Generative networks for particle physics

Lecture at QU Data Science Basics (Hamburg)
June 2021

Anja Butter

Paradigm shift

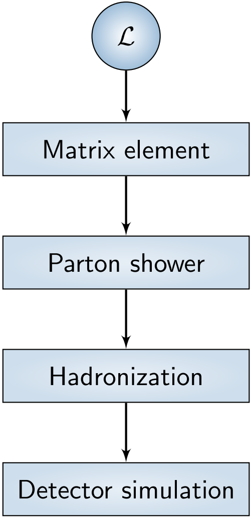
Given data $\rightarrow$ find label

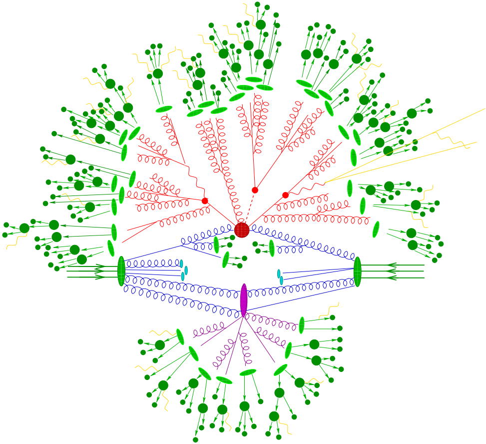
$\downarrow$

Given data $\rightarrow$ generate new data

What do we generate in high energy physics?

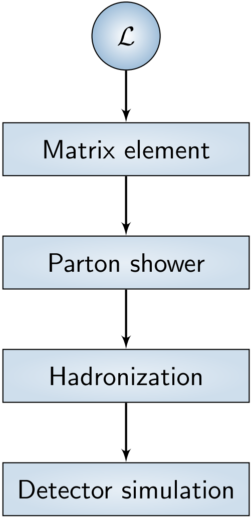
First principle based event generation

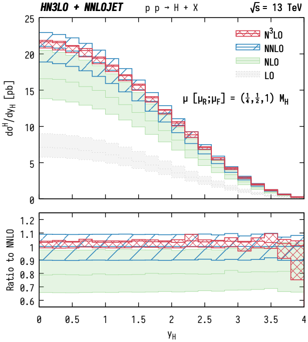




a sherpa artist

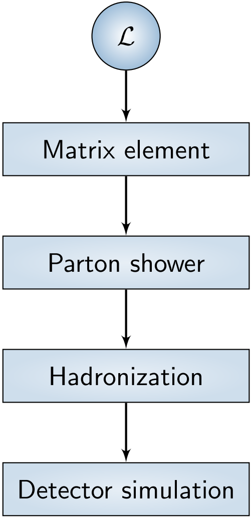
First principle based event generation

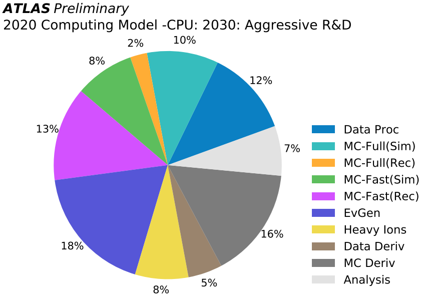




[1807.11501] Cieri, Chen, Gehrmann, Glover, Huss

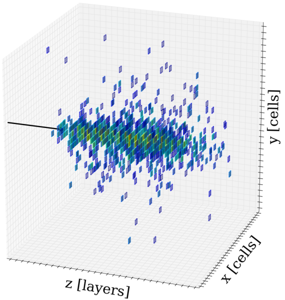
First principle based event generation

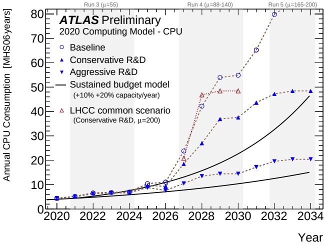




Fast detector simulations

- Important R&D potential
  NN evaluation $\times$100-1000
  faster than GEANT4





- Challenge:
  High-dimensional output
  $\leftarrow$ 30 $\times$ 30 $\times$ 30

Why do we need machine learning for data simulation?

precision
event generation

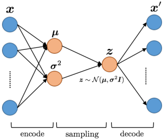
detailed
detector simulation

limited computing ressources

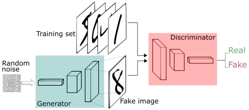
Speed → more events = higher order → Precision

Use NN to speed up simulations!

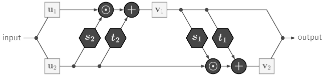
Neural network based generative networks



VAE

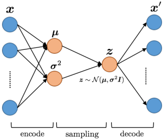


GAN

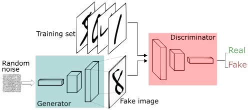


NF

Neural network based generative networks

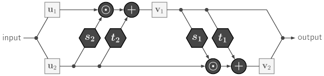


VAE



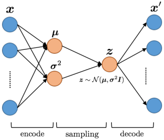
GAN

all kinds of hybrids

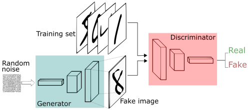


NF

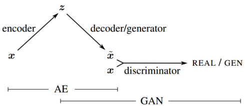
Neural network based generative networks



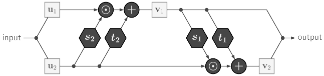
VAE



GAN

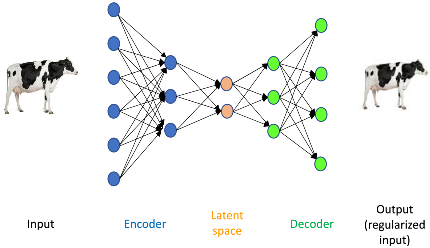


VAE-GAN



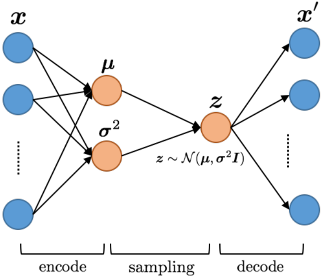
NF

From autoencoders to variational autoencoders



$$\text{AE: } \boldsymbol{x} \xrightarrow{\text{encoder}} \boldsymbol{z} \xrightarrow{\text{decoder}} \boldsymbol{x}' \qquad \mathcal{L}_{AE} = (\boldsymbol{x} - \boldsymbol{x}')^2$$

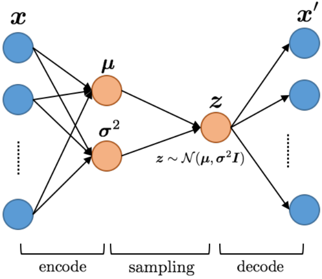
From autoencoders to variational autoencoders



$$\text{AE: } \boldsymbol{x} \xrightarrow{\qquad\qquad\text{encoder}\qquad\qquad} \boldsymbol{z} \xrightarrow{\text{decoder}} \boldsymbol{x}' \qquad \mathcal{L}_{AE} = (\boldsymbol{x} - \boldsymbol{x}')^2$$

$$\text{VAE: } \boldsymbol{x} \xrightarrow{\text{encoder}} \begin{pmatrix} \boldsymbol{\mu} \\ \boldsymbol{\sigma} \end{pmatrix} \xrightarrow[\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})]{\text{sample}} \boldsymbol{z} \xrightarrow{\text{decoder}} \boldsymbol{x}' \qquad \mathcal{L}_{VAE} = \mathcal{L}_{AE} + \mathcal{L}_{lat}$$

From autoencoders to variational autoencoders



$$\text{AE: } \boldsymbol{x} \xrightarrow{\text{encoder}} \boldsymbol{z} \xrightarrow{\text{decoder}} \boldsymbol{x}' \qquad \mathcal{L}_{AE} = (\boldsymbol{x} - \boldsymbol{x}')^2$$

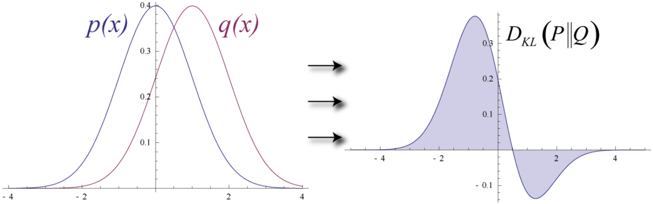
$$\text{VAE: } \boldsymbol{x} \xrightarrow{\text{encoder}} \begin{pmatrix} \boldsymbol{\mu} \\ \boldsymbol{\sigma} \end{pmatrix} \xrightarrow[\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})]{\text{sample}} \boldsymbol{z} \xrightarrow{\text{decoder}} \boldsymbol{x}' \qquad \mathcal{L}_{VAE} = \mathcal{L}_{AE} + \mathcal{L}_{lat}$$

Loss enforces Gaussian latent space

$$\mathcal{L}_{VAE} = \mathcal{L}_{AE} + \beta \cdot \text{KL}(\mathcal{N}(\mu, \sigma) | \mathcal{N}(0, 1)) \qquad \leftarrow \text{similarity measure}$$

Interlude: KL Divergence

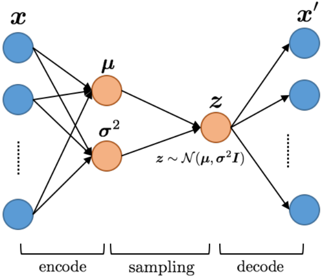
- Distance measure for probability distributions P and Q
- $D_{\mathrm{KL}}(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right)\, dx$



By Mundhenk at English Wikipedia, CC BY-SA 3.0

$$D_{\mathrm{KL}}(\mathcal{N}(\mu, \sigma) \parallel \mathcal{N}(0, 1)) = \frac{1}{2}(1 + \log(\sigma^2) - \mu^2 - \sigma^2)$$

From autoencoders to variational autoencoders



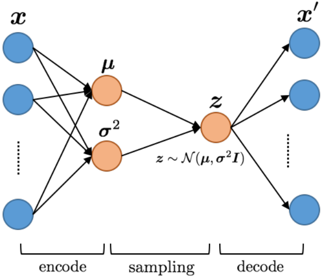
$$\text{AE: } \boldsymbol{x} \xrightarrow{\quad\quad\quad\text{encoder}\quad\quad\quad} \boldsymbol{z} \xrightarrow{\text{decoder}} \boldsymbol{x}' \qquad \mathcal{L}_{AE} = (\boldsymbol{x} - \boldsymbol{x}')^2$$

$$\text{VAE: } \boldsymbol{x} \xrightarrow{\text{encoder}} \begin{pmatrix} \boldsymbol{\mu} \\ \boldsymbol{\sigma} \end{pmatrix} \xrightarrow[\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})]{\text{sample}} \boldsymbol{z} \xrightarrow{\text{decoder}} \boldsymbol{x}' \qquad \mathcal{L}_{VAE} = \mathcal{L}_{AE} + \mathcal{L}_{lat}$$

Loss enforces Gaussian latent space

$$\mathcal{L}_{VAE} = \mathcal{L}_{AE} + \beta \cdot \mathrm{KL}(\mathcal{N}(\mu, \sigma) | \mathcal{N}(0, 1)) \qquad \leftarrow \text{similarity measure}$$

From autoencoders to variational autoencoders



$$\text{AE: } \boldsymbol{x} \xrightarrow{\qquad\qquad\text{encoder}\qquad\qquad} \boldsymbol{z} \xrightarrow{\text{decoder}} \boldsymbol{x}' \qquad \mathcal{L}_{AE} = (\boldsymbol{x} - \boldsymbol{x}')^2$$

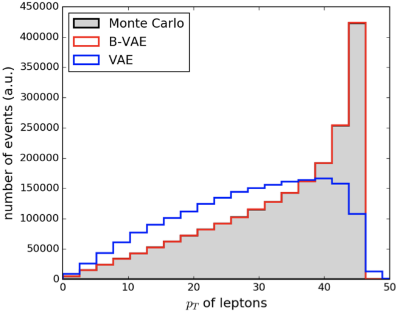
$$\text{VAE: } \boldsymbol{x} \xrightarrow{\text{encoder}} \begin{pmatrix} \boldsymbol{\mu} \\ \boldsymbol{\sigma} \end{pmatrix} \xrightarrow[\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})]{\text{sample}} \boldsymbol{z} \xrightarrow{\text{decoder}} \boldsymbol{x}' \qquad \mathcal{L}_{VAE} = \mathcal{L}_{AE} + \mathcal{L}_{lat}$$

Loss enforces Gaussian latent space

$$\begin{aligned} \mathcal{L}_{VAE} &= \mathcal{L}_{AE} + \beta \cdot \mathrm{KL}(\mathcal{N}(\mu, \sigma) | \mathcal{N}(0, 1)) \qquad \leftarrow \text{similarity measure} \\ &= \mathcal{L}_{AE} + \frac{\beta}{2} \sum_j 1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2 \end{aligned}$$

$e^+e^- \to Z \to l^+l^-$ with VAE

[1901.00875] S. Otten et al.

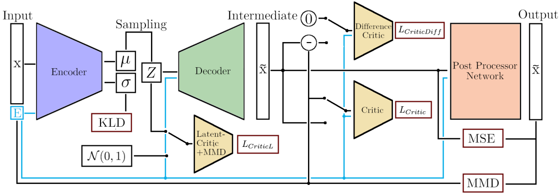


naive VAE fails to reproduce distributions

Why? $\to$ latent space not perfectly Gaussian
Fix: insert information buffer to sample from real latent distribution
$\to$ B-VAE shows excellent performance

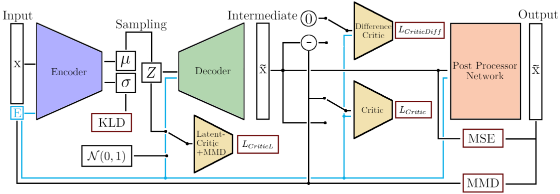
Detector simulation with BIB-AE PP

Bounded-Information-Bottleneck autoencoder with post processing



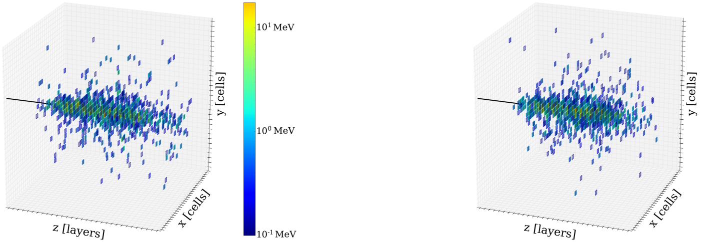
Detector simulation with BIB-AE PP

Bounded-Information-Bottleneck autoencoder with post processing

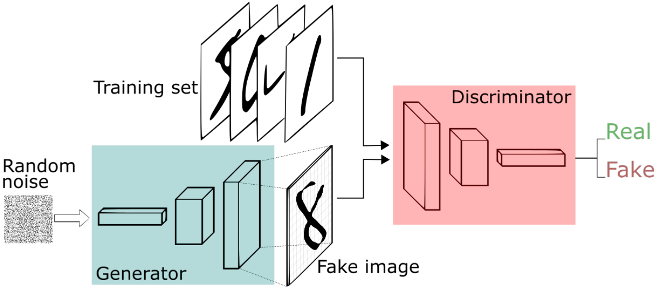


GEANT4

Simulation



Generative Adversarial Networks



Discriminator

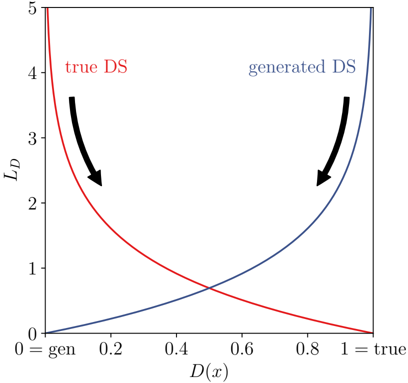
$$L_D = \langle -\log D(x) \rangle_{x \sim P_{Truth}} + \langle -\log(1 - D(x)) \rangle_{x \sim P_{Gen}}$$

Generator

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

Training the Discriminator

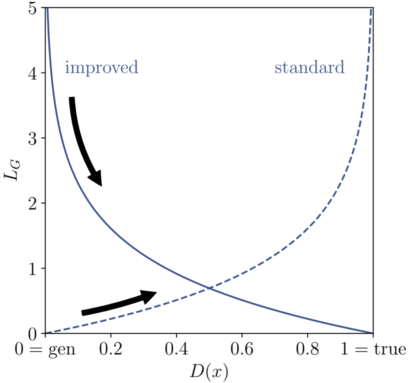
Discriminator loss



Minimize $\quad L_D = \big\langle -\log D(x) \big\rangle_{x\sim P_T} + \big\langle -\log(1 - D(x)) \big\rangle_{x\sim P_G}$

Training the Generator

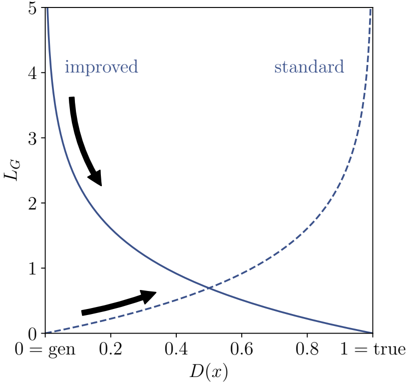
Generator loss



Maximize $\quad L_G = \big\langle -\log(1 - D(x)) \big\rangle_{x \sim P_G}$

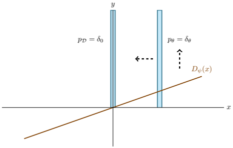
Training the Generator

Generator loss

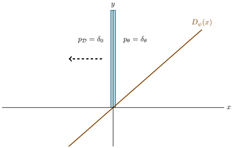


Minimize $\quad L_G = \big\langle -\log D(x) \big\rangle_{x\sim P_G}$

Regularization



(a) $t = t_0$



(b) $t = t_1$

[1801.04406]

Adding gradient penalty

$$\phi(x) = \log \frac{D(x)}{1 - D(x)} \qquad \Rightarrow \qquad \frac{\partial \phi}{\partial x} = \frac{1}{D(x)} \frac{1}{1 - D(x)} \frac{\partial D}{\partial x}$$

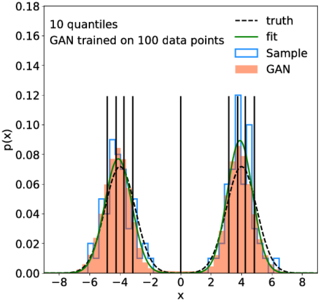
$$L_D \to L_D + \lambda_D \Big\langle \left(1 - D(x)\right)^2 |\nabla \phi|^2 \Big\rangle_{x \sim P_T} + \lambda_D \Big\langle D(x)^2 |\nabla \phi|^2 \Big\rangle_{x \sim P_G}$$

What is the statistical value of GANned events?[2008.06545]

- Camel function
- Sample vs. GAN vs. 5 param.-fit

Evaluation on quantiles:

$$\mathrm{MSE}^* = \sum_{j=1}^{N_\mathrm{quant}} \left(p_j - \frac{1}{N_\mathrm{quant}} \right)^2$$

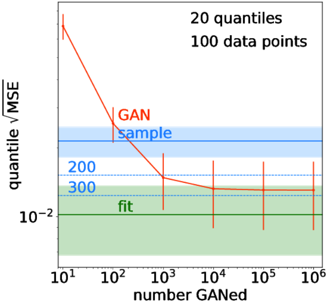


What is the statistical value of GANned events? [2008.06545]

- Camel function
- Sample vs. GAN vs. 5 param.-fit

Evaluation on quantiles:

$$\mathrm{MSE}^* = \sum_{j=1}^{N_\mathrm{quant}} \left(p_j - \frac{1}{N_\mathrm{quant}} \right)^2$$



→ Amplification factor 2.5

$\boxed{\text{Sparser data} \rightarrow \text{bigger amplification}}$

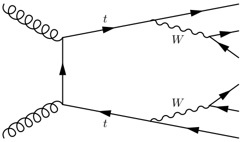
How to GAN LHC events

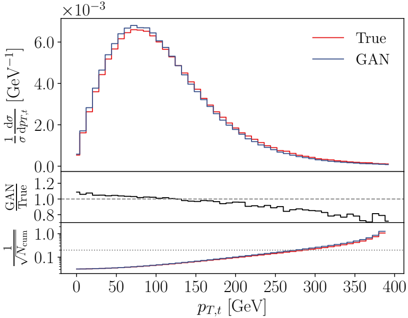
[1907.03764]

- $t\bar{t} \to 6$ quarks
- 18 dim output
  - external masses fixed
  - no momentum conservation

+ Flat observables ✓

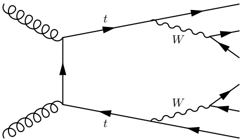
– Systematic undershoot in tails



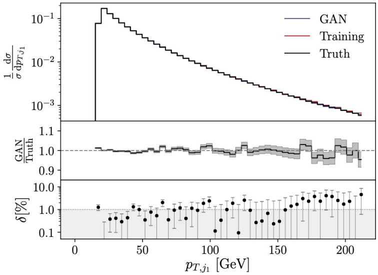


How to GAN LHC events

[1907.03764]



- $t\bar{t} \to 6$ quarks
- 18 dim output
  - external masses fixed
  - no momentum conservation
- + Flat observables ✓
- – Systematic undershoot in tails
  $\to$ improve network (symmetries, preprocessing, ...) ✓



unpublished

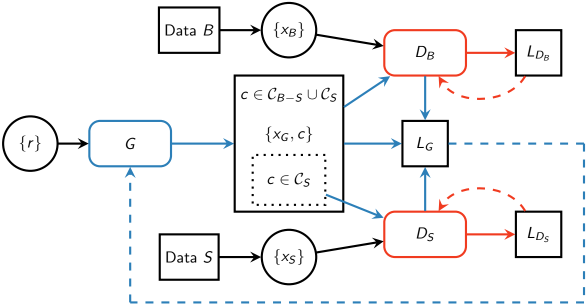
Generating the high-dim. difference of distributions

[1912.08824]

- Necessary to include negative events
- Beat bin-induced uncertainty

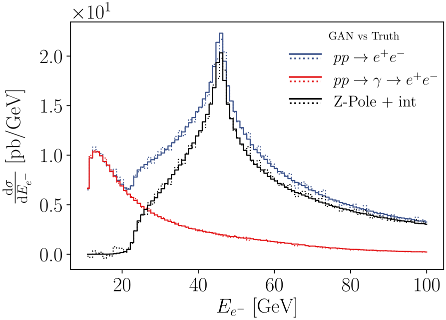
$$\Delta_{B-S} > \max(\Delta_B, \Delta_S)$$

- Applications:
  - Background subtraction, soft-collinear subtraction, ...



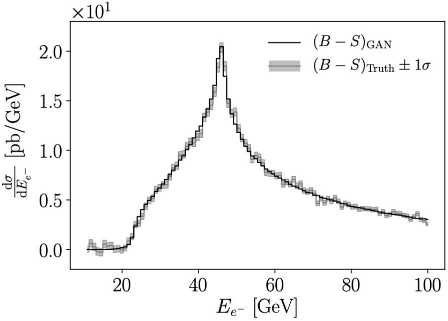
Generative background subtraction

- Training data:
  - $pp \to e^+e^-$
  - $pp \to \gamma \to e^+e^-$
- Generated events: Z-Pole + interference

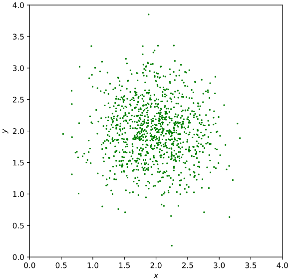


Generative background subtraction

- Training data:
  - $pp \to e^+e^-$
  - $pp \to \gamma \to e^+e^-$
- Generated events: Z-Pole + interference

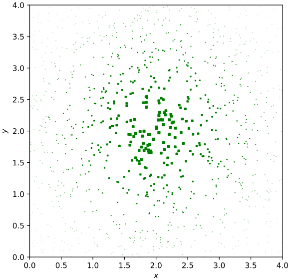


Information in distributions



Information in space distribution
(what we want)

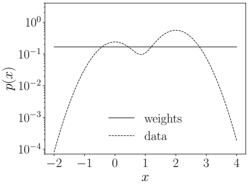
=



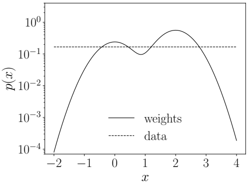
Information in weight
(what we have)

Training on weighted events

Information contained in distribution or event weights

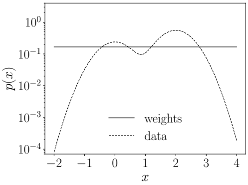


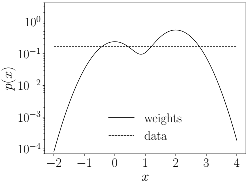
Train on
weighted events



Training on weighted events

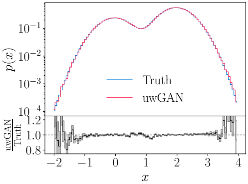
Information contained in distribution or event weights

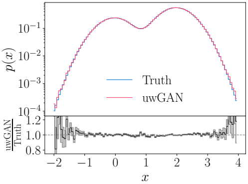




Train on
weighted events

Generate
unweighted events

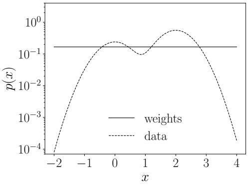


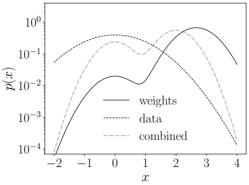


$$L_D = \Big\langle -w \log D(x) \Big\rangle_{x \sim P_{Truth}} + \Big\langle -\log(1 - D(x)) \Big\rangle_{x \sim P_{Gen}}$$

Training on weighted events

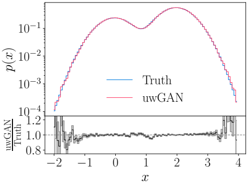
Information contained in distribution or event weights

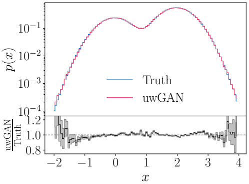




Train on
weighted events

Generate
unweighted events





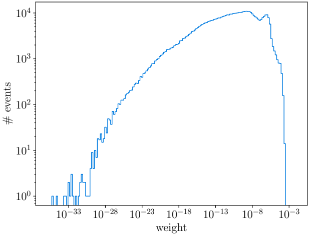
$$L_D = \Big\langle -w \log D(x) \Big\rangle_{x\sim P_{Truth}} + \Big\langle -\log(1-D(x)) \Big\rangle_{x\sim P_{Gen}}$$

The unweighting bottleneck

- High-multiplicity / higher-order $\rightarrow$ large variation of weights
  $\rightarrow$ unweighting efficiencies $< 1\%$

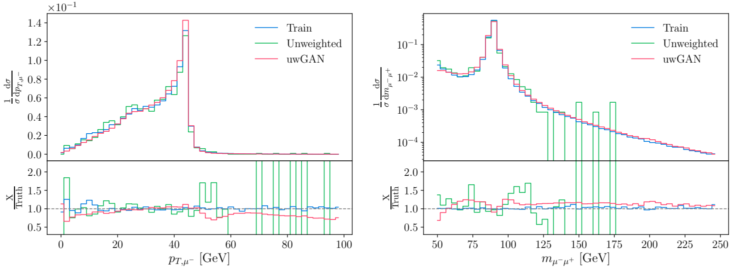
$\rightarrow$ Simulate conditions with naive Monte Carlo generator
ME by `Sherpa`, parton densities from `LHAPDF`, Rambo-on-diet

$pp \rightarrow \mu^+\mu^-$ with $m_{\mu\mu} > 50$ GeV



$\rightarrow$ unweighting efficieny 0.2%

uwGAN results

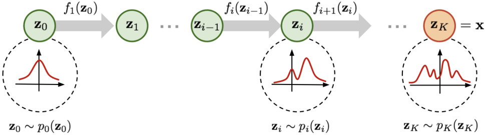


Populates high energy tails

Large amplification wrt. unweighted data!

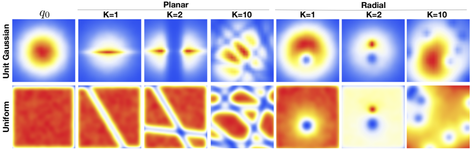
Normalizing flow

Transform input distribution into target distribution via invertible layers

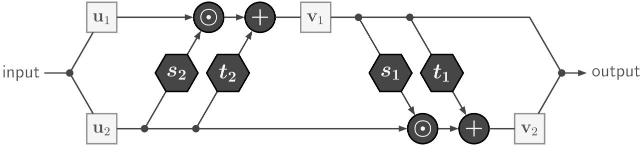


https://lilianweng.github.io/lil-log/2018/10/13/flow-based-deep-generative-models.html

- planar flow: $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{u}h(\boldsymbol{w}^T\boldsymbol{x} + b)$ [2015 Rezende, Mohamed]



Advanced coupling blocs



Features:
+ invertible
+ efficient evaluation in both directions
+ tractable Jacobian $(J(L_2(L_1)) = J(L_2) \cdot J(L_1))$

+ trainable on samples OR probability

Standard event generation in a nutshell

1. Generate phase space points

2. Calculate event weight

$$w_{event} = \boxed{f(x_1, Q^2) f(x_2, Q^2)} \times \boxed{\mathcal{M}(x_1, x_2, p_1, \ldots p_n)} \times \boxed{J(p_i(r))^{-1}}$$

3. Unweighting
$\rightarrow$ keep events if $w_{event}/w_{max} > r \in [0, 1]$
$\rightarrow$ optimal for $w \approx 1$

Standard event generation in a nutshell

Matrix element

$$w_{event} = \boxed{f(x_1, Q^2)f(x_2, Q^2)} \times \boxed{\mathcal{M}(x_1, x_2, p_1, \ldots p_n)} \times \boxed{J(p_i(r))^{-1}}$$

PDF

Phase space mapping

Find phase space mapping p_i such that $w \approx 1$

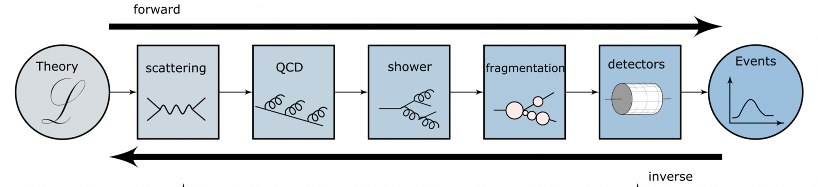
Training on samples

$$\text{events } \boldsymbol{x} \xleftrightarrow[\leftarrow \text{ generating } f^{-1}(\boldsymbol{z})]{\text{training } f(\boldsymbol{x})\rightarrow} \boldsymbol{z} \sim \mathcal{N}$$

How to formulate the loss?
$\rightarrow$ Maximize posterior over network parameters

$$\begin{aligned} L &= -\left\langle \log p(\theta|x) \right\rangle_{x\sim P_x} \\ &= -\left\langle \log p(x|\theta) \right\rangle_{x\sim P_x} - \log p(\theta) + \text{const. (Bayes' theorem)} \\ &= -\left\langle \log p(f(x)) + \log \left| \frac{\partial f(x)}{\partial x} \right| \right\rangle_{x\sim P_x} - \log p(\theta) + \text{const.} \\ &= -\left\langle -f(x)^2 + \log |J| \right\rangle_{x\sim P_x} - \log p(\theta) + \text{const.}\ . \end{aligned}$$

Can we invert the simulation chain?



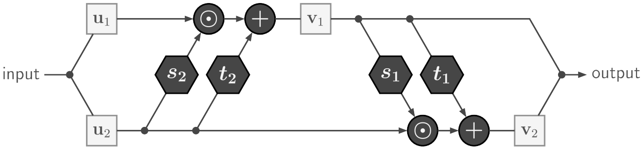
What we
want to know

What we
measure or simulate

wish list: ☐ multi-dimensional
☐ bin independent
☐ statistically well defined

Invertible networks

$$\left(x_{part}\right) \xleftrightarrow[\leftarrow \text{ unfolding:} \bar{g}]{\text{PYTHIA,DELPHES:} g \rightarrow} \left(x_{det}\right)$$



[1808.04730] L. Ardizzone, J. Kruse, S. Wirkert, D. Rahner, E. W. Pellegrini, R. S. Klessen, L. Maier-Hein, C. Rother, U. Köthe

+ Bijective mapping

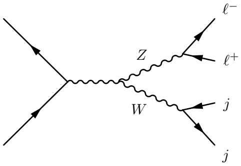
+ Tractable Jacobian

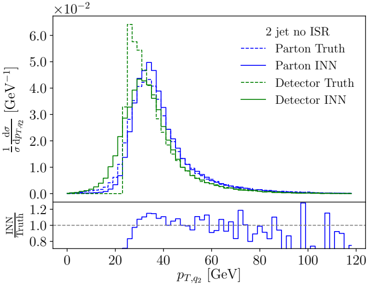
+ Fast evaluation in both directions

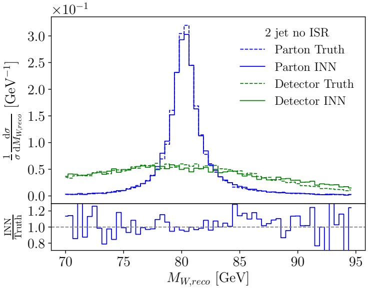
+ Arbitrary networks s and t

Inverting detector effects

- $pp \to ZW \to (ll)(jj)$
- Train: parton $\to$ detector
- Evaluate: parton $\leftarrow$ detector





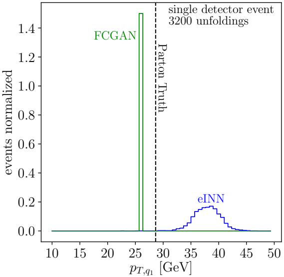


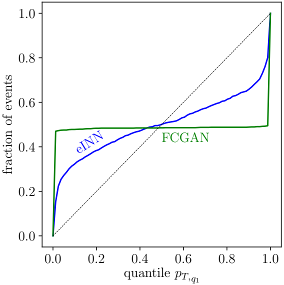
multi-dimensional ✓ bin independent ✓ statistically well defined ?

Including stochastical effects

$$\begin{pmatrix} x_p \\ r_p \end{pmatrix} \xleftrightarrow[\leftarrow \text{ unfolding:}\bar{g}]{\text{PYTHIA,DELPHES:}g\rightarrow} \begin{pmatrix} x_d \\ r_d \end{pmatrix}$$

Sample r_d for fixed detector event
How often is Truth included in distribution quantile?



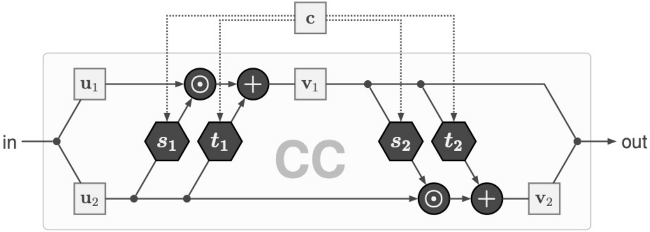


- Problem: arbitrary balance of many loss functions

Taking a different angle

Given an event x_d, what is the probability distribution at parton level?
$\rightarrow$ sample over r, condition on x_d

$$x_p \xleftrightarrow[\leftarrow \text{ unfolding: } \bar{g}(r, f(x_d))]{g(x_p, f(x_d))\rightarrow} r$$



Taking a different angle

Given an event x_d, what is the probability distribution at parton level?
$\rightarrow$ sample over r, condition on x_d

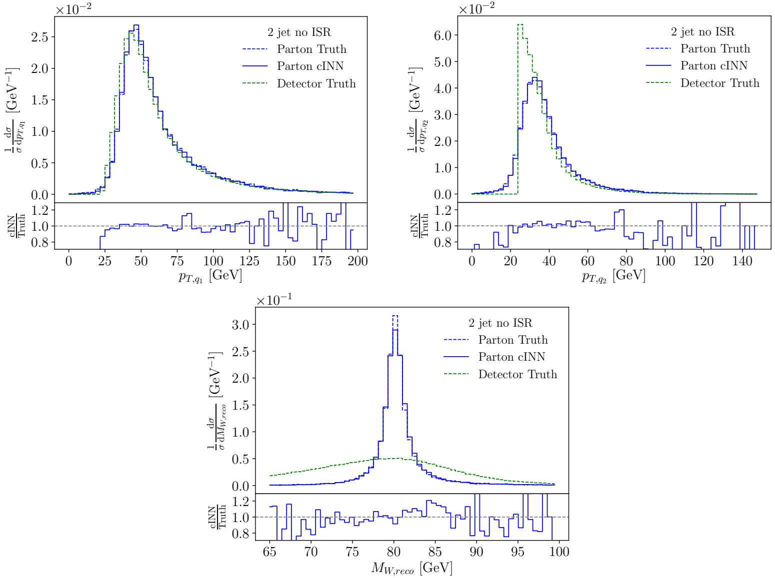
$$x_p \xleftrightarrow[\leftarrow \text{ unfolding: } \bar{g}(r, f(x_d))]{g(x_p, f(x_d)) \rightarrow} r$$

$\rightarrow$ Training: Maximize posterior over model parameters

$$\begin{aligned}
L &= -\left\langle \log p(\theta | x_p, x_d) \right\rangle_{x_p \sim P_p, x_d \sim P_d} \\
&= -\left\langle \log p(x_p | \theta, x_d) \right\rangle_{x_p \sim P_p, x_d \sim P_d} - \log p(\theta) + \text{const.} \quad \leftarrow \text{Bayes} \\
&= -\left\langle \log p(\bar{g}(x_p, x_d)) + \log \left| \frac{\partial \bar{g}(x_p, x_d)}{\partial x_p} \right| \right\rangle - \log p(\theta) \leftarrow \text{change of var} \\
&= \left\langle 0.5 ||\bar{g}(x_p, f(x_d))||_2^2 - \log |J| \right\rangle_{x_p \sim P_p, x_d \sim P_d} - \log p(\theta)
\end{aligned}$$

$\rightarrow$ Jacobian of bijective mapping

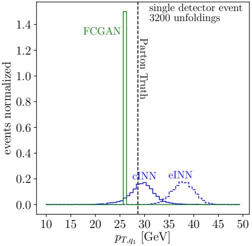
Cross check distributions

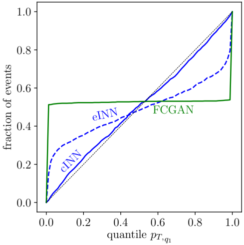


Condition INN on detector data [2006.06685]

$$x_p \underset{\leftarrow \text{ unfolding: } \bar{g}(r, f(x_d))}{\overset{g(x_p, f(x_d)) \rightarrow}{\longleftrightarrow}} r$$

$$\text{Minimizing } L = \left\langle 0.5 ||\bar{g}(x_p, f(x_d))||_2^2 - \log|J| \right\rangle_{x_p \sim P_p, x_d \sim P_d} - \log p(\theta)$$





multi-dimensional ✓ bin independent ✓ statistically well defined ✓

Summary

- Three types of generative models VAE, GAN, NF
- VAE: latent space encoding, KL loss can limit performance
- GAN: based on simple classifier, efficient training if stabilized
- NF: invertible, useable to train directly on probability

Now it's your turn