Machine Learning in Generating **Monte Carlo Events** for High Energy Colliders

Myeonghun Park

(Seoultech)

- Based on **ongoing project** with Raymundo Ramos (KIAS) and Kayoung Ban (Yonsei)
  arXiv:2312.XXXXX

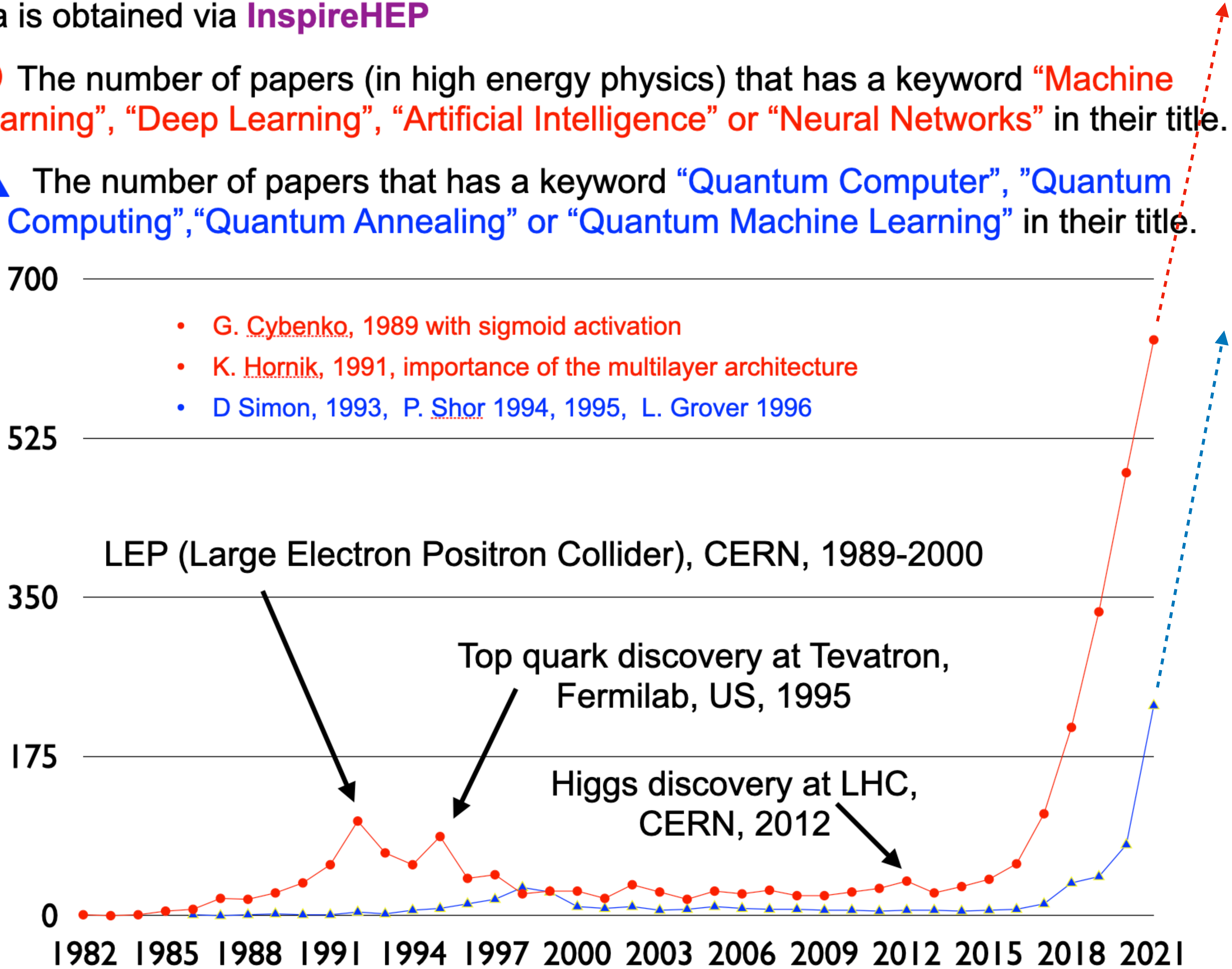
High Energy Physics and Machine Learning Workshop
Hanyang University (2023. 12. 01)

High Energy Physics & Computing frontier

Data is obtained via **InspireHEP**

● The number of papers (in high energy physics) that has a keyword "Machine Learning", "Deep Learning", "Artificial Intelligence" or "Neural Networks" in their title.

▲ The number of papers that has a keyword "Quantum Computer", "Quantum Computing","Quantum Annealing" or "Quantum Machine Learning" in their title.



Communication would be VERY DIFFICULT

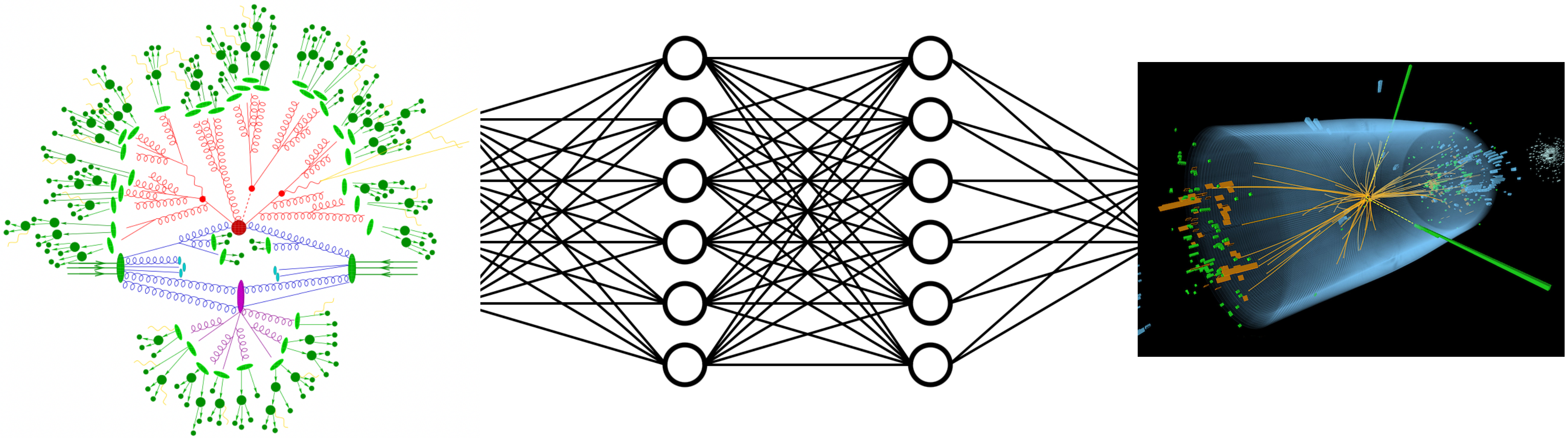
But it is worth it, so let's try

"ongoing" project

- The results here are semi-final.
  - we are checking about more "goodness" of our method

- Comments, contributions are **welcome** !

Theory-Compare-Data



- With our elaborated **theoretical model**,

  1) Get **expectations** from **MC simulations**

  2) Get **data** from **experiments** (e.g. the LHC)

  3) Compare our expectation to data with sophisticated computer **algorithms** (including Machine Learning)

Importance of Theory

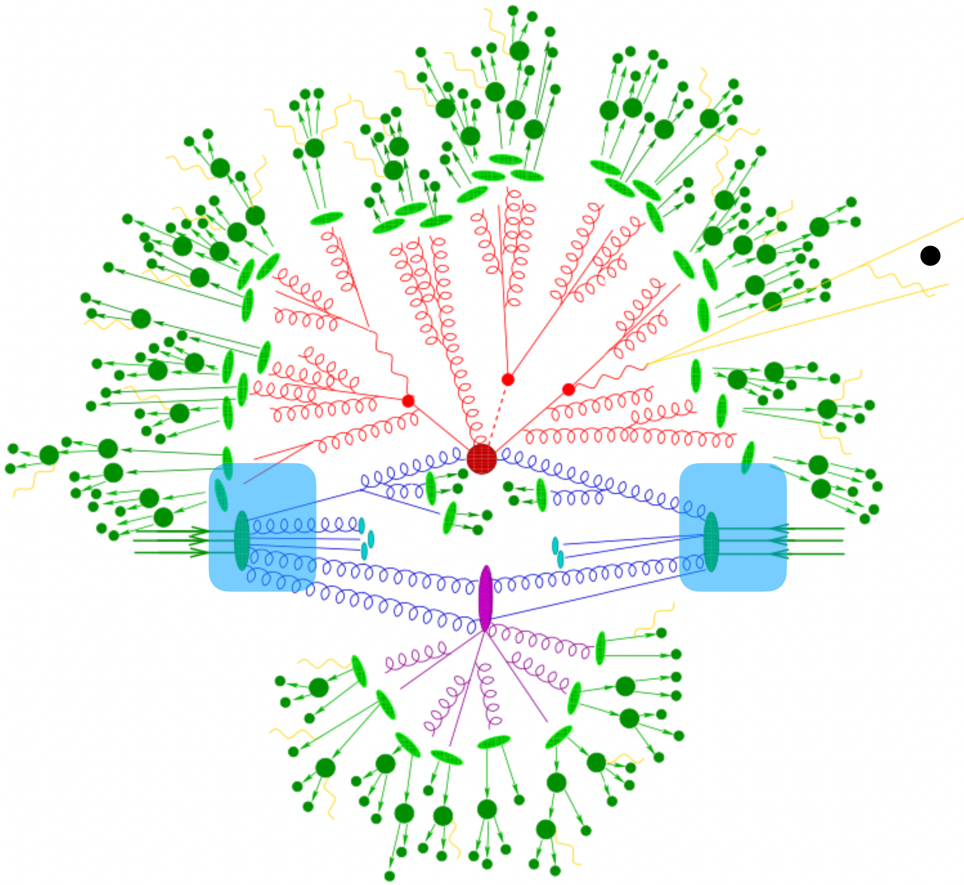
- We need **HUGE "training data"** to feed the **"data hungry" Neural Net**.

- One can dream of "data-driven" machine learning.

  - We cannot guarantee the estimation out of Controlled samples.
  : **NO magic can do "Exploration".**

Monte Carlo Simulation

$$\mathscr{L}_{\text{theory}} \ni -\frac{1}{4}G^a_{\mu\nu}G^{a,\mu\nu} - \frac{1}{4}W^i_{\mu\nu}W^{i,\mu\nu} + \bar{q}\left(i\gamma^\mu D_\mu - m\right)q$$

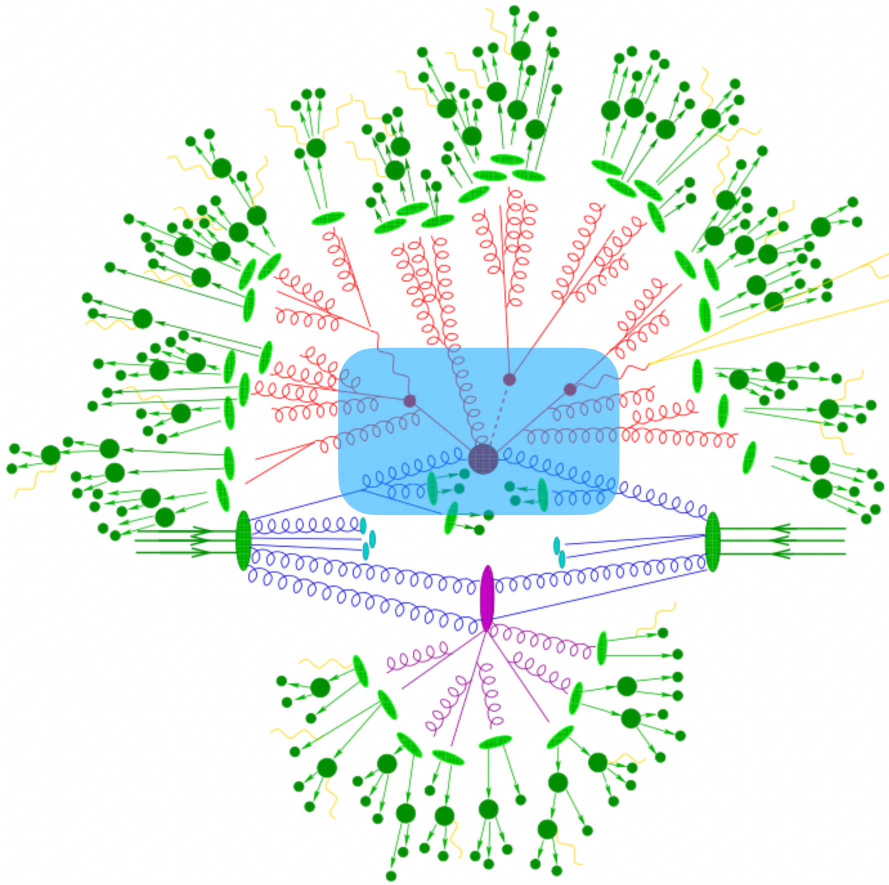


- With $\mathscr{L}_{\text{theory}}$, we simulate a collision process with various Monte Carlo tools.

  - **PDF : parton contributions**
  (e.g. : quark/gluons in protons)

Monte Carlo Simulation

$$\mathscr{L}_{\text{theory}} \ni -\frac{1}{4}G^a_{\mu\nu}G^{a,\mu\nu} - \frac{1}{4}W^i_{\mu\nu}W^{i,\mu\nu} + \bar{q}\left(i\gamma^\mu D_\mu - m\right)q$$

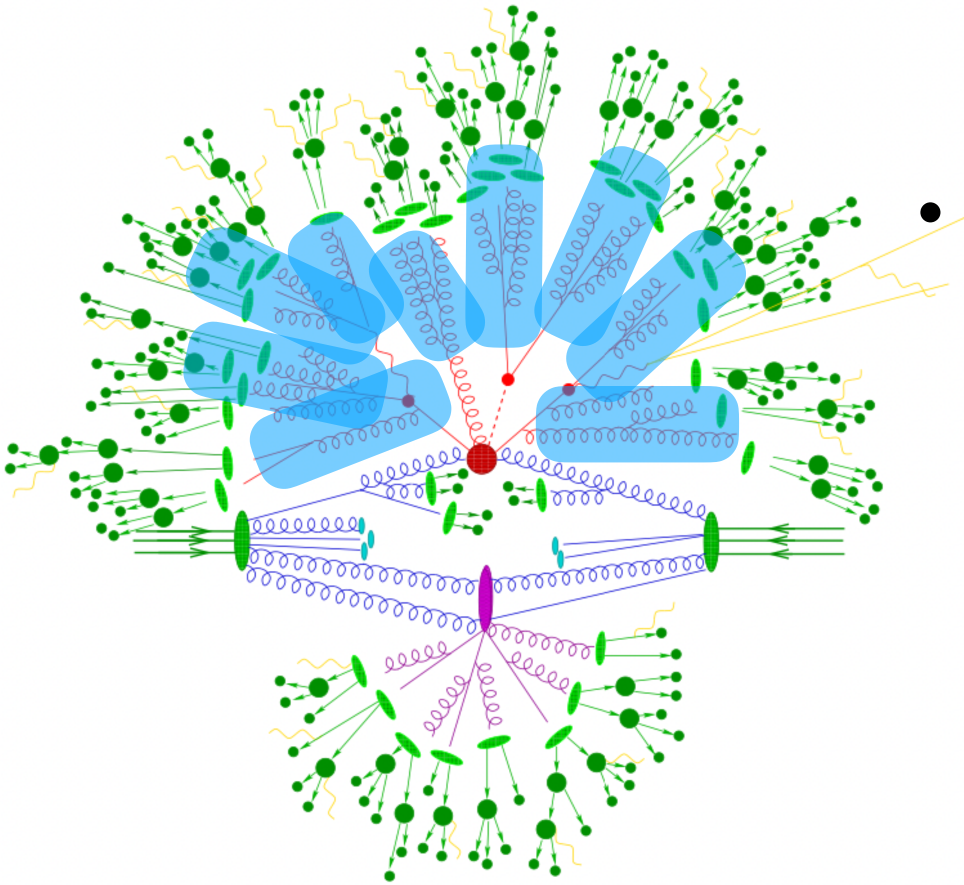


- With $\mathscr{L}_{\text{theory}}$, we simulate a collision process with various Monte Carlo tools.

- **Hard process**
  (e.g: $gg \to t\bar{t} \to bW^+\bar{b}W^- \to b\bar{b}jjjj$)

Monte Carlo Simulation

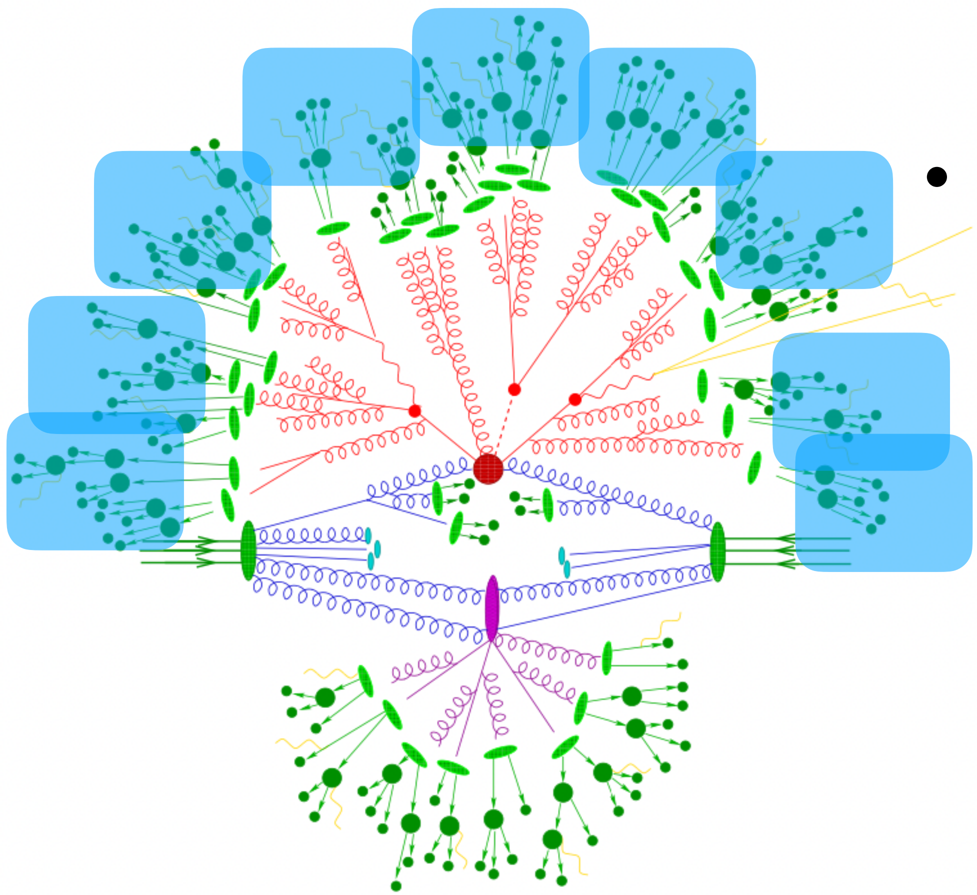
$$\mathscr{L}_{\text{theory}} \ni -\frac{1}{4}G^a_{\mu\nu}G^{a,\mu\nu} - \frac{1}{4}W^i_{\mu\nu}W^{i,\mu\nu} + \bar{q}\left(i\gamma^\mu D_\mu - m\right)q$$



- With $\mathscr{L}_{\text{theory}}$, we simulate a collision process with various Monte Carlo tools.

  \- **Parton Showering**
  (soft radiations of charged particles)

Monte Carlo Simulation



- With $\mathscr{L}_{\text{theory}}$, we simulate a collision process with various Monte Carlo tools.

- **Hadronization** **(approximation)**
(color dress-up : meson, hadron) and **corresponding decays**

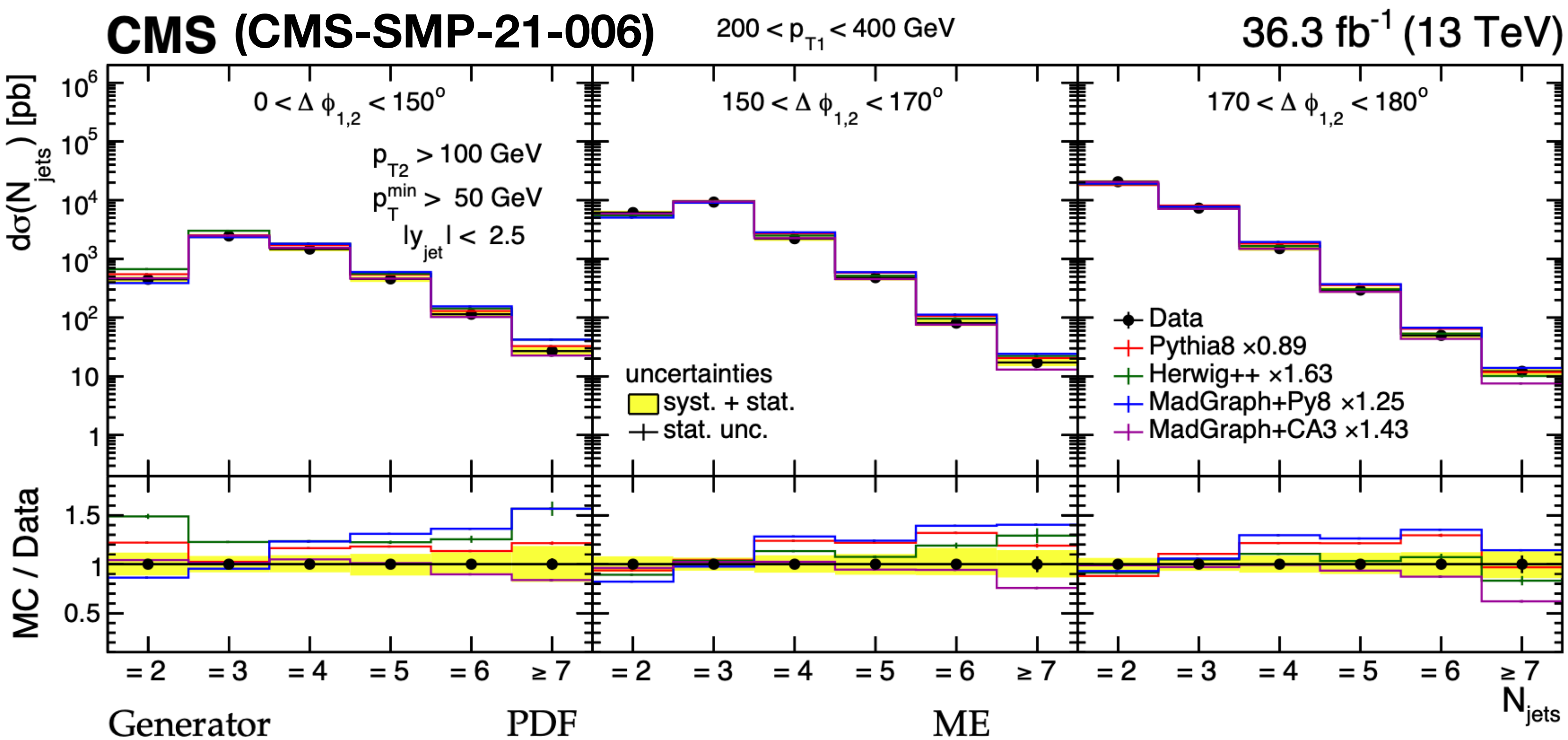
Monte Carlo Simulation

- With $\mathscr{L}_{\text{theory}}$, we simulate a collision process with various Monte Carlo tools.

  - PDF : parton contributions ← **PDF library (LHAPDF)**
  - Hard process ← **Matrix Element (e.g. Madgraph)**
  - Parton Showering
  - **Hadronization**
  (color dress-up : meson, hadron)
  and **corresponding decays**

An approximation (e.g. Lund string: pythia)

Current victory?!



Generator	PDF	ME
PYTHIA8 [23]	NNPDF 2.3 (LO) [25]	LO $2 \to 2$
MADGRAPH+PY8 [4]	NNPDF 2.3 (LO) [25]	LO $2 \to 2, 3, 4$
MADGRAPH+CA3 [4]	PB-TMD set 2 (NLO) [1]	LO $2 \to 2, 3, 4$
HERWIG++ [26]	CTEQ6L1 (LO) [27]	LO $2 \to 2$

- Comparison between the data and expectations from MC.
  - **Jet (correction of hadrons in a small cone) multiplicity**

  Distributions are normalized by the (inclusive) dijet cross section

Into the LOW statistics

- As we get a statistics,
  we are approaching a high energy region
  = **Huge multiplicity.**

$gg \to gg$

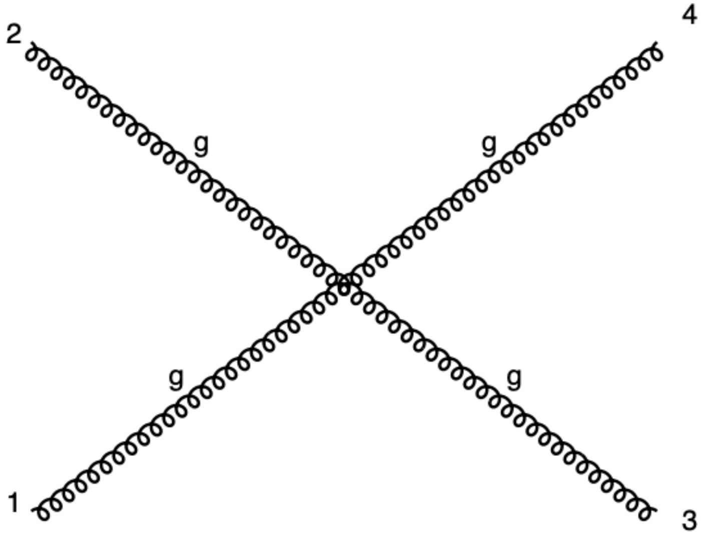


diagram 1 QCD=2, QED=0

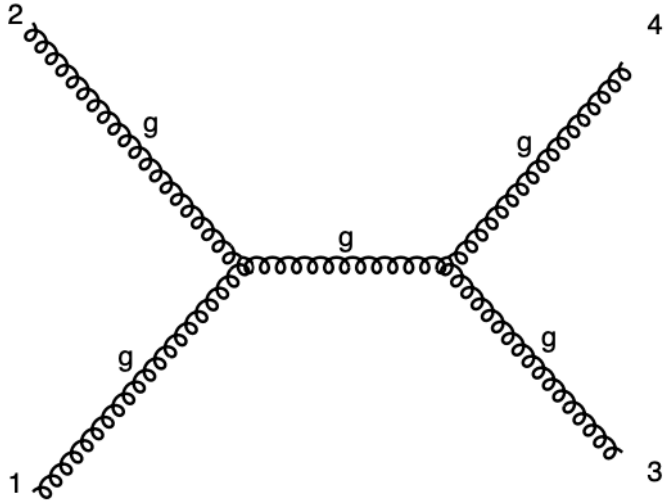


diagram 2 QCD=2, QED=0

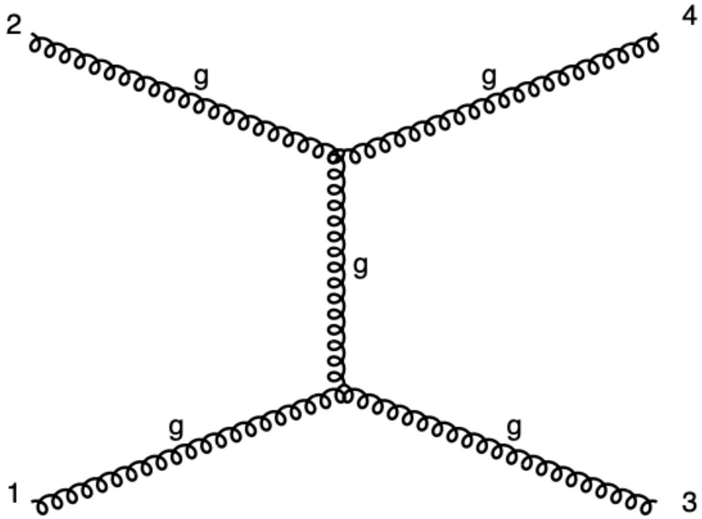


diagram 3 QCD=2, QED=0

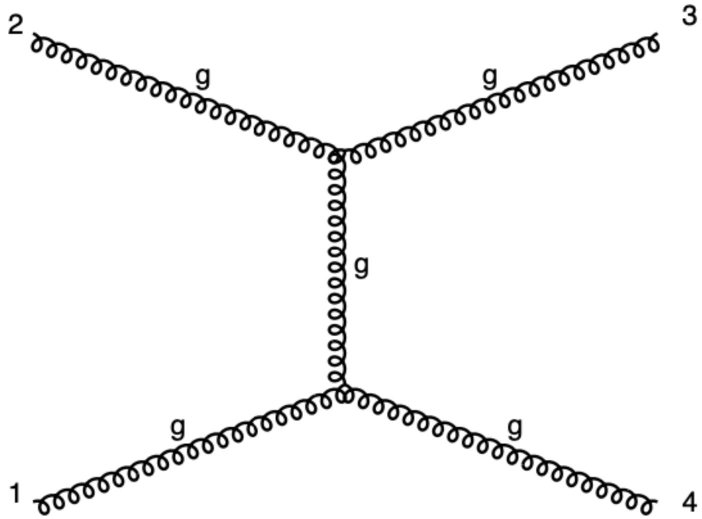
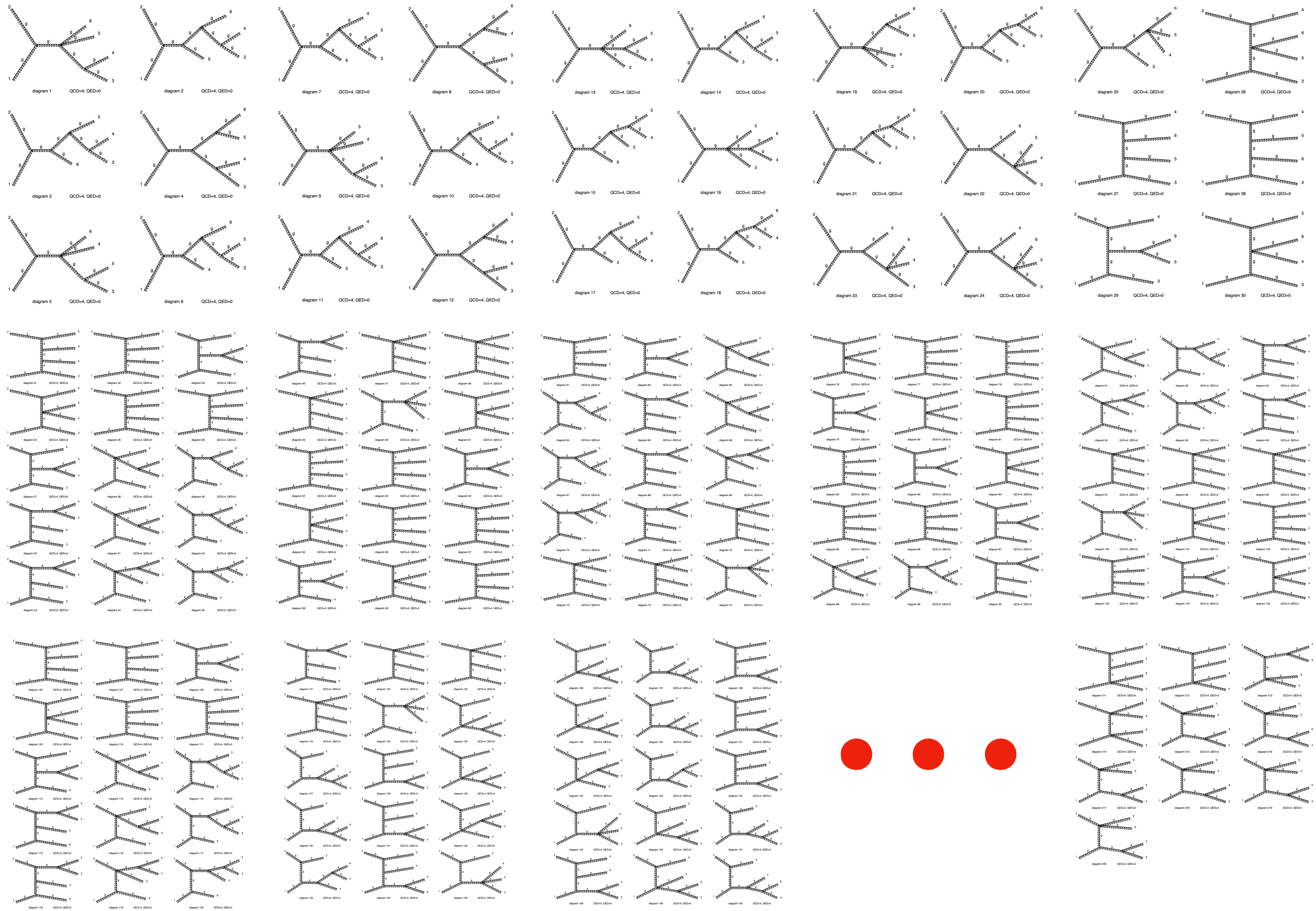


diagram 4 QCD=2, QED=0

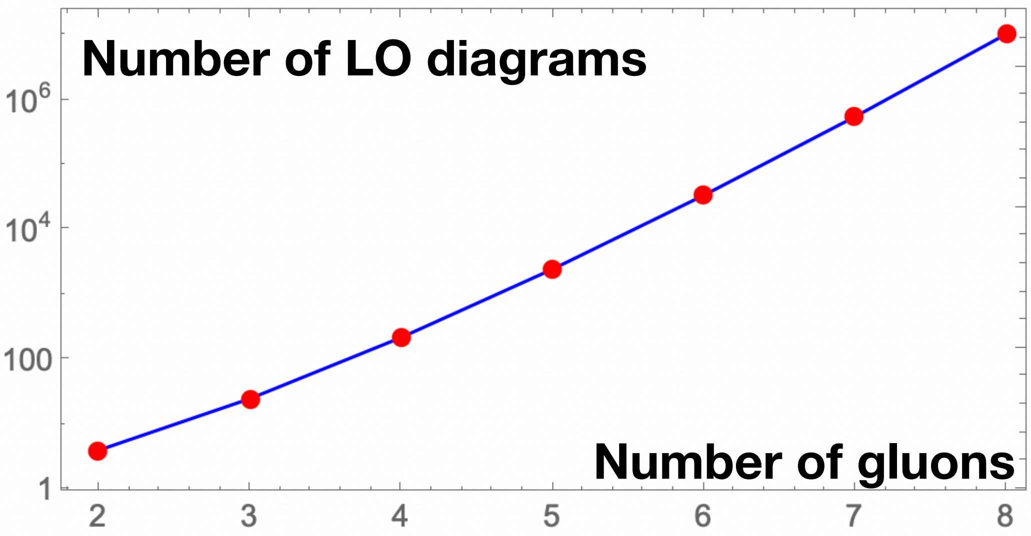
$gg \to gggg$



Multiplicity !

- The production of multi-particles will have **HUGE number** of "feynman" diagrams.

	$gg \to 2g$	$gg \to 3g$	$gg \to 4g$	$gg \to 5g$	$gg \to 6g$	$gg \to 7g$	$gg \to 8g$
Number of LO diagrams	4	25	220	2,485	34,300	559,405	10,525,900



- We should have an alternative way (**what I expect here**)
- This is very hard problem...

THEORY | FEATURE

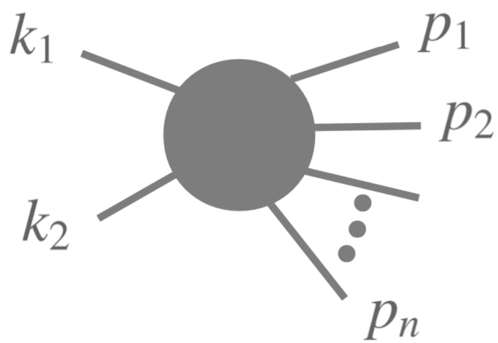
The roots and fruits of string theory

29 October 2018

In the summer of 1968, while a visitor in CERN's theory division, Gabriele Veneziano wrote a paper titled "Construction of a crossing-symmetric, Regge behaved amplitude for linearly-rising trajectories". He was trying to explain the strong interaction, but his paper wound up marking the beginning of string theory.

What led you to the 1968 paper for which you are most famous?

In the mid-1960s we theorists were stuck in trying to understand the strong interaction. We had an example of a relativistic quantum theory that worked: QED, the theory of interacting electrons and photons, but it looked hopeless to copy that framework for the strong interactions. One reason was the strength of the strong coupling compared to the electromagnetic one. But even more disturbing was that there were so many (and ever growing in number) different species of hadrons that we felt at a loss with field theory – how could we cope with so many different states in a QED-like framework? We now know how to do it and the solution is called quantum chromodynamics (QCD).

Our target for  is

- $$\sigma = \frac{1}{2s}\int\prod_{i=1}^{n}\frac{d^3p_i}{(2\pi)^3 2E_i}\delta\left(k_1+k_2-\sum_{i=1}^{n}p_i\right)|M_{fi}|^2\theta_{\rm cut}(p1,\cdots,p_n)$$

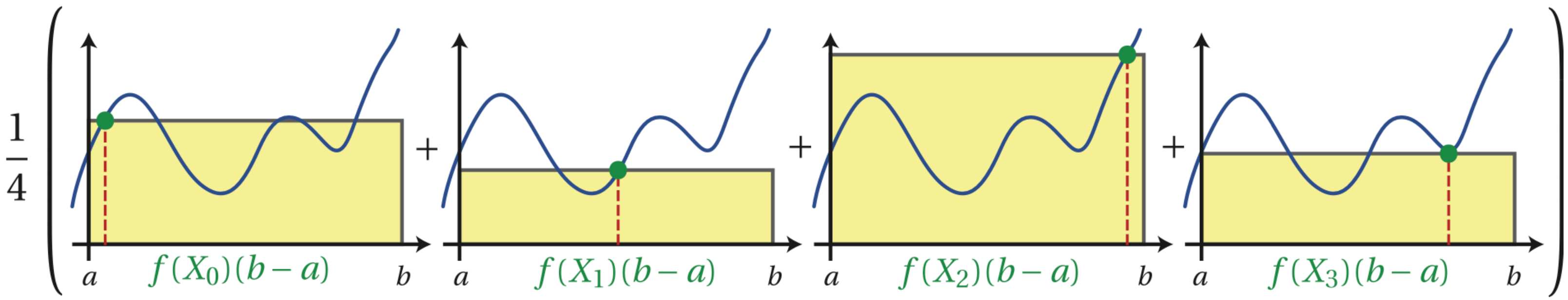
- For an observable $O(p_1,\cdots,p_n)$, we need to calculate the differential distribution of

$$\frac{d\sigma}{dO} = \frac{1}{2s}\int\prod_{i=1}^{n}\frac{d^3p_i}{(2\pi)^3 2E_i}\delta\left(k_1+k_2-\sum_{i=1}^{n}p_i\right)|M_{fi}|^2\theta_{\rm cut}(p1,\cdots,p_n)\delta(O-O(p1,\cdots,p_n))$$

(precise numerical) Integration in high dimensional phase space

Monte Carlo with Importance sampling

- With random N samples according to a uniform Probability Distribution Function $\mathrm{pdf}(x)$ within an integral domain $[a, b]$



$$\mathrm{pdf}(X_i) = \frac{1}{b-a}$$

$$\langle F^N \rangle = \frac{1}{N} \sum_{i=0}^{N} \frac{f(X_i)}{\mathrm{pdf}(X_i)} = (b-a)\frac{1}{N} \sum_{i=0}^{N} f(X_i)$$

$$\rightarrow E[\langle F^N \rangle] = (b-a)\frac{1}{N} \sum E[f(X_i)] = (b-a)\frac{1}{N} \sum \int_a^b f(X_i)\mathrm{pdf}(x)dx = \int_a^b f(x)dx$$

$$\sigma[\langle F^N \rangle] = \frac{1}{\sqrt{N}} \sqrt{\sum_{i=0}^{N-1} \sigma^2 \left[\frac{f(X_i)}{\mathrm{pdf}(X_i)} \right]}$$

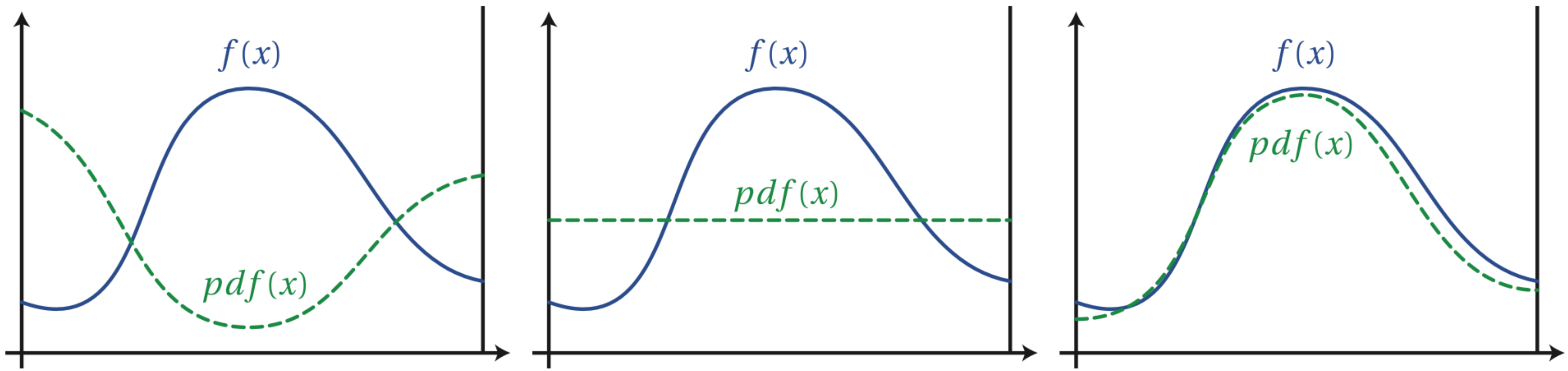
Random sampling

Importance sampling to reduce a variance

Importance sampling

- If we sample PDF $\propto f(x)$, we can reduce a variance

$$\sigma[\langle F^N \rangle] = \frac{1}{\sqrt{N}}\sqrt{\sum_{i=0}^{N-1}\sigma^2\left[\frac{f(X_i)}{\mathrm{pdf}(X_i)}\right]} \rightarrow \frac{1}{\sqrt{N}}\sqrt{\sum_{i=0}^{N-1}\sigma^2\left[\frac{f(X_i)}{c\,f(X_i)}\right]} \simeq \frac{1}{\sqrt{N}}\sqrt{\sum_{i=0}^{N-1}\sigma^2\left[\frac{1}{c}\right]} \equiv \frac{1}{\sqrt{N}} \times 0$$

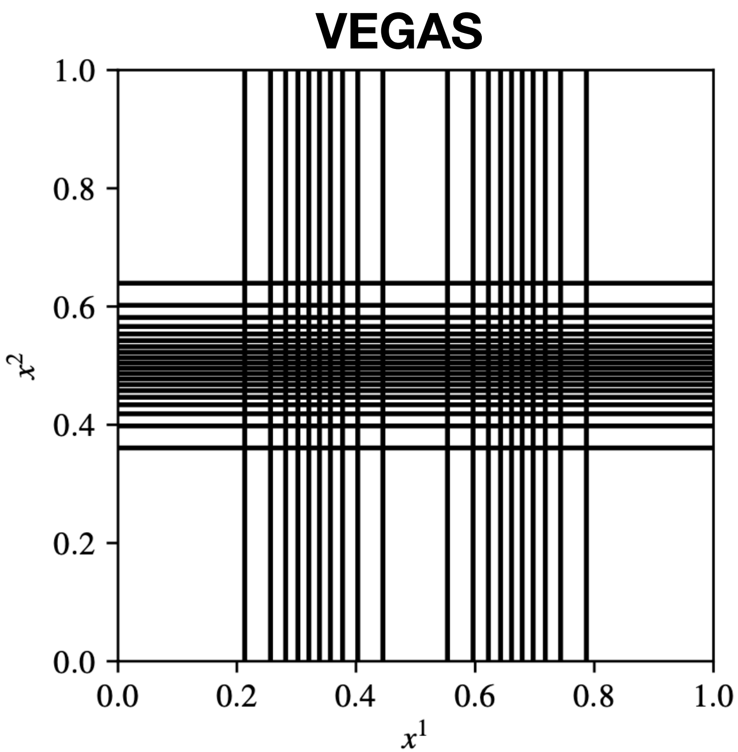


Bad PDF **Uniform PDF** **Good PDF**

- When we don't know a function $f(x)$ at all, how can we estimate a good PDF ?

Traditional method...

- **Stratified Sampling**: Divide domain **into sub-domains**.
  For example, if we divide the domain into N divisions,
  $\sigma \propto \frac{1}{N}$ instead of $\sigma \propto \frac{1}{\sqrt{N}}$



- "Classic" VEGAS: **Adaptive** importance sampling, since 1977

  Recently, there is an update, VEGAS+
  J.Comput.Phys. 439 (2021) 110386

Neural Net as a good estimator

- Due to the universal approximation theorem, **NN serves as a bonafide function approximator.**

- Design a process where the accuracy of NN becomes proportional to our interests in sampled regions:

  - spend, relatively, more time sampling regions of iterests

  - enough time for low importance region

Importance sampling with Machine Learning

- In fact, we already solved a similar problem in our previous study arXiv:2207.09959,

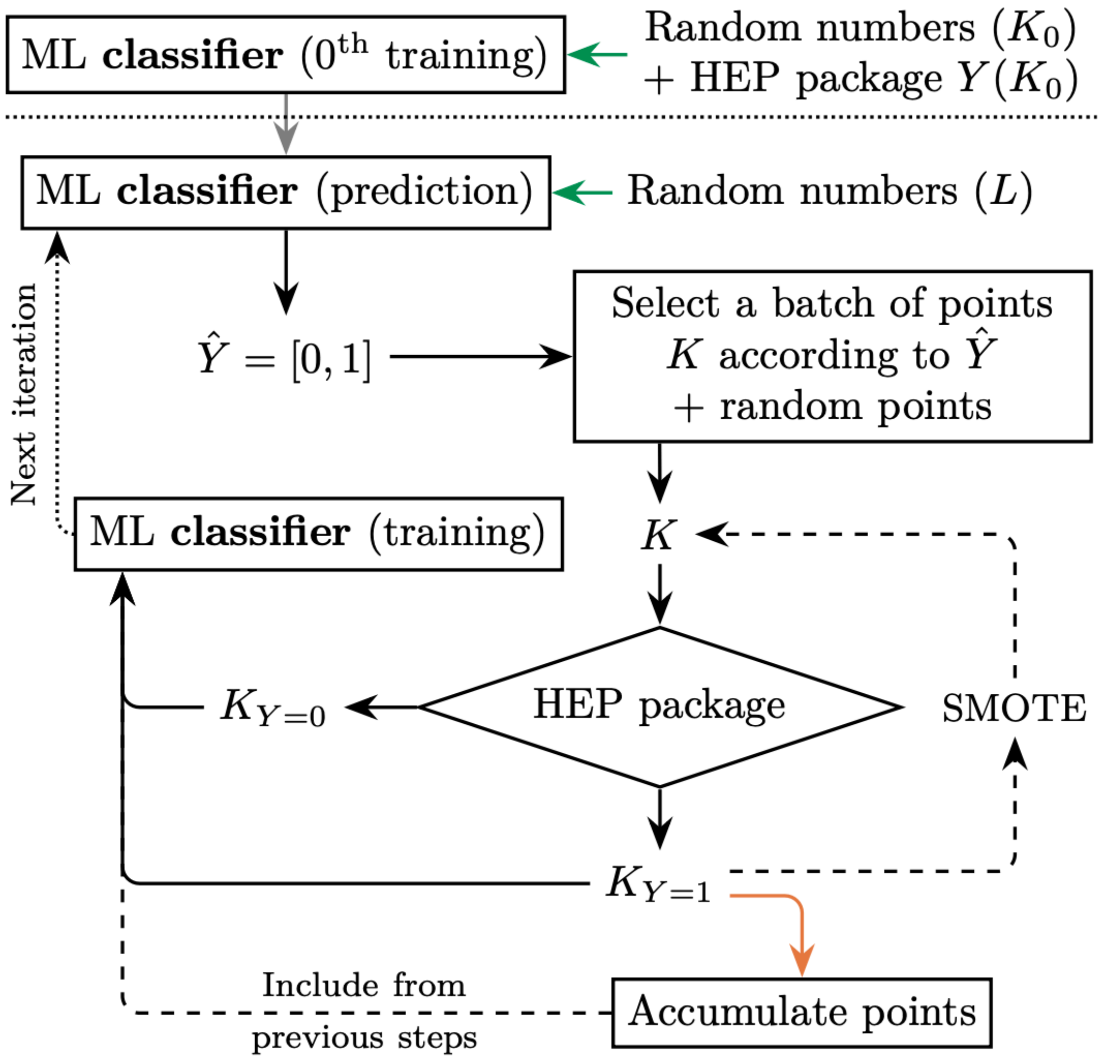
  "Exploration of Parameter Spaces Assisted by Machine Learning" (Computational Physics Communication, v293, 2023)

- Let me explain what we have done....

- The following example is the case **when** the function $f(\vec{x})$ **is "computationally expensive".**

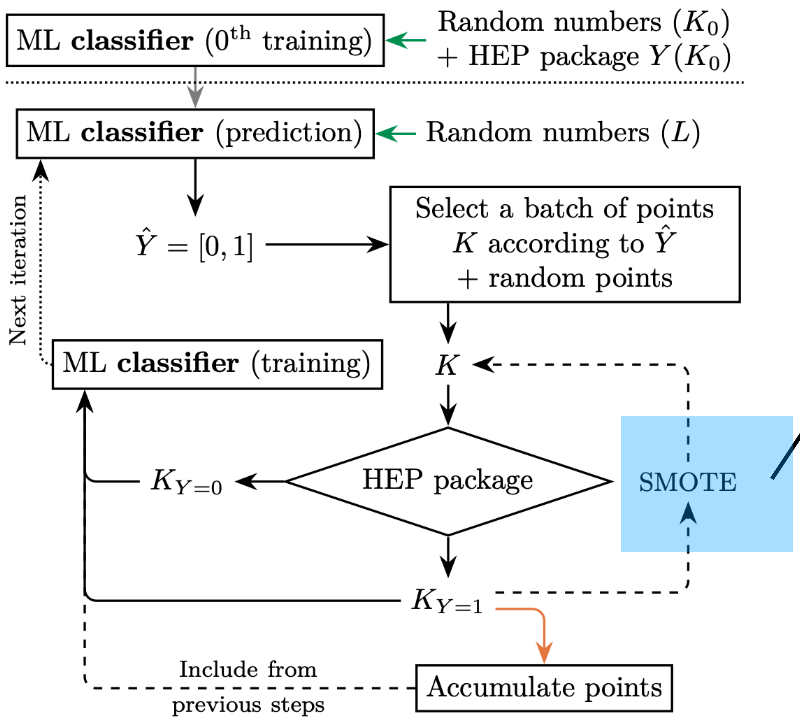
  : Let ML to approximate $f(\vec{x})$

Classifier type ML



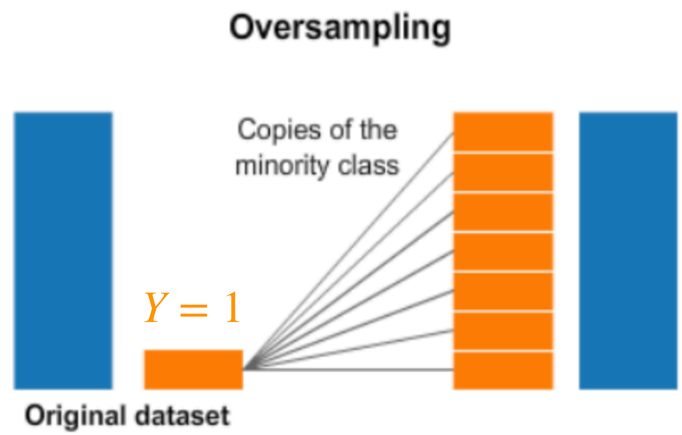
- We use a classifier type to predict the class of a points e.g: $\hat{Y} =$ 0 (reject) or 1 (accept)

Classifier type ML

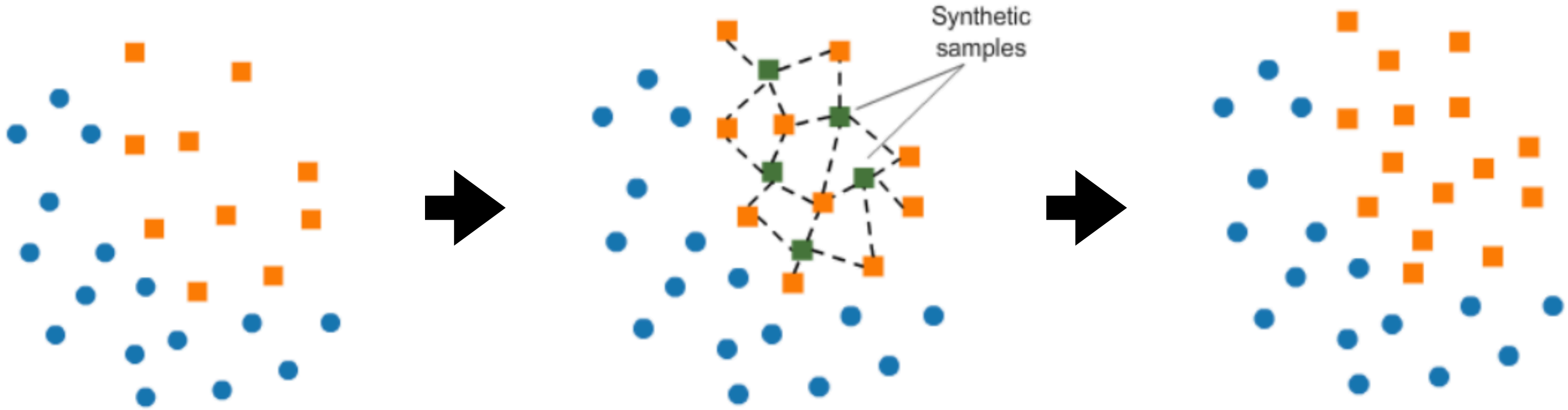


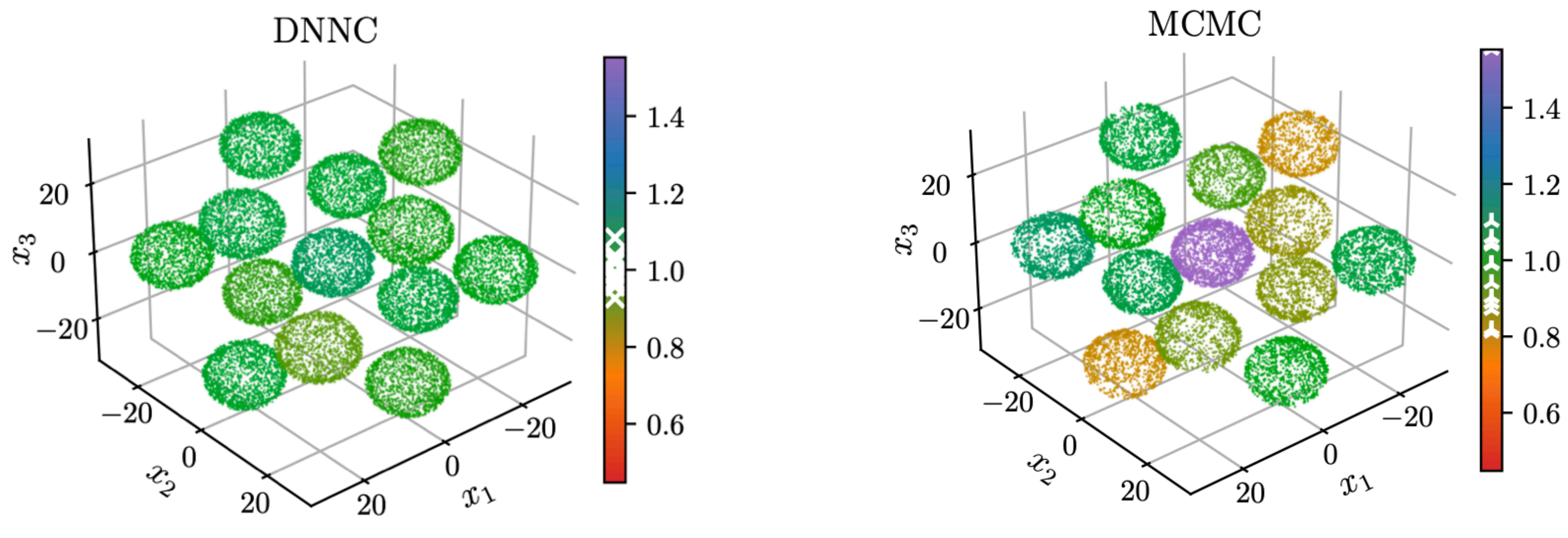
SMOTE (arXiv:1106.1813)

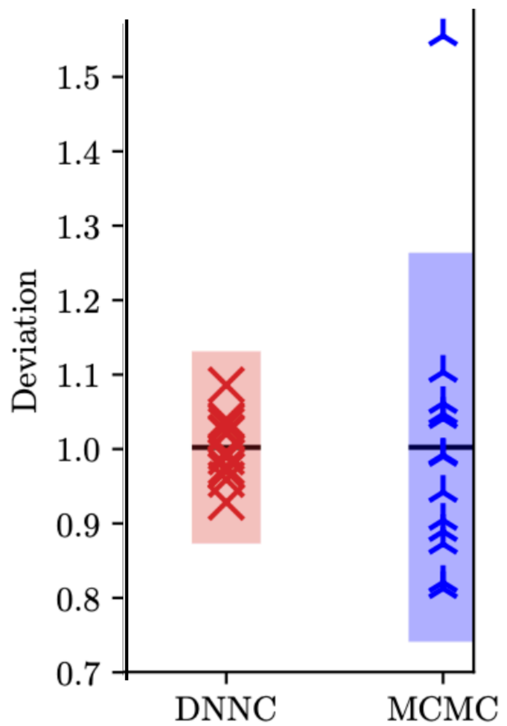
- **S**ynthetic **M**inority **O**versampling **T**echnique is used to "gather" more samples



using k-nearest neighbor algorithm, "make" samples





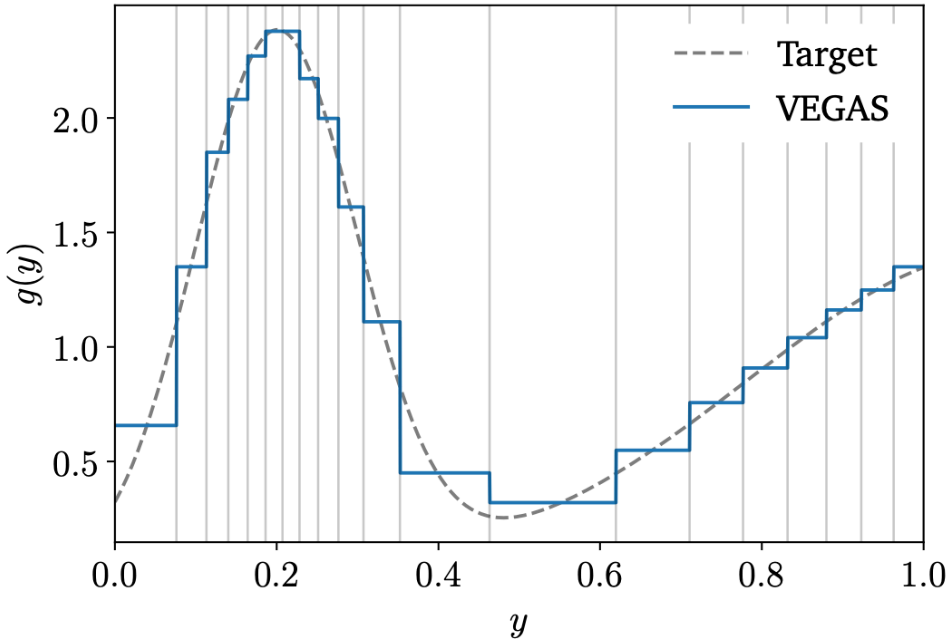


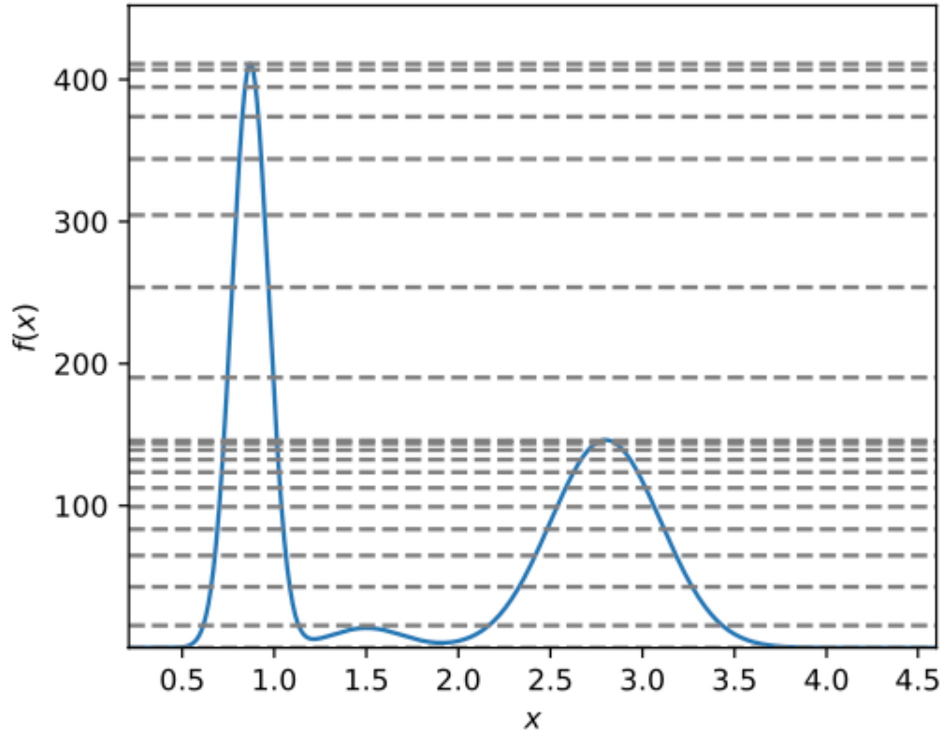
- Points are $\mathscr{L} > 0.9$ conditions.
- With $x_i \in [-10\pi, 10\pi]$, there are 13 cell.
- The "**deviation**" is the ratio of **a population in each "cell"** over an average population

Utilizing our ML algorithm for an importance sampling in an integration

Two integration methods

from MadNIS (Theo Heimel et.al. arXiv:2311.01548)



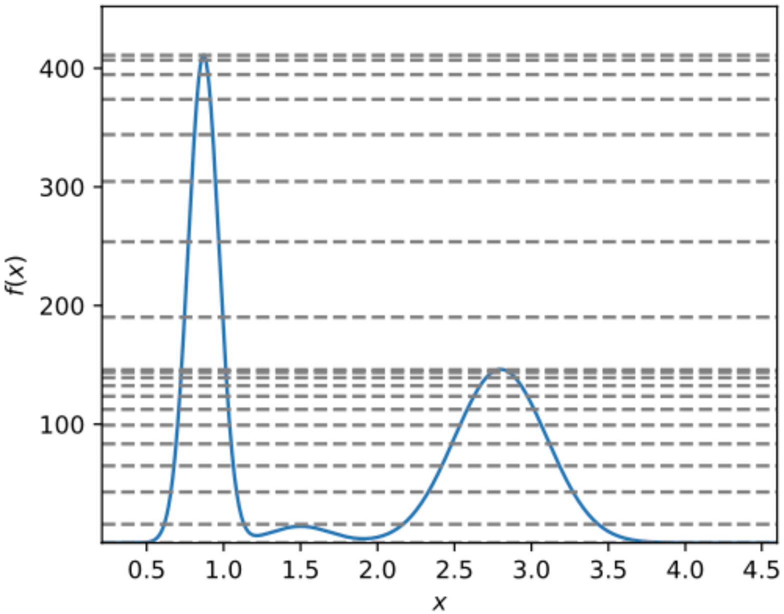


- **Riemann** Integration.
- **Lebesgue** Integration

- **Lebesgue integral** is more efficient(?) and **broad(!)**

  - A classical example: $f(x) = \begin{cases} 1 & \text{if } x \in P \\ 0 & \text{if } x \in Q \end{cases}$

Our approach: Lebesque



- **Divide the space of integrand** (**classes**)

$$\Phi_j = \{\vec{x} \,|\, l_j < f(\vec{x}) \leq l_{j+1}\}$$

- The integral : $I_\Phi\left[f(\vec{x})\right] = \int_\Phi \mathrm{d}^d x\, f(\vec{x}) = \sum_{j=1}^{n} \int_{\Phi_j} \mathrm{d}^d x\, f(\vec{x}) = \sum_{j=1}^{n} V_{\Phi_j} \langle f \rangle_{\Phi_j}$

  V_{Φ_j} : Volume of Φ_j .

- We recast the **problem of integration** → **classification problem**

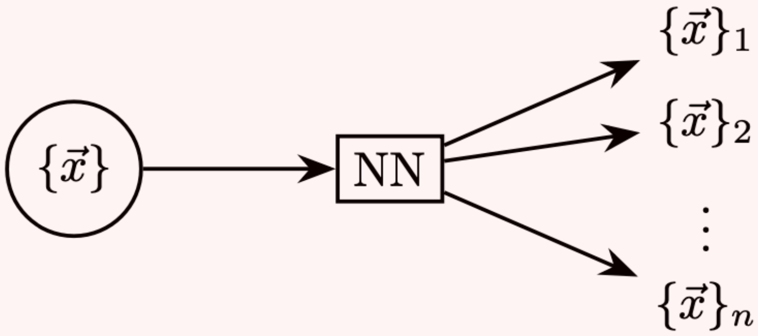
Monte Carlo with ML

$$I_\Phi[f(x)] = \int_\Phi \mathrm{d}^d x\, f(x) = \sum_{j=1}^{n} \int_{\Phi_j} \mathrm{d}^d x\, f(x) = \sum_{j=1}^{n} V_{\Phi_j} \langle f \rangle_{\Phi_j}$$

- here, if we can "correctly" decide $\vec{x} \in \Phi_j$, we can calculate

$$V_{\Phi_j} \simeq \frac{N_j}{N_{\mathrm{total}}} V_{\mathrm{total}}\,,\quad \langle f \rangle_{\Phi_j} \simeq \frac{1}{N_j} \sum_{i=1}^{N_j} f(x_i)$$ with large sample N_{total}

- **It is crucial to estimate V_{Φ_j}.** With previous an iterative ML algorithm,



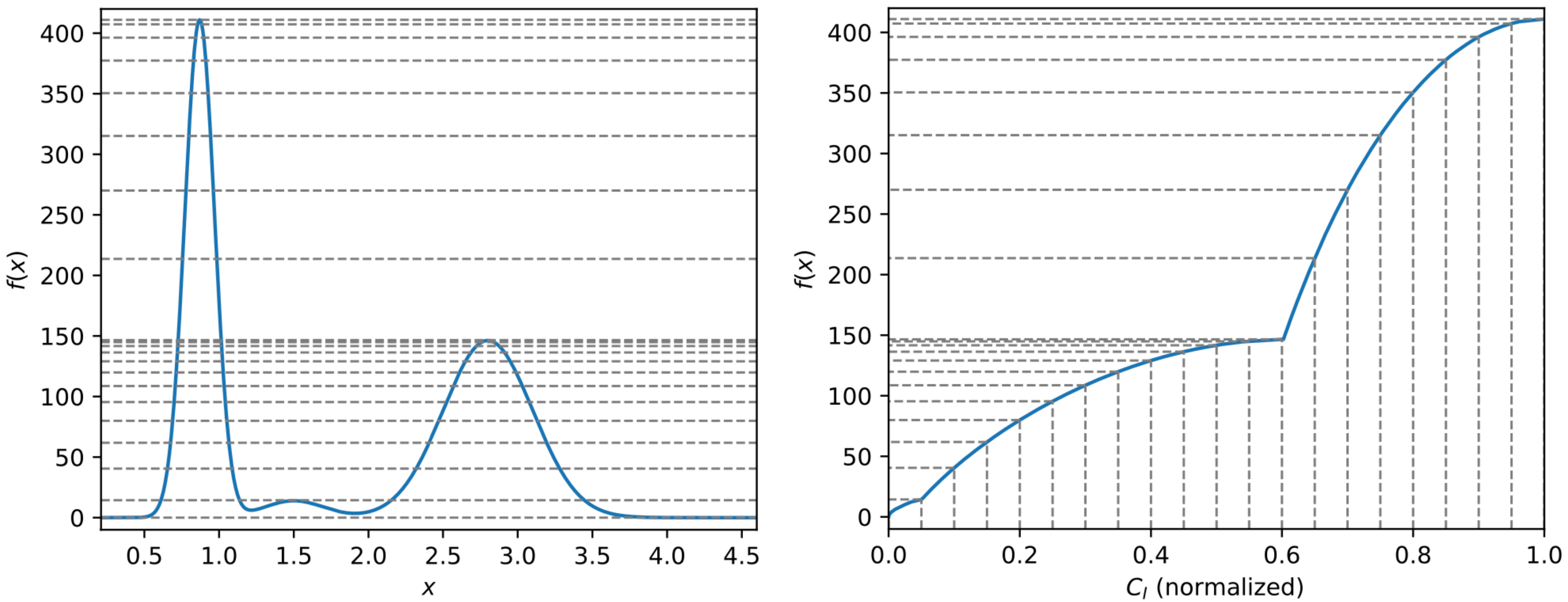
1. Train NN with a sample of points and function value.
2. Get predictions from the NN for a larger sample of new points.
3. Use function to correct wrong predictions.
4. Go back to training until NN is accurate enough.

Deciding division of $f(\vec{x})$

- We have a freedom to decide divisions on $f(\vec{x})$

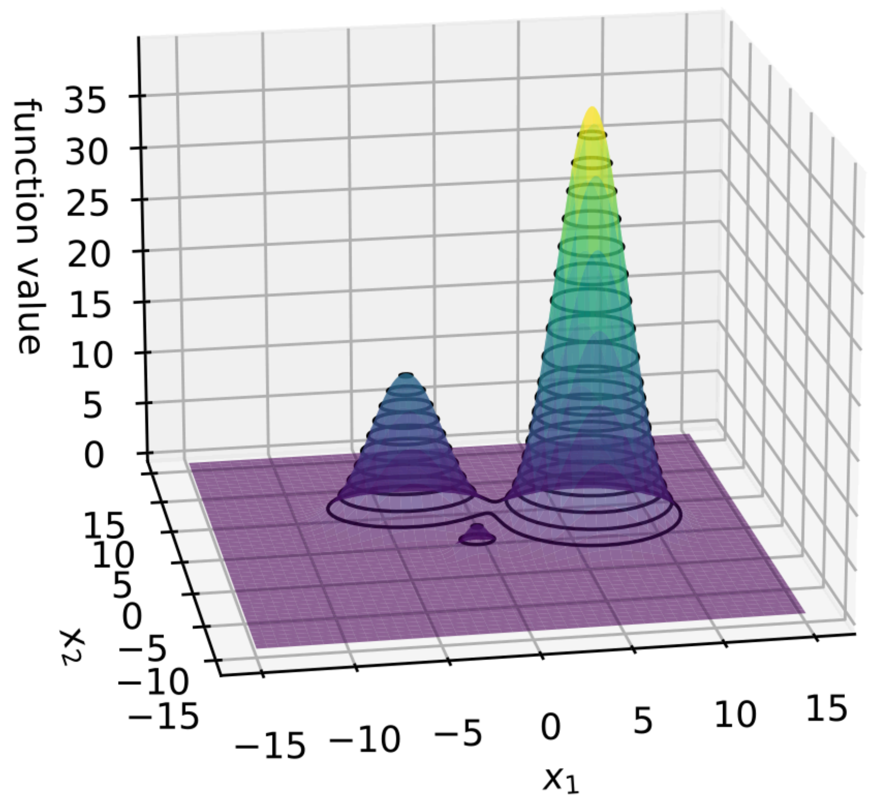
  - We can have divisions with equal contributions to the integral as

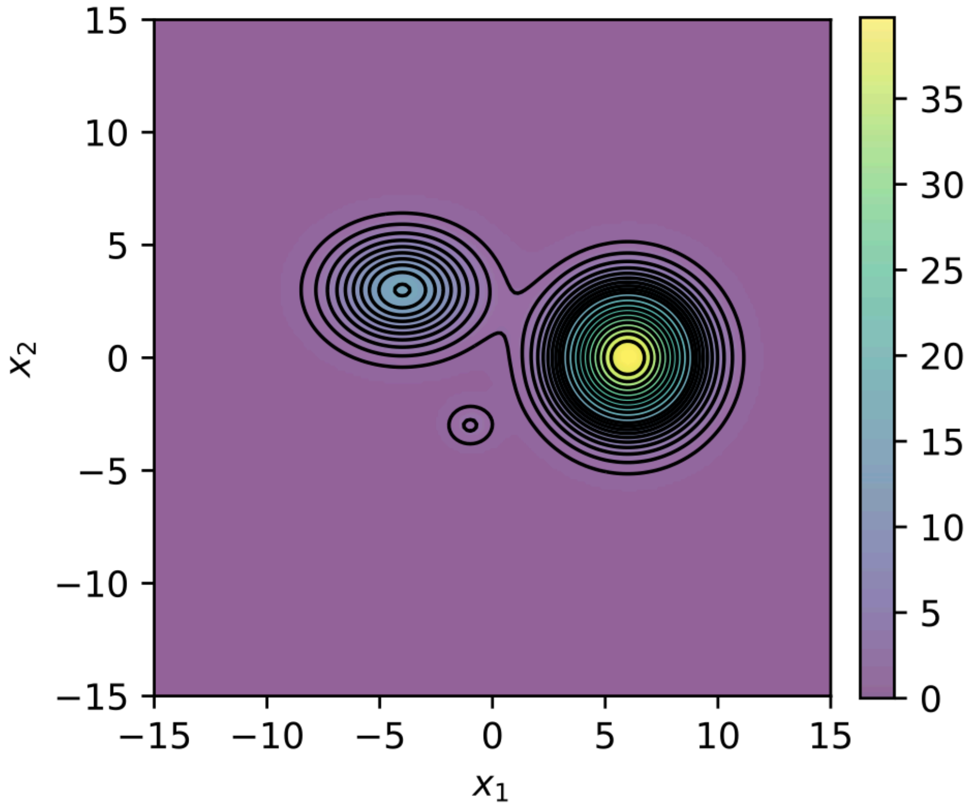
  $\langle f \rangle_{\Phi_i} V_{\Phi_i} \equiv \mathrm{const}$ by simply choosing $K_j \propto I_{\Phi_j}[f(x)]$ from each section Φ_j



example 1

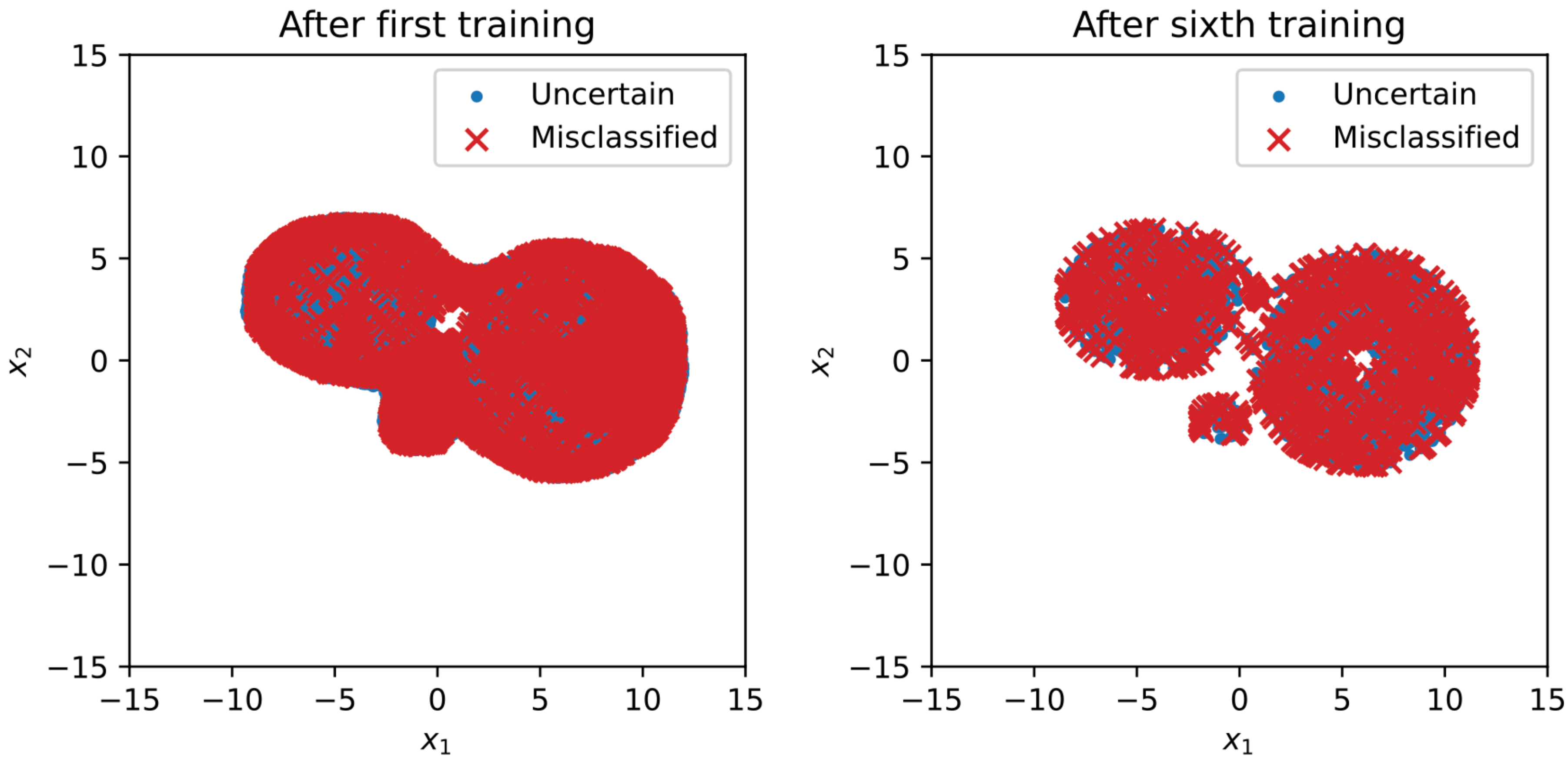
- Two channels 3D functions of multiple peaks





example 1

20 regions with similar contribution to value of integral



After sixth training step: above 99% accuracy (100 000 test points).

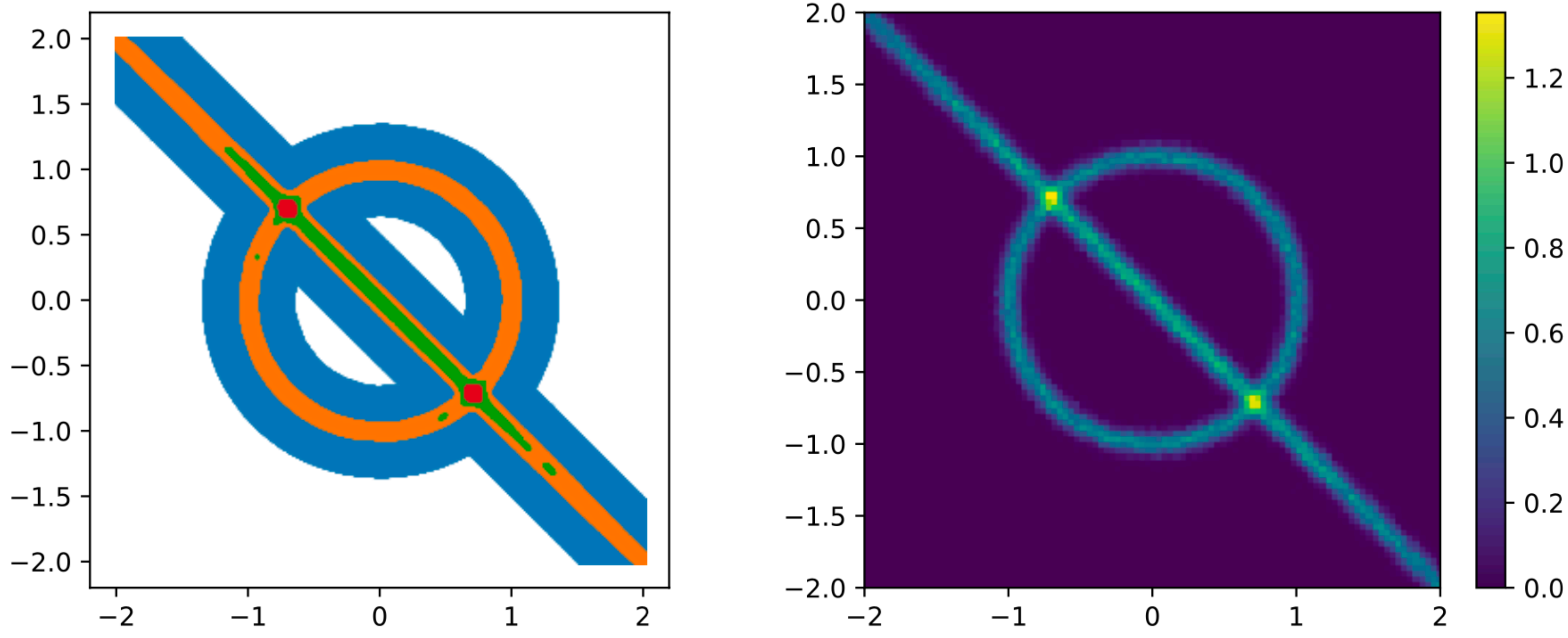
example 2

- Two channels 3D functions of $f(x, y) = f_1(x, y) + f_2(x, y)$

$$f_{\text{no-parking}}(x) = \frac{1}{2}\left[f_{\text{ring}}(x) + f_{\text{line}}(x)\right]$$

$$f_{\text{line}}(x) = N_1 \exp\left[-\frac{(\tilde{x}_1 - \mu_1)^2}{2\sigma_1^2}\right] \exp\left[-\frac{(\tilde{x}_2 - \mu_2)^2}{2\sigma_2^2}\right]$$

$$f_{\text{ring}}(x) = N_2 \exp\left[-\frac{\left(\sqrt{x_1^2 + x_2^2} - r_0\right)^2}{2\sigma_0^2}\right]$$



example 3

- $\infty - \infty = \text{finite}$: We are testing "fine-tuning" function of

$$f_1(x_1, x_2) = g(x_1; 5, 2)g(x_2; 0, 2)$$

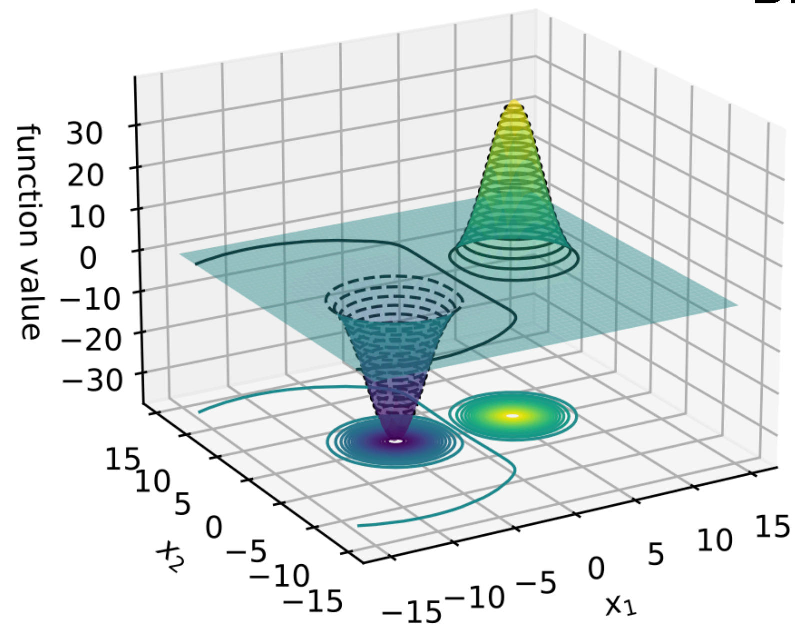
$$f_2(x_1, x_2) = g(x_1; -5, 2.1)g(x_2; 0, 2.1)$$

with $g(x; m, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[\frac{(x-m)^2}{2\sigma^2}\right]$

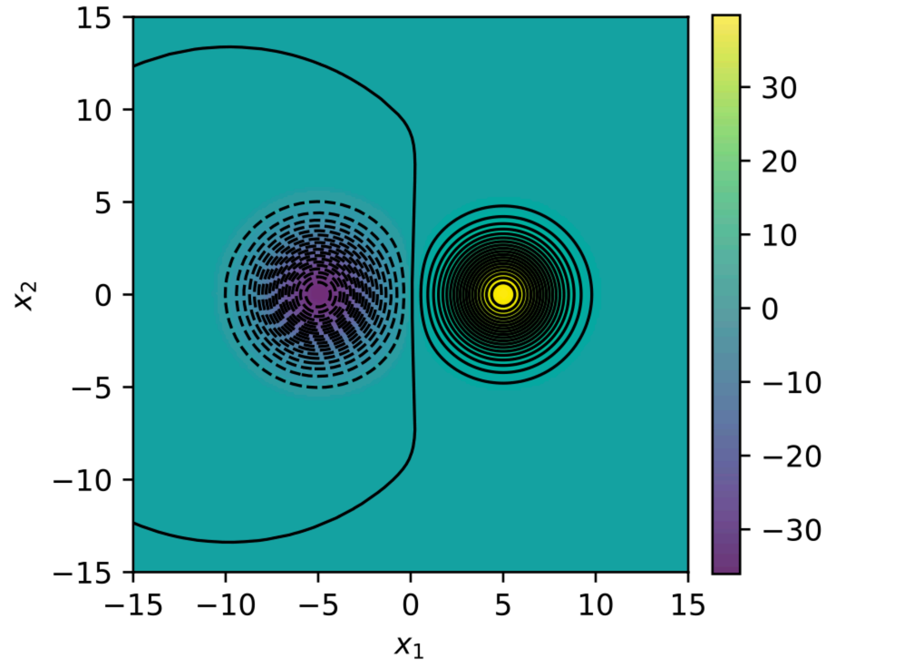
$$f_3(x_1, x_2) = g(x_1; 0, 3)g(x_2; 0, 3)$$

$$f(x_1, x_2) = 1000\left[f_1(x_1, x_2) - f_2(x_1, x_2)\right] + f_3(x_1, x_2)$$

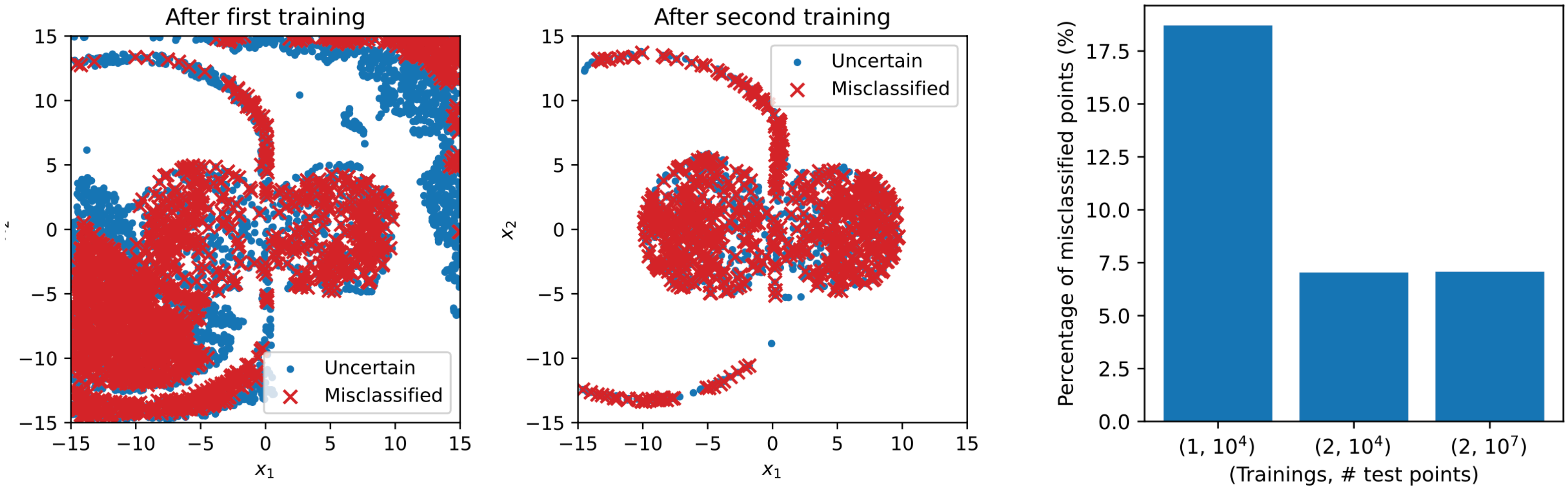
$$\int f(\vec{x})dxdy \simeq 1$$



Divisions for equal "absolute" contributions



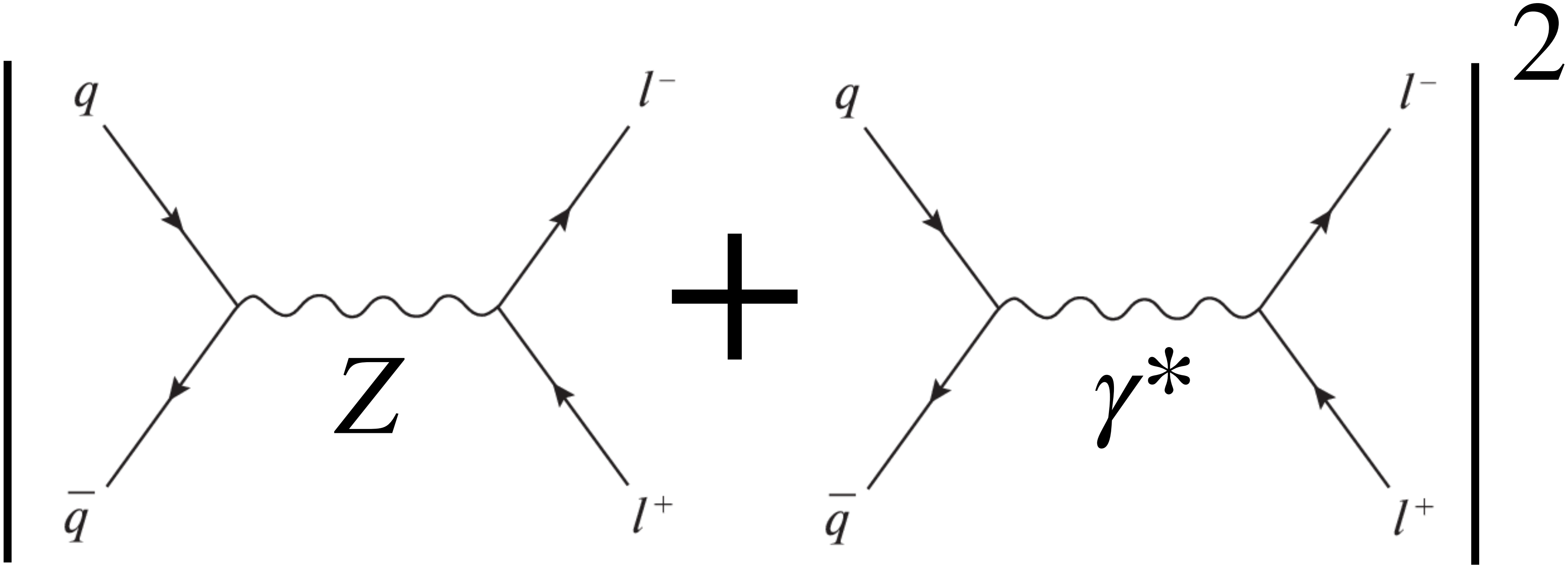
example 3



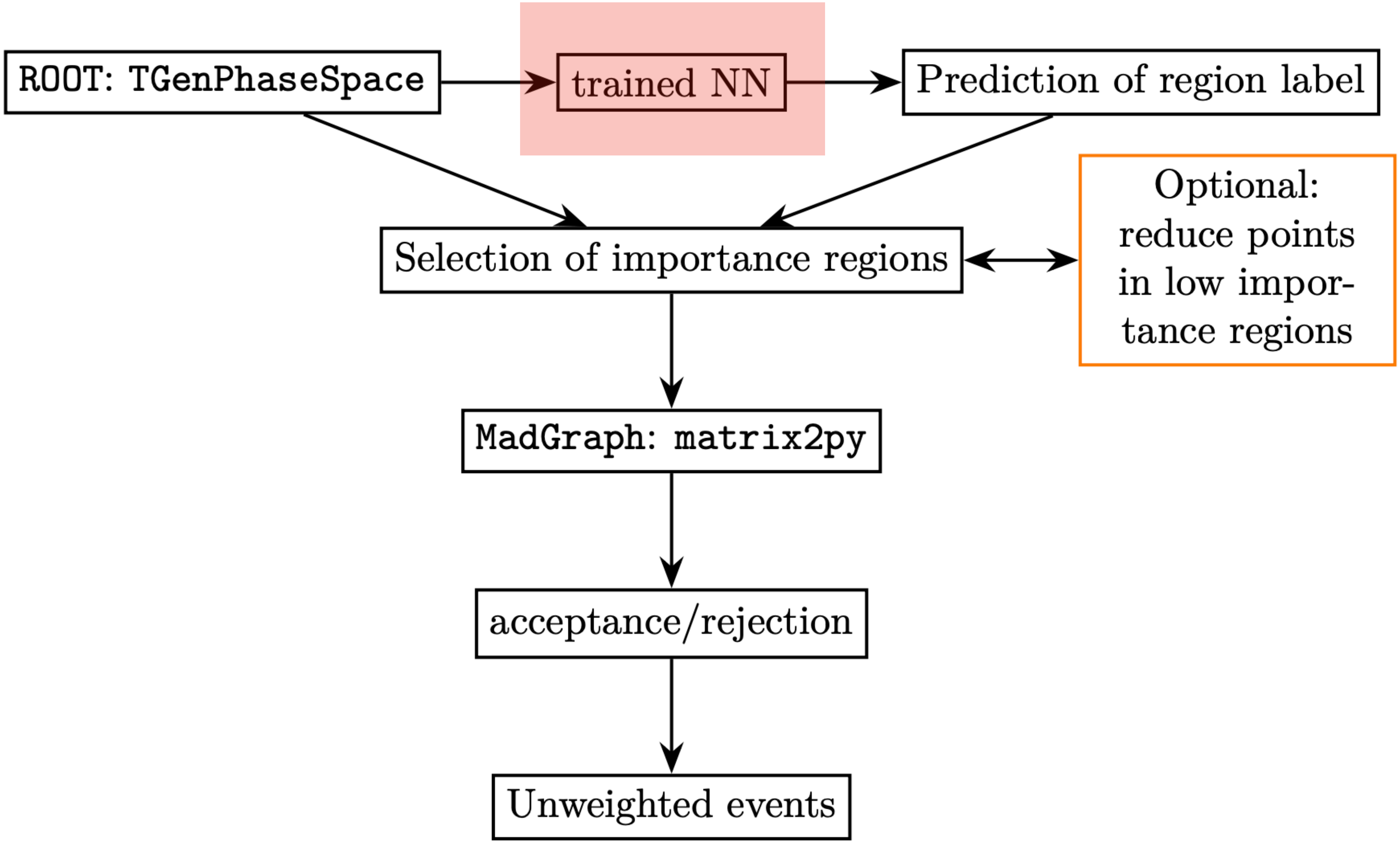
- The rate of misclassification is stable against increasing number of testing sample.

The Physics

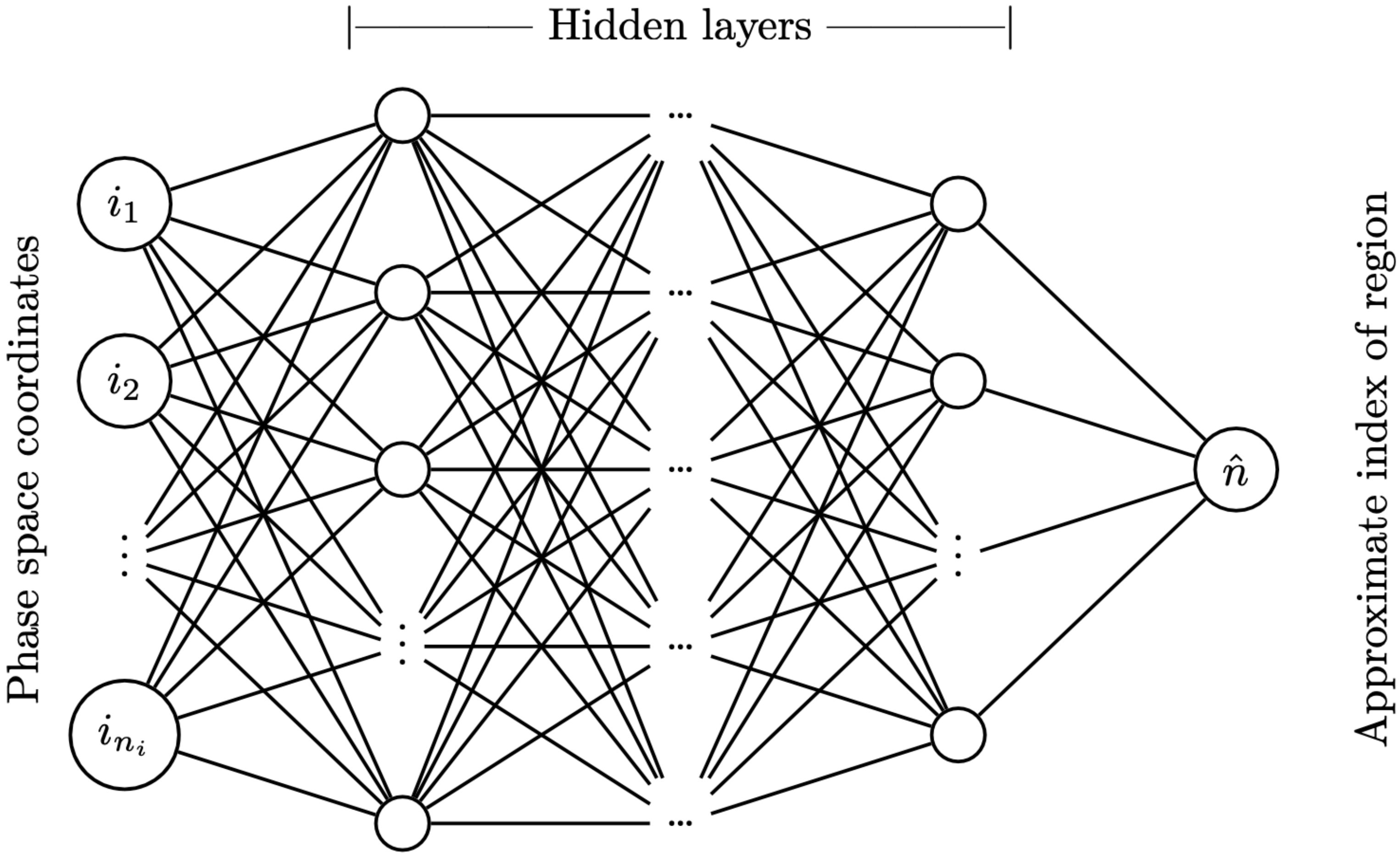
$2 \rightarrow 2$ process with interference

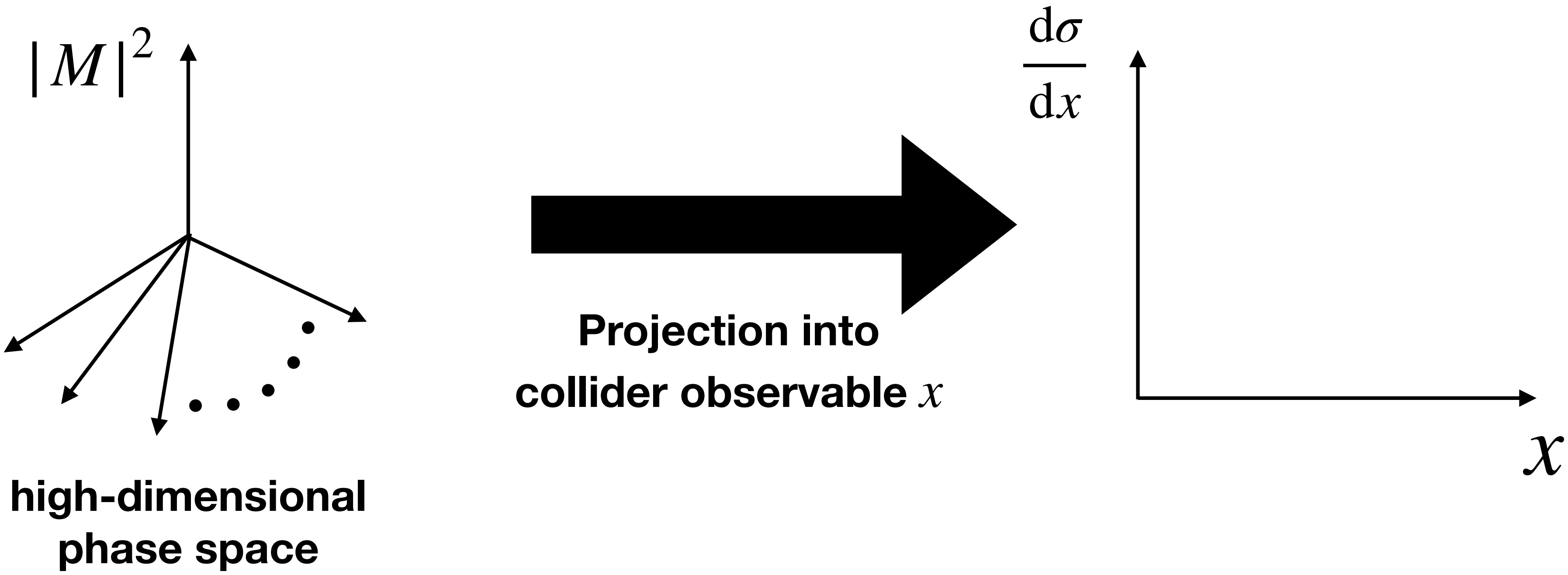


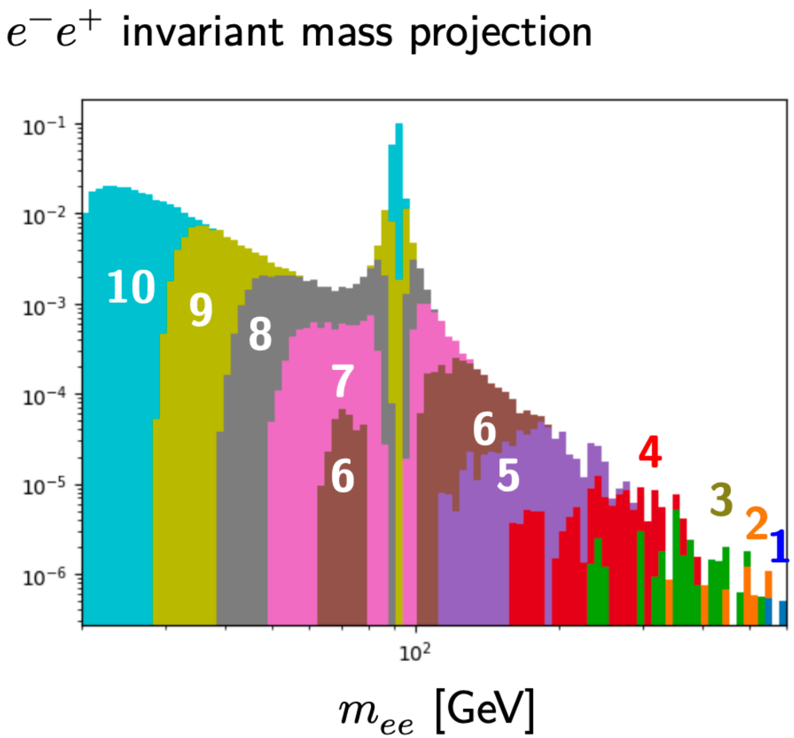
Generating MC samples with NN



Very simple NN structure

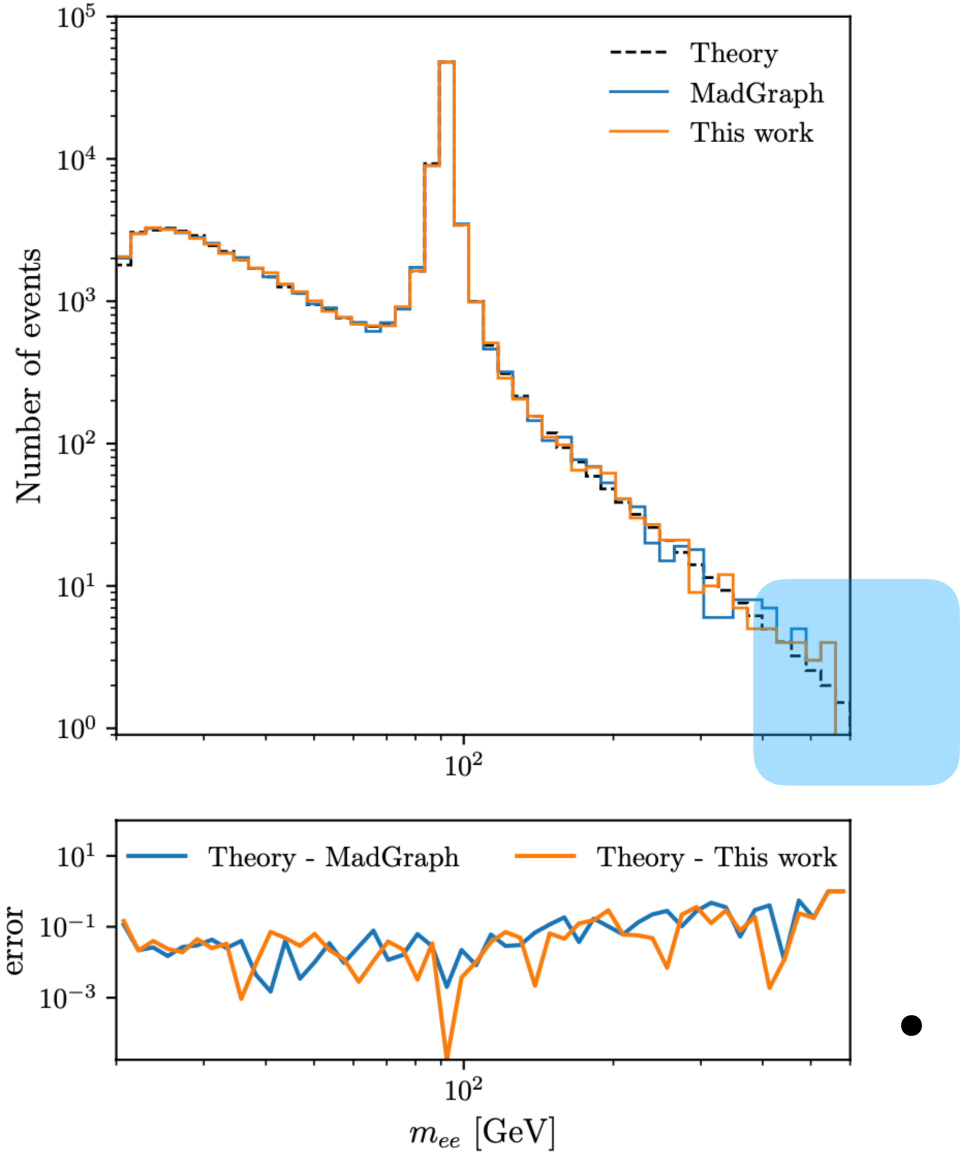


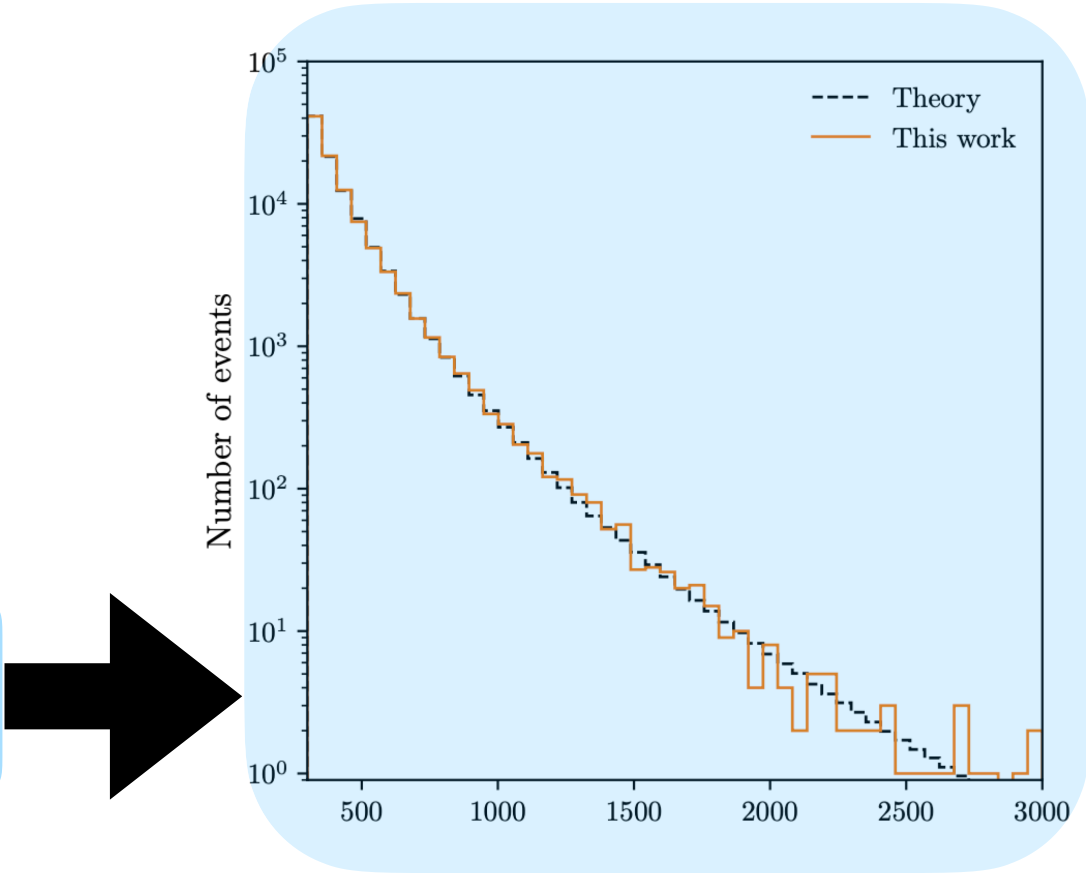




- Sample each region until enough events are accumulated.
  NN can tell which regions points belong to.
- Select points using correct result.

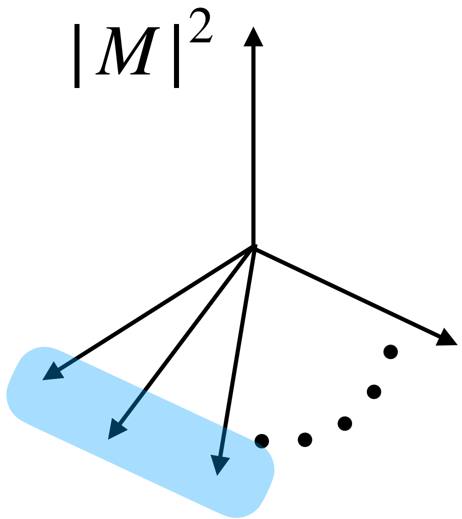
Sample as long as we want





- We can **zoom into** **"rare events"**

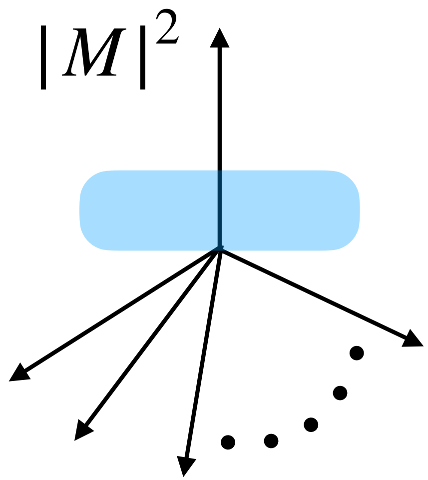
Conventional zoom



- Current zoom in MC (Madgraph) is focusing on phase space or some observable.

  \- **This cut gives effects on other observables.**

Our ZOOM



- We focus on the region of low statistics itself :

  \- This region can be mapped into various observable spaces.

I put the difficulties into my deep pockets

- We just started a journey into my dream, building up Monte Carlo Generator.

- The true difficulties are in $|M|^2$ itself. The HUGE number of diagrams.

  - I am collaborating with a string theorist, Kanghoon Lee (APCTP). He has a magic to simplify the calculation of amplitude.

- Still I need to have an advanced computing method for $|M|^2$ and more efficient importance sampling.