Machine Learning in Particle Physics

— CRC School on Particle Physics Pheno after the Higgs Discovery —

Claudius Krause
Nicole Hartman, Sofia Palacios Schweitzer

Institute of High Energy Physics (HEPHY), Austrian Academy of Sciences (OeAW)

October 2 & 3, 2024

Claudius.Krause@oeaw.ac.at

Some Ressources

If you have questions, please interrupt me and ask!

This lecture is based on:
⇒ "Modern Machine Learning for LHC Physicists",
SS2022 lecture notes of Heidelberg University, arXiv: 2211.01421

Further Reading:

- Summary of HEP-ML papers: "HEPML - Living Review"
  `https://iml-wg.github.io/HEPML-LivingReview/`
- Tipps for efficient training of NNs:
  `https://karpathy.github.io/2019/04/25/recipe/`
- About good coding practices in science: `https://goodresearch.dev/`

Tutorials and Hands-On Session

In the afternoons, we will have

- Wed: 1:15h hands-on session ML ("A Diffusion Model from Scratch")
  https://github.com/SofiaSchweitzer/crc_summer_school/tree/main
- Thu: 1h to finish hands-on and more Q&A

Led by the two ML experts:

Nicole Hartman (ATLAS, TU Munich)

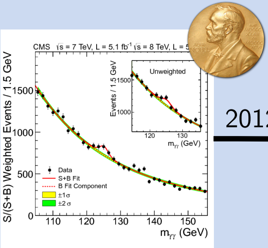
Sofia Palacios Schweitzer (TH, Uni Heidelberg)

Why Machine Learning?

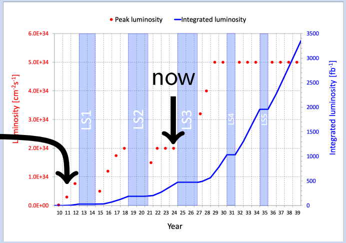
Who has used ML so far?

Why Machine Learning?

Data volume



2012



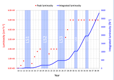
https://lhc-commissioning.web.cern.ch/schedule/HL-LHC-plots.htm

Large amounts of labeled (simulation) and unlabeled (experiment) data.

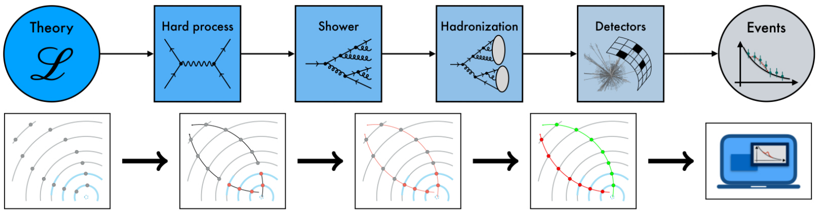
$\Rightarrow$ ML works best with lots of data

Why Machine Learning?

Data volume



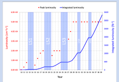
Data complexity



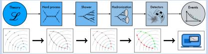
High-dimensional & highly correlated data.
$\Rightarrow$ ML can handle that well

Why Machine Learning?

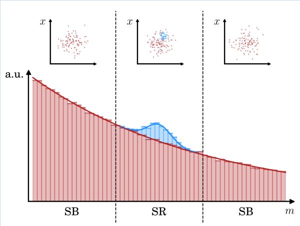
Data volume



Data complexity



Signal detection



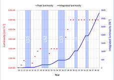
Hallin et al. [2109.00546]

Rare and elusive signals among large backgrounds.

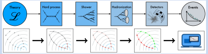
$\Rightarrow$ ML has high sensitivity

Why Machine Learning?

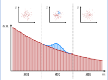
Data volume



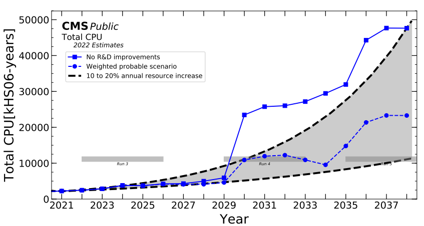
Data complexity



Signal detection



Computing budget



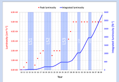
https://twiki.cern.ch/twiki/bin/view/CMSPublic/CMSOfflineComputingResults

Simulation & analysis are computationally expensive.

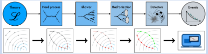
$\Rightarrow$ ML is fast

Why Machine Learning?

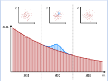
Data volume



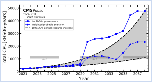
Data complexity



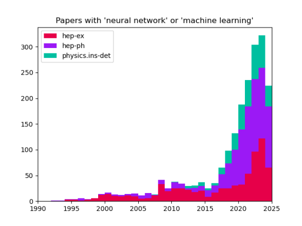
Signal detection



Speed



Increasing interest



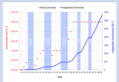
via “The INSPIRE REST API”

We see about 300 papers / year.

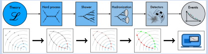
$\Rightarrow$ ML is everywhere

Why Machine Learning?

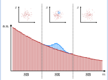
Data volume



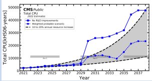
Data complexity



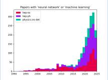
Signal detection



Speed



Interest



ML is fun

via midjourney: "Albert Einstein smiling while having fun coding"

⇒ Like Galileo Galilei looking through the telescope for the first time!

Machine Learning for Particle Physics

This week's plan:

1. Introduction (fits, optimization, and NNs)
2. Regression and Classification
3. Deep Generative Models
4. Anomaly Detection and Data-Driven Methods

What is Machine Learning?

Tom Mitchell, ML Pioneer

"ML ... is the study of algorithms that allow computer programs to automatically improve through experience and by use of data."

1. algorithm: a method to perform a task of interest.
2. experience: training data, which the algorithm can use to learn how to perform a task.
3. improve: a way to measure the performance on the training data.
4. automatically: a strategy to exploit the training data, without external input.

What is Machine Learning?

Tom Mitchell, ML Pioneer

Judea Pearl,
Turing Award Winner

"ML ... is the study of algorithms that allow computer programs to automatically improve through experience and by use of data."

1. algorithm: a method to perform a task of interest.
2. experience: training data, which the algorithm can use to learn how to perform a task.
3. improve: a way to measure the performance on the training data.
4. automatically: a strategy to exploit the training data, without external input.

" Machine Learning is just glorified 'curve fitting' "

What is Machine Learning?

Tom Mitchell, M...

"ML . . . is the study of algorithms that allow computer programs to automatically improve through experience and by use of data."

1. algorithm: a method to perform a task of interest.

...ithm can use

In physics we fit a function of interest to data
in a statistically well-defined way.

... the

4. automatically: a strategy to exploit the training data, without external input.

Judea Pearl,
Turing Award Winner

" Machine Learning is just glorified 'curve fitting' "

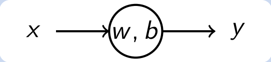
We fit a function of interest to data in a statistically well-defined way.

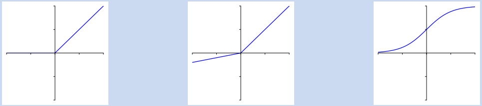
Neural networks are parametric numerical functions $y = f(x;\theta)$ that are inspired by biology.

$$y = \sigma\left(w \cdot x + b\right)$$

σ → non-linear activation

w, b → scalar weight w and scalar bias $b \Rightarrow \theta$

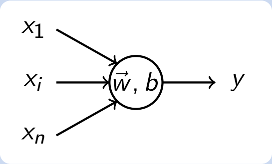




"ReLU" "leaky ReLU" "sigmoid/tanh"

We fit a function of interest to data in a statistically well-defined way.

Neural networks are parametric numerical functions $y = f(x; \theta)$ that are inspired by biology.

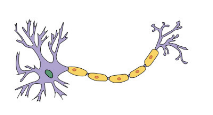


$$y = \sigma\left(\sum_{i=1}^{n} w_i \cdot x_i + b\right)$$

non-linear activation

vector weight $\vec{w}$ and scalar bias $b \Rightarrow \theta$

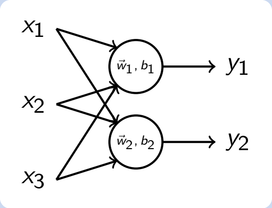
looks like a “real” neuron now:



by Dhp1080 via https://commons.wikimedia.org/w/index.php?curid=4293768

We fit a function of interest to data in a statistically well-defined way.

Neural networks are parametric numerical functions $y = f(x;\theta)$ that are inspired by biology.



$$y_j = \sigma\left(\sum_{i=1}^{n} w_{j,i} \cdot x_i + b_j\right)$$

σ: non-linear, element-wise activation

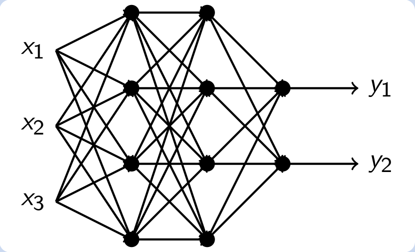
$w_{j,i}$, b_j: matrix weight w and vector bias $\vec{b} \Rightarrow \theta$

$\Rightarrow$ this is called a "layer".

We fit a function of interest to data in a statistically well-defined way.

Neural networks are parametric numerical functions $y = f(x;\theta)$ that are inspired by biology.

We can now put everything together to a Network:




```python
import torch

class DNN(torch.nn.Module):
    """ vanilla NN """
    def __init__(self):
        super(DNN, self).__init__()

        self.inputlayer = torch.nn.Linear(3, 4)
        self.hiddenlayer = torch.nn.Linear(4, 4)
        self.outputlayer = torch.nn.Linear(4, 2)

    def forward(self, x):
        x = torch.nn.LeakyReLU()(self.inputlayer(x))
        x = torch.nn.LeakyReLU()(self.hiddenlayer(x))
        x = self.outputlayer(x)
        return x
```

We fit a function of interest to data in a statistically well-defined way.

- The Loss function $\mathcal{L}(f(x;\theta), y)$ encodes our objective: smaller = better

? There are many different ways to encode the same objective, which one is the best?

- best model at $\theta_{\text{best}} = \text{argmin}_\theta \mathcal{L}(f(x;\theta), y)$
  Which set of θ describes the training data best? $\Rightarrow$ maximize likelihood $p(x_{\text{train}}|\theta)$

$\Rightarrow$ best loss is the negative (log) likelihood: $\mathcal{L} = -\log p(x_{\text{train}}|\theta)$

(We'll get back to this with examples in a few slides.)

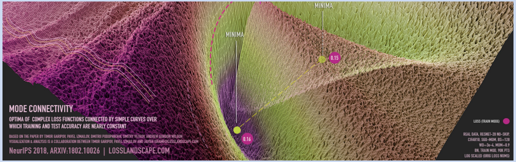
We fit a function of interest to data in a statistically well-defined way.

How do we minimize $\mathcal{L}(f(x;\theta), y)$?

- (stochastic) gradient descent: $\theta_j^{t+1} = \theta_j^t - \alpha \left\langle \frac{\partial \mathcal{L}^t}{\partial \theta_j} \right\rangle$
- backpropagation
- autodifferentiation

} taken care of "under the hood" by `pytorch`/`tensorflow`

```
my_DNN = DNN()
optimizer = torch.optim.Adam(my_DNN.parameters(), lr=1e-3)

for i in range(num_epochs):
    for batch, label in data:

        y = my_DNN(batch)
        loss = loss_func(y, label)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```



The loss landscape can be very complicated. Adaptive optimizers, like ADAM, use momentum to improve convergence.

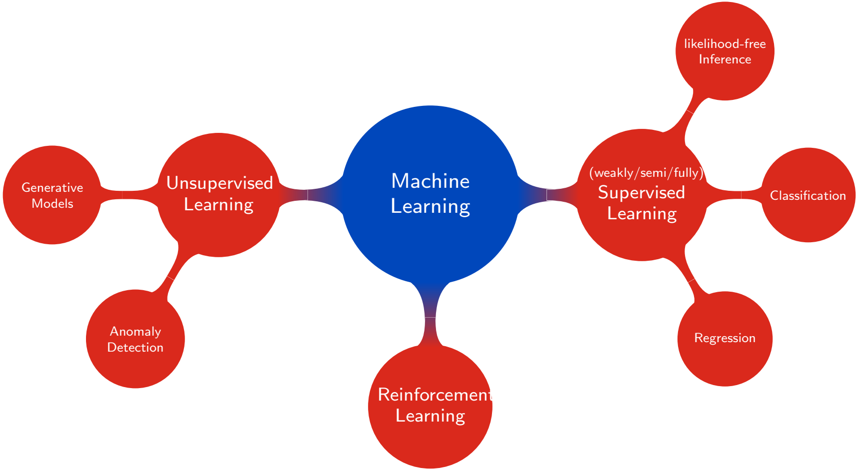
Adam: A Method for Stochastic Optimization [1412.6980]

But: we have to be careful!

- NN can overfit (memorize) training data and stop generalizing!
- to diagnose (and combat): introduce separate validation (for model selection) and test sets.
- to combat: regularize, for example with dropout or L2 norm
- Decreasing the approximation error increases the generalization error: the bias-variance trade-off

```python
class DNN_with_dpo(torch.nn.Module):
    """ vanilla NN with dropout"""
    def __init__(self, dropout_probability=0.):
        super(DNN_with_dpo, self).__init__()

        self.dpo = dropout_probability

        self.inputlayer = torch.nn.Linear(3, 4)
        self.hiddenlayer = torch.nn.Linear(4, 4)
        self.outputlayer = torch.nn.Linear(4, 2)

    def forward(self, x):
        x = torch.nn.LeakyReLU()(self.inputlayer(x))
        x = torch.nn.Dropout(self.dpo)(x)
        x = torch.nn.LeakyReLU()(self.hiddenlayer(x))
        x = torch.nn.Dropout(self.dpo)(x)
        x = self.outputlayer(x)
        return x


my_DNN = DNN_with_dpo(0.1)
optimizer = torch.optim.AdamW(my_DNN.parameters(), lr=1e-3, weight_decay=0.01)

for i in range(num_epochs):
    for batch, label in data:

        y = my_DNN(batch)
        loss = loss_func(y, label)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

Different Learning Paradigms



Particle Physics Analyses

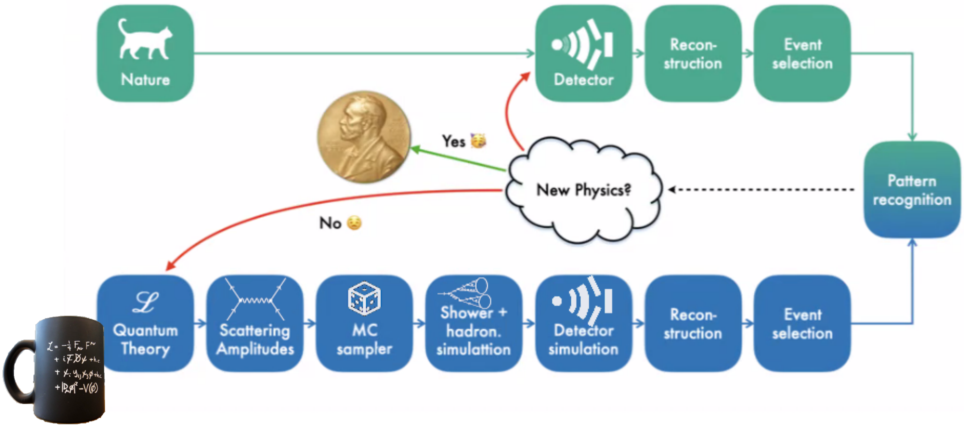
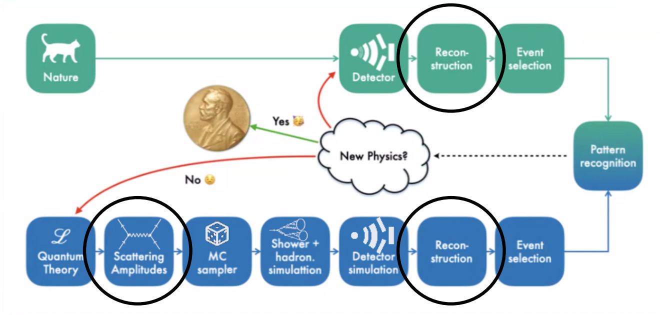


Figure inspired by R. Winterhalder

What kind of ML are we using and where?



- Regression
  - reconstruction: momenta, energy
  - expensive functions

Figure inspired by R. Winterhalder

What kind of ML are we using and where?

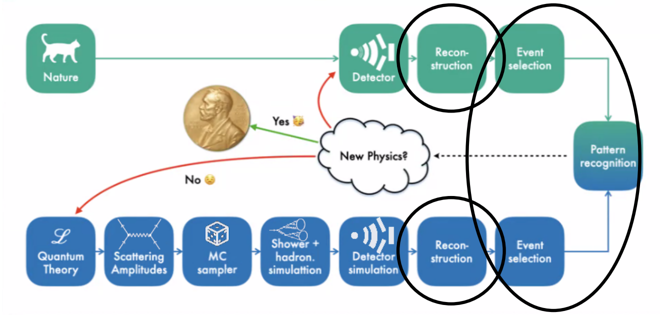
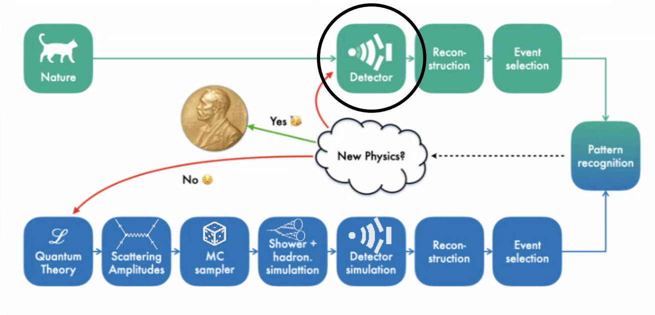


Figure inspired by R. Winterhalder

- Regression
  - reconstruction: momenta, energy
  - expensive functions
- Classification
  - reconstruction: particle type
  - signal vs. background

What kind of ML are we using and where?



- Regression
  - reconstruction: momenta, energy
  - expensive functions
- Classification
  - reconstruction: particle type
  - signal vs. background
- Reinforcement Learning
  - accelerator control

Figure inspired by R. Winterhalder

What kind of ML are we using and where?

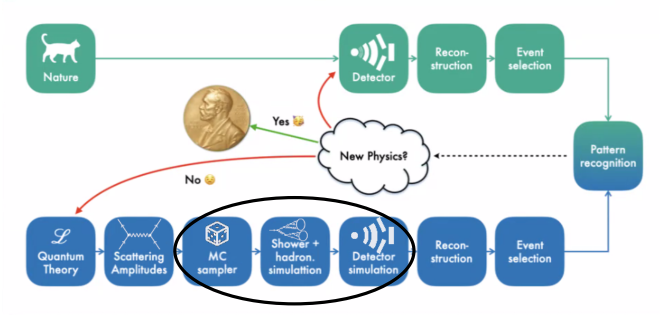
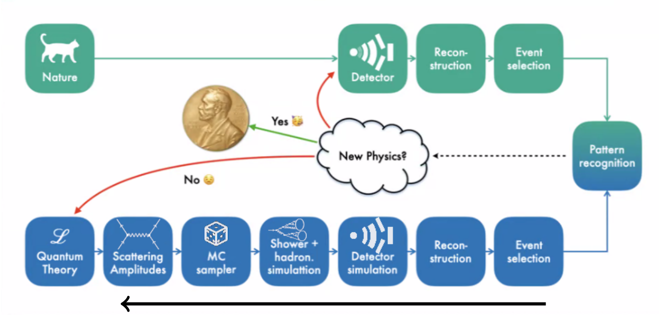


Figure inspired by R. Winterhalder

- Regression
  - reconstruction: momenta, energy
  - expensive functions
- Classification
  - reconstruction: particle type
  - signal vs. background
- Reinforcement Learning
  - accelerator control
- Generative Models
  - event generation
  - detector simulation

What kind of ML are we using and where?



- Regression
  - reconstruction: momenta, energy
  - expensive functions
- Classification
  - reconstruction: particle type
  - signal vs. background
- Reinforcement Learning
  - accelerator control
- Generative Models
  - event generation
  - detector simulation
- Simulation-based Inference

Figure inspired by R. Winterhalder

What kind of ML are we using and where?

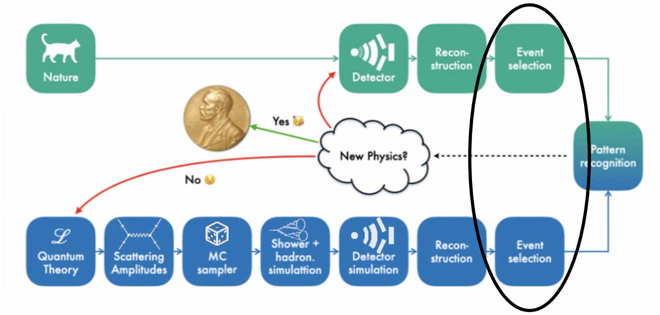
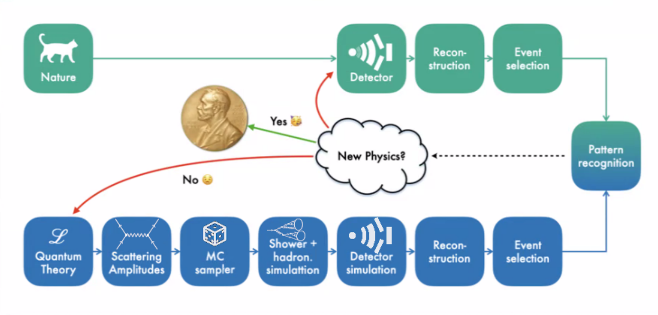


Figure inspired by R. Winterhalder

- Regression
  - reconstruction: momenta, energy
  - expensive functions
- Classification
  - reconstruction: particle type
  - signal vs. background
- Reinforcement Learning
  - accelerator control
- Generative Models
  - event generation
  - detector simulation
- Simulation-based Inference
- Anomaly Detection

What kind of ML are we using and where?



working on:
tabular data, point clouds, graphs, pixel/voxel

- Regression
  - reconstruction: momenta, energy
  - expensive functions
- Classification
  - reconstruction: particle type
  - signal vs. background
- Reinforcement Learning
  - accelerator control
- Generative Models
  - event generation
  - detector simulation
- Simulation-based Inference
- Anomaly Detection

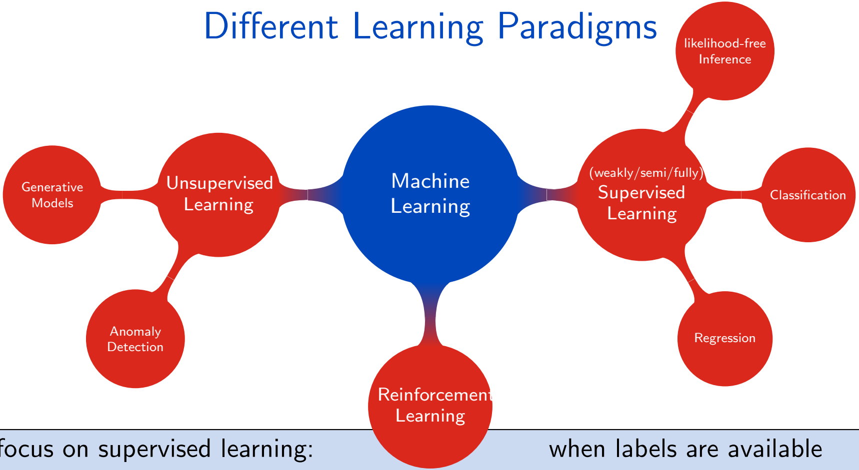
Figure inspired by R. Winterhalder

Machine Learning for Particle Physics

This week's plan:

1. Introduction (fits, optimization, and NNs)
2. Regression and Classification
3. Deep Generative Models
4. Anomaly Detection and Data-Driven Methods

Different Learning Paradigms



We first focus on supervised learning: when labels are available

- Regression: predict continuous values, like a scattering amplitude
- Classification: predict discrete label, like “signal” or “background”

Regression and the MSE-loss

We have data $(x_j, y_j = f(x_j))$ and want to learn $f_\theta(x) \approx f(x)$.

$\Rightarrow$ maximize the probability for the fit output $f_\theta(x_j)$ to correspond to the training points y_j.

$$p(x|\theta) = \prod_j \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{\left|y_j - f_\theta(x_j)\right|^2}{2\sigma_j^2}\right)$$

$$\Rightarrow \quad \log p(x|\theta) = -\sum_j \left(\frac{\left|y_j - f_\theta(x_j)\right|^2}{2\sigma_j^2}\right) + \text{const.}(\theta) \quad \Rightarrow \quad \mathcal{L}_{\text{fit}} = \sum_j \left(\frac{\left|y_j - f_\theta(x_j)\right|^2}{2\sigma_j^2 N}\right)$$

Regression and the MSE-loss

We have data $(x_j, y_j = f(x_j))$ and want to learn $f_\theta(x) \approx f(x)$.

$\Rightarrow$ maximize the probability for the fit output $f_\theta(x_j)$ to correspond

"usual" χ^2 minimization

$$p(x|\theta) = \prod_j \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{|y_j - f_\theta(x_j)|^2}{2\sigma_j^2}\right)$$

$$\Rightarrow \quad \log p(x|\theta) = -\sum_j \left(\frac{|y_j - f_\theta(x_j)|^2}{2\sigma_j^2}\right) + \text{const.}(\theta) \quad \Rightarrow \quad \mathcal{L}_{\text{fit}} = \sum_j \left(\frac{|y_j - f_\theta(x_j)|^2}{2\sigma_j^2 N}\right)$$

Regression and the MSE-loss

We have data $(x_j, y_j = f(x_j))$ and want to learn $f_\theta(x) \approx f(x)$.

$\Rightarrow$ maximize the probability for the fit output $f_\theta(x_j)$ to correspond to the training points y_j.

$$p(x|\theta) = \prod_j \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{\left|y_j - f_\theta(x_j)\right|^2}{2\sigma_j^2}\right)$$

$$\Rightarrow \quad \log p(x|\theta) = -\sum_j \left(\frac{\left|y_j - f_\theta(x_j)\right|^2}{2\sigma_j^2}\right) + \text{const.}(\theta) \qquad \Rightarrow \qquad \mathcal{L}_{\text{fit}} = \sum_j \left(\frac{\left|y_j - f_\theta(x_j)\right|^2}{2\sigma_j^2 N}\right)$$

If error σ_j unknown, or same for all: $\mathcal{L} = \frac{1}{2N\sigma} \left|y_j - f_\theta(x_j)\right|^2 \equiv \frac{1}{2\sigma}\text{MSE}$

Binary Classification and the BCE-loss

In Binary Classification, we want to predict a discrete label: class 0 or class 1.
$\Rightarrow$ interpret NN output as $p(\text{class } 1)$

$\Rightarrow$ maximize $p(x_i)$ predicting the correct label y_i.

$$p(x|\theta) = \prod_j \begin{cases} p(x_j) & \text{if } y_j = 1 \\ 1 - p(x_j) & \text{if } y_j = 0 \end{cases} = \prod_j p(x_j)^{y_j} (1 - p(x_j))^{(1-y_j)}$$

$$\Rightarrow \quad \log p(x|\theta) = \sum_j y_j \log p(x_j) + (1 - y_j) \log (1 - p(x_j))$$

Binary Classification and the BCE-loss

In Binary Classification, we want to predict a discrete label: class 0 or class 1.
$\Rightarrow$ interpret NN output as $p(\text{class } 1)$

$\Rightarrow$ maximize $p(x_i)$ predicting the correct label y_i.

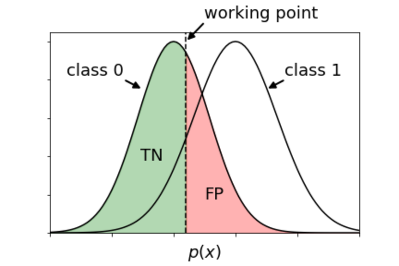
$$p(x|\theta) = \prod_j \begin{cases} p(x_j) & \text{if } y_j = 1 \\ 1 - p(x_j) & \text{if } y_j = 0 \end{cases} = \prod_j p(x_j)^{y_j} (1 - p(x_j))^{(1-y_j)}$$

$$\Rightarrow \quad \log p(x|\theta) = \sum_j y_j \log p(x_j) + (1 - y_j) \log (1 - p(x_j))$$

$$\Rightarrow \quad \mathcal{L}_{\text{BCE}} = -\sum_j y_j \log p(x_j) + (1 - y_j) \log (1 - p(x_j)) \qquad \mathcal{L}_{\text{CE}} = -\sum_{j \in C_i} y_j \log p_i(x_j)$$

Performance Metrics of Classifiers

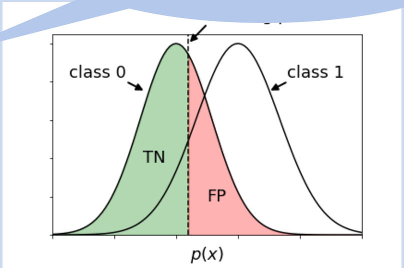
- false positive rate (background efficiency): $\frac{\text{FP}}{\text{FP} + \text{TN}}$



Performance Metrics of Classifiers

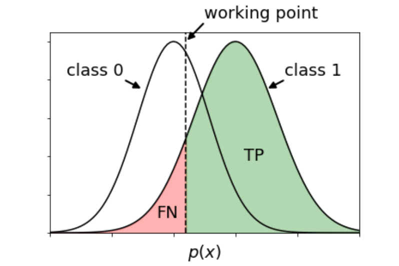
spam e-mail: no FP

- false positive rate (background efficiency): $\frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}$



Performance Metrics of Classifiers

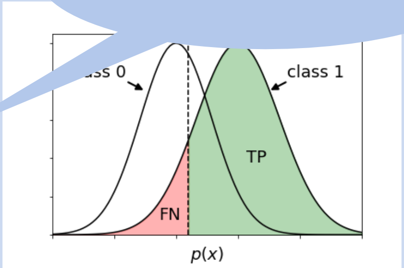
- false positive rate (background efficiency): $\frac{\text{FP}}{\text{FP} + \text{TN}}$
- true positive rate (signal efficiency): $\frac{\text{TP}}{\text{TP} + \text{FN}}$



Performance Metrics of Classifiers

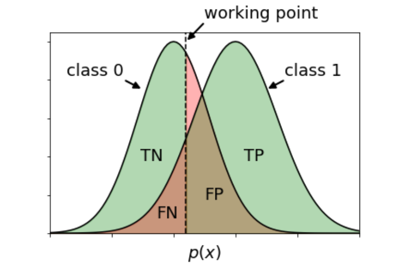
- false positive rate (background efficiency): $\frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}}$
- true positive rate (signal efficiency): $\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$

health screening: no FN



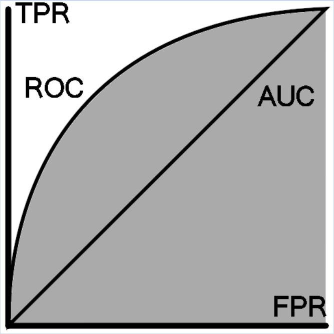
Performance Metrics of Classifiers

- false positive rate (background efficiency): $\frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}}$
- true positive rate (signal efficiency): $\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$
- accuracy: $\frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{FN}+\mathrm{FP}+\mathrm{TN}}$



Performance Metrics of Classifiers

- false positive rate (background efficiency): $\frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}}$
- true positive rate (signal efficiency): $\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$
- accuracy: $\frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{FN}+\mathrm{FP}+\mathrm{TN}}$



Machine Learning for Particle Physics

This week's plan:

1. Introduction (fits, optimization, and NNs)
2. Regression and Classification
3. Deep Generative Models
   - Normalizing Flows
   - Denoising Diffusion Probabilistic Models (DDPMs)
   - Conditional Flow Matching (CFM)
   - Applications
   - How to evaluate Generative Models
4. Anomaly Detection and Data-Driven Methods

Motivation: Generative Models

We have a distribution $p(x)$ and want to sample ("generate") new elements that follow it.

given: $\{x_i\}$ — want: $x \sim p(x)$

- or -

given: $f(x)$ — want: $x \sim f(x) / \int f(x)\, dx$

Motivation: Generative Models

We have a distribution $p(x)$ and want to sample ("generate") new elements that follow it.

given: $\{x_i\}$ want: $x \sim p(x)$

- or -

given: $f(x)$ want: $x \sim f(x) / \int f(x)\, dx$

They can be understood as fancy random number generators, with the numbers being:

- pixels of an image

via midjourney.com

⇒ image generators like `MidJourney`, `DALL·E`

Motivation: Generative Models

We have a distribution $p(x)$ and want to sample ("generate") new elements that follow it.

given: $\{x_i\}$ want: $x \sim p(x)$

- or -

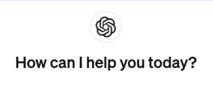
given: $f(x)$ want: $x \sim f(x) / \int f(x)\, dx$

They can be understood as fancy random number generators, with the numbers being:

- pixels of an image

via midjourney.com

- translated to words



$\Rightarrow$ chatbots like `ChatGPT`, `GitHub CoPilot`

$\Rightarrow$ image generators like `MidJourney`, `DALL·E`

Motivation: Generative Models

We have a distribution $p(x)$ and want to sample ("generate") new elements that follow it.

given: $\{x_i\}$ — want: $x \sim p(x)$

- or -

given: $f(x)$ — want: $x \sim f(x) / \int f(x)\, dx$

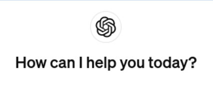
They can be understood as fancy random number generators, with the numbers being:

- pixels of an image

via midjourney.com

⇒ image generators like `MidJourney`, `DALL·E`

- translated to words



⇒ chatbots like `ChatGPT`, `GitHub CoPilot`

- four momenta of particles

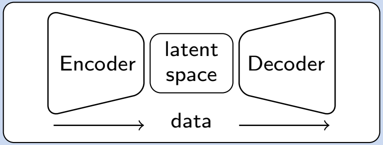
MADGRAPH 5 SHERPA

⇒ event generators like `MadGraph` and `Sherpa`

The Landscape of Generative Models.

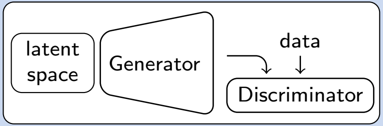
Variational Autoencoder (VAE)

⇒ Compressing data through a bottleneck.



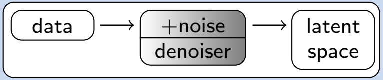
Generative Adversarial Network (GAN)

⇒ Generator and Discriminator play a game against each other.



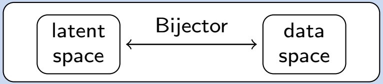
Diffusion Models

⇒ Gradually add noise and revert.



Normalizing Flows

⇒ Bijective map to a known distribution.



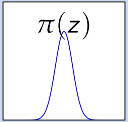
Machine Learning for Particle Physics

This week's plan:

1. Introduction (fits, optimization, and NNs)
2. Regression and Classification
3. Deep Generative Models
   - Normalizing Flows
   - Denoising Diffusion Probabilistic Models (DDPMs)
   - Conditional Flow Matching (CFM)
   - Applications
   - How to evaluate Generative Models
4. Anomaly Detection and Data-Driven Methods

Normalizing Flows in a Nutshell

"easy" base distribution

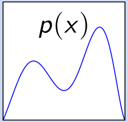


$\Leftrightarrow$

bijective transformation $z = f(x)$

$\Leftrightarrow$

"target" distribution



$$p(x) = \pi(f(x)) \left| \det \frac{\partial f(x)}{\partial x} \right|$$

density estimation, $p(x)$ ←

sample generation →

Training Normalizing Flows

Maximum Likelihood Estimation gives the best loss functions:

- Regression: Mean Squared Error Loss
- Binary classification: Binary Cross Entropy Loss
- ...

Normalizing Flows give us the log-likelihood (LL) explicitly!

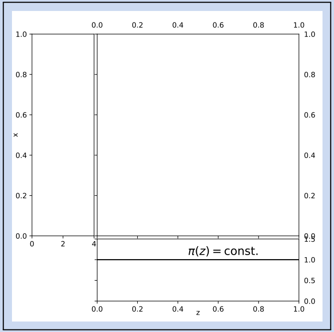
⇒ Maximize $\log p$ (the LL) over the given samples.
$\mathcal{L} = -\sum_i \log p_\theta(x_i)$

⇒ If we don't have samples, but a normalized target $q(x)$, we can use the KL-divergence.
$\mathcal{L} = D_{rKL}[p_\theta, q] = \int dx\ p_\theta(x)\ \log \frac{p_\theta(x)}{q(x)} = \left\langle \frac{p_\theta(x)}{p_\theta(x)} \log \frac{p_\theta(x)}{q(x)} \right\rangle_{x \sim p_\theta(x)}$

At the Core: Change of Coordinates Formula

Changing coordinates from $\vec{z}$ to $\vec{x}$ with a map $\vec{x} = f(\vec{z})$ changes the distribution according to

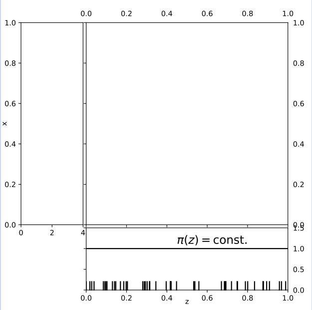
$$\bar{\pi}(\vec{x}) = \pi(\vec{z}) \left| \det \frac{\partial f(\vec{z})}{\partial \vec{z}} \right|^{-1} = \pi(f^{-1}(\vec{x})) \left| \det \frac{\partial f^{-1}(\vec{x})}{\partial \vec{x}} \right|$$



At the Core: Change of Coordinates Formula

Changing coordinates from $\vec{z}$ to $\vec{x}$ with a map $\vec{x} = f(\vec{z})$ changes the distribution according to

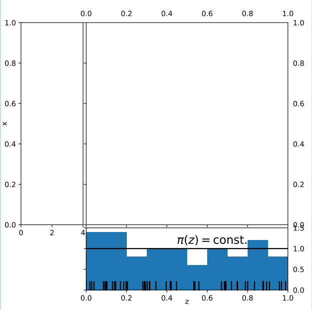
$$\bar{\pi}(\vec{x}) \; = \; \pi(\vec{z}) \left| \det \frac{\partial f(\vec{z})}{\partial \vec{z}} \right|^{-1} = \pi(f^{-1}(\vec{x})) \left| \det \frac{\partial f^{-1}(\vec{x})}{\partial \vec{x}} \right|$$



At the Core: Change of Coordinates Formula

Changing coordinates from $\vec{z}$ to $\vec{x}$ with a map $\vec{x} = f(\vec{z})$ changes the distribution according to

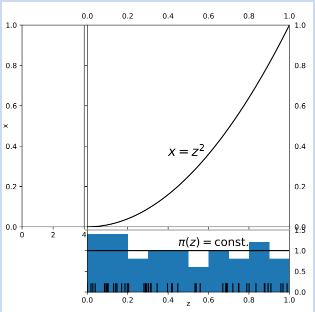
$$\bar{\pi}(\vec{x}) = \pi(\vec{z}) \left| \det \frac{\partial f(\vec{z})}{\partial \vec{z}} \right|^{-1} = \pi(f^{-1}(\vec{x})) \left| \det \frac{\partial f^{-1}(\vec{x})}{\partial \vec{x}} \right|$$



At the Core: Change of Coordinates Formula

Changing coordinates from $\vec{z}$ to $\vec{x}$ with a map $\vec{x} = f(\vec{z})$ changes the distribution according to

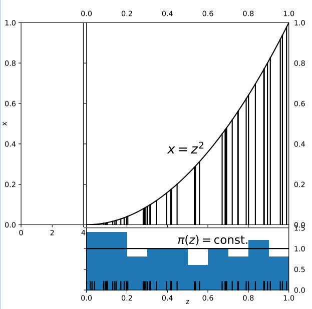
$$\bar{\pi}(\vec{x}) = \pi(\vec{z}) \left| \det \frac{\partial f(\vec{z})}{\partial \vec{z}} \right|^{-1} = \pi(f^{-1}(\vec{x})) \left| \det \frac{\partial f^{-1}(\vec{x})}{\partial \vec{x}} \right|$$



At the Core: Change of Coordinates Formula

Changing coordinates from $\vec{z}$ to $\vec{x}$ with a map $\vec{x} = f(\vec{z})$ changes the distribution according to

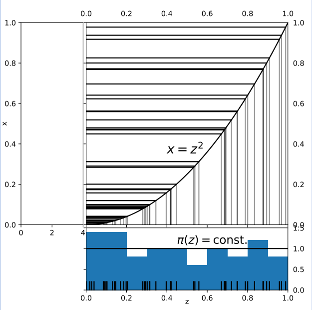
$$\bar{\pi}(\vec{x}) = \pi(\vec{z}) \left| \det \frac{\partial f(\vec{z})}{\partial \vec{z}} \right|^{-1} = \pi(f^{-1}(\vec{x})) \left| \det \frac{\partial f^{-1}(\vec{x})}{\partial \vec{x}} \right|$$



At the Core: Change of Coordinates Formula

Changing coordinates from $\vec{z}$ to $\vec{x}$ with a map $\vec{x} = f(\vec{z})$ changes the distribution according to

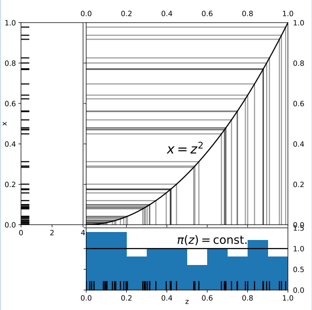
$$\bar{\pi}(\vec{x}) = \pi(\vec{z}) \left|\det \frac{\partial f(\vec{z})}{\partial \vec{z}}\right|^{-1} = \pi(f^{-1}(\vec{x})) \left|\det \frac{\partial f^{-1}(\vec{x})}{\partial \vec{x}}\right|$$



At the Core: Change of Coordinates Formula

Changing coordinates from $\vec{z}$ to $\vec{x}$ with a map $\vec{x} = f(\vec{z})$ changes the distribution according to

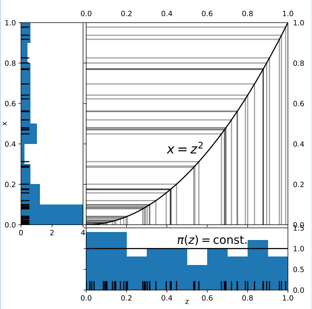
$$\bar{\pi}(\vec{x}) = \pi(\vec{z}) \left| \det \frac{\partial f(\vec{z})}{\partial \vec{z}} \right|^{-1} = \pi(f^{-1}(\vec{x})) \left| \det \frac{\partial f^{-1}(\vec{x})}{\partial \vec{x}} \right|$$



At the Core: Change of Coordinates Formula

Changing coordinates from $\vec{z}$ to $\vec{x}$ with a map $\vec{x} = f(\vec{z})$ changes the distribution according to

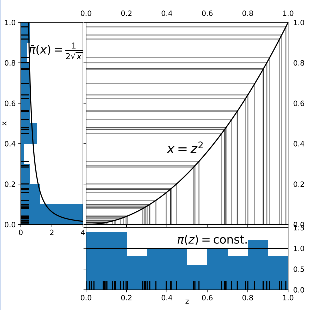
$$\bar{\pi}(\vec{x}) \;=\; \pi(\vec{z}) \left| \det \frac{\partial f(\vec{z})}{\partial \vec{z}} \right|^{-1} = \pi(f^{-1}(\vec{x})) \left| \det \frac{\partial f^{-1}(\vec{x})}{\partial \vec{x}} \right|$$



At the Core: Change of Coordinates Formula

Changing coordinates from $\vec{z}$ to $\vec{x}$ with a map $\vec{x} = f(\vec{z})$ changes the distribution according to

$$\bar{\pi}(\vec{x}) \;=\; \pi(\vec{z}) \left|\det \frac{\partial f(\vec{z})}{\partial \vec{z}}\right|^{-1} = \pi(f^{-1}(\vec{x})) \left|\det \frac{\partial f^{-1}(\vec{x})}{\partial \vec{x}}\right|$$



Base distributions

$$\bar{\pi}(\vec{x}) \quad = \quad \pi(\vec{z}) \left|\det \frac{\partial f(\vec{z})}{\partial \vec{z}}\right|^{-1} \quad = \quad \pi(f^{-1}(\vec{x})) \left|\det \frac{\partial f^{-1}(\vec{x})}{\partial \vec{x}}\right|$$

- Can be any distribution with only 2 requirements:
  - We can easily sample from it
  - We have access to $\pi(x)$
- Sets the initial domain of the coordinates.
- Most common choices:
  - uniform distribution (compact in $[a, b]$)
  - Gaussian distribution (in $\mathbb{R}$)
- Topology should match the topology of the target space.

We need a trackable Jacobian and Inverse.

$$\bar{\pi}(\vec{x}) \quad = \quad \pi(\vec{z}) \left|\det \frac{\partial f(\vec{z})}{\partial \vec{z}}\right|^{-1} \quad = \quad \pi(f^{-1}(\vec{x})) \left|\det \frac{\partial f^{-1}(\vec{x})}{\partial \vec{x}}\right|$$

- First idea: making f a NN.
  - × inverse does not always exist
  - × Jacobian slow via autograd
  - × $\left|\det \frac{\partial f}{\partial z}\right| \propto \mathcal{O}(n_{dim}^3)$

Dinh et al. [arXiv:1410.8516], Rezende/Mohamed [arXiv:1505.05770]

We need a trackable Jacobian and Inverse.

$$\bar{\pi}(\vec{x}) = \pi(\vec{z}) \left|\det \frac{\partial f(\vec{z})}{\partial \vec{z}}\right|^{-1} = \pi(f^{-1}(\vec{x})) \left|\det \frac{\partial f^{-1}(\vec{x})}{\partial \vec{x}}\right|$$

- First idea: making f a NN.
  - × inverse does not always exist
  - × Jacobian slow via autograd
  - × $\left|\det \frac{\partial f}{\partial z}\right| \propto \mathcal{O}(n_{dim}^3)$

⇒ Let a NN learn parameters κ of a pre-defined transformation!

- Each transformation is 1d & has an analytic Jacobian and inverse.
  $\Rightarrow \vec{f}(\vec{x};\vec{\kappa}) = (C_1(x_1;\kappa_1), C_2(x_2;\kappa_2), \dots, C_n(x_n;\kappa_n))^T$

Dinh et al. [arXiv:1410.8516], Rezende/Mohamed [arXiv:1505.05770]

We need a trackable Jacobian and Inverse.

$$\bar{\pi}(\vec{x}) \quad = \quad \pi(\vec{z}) \left|\det \frac{\partial f(\vec{z})}{\partial \vec{z}}\right|^{-1} \quad = \quad \pi(f^{-1}(\vec{x})) \left|\det \frac{\partial f^{-1}(\vec{x})}{\partial \vec{x}}\right|$$

- First idea: making f a NN.
  - × inverse does not always exist
  - × Jacobian slow via autograd
  - × $\left|\det \frac{\partial f}{\partial z}\right| \propto \mathcal{O}(n_{dim}^3)$

⇒ Let a NN learn parameters κ of a pre-defined transformation!

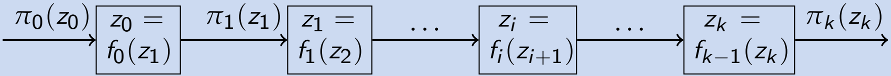
- Each transformation is 1d & has an analytic Jacobian and inverse.
  $\Rightarrow \vec{f}(\vec{x};\vec{\kappa}) = (C_1(x_1;\kappa_1), C_2(x_2;\kappa_2), \ldots, C_n(x_n;\kappa_n))^T$
- Require a triangular Jacobian for faster evaluation.

  ⇒ The parameters κ depend only on a subset of all other coordinates.

Dinh et al. [arXiv:1410.8516], Rezende/Mohamed [arXiv:1505.05770]

A chain of bijectors is also a bijector

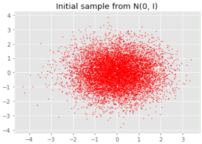
The full transformation is a chain of these bijectors.

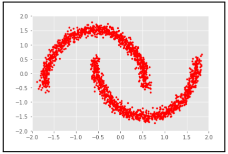


A chain of bijectors is also a bijector

The full transformation is a chain of these bijectors.

$$\xrightarrow{\pi_0(z_0)} \boxed{z_0 = f_0(z_1)} \xrightarrow{\pi_1(z_1)} \boxed{z_1 = f_1(z_2)} \xrightarrow{\dots} \boxed{z_i = f_i(z_{i+1})} \xrightarrow{\dots} \boxed{z_k = f_{k-1}(z_k)} \xrightarrow{\pi_k(z_k)}$$



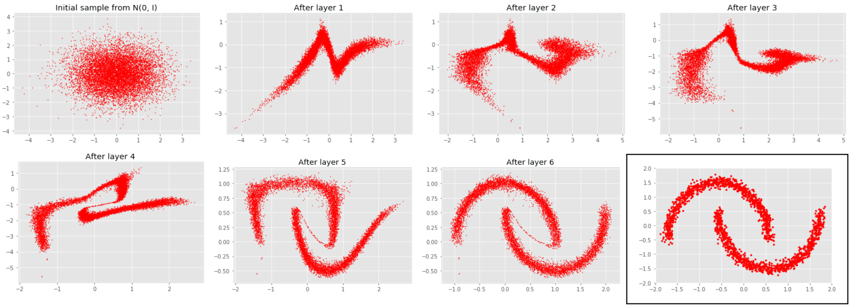


https://engineering.papercup.com/posts/normalizing-flows-part-2/

A chain of bijectors is also a bijector

The full transformation is a chain of these bijectors.

$$\xrightarrow{\pi_0(z_0)} \boxed{z_0 = f_0(z_1)} \xrightarrow{\pi_1(z_1)} \boxed{z_1 = f_1(z_2)} \xrightarrow{\dots} \boxed{z_i = f_i(z_{i+1})} \xrightarrow{\dots} \boxed{z_k = f_{k-1}(z_k)} \xrightarrow{\pi_k(z_k)}$$



https://engineering.papercup.com/posts/normalizing-flows-part-2/

Affine Transformations

The coupling function (transformation)

- must be invertible and expressive
- is chosen to factorize:
  $\vec{f}(\vec{x};\vec{\kappa}) = \left(C_1(x_1;\kappa_1), C_2(x_2;\kappa_2), \ldots, C_n(x_n;\kappa_n)\right)^T$,
  where $\vec{x}$ are the coordinates to be transformed and $\vec{\kappa}$ the parameters of the transformation.

historically first: the affine coupling function

$$C(x; s, t) = \exp(s)\, x + t$$

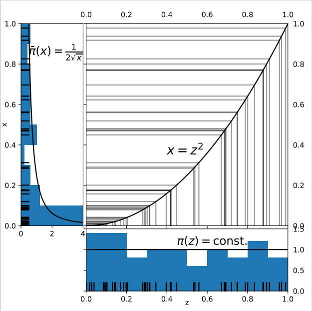
where s and t are predicted by a NN.

- It requires $x \in \mathbb{R}$.
- Inverse and Jacobian are trivial.
- Its transformation powers are limited.

Any monotonic function can be used.

Changing coordinates from $\vec{z}$ to $\vec{x}$ with a map $\vec{x} = f(\vec{z})$ changes the distribution according to

$$\bar{\pi}(\vec{x}) = \pi(\vec{z}) \left| \det \frac{\partial f(\vec{z})}{\partial \vec{z}} \right|^{-1} = \pi(f^{-1}(\vec{x})) \left| \det \frac{\partial f^{-1}(\vec{x})}{\partial \vec{x}} \right|$$

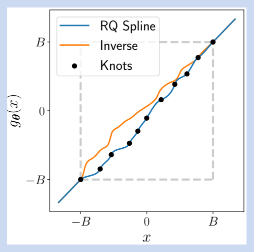


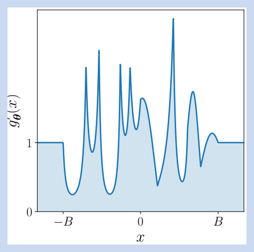
Any monotonic function can be used.

Changing coordinates from $\vec{z}$ to $\vec{x}$ with a map $\vec{x} = f(\vec{z})$ changes the distribution according to

$$\bar{\pi}(\vec{x}) = \pi(\vec{z}) \left| \det \frac{\partial f(\vec{z})}{\partial \vec{z}} \right|^{-1} = \pi(f^{-1}(\vec{x})) \left| \det \frac{\partial f^{-1}(\vec{x})}{\partial \vec{x}} \right|$$

A more complicated transformation then leads to a more complicated transformed distribution. Splines act in a finite domain.

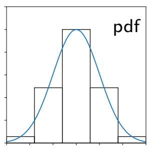


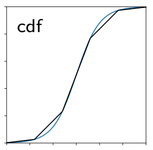


figures from Durkan et al. [arXiv:1906.04032]

Piecewise Transformations (Splines)

piecewise linear coupling function: Müller et al. [arXiv:1808.03856]



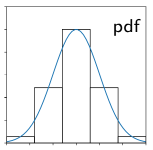


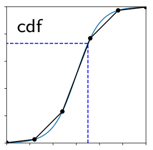
The NN predicts the pdf bin heights Q_i.

Piecewise Transformations (Splines)

piecewise linear coupling function:

Müller et al. [arXiv:1808.03856]





$$C = \sum_{k=1}^{b-1} Q_k + \alpha Q_b, \qquad \alpha = \frac{x-(b-1)w}{w}$$

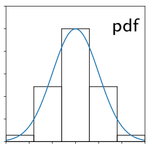
$$\left| \frac{\partial C}{\partial \vec{x}} \right| = \prod_i \frac{Q_{b_i}}{w}$$

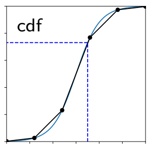
The NN predicts the pdf bin heights Q_i.

Piecewise Transformations (Splines)

piecewise linear coupling function:

Müller et al. [arXiv:1808.03856]





$$C = \sum_{k=1}^{b-1} Q_k + \alpha Q_b, \qquad \alpha = \frac{x-(b-1)w}{w}$$

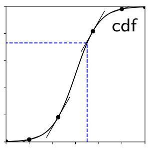
$$\left|\frac{\partial C}{\partial \vec{x}}\right| = \prod_i \frac{Q_{b_i}}{w}$$

The NN predicts the pdf bin heights Q_i.

rational quadratic spline coupling function:

Durkan et al. [arXiv:1906.04032]

Gregory/Delbourgo [IMA Journal of Numerical Analysis, '82]



$$C = \frac{a_2\alpha^2 + a_1\alpha + a_0}{b_2\alpha^2 + b_1\alpha + b_0}$$

- still rather easy
- more flexible

The NN predicts the cdf bin widths, heights, and derivatives that go in a_i&b_i.

Taming Jacobians: Bipartite Flows ("INNs")

$$\kappa_{x\in A}(x \in B) \qquad \& \qquad \kappa_{x\in B}(x \in A)$$

⇒ Coordinates are split in 2 sets, transforming each other.

forward:

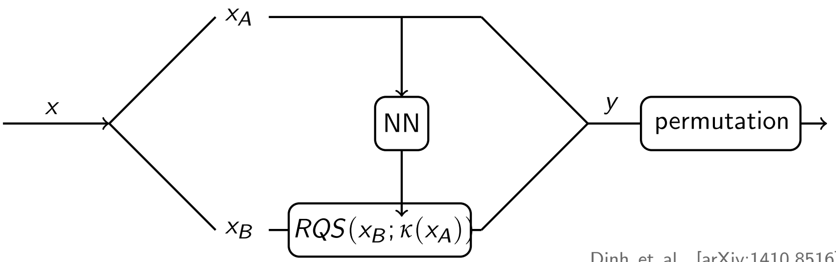
$$y_A = x_A$$
$$y_{B,i} = C(x_{B,i}; \kappa(x_A))$$

inverse:

$$x_A = y_A$$
$$x_{B,i} = C^{-1}(y_{B,i}; \kappa(x_A))$$

Jacobian:

$$\begin{vmatrix} 1 & \frac{\partial C}{\partial x_A} \\ 0 & \frac{\partial C}{\partial x_B} \end{vmatrix} = \prod_i \frac{\partial C(x_{B,i}; \kappa(x_A))}{\partial x_{B,i}}$$



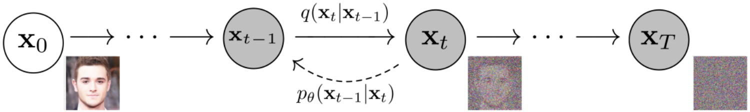
Dinh et al. [arXiv:1410.8516]

Machine Learning for Particle Physics

This week's plan:

1. Introduction (fits, optimization, and NNs)
2. Regression and Classification
3. Deep Generative Models
   - Normalizing Flows
   - Denoising Diffusion Probabilistic Models (DDPMs)
   - Conditional Flow Matching (CFM)
   - Applications
   - How to evaluate Generative Models
4. Anomaly Detection and Data-Driven Methods

Denoising Diffusion Probabilistic Models

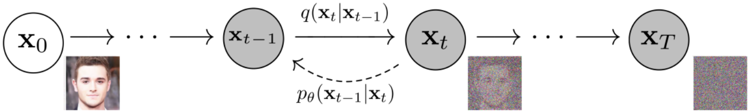


by Sofia Palacios Schweitzer and Ho et al. [arXiv:2006.11239]

$q(x_1, \dots, x_T|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$, with $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t} x_{t-1}, \beta_t)$ and a noise schedule β_t.

$\Rightarrow$ now learn inverse: $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_\theta^2(x_t, t))$

Denoising Diffusion Probabilistic Models



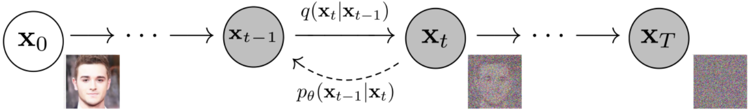
by Sofia Palacios Schweitzer and Ho et al. [arXiv:2006.11239]

$q(x_1, \dots, x_T|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$, with $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t)$ and a noise schedule β_t.

$\Rightarrow$ now learn inverse: $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_\theta^2(x_t, t))$

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{1-\bar{\beta}_t}x_0, \bar{\beta}_t)$$
with $1-\bar{\beta}_t = \prod_{i=1}^{t} 1-\bar{\beta}_i$

Denoising Diffusion Probabilistic Models



by Sofia Palacios Schweitzer and Ho et al. [arXiv:2006.11239]

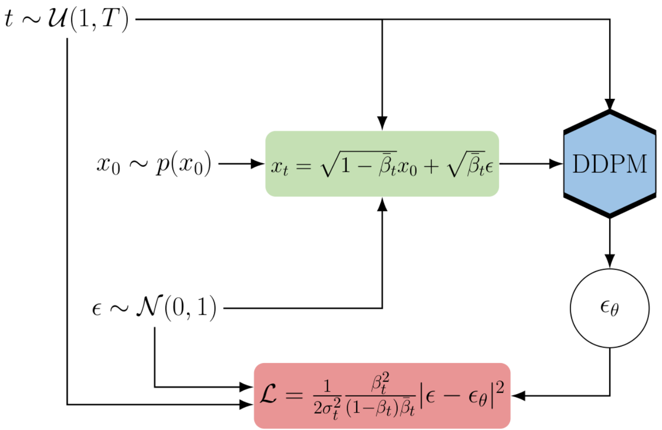
$q(x_1, \ldots, x_T|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}),$ with $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t} x_{t-1}, \beta_t)$ and a noise schedule β_t.

$\Rightarrow$ now learn inverse: $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_\theta^2(x_t, t))$

$$\mathcal{L}_{\text{DDPM}} = \frac{1}{2\sigma_t^2} \frac{\beta_t^2}{(1-\beta_t)\bar{\beta}_t} \left|\epsilon_t - \epsilon_\theta(x_t, t)\right|^2$$

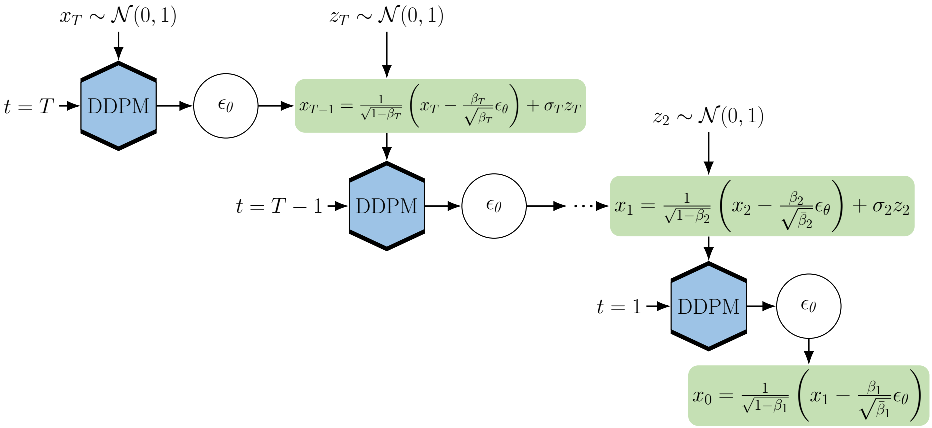
more math and details by Sofia Palacios Schweitzer et al. [arXiv:2305.10475] and Ho et al. [arXiv:2006.11239]

Denoising Diffusion Probabilistic Models Training



Sofia Palacios Schweitzer et al. [arXiv:2305.10475]

Denoising Diffusion Probabilistic Models Sampling



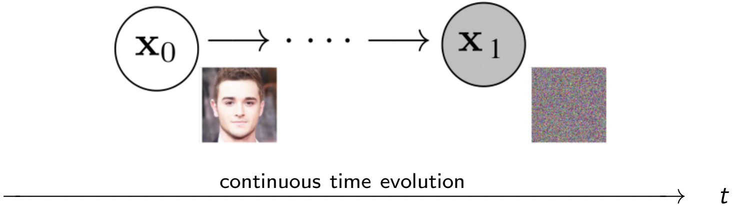
Sofia Palacios Schweitzer et al. [arXiv:2305.10475]

Machine Learning for Particle Physics

This week's plan:

1. Introduction (fits, optimization, and NNs)
2. Regression and Classification
3. Deep Generative Models
   - Normalizing Flows
   - Denoising Diffusion Probabilistic Models (DDPMs)
   - Conditional Flow Matching (CFM)
   - Applications
   - How to evaluate Generative Models
4. Anomaly Detection and Data-Driven Methods

Conditional Flow Matching: Connecting Normalizing Flows and Diffusion Models



by Sofia Palacios Schweitzer and Ho et al. [arXiv:2006.11239]

Continuous Normalizing Flow: $x_1 = x_0 + \int_0^1 v(x,t)\ dt \quad \Leftrightarrow \quad \frac{d}{dt}x(t) = v(x,t)$

$\Rightarrow$ connect data and latent space with ODE instead of discrete bijector

Conditional Flow Matching Setup

Huang/Yeh [arXiv:2012.04228]

Ordinary Differential Equation

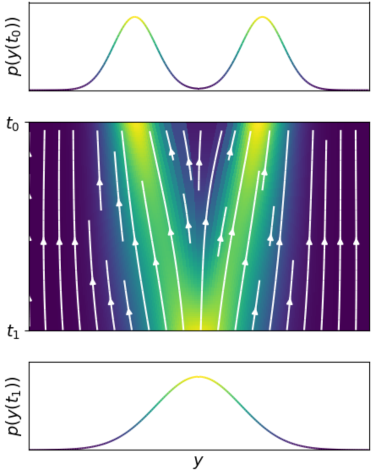
$\frac{d}{dt} x(t) = v(x(t), t), \quad \text{with } x(t = 0) = x_0$

Continuity Equation

$\frac{\partial}{\partial t} p(x, t) + \nabla_x (p(x, t) v(x, t)) = 0$

Diffusion Process

$$p(x, t) = \begin{cases} p_{\text{data}}(x) & t \to 0 \\ p_{\text{latent}}(x) \equiv \mathcal{N}(x; 0, 1) & t \to 1 \end{cases}$$



Conditional Flow Matching Training

naive regression of $v(x,t)$:

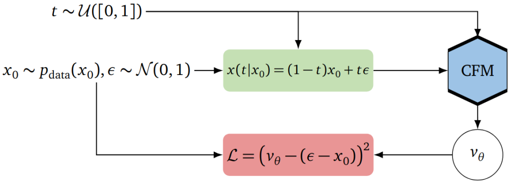
$$\mathcal{L}_{\text{FM}} = \left\langle \left(v_\theta(x,t) - v(x,t)\right)^2 \right\rangle_{\substack{t\sim\mathcal{U}[0,1] \\ x\sim p(x,t)}}$$

but: $v(x,t)$ and $p(x,t)$ are not tractable!

Solution:

$v(x,t|x_0)$ and $p(x,t|x_0)$ are!

$$\mathcal{L}_{\text{CFM}} = \left\langle \left(v_\theta(x(t|x_0),t) - v(x(t|x_0),t|x_0)\right)^2 \right\rangle_{\substack{t\sim\mathcal{U}[0,1] \\ x_0\sim\text{data}}}$$



Sofia Palacios Schweitzer et al. [arXiv:2305.10475]

Machine Learning for Particle Physics

This week's plan:

1. Introduction (fits, optimization, and NNs)
2. Regression and Classification
3. Deep Generative Models
   - Normalizing Flows
   - Denoising Diffusion Probabilistic Models (DDPMs)
   - Conditional Flow Matching (CFM)
   - Applications
   - How to evaluate Generative Models
4. Anomaly Detection and Data-Driven Methods

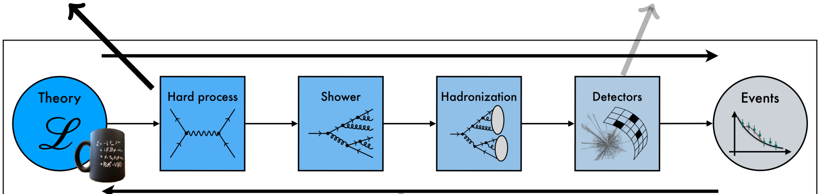
Applications of Generative Models

Event Generation

$p(\text{momenta, angles}|\text{process})$

Detector Simulation

$p(\text{particle shower}|\text{initial condition})$



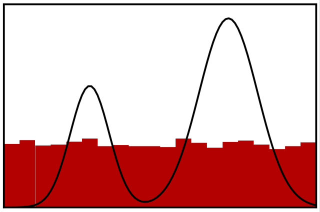
Machine Learning and LHC Event Generation, A. Butter et al. [2203.07460], R. Winterhalder

Inverse Problems

$p(\text{parameters}|\text{data})$

Event Generation uses Importance Sampling.

$$I = \int_0^1 f(\vec{x})\ d\vec{x}$$

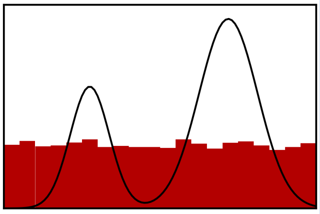


flat sampling: inefficient.

$$I = \langle f(\vec{x}) \rangle_{x\sim\text{uniform}}$$

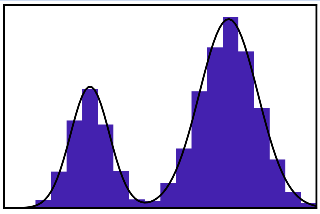
Event Generation uses Importance Sampling.

$$I = \int_0^1 f(\vec{x})\, d\vec{x}$$



flat sampling: inefficient.

$$I = \langle f(\vec{x}) \rangle_{x\sim\text{uniform}}$$

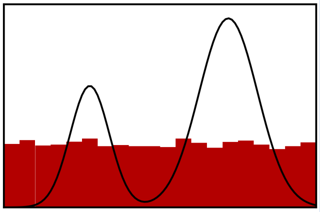


importance sampling: find g close to f

$$I = \left\langle \frac{f(\vec{x})}{g(\vec{x})} \right\rangle_{x\sim g(x)}$$

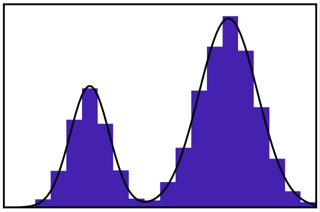
Event Generation uses Importance Sampling.

$$I = \int_0^1 f(\vec{x})\, d\vec{x}$$



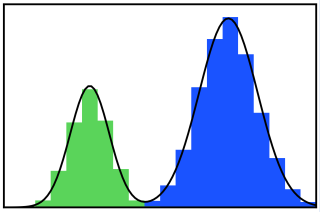
flat sampling: inefficient.

$$I = \langle f(\vec{x})\rangle_{x\sim\text{uniform}}$$



importance sampling: find g close to f

$$I = \left\langle \frac{f(\vec{x})}{g(\vec{x})} \right\rangle_{x\sim g(x)}$$

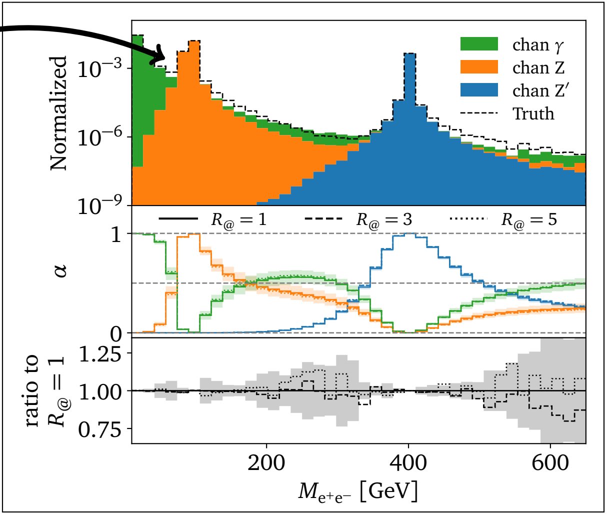


multichannel: one map per channel

$$I = \sum_i \left\langle \alpha_i(x) \frac{f(\vec{x})}{g_i(\vec{x})} \right\rangle_{x\sim g_i(x)}$$

Neural Importance Sampling — Results for $q\bar{q} \to \gamma / Z / Z' \to e^+e^-$

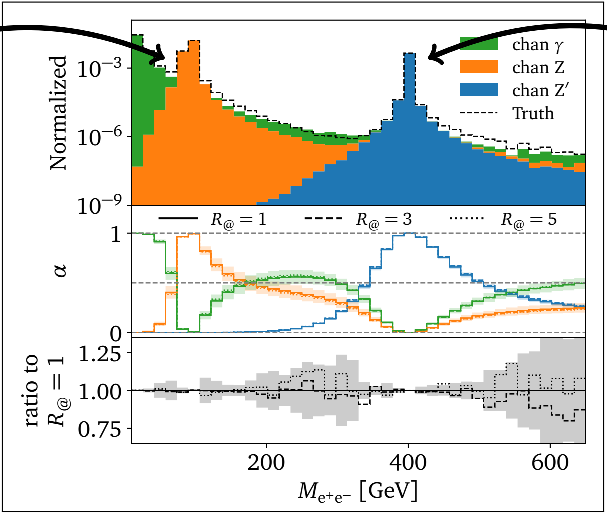
Learned distribution matches truth.



Heimel, CK et al. [2212.06172, SciPost]

Neural Importance Sampling — Results for $q\bar{q} \to \gamma / Z / Z' \to e^+e^-$

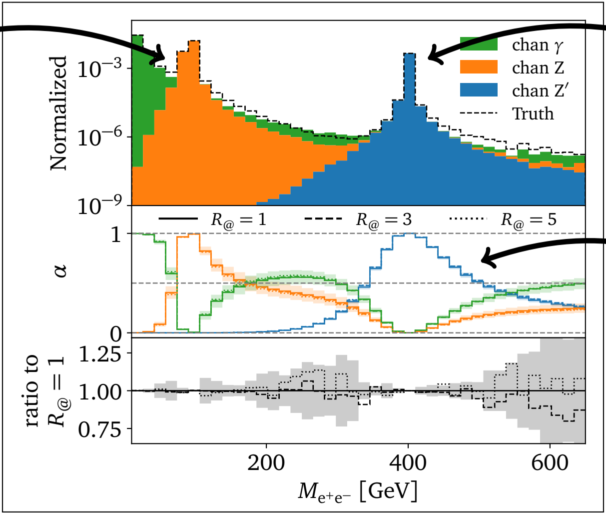
Learned distribution matches truth.



Peaks are learned by different channels.

Heimel, CK et al. [2212.06172, SciPost]

Neural Importance Sampling — Results for $q\bar{q} \to \gamma / Z / Z' \to e^+e^-$



Learned distribution matches truth.

Peaks are learned by different channels.

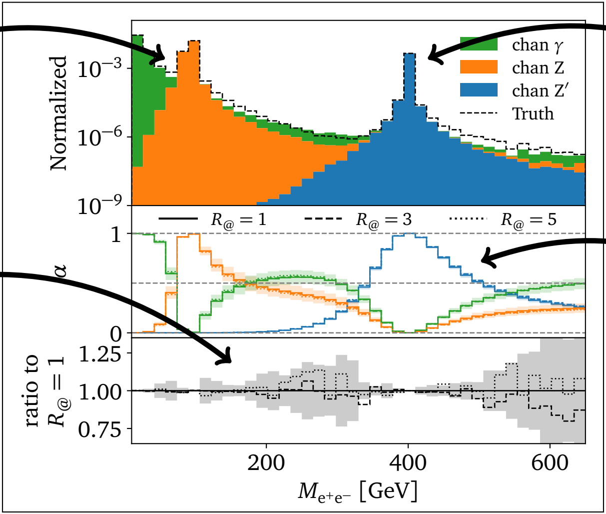
Channel weights are learned by the network

Heimel, CK et al. [2212.06172, SciPost]

Neural Importance Sampling — Results for $q\bar{q} \to \gamma / Z / Z' \to e^+e^-$

Learned distribution matches truth.

Re-uses samples to make training faster.



Peaks are learned by different channels.

Channel weights are learned by the network

Heimel, CK et al. [2212.06172, SciPost]

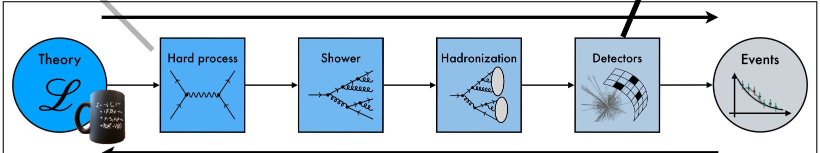
Applications of Generative Models

Event Generation

$p(\text{momenta, angles}|\text{process})$

Detector Simulation

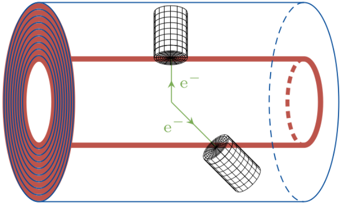
$p(\text{particle shower}|\text{initial condition})$

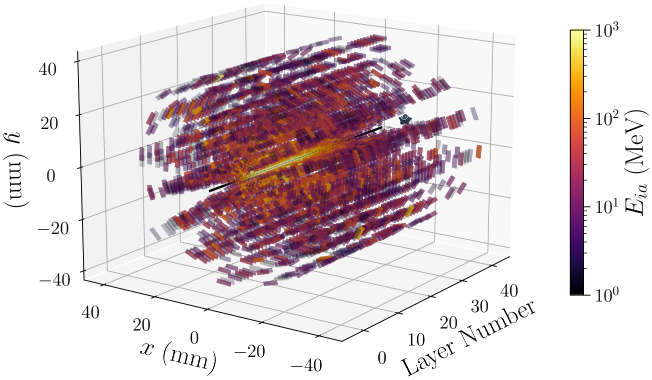


Machine Learning and LHC Event Generation, A. Butter et al. [2203.07460], R. Winterhalder

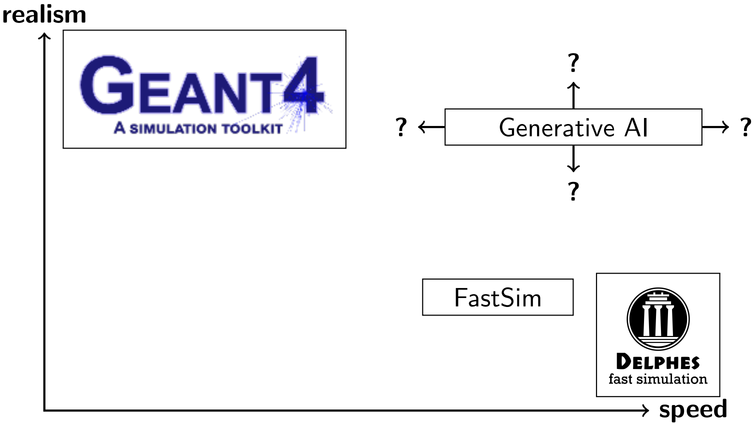
Inverse Problems $p(\text{parameters}|\text{data})$

Detector simulation is computationally expensive.

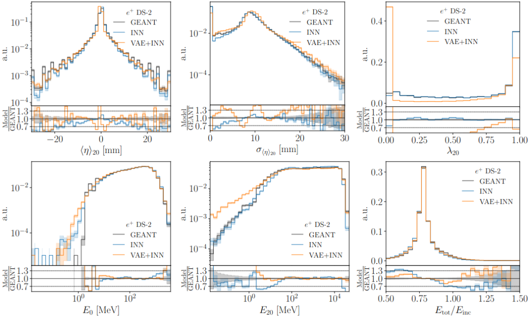




Detector simulation is computationally expensive.



Generative Models are fast and faithful surrogates.



	Batch size	INN 1-photon	INN 1-pion	INN 2-positron
GPU	1	24.79 ± 0.49	24.76 ± 0.35	50.90 ± 0.37
	100	0.385 ± 0.002	0.406 ± 0.003	1.900 ± 0.026
	10000	0.162 ± 0.002	0.191 ± 0.006	exceeding memory
CPU	1	17.48 ± 0.09	18.88 ± 0.33	117.5 ± 1.8
	100	0.827 ± 0.028	1.004 ± 0.047	14.26 ± 0.18
	10000	0.510 ± 0.008	0.719 ± 0.016	15.24 ± 1.36

Generation time per shower in ms.

Ernst, CK et al. [2312.09290]

CaloDiffusion [2308.03876] Normalizing-Flow-based models are very promising! CaloDREAM [2405.09629]
DDPM and CFM models have even better quality, but are slower.

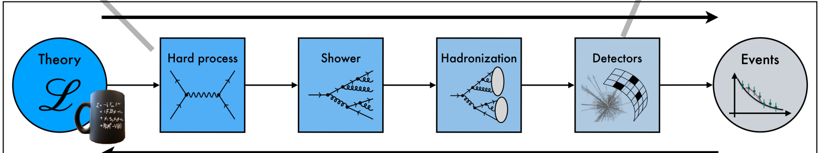
Applications of Generative Models

Event Generation

$p(\text{momenta, angles}|\text{process})$

Detector Simulation

$p(\text{particle shower}|\text{initial condition})$

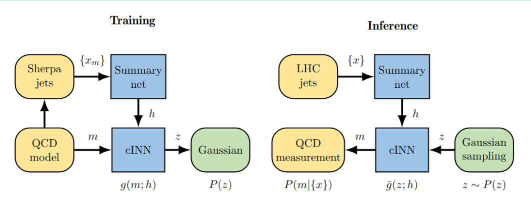


Machine Learning and LHC Event Generation A. Butter et al. [2203.07460], R. Winterhalder

Inverse Problems $p(\text{parameters}|\text{data})$

Inverse Problems: learn $p(\text{parameters}|\text{data})$

A DGM can learn $p(\text{parameters}|\text{data})$ directly.



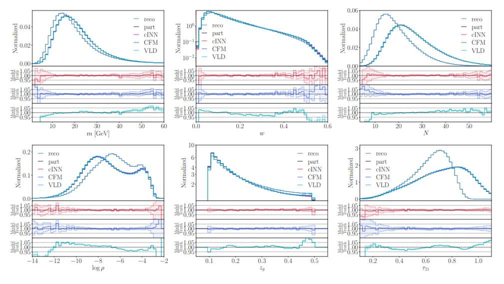
Bieringer et al. [2012.09873, SciPost]

Examples:

CP-observables: Ackerschott et al. [arXiv:2308.00027]

Neutrino momenta: [arXiv:2207.00664, 2307.02405]

Or be used for unfolding detector effects: $p(x_{\text{part}}|x_{\text{reco}})$



[2212.08674, 2310.17037, **2404.18807**]

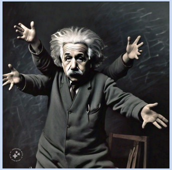
Machine Learning for Particle Physics

This week's plan:

1. Introduction (fits, optimization, and NNs)
2. Regression and Classification
3. Deep Generative Models
   - Normalizing Flows
   - Denoising Diffusion Probabilistic Models (DDPMs)
   - Conditional Flow Matching (CFM)
   - Applications
   - How to evaluate Generative Models
4. Anomaly Detection and Data-Driven Methods

How to evaluate generative models?

In text / image / video generation: "by eye".
⇒ Our brains are incredible good at this task, but it doesn't scale.



imagined with Meta AI.

In high-energy physics: need to find something better!
⇒ We want to correctly cover $p(x)$ of the entire phase space.

1. Can look at histograms of derived features / observables.

⇒ To quantify, we use the *separation power* of high-level feature histograms:

$$S(h_1, h_2) = \frac{1}{2} \sum_{i=1}^{n_{\text{bins}}} \frac{(h_{1,i} - h_{2,i})^2}{h_{1,i} + h_{2,i}}$$

But: this is just a 1-dim projection!

A Classifier provides the "ultimate metric".

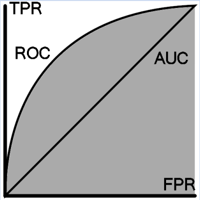
According to the Neyman-Pearson Lemma we have:

- The likelihood ratio is the most powerful test statistic to distinguish two samples.
- A powerful classifier trained to distinguish the samples should therefore learn (something monotonically related to) $w = \frac{p_{\text{data}}}{p_{\text{model}}}$.
- If this classifier is confused, we conclude $\Rightarrow$ $p_{\text{data}}(x) = p_{\text{model}}(x)$

$\Rightarrow$ This captures the full phase space incl. correlations. CK/D. Shih [2106.05285, PRD]

2. Now, the AUC provides a single number to compare different models.

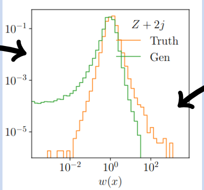
But: are AUCs of different models really comparable?



A Classifier tells us much more about the model.

Failure modes of the model can now be seen in the $w = \frac{p_{\text{data}}}{p_{\text{model}}}$ histogram:

Data manifold over-populated by model:
$\Rightarrow$ mismodeled feature



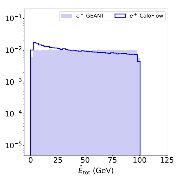
Data manifold not populated by model:
$\Rightarrow$ missed feature

R. Das, CK, et al. [2305.16774, SciPost]

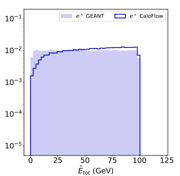
Cluster plots show where events lie in phase space:

figures by B. Schmidthaler / M. Rosendorf

small weights:



large weights:



How to decide which model is closest to the reference: the Multiclass Classifier

A multi-class classifier:

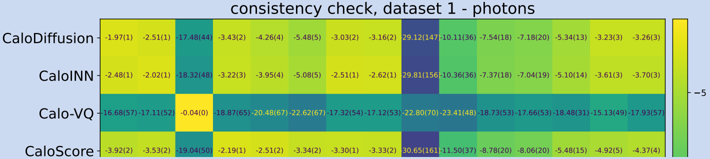
Train on submission 1 vs. submission 2 vs. ... vs. submission n
and evaluate the *log posterior*:

$$L = \langle \log \left(p(x_{\in \text{class } i} | x_{\text{taken from } j}) \right) \rangle \qquad j \in \{\text{submission } k, \text{GEANT4}\}$$

3. As metric: evaluate with GEANT4

Lim et al. [2211.11765, MNRAS]

As cross-check: validate with all submissions j



ÖAW AUSTRIAN ACADEMY OF SCIENCES

HEPHY INSTITUTE OF HIGH ENERGY PHYSICS

Machine Learning for Particle Physics

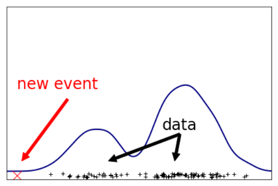
This week's plan:

1. Introduction (fits, optimization, and NNs)
2. Regression and Classification
3. Deep Generative Models
4. Anomaly Detection and Data-Driven Methods

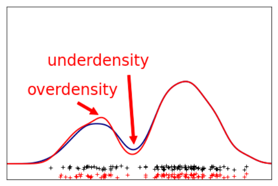
What is Anomaly Detection?

We distinguish different types of Anomaly Detection:

Out-of-Distribution Anomaly Detection



Group Anomalies

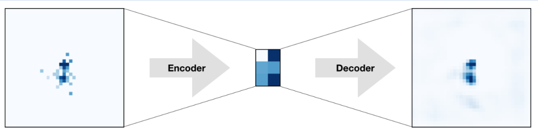


Real-world applications are usually about out-of-distribution events:

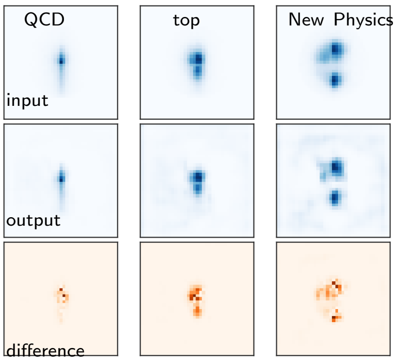
- Finance (credit card fraud, malicious transactions, . . .)
- IT / Network Security
- Medical imaging

Anomaly Detection: Out Of Distribution Data

Train an AutoEncoder on “normal” data:



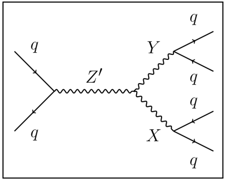
OOD samples will then be harder to reconstruct:

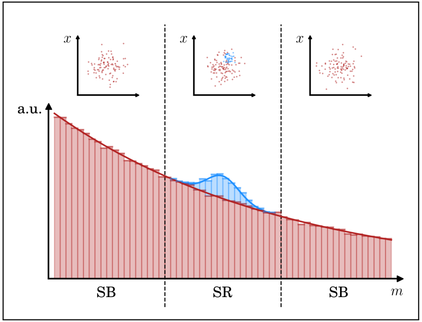


Farina, Nakai, Shih [1808.08992 PRD]

Additional techniques like self-supervision and contrastive learning increase robustness.
Dillon et al. [2301.04660]

Anomaly Detection in Overdensities: Bump Hunts





Assumptions

- signal is localized in m
- background in m is smooth
- $\exists$ additional discriminating features x

Select events with

$$\Rightarrow \frac{p_{\text{data}}}{p_{\text{background}}} \sim \frac{p_{\text{signal}}}{p_{\text{background}}}$$

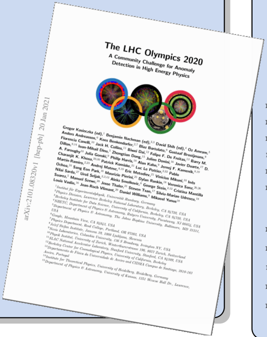
1. Scan Signal Region (SR) across m
2. Perform background fit and obtain p-value for bump.

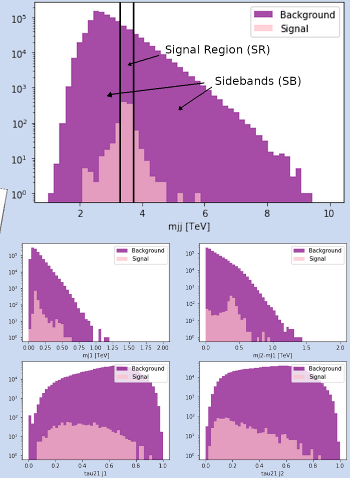
The LHC-Olympics looked at di-jet Resonances.

LHC Olympics R&D dataset:

- 1,000,000 QCD dijet events
- 1,000 signal events $W' \to X(\to qq)Y(\to qq)$
- $m_{W'} = 3.5\text{TeV}$,
  $m_X = 500\text{GeV}$, $m_Y = 100\text{GeV}$
- In SR, $3.3\text{TeV} < m_{JJ} < 3.7\text{TeV}$:
  - 121,352 bg events
  - 772 sg events
- $S/\sqrt{B} = 2.2$

LHCO: G. Kasieczka et al. [2101.08320]

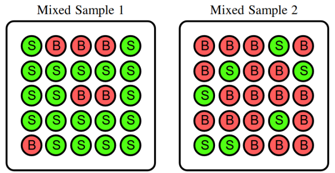




We can get the likelihood ratio using ML: Classifiers.

According to the Neyman-Pearson Lemma we have:

- The likelihood ratio is the most powerful test statistic to distinguish two samples.
- A powerful classifier trained to distinguish the samples should therefore learn (something monotonically related to) this.



- Classification without Labels (CWoLa) learns from mixed samples.
- An optimal classifier is also optimal for distinguishing S from B.

E.M. Metodiev, B. Nachman, J. Thaler, [1708.02949 JHEP]

“Coala Hunting” via midjourney.com ⇒

Simulation-based approaches are model-dependent.

Simulation-based approaches:

- fully supervised:

  train classifier on simulated signal and background

  - depends on quality of simulation
  - high signal model dependence
  - provides upper limit on all approaches

- idealized anomaly detector:

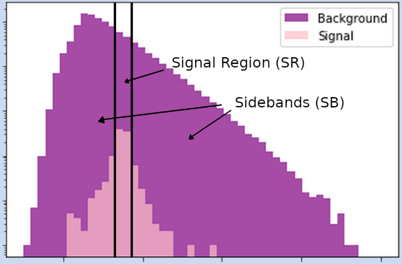
  train classifier on data and simulated background

  - depends on quality of simulation
  - still background model dependent
  - provides upper limit on data-driven anomaly detection

Data-driven approaches are background model-independent.

Anomaly Detection with Density Estimation (ANODE):

- train "outer" density estimator
  $p_{\text{data}}(x|m_{JJ} \in SB)$
- train "inner" density estimator
  $p_{\text{data}}(x|m_{JJ} \in SR)$
- compute
  $\frac{p_{\text{inner}}(x|m_{JJ})}{p_{\text{outer}}(x|m_{JJ})}$ for $m_{JJ} \in SR$
- robust against correlations, but harder learning task.

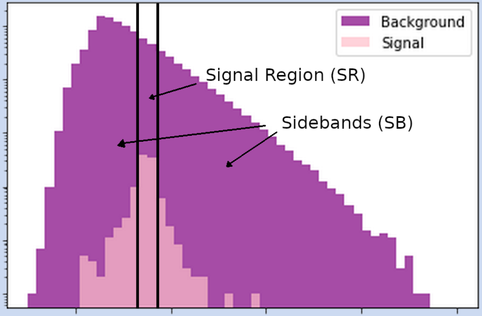


B. Nachman, D. Shih, [2001.04990, PRD]

Anomaly Detection in Overdensities: Bump Hunts

Classifying Anomalies THrough Outer Density Estimation (CATHODE):

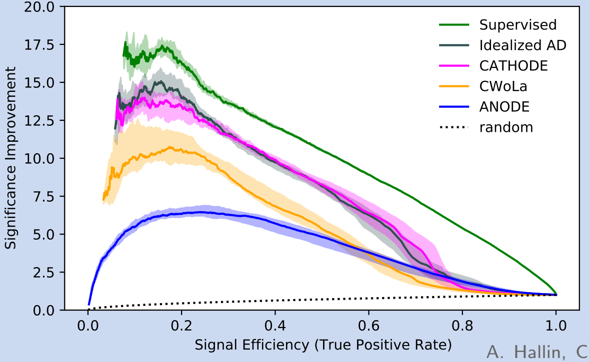
- train "outer" density estimator $p_{\text{data}}(x|m_{JJ} \in SB)$
- sample "artificial" events from $p_{\text{outer}}(x|m_{JJ} \in SR)$
- can also oversample
- train a classifier on these samples vs data



A. Hallin, CK et al. [2109.00546, PRD]

Anomaly Detection in Overdensities: Bump Hunts

Significance Improvement Characteristic = $\text{TPR}/\sqrt{\text{FPR}}$

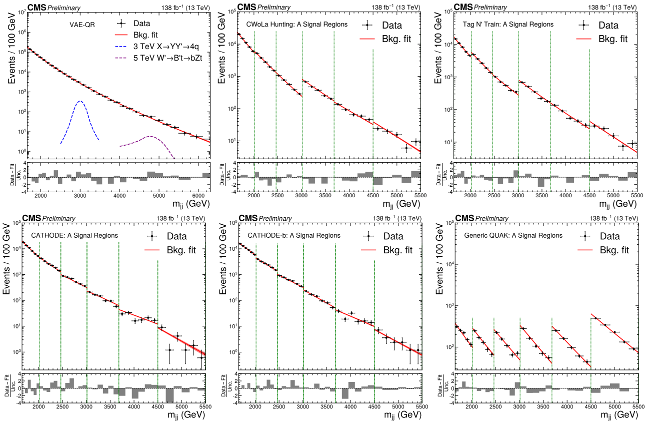


A. Hallin, CK et al. [2109.00546, PRD]

⇒ These strategies are now being explored in ATLAS and CMS.

ATLAS [2005.02983, PRL], CMS [CMS-PAS-EXO-22-026]

Anomaly Detection in deployment: recent CMS results



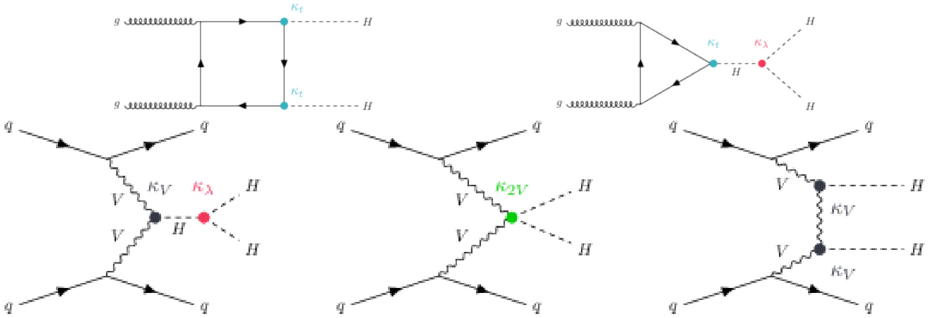
[CMS-PAS-EXO-22-026]

Machine Learning for Particle Physics

This week's plan:

1. Introduction (fits, optimization, and NNs)
2. Regression and Classification
3. Deep Generative Models
4. Anomaly Detection and Data-Driven Methods

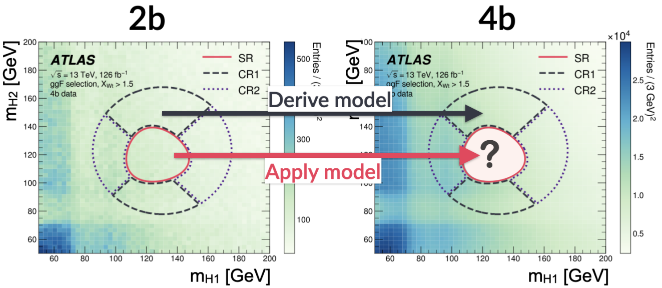
Data-driven methods I: Experimental Background Estimation



ATLAS [arXiv:2301.03212]

Nonresonant Higgs pair production: $ggF\,/\,VBF \rightarrow HH \rightarrow \bar{b}b\bar{b}b$
Upper limits on anomalous couplings.

Data-driven methods I: Experimental Background Estimation

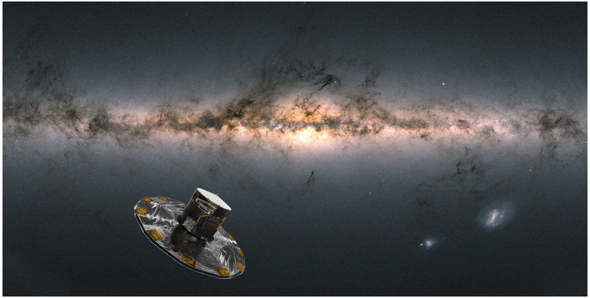


Nicole Hartman [ATLAS Thesis Award Presentation and arXiv:2301.03212]

⇒ Reweighting with a classifier: 7.5% extrapolation uncertainty, ATLAS[arXiv:2301.03212]

⇒ Interpolate with Normalizing Flow: no extrapolation uncertainty, Nicole Hartman, PhD Thesis

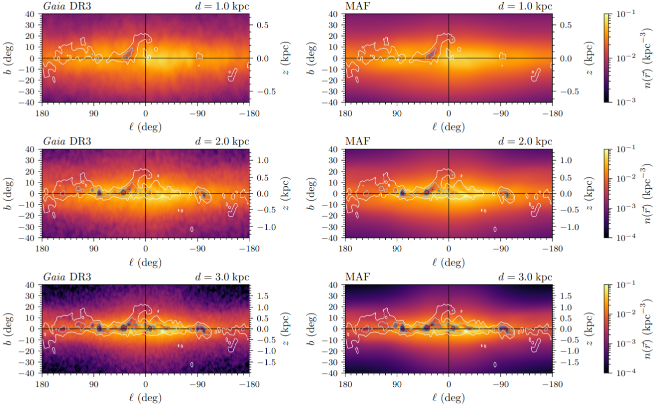
Data-driven methods II: the DM density in the Milky Way from Gaia Data.



[www.esa.int]

- ESA Mission launched in 2013
- measures: position, proper motion, color, and magnitude of stars
- some even have radial velocities and parallax (distance) available
- DR3 has $1.8 \cdot 10^9$ stars, $1.4 \cdot 10^9$ of them have 6D data, DR2 has $1.7(1.3) \cdot 10^9$.

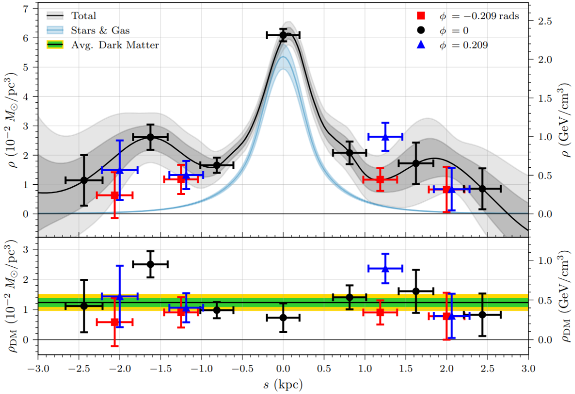
Data-driven methods II: the DM density in the Milky Way from Gaia Data.



Stellar Number Density

Lim et al. [arXiv:2305.13358]

Data-driven methods II: the DM density in the Milky Way from Gaia Data.



Dark Matter Density

Lim et al. [arXiv:2305.13358]

Ressources again

If you have questions, please ask!

This lecture is based on:
⇒ "Modern Machine Learning for LHC Physicists", SS2022 lecture notes of Heidelberg University, arXiv: 2211.01421

Further Reading:

- Summary of HEP-ML papers: "HEPML - Living Review" `https://iml-wg.github.io/HEPML-LivingReview/`
- Tipps for efficient training of NNs: `https://karpathy.github.io/2019/04/25/recipe/`
- About good coding practices in science: `https://goodresearch.dev/`