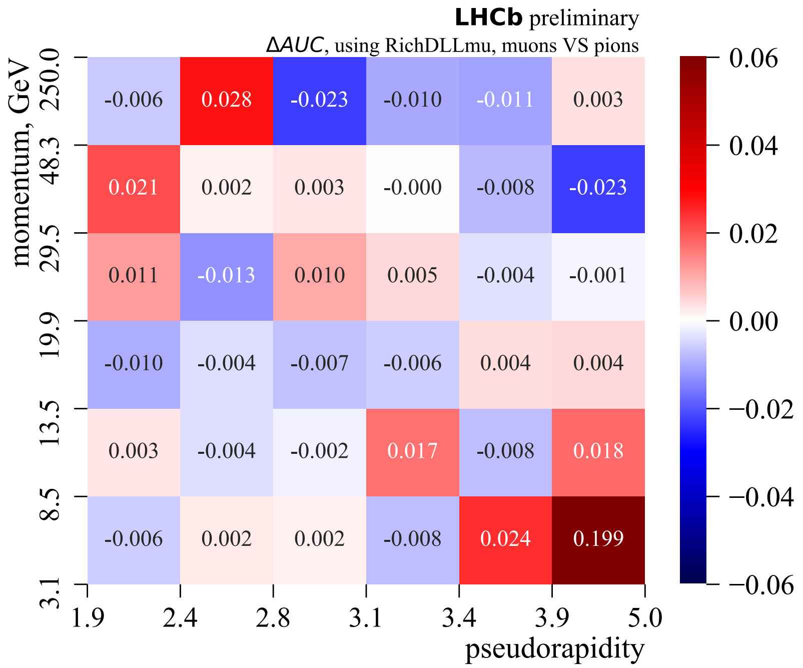


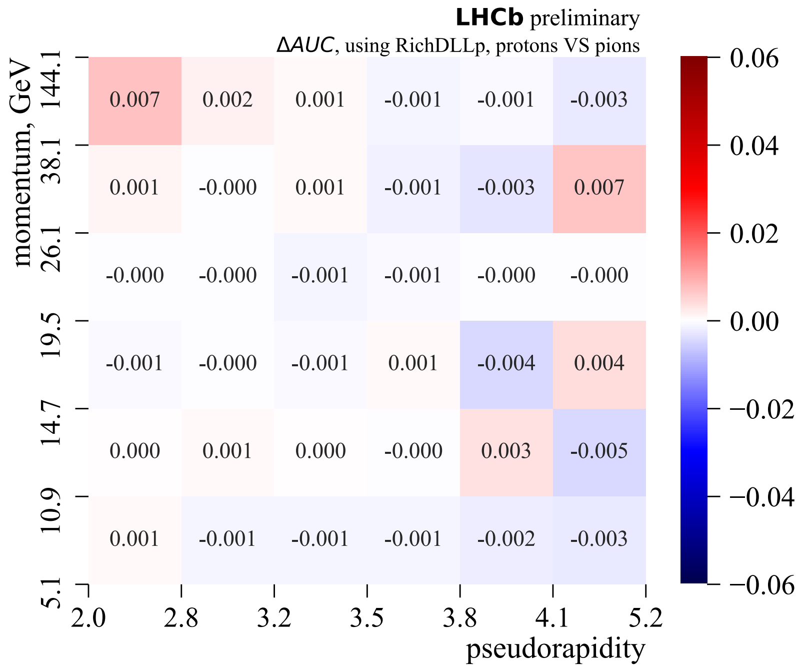




$AUC_{REAL}-AUC_{GEN}$







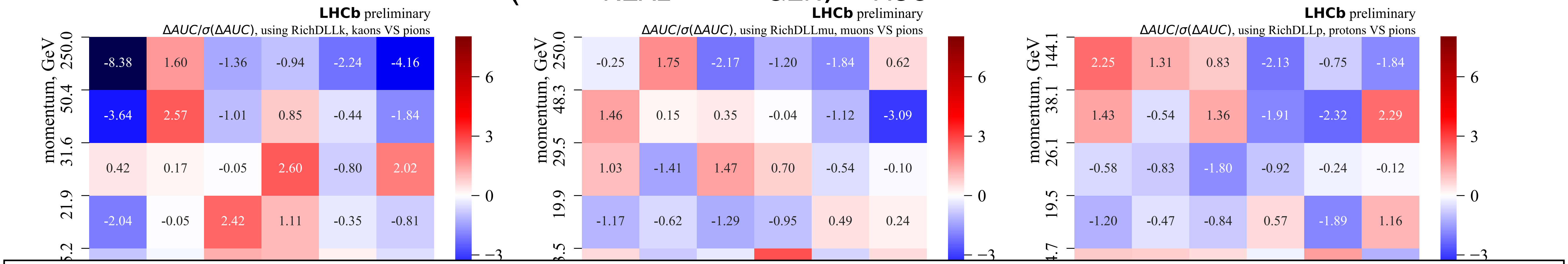
K vs π, using RichDLLk

μ vs π, using RichDLLmu

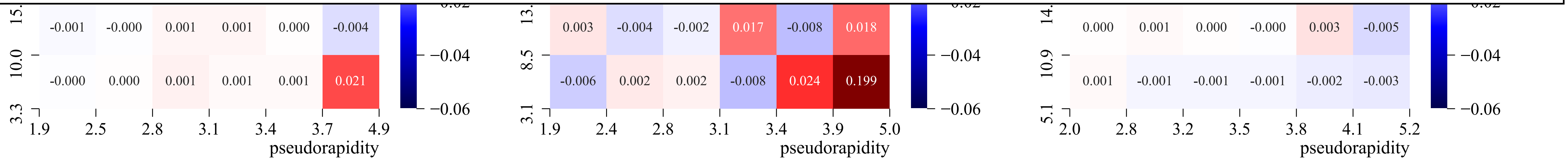
p vs π, using RichDLLp

Figure Of Merit

$(AUC_{REAL}-AUC_{GEN})/\sigma_{AUC}$



- The final figure of merit is driven by the particular physics analysis that uses this generated ID
  - FOM is as accurate as can be evaluated from available calibration statistics
    - fundamental limitation



K vs π, using RichDLLk μ vs π, using RichDLLmu p vs π, using RichDLLp

Conclusions

- Surrogate generative models demonstrate extraordinary progress in current years
- There are many applications for use in HEP
- Generative models need attention to ensure scientifically solid results
    - satisfying boundary conditions, control of scientifically important but marginal statistics
    - appropriate evaluating the quality of the model
    - propagating model intrinsic systematics to the systematic uncertainties of the final scientific result
- Success stories are available

Generative Model. ML Perspective

- Generative models look very different from regression/classification models
  - actually they are not that different
- Consider set of objects each of which is described by a vector of parameters
  - we arbitrary split this vector into “features” **x** and “labels” **y**
- For classification/regression problem we search for deterministic function f which approximates dependency **y** from **x**: **y=f(x)**
  - in probabilistic approach we search for probability **p(ylx)**
- For generation problem we want to sample objects for a given label
  - we search for probability **p(xly)**
    - **y** for generative model is called “condition”
    - condition may be absent - unconditional generative model

Generative Model. ML Perspective

- In both discriminative model and generative model we want to get probability for subset of object parameters conditioned by another subset of object parameters
- Discriminative models:
  - evaluate distributions for few, usually redundant, parameters conditioned by many features
    - can discriminate basing on this parameters
- Generative models:
  - evaluate many features conditioned by few parameters (conditions)
    - can sample these features
- NB: logistic regression + binomial distribution = generative model
  - for the binary objects