Deep Learning at colliders

SOS 2022

Anja Butter, ITP Heidelberg/LPNHE Paris

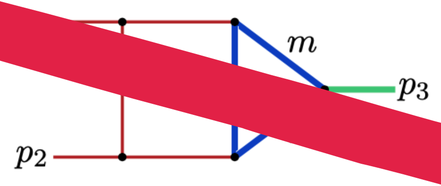
LPNHE
PARIS

Pre Cocktail Talk

Multi-loop calculations with INNs

Profiting from the Jacobian

Precision predictions based on loop diagrams



Analytic expression for loop amplitude

$$G = \int_{-\infty}^{\infty} \left(\prod_{l=1}^{L} \frac{\mathrm{d}^D k_l}{i\pi^{\frac{D}{2}}} \right) \prod_{j=1}^{N} \frac{1}{(q_j^2 - \quad i\delta)^{\nu_j}}$$

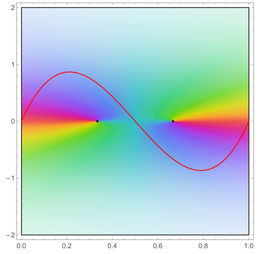
$$= \int_0^1 \prod_{j}^{N-1} \mathrm{d}x_j \; \frac{\quad +1)D/2}{F^{\nu - LD/2}} = \int_0^1 \prod_{j=1}^{N-1} \mathrm{d}x_j \; I(\vec{x})$$

Parametrize with Feynman parameters

Still contains singularities

Solved by contour deform[illegible] due to Cauchy's theorem

$$\int_0^1 \prod_{j}^{N} \mathrm{d}x_j \; I(\quad = \int_0^1 \prod_{j=1} \mathrm{d}x_j \; \det\left(\frac{\partial \vec{z}(\vec{x})}{\partial \vec{x}} \right) I(\vec{z}(\vec{x}))$$

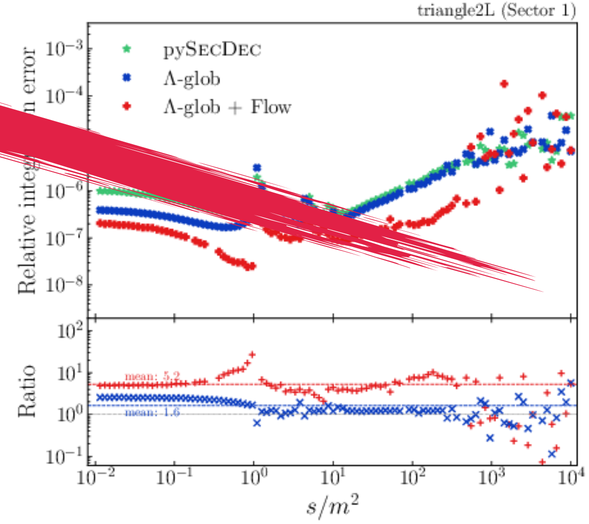


Optimal parametrization = minimal variance

Turn it into an ML Problem

Parametrization $\rightarrow z = \mathrm{INN}(x)$

Variance $\rightarrow \mathscr{L}$



R. Winterhalder, et al. [2112.09145]

A short history of ML in HEP

First HEP NN papers

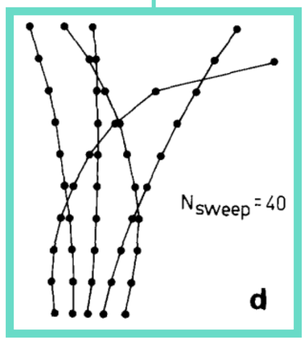
Track finding

Denby (LAL, Orsay) ’87

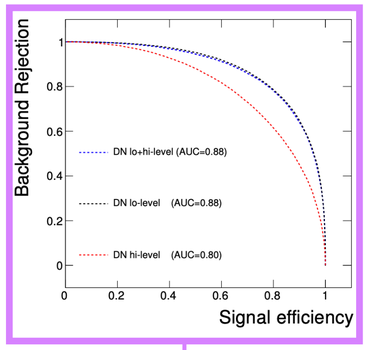
Peterson (Lund) ’88

→ Jet identification

1987/88



1987



2014

Relaunch

Deep Learning in HEP

Signal vs Background

P. Baldi, P. Sadowski, D. Whiteson

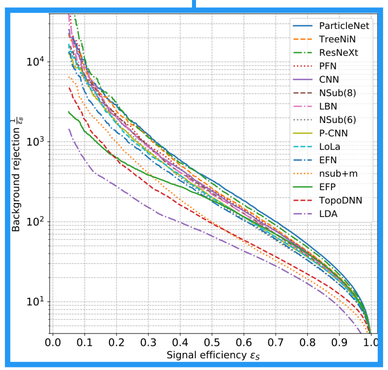
2014

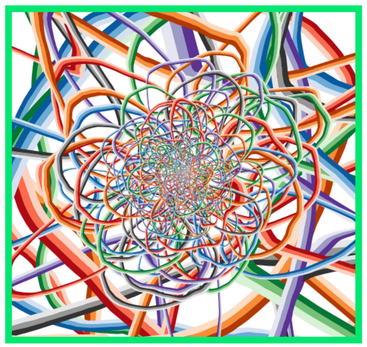
First community Paper

Machine Learning Landscape of TopTagging

G. Kasieczka, et al.

2019





2019

First attempts at “understanding” neural networks

Deep Thinking

J. Thaler

2019

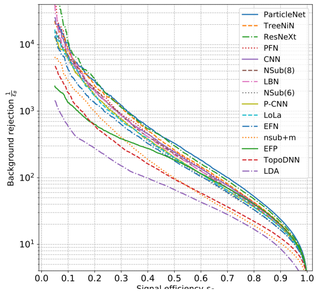
Today

?

2021

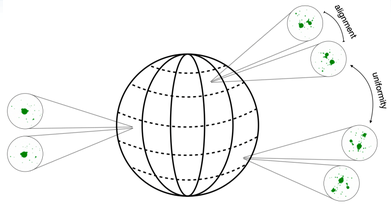
ML in particle physics 2022

Top tagging



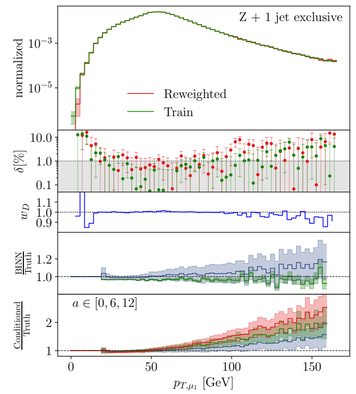
G. Kasieczka et al. [1902.09914]

Anomaly detection



B. Dillon et al. [2108.04253]

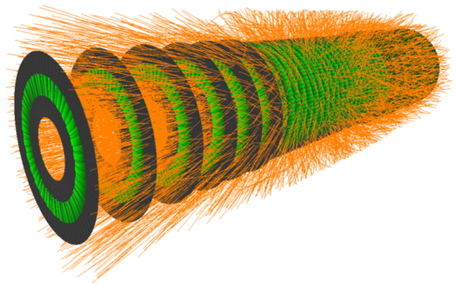
Event generation



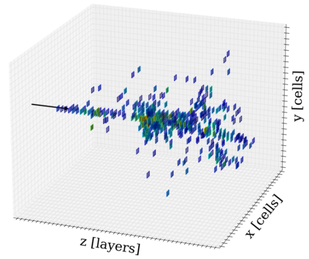
A. Butter et al. [2110.13632]

Track reconstruction

Kaggle challenge



Detector simulation



E. Buhmann et al. [2112.09709]

Classification

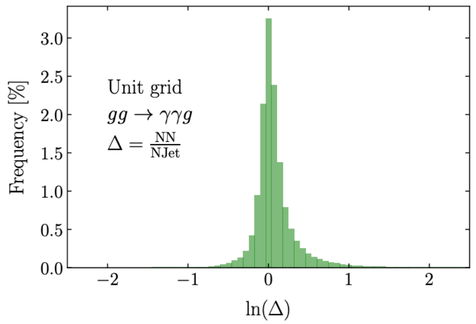
Generative models

Graph networks

Bayesian networks

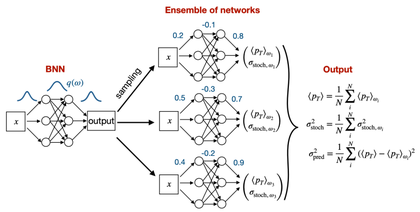
Regression

Amplitude estimation



J. Aylett-Bullock, et al. [2106.09474]

Jet calibration & uncertainties



G. Kasieczka et al. [2003.11099]

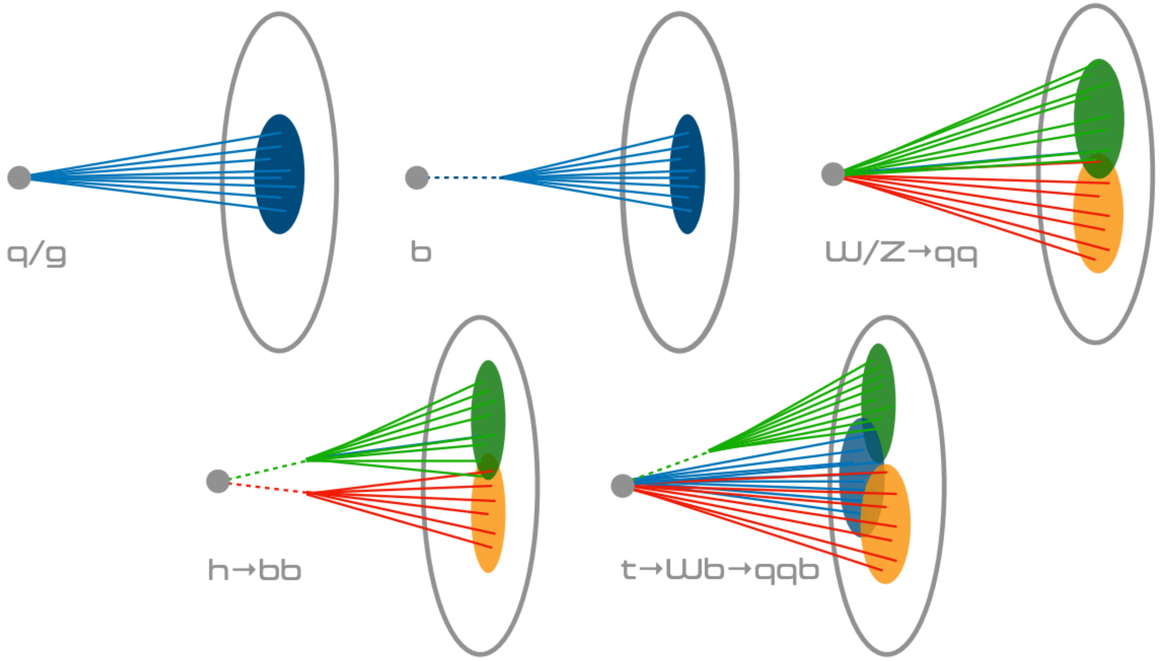
Complete citations $\mathcal{O}(800)$
https://iml-wg.github.io/HEPML-LivingReview/

I. Top tagging

Starting with jet classification

- How to distinguish **top** from **QCD** jets?
- Immensely important for top & Higgs physics studies
- Standard supervised classification task

- **ML ingredients**
  1. Loss function
  2. Data format & Preprocessing
  3. Architecture → follows from data format
  4. Training parameters



E. Moreno et al. [1909.12285]

1. Classification loss function

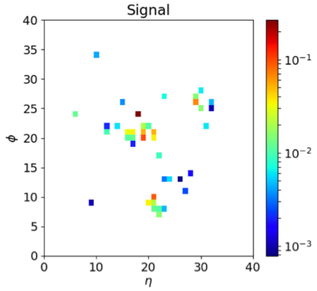
$$\mathscr{L} = \sum_{x_i} -\log C(x_i)\, y_i - \log(1 - C(x_i))\,(1 - y_i)$$

$$= -\int \mathrm{d}x\, p_{top}(x)\ \log C(x) + p_{QCD}\ \log(1 - C(x))$$

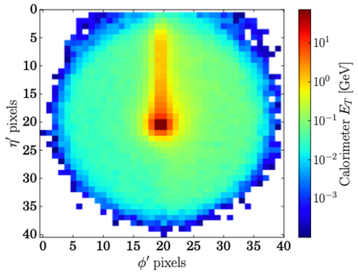
Variance yields $\rightarrow$ $\dfrac{p_{top}(x)}{p_{QCD}(x)} = \dfrac{C(x)}{1 - C(x)}$

→ See Veronica's lecture

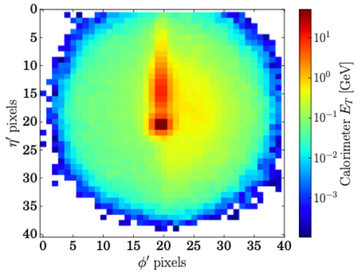
Jet representation I



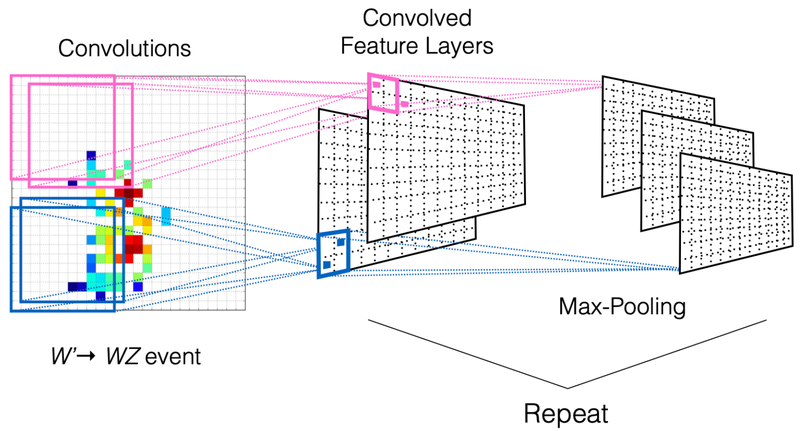
Single jet



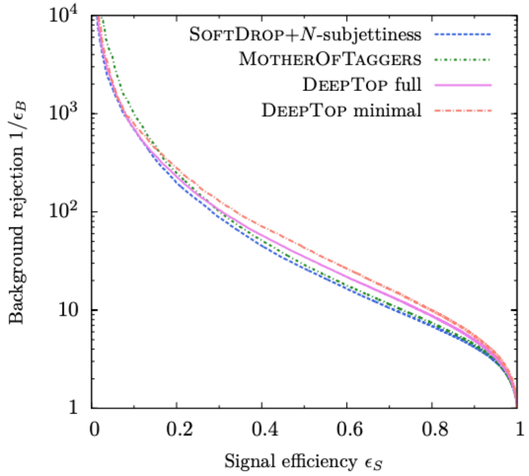
QCD jet averaged



top jet Averaged



L. Oliveira et al. [1511.05190]



G. Kasieczka et al. [1701.08784]

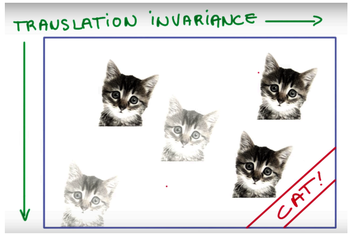
CNNs

Use **equivariance** principle for translation t:

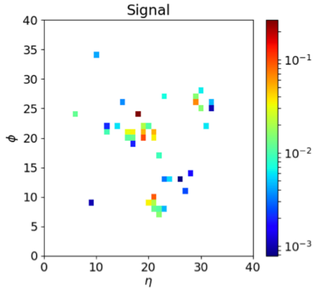
$$\mathrm{CNN}(t(x)) = t(\mathrm{CNN}(x))$$

Pooling layer

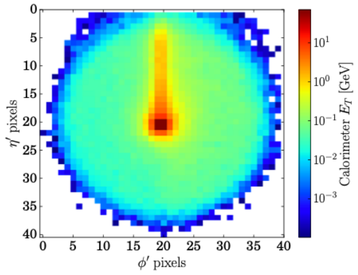
→ **Invariance**



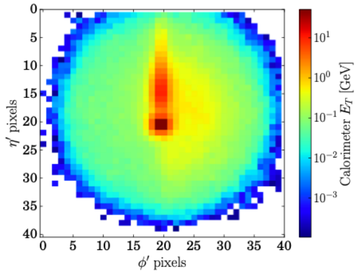
Preprocessing and hyperparameters



Single jet



QCD jet
averaged



top jet
Averaged

1. **Preprocessing**

   Normalization!!! $(x - \mu)/\sigma$

   Shift centroid → center,

   Rotate p_T-axis → 12 o'clock

2. **Optimizer**

   Stochastic gradient descent → Adam (standard)

   LR scheduling, …

3. **Network architecture**

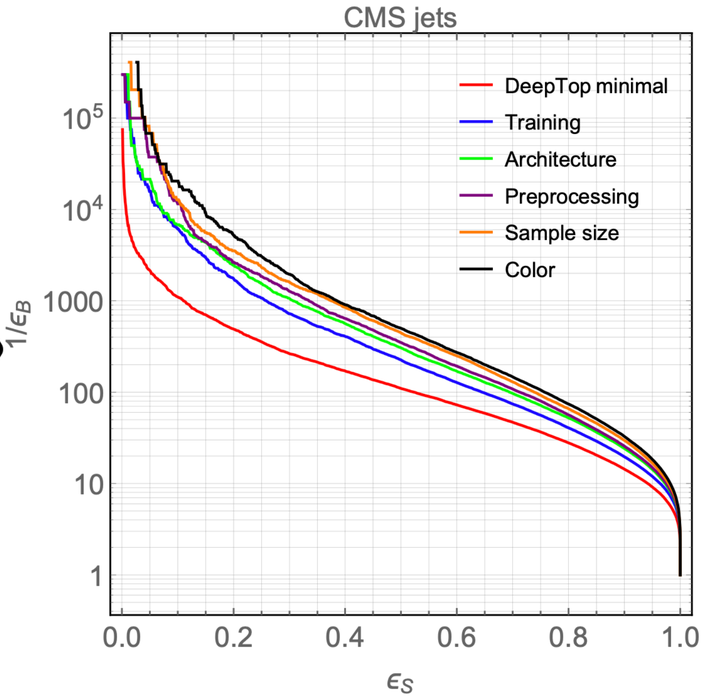
   #layers, #nodes/layer, #feature maps, …

   Activation function

   Batch size

   Weight initialization

4. **Sample size** (#Training data)

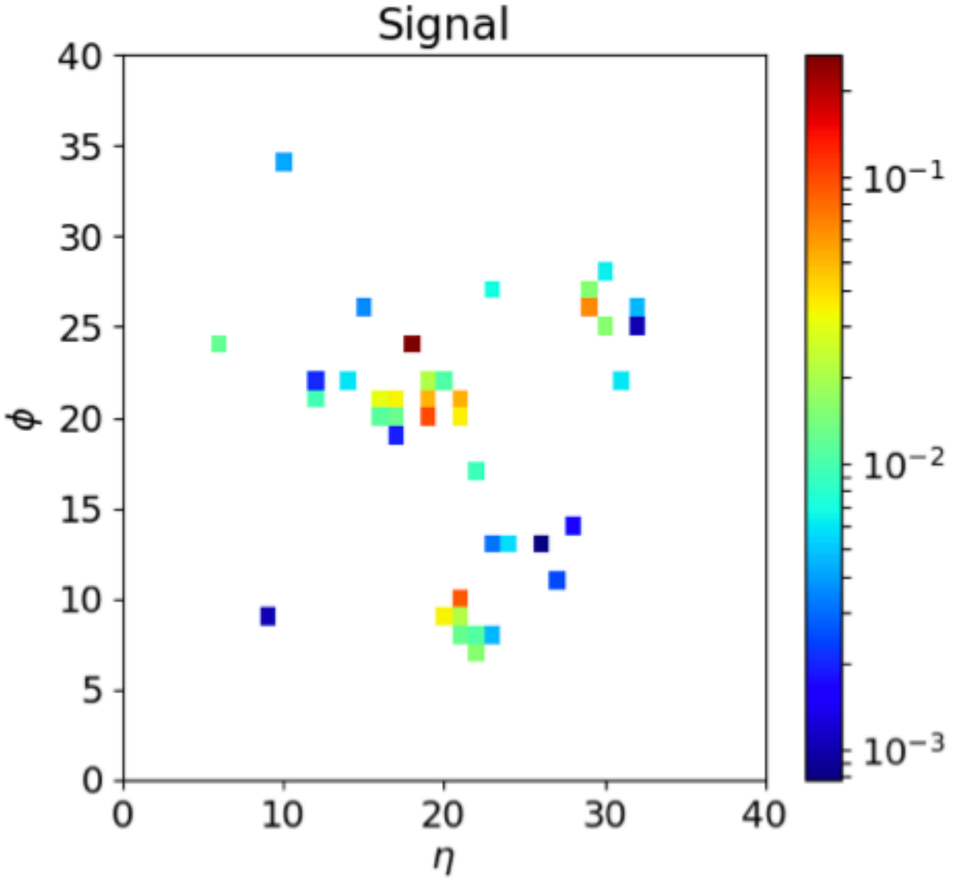


S. Macaluso et al. [1803.00107]

Our image ain't a very good image...

Not continuity, no edges, no cats....

vs



Jet representation II

How to represent a graph

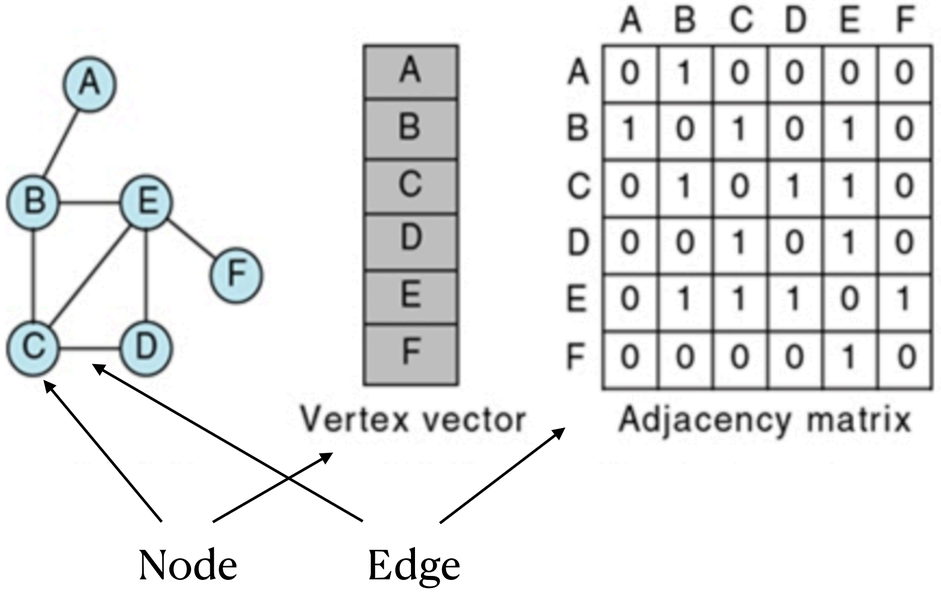
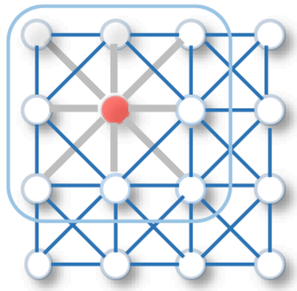
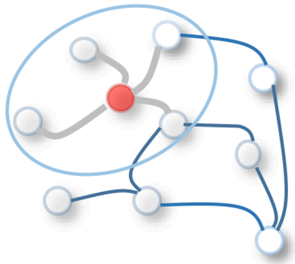


Image vs Graph





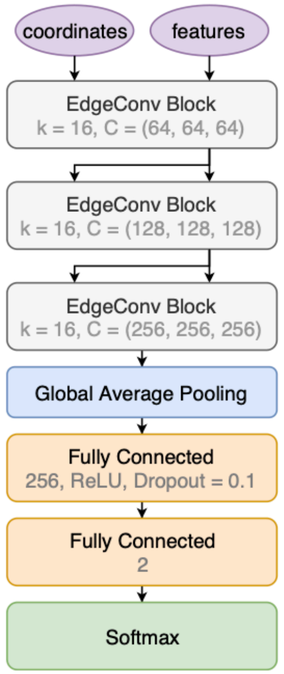
pixels → node

neighbouring pixel → neighbouring node (graph edges)

CNN → edge convolution

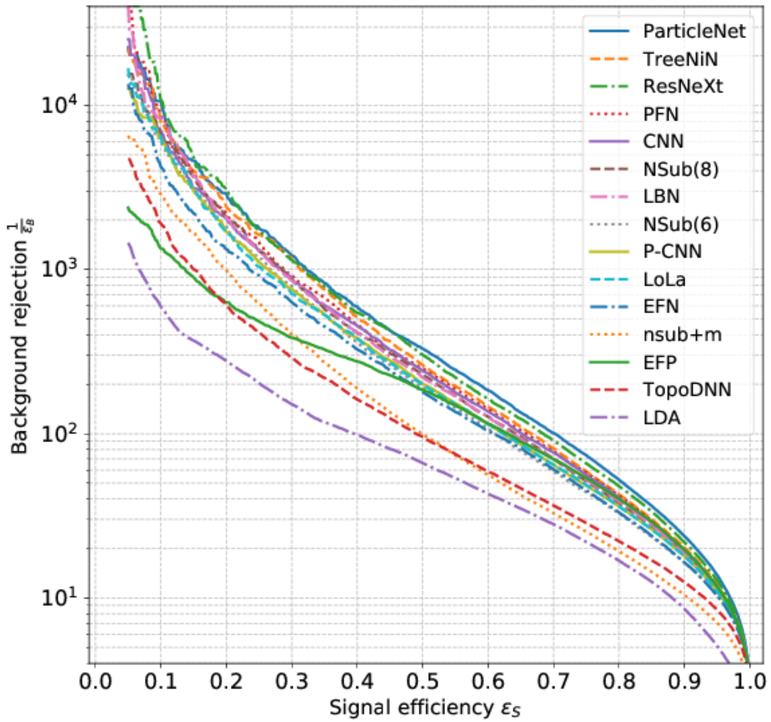
$$\vec{x}'_i = \frac{1}{k} \sum_{j=1}^{k} h_\Theta(\vec{x}_i, \vec{x}_{i_j} - \vec{x}_i)$$

The ML landscape of top taggers



(a) ParticleNet

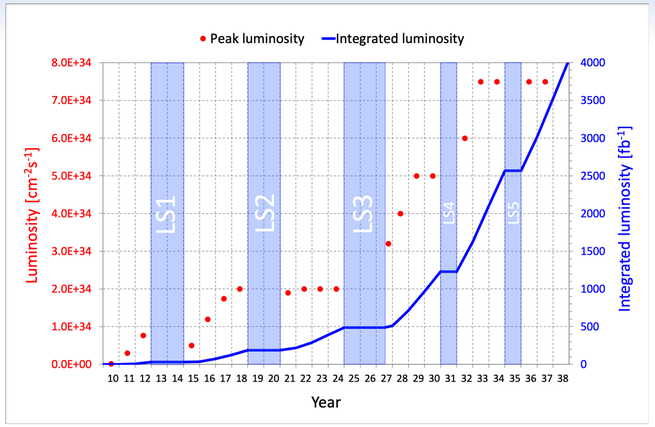
H. Qu, L. Gouskos [1902.08570]



Excellent performance for various different approaches!

Open questions towards HL-LHC

A biased selection



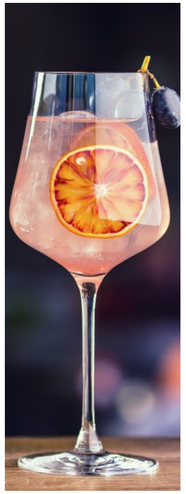
- Facing **25 times** the amount of data
- What do we need to understand the data? (*read:* find new physics)

- **Precision predictions**
  - Higher order amplitudes
  - Event generation
  - Shower
  - Detector simulation

- **Optimized analysis for high-dimensional data**
  - Likelihood free inference
    - Optimal Observables, Unfolding
  - Anomaly detection
  - Uncertainty treatment ?

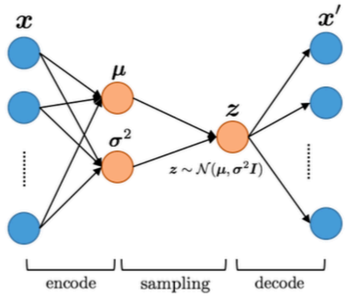
Problems beyond supervised classification/regression → How can machine learning help?

II. Event generation and detector simulation





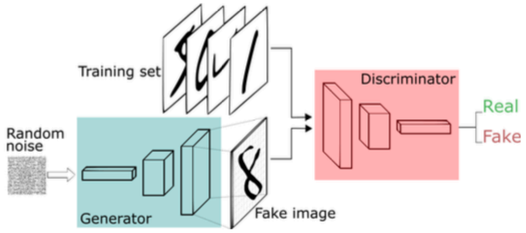
Generative Networks



VAE

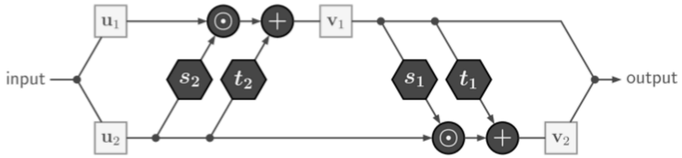
Variational autoencoders

all kinds of hybrids



GAN

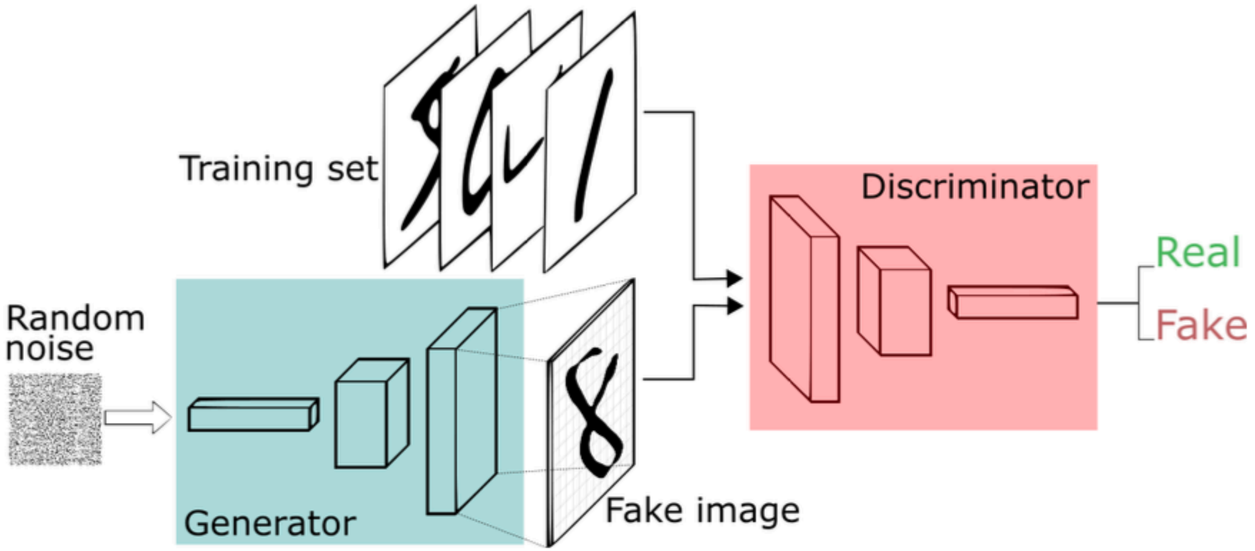
Generative adversarial networks



NF

Normalizing Flows

Generative Adversarial Networks



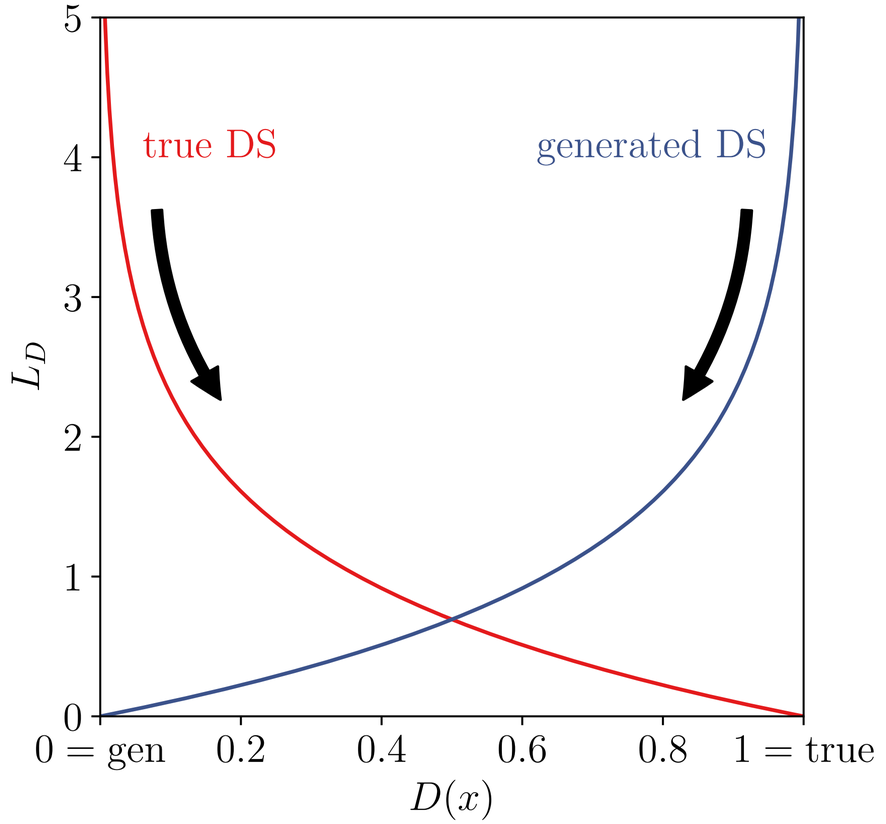
Discriminator

$$L_D = \left\langle -\log D(x) \right\rangle_{x \sim P_{Truth}} + \left\langle -\log(1 - D(x)) \right\rangle_{x \sim P_{Gen}}$$

Generator

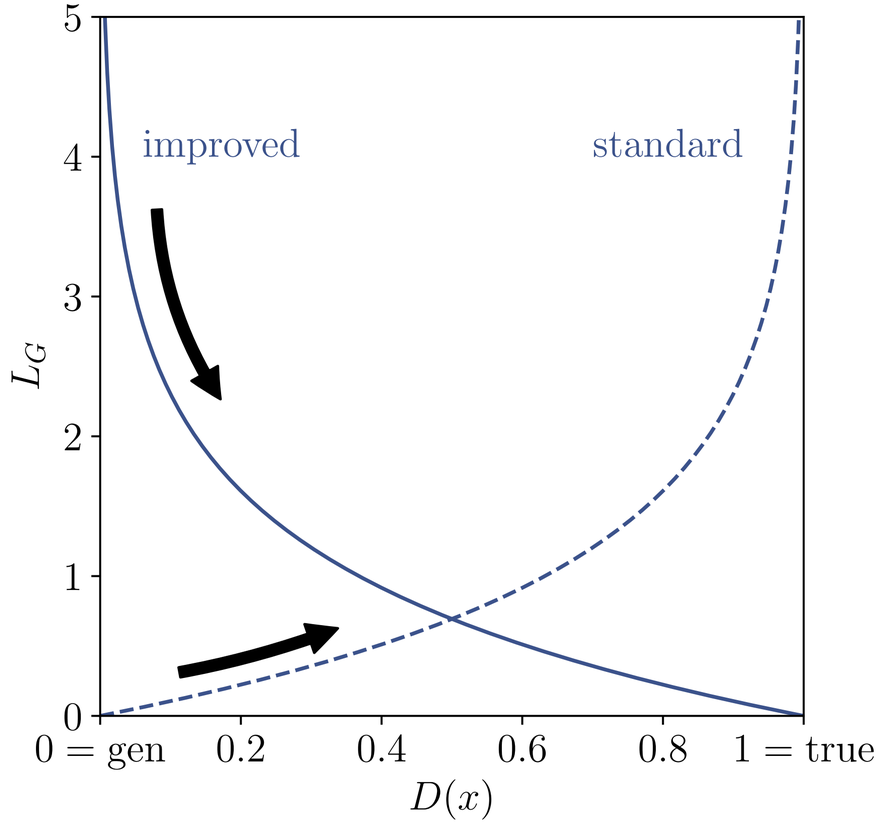
$$L_G = \left\langle -\log D(x) \right\rangle_{x \sim P_{Gen}}$$

Training the discriminator



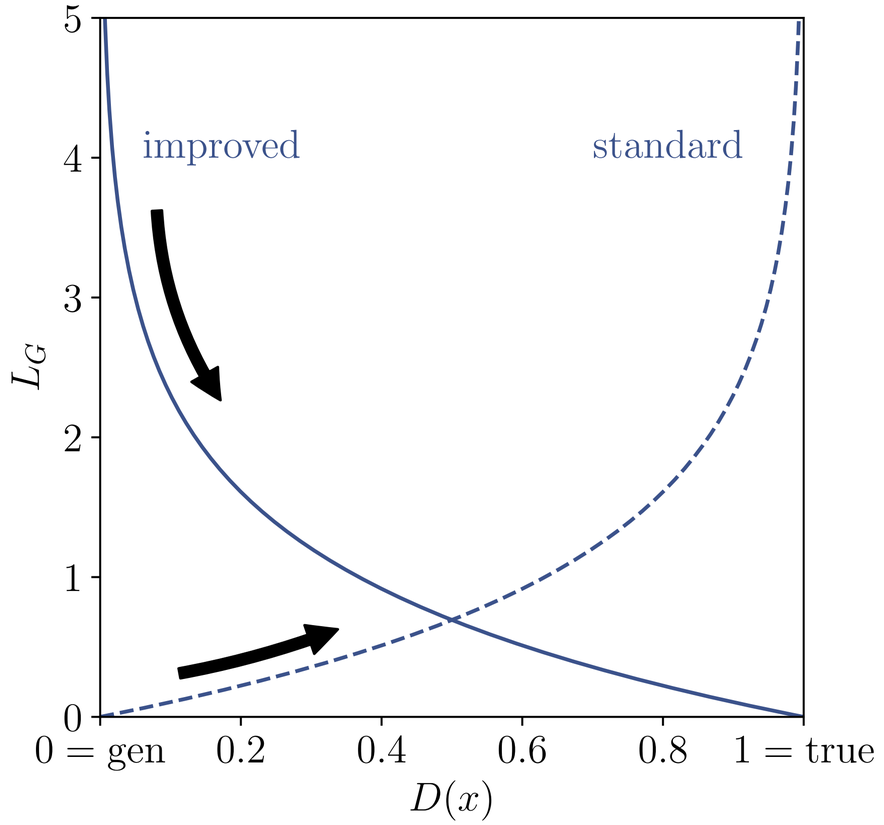
Minimize $L_D = \left\langle -\log D(x) \right\rangle_{x \sim P_{true}} + \left\langle -\log 1 - D(x) \right\rangle_{x \sim P_{gen}}$

Training the discriminator



Maximize $L_D = \left\langle -\log(1 - D(x)) \right\rangle_{x \sim P_{gen}}$

Training the discriminator



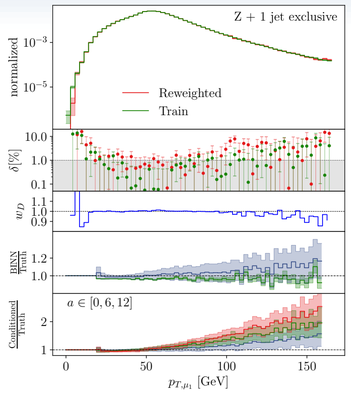
+ Great at generating individual samples
+ Works for event generation

- not the best control over DOF
- not the best handle for distributions ?

Minimize $L_D = \left\langle -\log D(x) \right\rangle_{x \sim P_{gen}}$

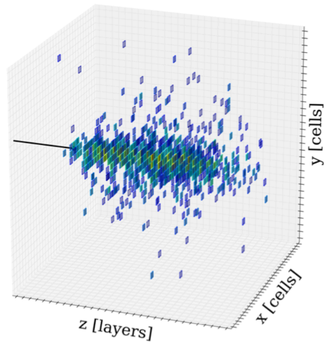
Forward simulations with generative networks

Event generation

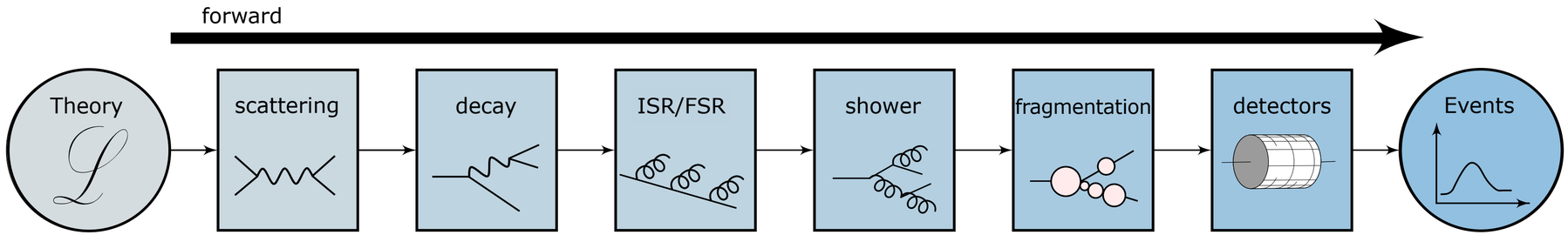


→ Otten et al.
→ Gao et al.
→ Bothmann et al.
→ Stienen et al.
→ AB, et al.
→ and many more

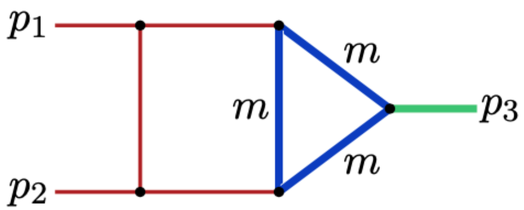
Detector simulation



→ CaloGAN by M. Paganini et al.
→ BIBAE by E. Buhman, S. Diefenbacher et al.
→ CaloFlow by C. Krause , D. Shih
→ and many more

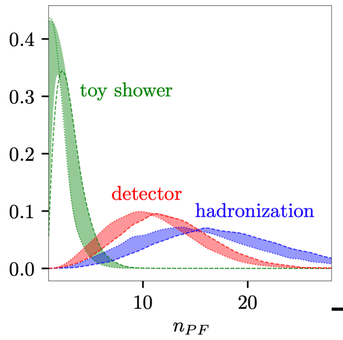


Loop amplitudes



→ R. Winterhalder, et al.

Shower simulation



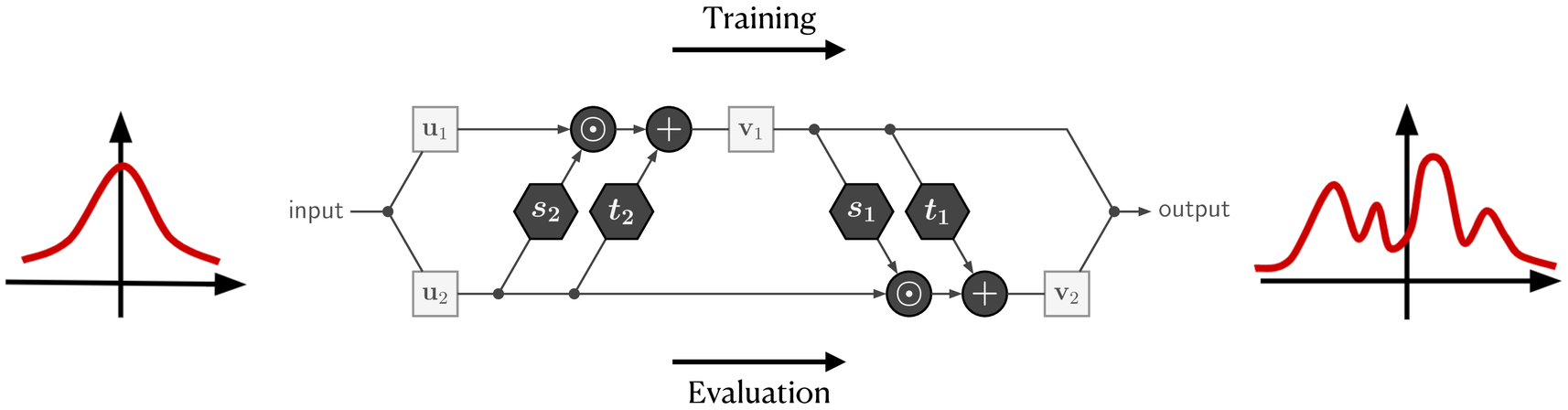
→ S. Bieringer, et al.

Particularly promising architecture
→ Normalizing flows

Normalizing flows

Invertible networks for complex transformations

+ Bijective mapping
+ Tractable Jacobian $\rightarrow p_x(x) = p_z(z) \cdot J_{NN}$
+ Fast evaluation in both direction



Training on density $t(x)$

$\rightarrow$ Minimize difference

$$\mathcal{L} = \log p_x(x)/t(x)$$
$$= \log p_z(z(x))\ J_{NN} / t(x)$$

Training on samples x

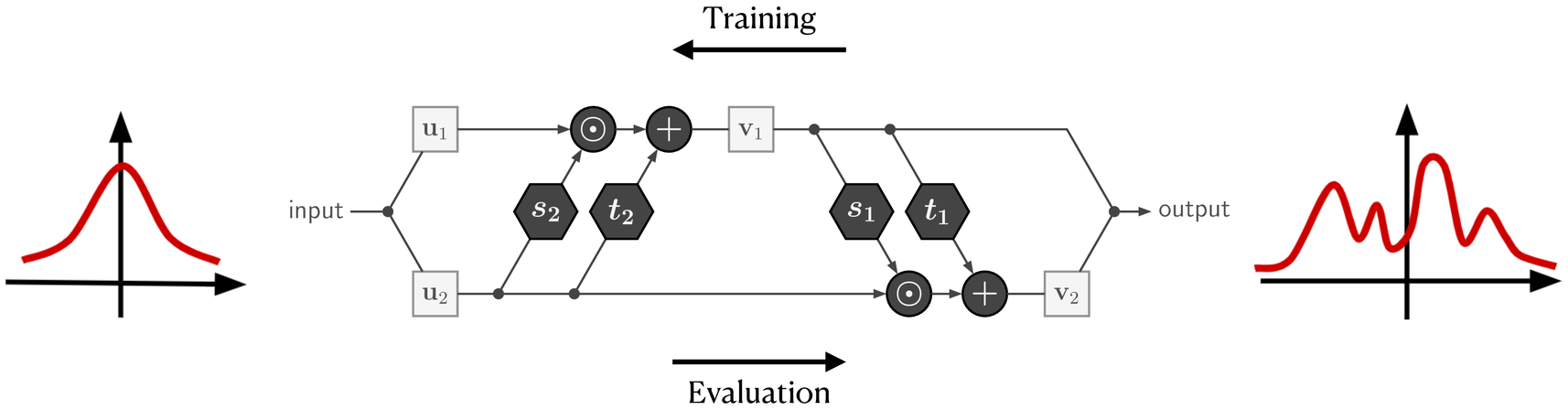
$\rightarrow$ Maximize the log-likelihood

$$\mathcal{L} = \log p(\theta \mid x)$$
$$= \log p(z \mid \theta) + \log J_{NN} + p(\theta)$$

Normalizing flows

Invertible networks for complex transformations

+ Bijective mapping
+ Tractable Jacobian $\rightarrow p_x(x) = p_z(z) \cdot J_{NN}$
+ Fast evaluation in both direction



Training on density $t(x)$

$\rightarrow$ Minimize difference

$$\mathcal{L} = \log p_x(x)/t(x)$$
$$= \log p_z(z(x))\ J_{NN} / t(x)$$

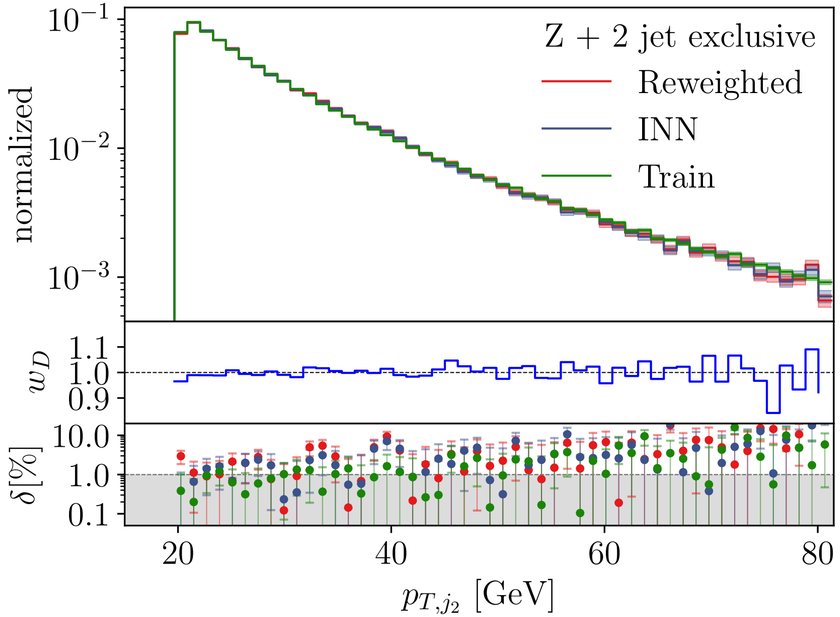
Training on samples x

$\rightarrow$ Maximize the log-likelihood

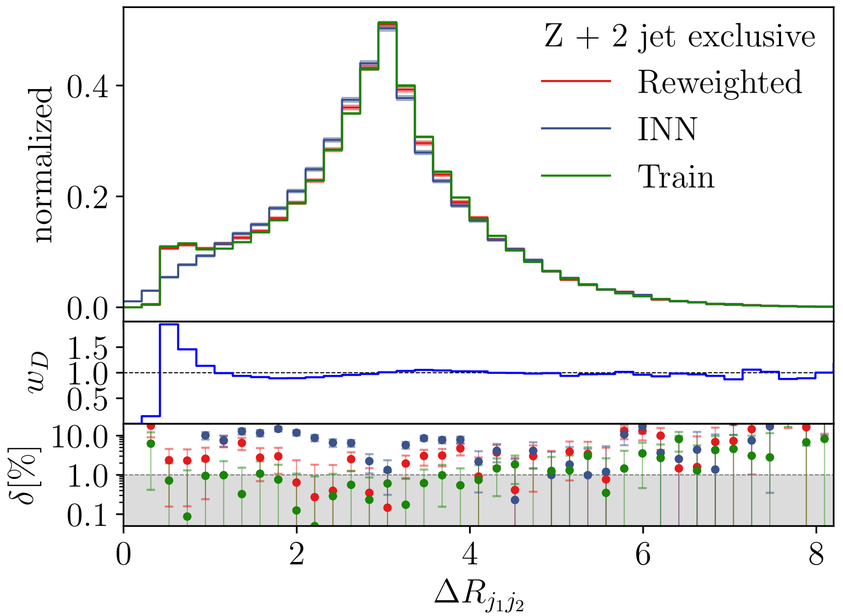
$$\mathcal{L} = \log p(\theta \mid x)$$
$$= \log p(z \mid \theta) + \log J_{NN} + p(\theta)$$

Basic INN

Some things work out of the box ...



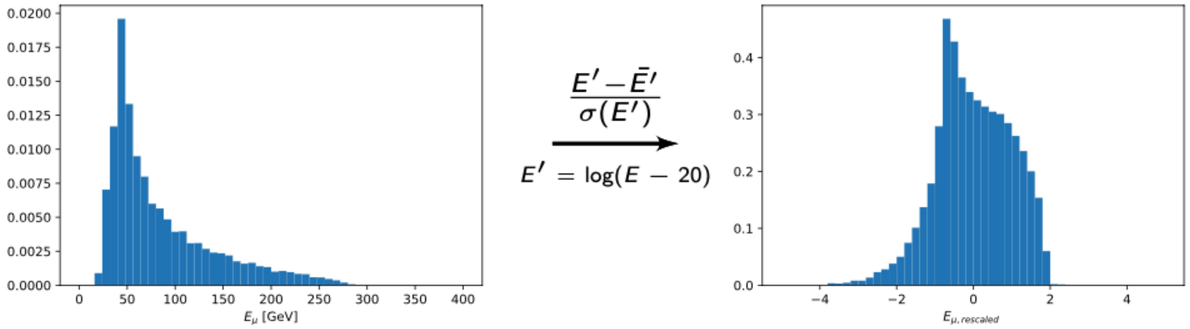
... others not so much



So what's the problem?

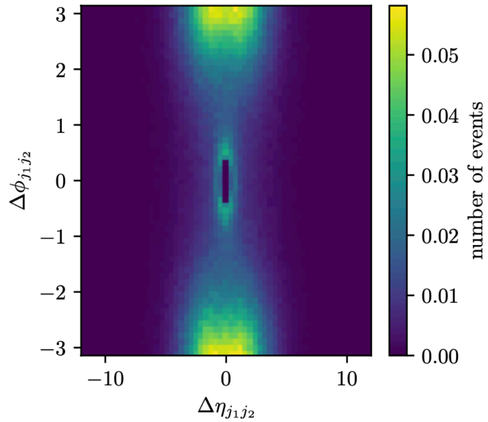
Challenges

Edges and narrow features



- Phase space symmetries in architecture
  - Eg. p_T, ϕ instead of p_x, p_y
- Preprocessing $\log(p_T - p_{T\text{min}})$

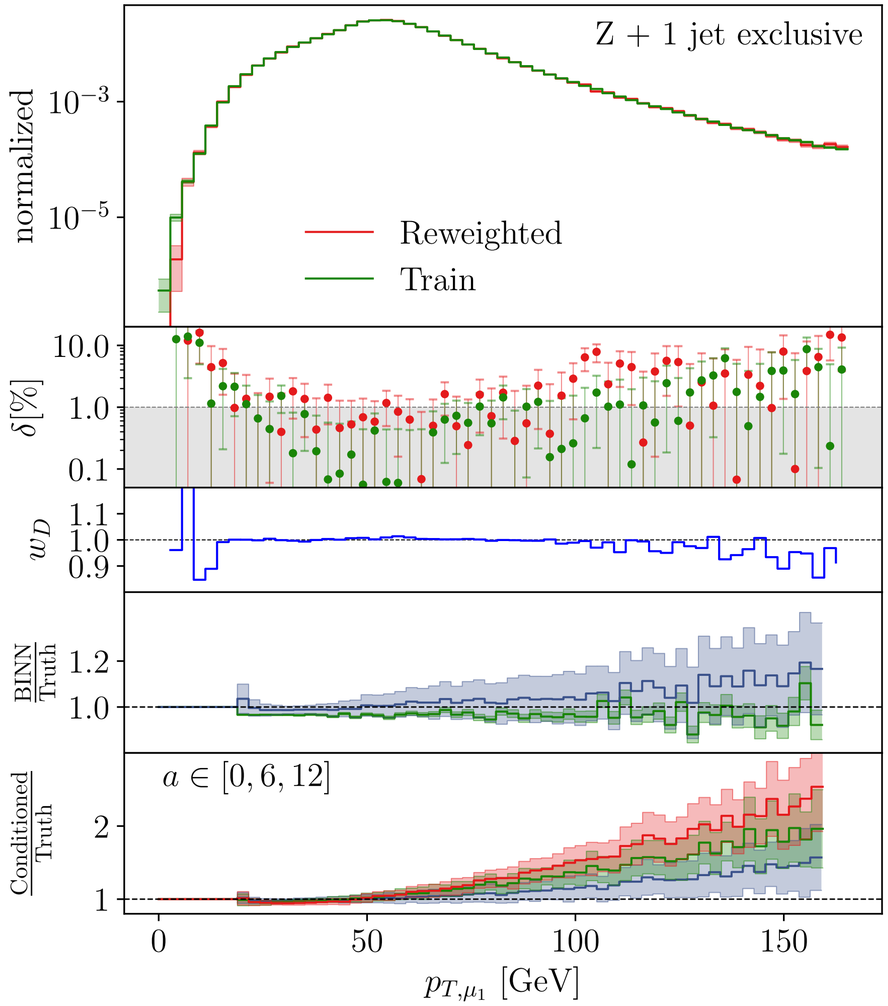
Topological holes



- Control via classifier D
- $$\frac{p_{\text{truth}}(x)}{p_{\text{INN}}(x)} = \frac{D(x)}{1 - D(x)}$$
- Precision via reweighting
  - Correct deviations of p_{INN}

Putting flows to work

Event generation



- Basis: INN
- Control via classifier D
  - $$\frac{p_{\text{truth}}(x)}{p_{\text{INN}}(x)} = \frac{D(x)}{1 - D(x)}$$
- Precision via reweighting
  - Correct deviations of p_{INN}

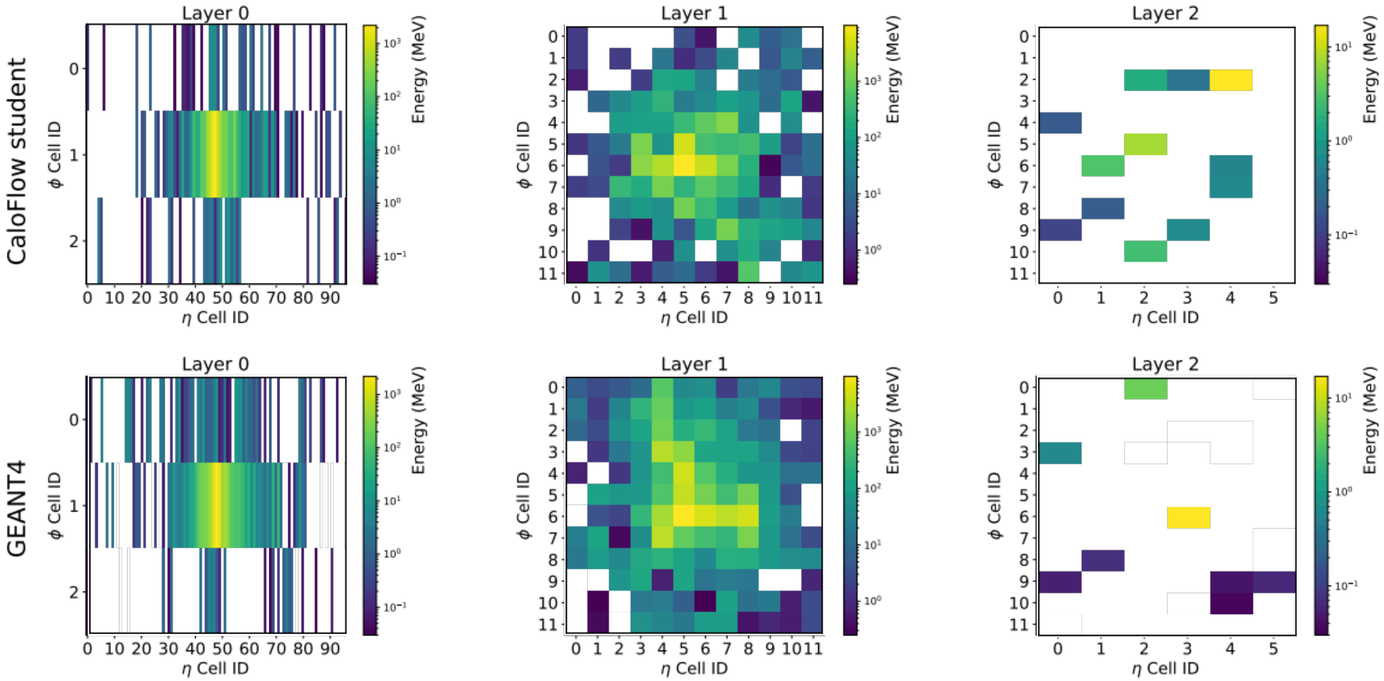
➡ Uncertainty estimation via Bayesian NN

➡ Uncertainty propagation via conditioning

Putting flows to work

Detector simulation

Challenge: large dimensionality $(3 \times 96, 12 \times 12, 12 \times 6)$



C. Krause & D. Shih [2110.11377]

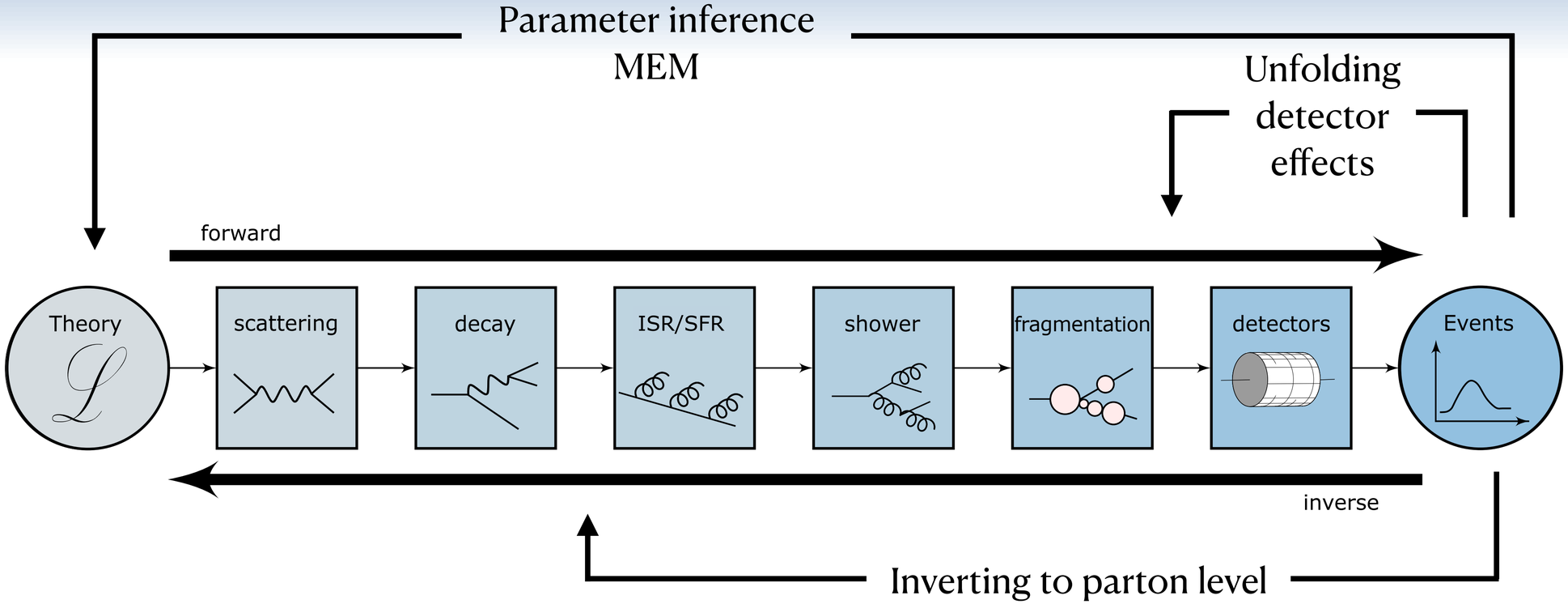
π^+ shower individual & average

III. Unfolding or inverse problems

Mixing

Unfolding

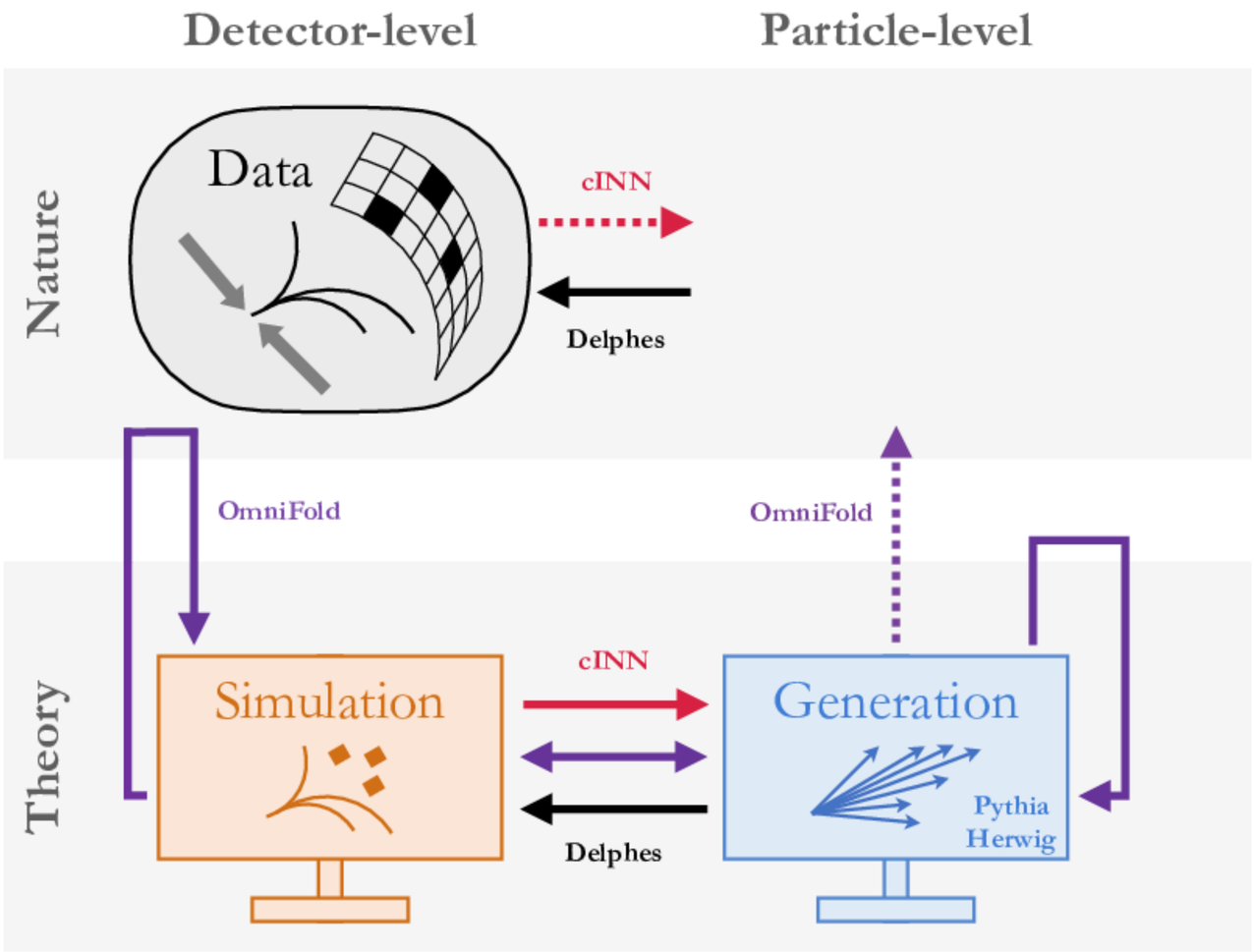
Inverting the simulation chain



Requirements

- ☐ Highdimensional
- ☐ Bin independent
- ☐ Statistically well defined

ML unfolding methods



Classifier based aproach
Output: reweighted distribution of MC events

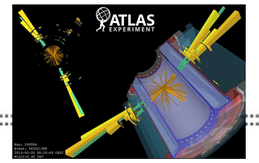
Density based approach
Output: probability density per unfolded event

VAE alternative: OTUS by J. N. Howard et al.

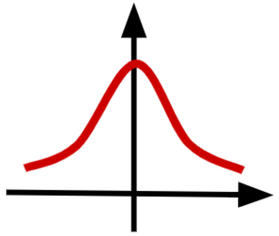
M. Arratia et al. [2109.13243]

cINN unfolding

Given a reconstructed event:
What is the probability distribution at particle level?

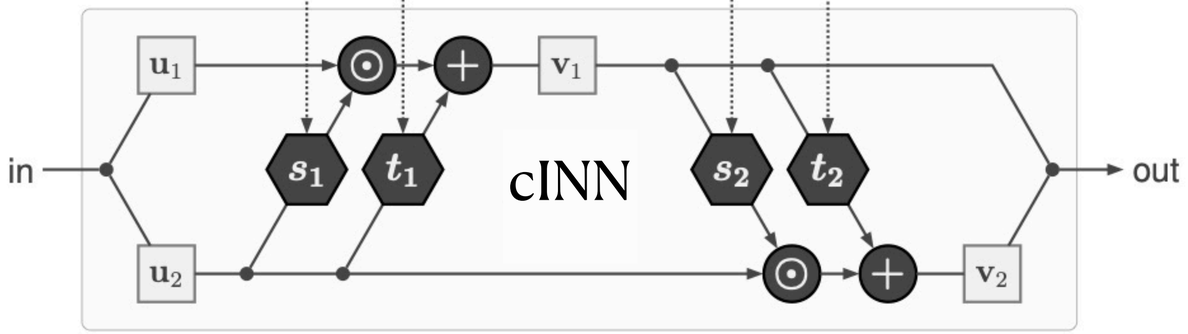


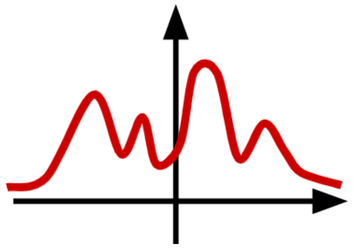
← Reconstructed objects



Normal distribution

$z \sim \mathcal{N}$





Probability distribution

$x \sim P_{\text{parton}}$

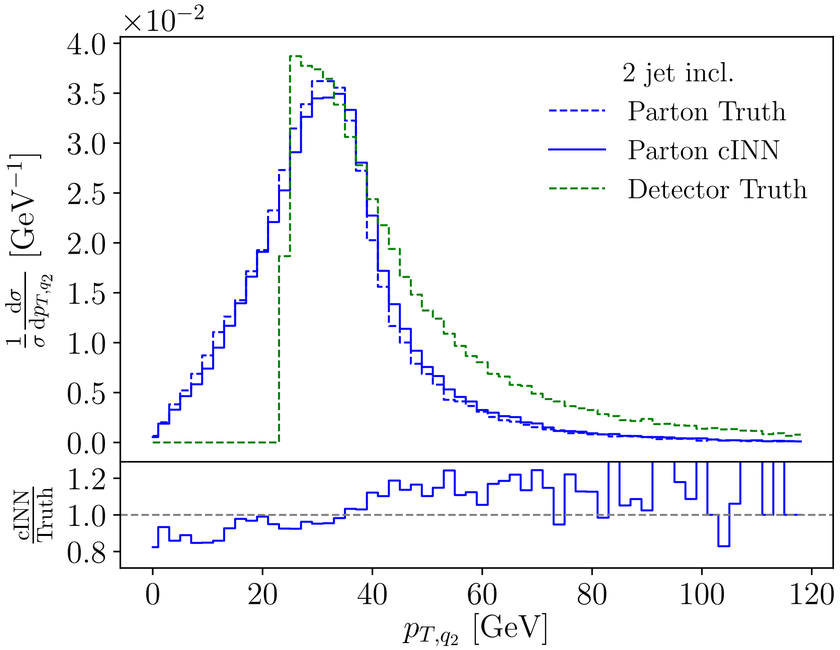
Training

Unfolding

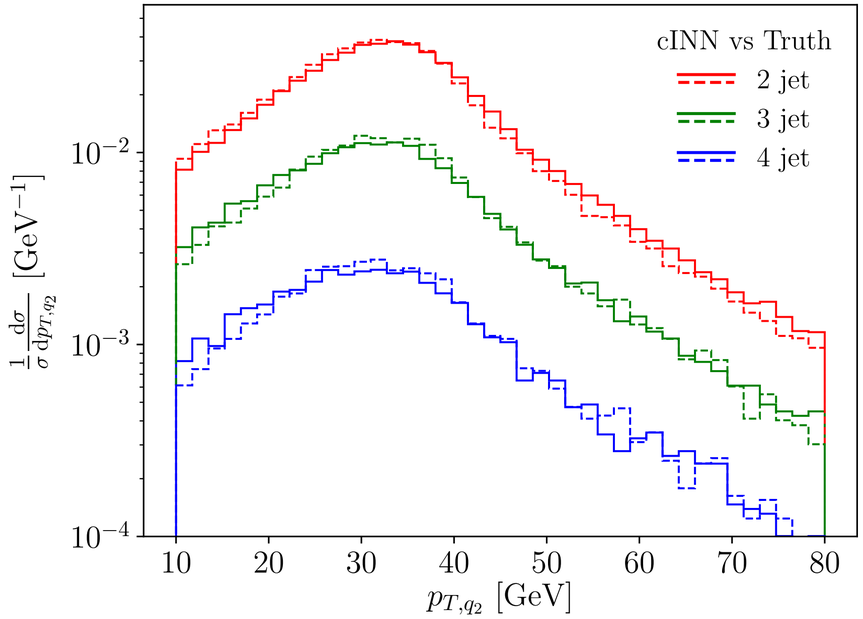
Inverting inclusive distributions

$pp > WZ > q\bar{q}l^+l^-$ + ISR → 2/3/4 jet events

Training on inclusive dataset



Evaluate exclusive 2/3/4 jet events

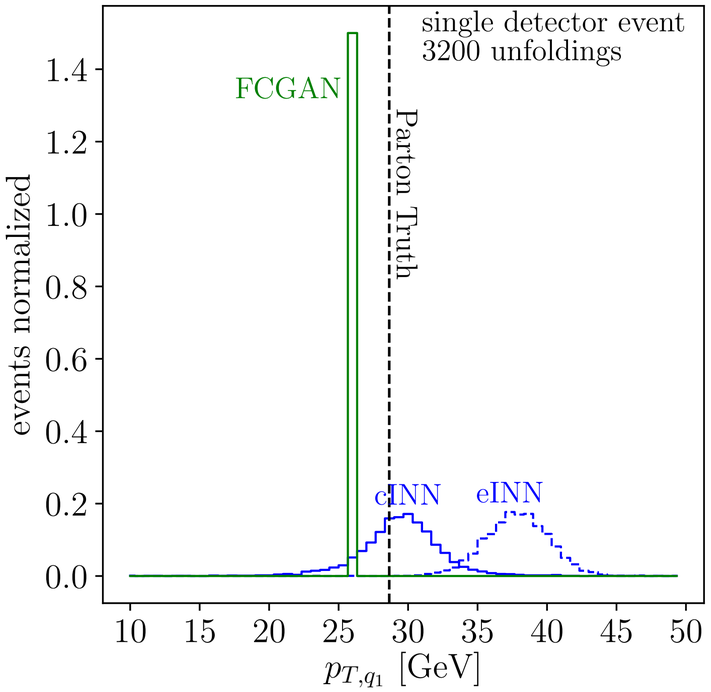


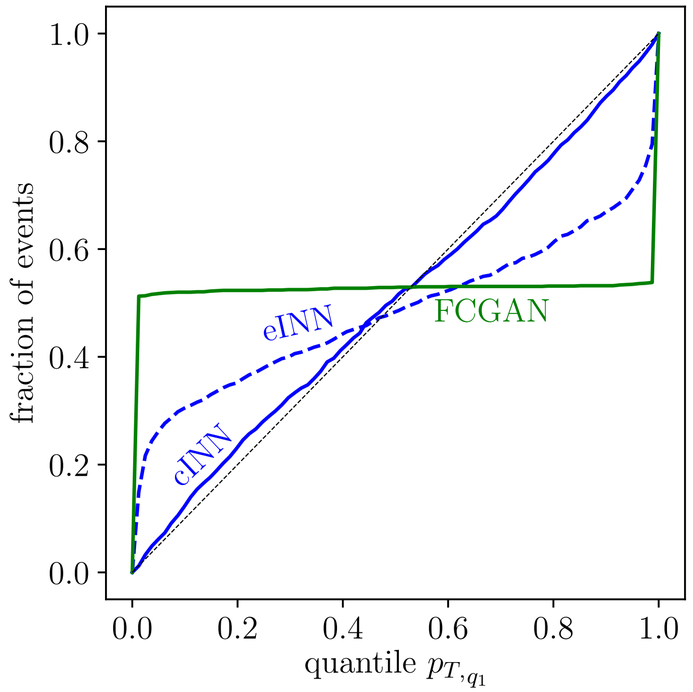
- ☑ High-dimensional
- ☑ Bin-independent
- ☐ Statistically well defined ?

M. Bellagente et al. [2006.06685]

Event-wise unfolding

No deterministic mapping!
Check calibration of probability density for individual event unfolding





- ☑ High-dimensional
- ☑ Bin-independent
- ☑ Statistically well defined

M. Bellagente et al. [2006.06685]

IV. Ideas for uncertainty estimation

?

ML Uncertainties

When do we (not) need them?

- Analyses with poorly trained NNs are sub-***optimal*** but not ***wrong***

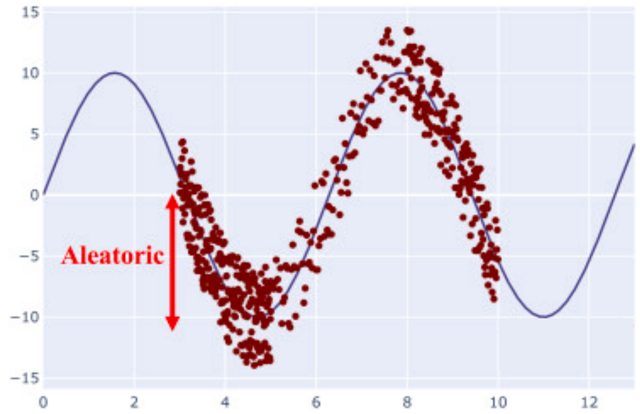
- Example 1: Enhance **Signal vs Background** with NN
  - Use NN output as observable
  - Poor NN yields **low S vs B**
  - Does not prevent correct statistical analysis

- Example 2: INN for **integration**
  - Sub-optimal contour deformation
  - **High variance** of integral
  - Not efficient but not wrong

➡ **Control** comes from **simulation !**

More details eg. B. Nachman [1909.03081]

- Example 3: Understanding a calibration output
  - Regression problem with uncertainties



modified from M. Abdar [doi.org/10.1016/j.inffus.2021.05.008]

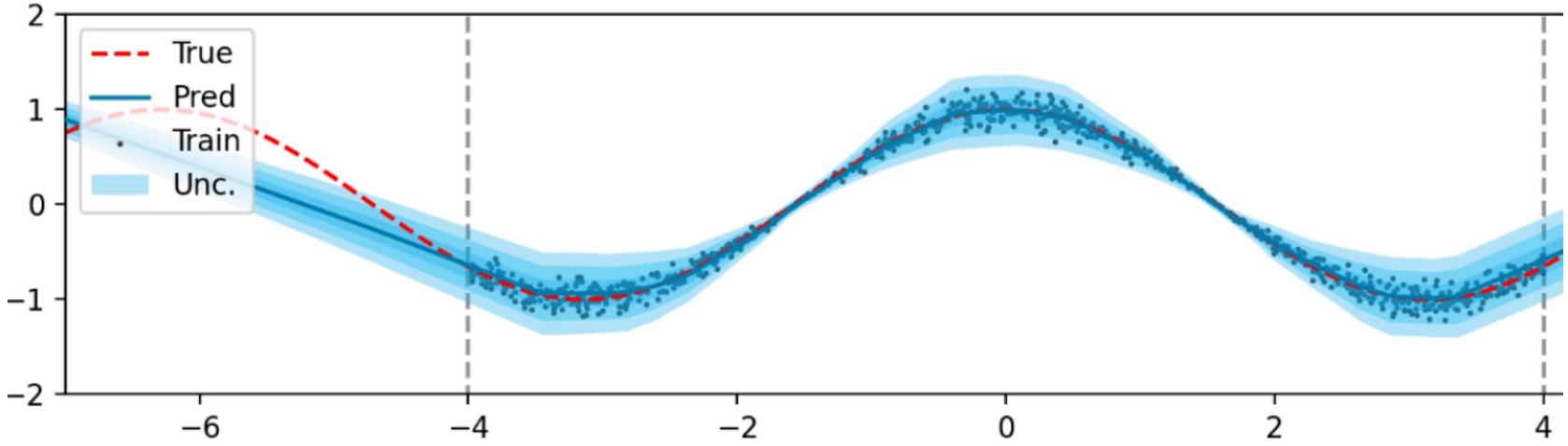
➡ How can we estimate this uncertainty?

Estimating uncertainties in ML

1. **Extend** standard network **output** to include uncertainty

   $\rightarrow (\mu(x), \sigma(x))$

   - Gaussian approximation
   - $\mathscr{L}_{\text{Gauss}} = -\log(\sqrt{2\pi}\sigma(x)) - \frac{1}{2}\frac{(\mu(x) - y)^2}{\sigma(x)^2}$
   - Captures only $\mathbf{p}(\mathbf{y} \mid \mathbf{x}, \mathbf{w})$ for fixed network weights
   - w varies for different trainings!

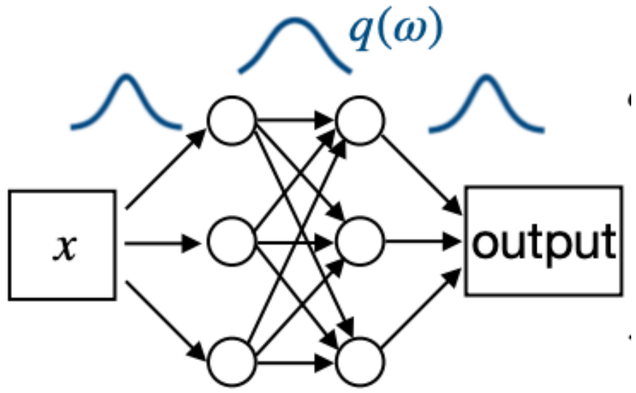


Estimating uncertainties in ML

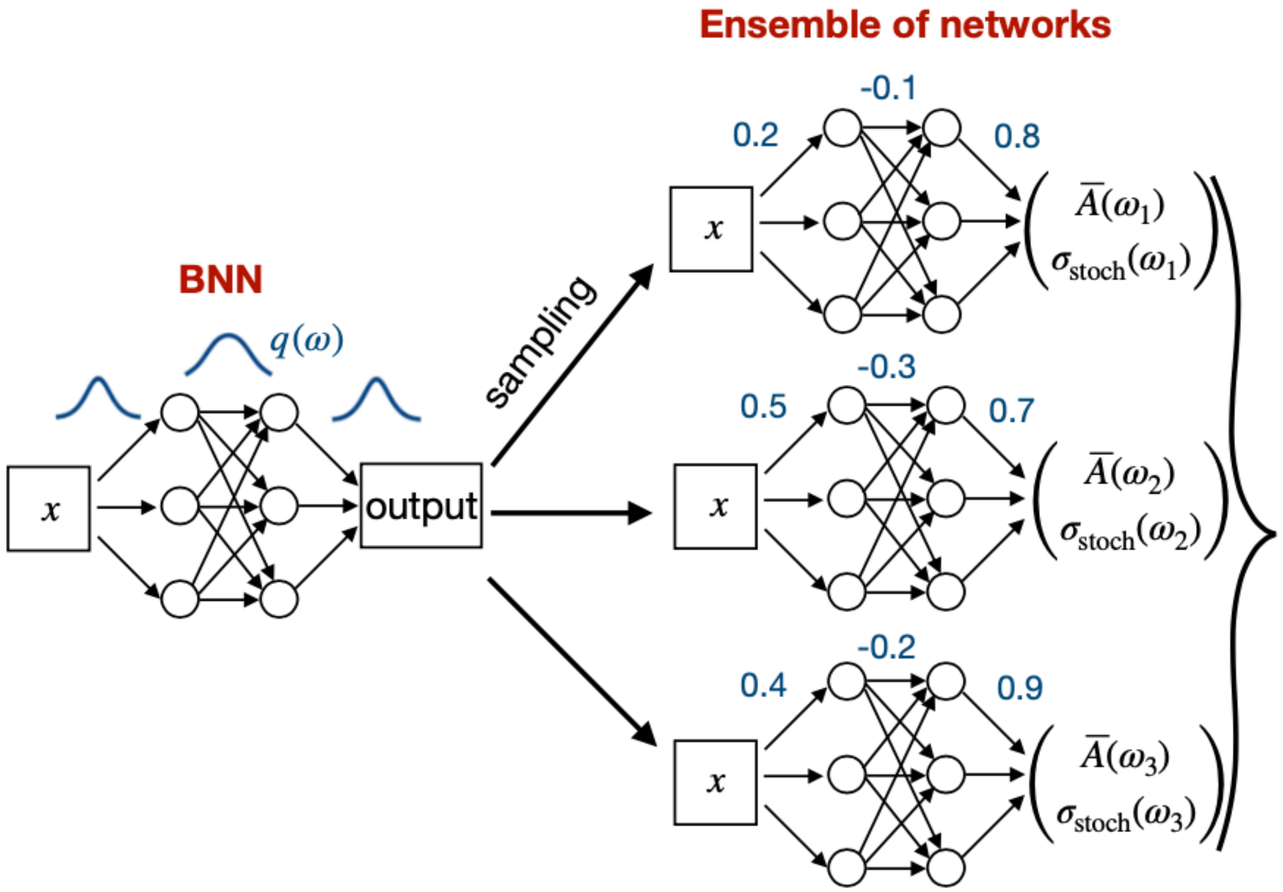
2. Estimating $\mathbf{p(y \mid x, D)}$ with training dataset D

- $p(y \mid x, D) = \int \mathrm{d}w \; p(y \mid x, w) \; p(w \mid D)$

$\mathscr{L}_{\text{Gauss}}$ BNN



Bayesian Neural Network



Output

$$\langle A \rangle = \frac{1}{N} \sum_i^N \bar{A}(\omega_i)$$

$$\sigma^2_{\text{stoch}} = \frac{1}{N} \sum_i^N \sigma^2_{\text{stoch}}(\omega_i)$$

$$\sigma^2_{\text{pred}} = \frac{1}{N} \sum_i^N (\langle A \rangle - \bar{A}(\omega_i))^2$$

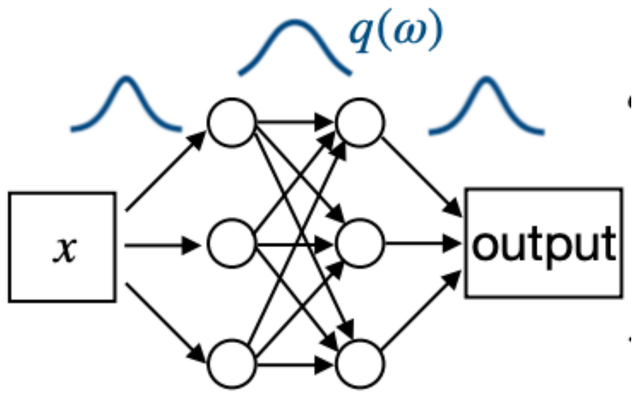
Estimating uncertainties in ML

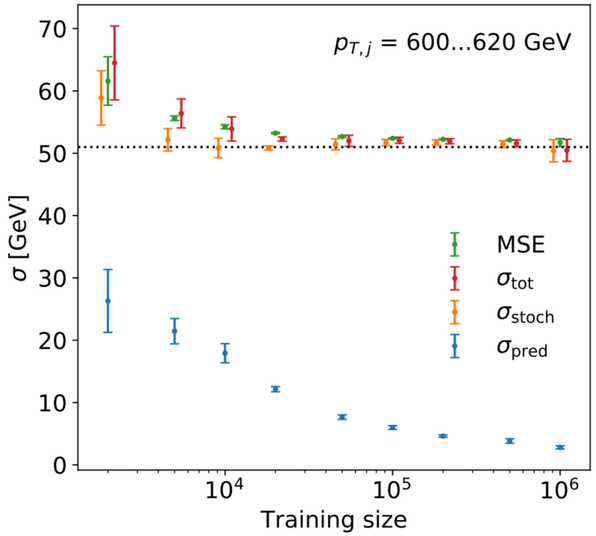
2. Estimating $\mathbf{p(y \mid x, D)}$ with training dataset D

- $p(y \mid x, D) = \int \mathrm{d}w \; p(y \mid x, w) \; p(w \mid D)$

$\mathscr{L}_{\text{Gauss}}$

BNN





G. Kasieczka et al. [2003.11099]

Jet calibration

→ $\sigma(x)$ captures intrinsic uncertainty

For large dataset:

→ $p(w \mid D)$ approaches δ-function

3. **Alternative approaches**: Ensembling, Normalizing flows (calibration curves), ...

Summary - What ML can do for you

1. **Supervised Classification and Regression tasks**
   Particle identification, Calibration, track reconstruction,...

2. **Boosting simulations with generative networks**
   Phase space sampling, Calorimeter simulations,
   Loop calculations, ...

3. **New inference methods**
   Unfolding, optimal observables, likelihood ratio tests

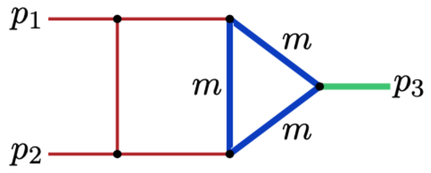
4. **Uncertainty estimation**
   Bayesian networks, ensembles, generative approaches, ...

Enjoy the cocktail dinner !

Multi-loop calculations with INNs

Profiting from the Jacobian

Precision predictions based on loop diagrams



Analytic expression for loop amplitude

$$G = \int_{-\infty}^{\infty} \left(\prod_{l=1}^{L} \frac{\mathrm{d}^D k_l}{i\pi^{\frac{D}{2}}} \right) \prod_{j=1}^{N} \frac{1}{(q_j^2 - m_j^2 + i\delta)^{\nu_j}}$$

$$= \int_0^1 \prod_{j=1}^{N-1} \mathrm{d}x_j \, x_j^{\nu_j - 1} \frac{U^{\nu-(L+1)D/2}}{F^{\nu-LD/2}} = \int_0^1 \prod_{j=1}^{N-1} \mathrm{d}x_j \; I(\vec{x})$$

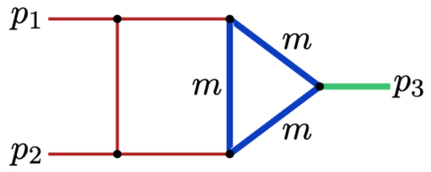
Rewrite with Feynman parameters

Still contains singularities

Multi-loop calculations with INNs

Profiting from the Jacobian

Precision predictions based on loop diagrams



Analytic expression for loop amplitude

$$G = \int_{-\infty}^{\infty} \left(\prod_{l=1}^{L} \frac{\mathrm{d}^D k_l}{i\pi^{\frac{D}{2}}} \right) \prod_{j=1}^{N} \frac{1}{(q_j^2 - m_j^2 + i\delta)^{\nu_j}}$$

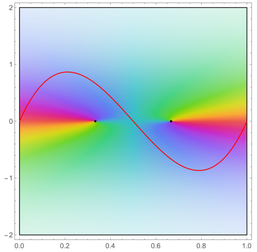
$$= \int_0^1 \prod_{j=1}^{N-1} \mathrm{d}x_j \, x_j^{\nu_j - 1} \frac{U^{\nu-(L+1)D/2}}{F^{\nu-LD/2}} = \int_0^1 \prod_{j=1}^{N-1} \mathrm{d}x_j \; I(\vec{x})$$

Rewrite with Feynman parameters

Still contains singularities

Solved by contour deformation due to Cauchy's theorem

$$\int_0^1 \prod_{j=1}^{N} \mathrm{d}x_j \, I(\vec{x}) = \int_0^1 \prod_{j=1}^{N} \mathrm{d}x_j \; \det\left(\frac{\partial \vec{z}(\vec{x})}{\partial \vec{x}} \right) I(\vec{z}(\vec{x}))$$

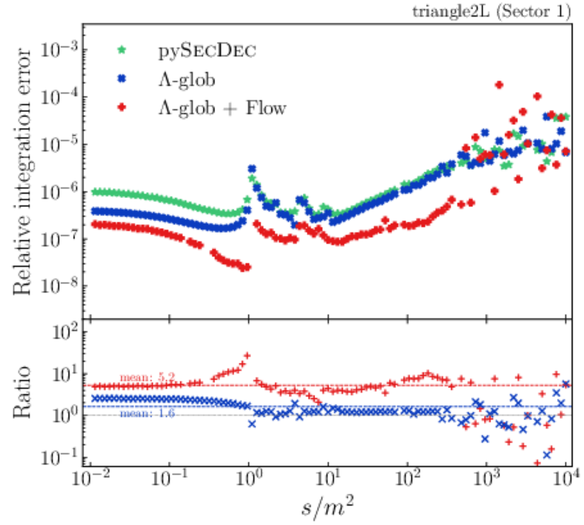


Optimal parametrization = minimal variance

Turn it into an ML Problem

Parametrization $\rightarrow z = \mathrm{INN}(x)$

Variance $\rightarrow \mathscr{L}$



R. Winterhalder, et al. [2112.09145]

1. Contour deformation:
used if multi-scale integral

2. Λ-glob:
optimization of λ_j parameters

$$\int_0^1 \prod_{j=1}^{N} \mathrm{d}y_j \, \mathcal{I}(\vec{y})$$

$y_j \in \mathbb{R}$

Analytic continuation

$z_j = y_j - \mathrm{i}\tau_j$

$$\int_\gamma \prod_{j=1}^{N} \mathrm{d}z_j \, \mathcal{I}(\vec{z})$$

$z_j \in \mathbb{C}$

$\lambda_j = \lambda_{\text{opt}}$

$\tau_j = \lambda_j \, y_j(1-y_j) \dfrac{\partial F}{\partial y_j}$

$$\int_0^1 \prod_{j=1}^{N} \mathrm{d}y_j \, \det\left(\frac{\partial \vec{z}(\vec{y})}{\partial \vec{y}}\right) \mathcal{I}(\vec{z}(\vec{y}))$$

$y_j \in \mathbb{R}$

3. Normalizing flow:
remapping of reals

$z_j = y_j(x)$

$\lambda_j = 0$

$\tau_j = 0$

$y_j \equiv y_j(x)$

$$\int_0^1 \prod_{j=1}^{N} \mathrm{d}x_j \, \det\left(\frac{\partial \vec{y}(\vec{x})}{\partial \vec{x}}\right) \mathcal{I}(\vec{y}(\vec{x}))$$

$x_j \in \mathbb{R}$

$$\int_0^1 \prod_{j=1}^{N} \mathrm{d}x_j \, \det\left(\frac{\partial \vec{z}(\vec{y})}{\partial \vec{y}}\right) \det\left(\frac{\partial \vec{y}(\vec{x})}{\partial \vec{x}}\right) \mathcal{I}(\vec{z}(\vec{y}(\vec{x})))$$

$x_j \in \mathbb{R}$