DLScanner: Parameter space scanner assisted by deep learning methods

Ahmed Hammad (KEK, Japan)

In collaboration with Raymundo Ramos (KIAS, Korea)

MLPhYs 学術変革領域研究(A) 学習物理学の創成
Foundation of "Machine Learning Physics"

AEI 2025 , Durham

Outline

1- Introduction to traditional sampling methods

2- Base idea for ML assisted sampling

3- Improved ML assisted sampling (DLScanner)

4- Results

Problem to be solved:

$$Y = F(X)$$

What are the values of X to have a certain value of Y

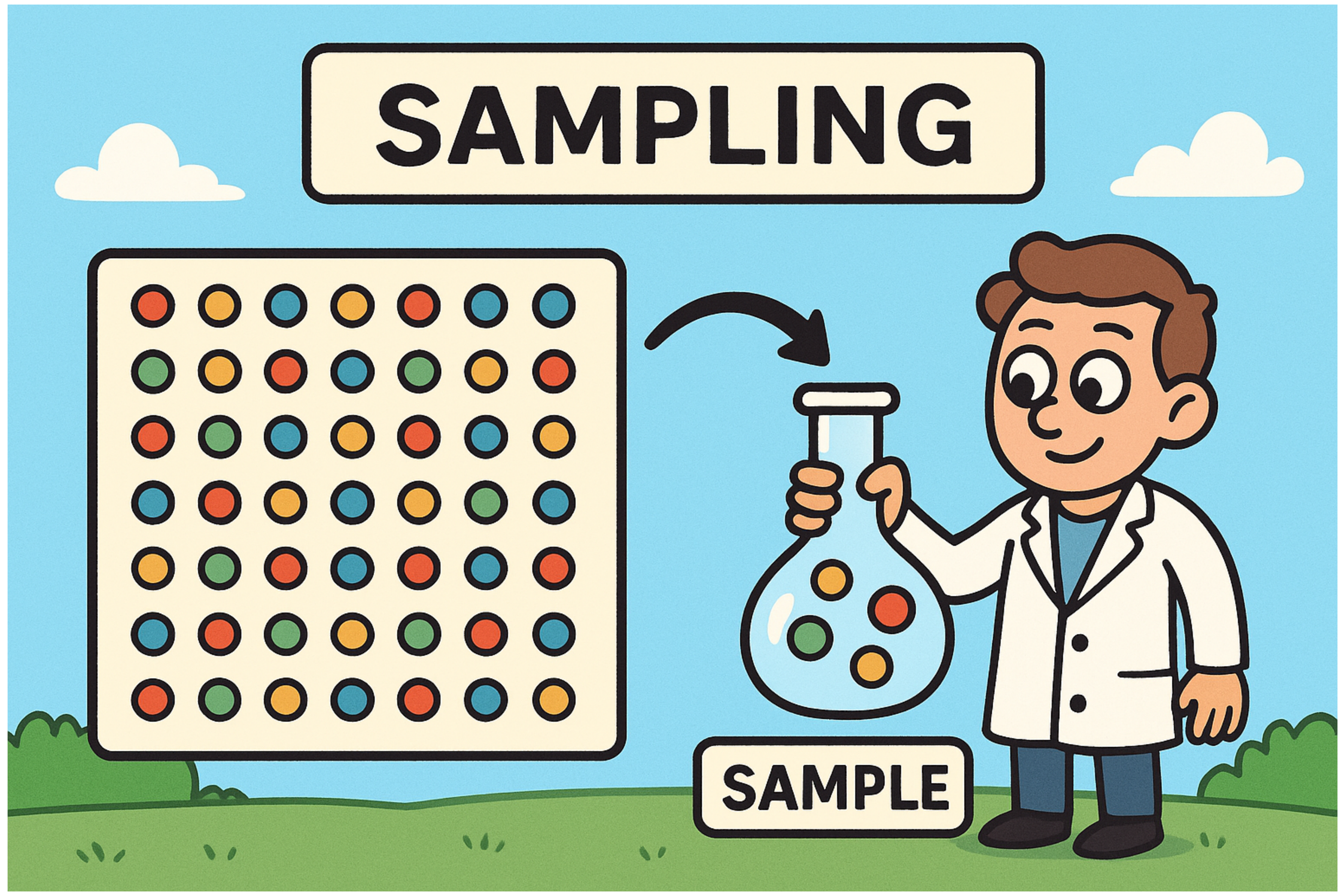
Example:

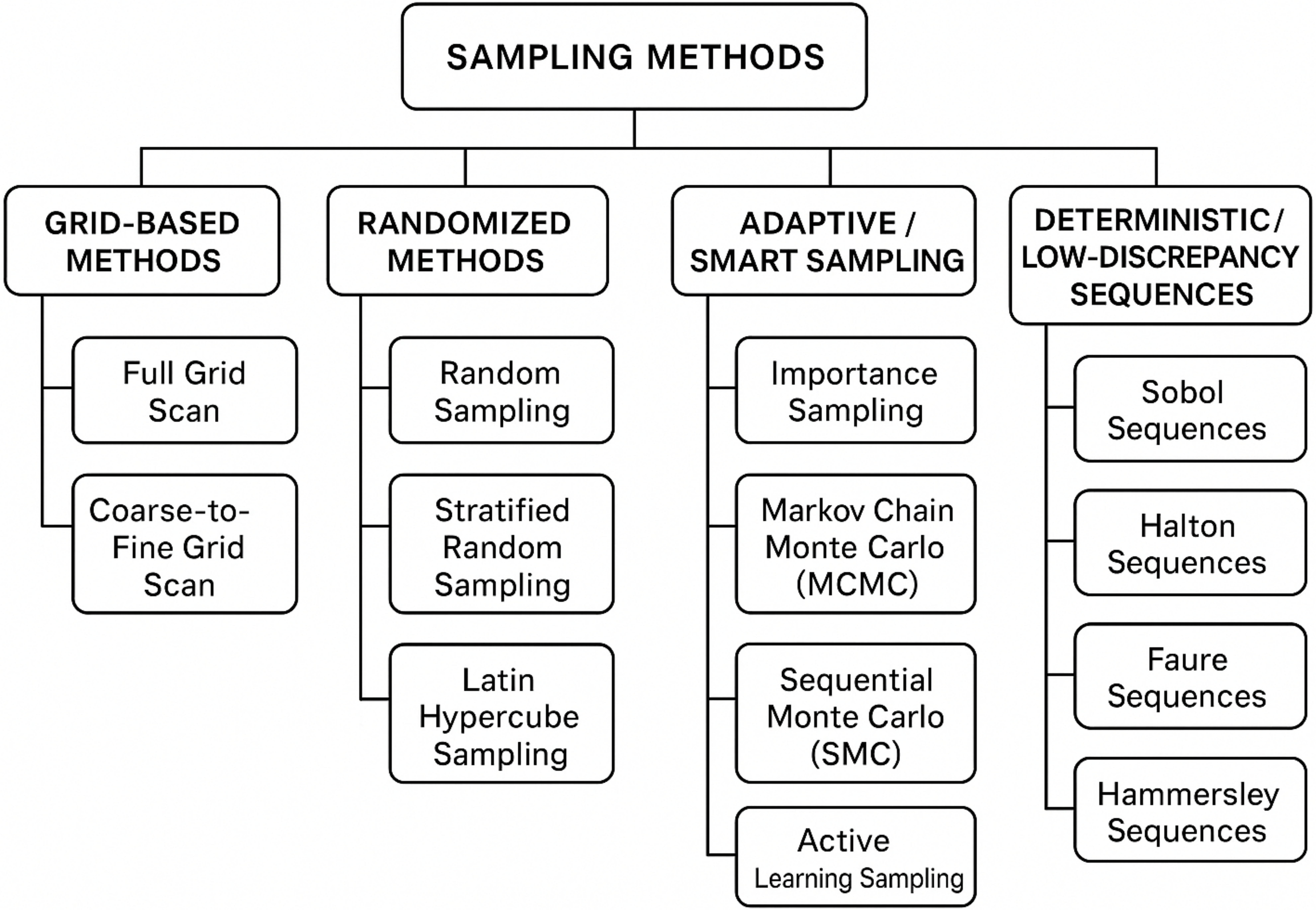
$$F_{2\mathrm{d}} = \left[2 + \cos\frac{x_1}{5}\cos\frac{x_2}{7}\right]^5$$

Task: Find the X values that correspond to F = 100 +- 5

Solution: Sample from a uniform distribution points for x1 and x2. Each time compute the function value and keep the sampled points that satisfy the condition

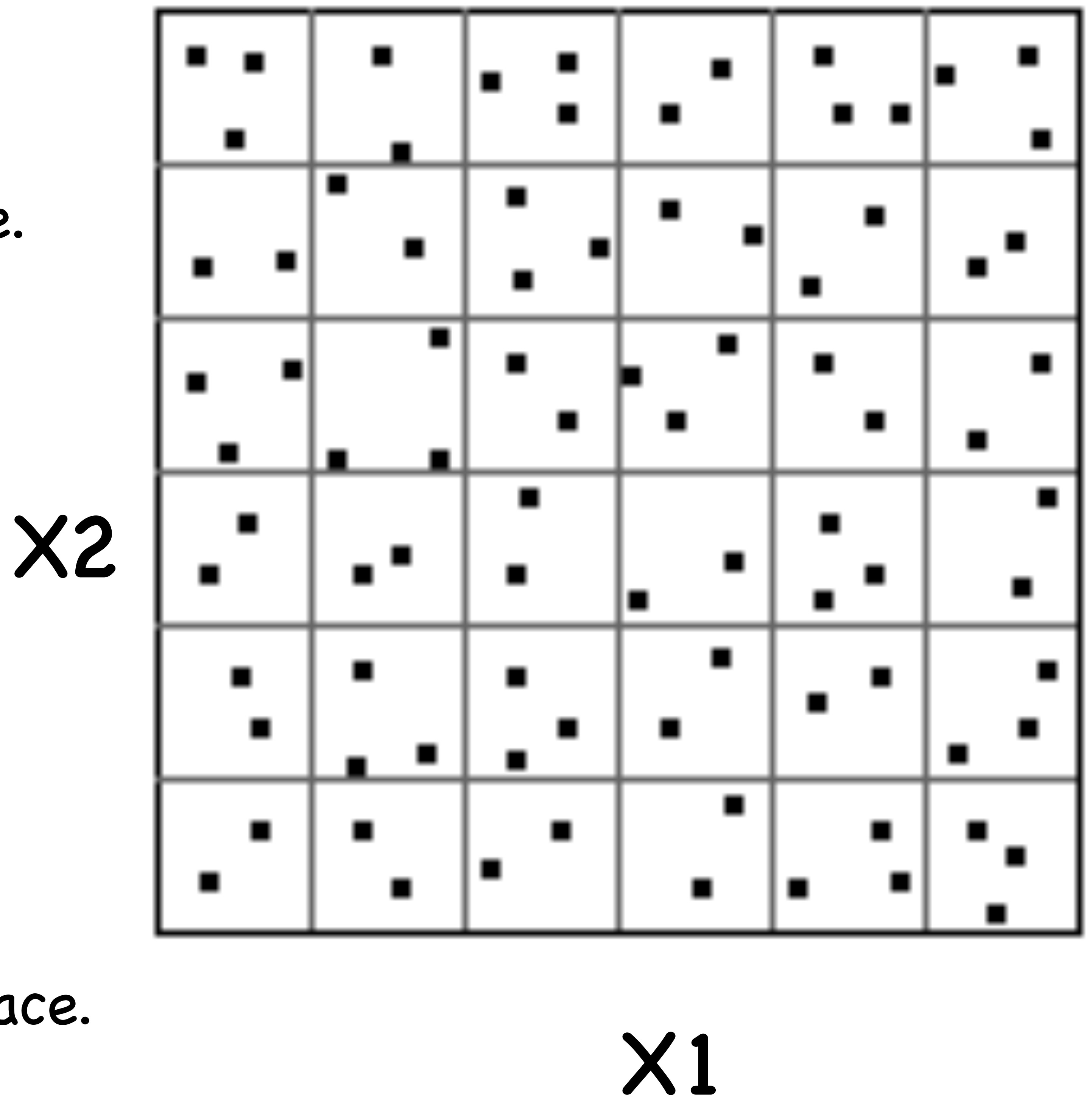
There are many sampling methods, which one shall we use?




SAMPLING METHODS
GRID-BASED METHODS
Full Grid Scan
Coarse-to-Fine Grid Scan
RANDOMIZED METHODS
Random Sampling
Stratified Random Sampling
Latin Hypercube Sampling
ADAPTIVE / SMART SAMPLING
Importance Sampling
Markov Chain Monte Carlo (MCMC)
Sequential Monte Carlo (SMC)
Active Learning Sampling
DETERMINISTIC/ LOW-DISCREPANCY SEQUENCES
Sobol Sequences
Halton Sequences
Faure Sequences
Hammersley Sequences

Full Grid sampling:

- Curse of dimensionality: The number of grid points grows exponentially with the number of parameters.
- Computationally expensive: Requires huge resources for high-dimensional parameter spaces
- Inefficient: Many sampled points may lie in unimportant regions where the function is negligible.
- Rigid resolution: Cannot adapt focus to regions of interest; same effort is spent everywhere.



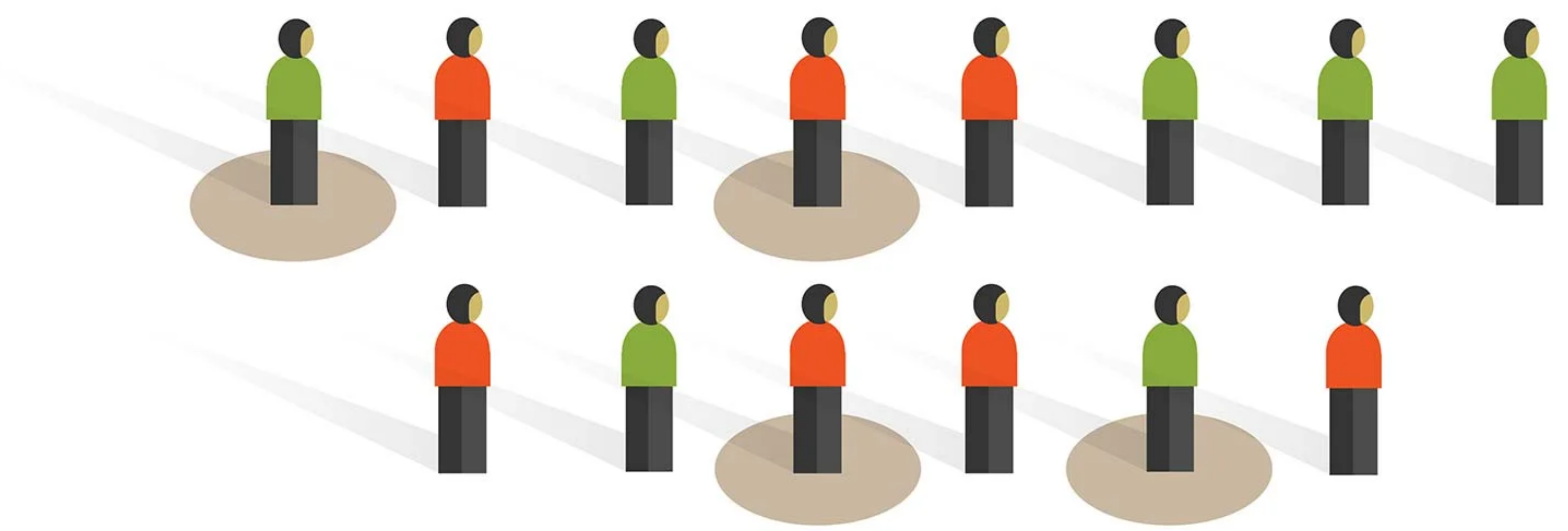
Coarse Grid sampling:

- Low resolution: May miss fine features, narrow peaks, or sharp boundaries in the parameter space.
- Risk of bias: If the true optimum lies between grid points, it won't be captured.
- Poor generalization: Gives only a rough idea of the landscape, not accurate for detailed analysis.

Simple random sampling:

- Not always representative
- Inefficient for heterogeneous population
- Does not guarantee convergence

Simple random sampling



MCMC sampling:

$$X_{t+1} = \begin{cases} x' & \text{with probability } \min\left(1, \frac{\pi(x')\, q(X_t|x')}{\pi(X_t)\, q(x'|X_t)}\right), \\ X_t & \text{otherwise.} \end{cases}$$

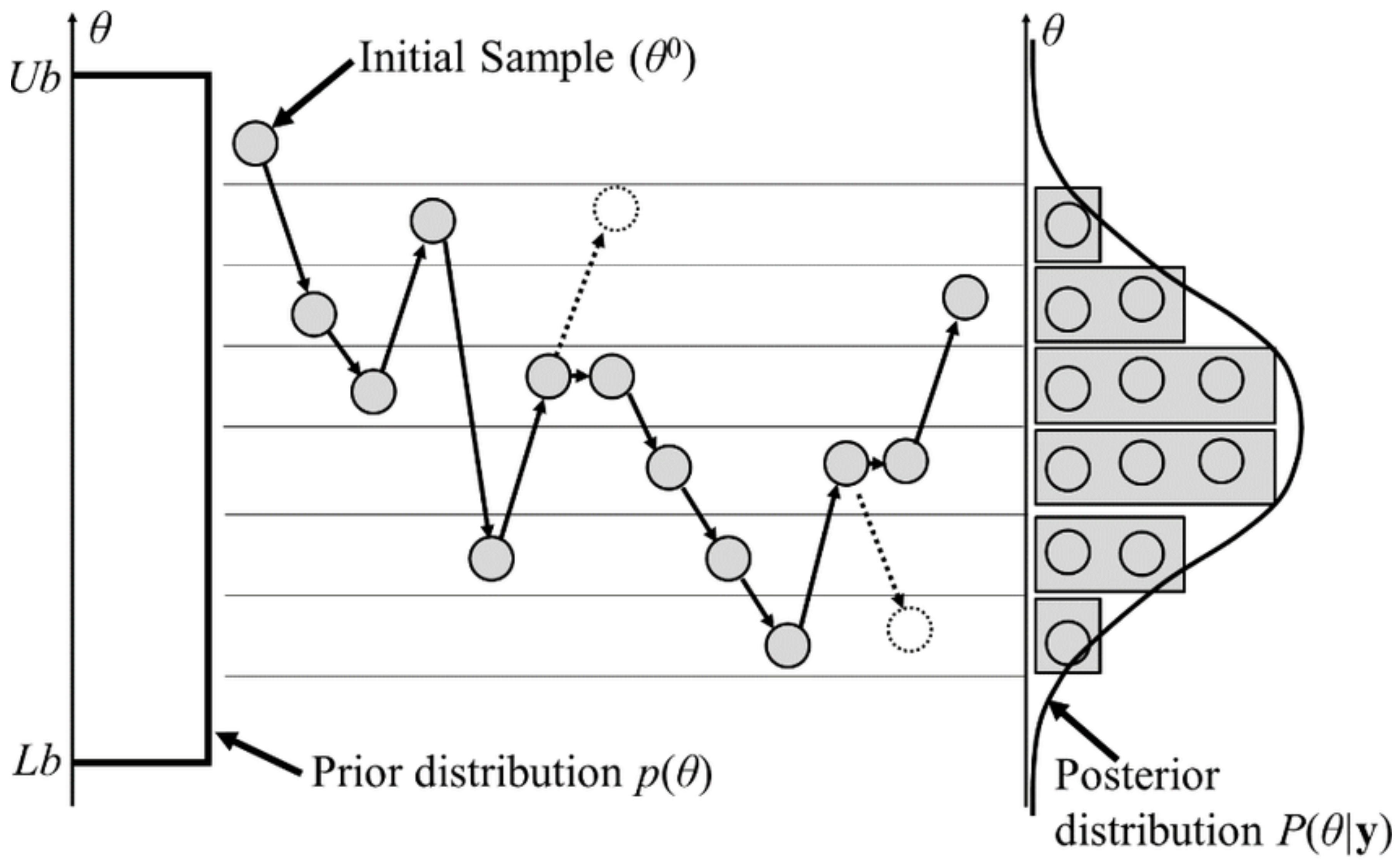
x' is the proposed point.

X_t is the current point

$\pi(x)$ is the target distribution

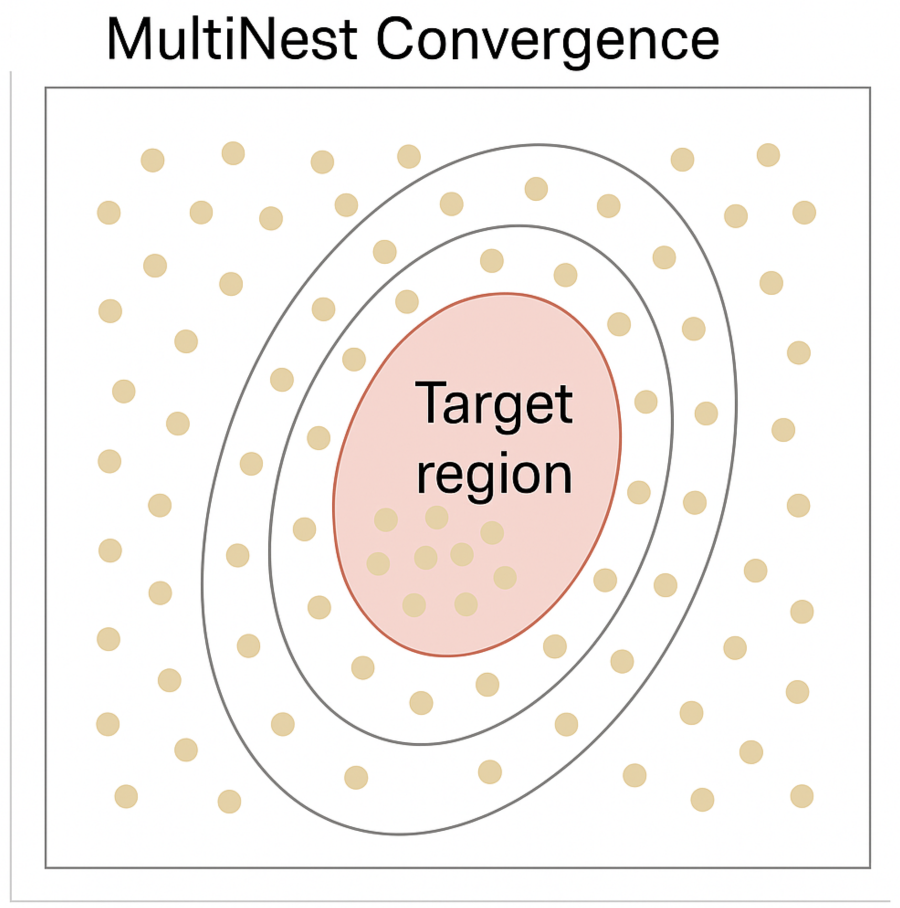
Drawbacks of MCMC

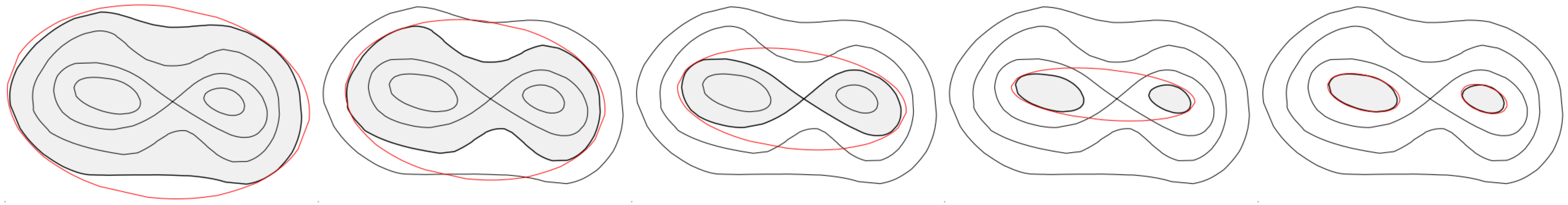
- **Slow Convergence:** The chain may take many iterations to reach the target distribution.
- **Computationally Expensive:** High-dimensional distributions require many iterations and likelihood evaluations.
- **Generalization to degenerate minima:** easily stuck to one minimum



Nested Sampling

MultiNest sampling:





MultiNest converges by using Bayesian evidence calculation, maintaining a set of live points sampled from the prior and iteratively replacing the lowest-likelihood point with a new one drawn from the prior but constrained to higher likelihood regions.

Generic drawbacks:

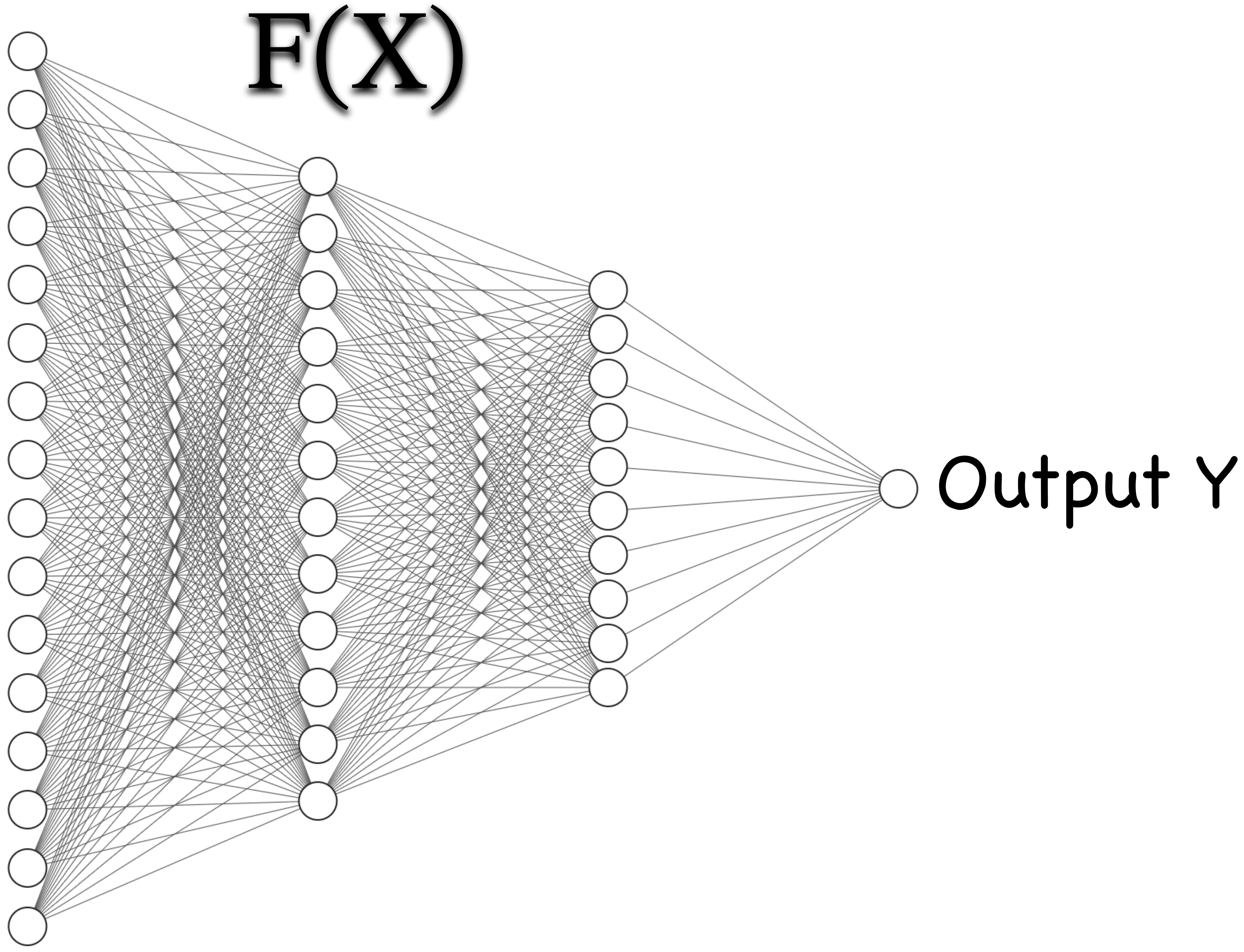
- Time consuming as computing likelihood requires to compute the exact value of the function at each sampled point
- Slow Convergence in High Dimensions
- Sampling Inefficiency by sampling low-likelihood regions or redundant points.
- MultiNest requires choosing the number of live points
- Difficulty with Multimodal Posteriors MCMC often struggles to jump between separated modes, while MultiNest can miss narrow or isolated modes if the live points don't cover them well.

ML assisted Sampling

Machine learning network can estimate the function values by adjusting the learnable weights in the hidden layers.

$$Y = \left(\prod_{\text{layer}} \sigma^{(l)} \left(W^{(l)}(\cdot) + b^{(l)} \right) \right) (X_i),$$

Input Xi



ML regressor:

Maps each input X to the exact function value

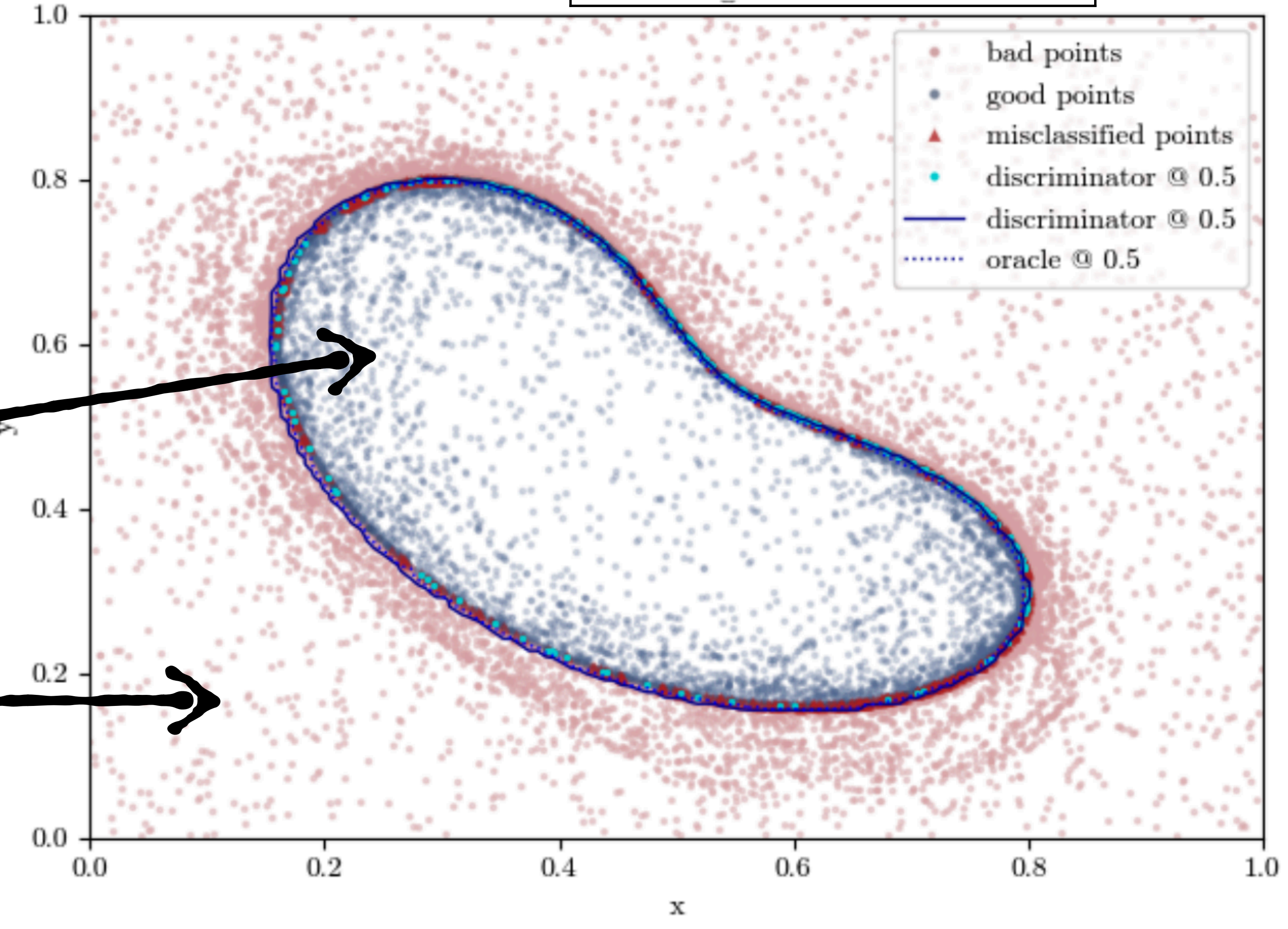
ML Classifier:

Maps each input X to two regions:

Inside target region

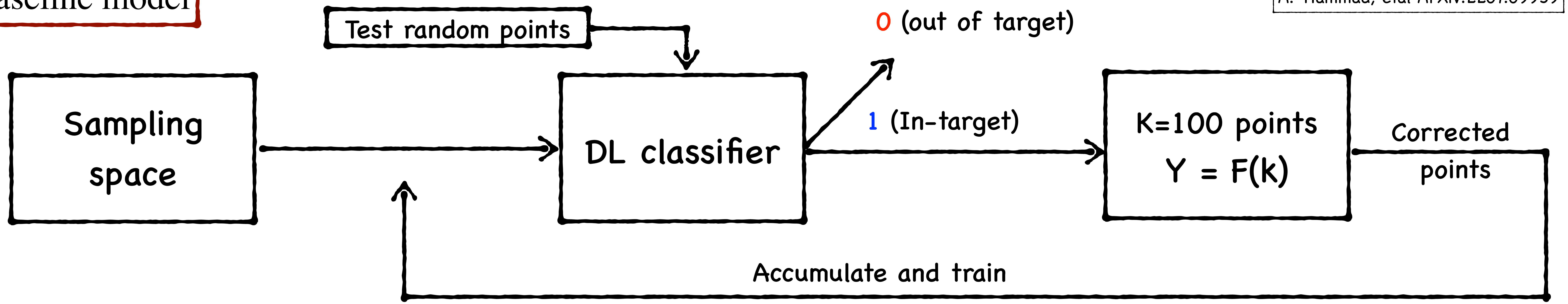
Outside target region

arXiv:2204.13950



Our baseline model

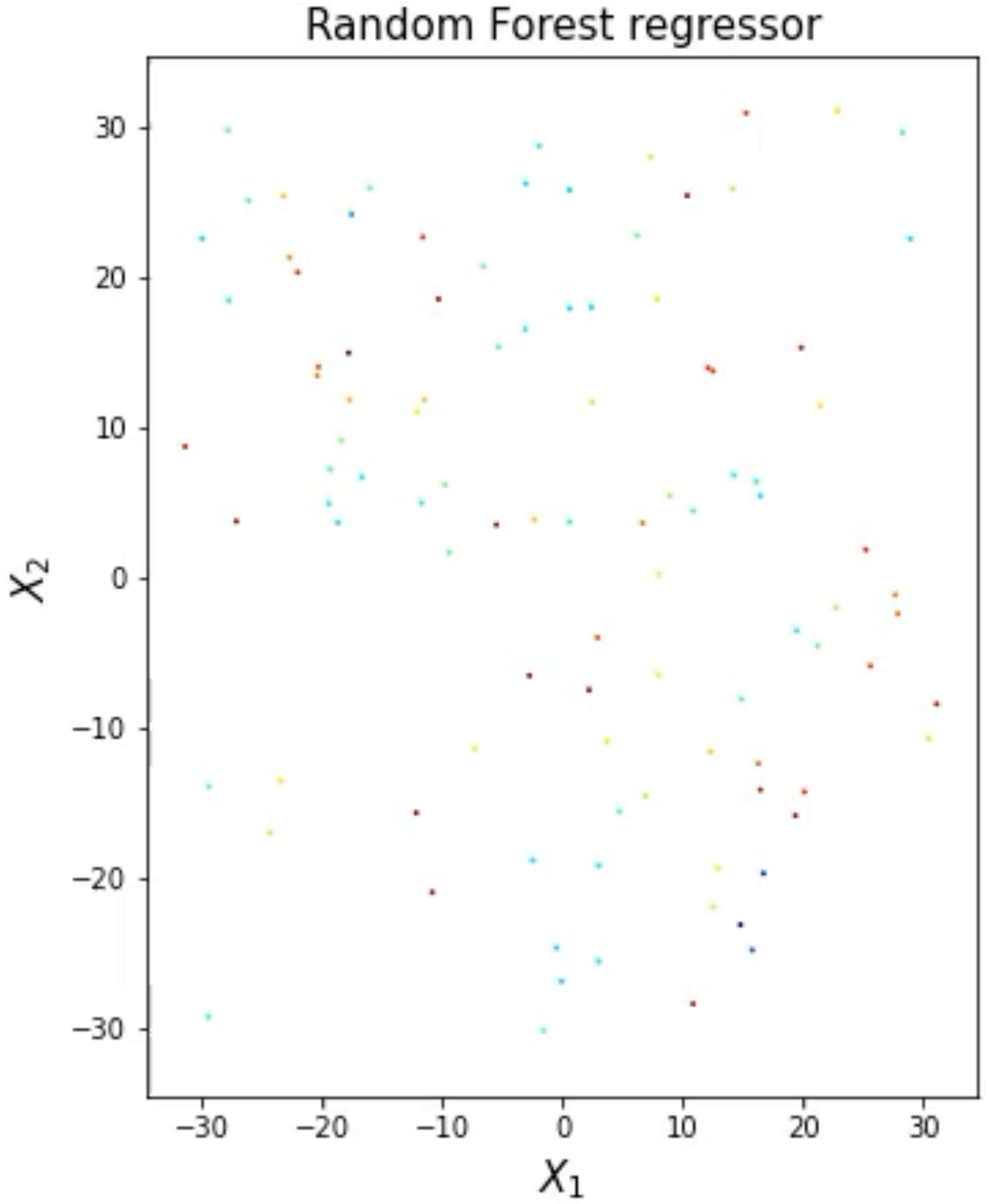
A. Hammad, etal ArXiv:2207.09959

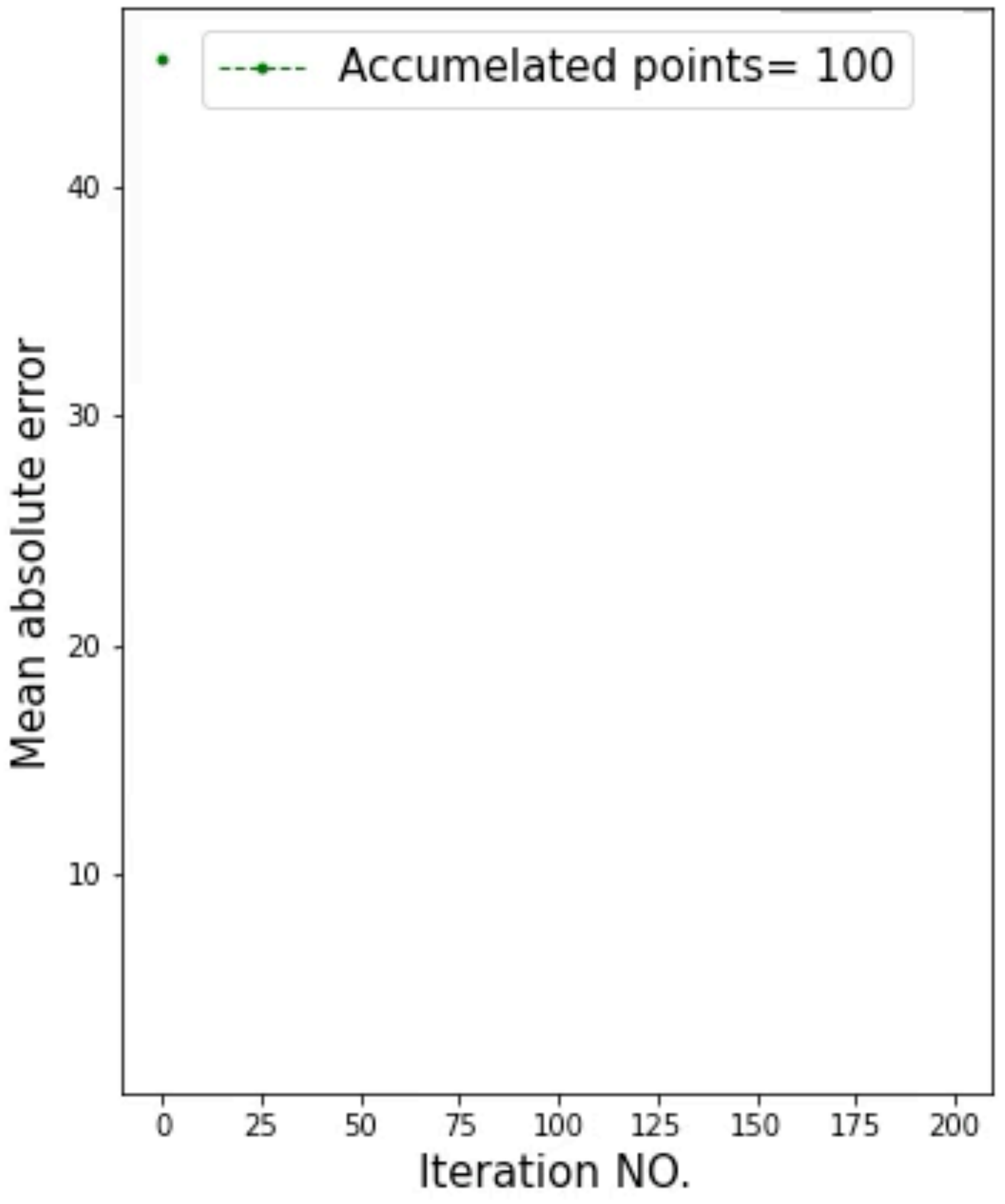


A likelihood free method, while accumulating more points the network learns the target region.
Using a DL classifier, the network learns from both in- and out-target region, avoiding the rejection sampling problems

$$F_{2d} = \left[2 + \cos\frac{x_1}{5}\cos\frac{x_2}{7}\right]^5$$

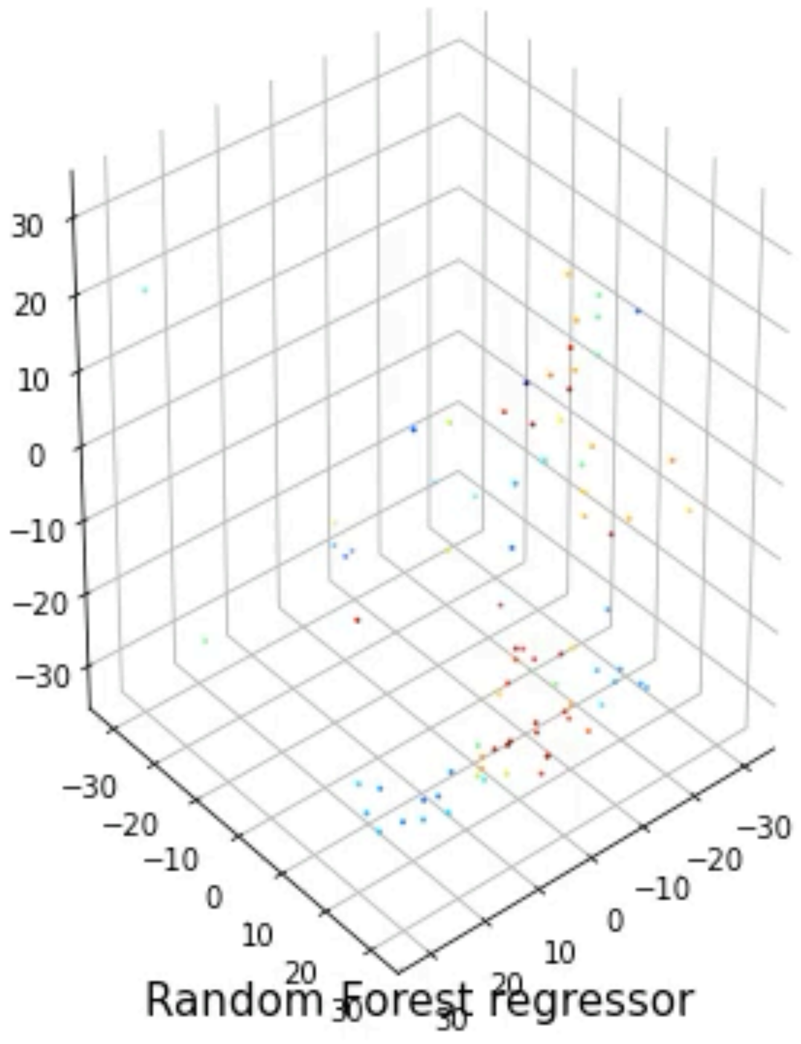
$$F_{3d} = \left[2 + \cos\frac{x_1}{7}\cos\frac{x_2}{7}\cos\frac{x_3}{7}\right]$$

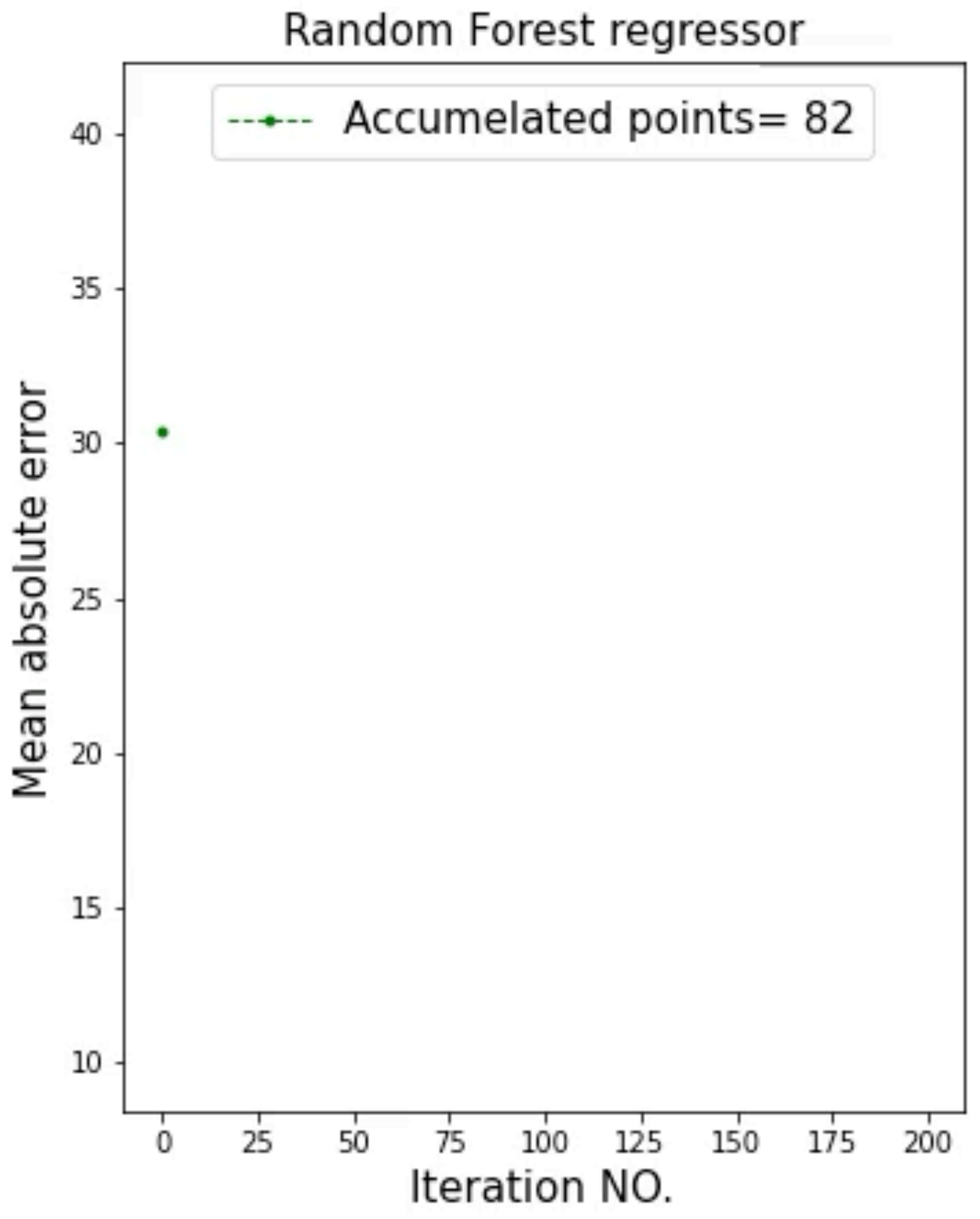




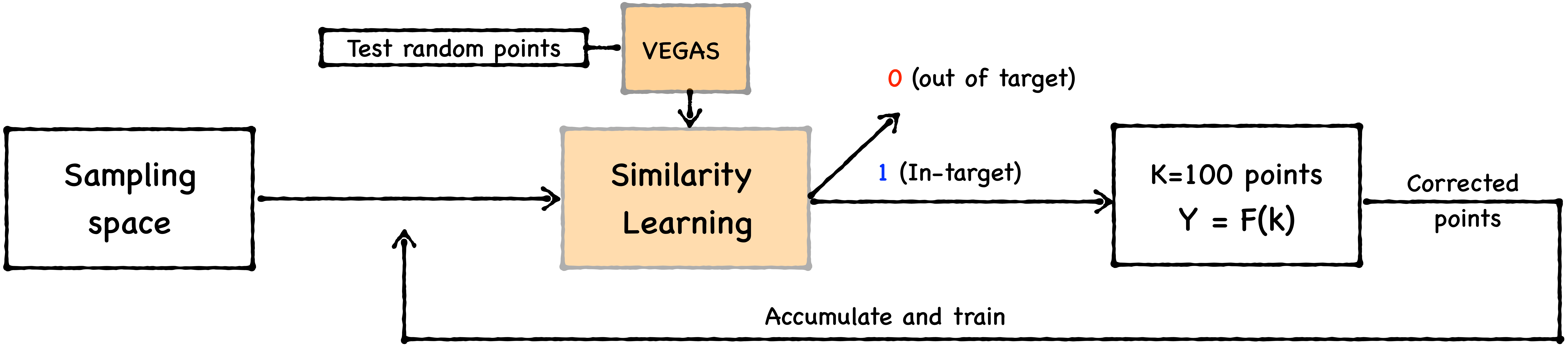
Target condition:

F = 100 +- 5





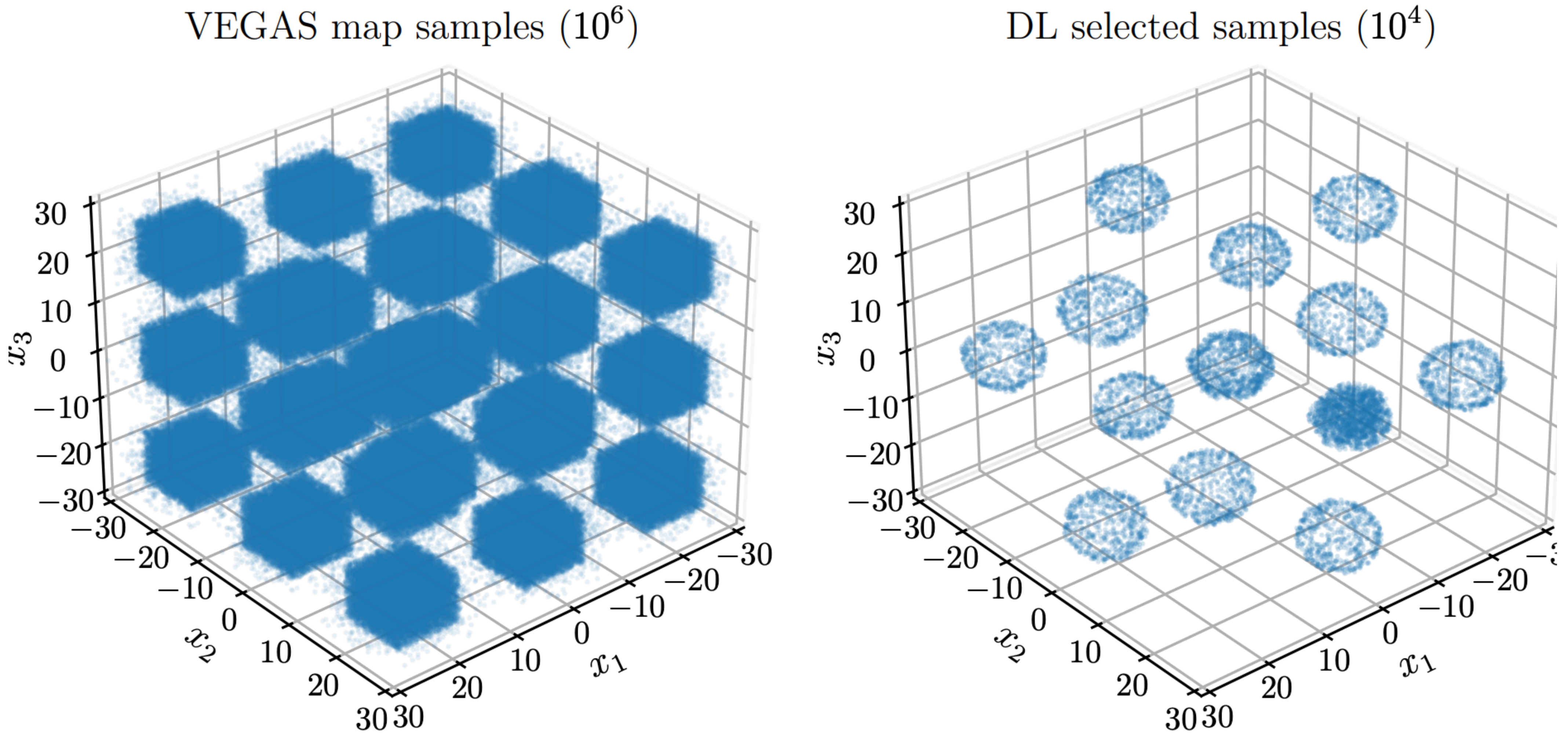
Improved ML assisted Sampling (DLScanner)



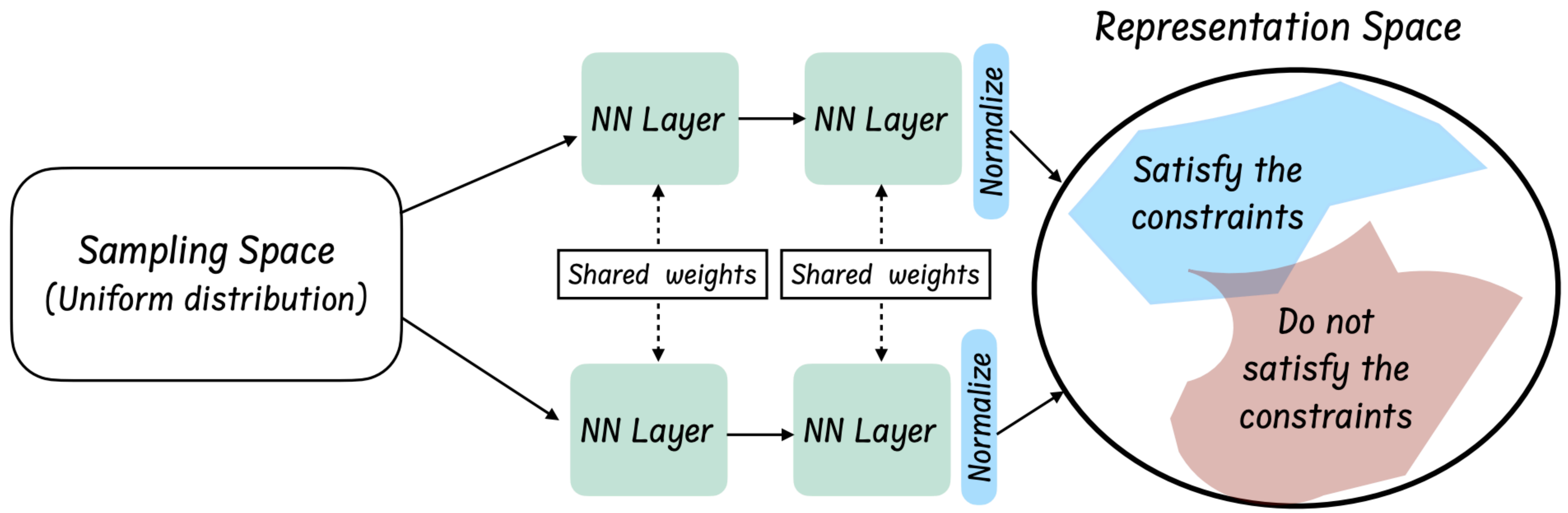
Similarity Learning for Parameter Scans. Shifts the problem from "predicting observables" to "learning geometry of viable regions."

Dynamic sampling via VEGAS.
avoids hyper-parameters tuning

arXiv:2009.05112

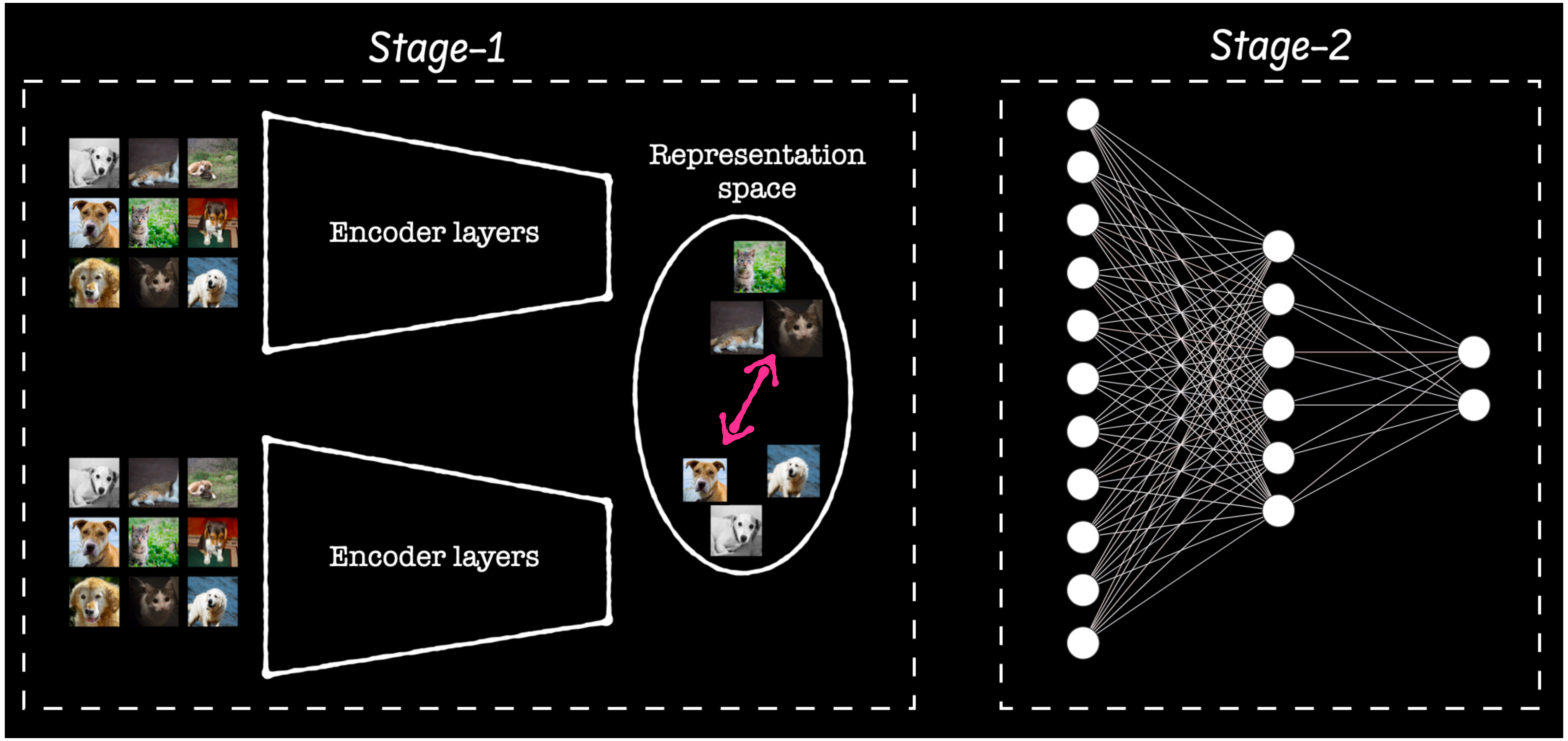


Very efficient in large dimension scan

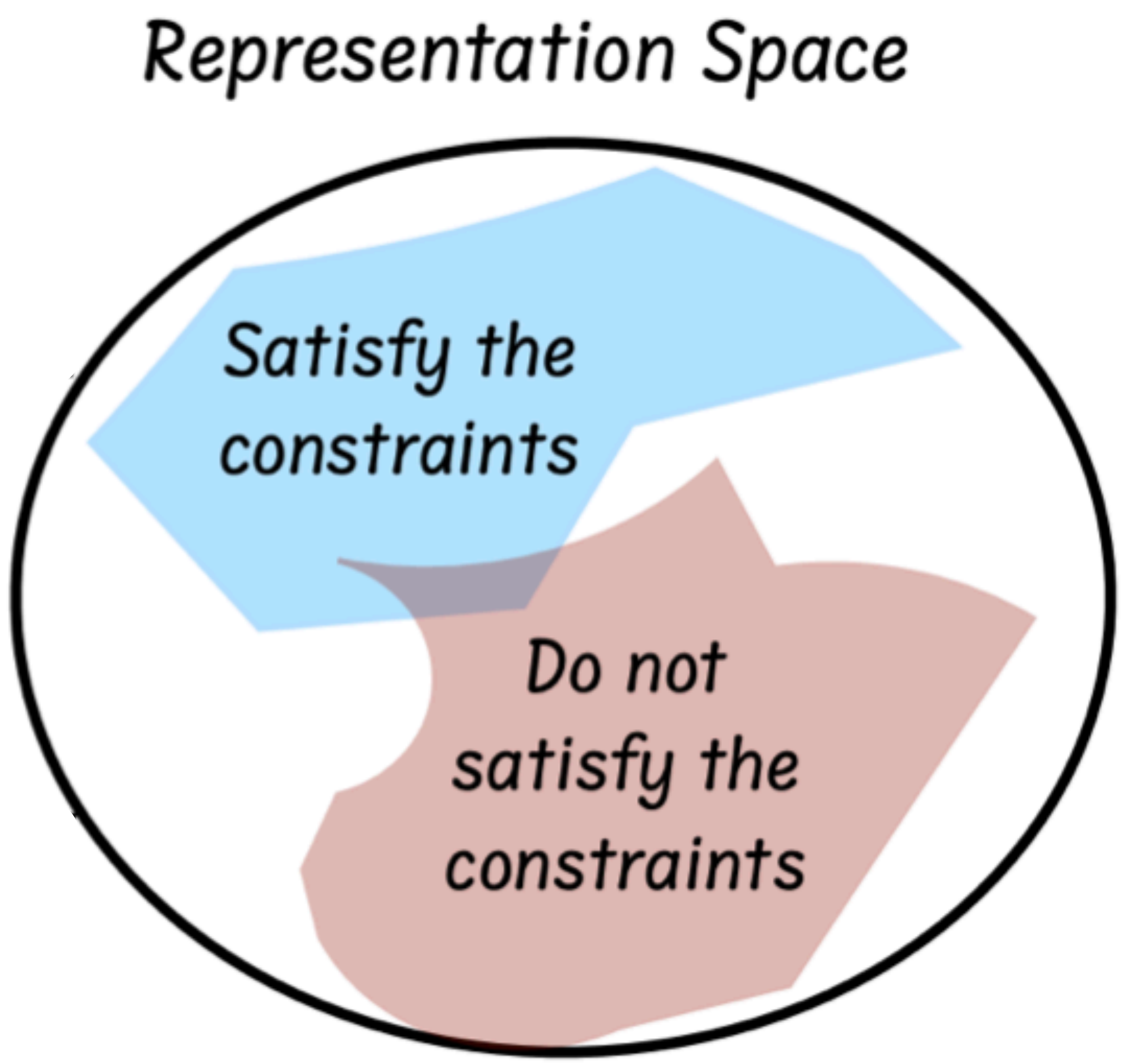


The role of similarity learning

The goal of similarity learning is to learn representations that capture meaningful features of the input data by minimize the distance between points inside the target region



arXiv:2004.11362

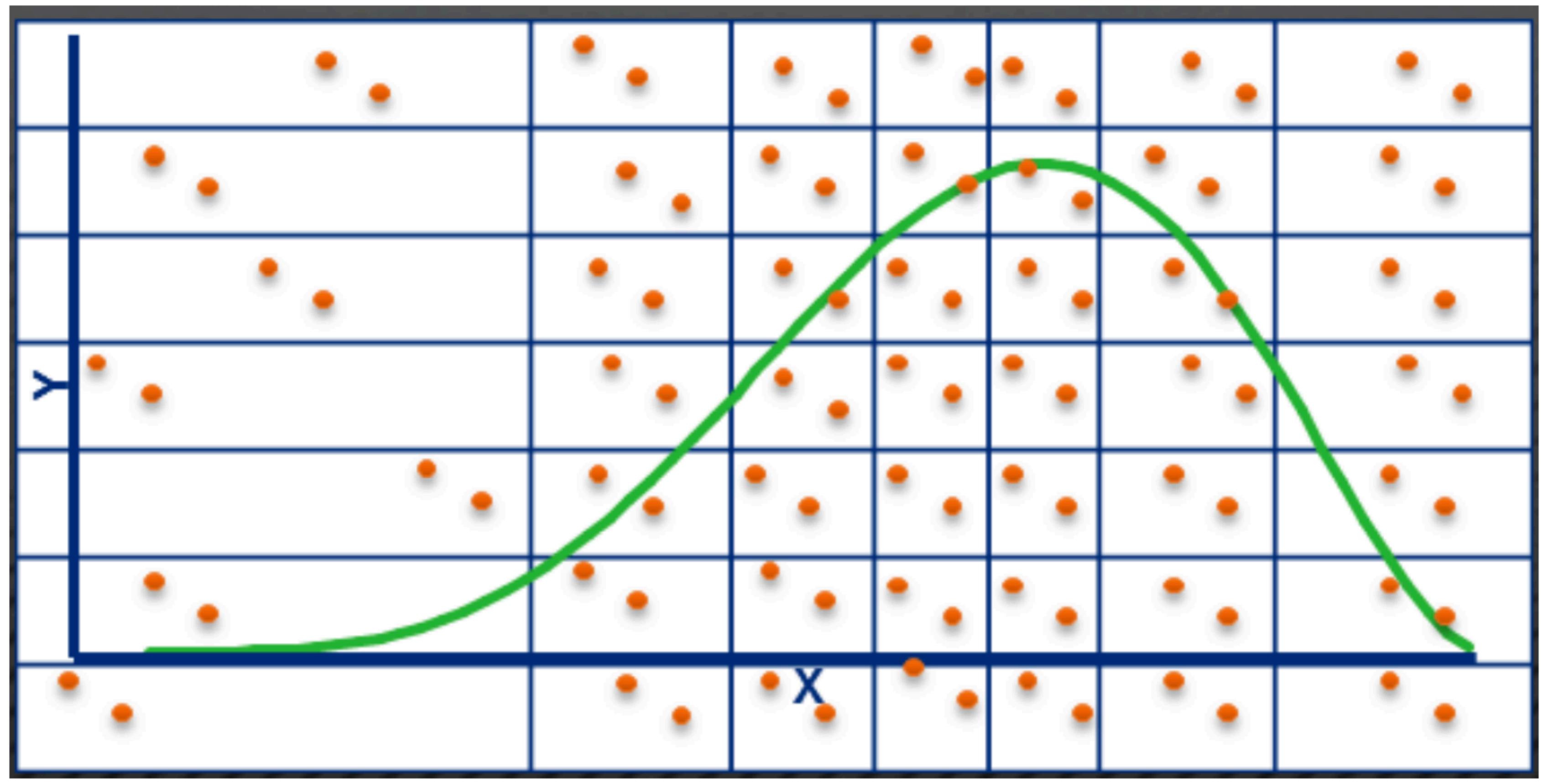


Representation space is a low dimensional space in which points that satisfy the constraints grouped together. This enhances the network convergence and make it valid for scan in high dimension

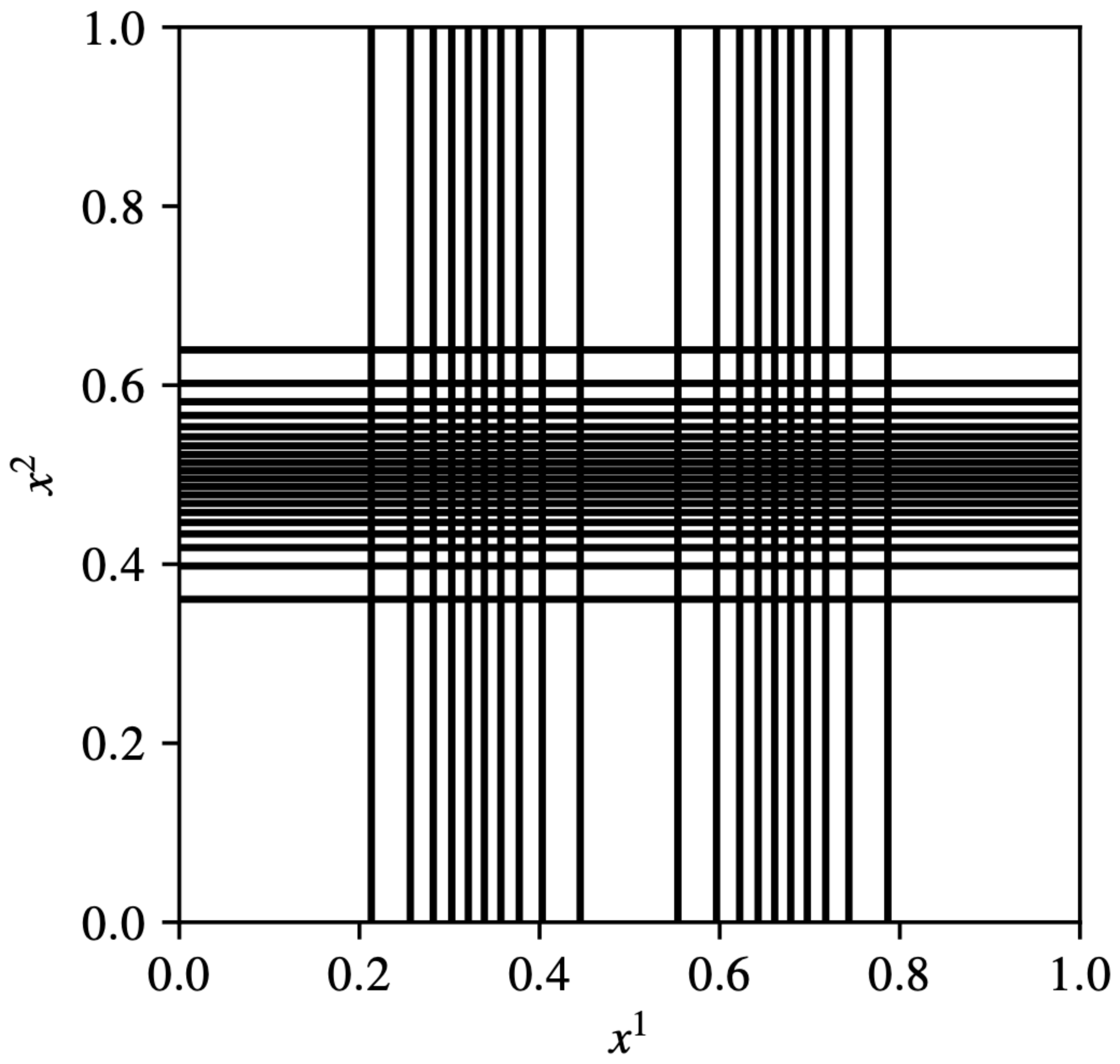
The role of VEGAS

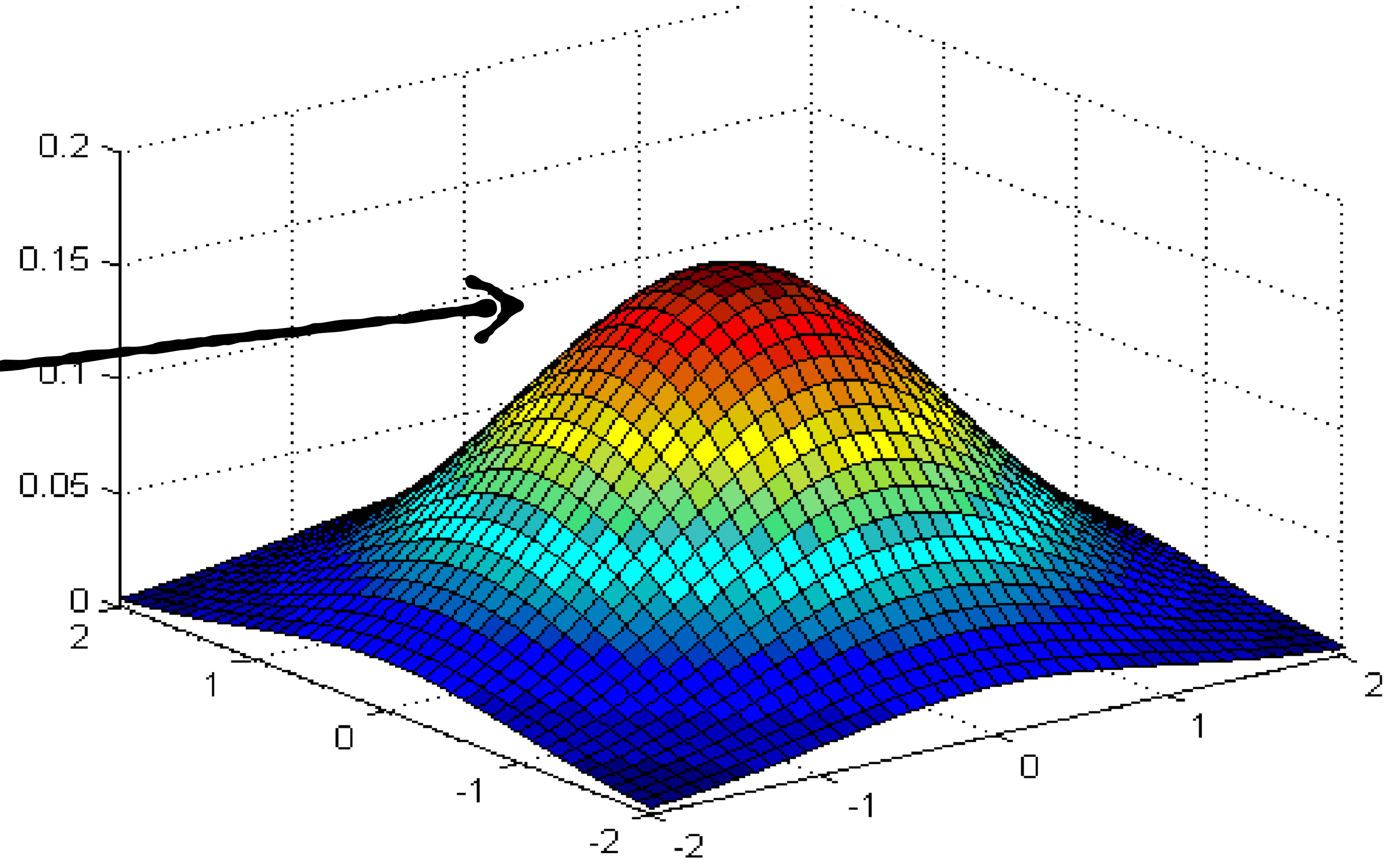
VEGAS is a well-known algorithm originally developed for adaptive multidimensional Monte Carlo integration

For integration, it suggests more points around the integral



Suggested points





VEGAS suggests point near to the target region to the ML network for prediction.

BaseLine ML assisted scan uses random points for network prediction

Results

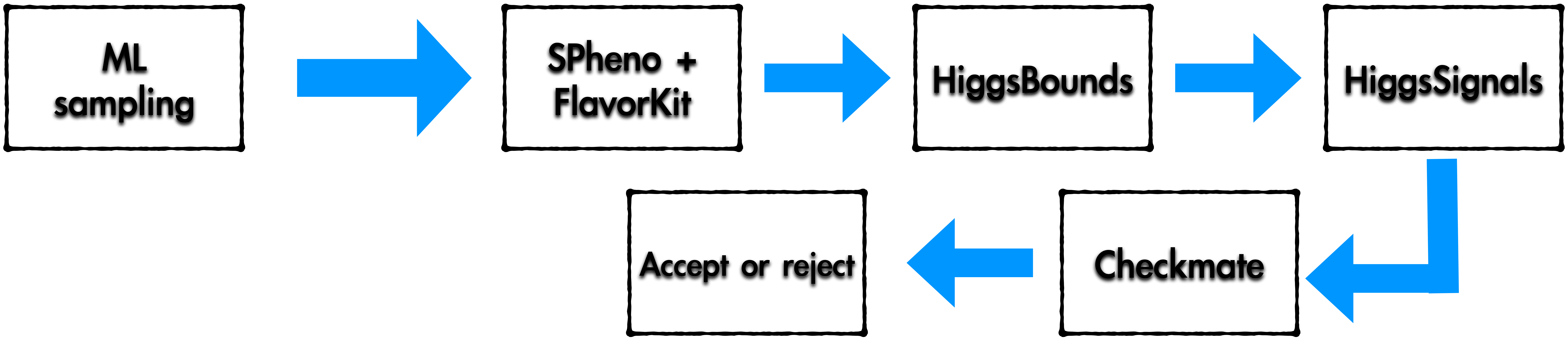
(Application for particle physics)

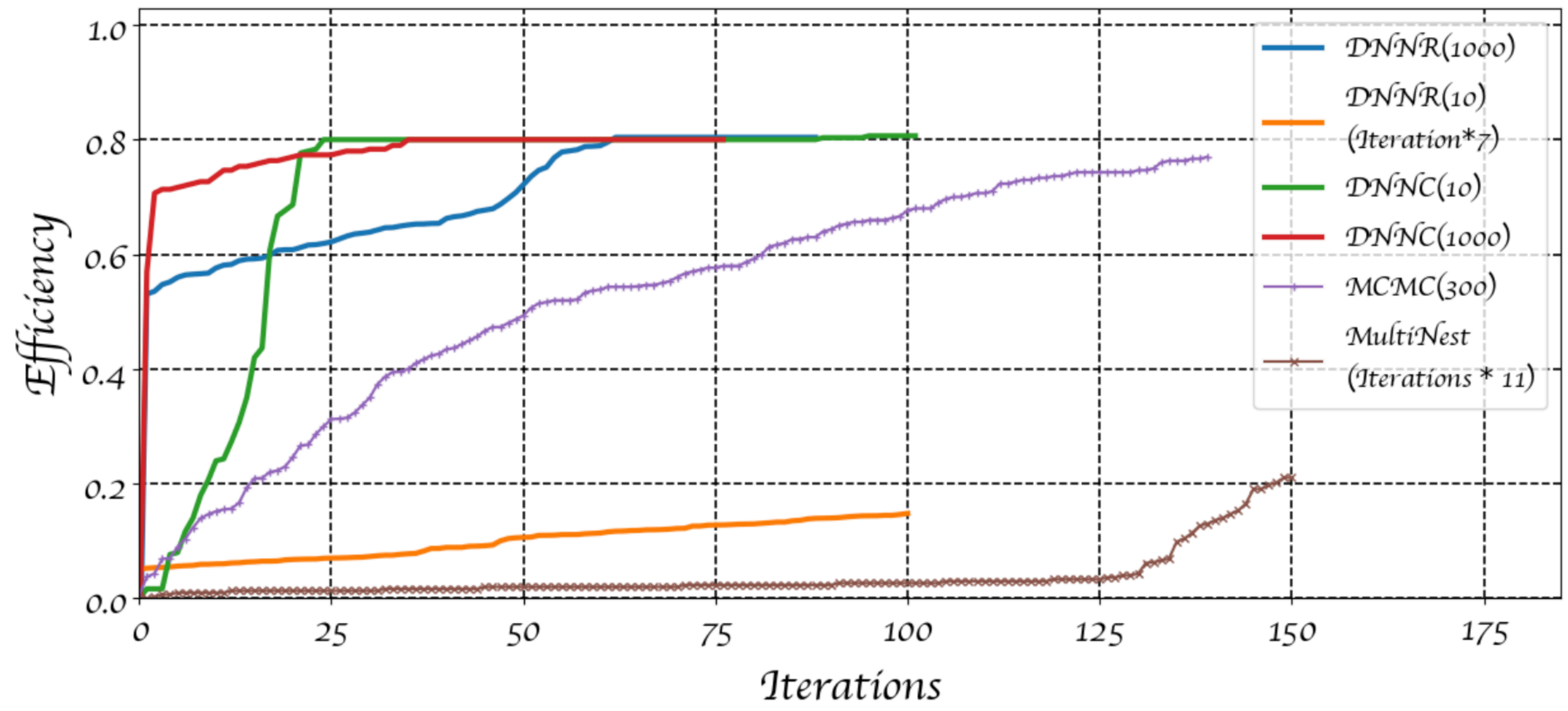
Scan over the scalar potential of the THDM to find the parameter space that satisfy
All current theoretical and experimental constrains

$$V_\phi = m_{11}^2(\phi_1^\dagger\phi_1) + m_{22}^2(\phi_2^\dagger\phi_2) - \left[m_{12}^2(\phi_1^\dagger\phi_2) + \text{h.c.}\right] + \lambda_1(\phi_1^\dagger\phi_1)^2 + \lambda_2(\phi_2^\dagger\phi_2)^2$$
$$+ \lambda_3(\phi_1^\dagger\phi_1)(\phi_2^\dagger\phi_2) + \lambda_4(\phi_1^\dagger\phi_2)(\phi_2^\dagger\phi_1) + \frac{1}{2}\left[\lambda_5(\phi_1^\dagger\phi_2)^2 + \text{H.c.}\right],$$

Scan over 7 free parameters with the following ranges:

$$0 \le \lambda_1 \le 10, \quad 0 \le \lambda_2 \le 0.2, \quad -10 \le \lambda_3 \le 10, \quad -10 \le \lambda_4 \le 10,$$
$$-10 \le \lambda_5 \le 10, \quad 5 \le \tan\beta \le 45, \quad -3000\ \text{GeV}^2 \le m_{12}^2 \le 0\ \text{GeV}^2,$$





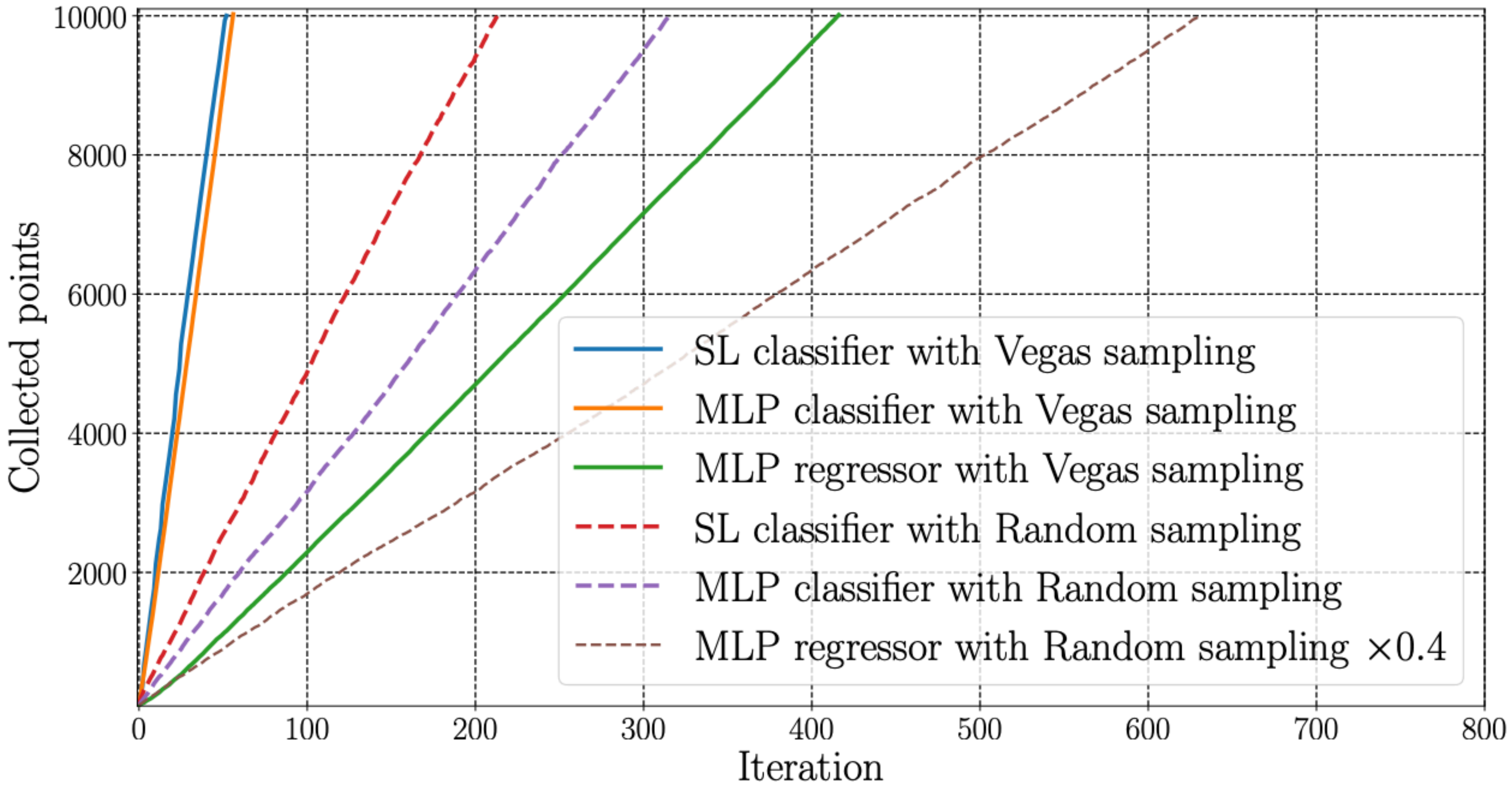
DNNR(i): ML regressor with (i) is the number of initial points

DNNC(i): ML Classifier

All sampling methods are required to accumulate 20K points in the target region.

DL classifier converges very fast to the target region. MCMC and MultiNet require large number of iteration to coverage to the target region

Similarity Learning with Vegas sampling (DLScanner) has the best perfromance over all other ML assisted sampling methods



Machine learning models are trained on CPU for fair comparison with other methods

In large dimension scan traditional methods fail

Results

Searching for the “golden” region of the NMSSM parameter space

arXiv:2508.13912

Given the current anomalies can we find the NMSSM parameter space that satisfy current constraints and fits all the anomalies ?

Anomalies

- The 95 GeV excess
- The 650 GeV excess
- Electro-Weakinos
- Muon g-2

Constraints

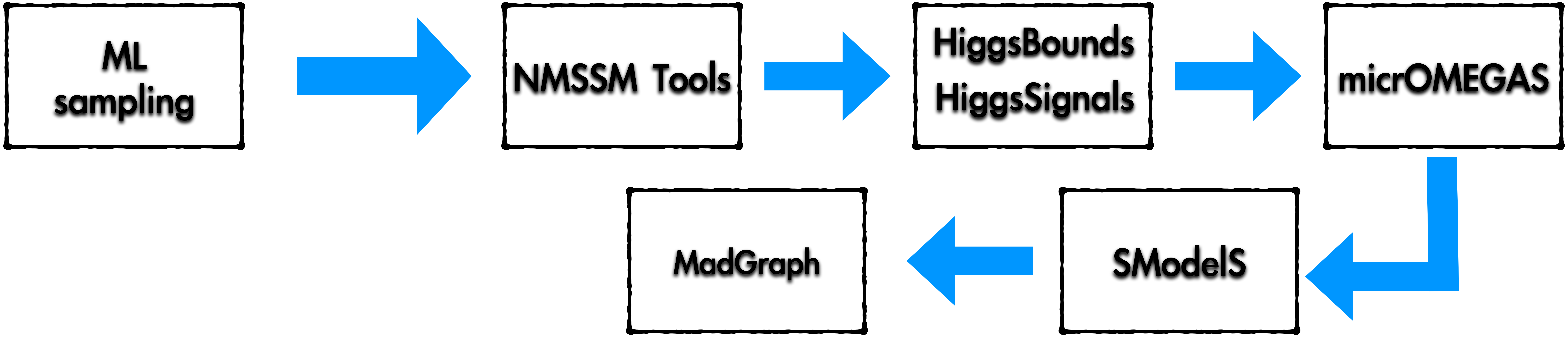
- Higgs measurements
- DM searches
- Low energy observables
- Direct collider searches

Scan over 12 free parameters with the following ranges:

	$\tan\beta$	λ	κ	A_λ
wide	[1.97, 10.9]	[0.013, 0.687]	[0.0058, 0.391]	[−5000, 480]
narrow	[3.2, 6.2]	[0.07, 0.42]	[0.05, 0.3]	[351, 834]
	A_κ	$\mu_{\rm eff}$	M_1	M_2
wide	[−621, 362]	[−244, 291]	[178, 3000]	[304, 10000]
narrow	[−300, −150]	[120, 220]	[500, 3000]	[750, 10000]
	M_3	A_t	M_{Q_3}	M_{U_3}
	[423, 5000]	[−5000, 1288]	[272, 10000]	[570, 10000]

Penalize the points that do not satisfy the constraints

$$P_{\rm constraint}(O^{\rm th}, O^{\rm exp}_{\pm 2\sigma}) = \begin{cases} 0, & O^{\rm exp}_{-2\sigma} < O^{\rm th} < O^{\rm exp}_{+2\sigma}, \\ |O^{\rm exp}_{+2\sigma} - O^{\rm th}|^2, & O^{\rm th} > O^{\rm exp}_{+2\sigma}, \text{ if } O^{\rm exp}_{+2\sigma} \text{ exists}, \\ |O^{\rm th} - O^{\rm exp}_{-2\sigma}|^2, & O^{\rm th} < O^{\rm exp}_{-2\sigma}, \text{ if } O^{\rm exp}_{-2\sigma} \text{ exists}, \end{cases}$$



The 95 GeV Excess

We consider light scalar of mass 95 +- 5 GeV

2 sigma excess at LEP

$$\mu_{b\bar{b}} = \frac{\sigma(e^+e^- \to Zh_{95} \to Zb\bar{b})}{\sigma(e^+e^- \to Zh_{95}^{\rm SM} \to Zb\bar{b})} = 0.117 \pm 0.057$$

1.7 sigma excess at ATLAS and CMS

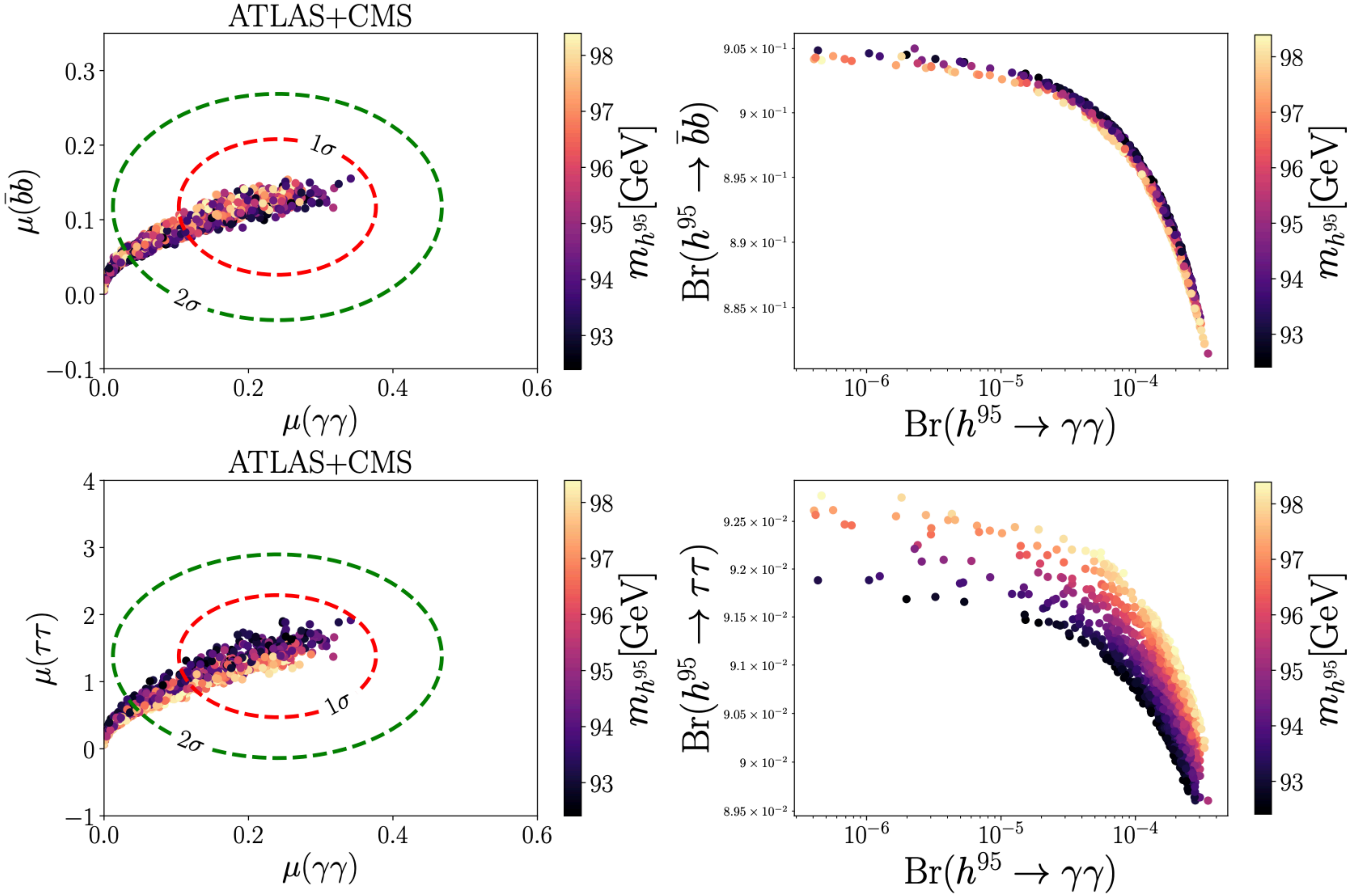
arXiv:2306.03889

$$\mu_{\gamma\gamma}^{LHC} = \frac{\sigma(gg \to H_1 \to \gamma\gamma)}{\sigma(gg \to H_{SM}^{95} \to \gamma\gamma)} = 0.24^{+0.09}_{-0.08} \,.$$

2.7 sigma excess at CMS

arXiv:2208.02717

$$\mu_{\tau\tau}^{LHC} = \frac{\sigma(gg \to H_1 \to \tau\tau)}{\sigma(gg \to H_{SM}^{95} \to \tau\tau)} = 1.38^{+0.69}_{-0.55} \,.$$

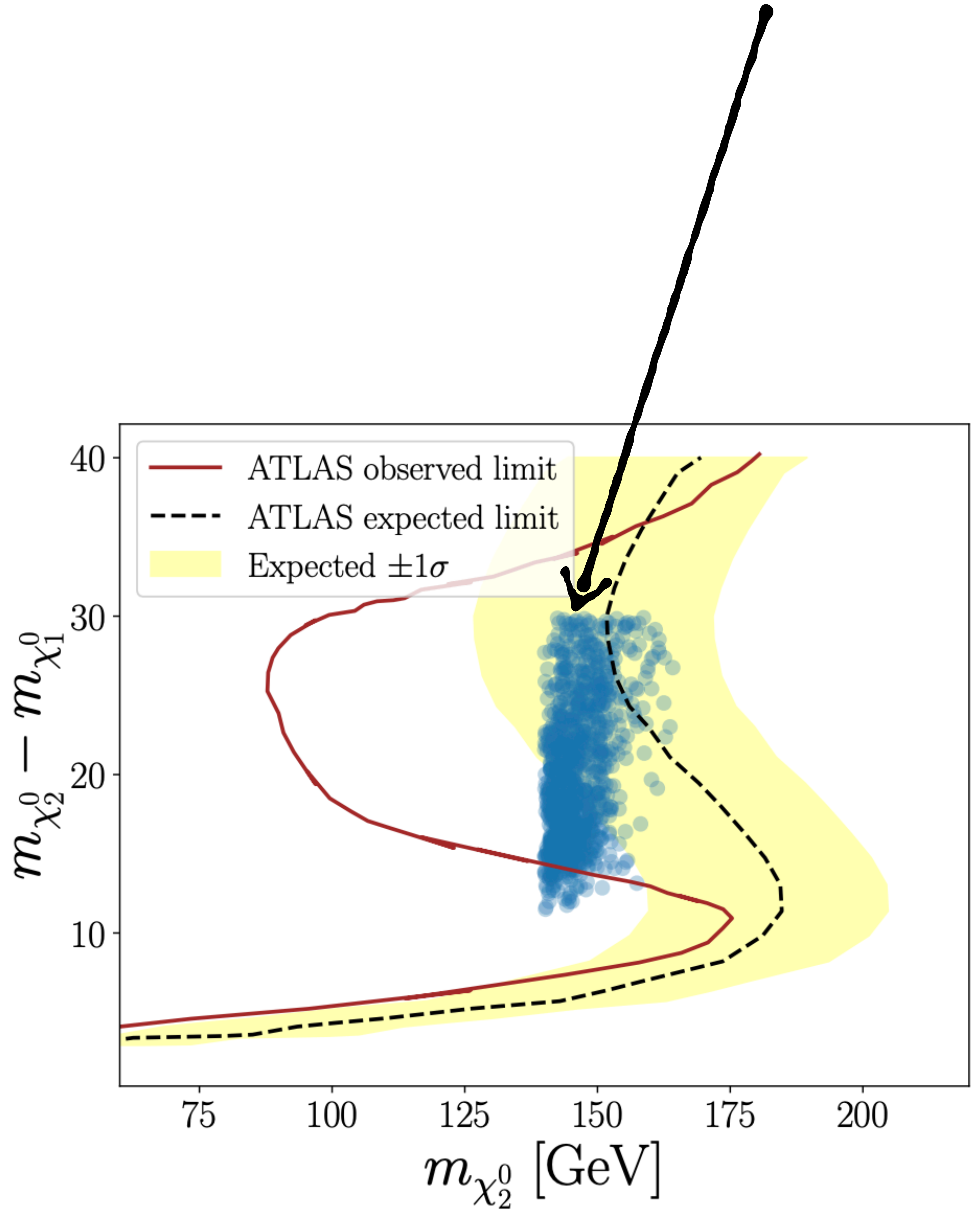


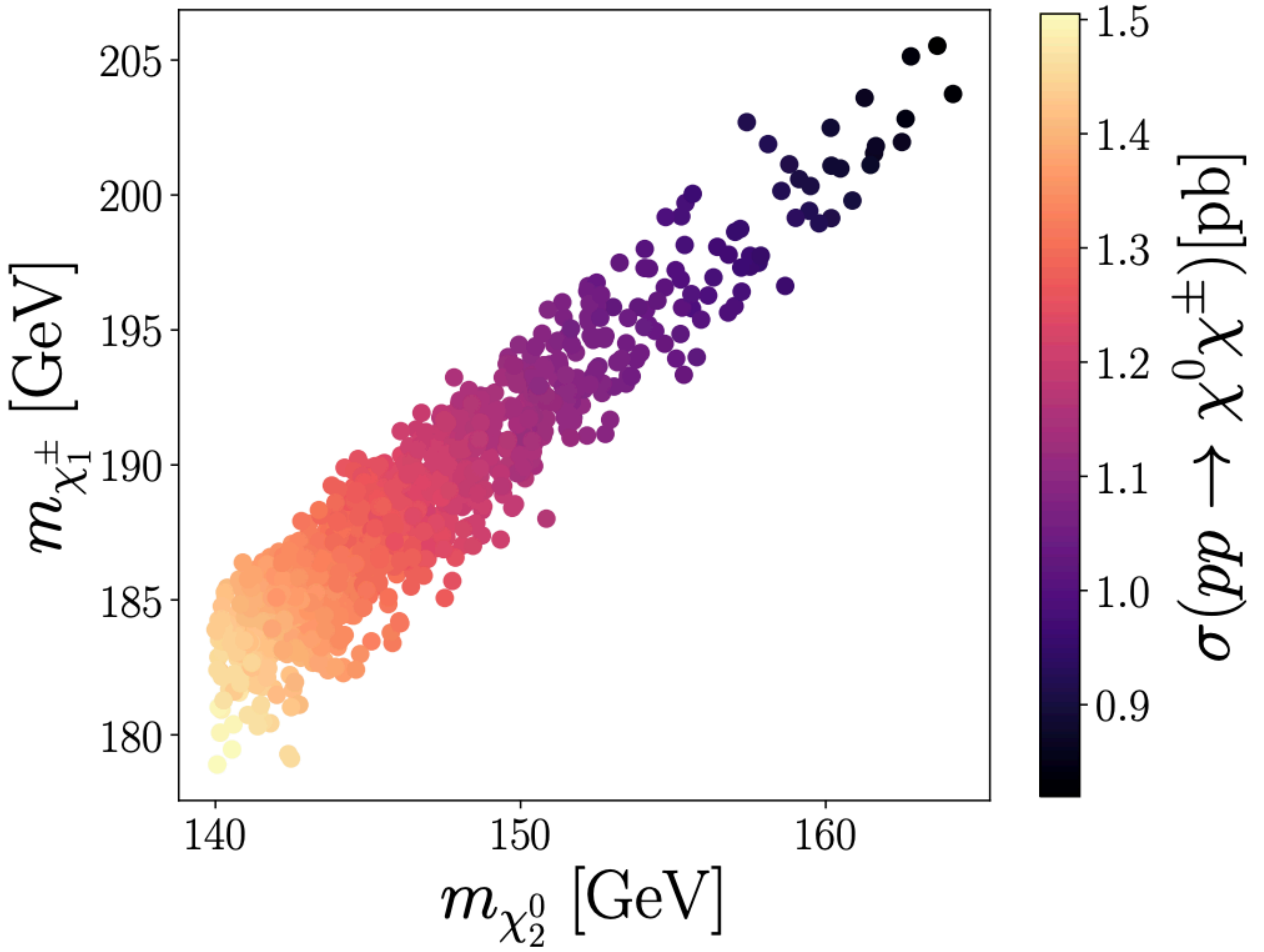
The EWino Excess

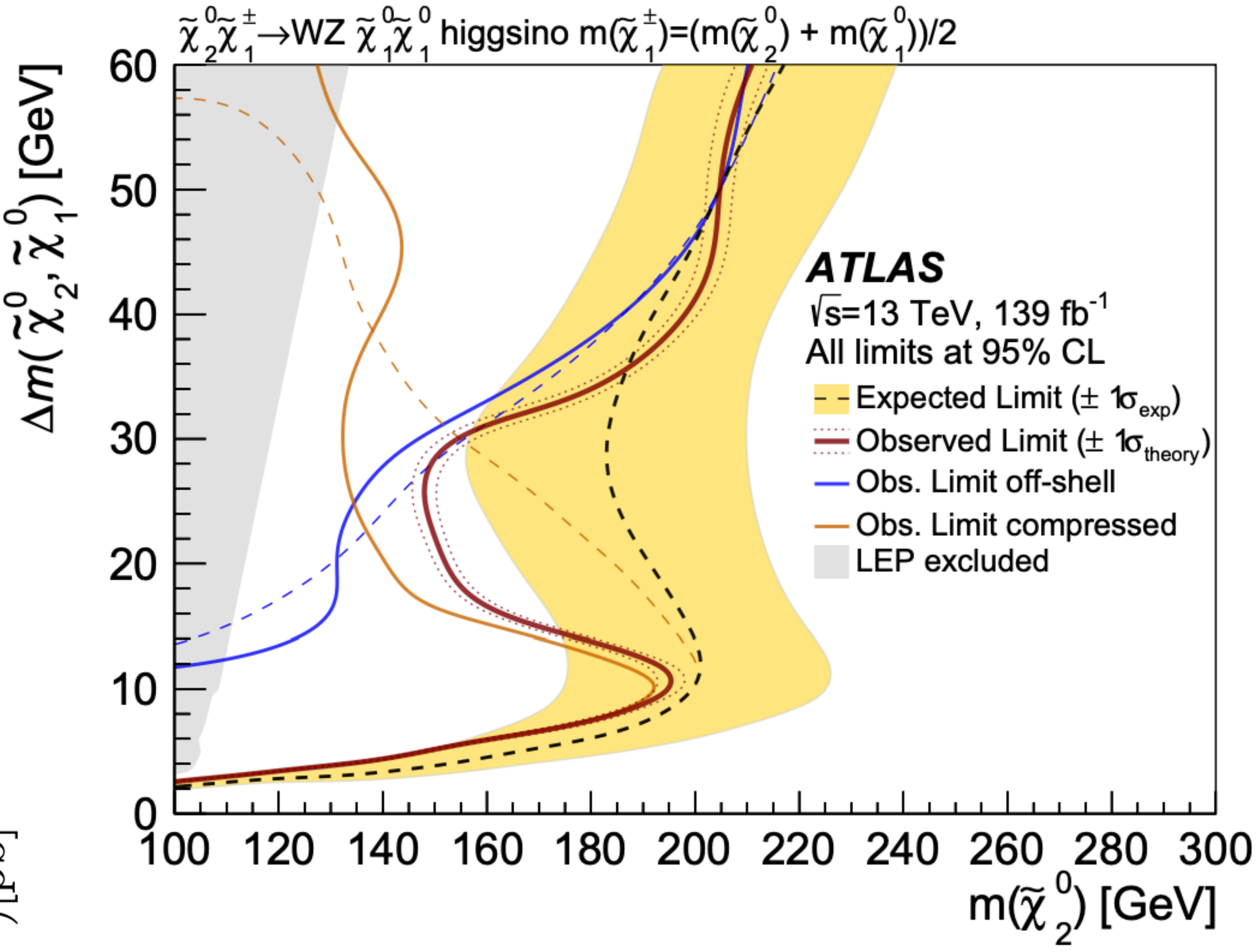
For compressed spectrum, ATLAS has found 2.4 sigma exces
for neutralino LSP dark matter mass around 150 GeV

arXiv:2106.01676

Scanned points



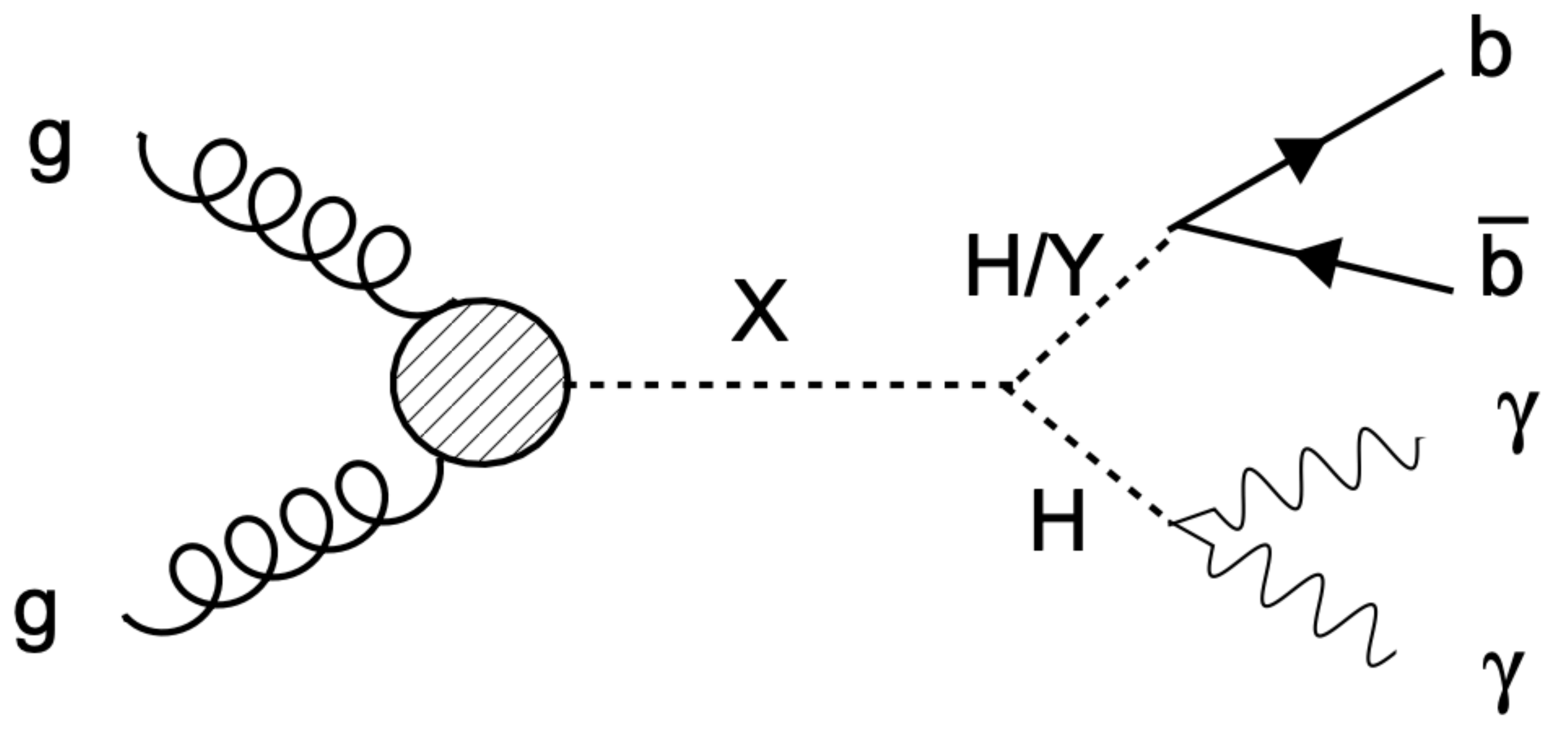




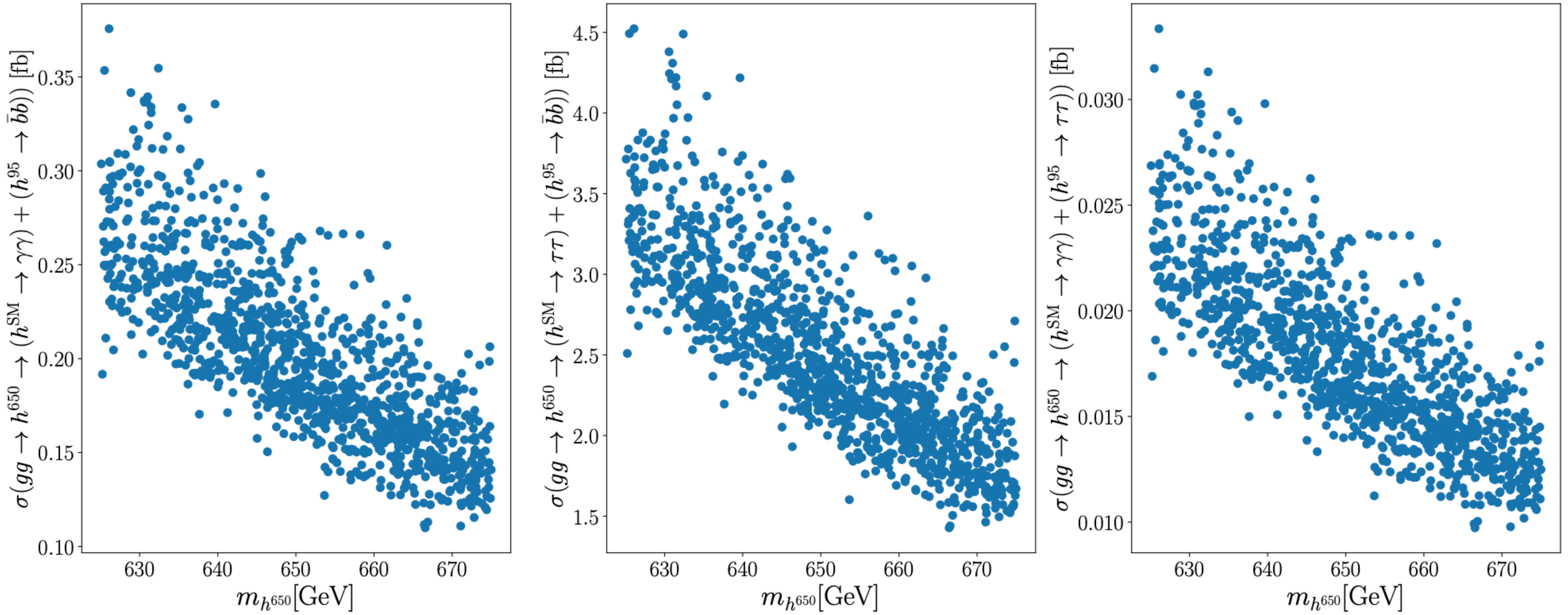
The 650 GeV Excess

3.8 sigma local excess has been reported by CMS
arXiv:2310.01643

$$\sigma_{bb\gamma\gamma} = \sigma(gg \to X_{650} \to (H_1 \to b\bar{b}) + (H_{SM} \to \gamma\gamma)) = 0.35^{+0.17}_{-0.13}\,\text{fb}\,.$$

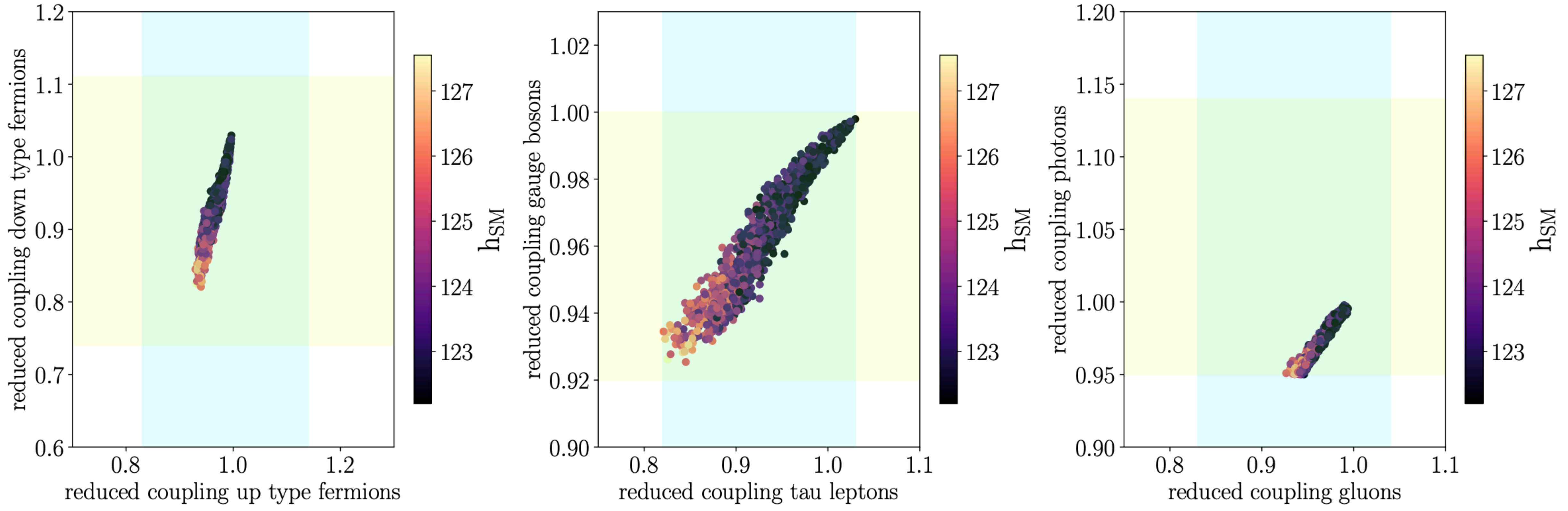


All scanned points are within the reported excess

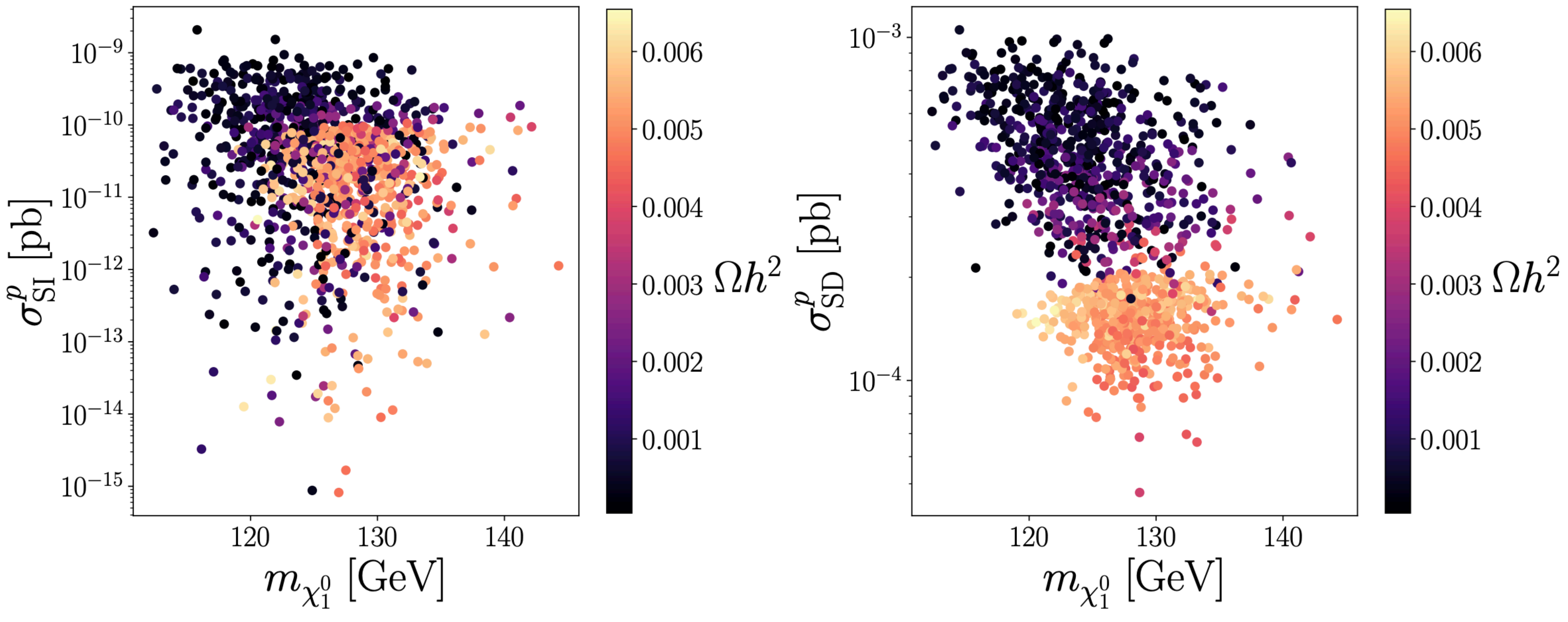


Current bounds

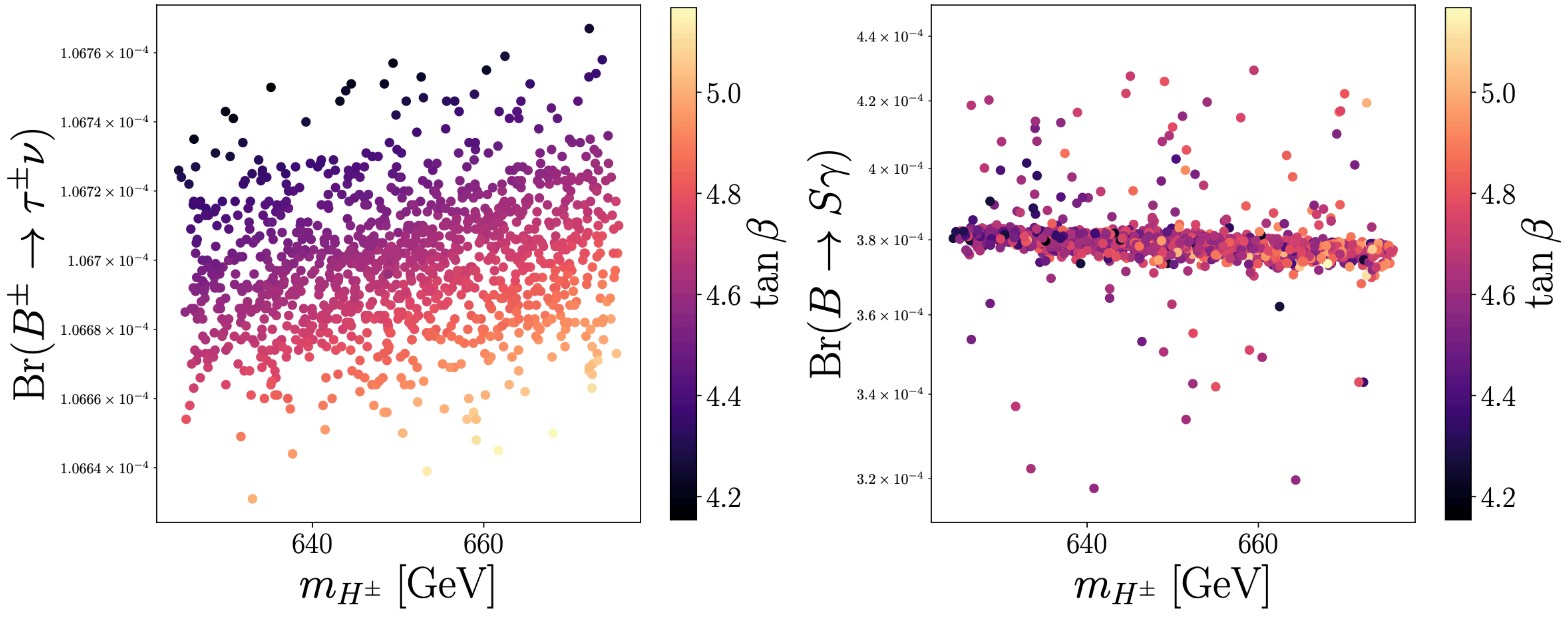
2 sigma within the SM Higgs measurement



DM constraints



B decays



Other constraints are also checked

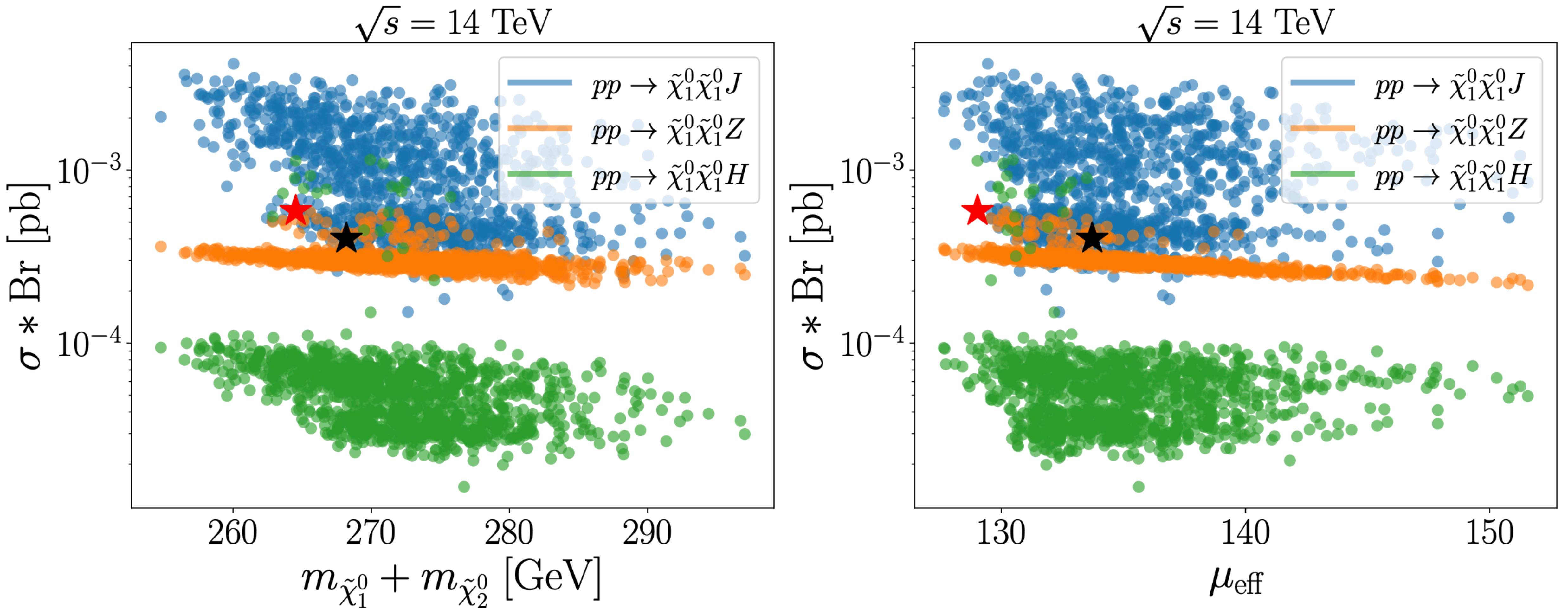
$$S = 0.05 \pm 0.11, \quad T = 0.09 \pm 0.13, \quad \text{and} \quad U = 0.01 \pm 0.11.$$

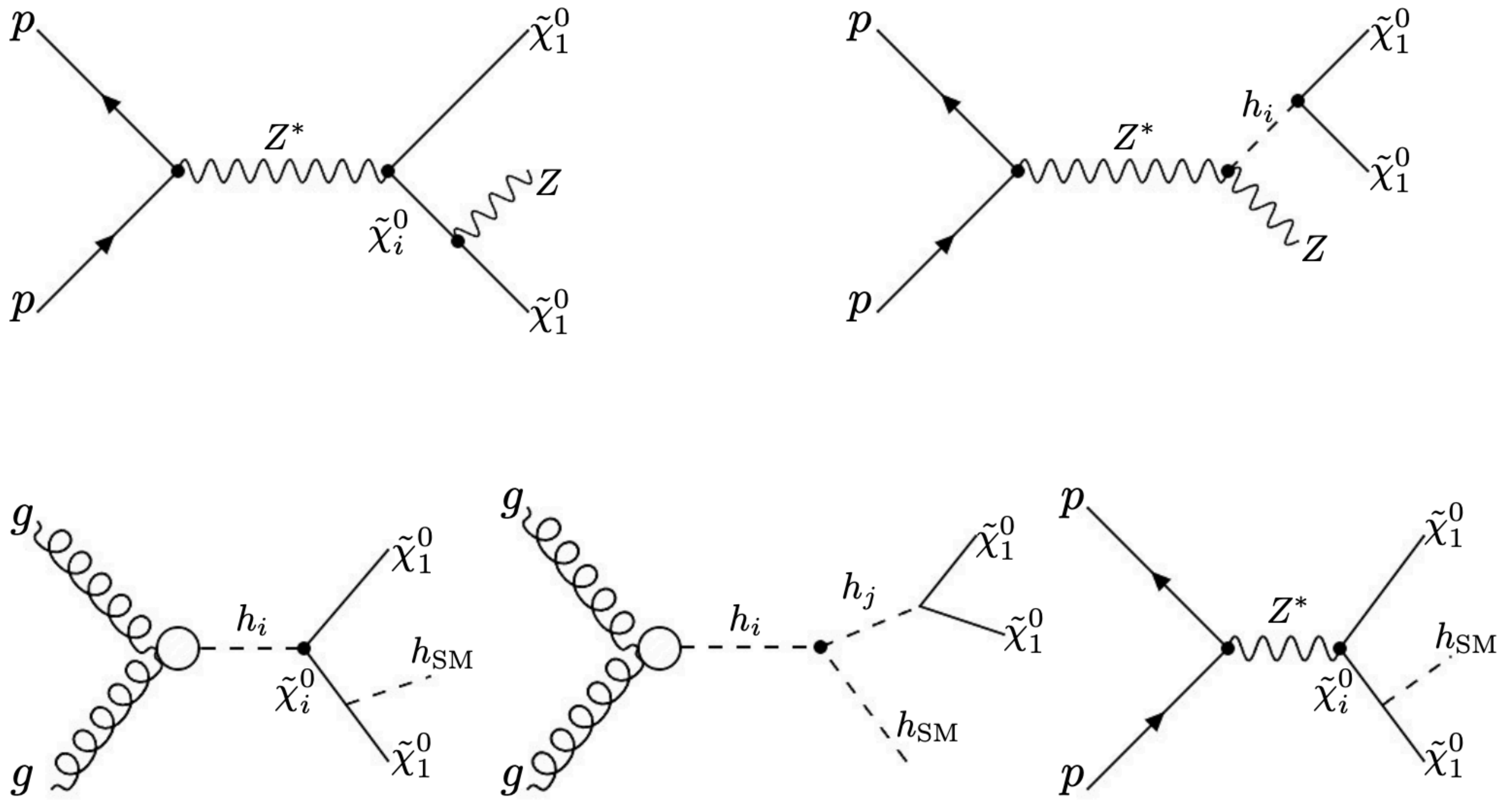
In addition, we checked bounds from the LHC searches and Indirect measurements

Finally we identified the golden region

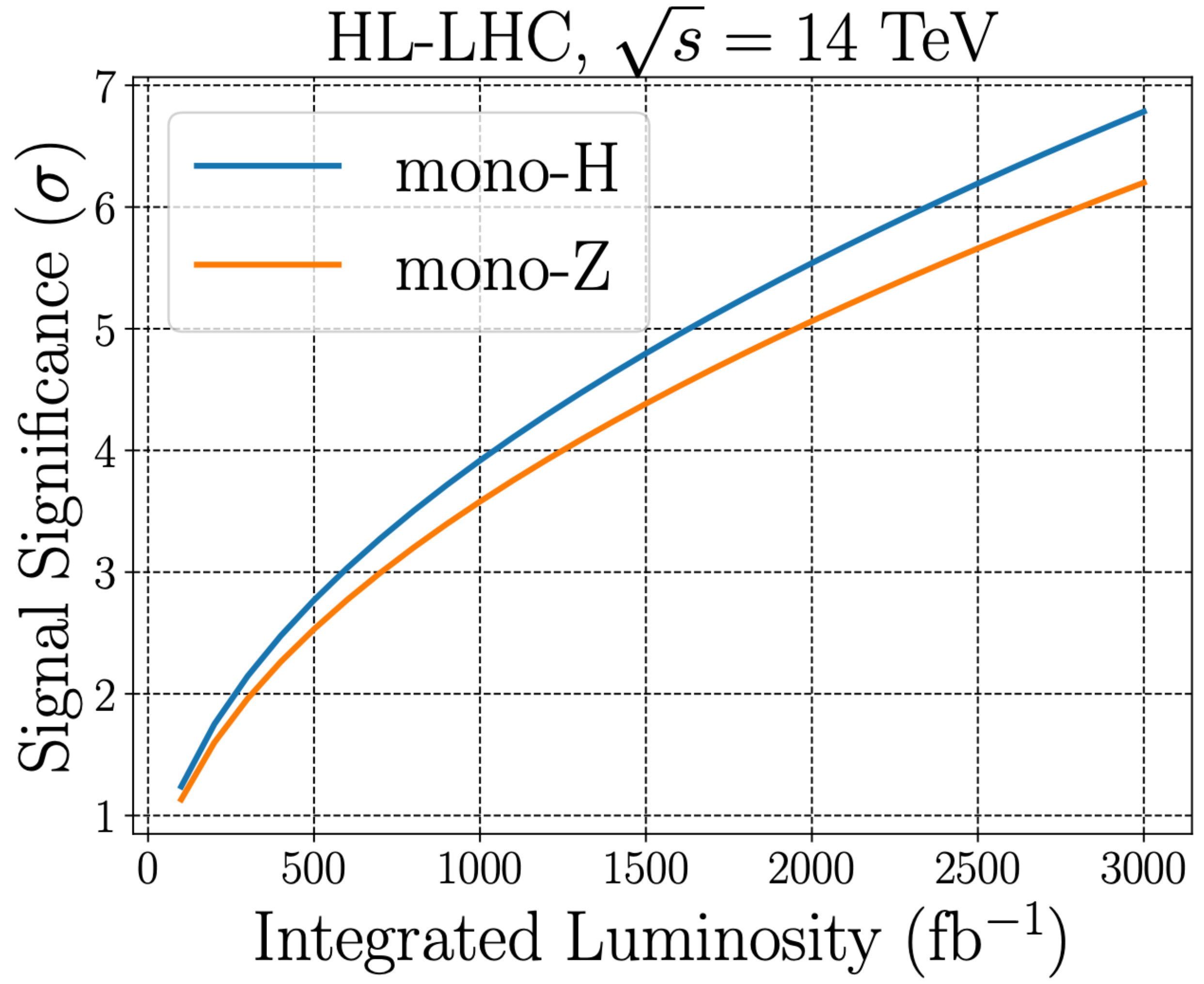
$\tan\beta$	λ	κ	A_λ
[4.15, 4.68]	[0.23,0.36]	[0.16,0.23]	[531.91, 620.72]
A_κ	$\mu_{\rm eff}$	M_1	M_2
[−297.87, −202.34]	[128.11, 151.25]	[559.75, 2988.48]	[768.88, 3971.61]
M_3	A_t	M_{Q_3}	M_{U_3}
[831.99, 4730.45]	[−4999.07, −3882.34]	[957.64, 4385.60]	[944.90, 4667.27]

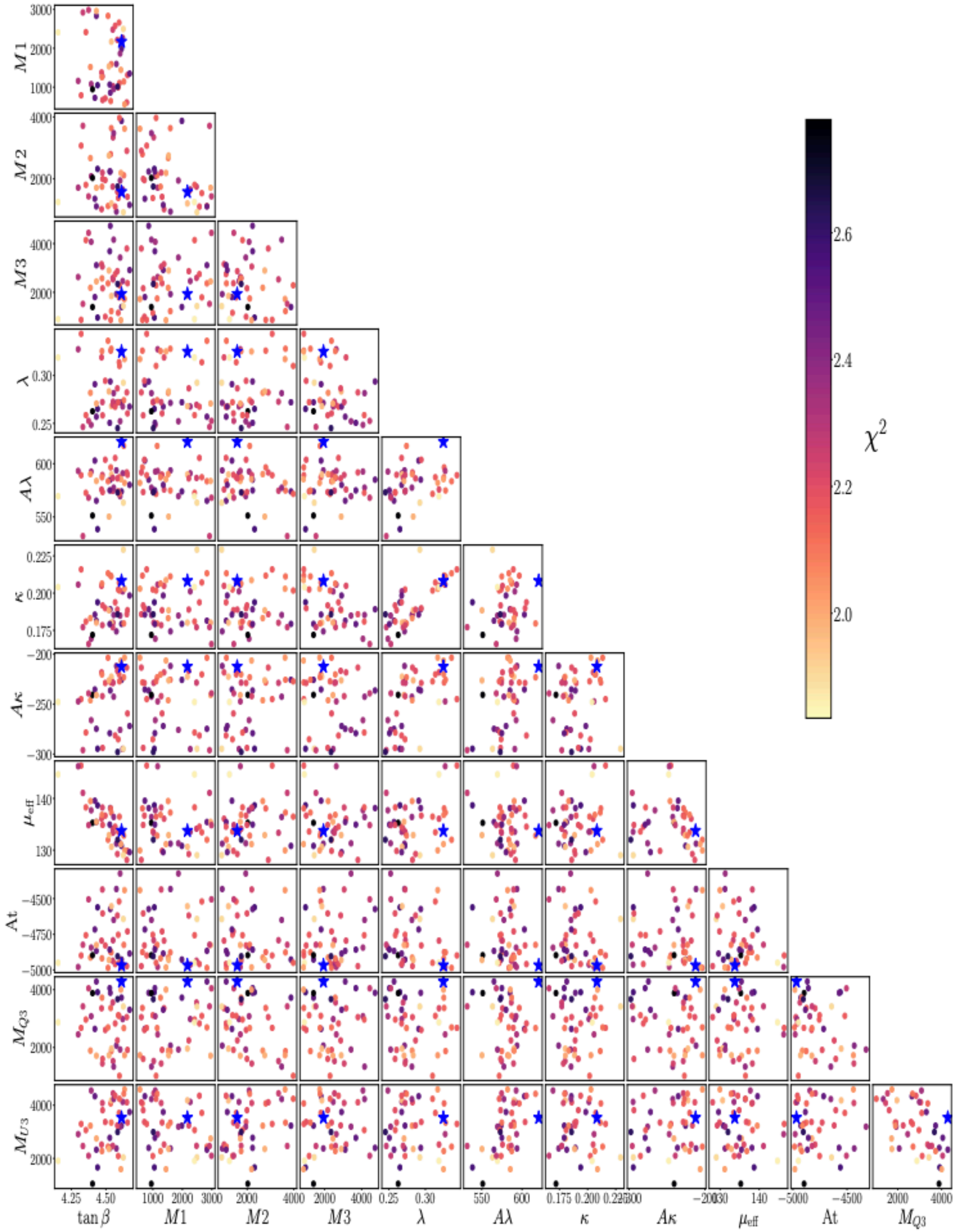
Inside this region we compute the prospective sensitivity for mono-Z and mono-H analyses





Detailed information about the analyses can be found in the paper

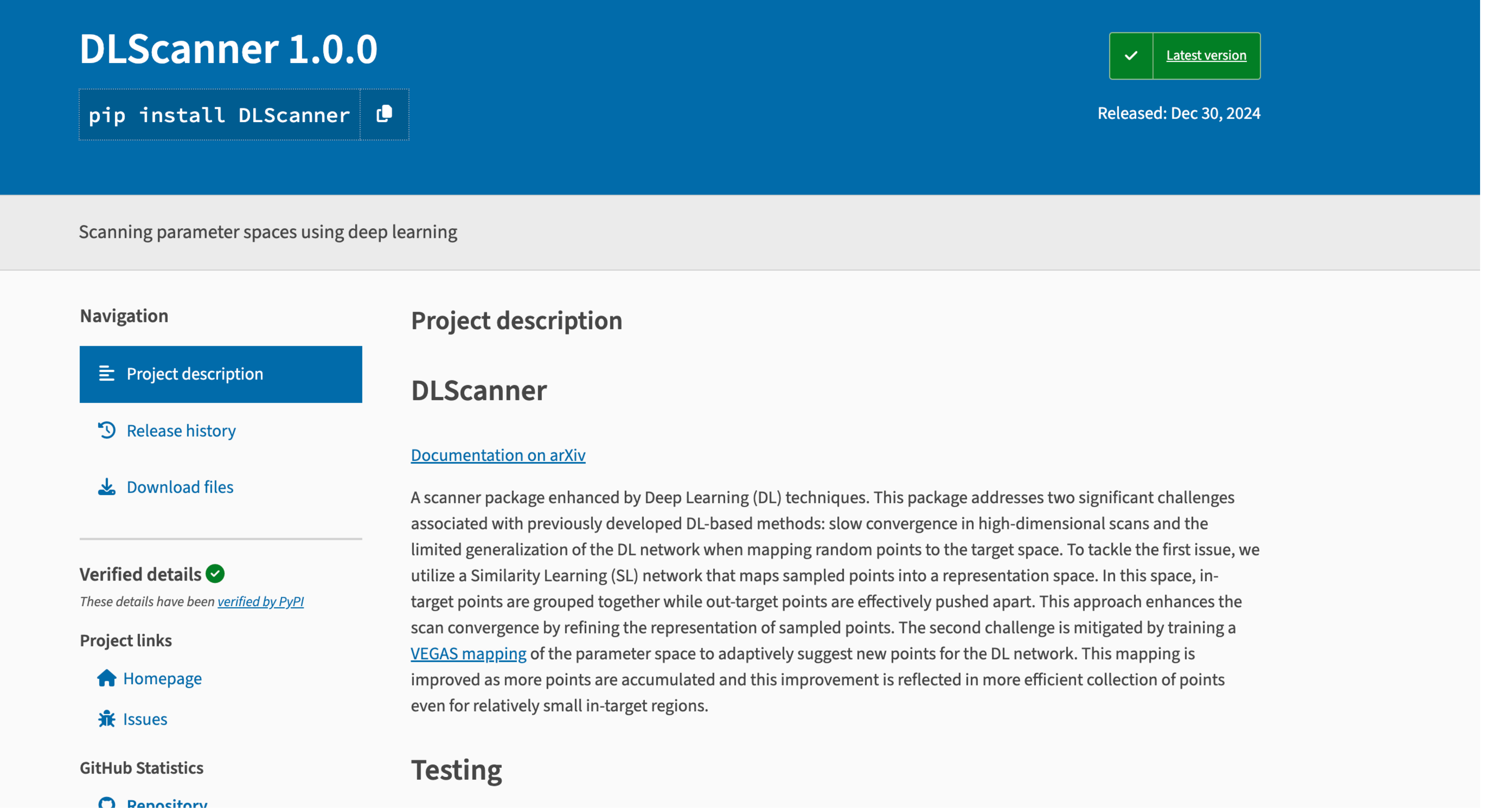




Chi squared is computed to identify the best fit point in the scanned parameter space

$\tan\beta$	λ	κ	A_λ	A_κ	$\mu_{\rm eff}$
4.61	0.32	0.21	620.72	−212.41	133.75
M_1	M_2	M_3	A_t	M_{Q_3}	M_{U_3}
2169.49	1574.89	1932.62	−4967.71	4280.64	3529.45

Best fit point with chi squared = 1.85



DLScanner is a friendly user easy to install package:
pip install DLScanner

It is a generic sampling method that can be used for a wide range of problems

Create your own ML scanning tool

```python
import numpy as np
from subprocess import Popen, PIPE
# It is assumed that the user knows how to parse the output of the program
# and, for the classifier, that has decided on a condition for points that
# are in- and out-target
from user_module import parse_my_output, write_parameters, user_condition

my_program = "./my_executable"
# If parameters are read from file
parameters_file = "./my_parameters"

# Function to run program and parse content.  Parameters read from command
    line arguments
def run_my_program_1(pvector):
    par1, par2, par3 = pvector
    process = Popen([my_program, par1, par2, par3], stdout=PIPE, stderr=PIPE
        )
    output, error = process.communicate()
    return parse_my_output(output)  # Parsing returns only numerical value

# Function to run program and parse content.  Parameters read from file
def run_my_program_2(pvector):
    write_parameters(parameters_file, pvector)
    process = Popen([my_program, parameters_file], stdout=PIPE, stderr=PIPE)
    output, error = process.communicate()
    return parse_my_output(output)  # Parsing returns only numerical value

# Function to take an array of parameter vectors and output array of output
def run_array(array):
    result = np.empty(len(array))
    for j in range(len(array)):
        result[j] = run_my_program(pvector[j])
    return result  # Output an array of calculation results

# For the classifier: function that separates into classes: in- and out-
    target
def true_class(array):
    result = run_array(array)
    labels = user_condition(result)
    return labels  # Array of 0 and 1 values
```

Thanks