# Machine learning for particle physicists

## V. Generative networks for particle physics

Anja Butter

26th Vietnam School of Physics

# Recap - Neural networks in particle physics

Different architectures for classification in HEP:

1. CNN for jet classification
2. Graph networks & Particle Net
3. Autoencoders for anomaly detection

# Paradigm shift

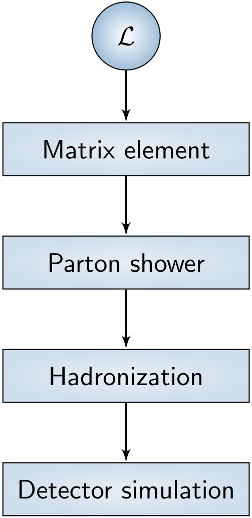
Given data $\rightarrow$ find label

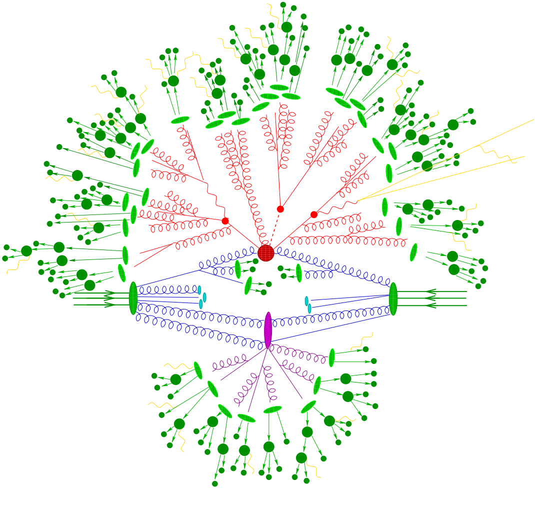
$\downarrow$

Given data $\rightarrow$ generate new data

What do we generate in high energy physics?

# First principle based event generation



a sherpa artist

# Why do we need machine learning for data simulation?

precision
event generation

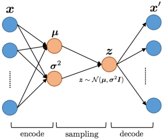
detailed
detector simulation

limited computing ressources

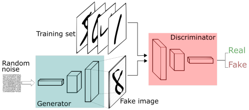
$$\text{Speed} \nearrow \text{more events} = \text{higher order} \searrow \text{Precision}$$

Use NN to try out new approaches!

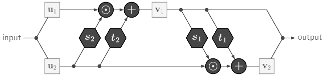
# Neural network based generative networks




VAE

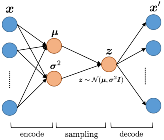



GAN

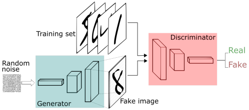



NF

# Neural network based generative networks

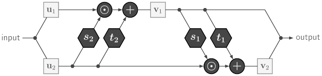



VAE



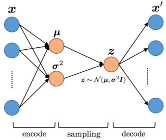

GAN

all kinds of hybrids

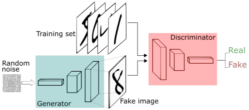



NF

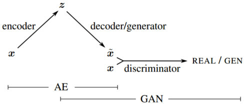
# Neural network based generative networks



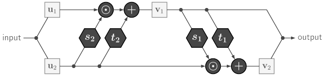

VAE




GAN

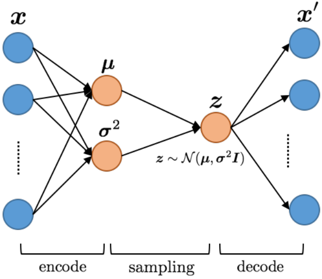



VAE-GAN




NF

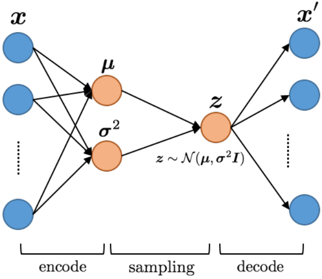
# From autoencoders to variational autoencoders




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

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

# From autoencoders to variational autoencoders




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

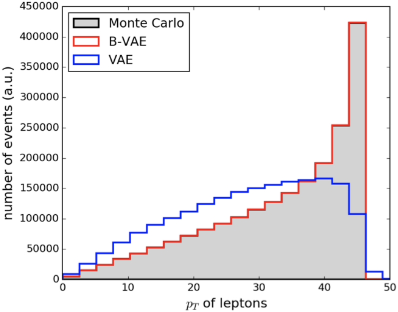
$$\text{VAE: } \boldsymbol{x} \xrightarrow{\text{encoder}} \begin{pmatrix} \boldsymbol{\mu} \\ \boldsymbol{\sigma} \end{pmatrix} \xrightarrow[\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})]{\text{sample}} \boldsymbol{z} \xrightarrow{\text{decoder}} \boldsymbol{x}' \qquad \mathcal{L}_{VAE}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

Loss enforces Gaussian latent space

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

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

[1901.00875] S. Otten et al.



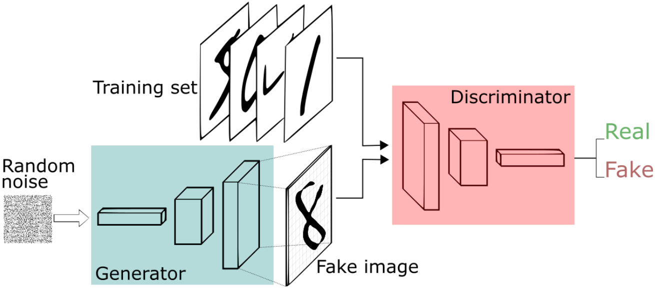

naive VAE fails to reproduce distributions

Why? $\to$ latent space not perfectly Gaussian

Fix: insert information buffer to sample from real latent distribution

$\to$ B-VAE shows excellent performance

# Generative Adversarial Networks




**Discriminator**

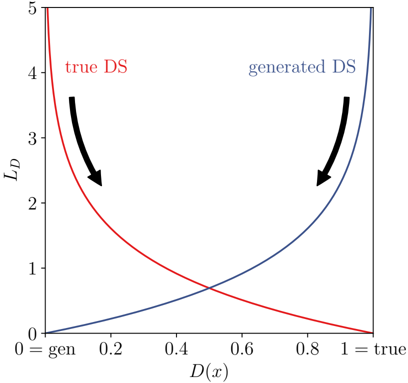
$$L_D = \big\langle -\log D(x) \big\rangle_{x \sim P_{Truth}} + \big\langle -\log(1 - D(x)) \big\rangle_{x \sim P_{Gen}}$$

**Generator**

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

# Training the Discriminator

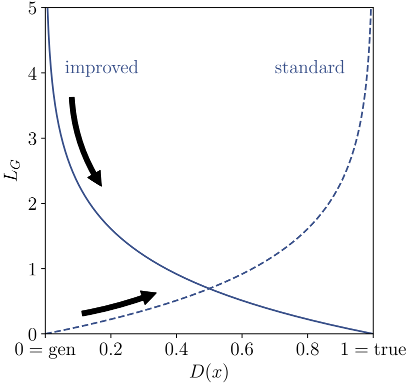
Discriminator loss




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

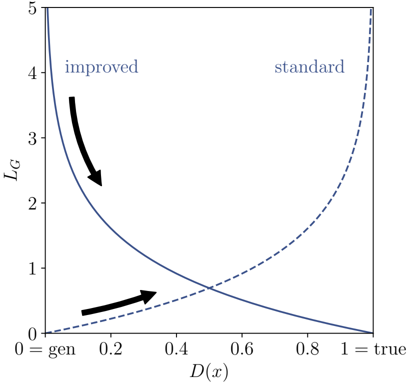
Generator loss




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

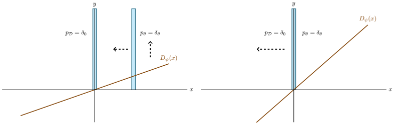
# Training the Generator

Generator loss




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

# Regularization




(a) $t = t_0$ (b) $t = t_1$

[1801.04406]

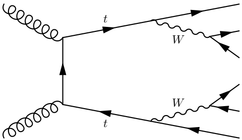
Adding gradient penalty

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

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

# How to GAN LHC events

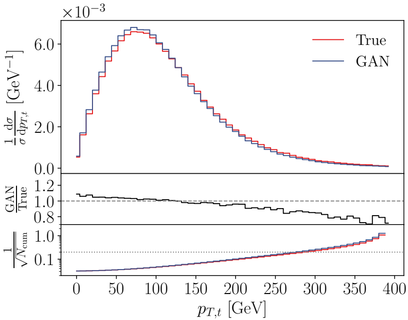
[1907.03764] AB, T. Plehn, R. Winterhalder




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

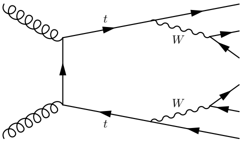
\+ Flat observables ✓

– Systematic undershoot in tails



# How to GAN LHC events

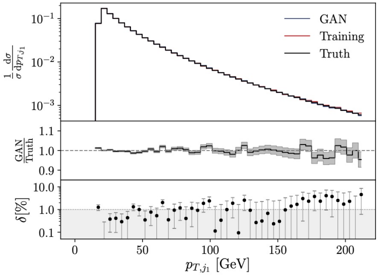
[1907.03764] AB, T. Plehn, R. Winterhalder




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

+ Flat observables ✓

– Systematic undershoot in tails
  $\to$ improve network (3. lecture) ✓



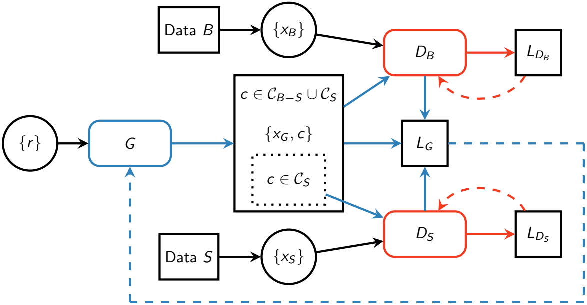
# Generating the high-dim. difference of distributions

[1912.08824] AB, T. Plehn, R. Winterhalder

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

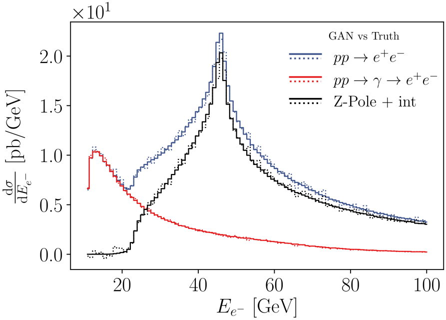
  $\Delta_{B-S} > \max(\Delta_B, \Delta_S)$

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



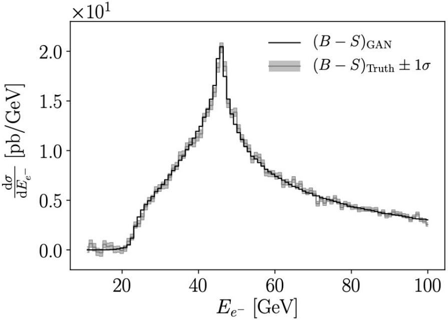
# Generative background subtraction

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



# Generative background subtraction

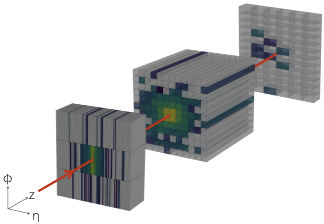
- Training data:
  - $pp \to e^+e^-$
  - $pp \to \gamma \to e^+e^-$
- Generated events: Z-Pole + interference



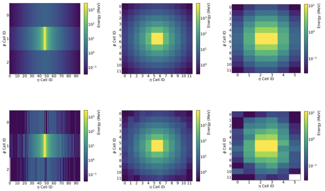
# CaloGAN for detector simulation

[1712.10321] M. Paganini, L. Oliveira, B. Nachman

3D response of an electromagnetic calorimeter for a $e^+$



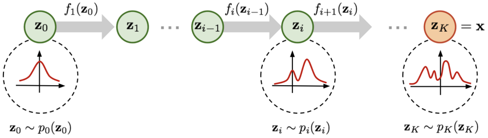

Truth



GAN

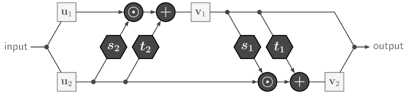
# Normalizing flow

Transform input distribution into target distribution via invertible layers




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

Invertible layer




Features:

$+$ tractable Jacobian

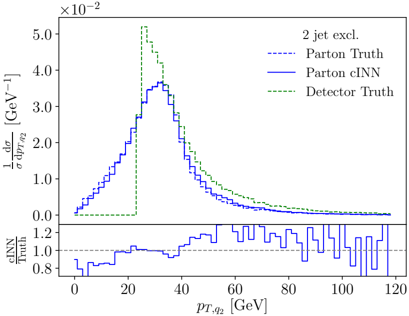
$\rightarrow$ we can compute the density of the target distribution

$+$ invertible

# Inverting a simulation

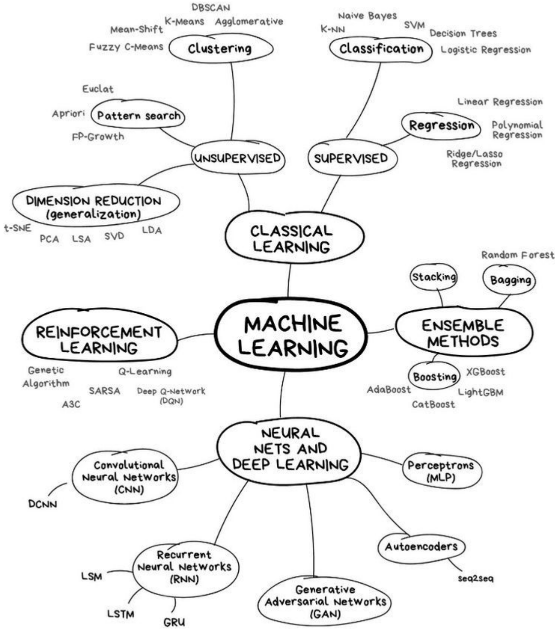
[2006.06685] M. Bellagente et al.

$$\begin{pmatrix} x_p \\ r_p \end{pmatrix} \xleftrightarrow[\leftarrow \text{ unfolding: } g^{-1}]{\textsc{Pythia,Delphes}: \, g \rightarrow} \begin{pmatrix} x_d \\ r_d \end{pmatrix}$$



# Summary

1. Linear models for regression and classification
2. Introduction to neural networks
3. How to train better networks
4. Neural networks for particle physics problems
5. Generative networks for particle physics


MACHINE LEARNING
CLASSICAL LEARNING
UNSUPERVISED
SUPERVISED
Clustering
DBSCAN
K-Means
Agglomerative
Mean-Shift
Fuzzy C-Means
Pattern search
Euclat
Apriori
FP-Growth
DIMENSION REDUCTION (generalization)
t-SNE
PCA
LSA
SVD
LDA
Classification
Naive Bayes
K-NN
SVM
Decision Trees
Logistic Regression
Regression
Linear Regression
Polynomial Regression
Ridge/Lasso Regression
REINFORCEMENT LEARNING
Genetic Algorithm
Q-Learning
SARSA
Deep Q-Network (DQN)
A3C
ENSEMBLE METHODS
Stacking
Bagging
Random Forest
Boosting
XGBoost
AdaBoost
LightGBM
CatBoost
NEURAL NETS AND DEEP LEARNING
Convolutional Neural Networks (CNN)
DCNN
Perceptrons (MLP)
Recurrent Neural Networks (RNN)
LSM
LSTM
GRU
Generative Adversarial Networks (GAN)
Autoencoders
seq2seq

vas3k.com

Now it's your turn ☺