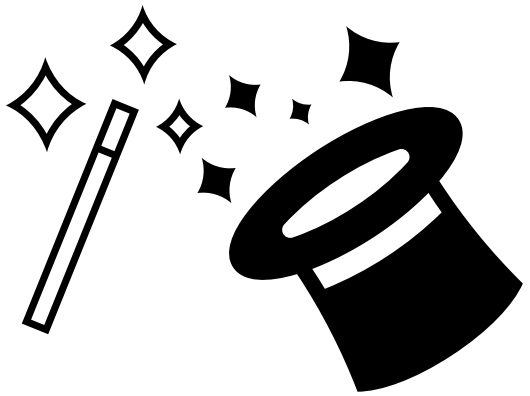


Parameter inference for particle physics *using Magic!*

Gert Kluge
PhD candidate at the University of Oslo

CERN School of Computing 2022 – Krakow
7th of September 2022

Particle physics in a nutshell: the search for $p(\boldsymbol{\vartheta}|\boldsymbol{x})$

Bayes theorem:

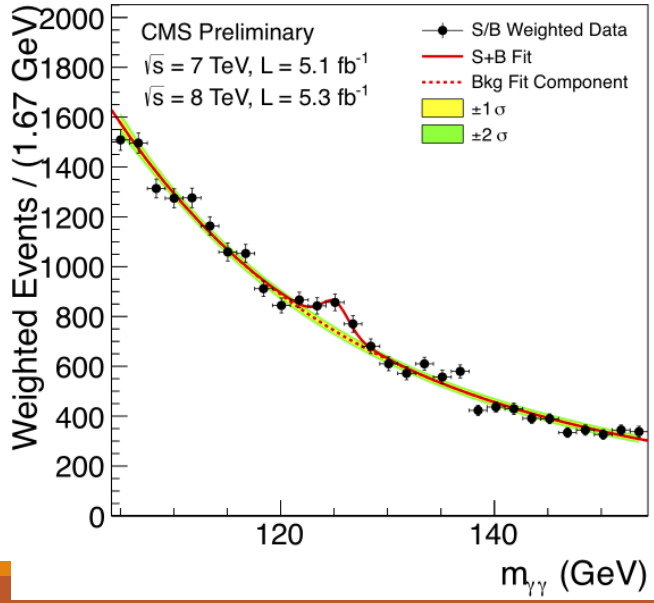
$$p(\boldsymbol{\vartheta}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{\vartheta})}{p(\boldsymbol{x})} p(\boldsymbol{\vartheta})$$

Particle physics in a nutshell: the search for $p(\boldsymbol{\vartheta}|\boldsymbol{x})$

Bayes theorem:

$$p(\boldsymbol{\vartheta}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{\vartheta})}{p(\boldsymbol{x})} p(\boldsymbol{\vartheta})$$

$\boldsymbol{x} =$



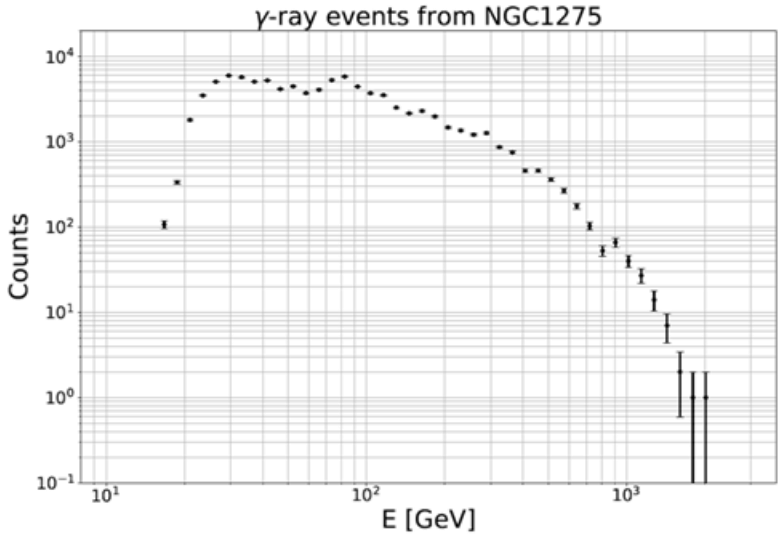
$$\boldsymbol{\vartheta} = \begin{matrix} \boldsymbol{m} \\ \boldsymbol{\sigma} \end{matrix}$$

Particle physics in a nutshell: the search for $p(\boldsymbol{\vartheta}|\boldsymbol{x})$

Bayes theorem:

$$p(\boldsymbol{\vartheta}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{\vartheta})}{p(\boldsymbol{x})} p(\boldsymbol{\vartheta})$$

$\boldsymbol{x} =$



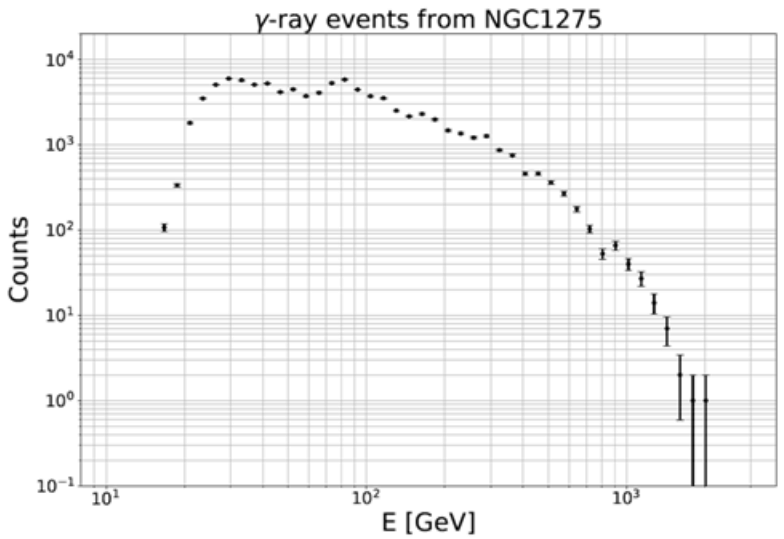
$$\boldsymbol{\vartheta} = \begin{matrix} \boldsymbol{\varphi_0} \\ \boldsymbol{\gamma} \\ \mathbf{E_{cut}} \end{matrix}$$

Particle physics in a nutshell: the search for $p(\boldsymbol{\vartheta}|\boldsymbol{x})$

Bayes theorem:

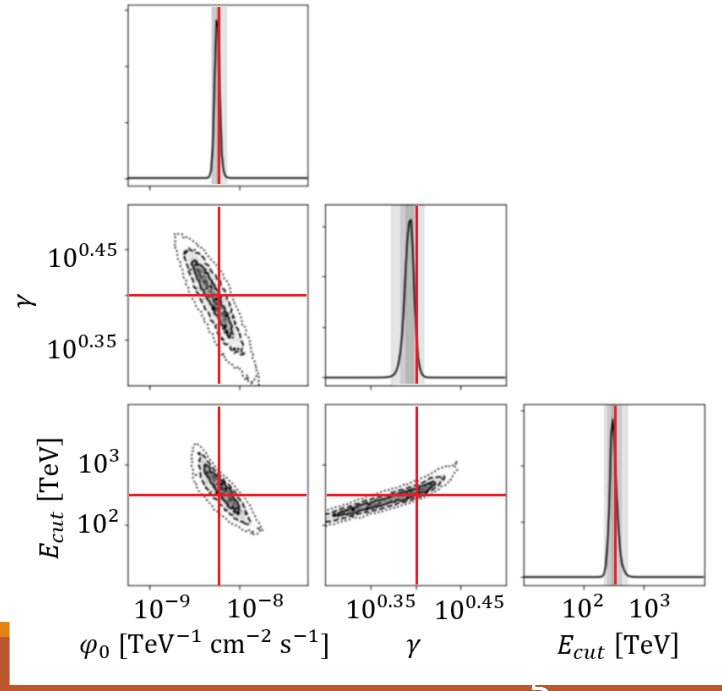
$$p(\boldsymbol{\vartheta}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{\vartheta})}{p(\boldsymbol{x})} p(\boldsymbol{\vartheta})$$

$\boldsymbol{x} =$



$\boldsymbol{\vartheta} =$ $\boldsymbol{\varphi_0}$, $\boldsymbol{\gamma}$, $\mathbf{E_{cut}}$

$\Longrightarrow$



Particle physics in a nutshell: the search for $p(\vartheta|x)$

Bayes theorem:

$$p(\vartheta|x) = \frac{p(x|\vartheta)}{p(x)} p(\vartheta)$$

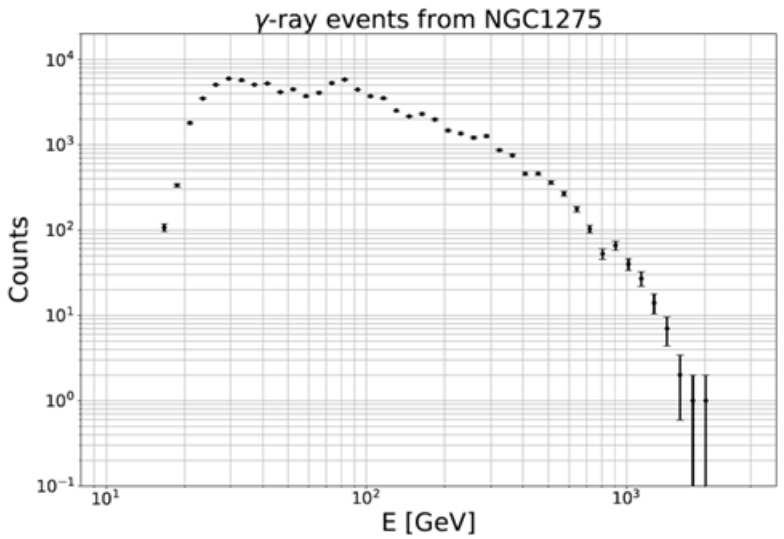
Marginal likelihood
(likelihood integrated over nuisance parameters)

Prior ("a priori" assumption)

ϑ = parameters of interest (m and g)

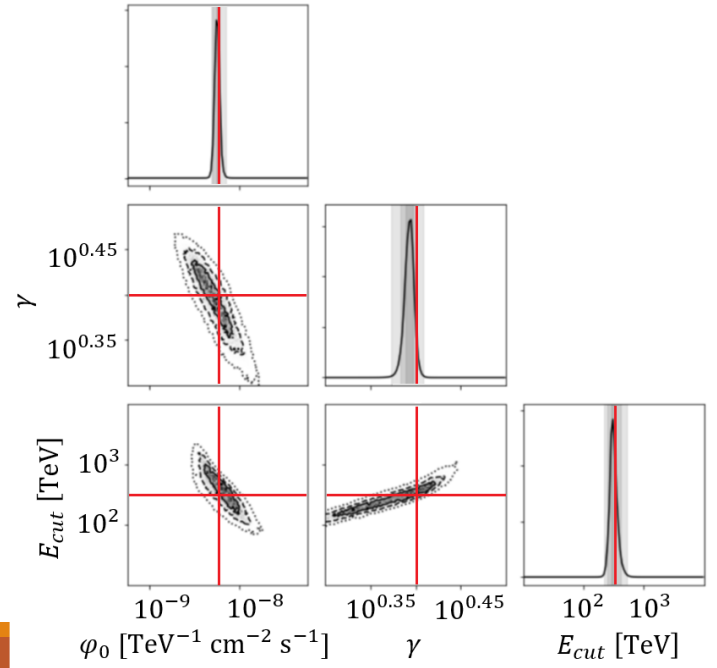
Evidence = $\int \mathrm{d}\boldsymbol{\theta}\, p(x|\boldsymbol{\theta})$

$x =$

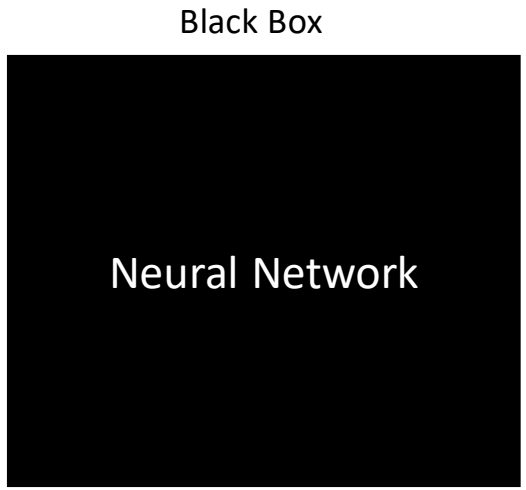


$\vartheta =$ φ_0, γ, E_{cut}

$\Rightarrow$

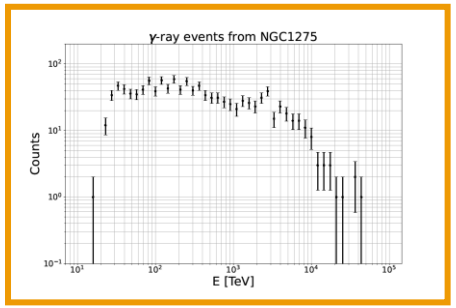


Neural networks can approximate posteriors!



Neural networks can approximate posteriors!

$\boldsymbol{x} =$



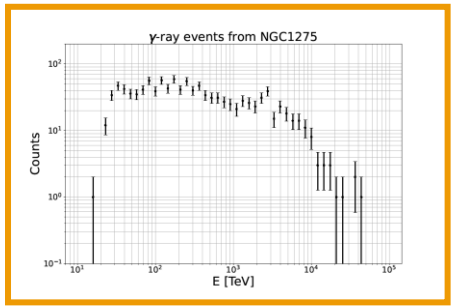
Input

Black Box

Neural Network

Neural networks can approximate posteriors!

$\boldsymbol{x} =$



$\boldsymbol{\vartheta} =$ $\boldsymbol{\varphi_0 = 10}$, $\boldsymbol{\gamma = 1}$, $\boldsymbol{E_{cut} = 3}$

Input

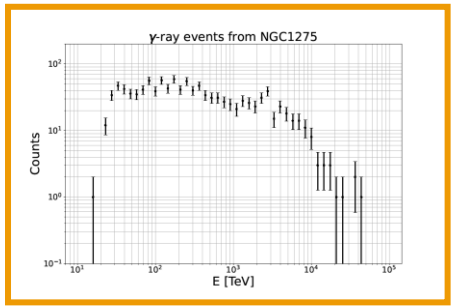
Input

Black Box

Neural Network

Neural networks can approximate posteriors!

$\boldsymbol{x} =$



Input

$\boldsymbol{\vartheta} = \begin{matrix} \boldsymbol{\varphi_0 = 10} \\ \boldsymbol{\gamma = 1} \\ \boldsymbol{E_{cut} = 3} \end{matrix}$

Input

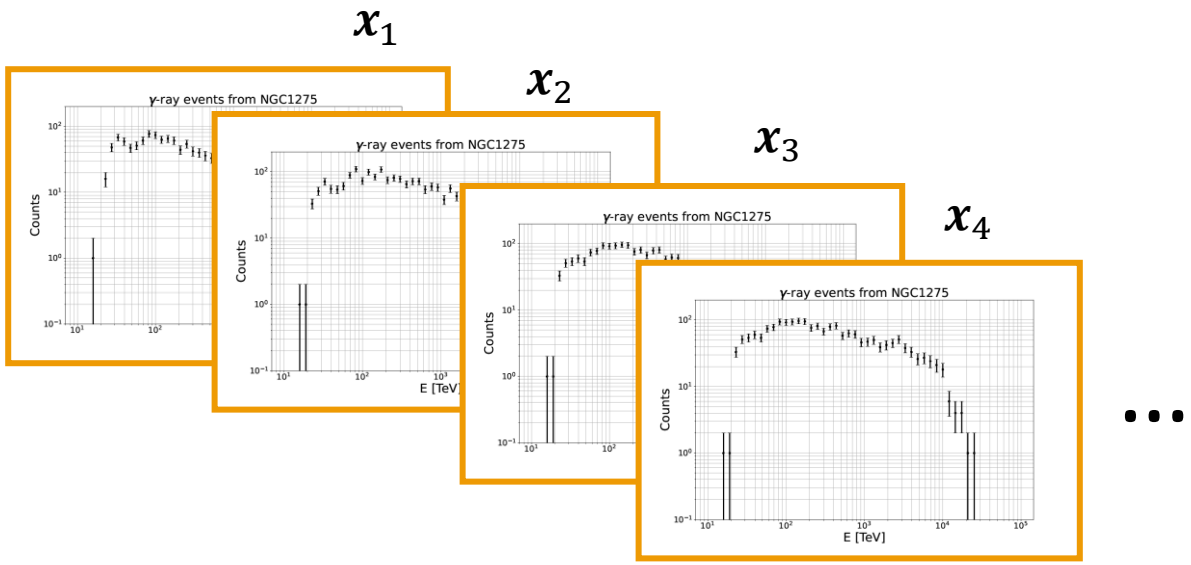
Black Box

Neural Network

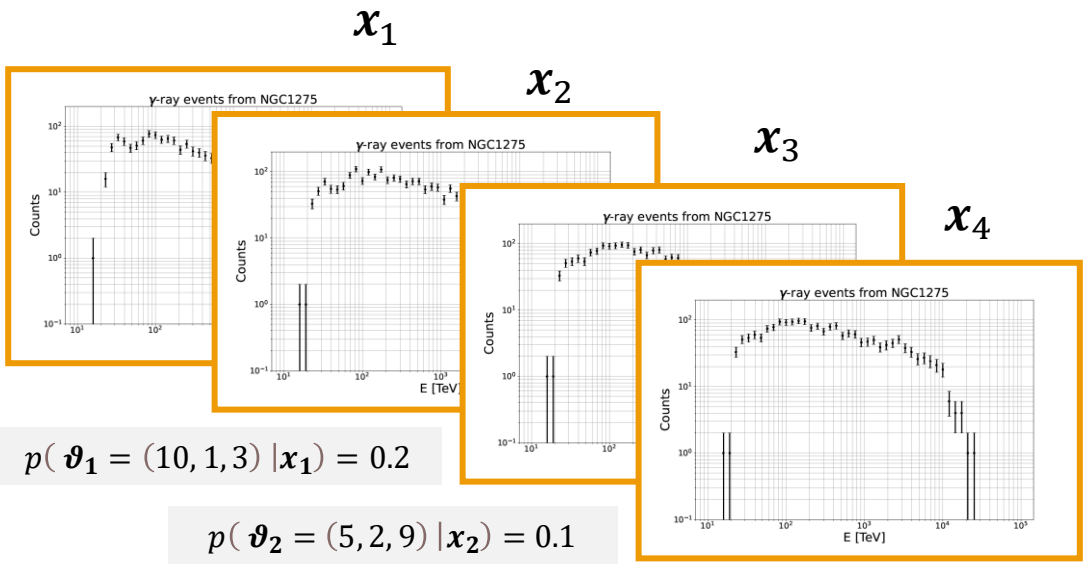
Output

$p(\boldsymbol{\vartheta} = (10,1,3)\,|\boldsymbol{x}) = 0.2$

A network learns to do approximations by example:



A network learns to do approximations by example:



$p(\boldsymbol{\vartheta_1} = (10, 1, 3)\,|\boldsymbol{x_1}) = 0.2$

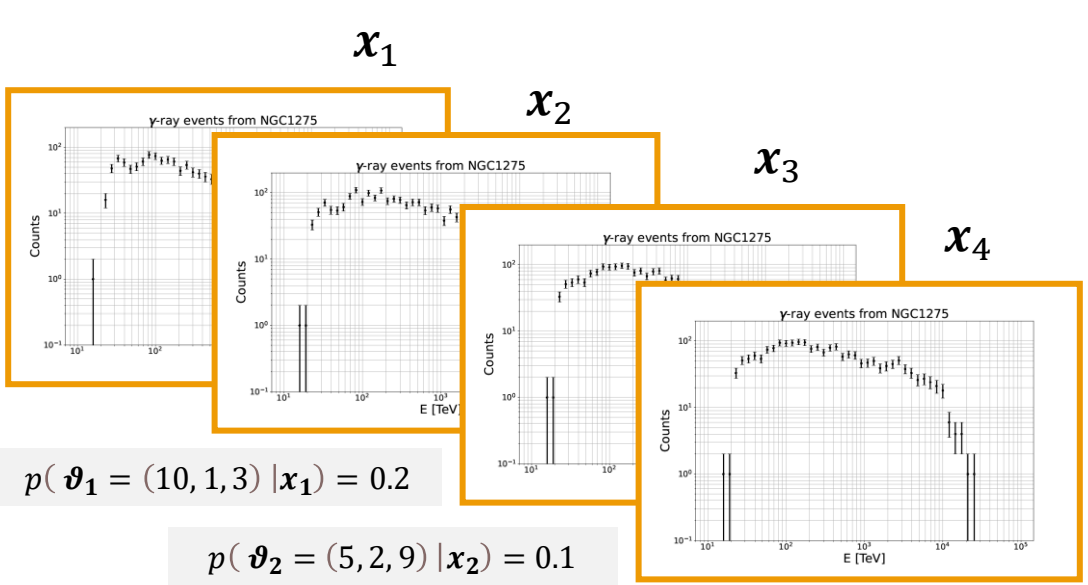
$p(\boldsymbol{\vartheta_2} = (5, 2, 9)\,|\boldsymbol{x_2}) = 0.1$

$p(\boldsymbol{\vartheta_3} = (1, 0.5, 4)\,|\boldsymbol{x_3}) = 0.07$

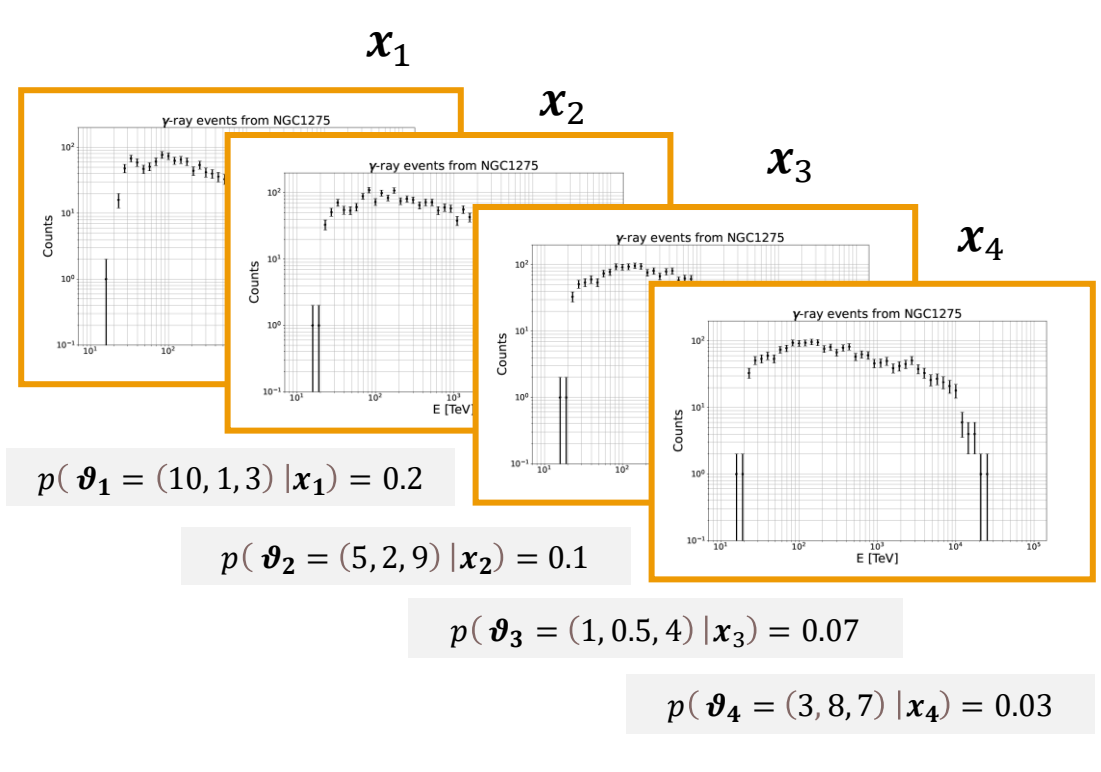
$p(\boldsymbol{\vartheta_4} = (3, 8, 7)\,|\boldsymbol{x_4}) = 0.03$

...

A network learns to do approximations by example:

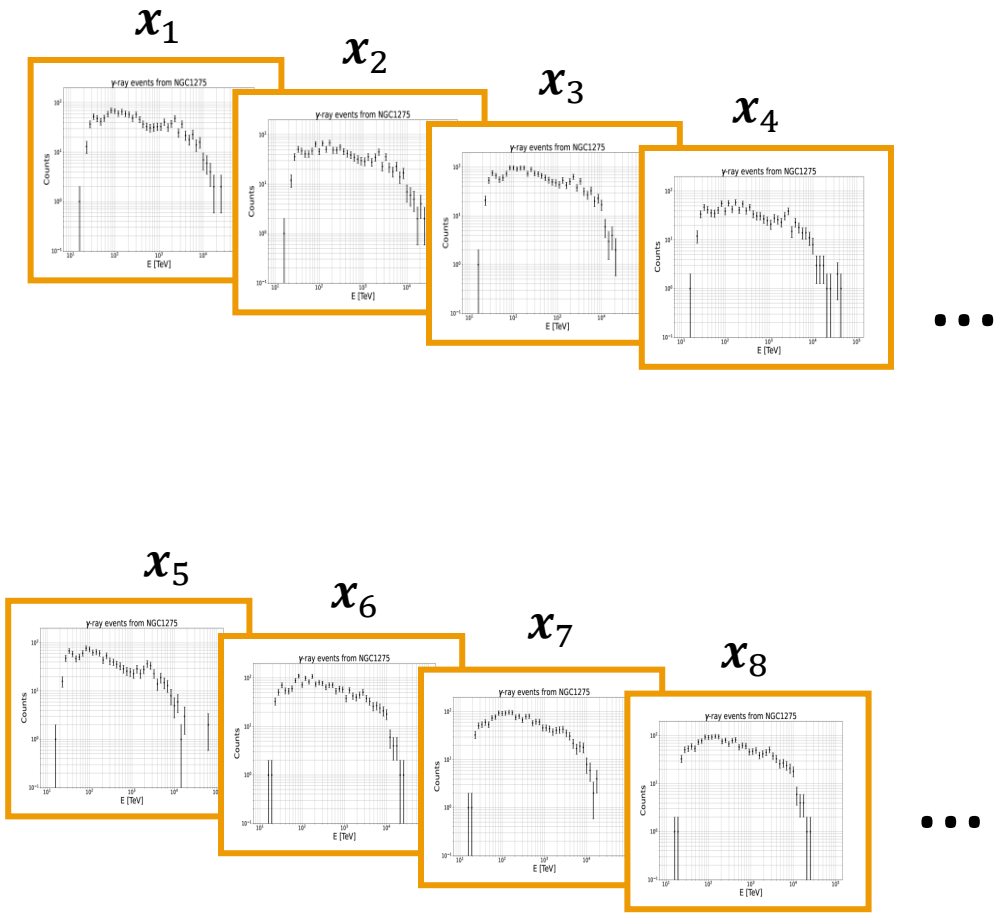


A network learns to do approximations by example:

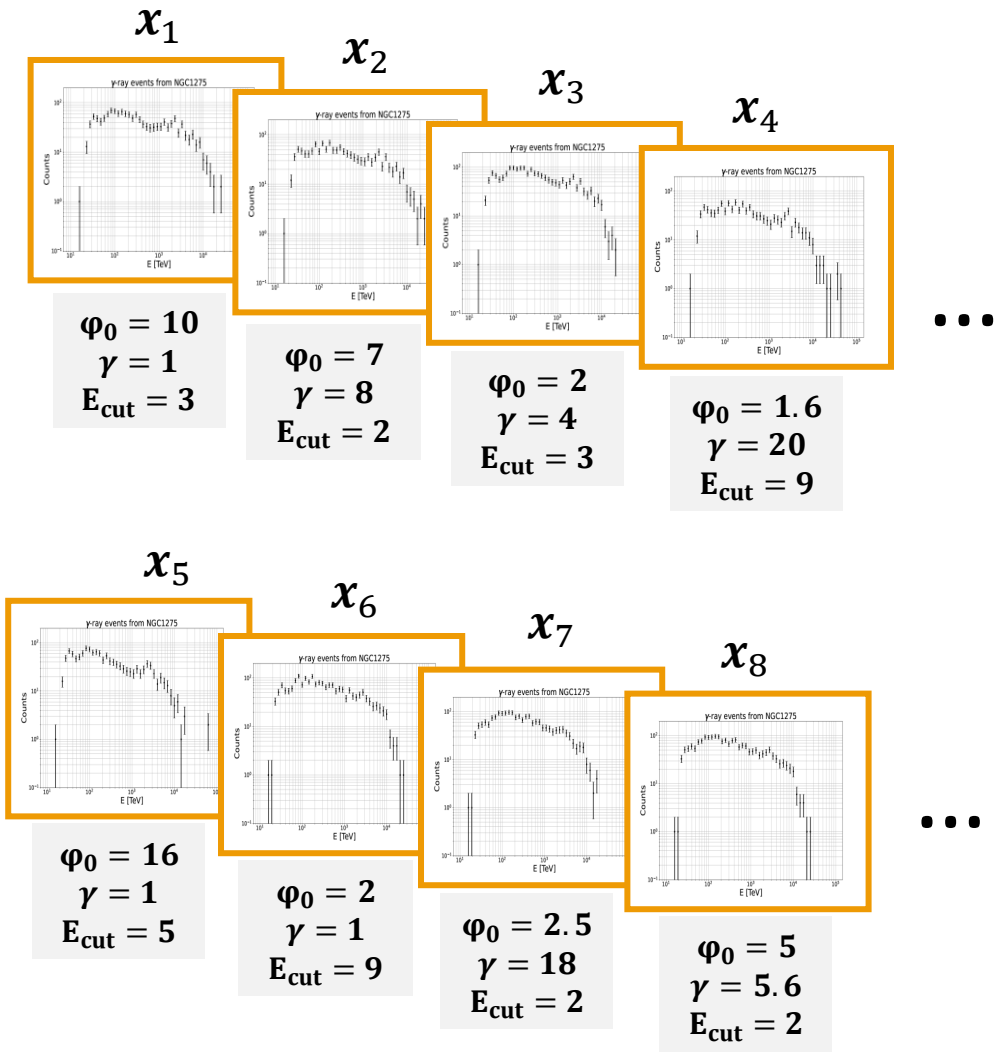


... but this assumes that we can already calculate the posterior!

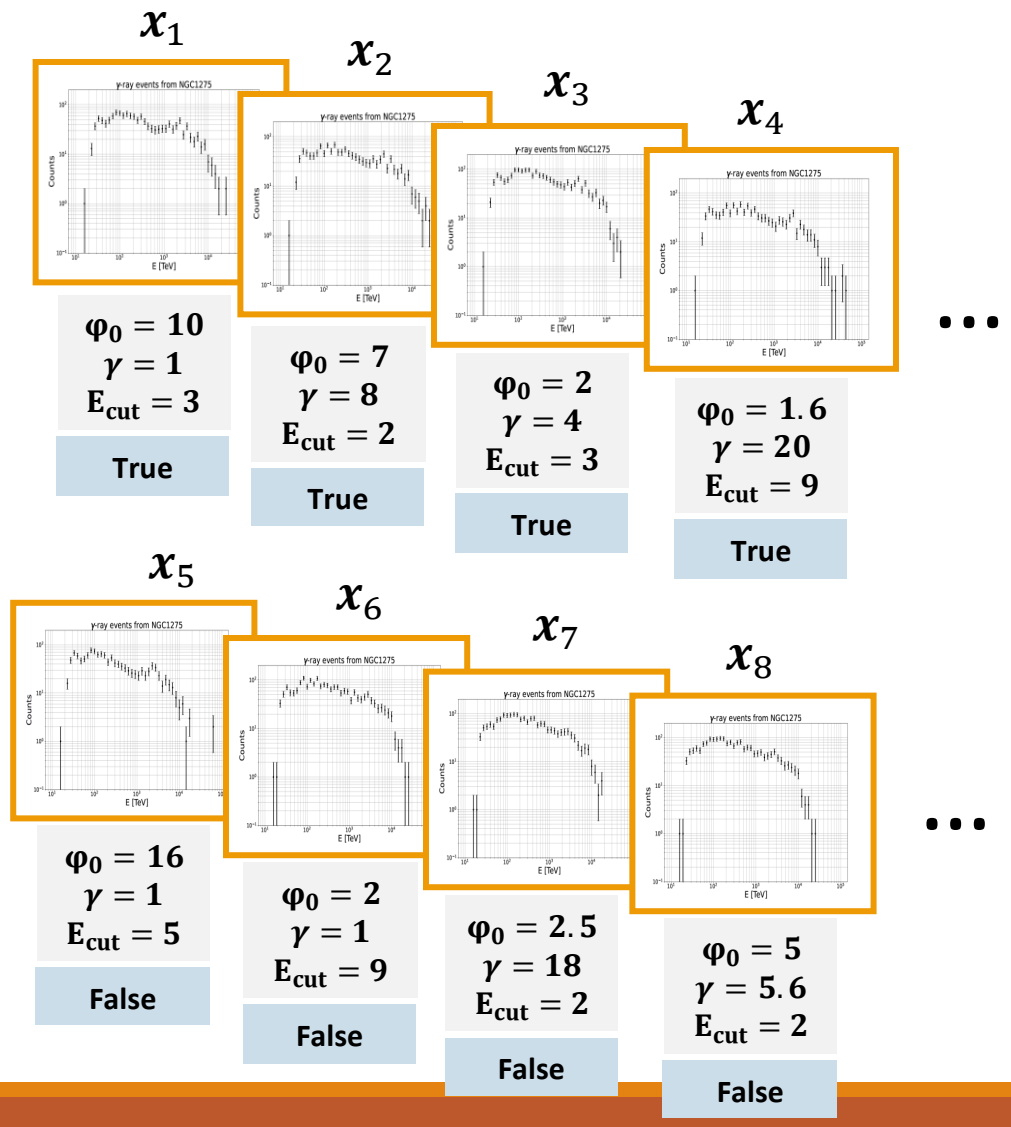
We can “trick” a network to learn the posterior implicitly



We can “trick” a network to learn the posterior implicitly



We can “trick” a network to learn the posterior implicitly



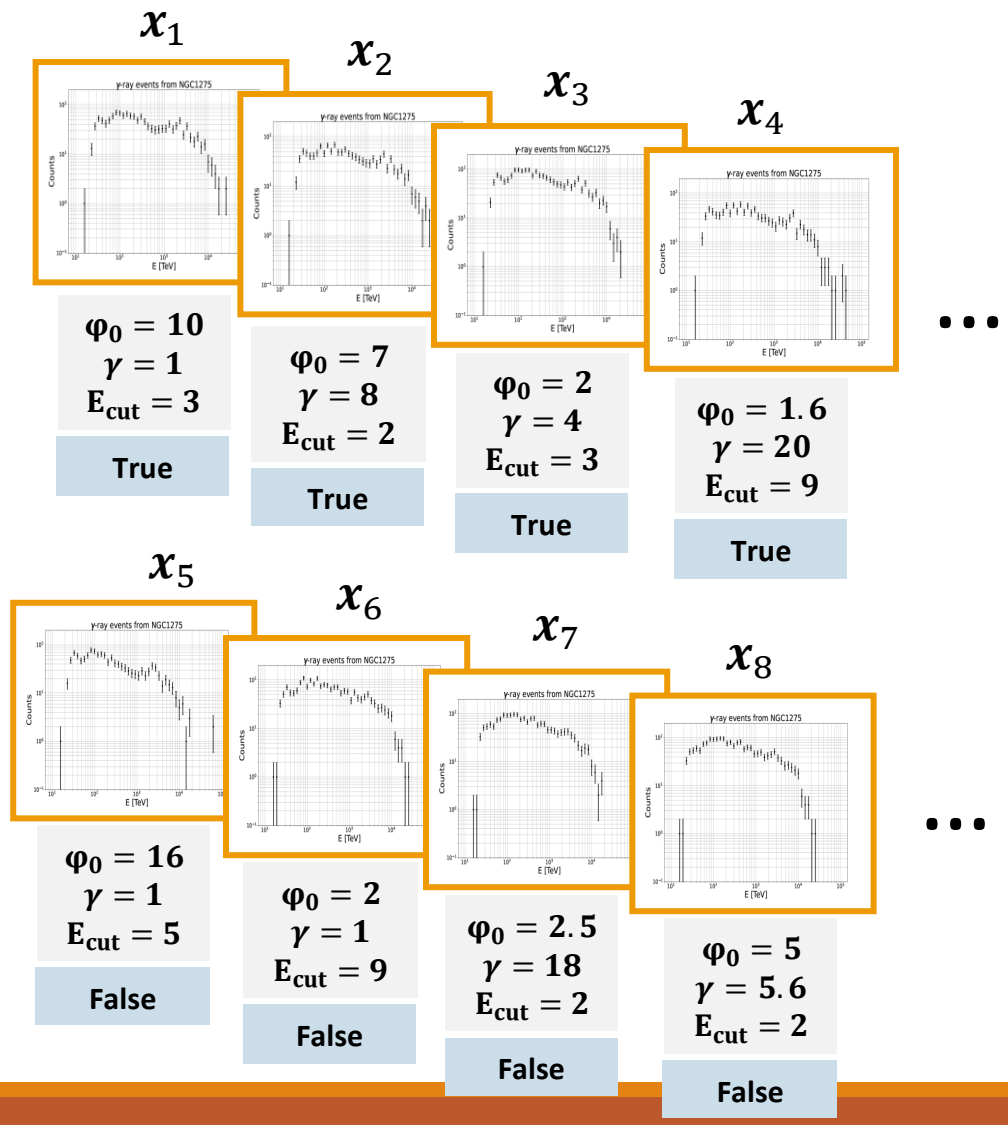
We can “trick” a network to learn the posterior implicitly

True

The observations are simulated according to the parameter values

False

The parameter values are chosen independently from the simulations



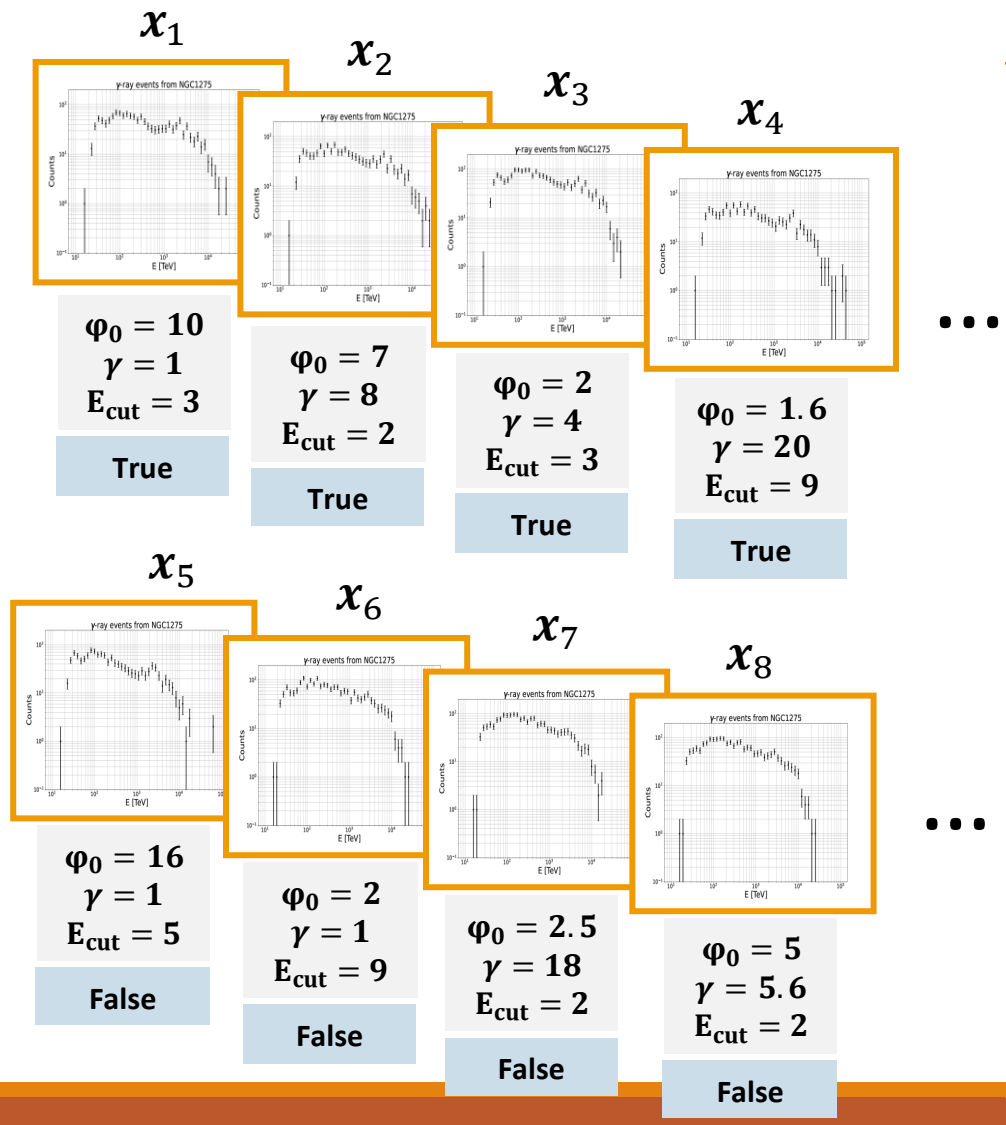
We can “trick” a network to learn the posterior implicitly

True

The observations are simulated according to the parameter values

False

The parameter values are chosen independently from the simulations



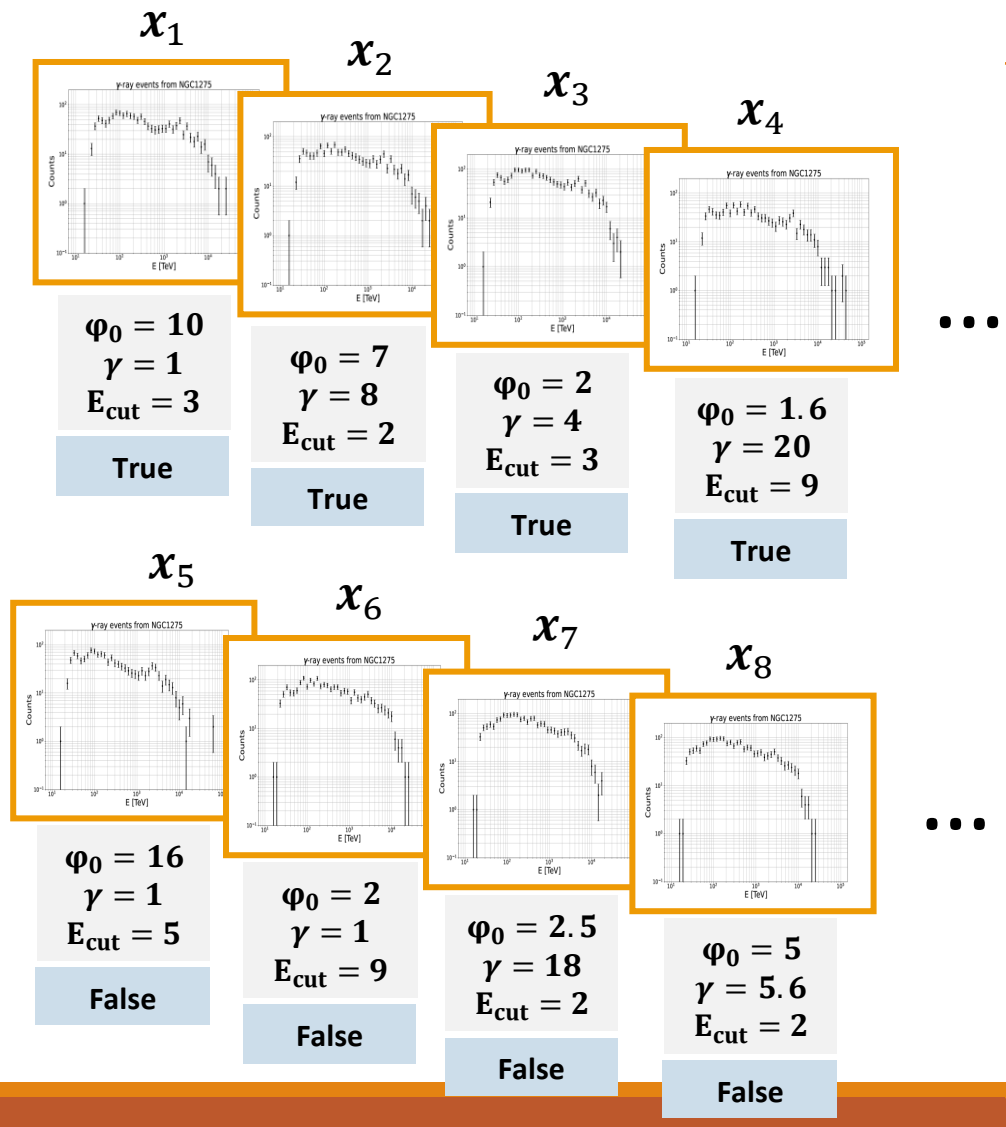
We can “trick” a network to learn the posterior implicitly

True

The observations are simulated according to the parameter values

False

The parameter values are chosen independently from the simulations



Training

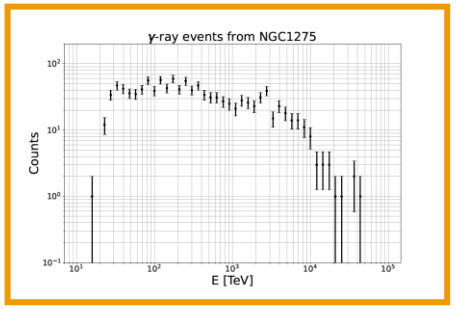
Neural Network

Note: We have to draw all of the parameters of interest from the prior distribution!

Also, the network must use the binary cross-entropy as cost function.

The network now has an implicit understanding of the posterior

$\boldsymbol{x} =$



Input

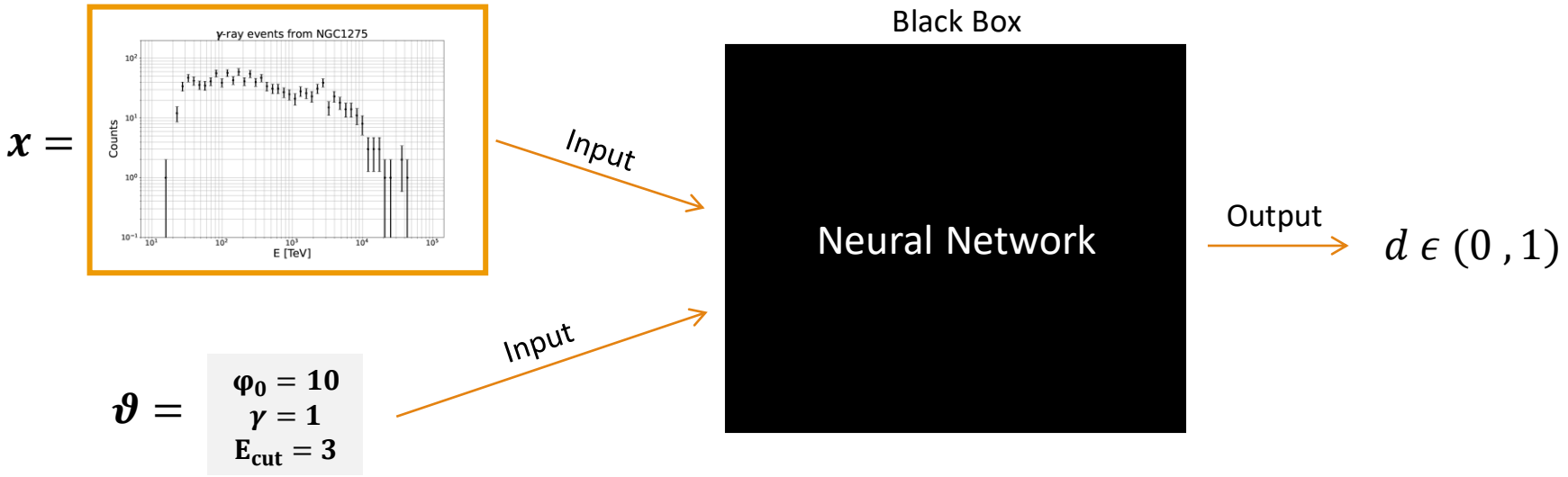
Black Box

Neural Network

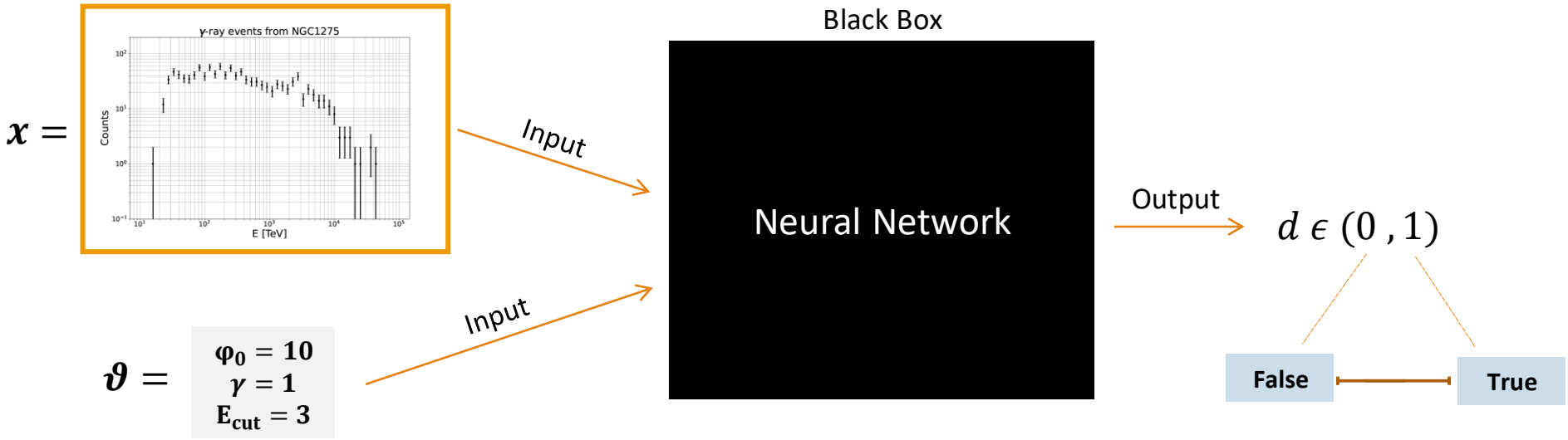
$\boldsymbol{\vartheta} =$ $\boldsymbol{\varphi_0 = 10}$, $\boldsymbol{\gamma = 1}$, $\boldsymbol{E_{cut} = 3}$

Input

The network now has an implicit understanding of the posterior

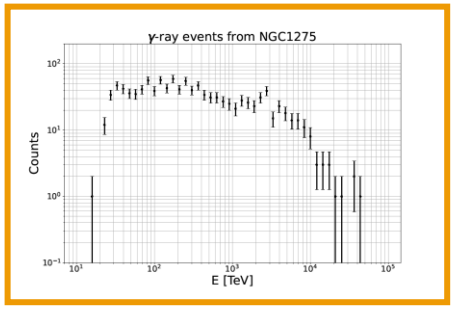


The network now has an implicit understanding of the posterior



The network now has an implicit understanding of the posterior

$\boldsymbol{x} =$



Input

$\boldsymbol{\vartheta} = \begin{matrix} \varphi_0 = 10 \\ \gamma = 1 \\ E_{cut} = 3 \end{matrix}$

Input

Black Box

Neural Network

Output

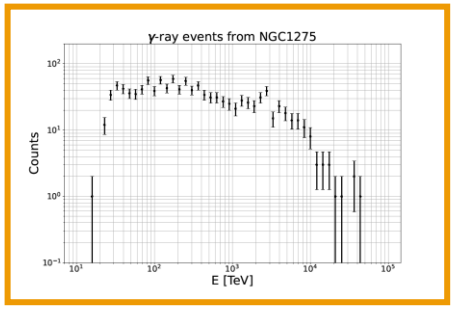
$d \in (0\ , 1)$

False — True

Mathematical trick

$$p(\boldsymbol{\vartheta}|\boldsymbol{x}) = \frac{d}{1-d} p(\boldsymbol{\vartheta})$$

The network now has an implicit understanding of the posterior

$\boldsymbol{x} =$ 

Input

Black Box

Neural Network

Output

$d \in (0\,, 1)$

Mathematical trick

$p(\boldsymbol{\vartheta}|\boldsymbol{x}) = \frac{d}{1-d} p(\boldsymbol{\vartheta})$

False — True

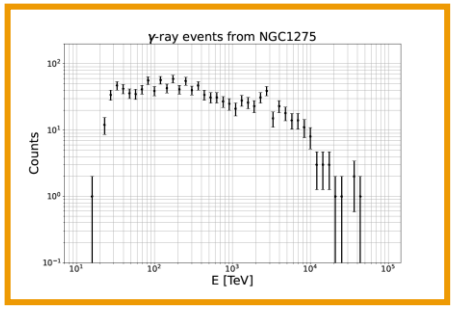
$\boldsymbol{\vartheta} =$ $\varphi_0 = 10$, $\gamma = 1$, $E_{cut} = 3$

Input

Vary this input
to scan the posterior
over parameter space

The network now has an implicit understanding of the posterior

$\boldsymbol{x} =$



Input

Black Box

Neural Network

Output

$d \in (0\,,1)$

Mathematical trick

$p(\boldsymbol{\vartheta}|\boldsymbol{x}) = \frac{d}{1-d} p(\boldsymbol{\vartheta})$

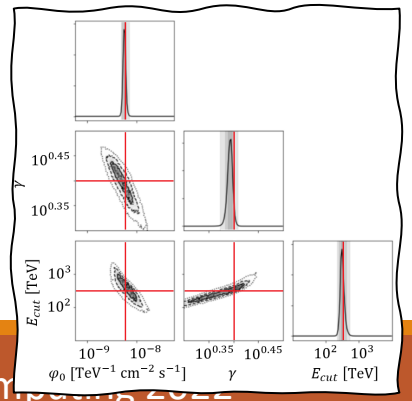
False — True

$\boldsymbol{\vartheta} =$ $\boldsymbol{\varphi_0 = 10}$, $\boldsymbol{\gamma = 1}$, $\boldsymbol{E_{cut} = 3}$

Input

Vary this input
to scan the posterior
over parameter space

$\Rightarrow$

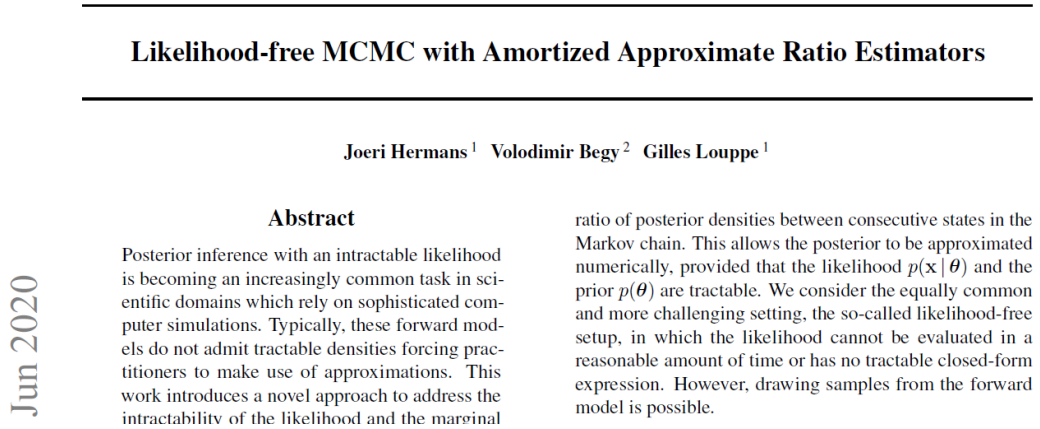


Neural Ratio Estimation: some resources to get you started

- Original paper (to my knowledge) to introduce the concept:
  https://arxiv.org/abs/1903.04057

- B. K. Miller, A. Cole, P. Forŕe, G. Louppe, and C. Weniger, "Truncated marginal neural ratio estimation". https://arxiv.org/abs/2107.01214

- B. K. Miller, A. Cole, G. Louppe, and C. Weniger, "Simulation-efficient marginal posterior estimation with swyft: stop wasting your precious time," https://arxiv.org/abs/2011.13951



Jun 2020

Likelihood-free MCMC with Amortized Approximate Ratio Estimators

Joeri Hermans [1] Volodimir Begy [2] Gilles Louppe [1]

Abstract

Posterior inference with an intractable likelihood is becoming an increasingly common task in scientific domains which rely on sophisticated computer simulations. Typically, these forward models do not admit tractable densities forcing practitioners to make use of approximations. This work introduces a novel approach to address the intractability of the likelihood and the marginal ratio of posterior densities between consecutive states in the Markov chain. This allows the posterior to be approximated numerically, provided that the likelihood $p(\mathbf{x}|\boldsymbol{\theta})$ and the prior $p(\boldsymbol{\theta})$ are tractable. We consider the equally common and more challenging setting, the so-called likelihood-free setup, in which the likelihood cannot be evaluated in a reasonable amount of time or has no tractable closed-form expression. However, drawing samples from the forward model is possible.

- SWYFT (a package under development that does NRE and more): https://github.com/undark-lab/swyft

One take-away:

- You may be able to do Bayesian parameter inference, without having to do impossible integrals, and without putting unrealistic constraints on your nuisance parameters.

One take-away:

- You may be able to do Bayesian parameter inference, without having to do impossible integrals, and without putting unrealistic constraints on your nuisance parameters.

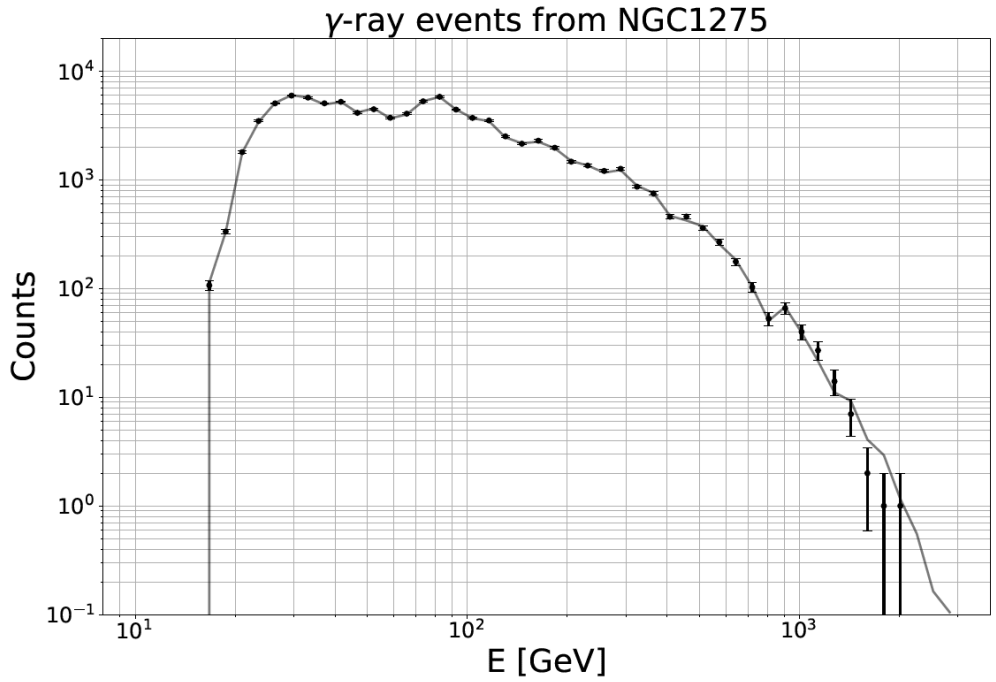
- Basic requirements:

  1. It must be possible to efficiently simulate the experimental observations as a function of physical parameters (of interest and nuisance)

  2. It must be possible for a neural network to adequately distinguish between “matching” and “non-matching” pairs of observations and input parameters.

Backup

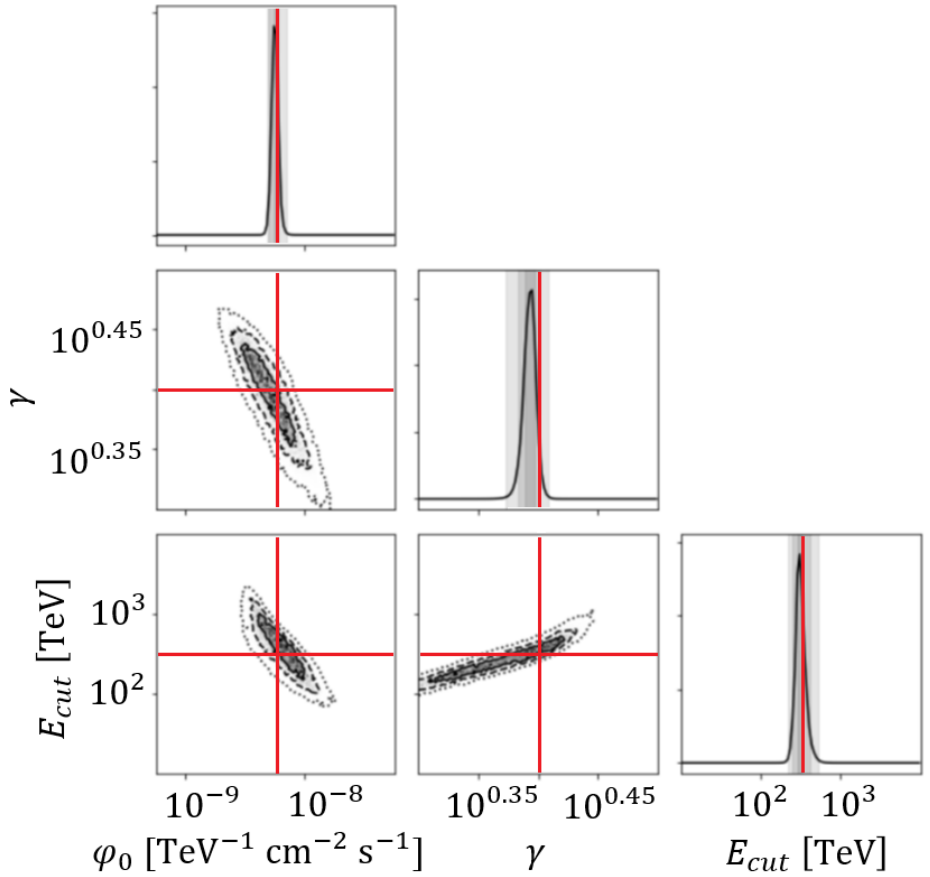
Particle physics in a nutshell: find the values of some free parameters



$$\varphi(E) = \varphi_0 \left(\frac{E}{E_0}\right)^{\gamma} e^{-E/E_{cut}} \quad \text{+ poisson noise}$$

- Parameters of interest:
  - Amplitude φ_0
  - Spectral index γ
  - E_{cut}

Goal: Find the “probability-distributions” for the true parameter values



NRE is easy to implement without any knowledge of the underlying physics

NRE is easy to implement without any knowledge of the underlying physics

Workflow:

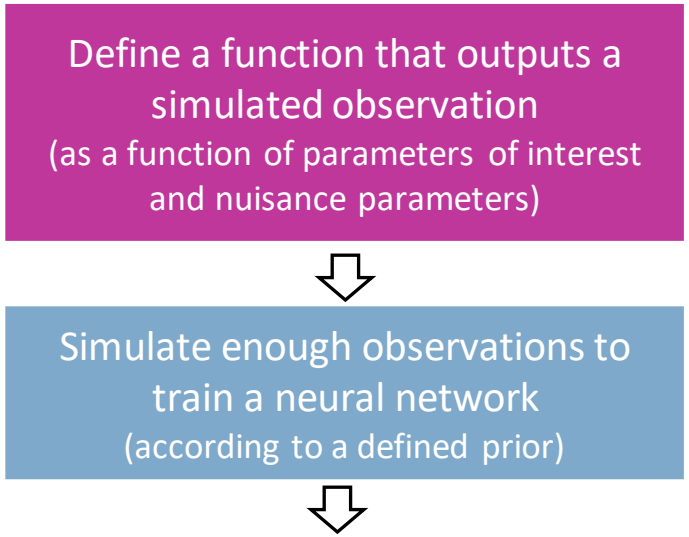
Define a function that outputs a simulated observation
(as a function of parameters of interest and nuisance parameters)

⇩

Using `gammapy`

NRE is easy to implement without any knowledge of the underlying physics

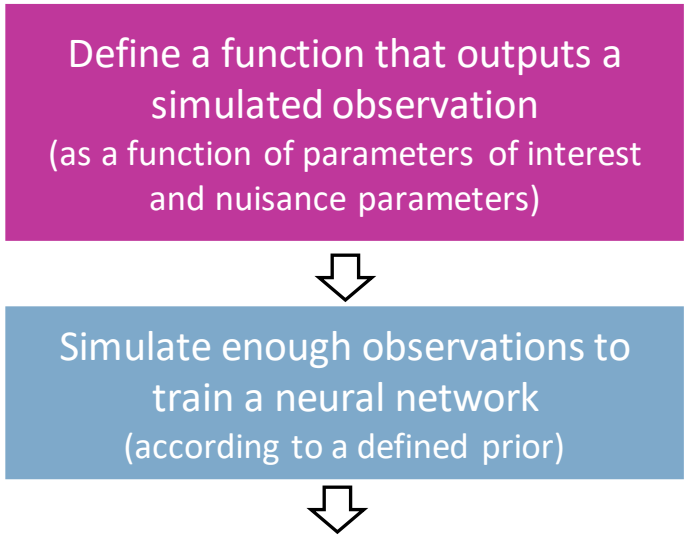
Workflow:



Using `gammapy`

NRE is easy to implement without any knowledge of the underlying physics

Workflow:

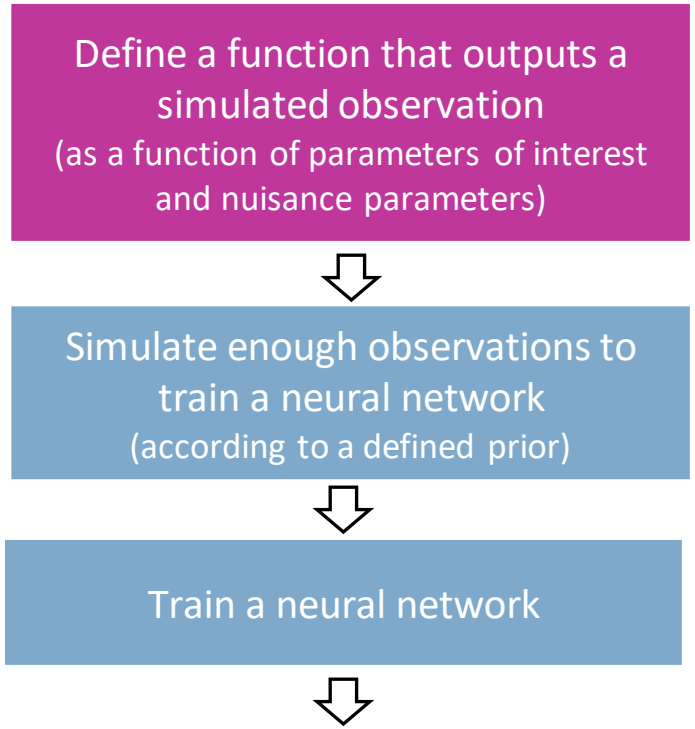


Using `gammapy`

- The simulations implicitly contain the information on the relationship between parameters and observations

NRE is easy to implement without any knowledge of the underlying physics

Workflow:

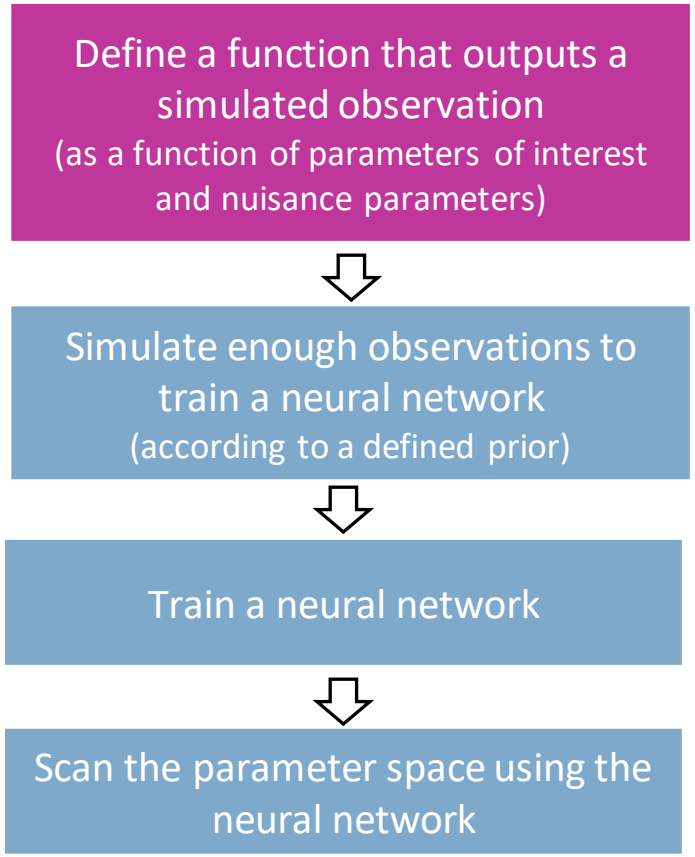


Using `gammapy`

- The simulations implicitly contain the information on the relationship between parameters and observations

NRE is easy to implement without any knowledge of the underlying physics

Workflow:

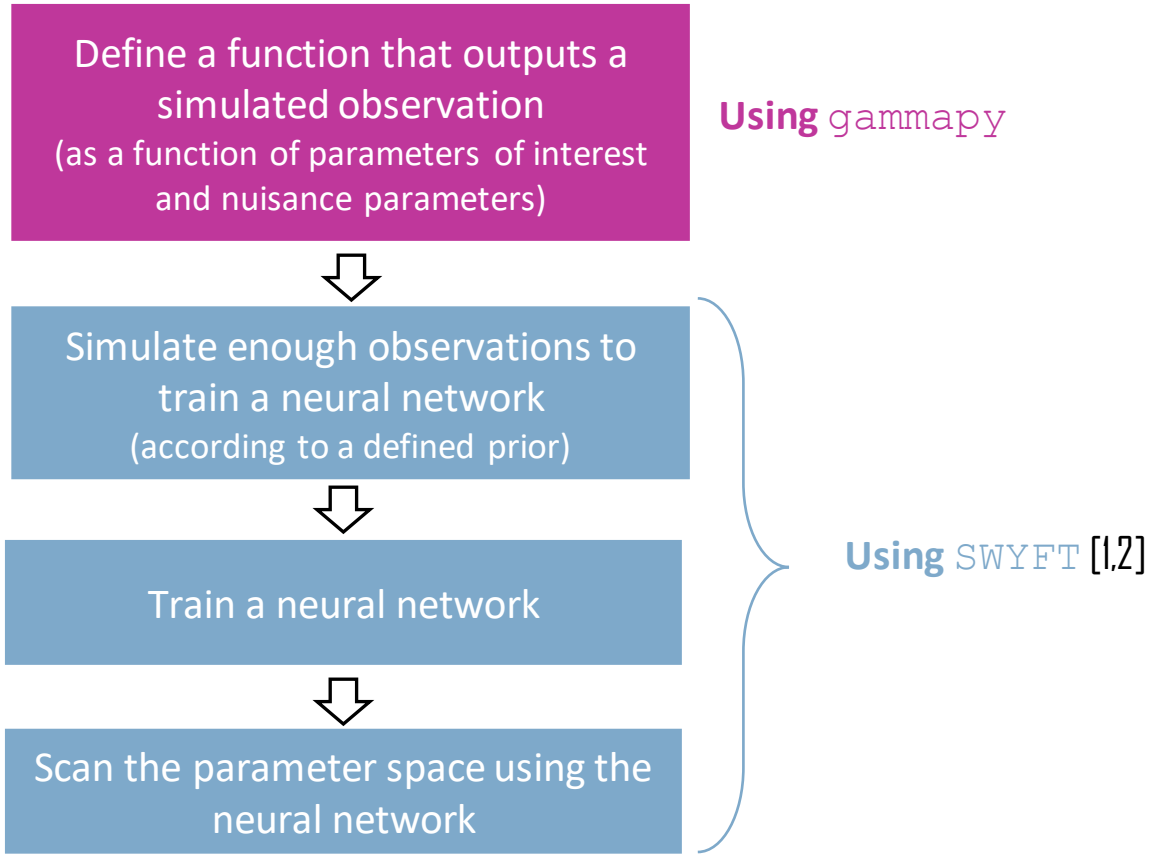


Using `gammapy`

- The simulations implicitly contain the information on the relationship between parameters and observations

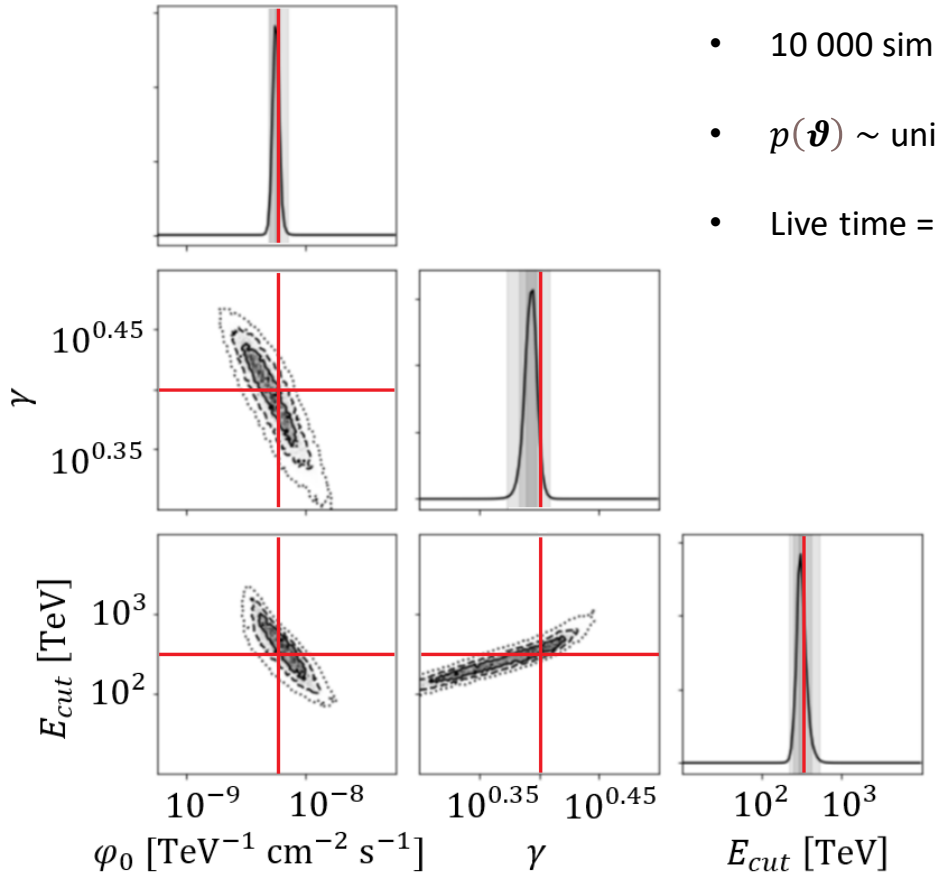
NRE is easy to implement without any knowledge of the underlying physics

Workflow:



- The simulations implicitly contain the information on the relationship between parameters and observations

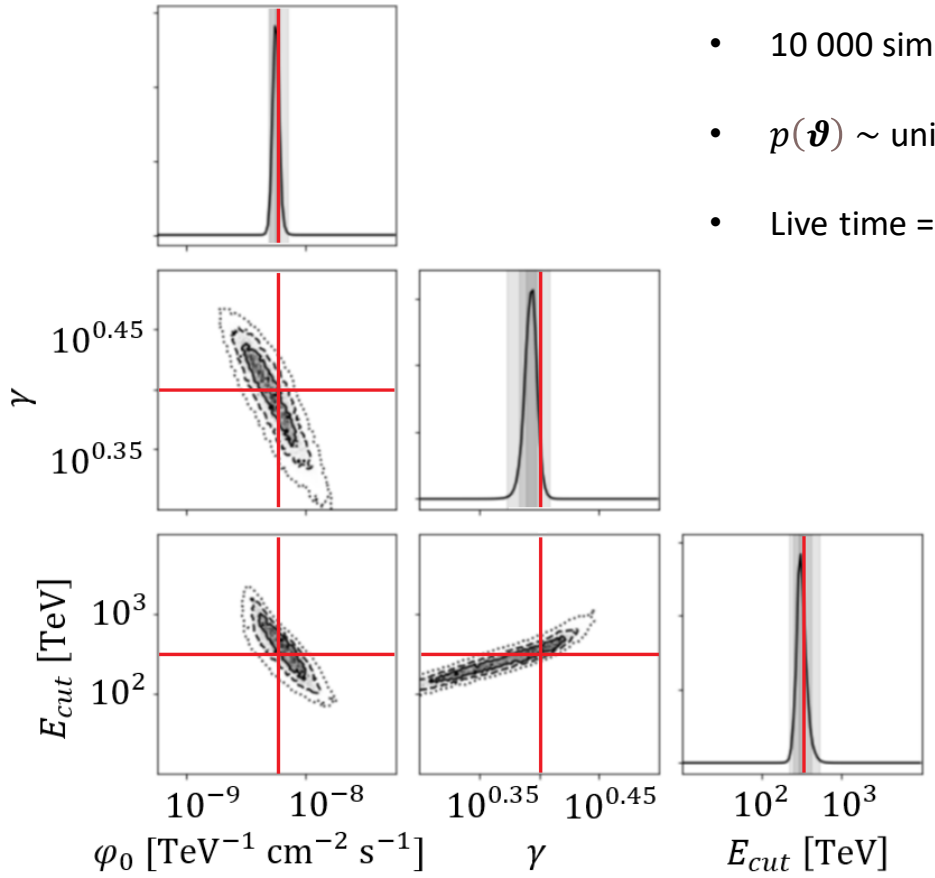
Inference with NRE seems to be precise for the spectral fit

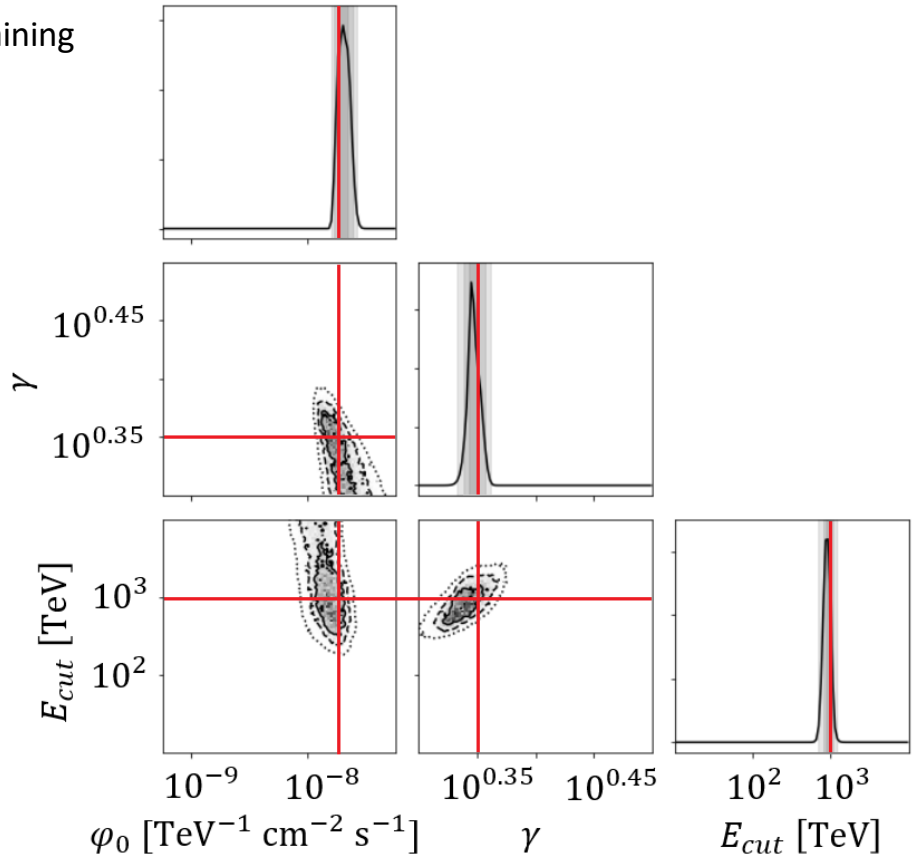


- 10 000 simulations used in training
- $p(\boldsymbol{\vartheta}) \sim$ uniform on log scale
- Live time = 50 hr

Inference with NRE seems to be precise for the spectral fit

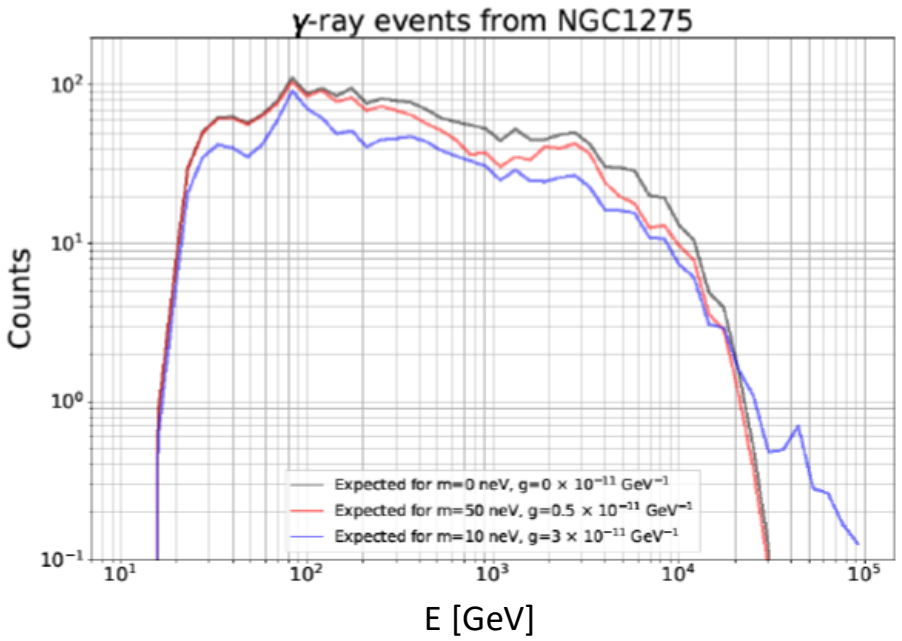
- 10 000 simulations used in training
- $p(\boldsymbol{\vartheta}) \sim$ uniform on log scale
- Live time = 50 hr





The method seems particularly useful for ALP searches with gamma-telescopes

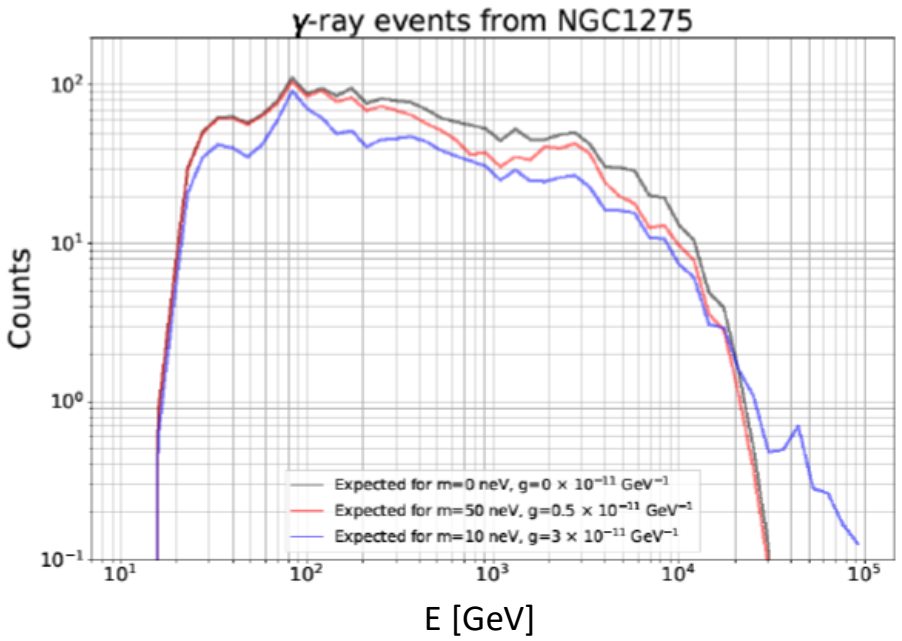
Expected result assuming ALPs with given mass *m* and coupling *g* (using `gammaALPs`):



The expected spectrum can be simulated using `gammapy` and `gammaALPs` (by M. Meyer)

The method seems particularly useful for ALP searches with gamma-telescopes

Expected result assuming ALPs with given mass *m* and coupling *g* (using `gammaALPs`):

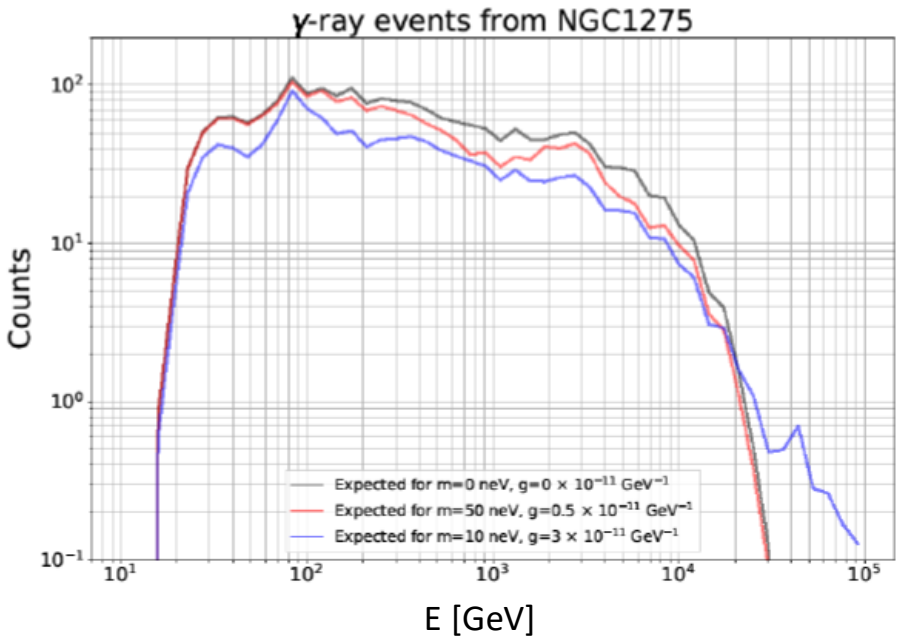


The expected spectrum can be simulated using `gammapy` and `gammaALPs` (by M. Meyer)

- Parameters of interest:
  - Mass of ALPs, *m*
  - ALP-photon coupling, *g*

The method seems particularly useful for ALP searches with gamma-telescopes

Expected result assuming ALPs with given mass *m* and coupling *g* (using `gammaALPs`):

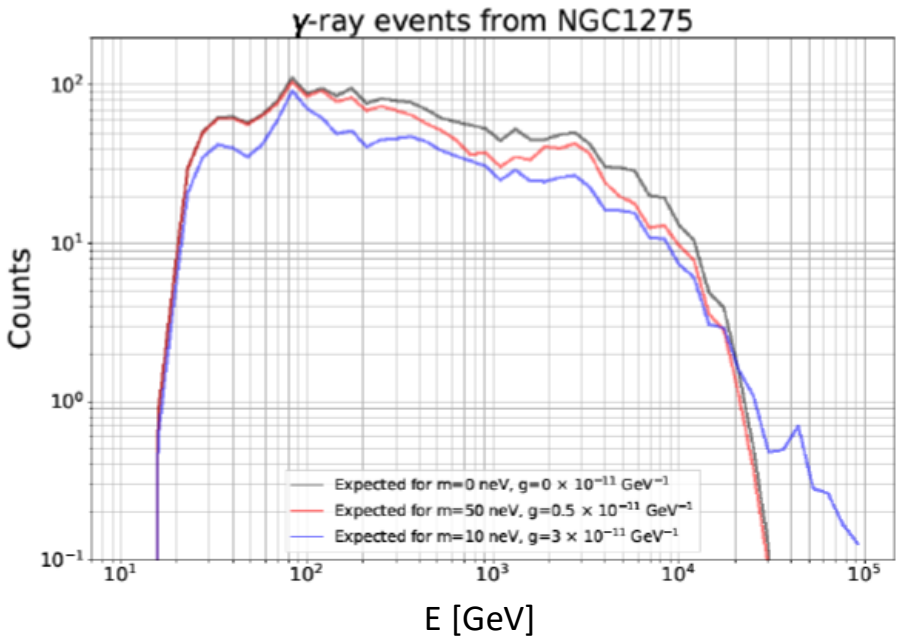


The expected spectrum can be simulated using `gammapy` and `gammaALPs` (by M. Meyer)

- Parameters of interest:
  - Mass of ALPs, *m*
  - ALP-photon coupling, *g*
- Nuisance parameters:
  - Amplitude
  - Spectral index
  - Cut-off energy
  - Magnetic field configuration
  - + 12 more related to configuration of NGC1275

The method seems particularly useful for ALP searches with gamma-telescopes

Expected result assuming ALPs with given mass *m* and coupling *g* (using `gammaALPs`):



The expected spectrum can be simulated using `gammapy` and `gammaALPs` (by M. Meyer)

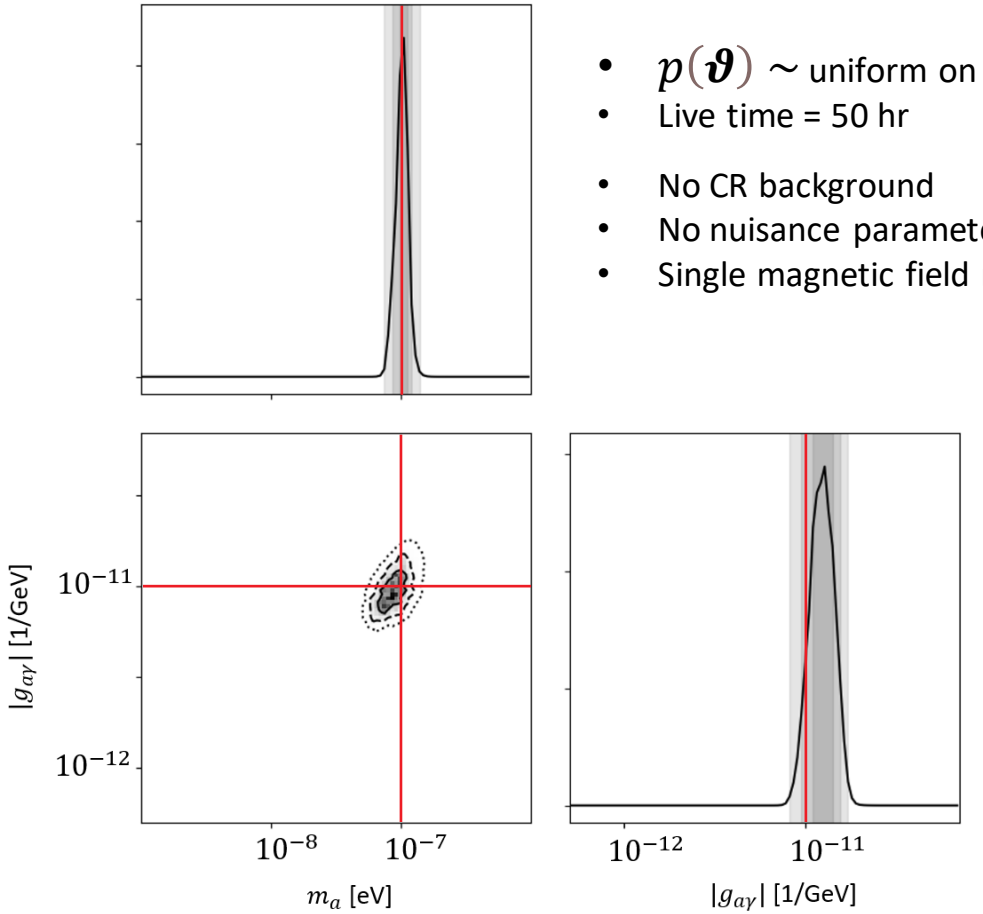
- Parameters of interest:
  - Mass of ALPs, *m*
  - ALP-photon coupling, *g*
- Nuisance parameters:
  - Amplitude
  - Spectral index
  - Cut-off energy
  - Magnetic field configuration
  - + 12 more related to configuration of NGC1275

- For ALP searches, the frequentist test statistic does *not* obey Wilk's theorem!

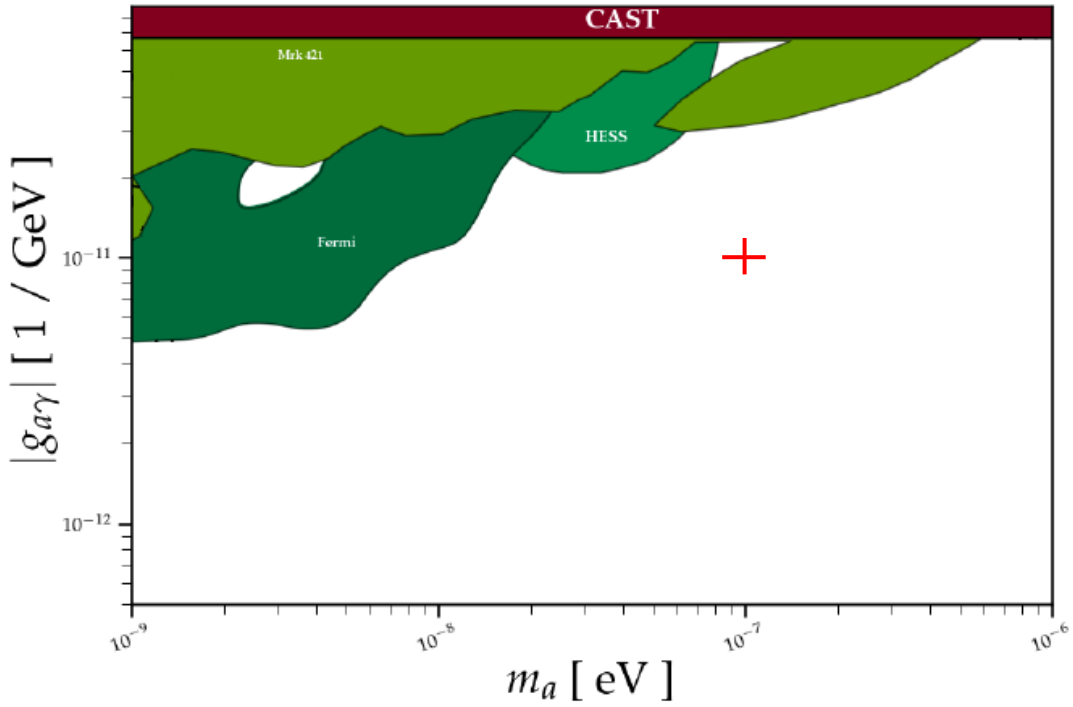
→ Monte Carlo simulations are necessary to relate the *TS* to a significance of detection or exclusion **for each point in parameter space.**

→ Conventional computations are extremely expensive

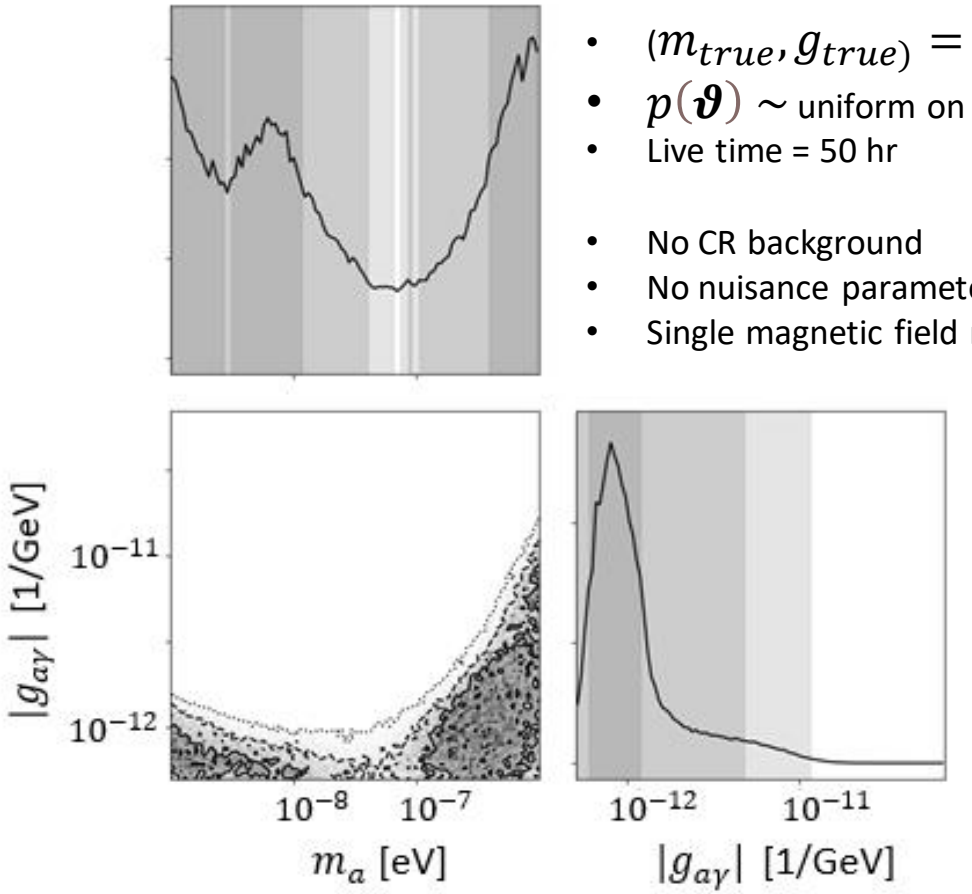
Preliminary results indicate the method is suitable for ALP searches



- $p(\boldsymbol{\vartheta})$ ~ uniform on log scale
- Live time = 50 hr
- No CR background
- No nuisance parameters
- Single magnetic field realization



Preliminary results indicate the method is suitable for ALP searches



- $(m_{true}, g_{true}) = (0,0)$
- $p(\boldsymbol{\vartheta}) \sim$ uniform on log scale
- Live time = 50 hr

- No CR background
- No nuisance parameters
- Single magnetic field realization