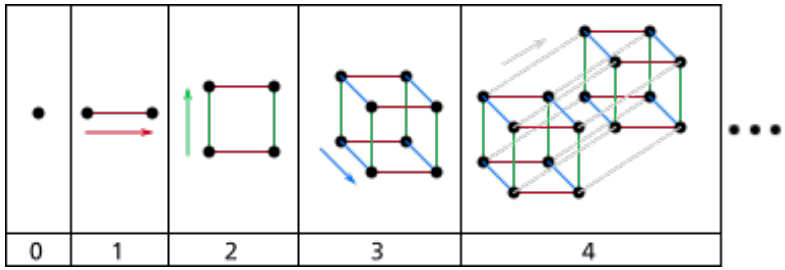
Predicting the importance of complicated regions in Monte Carlo sampling

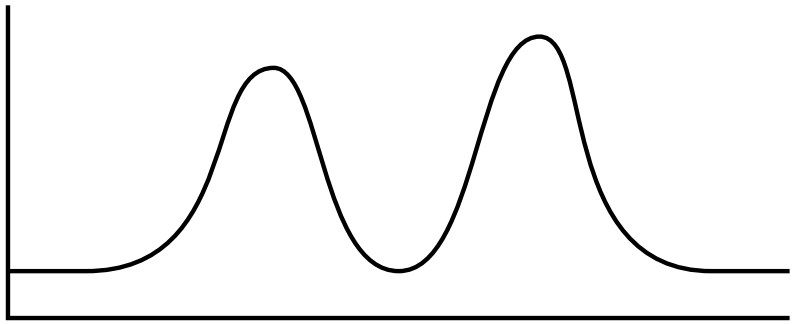
Raymundo Ramos
KIAS
(based on work with: M. Park (SEOULTECH))

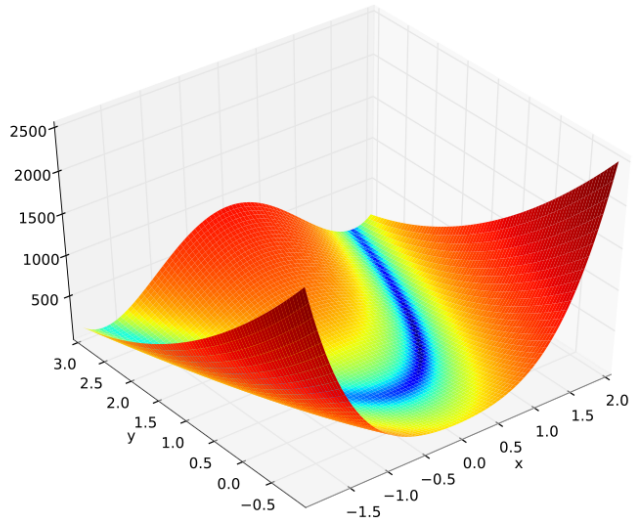
The 3rd International Joint Workshop on the Standard Model and Beyond
The 11th KIAS Workshop on Particle Physics and Cosmology
Jeju Island, South Korea, Nov 13, 2023

What makes a complicated space of parameters?

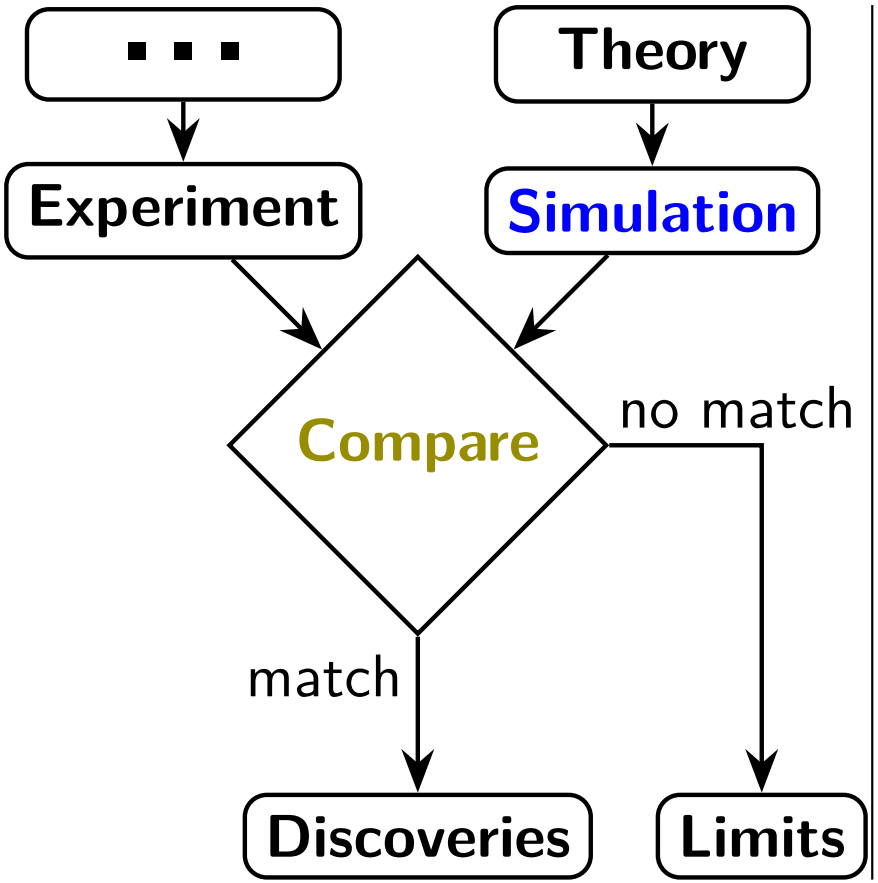
- Several dimensions
- Multimodality
- Curved degeneracy
- ...







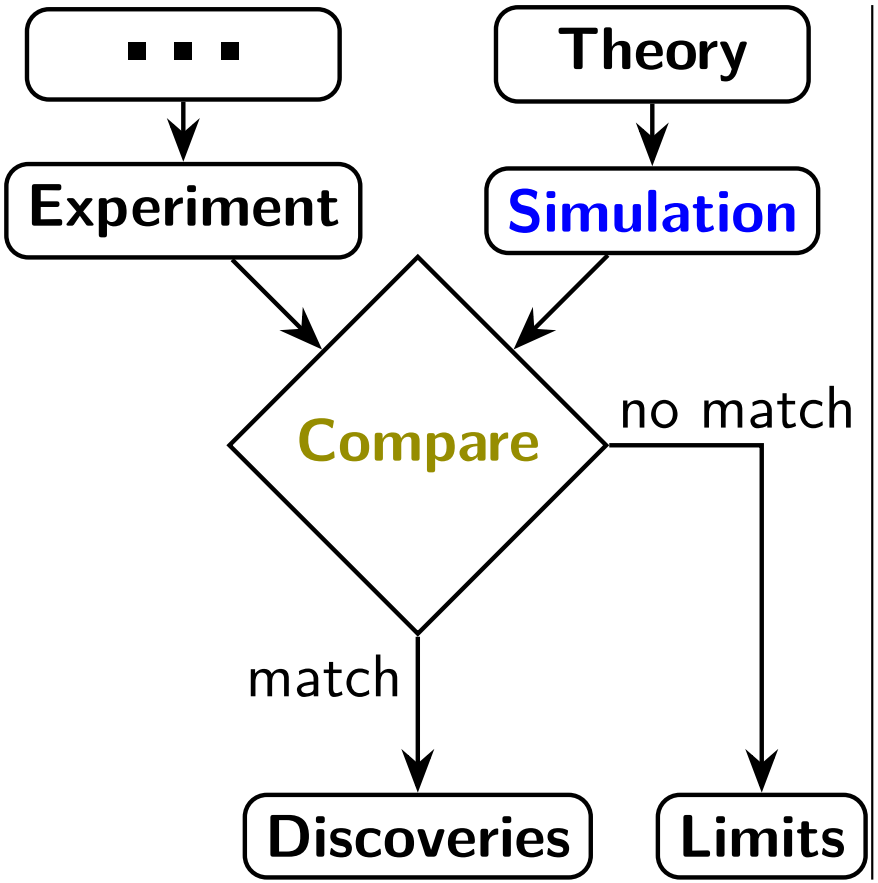
From theory to discovery (or limits)



More diverse and **more precise** experimental results.

Simulations have to keep up with the **complexity** of experiments and provide **accurate** predictions.

From theory to discovery (or limits)

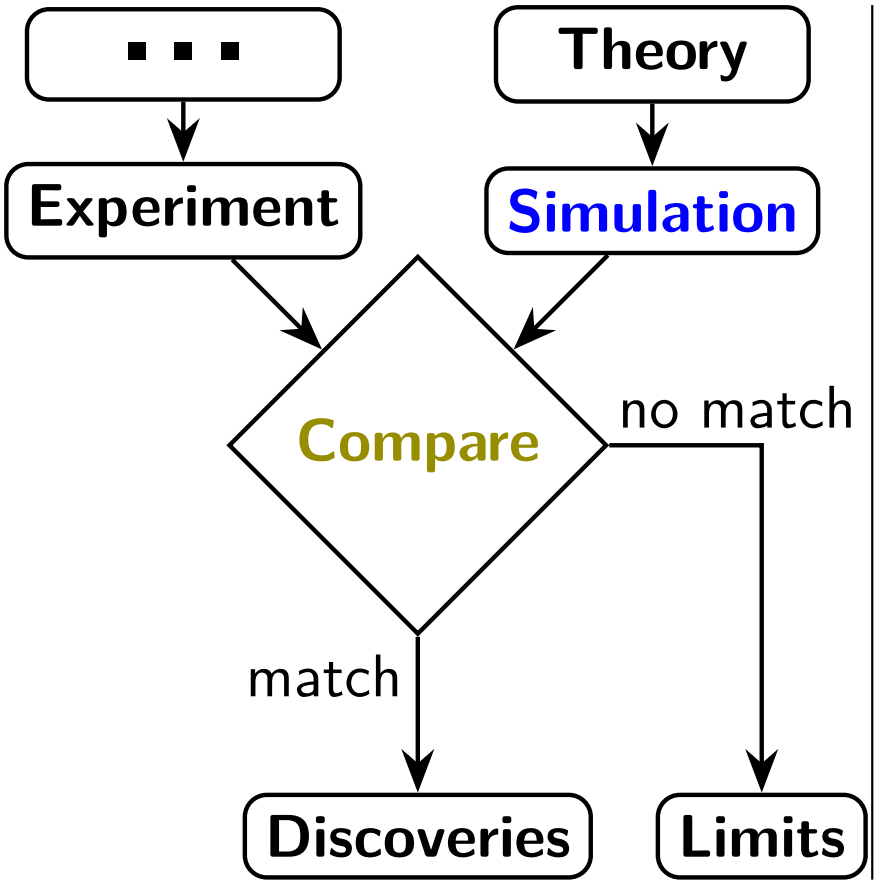


More diverse and **more precise** experimental results.

Simulations have to keep up with the **complexity** of experiments and provide **accurate** predictions.

We need more powerful and expensive computers!

From theory to discovery (or limits)



More diverse and **more precise** experimental results.

Simulations have to keep up with the **complexity** of experiments and provide **accurate** predictions.

We need ~~**more powerful and expensive computers!**~~ **improved techniques for data analysis!**

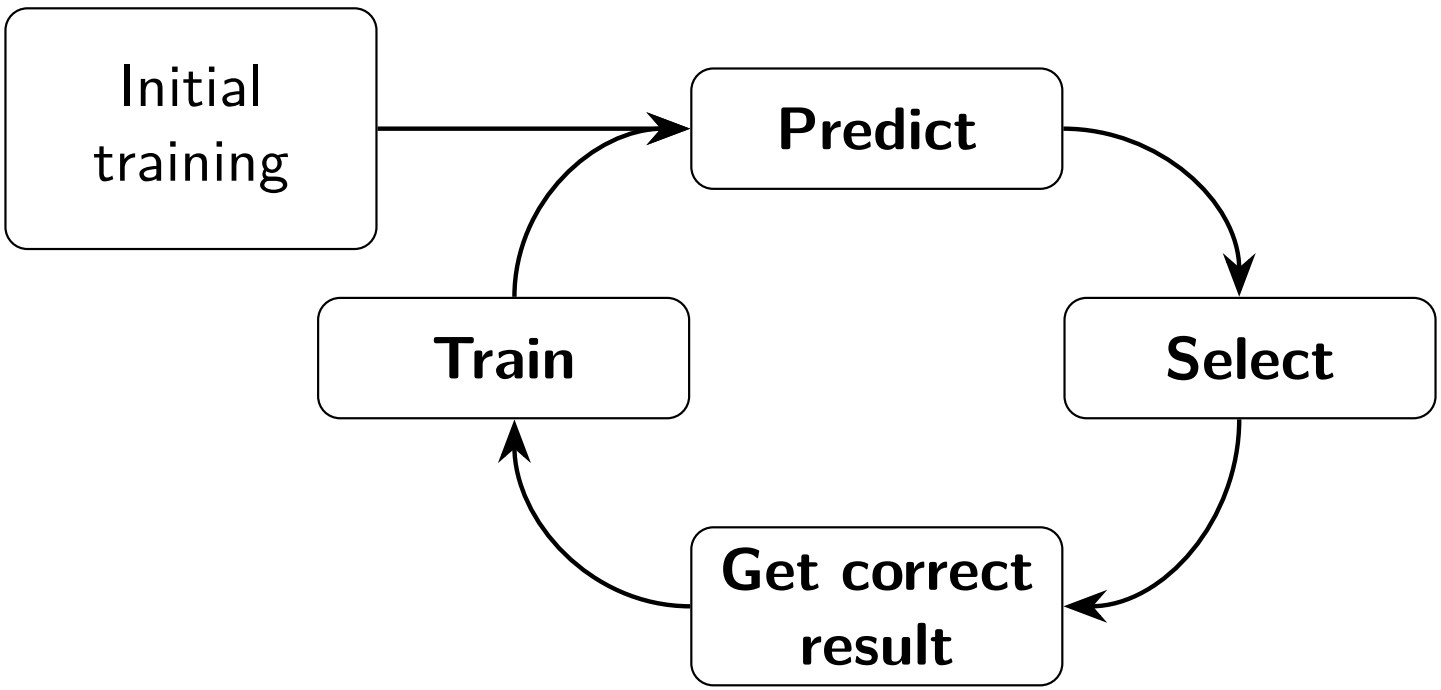
How we want to improve the data analysis

- Neural networks (NN) as generic function approximators
- Useful when training a NN **could be more efficient** than passing every single point through a heavy calculation
- Design a process where the accuracy of the NN becomes proportional to our interest in sampled regions:
  - spend, relatively, **more time sampling regions of interest**,
  - just enough time for low importance regions

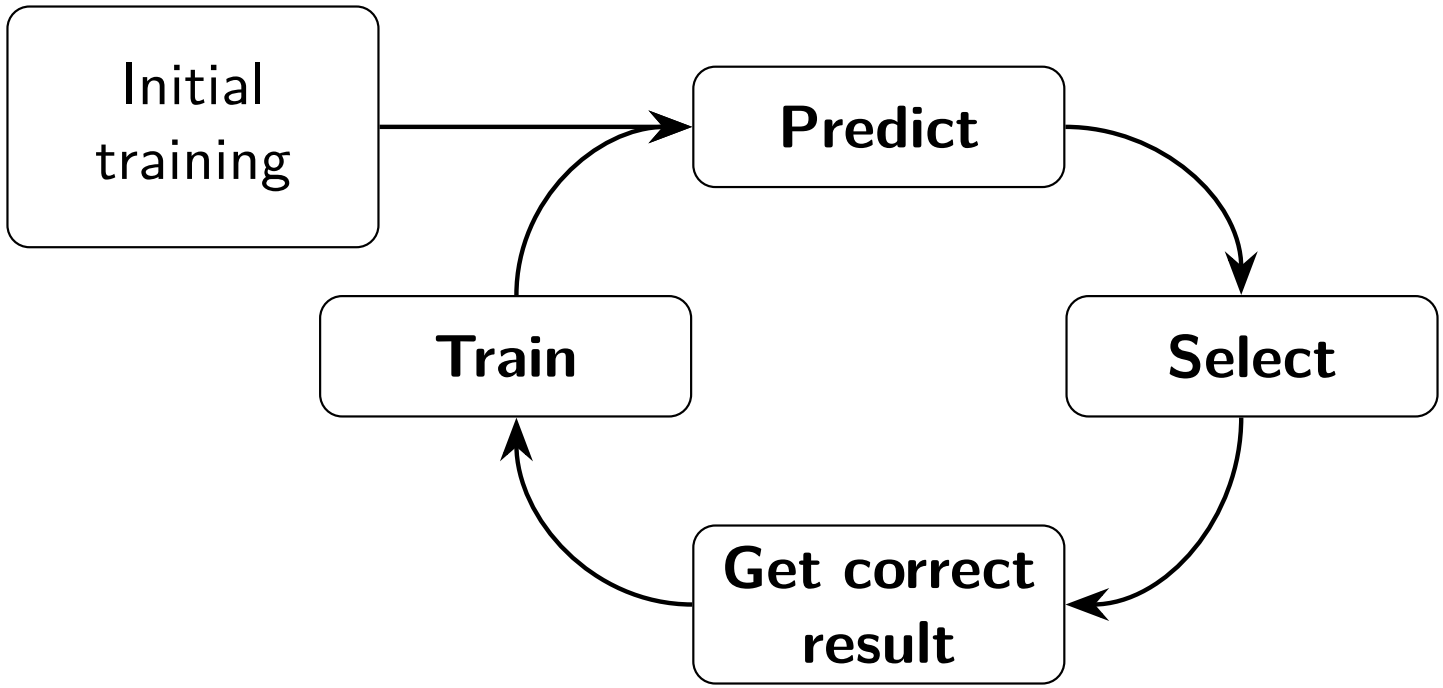
Follow an iterative process similar to others, e.g.

Ren, Wu, Yang and Zhao [arXiv:1708.06615];
Caron, Heskes, Otten and Stienen [arXiv:1905.08628];
Goodsell and Joury [arXiv:2204.13950]

An iterative process



An iterative process



Until some goal accuracy has been reached with the NN

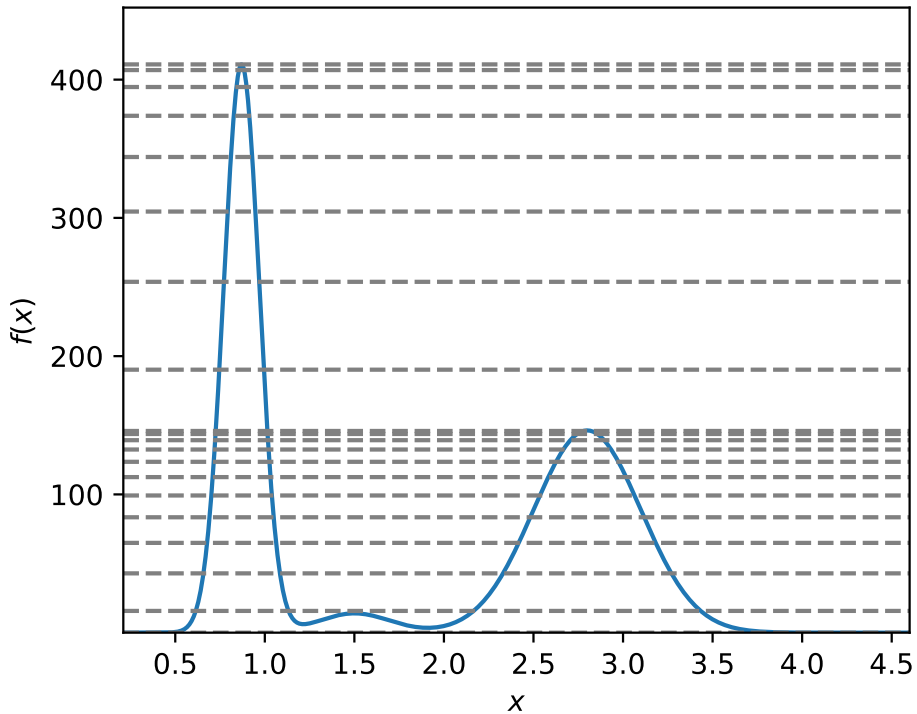
Monte Carlo integration with stratification

Divide the parameter space according to values of the function we want to integrate (**Lebesgue integration**):

$$f : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$$

Divide the space in n sections (the classes)

$$\Phi_j = \left\{ x \mid l_j < f(x) \leq l_{j+1} \right\}$$



The integral becomes

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

where V_{Φ_j} is the volume or ***length*** of Φ_j

Monte Carlo integration with classification

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

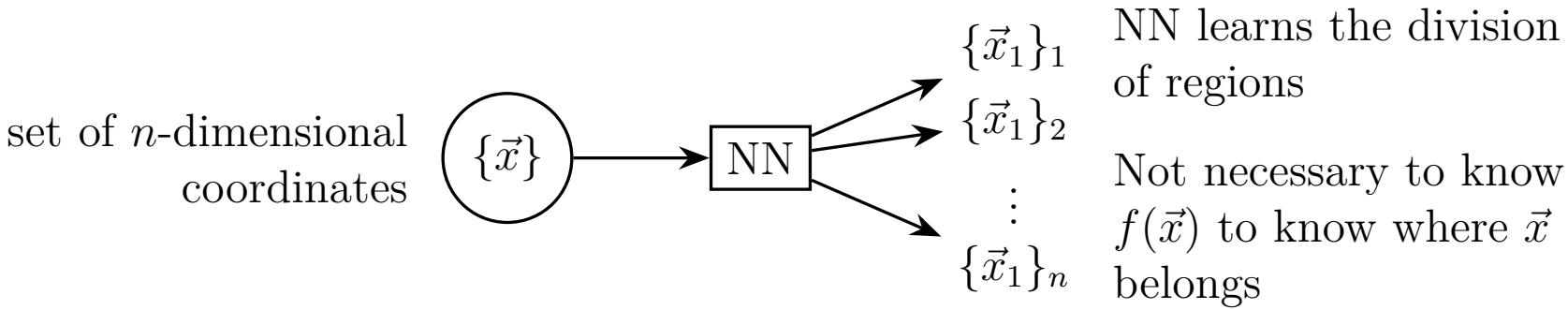
Hard question: $\vec{x} \in \Phi_j$?

With an anwser for a large sample of N points:

$$V_{\Phi_j} \approx \frac{N_j}{N} V_\Phi\,, \quad \langle f \rangle_{\Phi_j} \approx \frac{1}{N_j} \sum_{i=1}^{N_j} f(x_i)$$

Can we get an answer for this question?

Monte Carlo with classification and ML



Train the neural network (NN) with an iterative process:

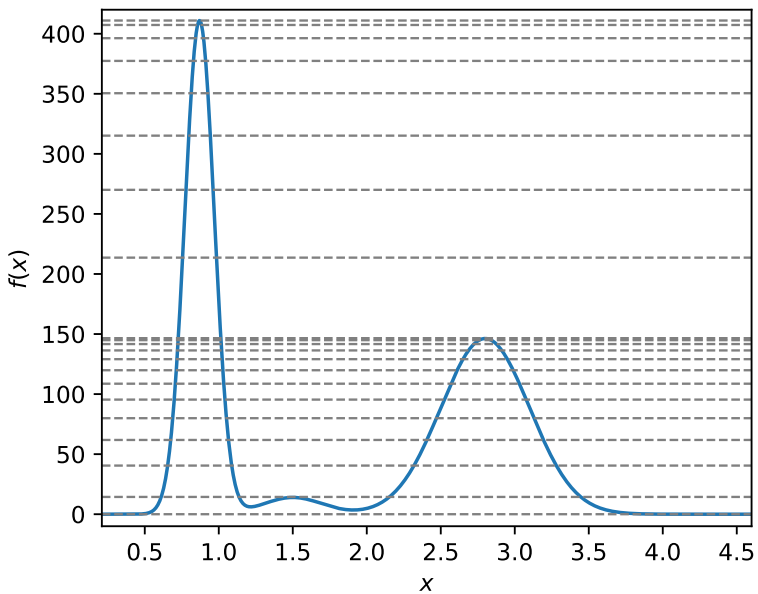
1. Train NN with a sample of points and function value.
2. Get predictions from the NN for a larger sample of new points.
3. Use function to correct wrong predictions.
4. Go back to training.

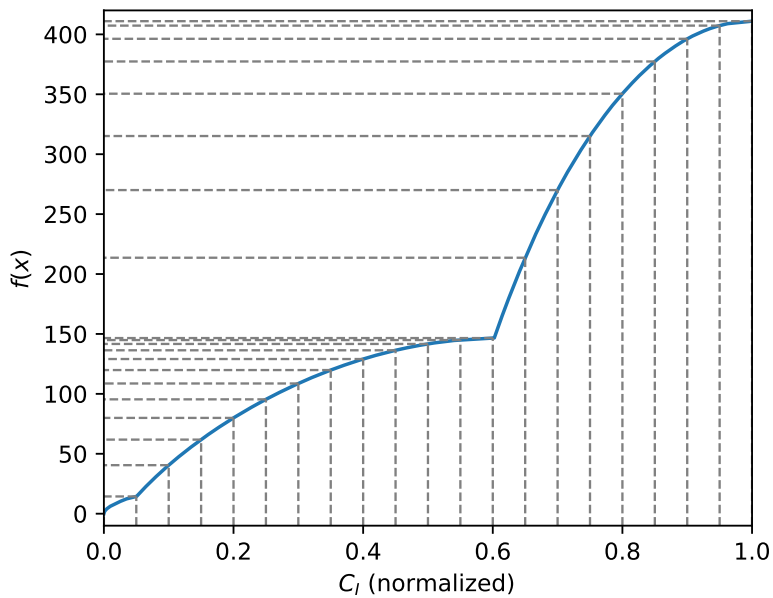
Repeat the process until NN is accurate enough

Monte Carlo with classification and ML

Next hard question: How to divide $f(\vec{x})$?

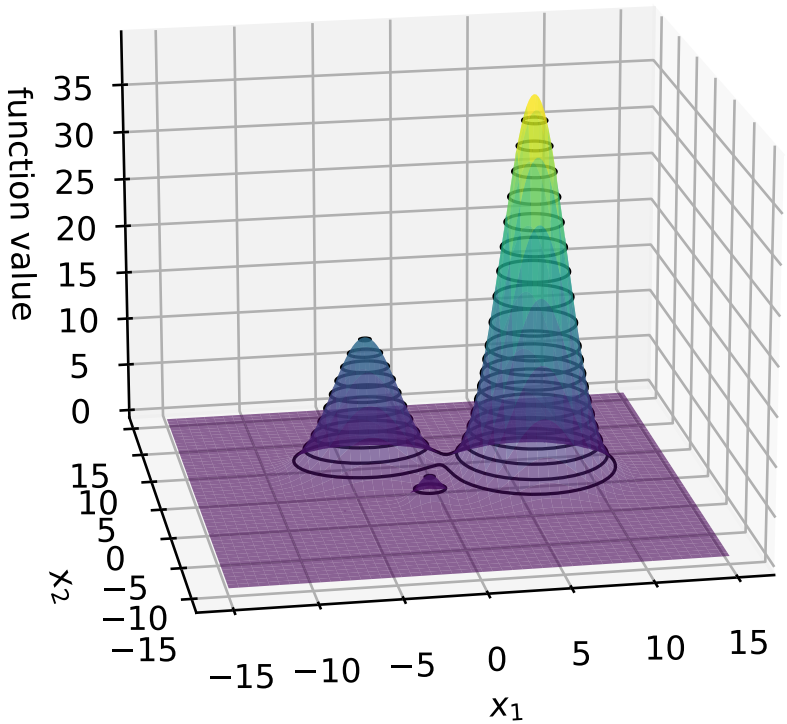
- **Infinite possibilities**
- a few simple examples, choose limits on $f(\vec{x})$ such that:
  - ➜ Φ_j with similar lengths V_{Φ_j}
  - ▼ Φ_j with similar contributions to $I_\Phi[f(x)]$

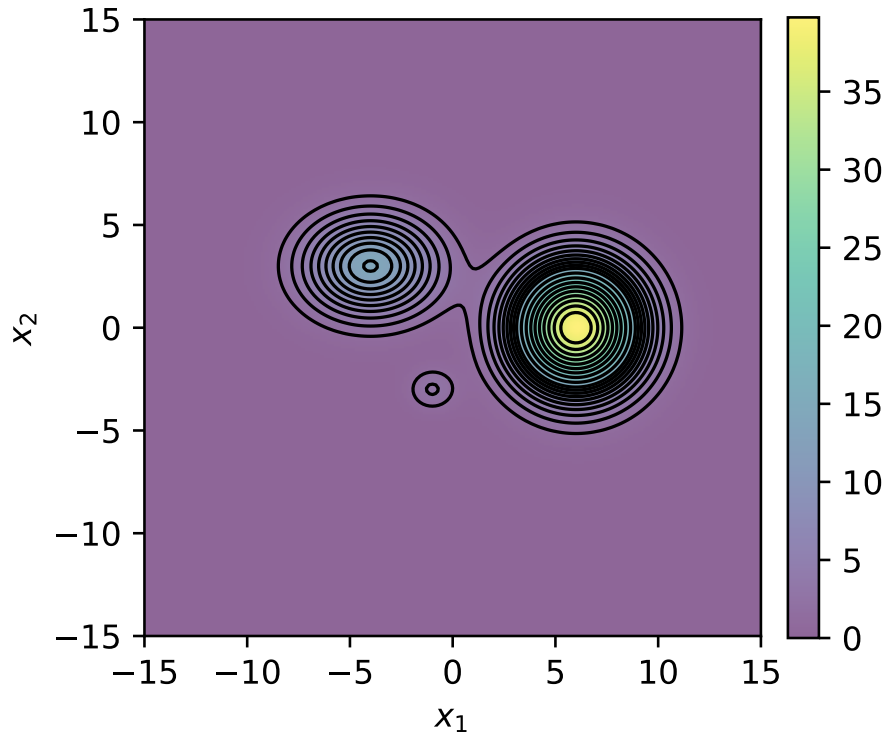




Learn divisions of a function with multiple peaks

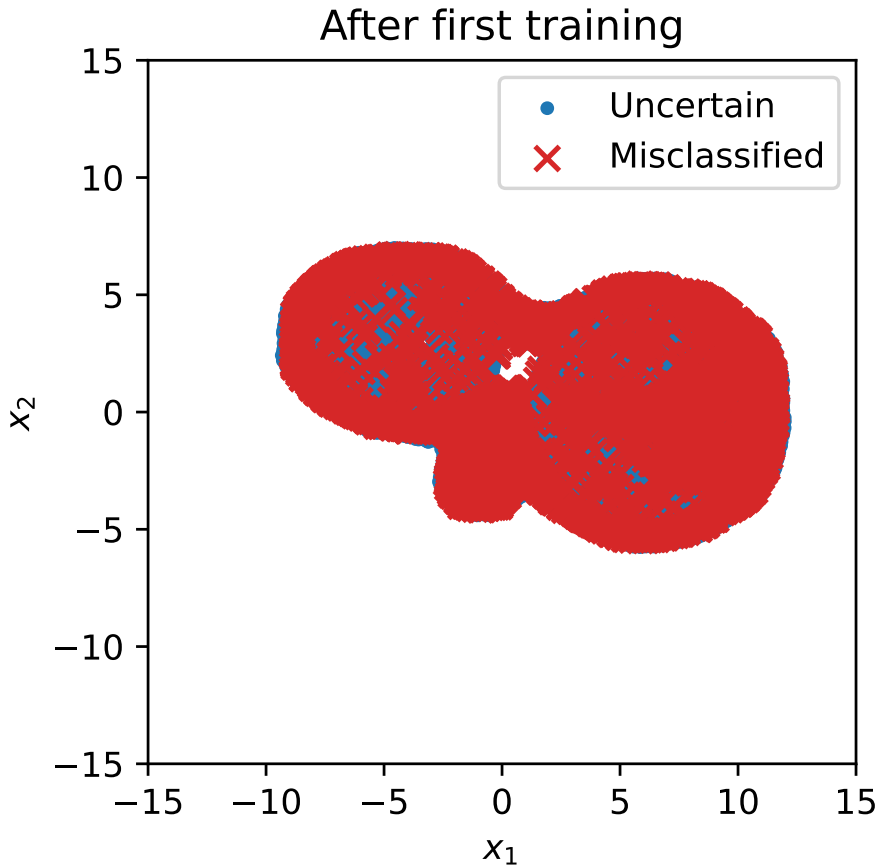
20 regions with similar contribution to value of integral

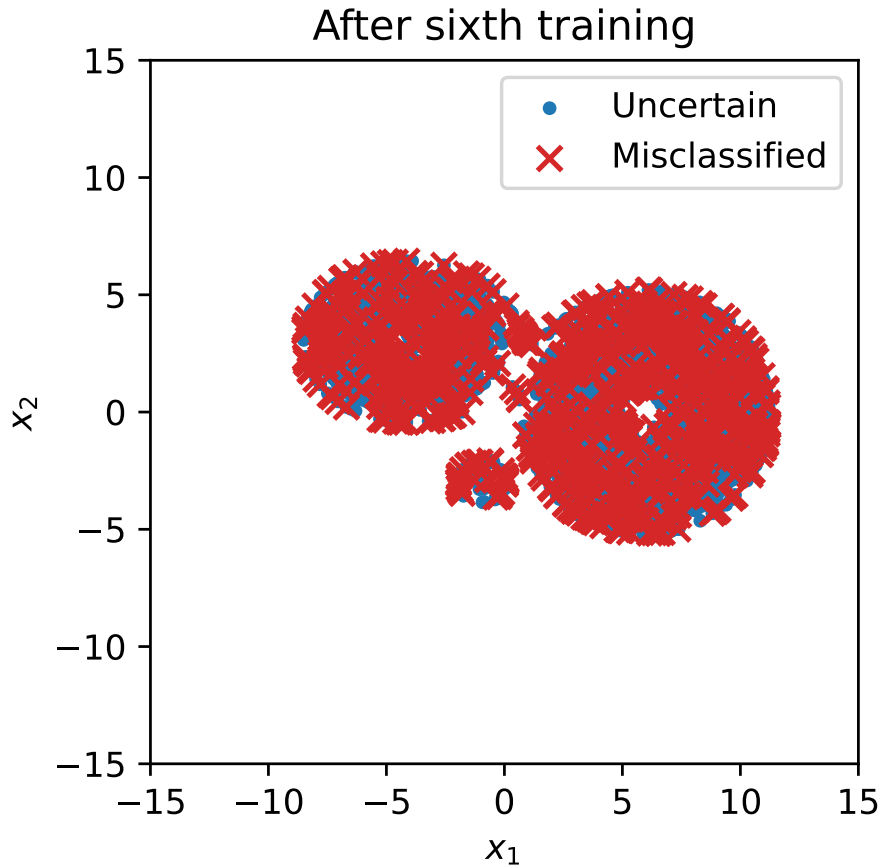




Learn divisions of a function with multiple peaks

20 regions with similar contribution to value of integral



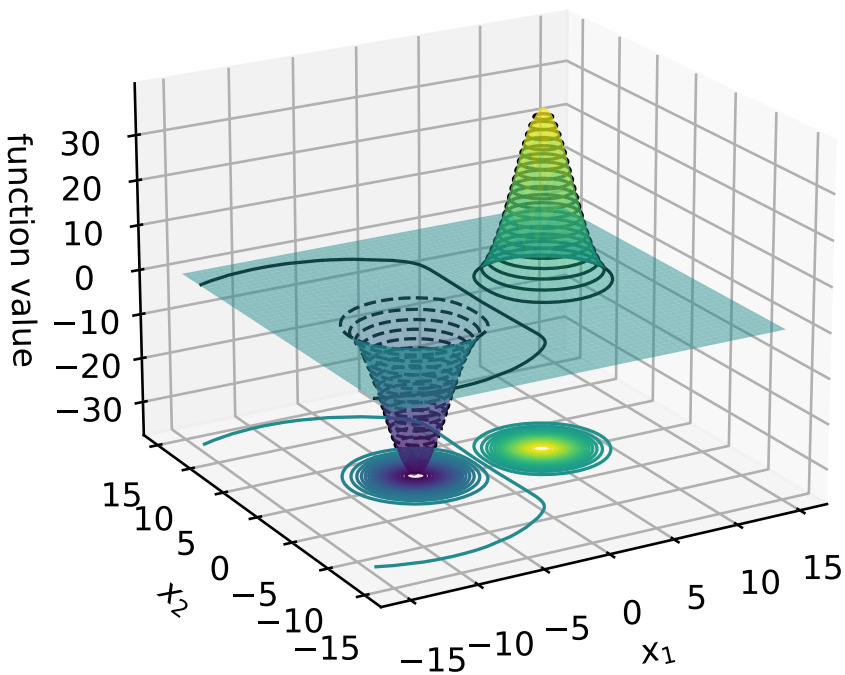


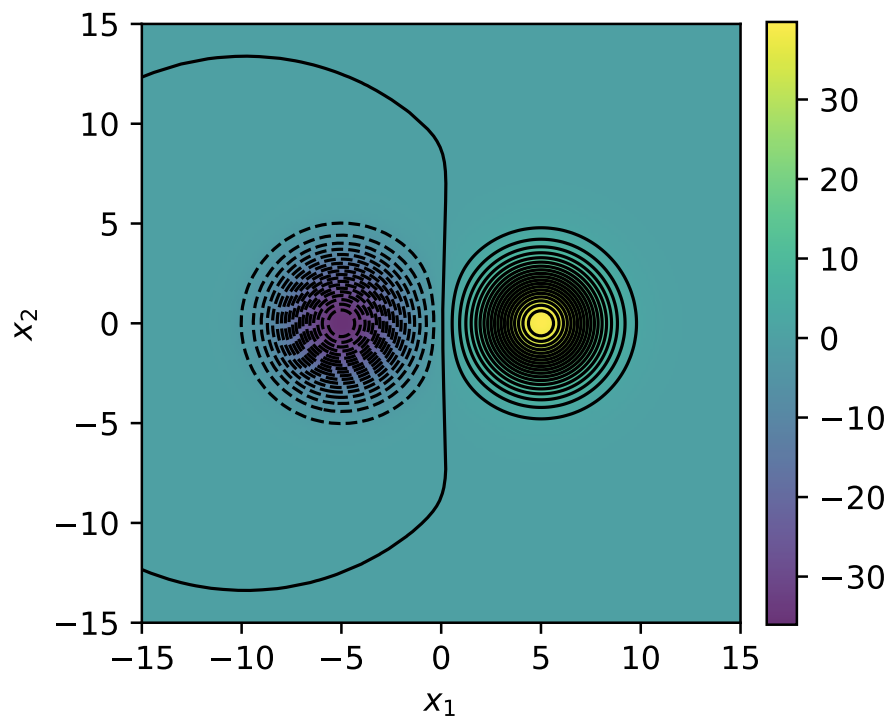
After sixth training step: above 99% accuracy (100 000 test points).

Function with large cancellation

$$f(x_1, x_2) = 1000[f_+(x_1, x_2) - f_-(x_1, x_2)] + f_{\text{bg}}(x_1, x_2)$$

$$\int f dx_1 dx_2 = \int f_{\text{bg}}(x_1, x_2)$$

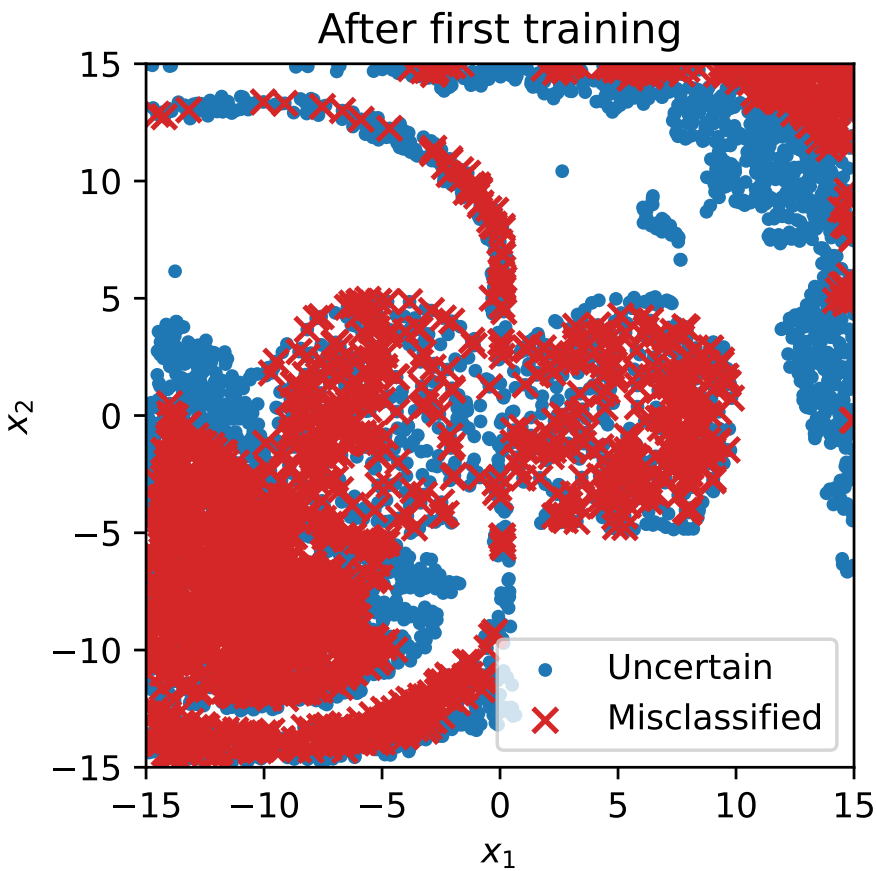


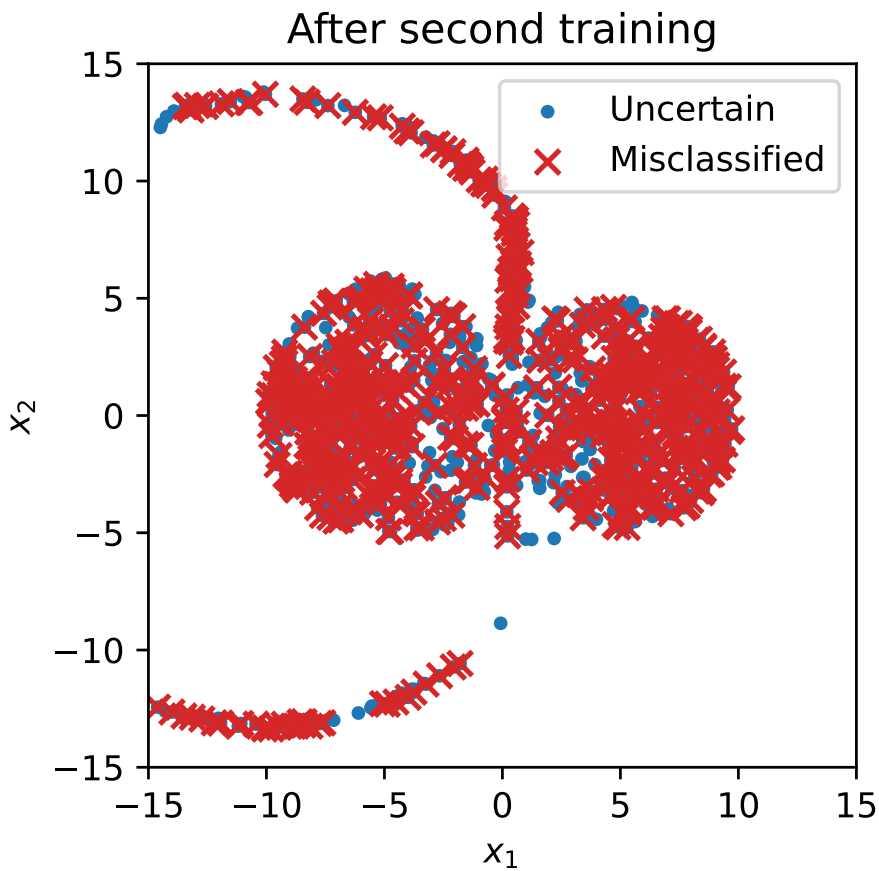


function with large cancellation

$$f(x_1, x_2) = 1000[f_+(x_1, x_2) - f_-(x_1, x_2)] + f_{\text{bg}}(x_1, x_2)$$

$$\int f dx_1 dx_2 = \int f_{\text{bg}}(x_1, x_2)$$





Quark pair to electron + positron

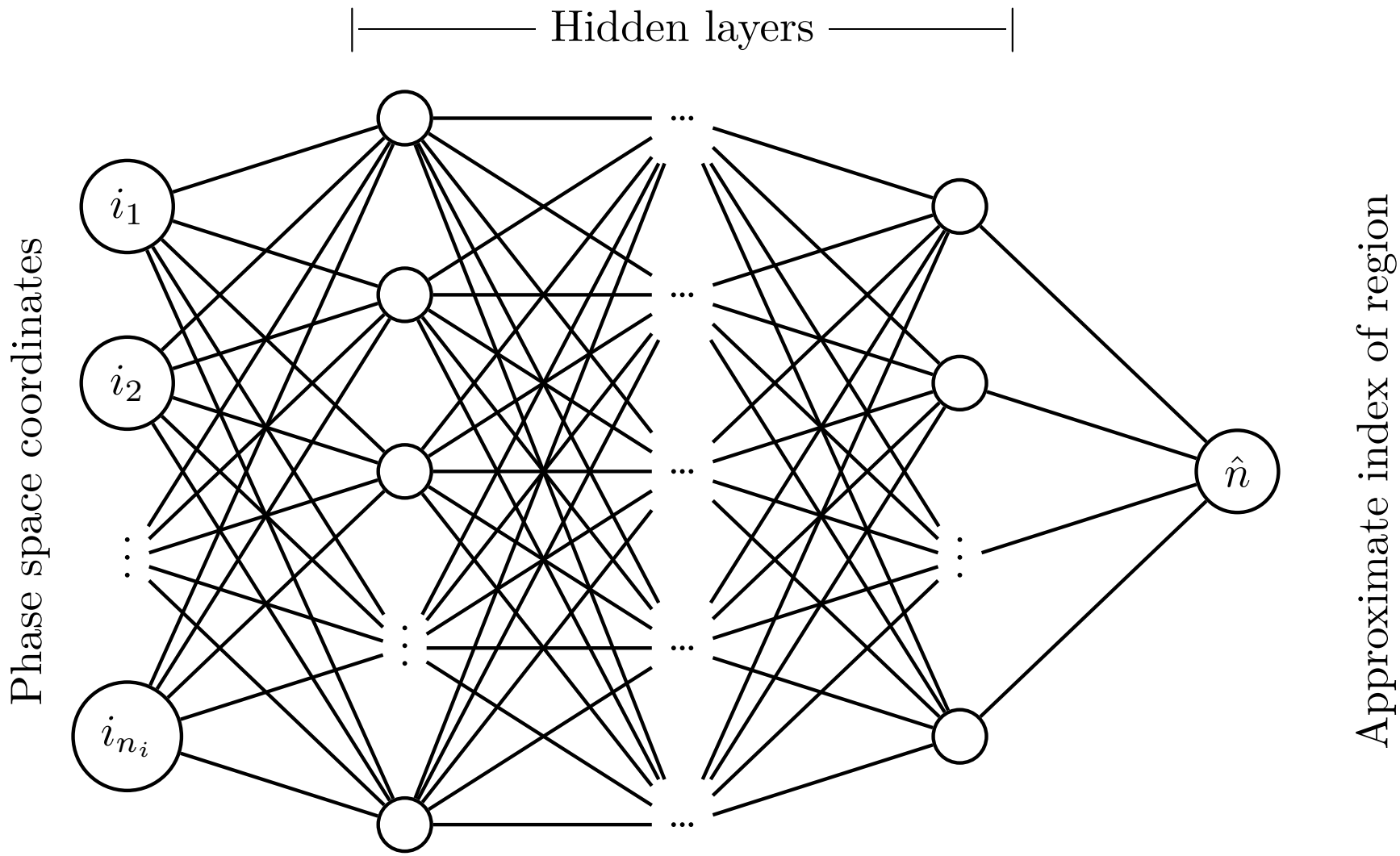
Very simple example:

$$u\bar{u} \rightarrow e^- e^+$$

- `ROOT - TGenPhaseSpace`: phase space generator.
- `Madgraph` (standalon mode): matrix element.
- `NNPDF23`: parton density function.
- cuts: leptons: $p_T > 10\,\text{GeV}$, $|\eta| < 2.5$

Quark pair to electron + positron

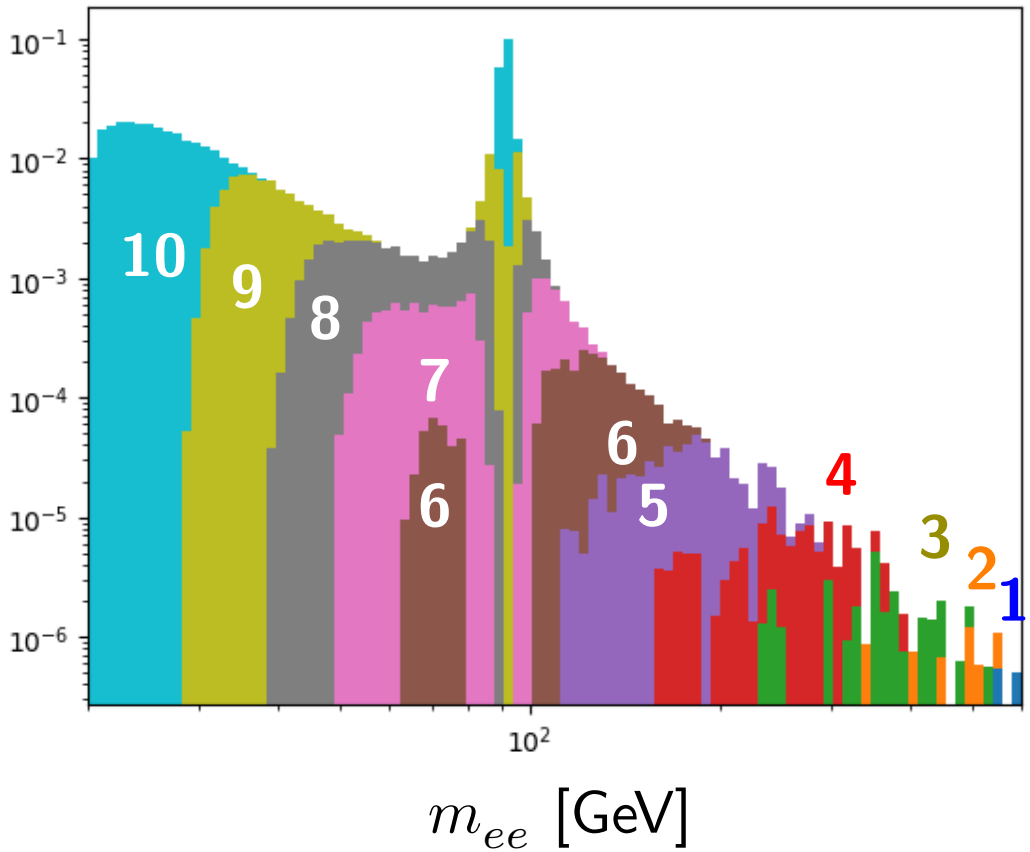
Very simple setup:



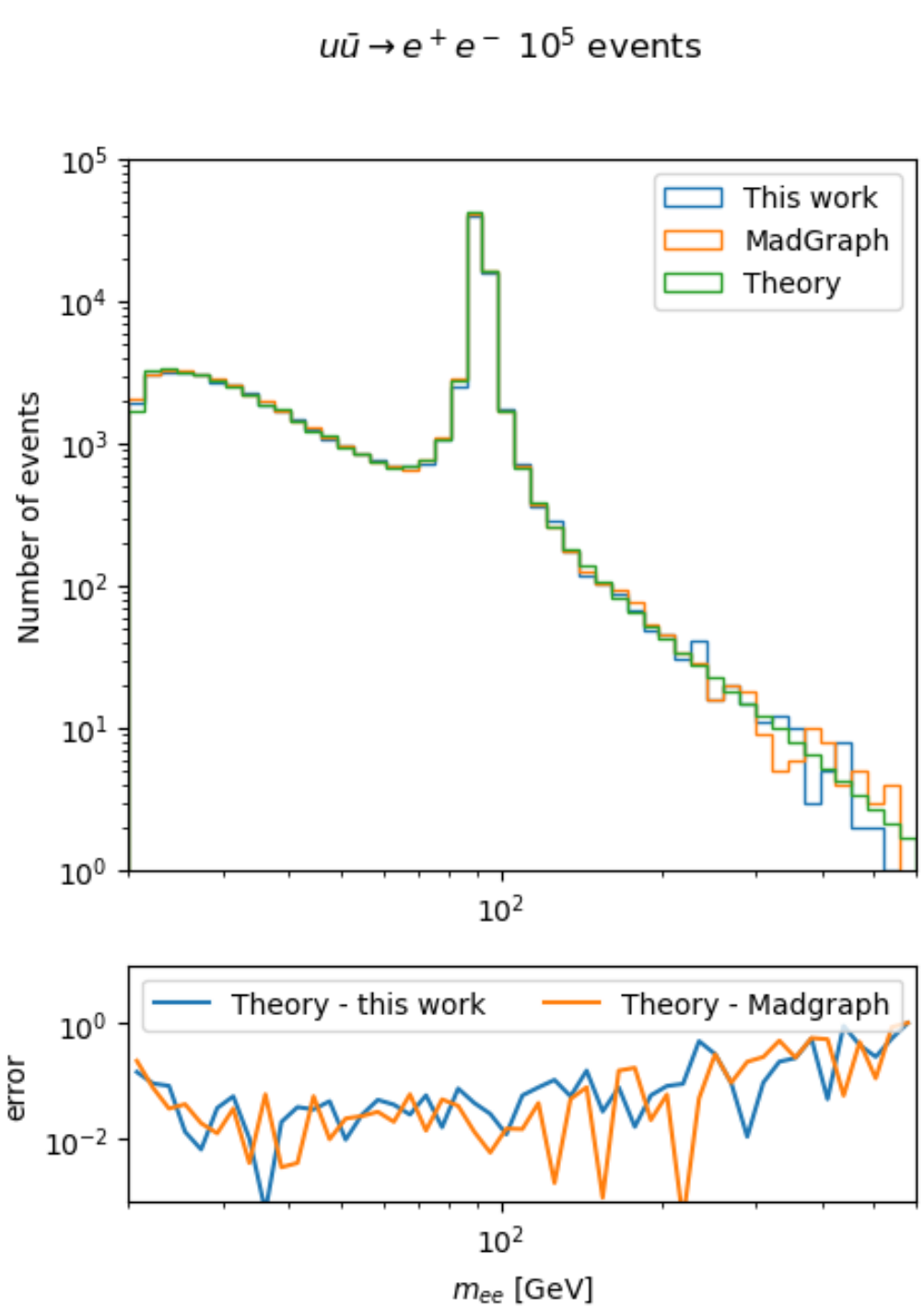
10 usable divisions + 1 irrelevant region

Generate events: 10 usable regions

e^-e^+ invariant mass projection

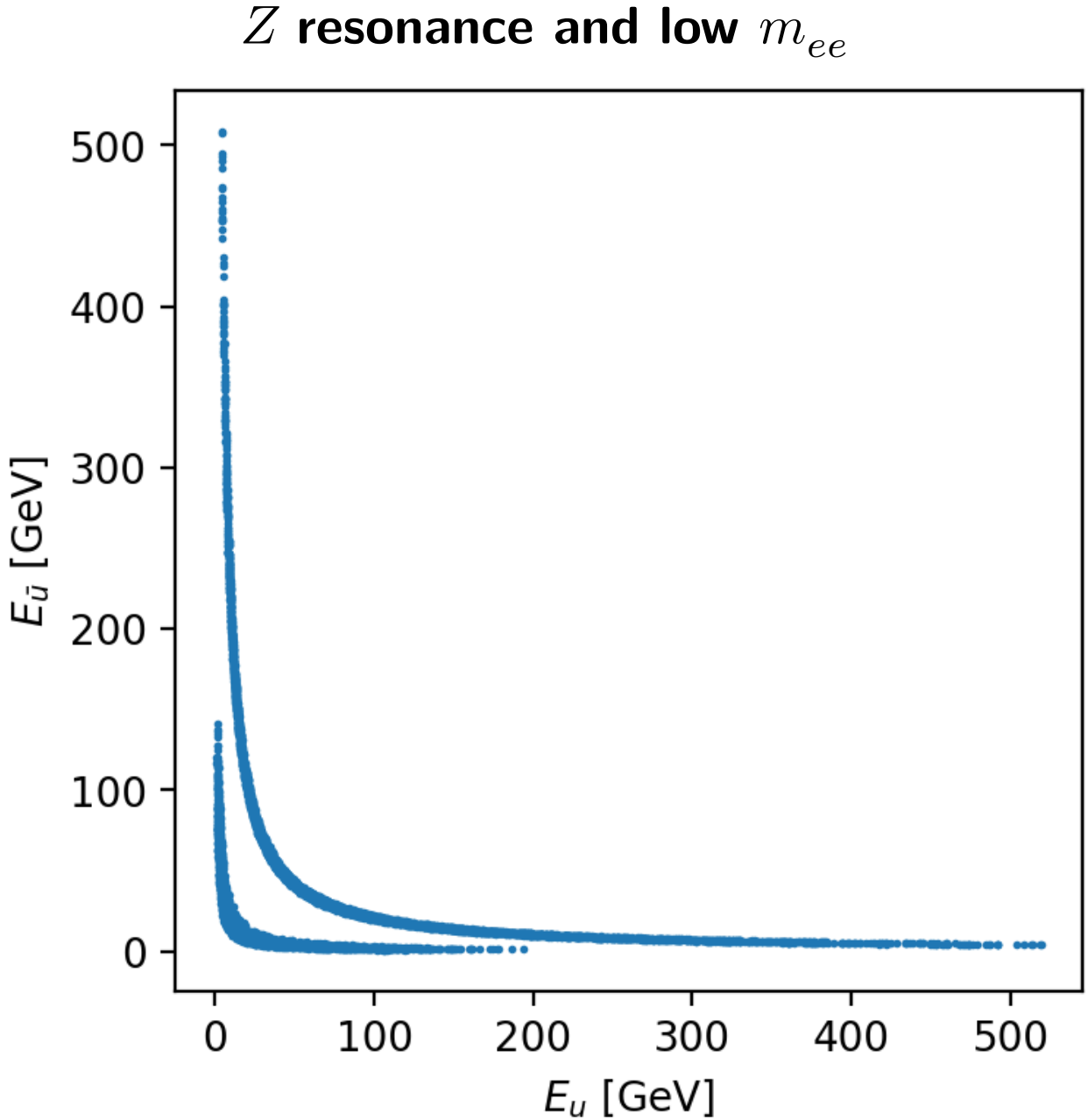


- Sample each region until enough events are accumulated.
  NN can tell which regions points belong to.
- Select points using correct result.

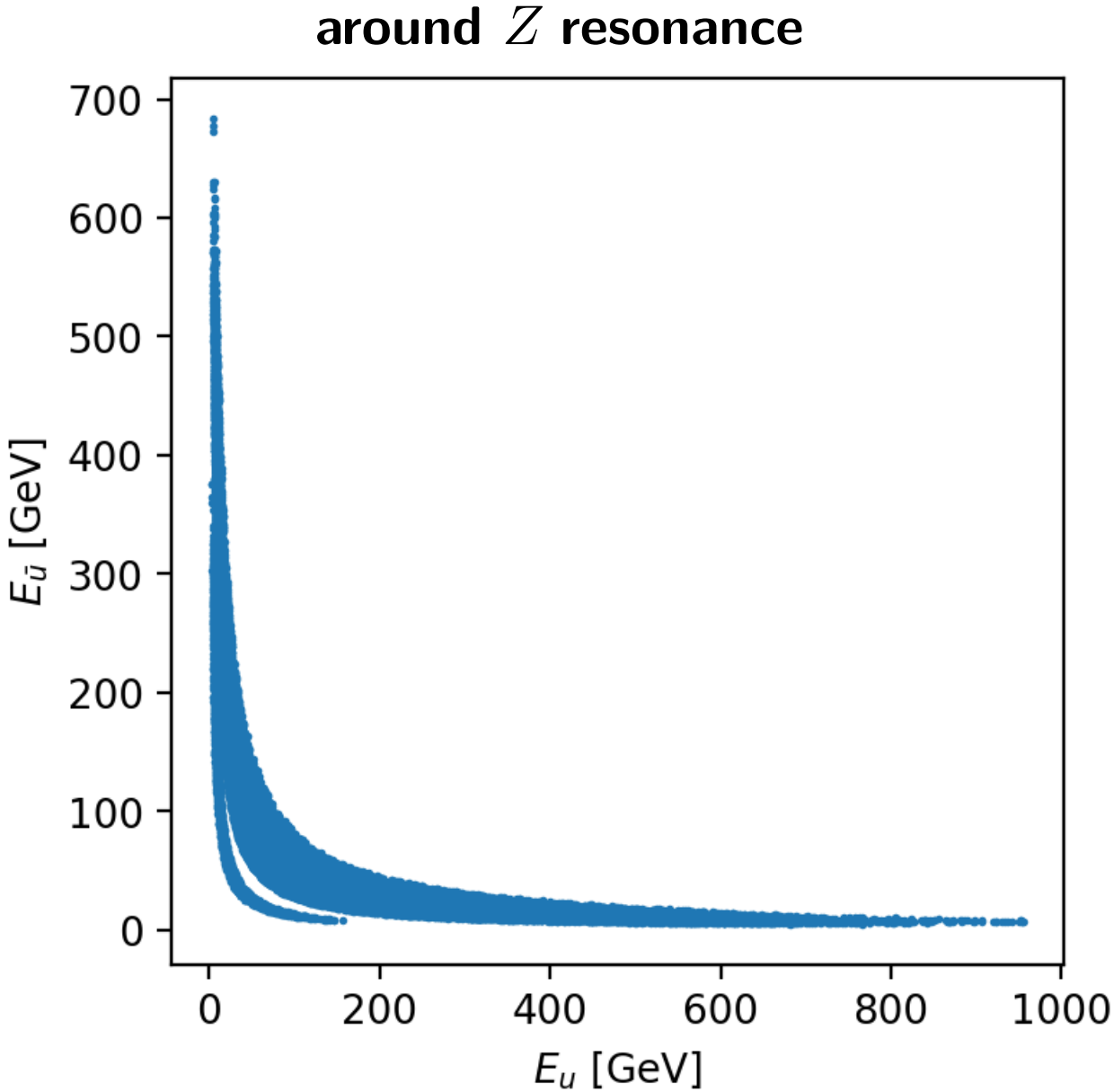


- 10^5 unweighted events
- High m_{ee} error expected from thinning of sample.
- Invariant mass around Z resonance is similar when comparing to MadGraph
- Efficiency of selection of unweighted events increases with more regions. But more regions requires more points for training

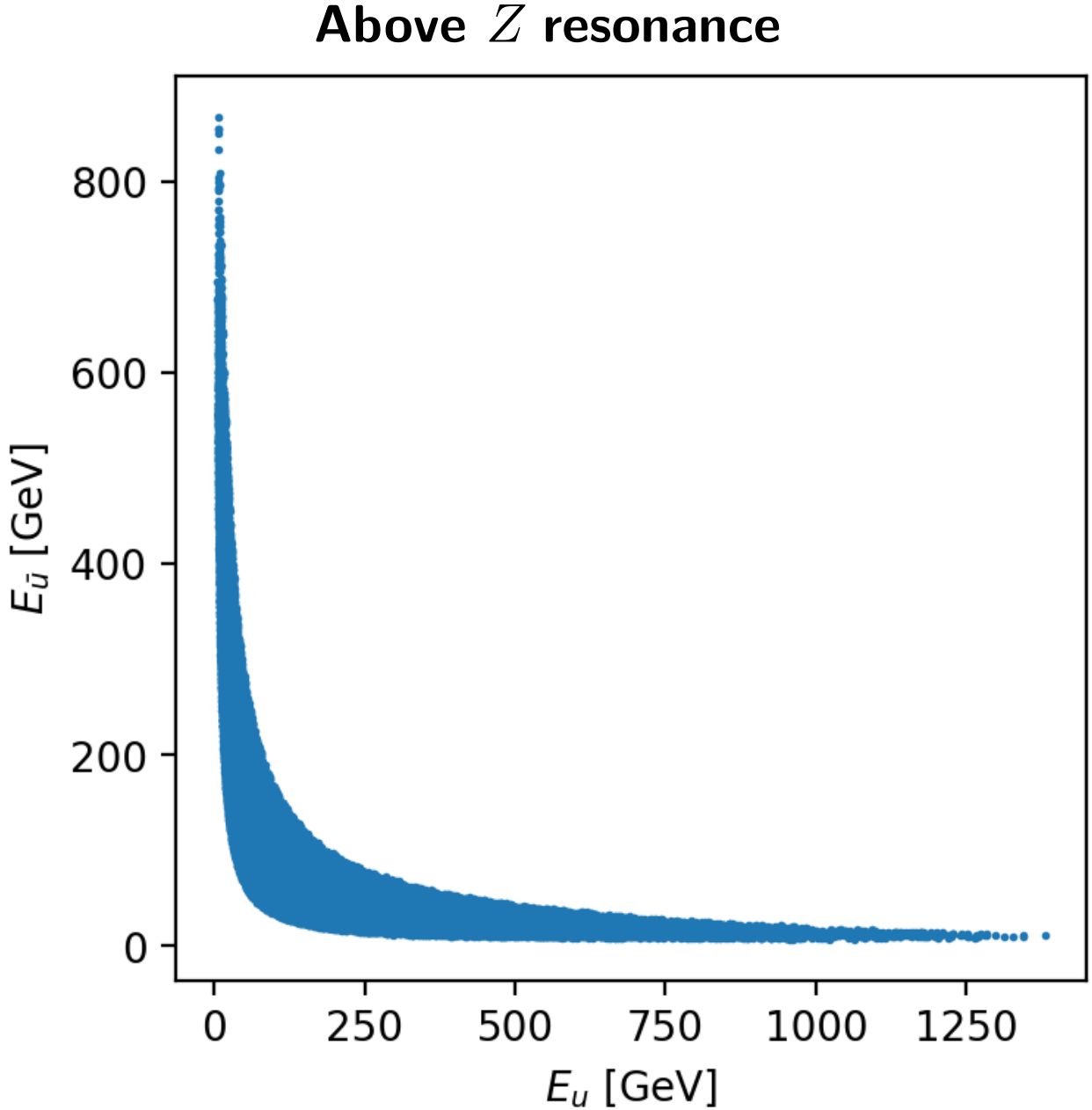
Vanity plots: Region 10 as seen by the NN



Vanity plots: Region 6 as seen by the NN



Vanity plots: Region 5 as seen by the NN



Summary

- Monte Carlo simulations could be challenging due to
  - $$ Time consuming costly operations
  - ❋ High dimensional spaces
- Machine learning can improve the situation, but many options exist.

➔ We presented an iterative process to accelerate sampling of points in a parameter space using a neural network.

➔ The main idea is to **separate (preclassify)** regions according to importance.

- Concentrate on high importance regions
- Forget about regions that do not contribute to results

➔ Selection based on a sigle number.

Thanks for listening!