`Swyft` for cosmological N-body simulations

arXiv:2206.11312

Androniki Dimitriou, Christoph Weniger, Camila Correa

Simulation-based Inference with Swyft
24 January 2023, Amsterdam

- **Goal:** reconstruction of **halo clustering** and **halo mass function** of DM-only cosmological simulations generated by the **EAGLE project**

- **Goal:** reconstruction of **halo clustering** and **halo mass function** of DM-only cosmological simulations generated by the **EAGLE project**

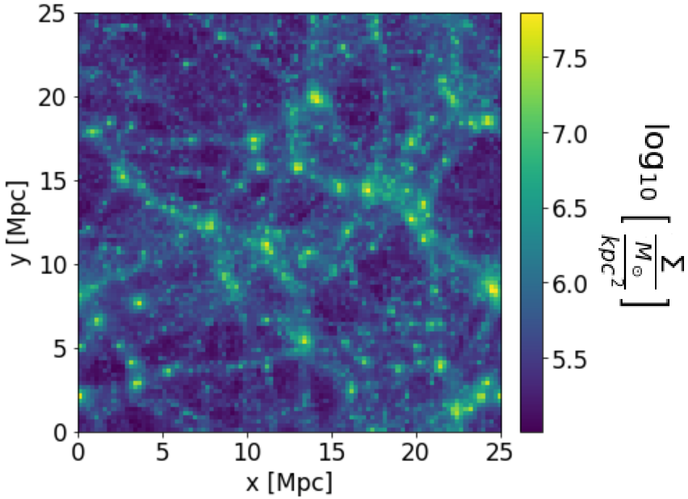
- **Approach:** direct calibration of **a analytical halo** model/simulator **based on a toy implementation of two body correlation functions** on DM-only **simulations**

- **Goal:** reconstruction of **halo clustering** and **halo mass function** of DM-only cosmological simulations generated by the **EAGLE project**

- **Approach:** direct calibration of **a analytical halo** model/simulator **based on a toy implementation of two body correlation functions** on DM-only **simulations**

- **The EAGLE project**

$(25\ \text{Mpc})^3$ box with 376^3 particles

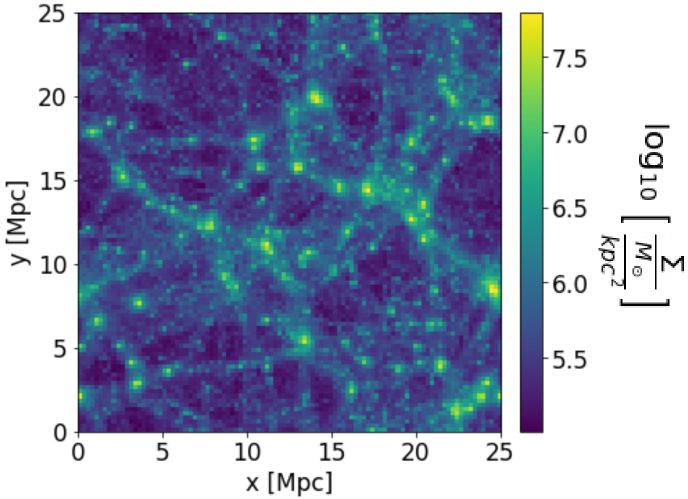


- **Goal:** reconstruction of **halo clustering** and **halo mass function** of DM-only cosmological simulations generated by the **EAGLE project**

- **Approach:** direct calibration of **a analytical halo** model/simulator **based on a toy implementation of two body correlation functions** on DM-only **simulations**

- **The EAGLE project**

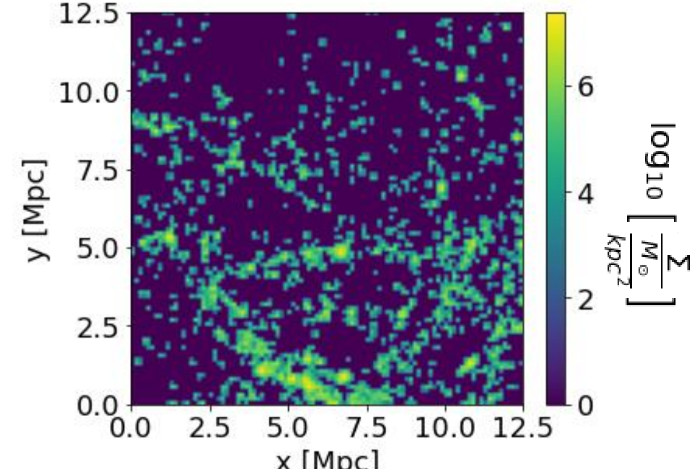
$(25\ \text{Mpc})^3$ box with 376^3 particles



FOF halo finder

Particle data

$M_h \in (10^9, 10^{12})\ M_\odot$, $(12.5\ \text{Mpc})^3$ box

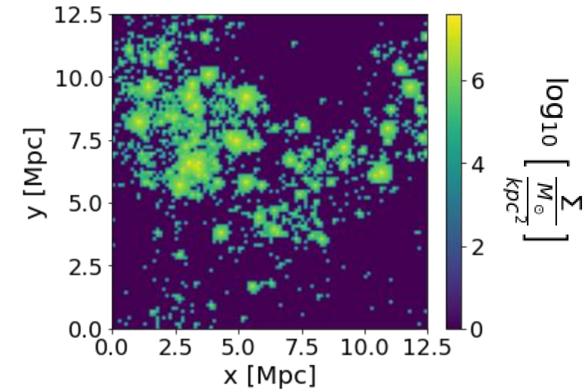


- **The toy halo model**

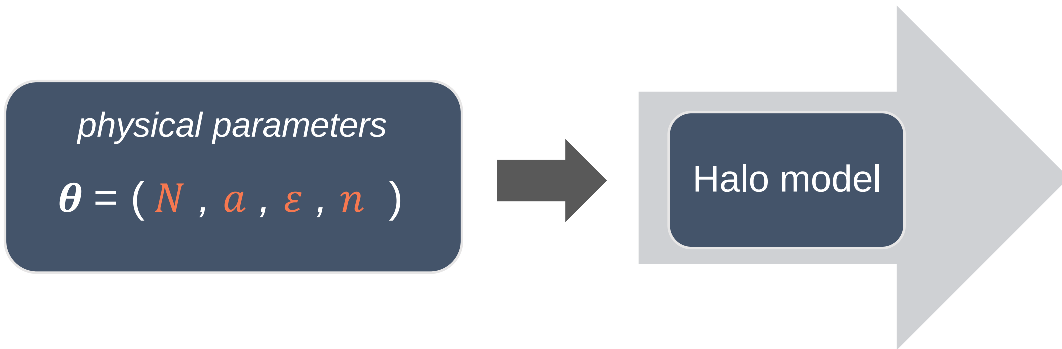
physical parameters

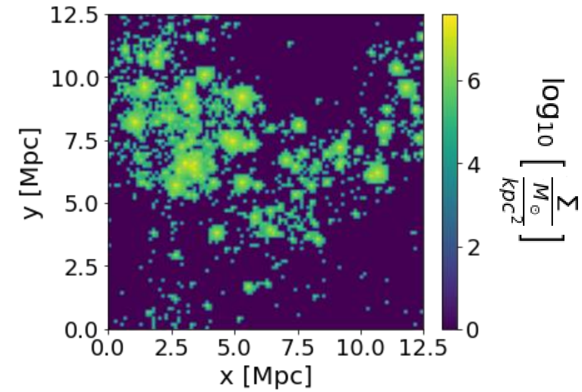
$\boldsymbol{\theta} = (N, a, \varepsilon, n)$

Halo model



- **The toy halo model**





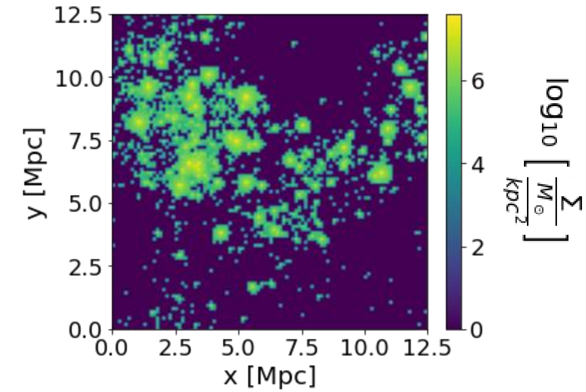
➢ The **first physical parameter** is the number, *N* , of the haloes

- **The toy halo model**

physical parameters

$\boldsymbol{\theta} = (\,N\,,\,a\,,\,\varepsilon\,,\,n\,)$

Halo model

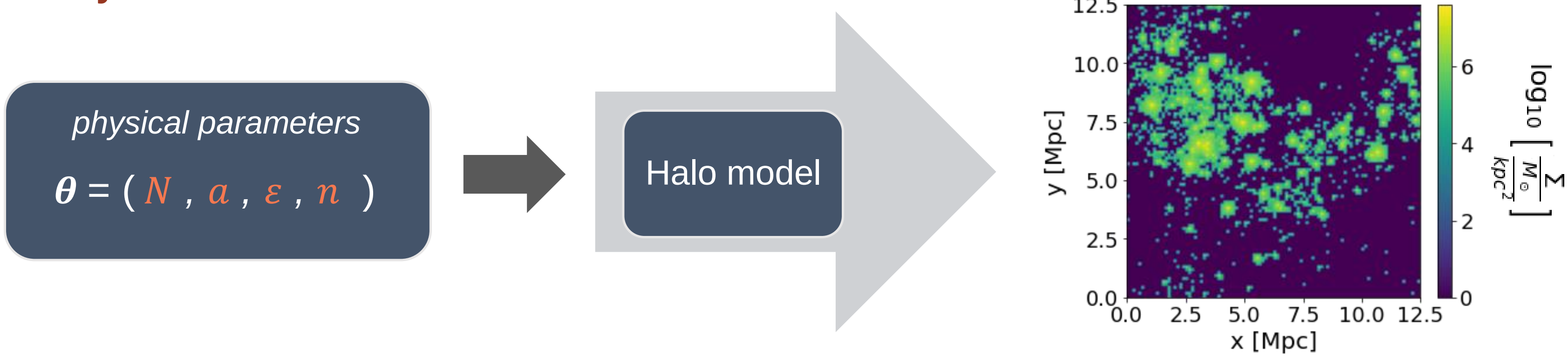


➢ The **first physical parameter** is the number, N , of the haloes

➢ The masses of the haloes, $M_h \in (10^9, 10^{12})\ \mathrm{M}_\odot$, can be sampled from a halo mass function

$$\frac{dn}{dM} \propto M^{-a}$$

- **The toy halo model**



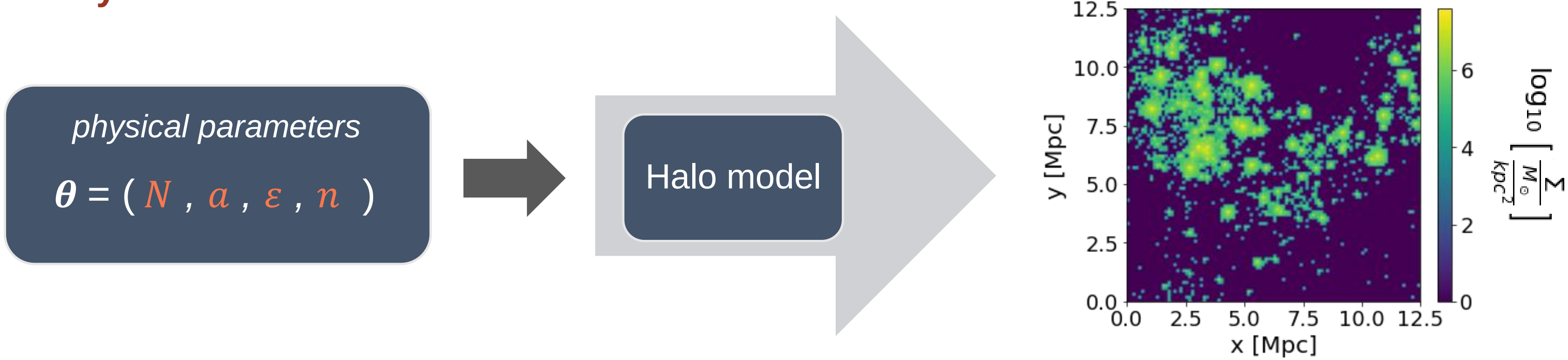
➢ The **first physical parameter** is the number, N , of the haloes

➢ The masses of the haloes, $M_h \in (10^9, 10^{12})$ $M_\odot$, can be sampled from a halo mass function

$$\frac{dn}{dM} \propto M^{-a}$$

➢ The **second physical parameter** is the slope, a, of the halo mass function

- **The toy halo model**



- The **first physical parameter** is the number, N , of the haloes

- The masses of the haloes, $M_h \in (10^9, 10^{12})$ $M_\odot$, can be sampled from a halo mass function

$$\frac{dn}{dM} \propto M^{-a}$$

- The **second physical parameter** is the slope, a, of the halo mass function

- We construct a 100x100 grid whose values correspond to pairs of x and y coordinates, where $(x, y) \in (0,12.5)$ Mpc

- **Adding Clustering to the Model**

  - We will sample the positions according to distributions generated from 2D realizations of gaussian random fields, $\boldsymbol{\delta}$, on an 100x100 grid

- **Adding Clustering to the Model**

  - We will sample the positions according to distributions generated from 2D realizations of gaussian random fields, $\boldsymbol{\delta}$, on an 100x100 grid

  - The gaussian fields will be specified by a power-law power spectrum

$$P(k) \propto \frac{1}{k^n}$$

- **Adding Clustering to the Model**

  - We will sample the positions according to distributions generated from 2D realizations of gaussian random fields, $\boldsymbol{\delta}$, on an 100x100 grid

  - The gaussian fields will be specified by a power-law power spectrum

$$P(k) \propto \frac{1}{k^n}$$

  - The slope of the power spectrum, n, is the **third physical parameter** of our model

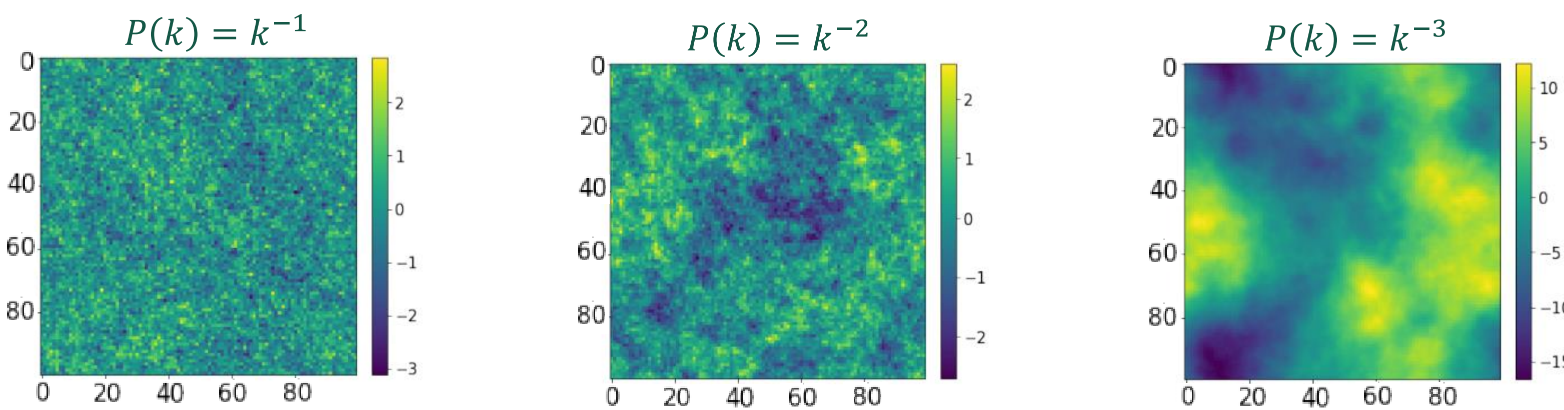
- **Adding Clustering to the Model**

  - We will sample the positions according to distributions generated from 2D realizations of gaussian random fields, $\boldsymbol{\delta}$, on an 100x100 grid

  - The gaussian fields will be specified by a power-law power spectrum

$$P(k) \propto \frac{1}{k^n}$$

  - The slope of the power spectrum, n, is the **third physical parameter** of our model



- **Adding Clustering to the Model**

  - We transform the field $\boldsymbol{\delta}$ to a probability distribution function in order to sample from it:

- **Adding Clustering to the Model**

  - We transform the field $\boldsymbol{\delta}$ to a probability distribution function in order to sample from it:

    - We first multiply $\boldsymbol{\delta}$ with a **fourth physical parameter** ε,

- **Adding Clustering to the Model**

  - We transform the field $\boldsymbol{\delta}$ to a probability distribution function in order to sample from it:

    - We first multiply $\boldsymbol{\delta}$ with a **fourth physical parameter** ε,

    - We exponentiate $\boldsymbol{\delta} \cdot \varepsilon$,

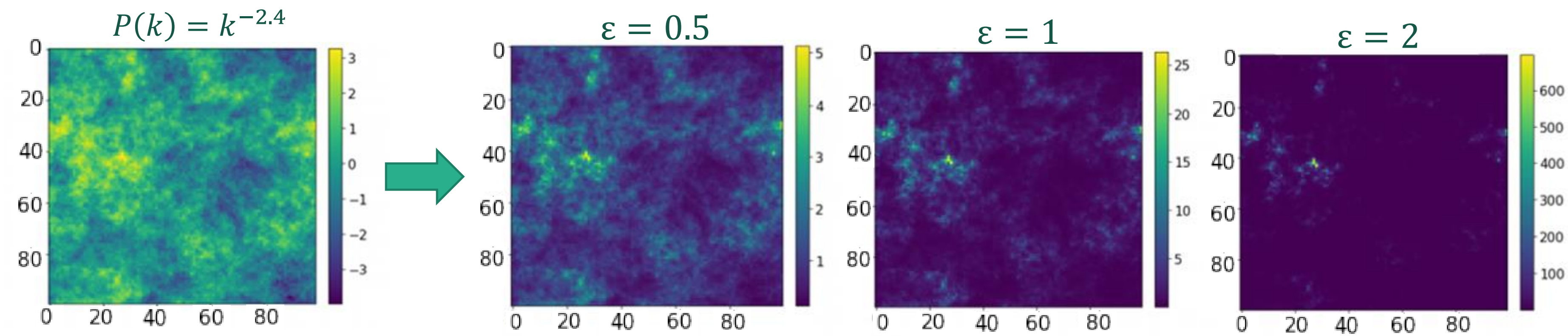
- **Adding Clustering to the Model**

  - We transform the field $\boldsymbol{\delta}$ to a probability distribution function in order to sample from it:

    - We first multiply $\boldsymbol{\delta}$ with a **fourth physical parameter** ε,
    - We exponentiate $\boldsymbol{\delta} \cdot \varepsilon$,
    - We normalize the field $\boldsymbol{f} = e^{\boldsymbol{\delta} \cdot \varepsilon}$, s.t., its values sum to 1

- **Adding Clustering to the Model**

  - We transform the field $\boldsymbol{\delta}$ to a probability distribution function in order to sample from it:
    - We first multiply $\boldsymbol{\delta}$ with a **fourth physical parameter** ε,
    - We exponentiate $\boldsymbol{\delta} \cdot \varepsilon$,
    - We normalize the field $\boldsymbol{f} = e^{\boldsymbol{\delta} \cdot \varepsilon}$, s.t., its values sum to 1



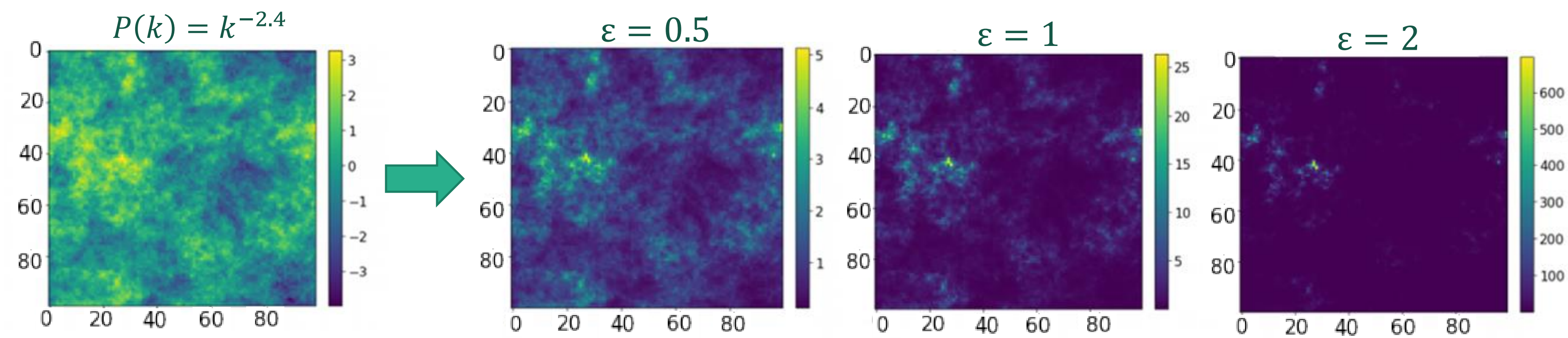
- **Adding Clustering to the Model**

  - We transform the field $\boldsymbol{\delta}$ to a probability distribution function in order to sample from it:

    - We first multiply $\boldsymbol{\delta}$ with a **fourth physical parameter** ε,

    - We exponentiate $\boldsymbol{\delta} \cdot \varepsilon$,

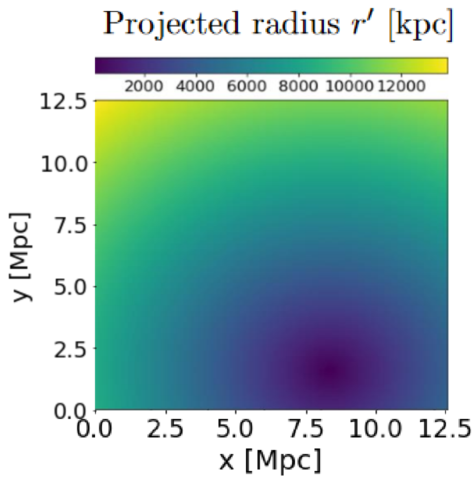
    - We normalize the field $\boldsymbol{f} = e^{\boldsymbol{\delta} \cdot \varepsilon}$, s.t., its values sum to 1



  - We sample the positions of the haloes according to this distribution

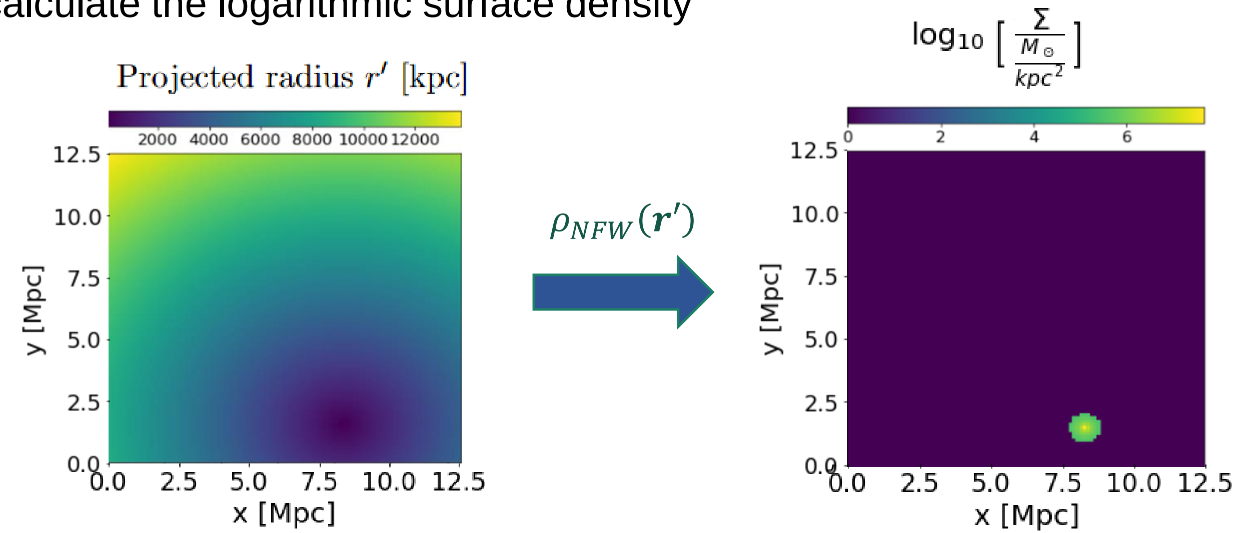
- **The toy halo model**

  - ➢ We calculate the logarithmic surface density



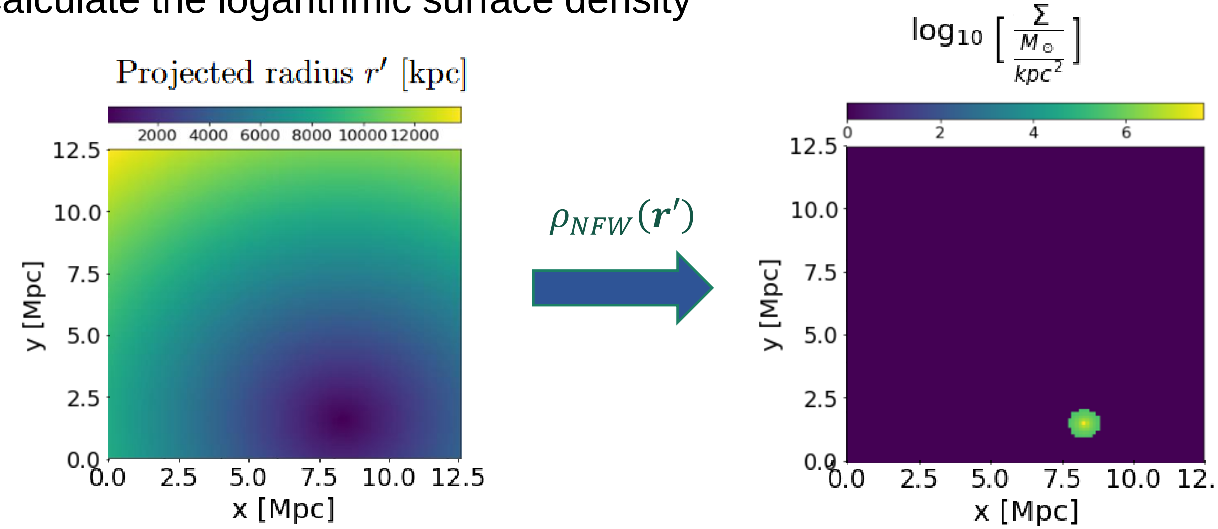
- **The toy halo model**

  - We calculate the logarithmic surface density



- **The toy halo model**

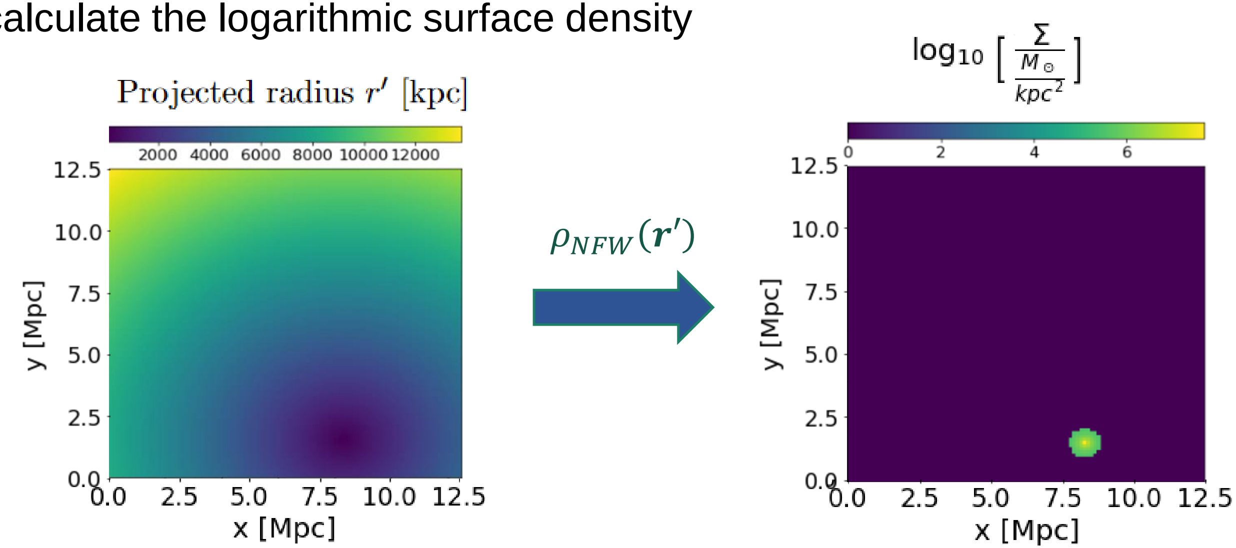
  - We calculate the logarithmic surface density



  - We add all the images of the individual haloes together to obtain the total surface density field

- **The toy halo model**

  - We calculate the logarithmic surface density

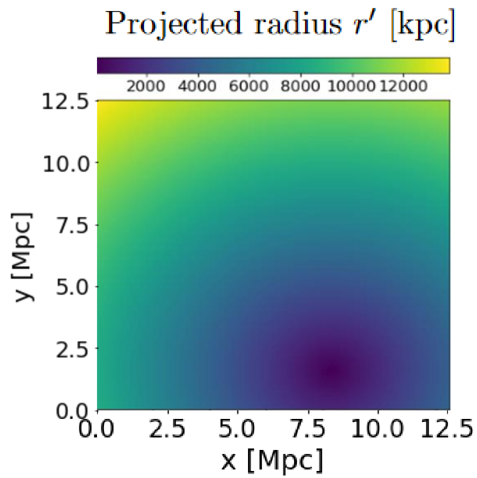


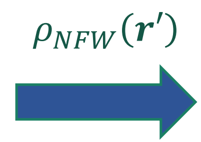
  - We add all the images of the individual haloes together to obtain the total surface density field

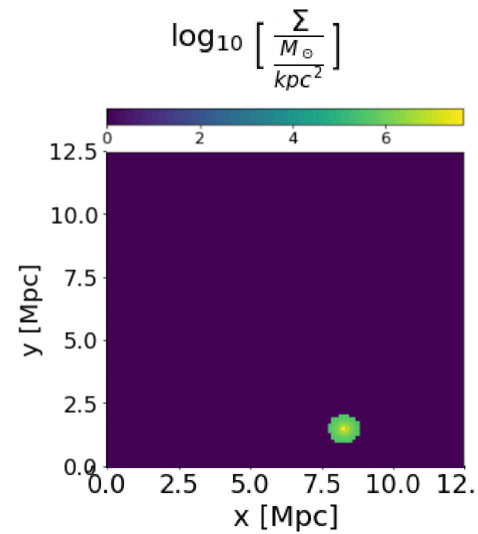
  - We add poisson noise to the final image

- **The toy halo model**

  - We calculate the logarithmic surface density

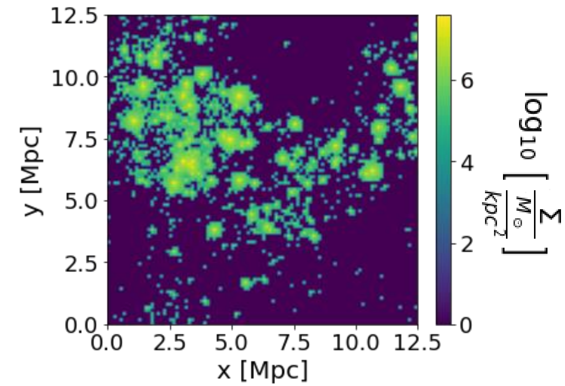






  - We add all the images of the individual haloes together to obtain the total surface density field

  - We add poisson noise to the final image



- **The toy halo model**

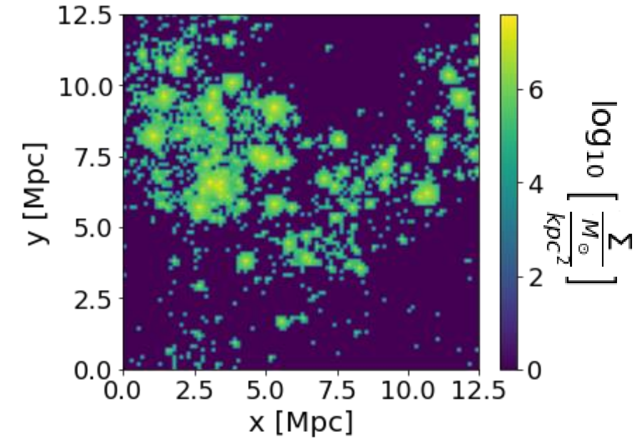
physical parameters

$\boldsymbol{\theta} = (N, a, \varepsilon, n)$

Simulator

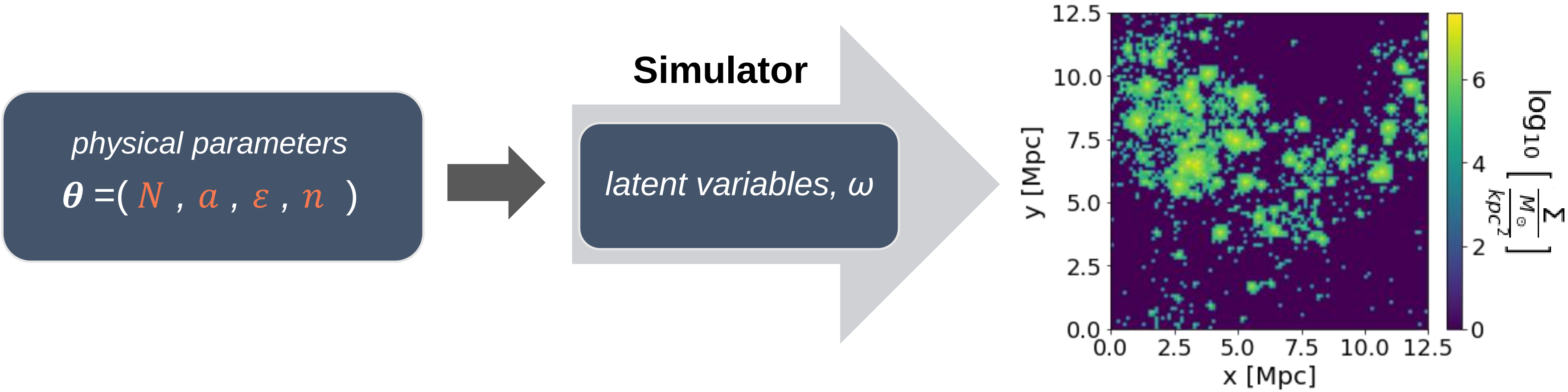
latent variables, ω

$$\boldsymbol{x} \sim p(\boldsymbol{x}|\boldsymbol{z}) = p(\boldsymbol{x}|\boldsymbol{\theta}, \boldsymbol{\omega})$$



- **The toy halo model**

$$\boldsymbol{x} \sim p(\boldsymbol{x}|\boldsymbol{z}) = p(\boldsymbol{x}|\boldsymbol{\theta}, \boldsymbol{\omega})$$



➢ Given an image, we want to know which parameters, $\boldsymbol{\theta}$, generated it

- ## The toy halo model

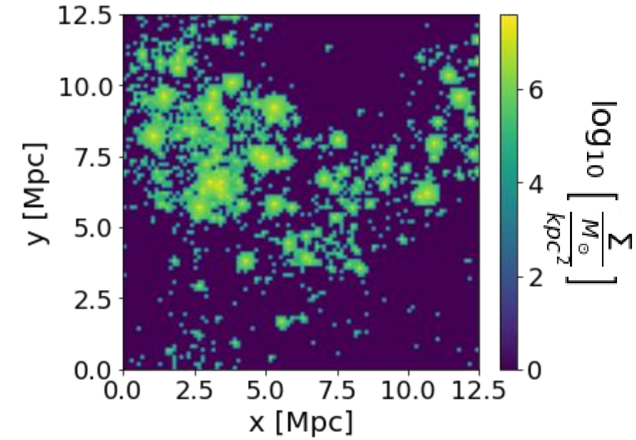
physical parameters

$\boldsymbol{\theta} = (N, a, \varepsilon, n)$

Simulator

latent variables, ω

$$\boldsymbol{x} \sim p(\boldsymbol{x}|\boldsymbol{z}) = p(\boldsymbol{x}|\boldsymbol{\theta}, \boldsymbol{\omega})$$



- Given an image, we want to know which parameters, $\boldsymbol{\theta}$, generated it
- We want to calculate **marginal posteriors** of the parameters of interest, $\boldsymbol{\vartheta}$

$$p(\boldsymbol{\vartheta}|\boldsymbol{x}) = \int d\boldsymbol{\eta}\, p(\boldsymbol{\eta}, \boldsymbol{\vartheta}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{\vartheta})p(\boldsymbol{\vartheta})}{p(\boldsymbol{x})}$$

nuisance parameters

- **The toy halo model**

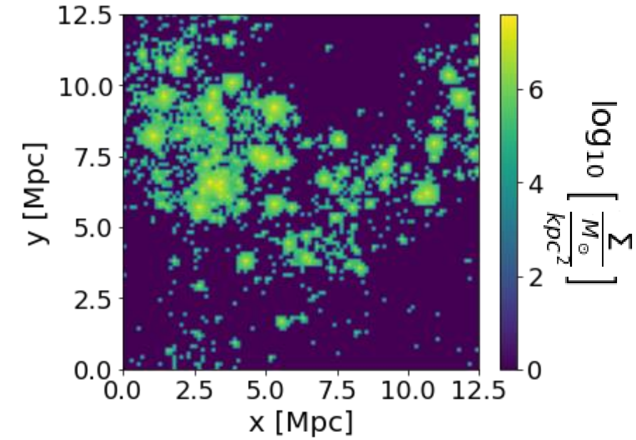
physical parameters

$\boldsymbol{\theta} = (N, a, \varepsilon, n)$

Simulator

latent variables, ω

$\boldsymbol{x} \sim p(\boldsymbol{x}|\boldsymbol{z}) = p(\boldsymbol{x}|\boldsymbol{\theta}, \boldsymbol{\omega})$



➢ Given an image, we want to know which parameters, $\boldsymbol{\theta}$, generated it

➢ We want to calculate **marginal posteriors** of the parameters of interest, $\boldsymbol{\vartheta}$

$$p(\boldsymbol{\vartheta}|\boldsymbol{x}) = \int d\boldsymbol{\eta}\, p(\boldsymbol{\eta}, \boldsymbol{\vartheta}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{\vartheta})p(\boldsymbol{\vartheta})}{p(\boldsymbol{x})}$$

nuisance parameters

Intractable due to high dimensionality

Training with `swyft` (MNRE)

arXiv:2107.01214

- **Physical parameters:**

  - N: Number of halos, where $N \in (100, 2100)$
  - α : Inner slope of the halo mass function, where $\alpha \in (1, 3)$

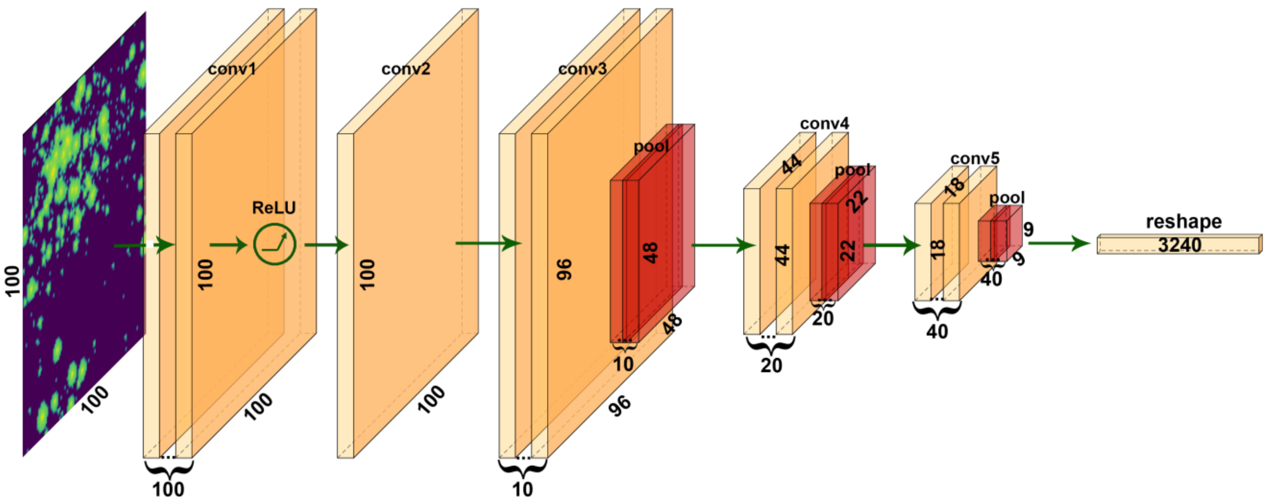
ε: Exponent of the density field, where $\varepsilon \in (0,2)$

n: Slope of the power spectrum, where $n \in (0,10)$

Training with `swyft` (MNRE)

arXiv:2107.01214

- **Physical parameters:**
  - *N*: Number of halos, where $N \in (100, 2100)$
  - a : Inner slope of the halo mass function, where $a \in (1, 3)$
  - ε: Exponent of the density field, where $\varepsilon \in (0,2)$
  - *n*: Slope of the power spectrum, where $n \in (0,10)$
- We define a **CNN**:



Training with `swyft` (MNRE)

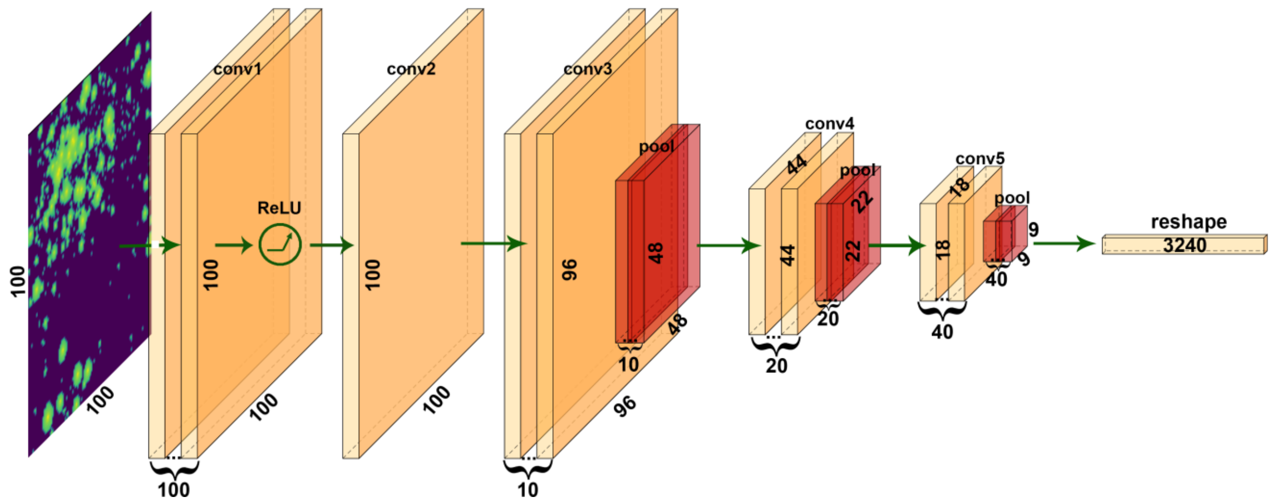
arXiv:2107.01214

- **Physical parameters:**
  - N: Number of halos, where $N \in (100, 2100)$
  - α : Inner slope of the halo mass function, where $\alpha \in (1, 3)$

  ε: Exponent of the density field, where $\varepsilon \in (0,2)$

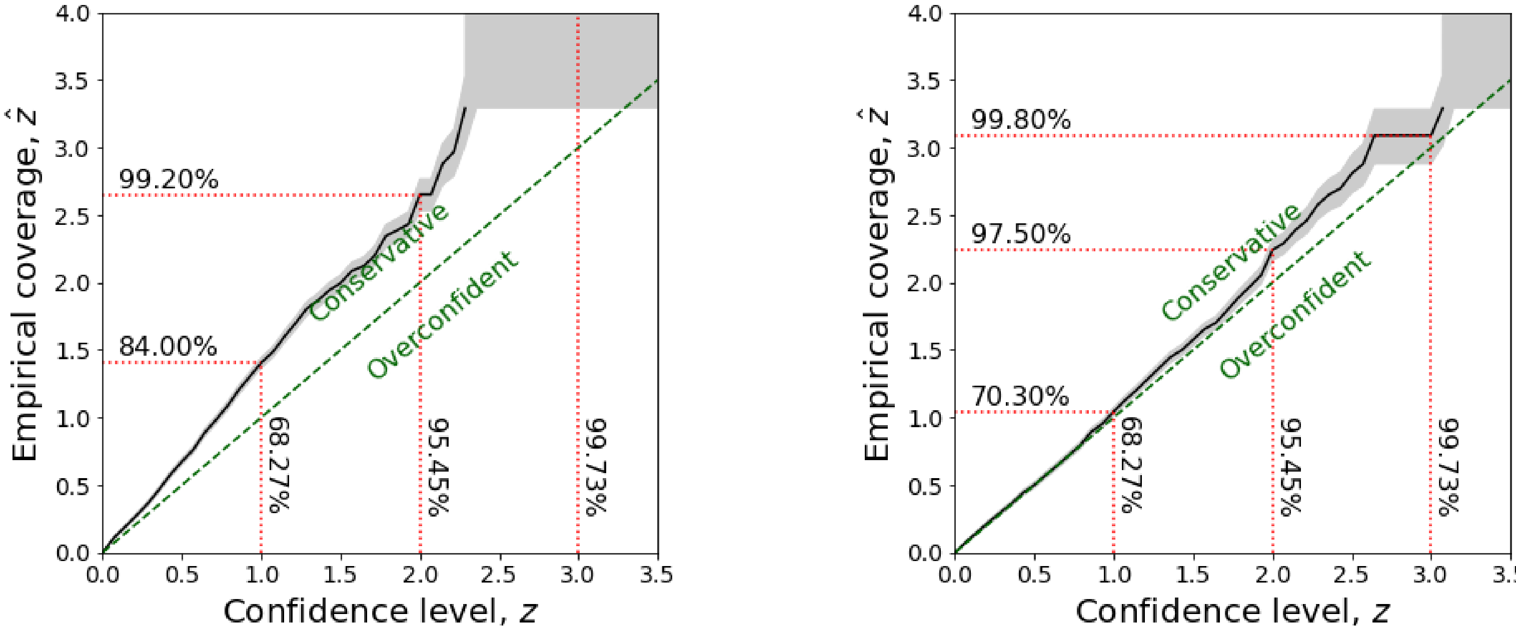
  n: Slope of the power spectrum, where $n \in (0,10)$

- We define a **CNN**:



- We **train** using 200.000 mock images

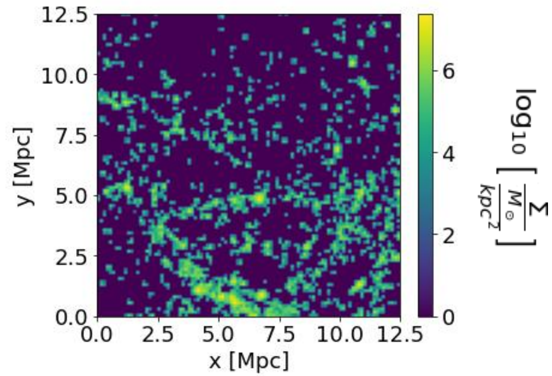
Results on mock data



number of halos (N)   slope of power spectrum (n)

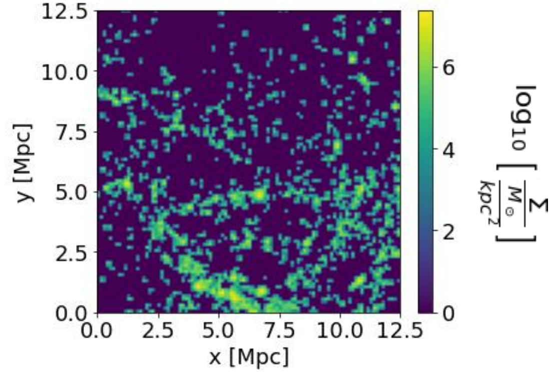
Results on actual N body simulations

- one simulation box

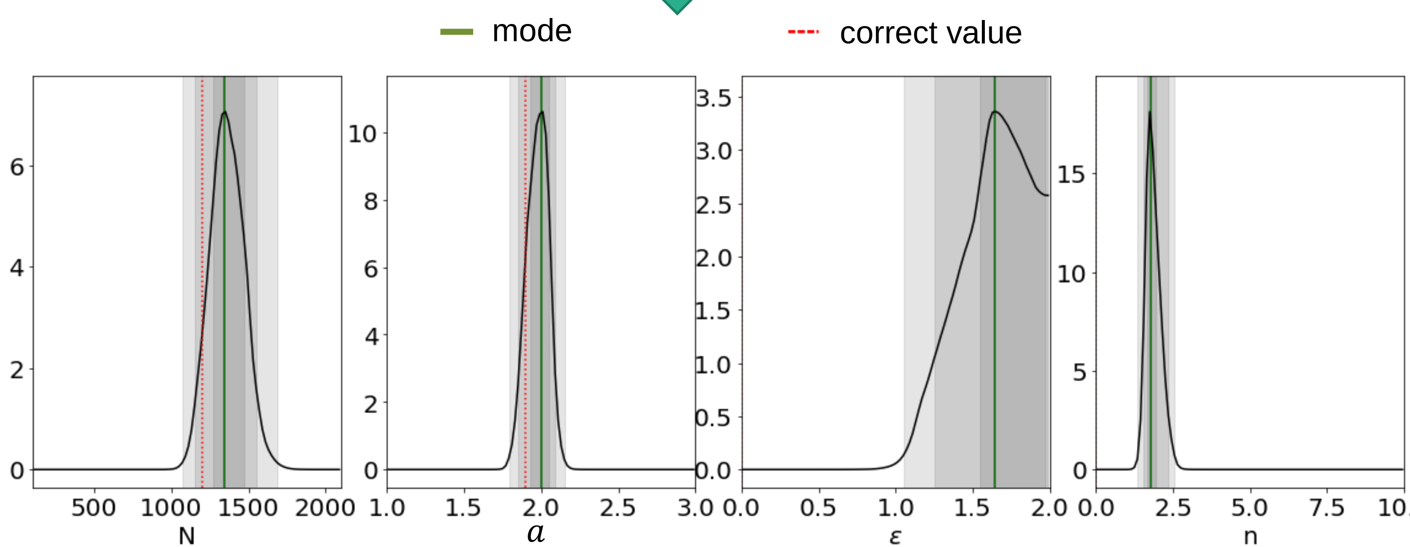


Results on actual N body simulations

- one simulation box



trained NN

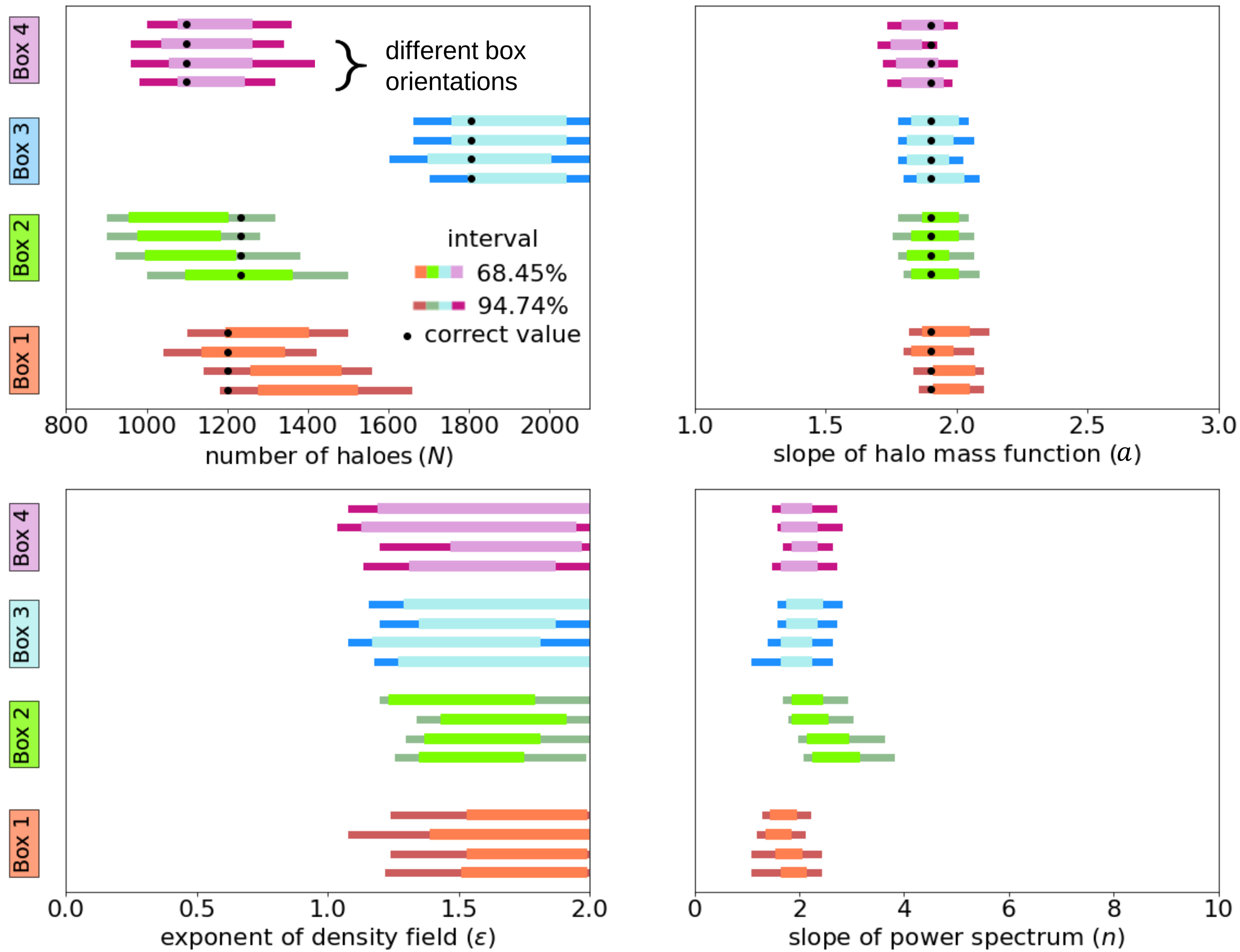


1D Posteriors

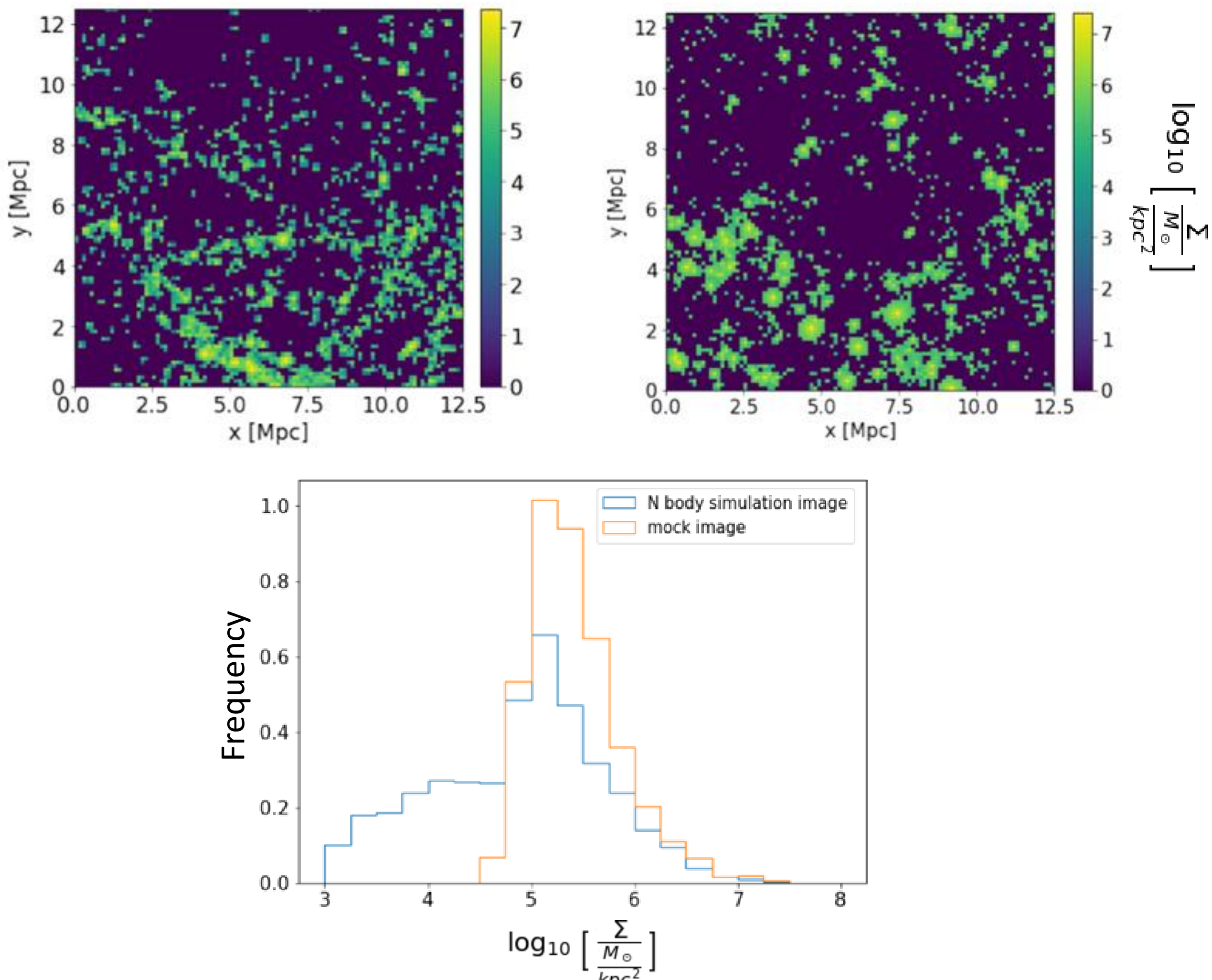
Results on actual N body simulations

- 4 simulation boxes and 3 rotations of them

Accurate marginal posteriors!



Comparison of a N-body simulation and a mock image



- **Using a toy halo model and `swyft` (MNRE)**

- **Using a toy halo model and `swyft` (MNRE)**
  - We reconstructed the halo mass function

- **Using a toy halo model and `swyft` (MNRE)**
    - ➢ We reconstructed the halo mass function
    - ➢ We produced images similar to N-body simulations

- **Using a toy halo model and `swyft` (MNRE)**
    - We reconstructed the halo mass function
    - We produced images similar to N-body simulations
- As a **next step** we can test if we can identify the lowest mass of the haloes in these images

- **Using a toy halo model and `swyft` (MNRE)**

  - We reconstructed the halo mass function

  - We produced images similar to N-body simulations

- As a **next step** we can test if we can identify the lowest mass of the haloes in these images

  To do that:

  - Instead of sampling haloes with masses: $M_h \in (10^9, 10^{12})\ M_\odot$,
    we will sample masses: $M_h \in (10^c, 10^{12})\ M_\odot$,

- **Using a toy halo model and `swyft` (MNRE)**

  - We reconstructed the halo mass function
  - We produced images similar to N-body simulations

- As a **next step** we can test if we can identify the lowest mass of the haloes in these images

  To do that:

  - Instead of sampling haloes with masses: $M_h \in (10^9, 10^{12})\ M_\odot$,
    we will sample masses: $M_h \in (10^c, 10^{12})\ M_\odot$,
  - where c is **a new parameter** of our model

- **Using a toy halo model and `swyft` (MNRE)**

    - We reconstructed the halo mass function

    - We produced images similar to N-body simulations

- As a **next step** we can test if we can identify the lowest mass of the haloes in these images

  To do that:

    - Instead of sampling haloes with masses: $M_h \in (10^9, 10^{12})\ M_{\odot}$,
      we will sample masses: $M_h \in (10^c, 10^{12})\ M_{\odot}$,

    - where c is **a new parameter** of our model

    - We set the values of the parameters a, ε and n equal to the modes of their combined posteriors

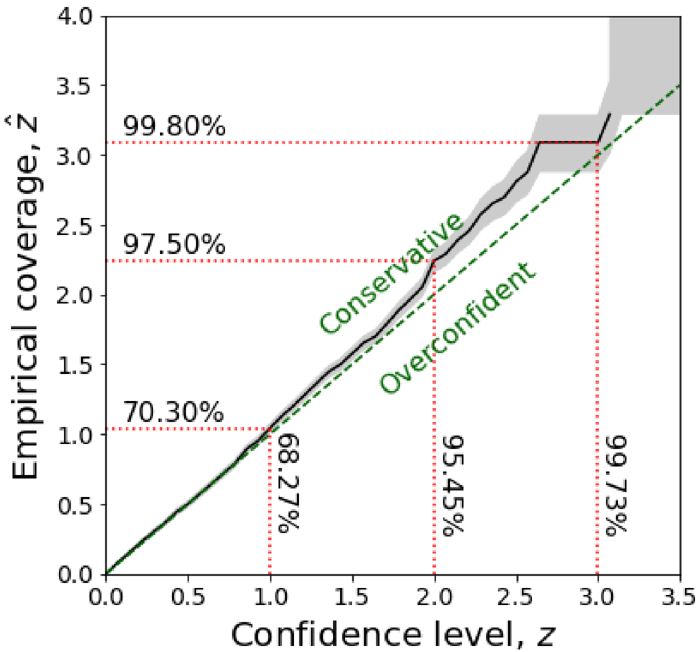
- **Physical parameters:**

  - N: Number of halos, where N ϵ (100,2100)

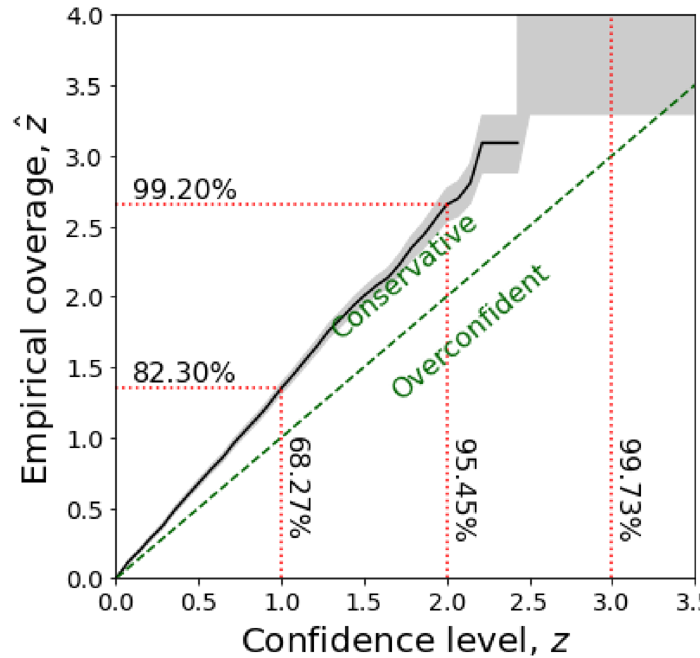
  - c: Lower cutoff of the halo mass function, where c ϵ (8,10.5)

- We **train** using 50.000 mock images

Results on mock data



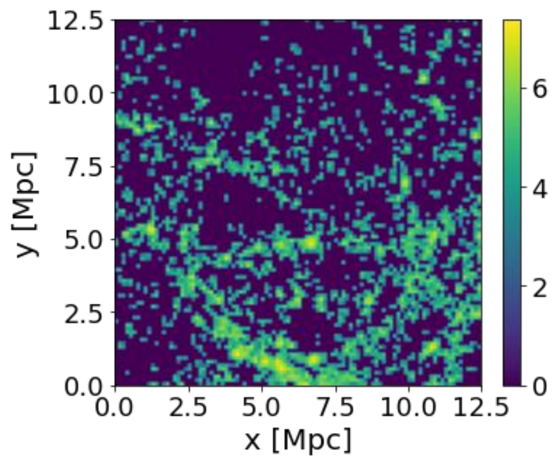
number of haloes (N)



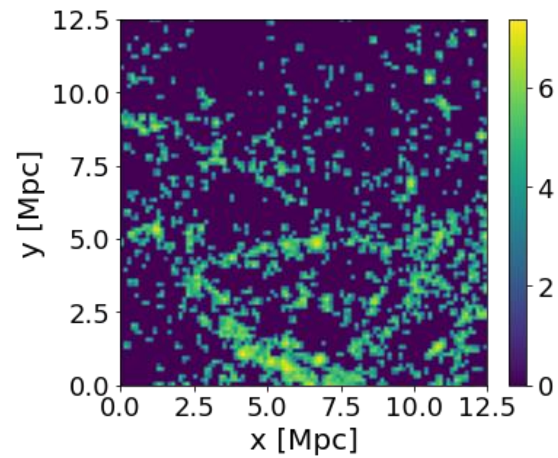
$\log_{10}$ cutoff of
halo mass function (c)

Results on actual N body simulations

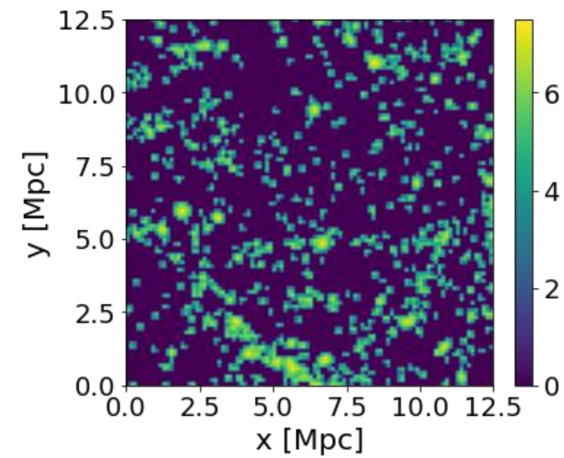
- one simulation box with different cutoffs



c=8.75



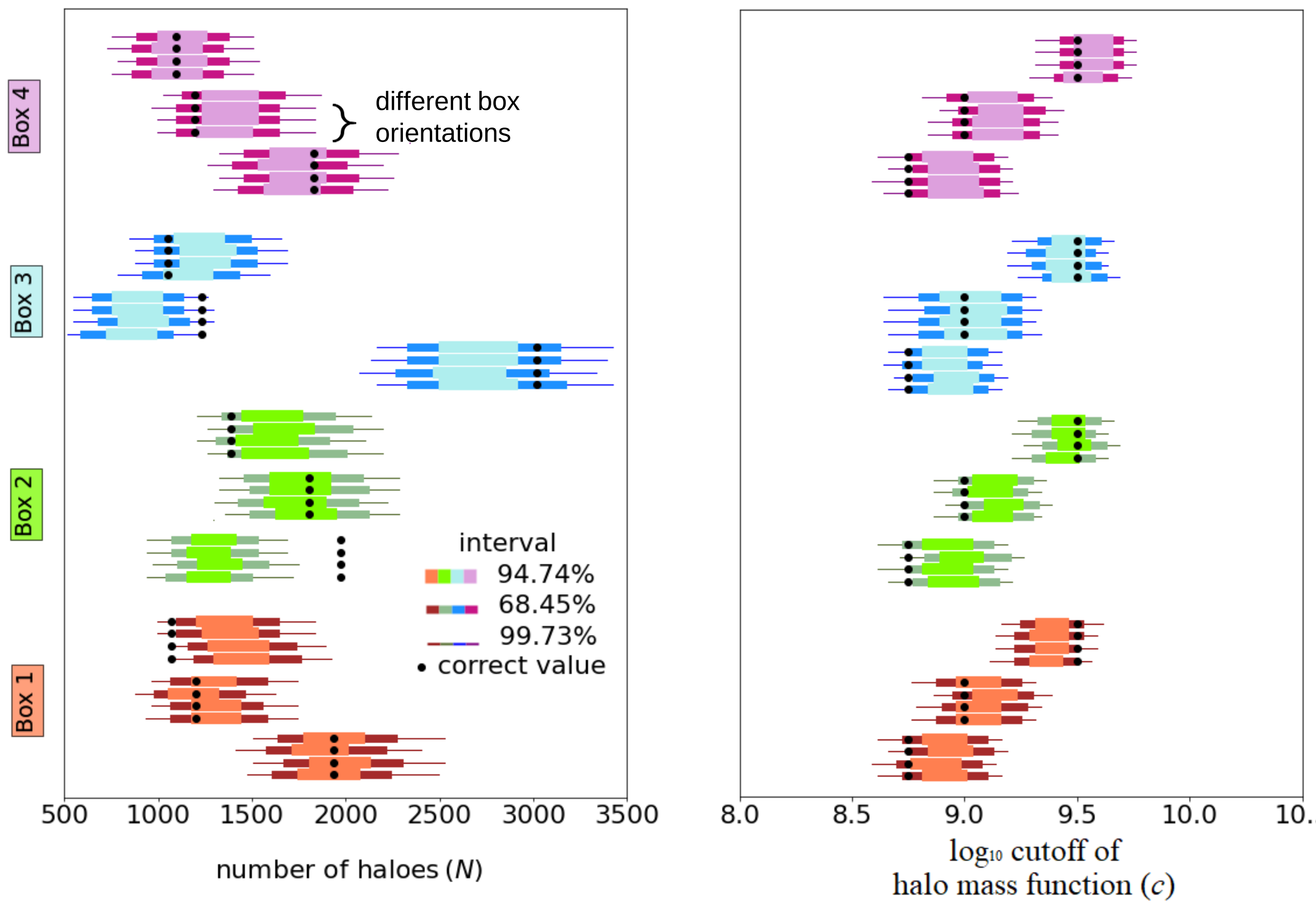
c=9



c=9.5

Results on actual N body simulations

- 4 simulation boxes with 3 different cutoffs and 3 rotations of them



- **Using a toy halo model and `swyft` (MNRE)**

- **Using a toy halo model and `swyft` (MNRE)**
  - We reconstructed the halo mass function

- **Using a toy halo model and `swyft` (MNRE)**
  - We reconstructed the halo mass function
  - We reconstructed a lower cutoff of the halo mass function

- **Using a toy halo model and `swyft` (MNRE)**
  - We reconstructed the halo mass function
  - We reconstructed a lower cutoff of the halo mass function
  - We produced images similar to N-body simulations

- **Using a toy halo model and `swyft` (MNRE)**

    - We reconstructed the halo mass function

    - We reconstructed a lower cutoff of the halo mass function

    - We produced images similar to N-body simulations

- **Long-term goal**

A analytical model for haloes, subhaloes, clustering and baryonic matter that generates actual N-body simulations

- **Using a toy halo model and `swyft` (MNRE)**

    - ➢ We reconstructed the halo mass function
    - ➢ We reconstructed a lower cutoff of the halo mass function
    - ➢ We produced images similar to N-body simulations

- **Long-term goal**

A analytical model for haloes, subhaloes, clustering and baryonic matter that generates actual N-body simulations

Thank you!

Backup Slides

The Model

- The masses of the haloes can be sampled from a halo mass function: $\frac{dn}{dM} = bM^{a}$

Inverse Transform Sampling:

$$PDF(M) = \frac{bM^{-a}}{\int_{10^9}^{10^{12}} bM^{-a}dM} = \frac{M^{-a}}{\int_{10^9}^{10^{12}} M^{-a}dM} = \frac{M^{-a}(1-a)}{(10^{12})^{1-a} - (10^9)^{1-a}}$$

$$F(M) = \frac{\int_{10^9}^{M}(M')^{-a}dM'}{\int_{10^9}^{10^{12}}(M')^{-a}dM'} = \frac{M^{1-a} - (10^9)^{1-a}}{(10^{12})^{1-a} - (10^9)^{1-a}}\ ,\ \text{for } M \in (10^9, 10^{12})M_{\odot}$$

$$F^{-1}(y) = (y \cdot (10^{12})^{1-a} - y \cdot (10^9)^{1-a} + (10^9)^{1-a})^{\frac{1}{1-a}}$$

The samples are the values of function $F^{-1}(y)$ for y ∈ (0,1)

- The **second physical parameter** is the inner slope, a**,** of the halo mass function, while:

$$b = (1-a) \cdot \frac{N}{\left((10^{12})^{1-a} - (10^9)^{1-a}\right)V}$$

Constructing the Realizations of the Gaussian Fields

- We generate position space realization of a white noise field, φ_{ab}, with unit amplitude, on a 100x100 grid, i.e., a,b ∈ {0,..,99}
- We Fourier transform the white noise realization: $\varphi_{ab} \rightarrow \varphi_{k_a k_b}$, where $k_a,\, k_b \in \frac{2\pi}{N}$ {0,..,99}
- We want to multiply $\varphi_{k_a k_b}$ with $\sqrt{P(k)}$ to get $\delta_{k_a k_b}$
- Naive way: Calculate $P(k)$ at points $k = \sqrt{k_a^2 + k_b^2}$ $\Rightarrow$ leads to imaginary fields
- Alternatively: Calculate $P(k)$ at points $k = \sqrt{k_a'^2 + k_b'^2}$, where $k'_a,\, k'_b \in \frac{2\pi}{N}$ {0,..,50,−49, … , −1}
- $\delta_{k_a k_b} = \sqrt{P(k)}\,\varphi_{k_a k_b}$
- $\delta_{k_a k_b} \rightarrow \delta_{ab}$

Marginal Neural Ratio Estimation (MNRE)

Our goal is to train a NN to estimate marginal posteriors for parameters of interest.

Starting point: for any pair of observations $\boldsymbol{x}$ and model parameter $\boldsymbol{\vartheta}$, the goal is to estimate the probability that this pair belongs to one of the following classes:

H_0: Data $\boldsymbol{x}$ correspond to model parameters $\boldsymbol{\vartheta}$: $(\boldsymbol{x},\boldsymbol{\vartheta}) \sim p(\boldsymbol{x},\boldsymbol{\vartheta}) = p(\boldsymbol{x}|\boldsymbol{\vartheta})\, p(\boldsymbol{\vartheta})$

H_1: Data $\boldsymbol{x}$ unrelated to model parameters $\boldsymbol{\vartheta}$: $(\boldsymbol{x},\boldsymbol{\vartheta}) \sim p(\boldsymbol{x})\, p(\boldsymbol{\vartheta})$

Credit: C. Weniger

Loss function

- **Strategy:** We train a neural network $d_\varphi(\boldsymbol{x},\boldsymbol{\vartheta}) \in [0,1]$ as binary classifier to estimate the probability of hypothesis H_0 or H_1. The Network output can be interpreted, for a given input pair $\boldsymbol{x}$ and $\boldsymbol{\vartheta}$, as probability that H_0 is true.

    - H_0 is true: $d_\varphi(\boldsymbol{x},\boldsymbol{\vartheta}) \simeq 1$

    - H_1 is true: $d_\varphi(\boldsymbol{x},\boldsymbol{\vartheta}) \simeq 0$

- The corresponding loss function is the so-called “binary cross-entropy”:

$$L[d(\boldsymbol{x},\boldsymbol{\vartheta})] = -\int d\boldsymbol{x}d\boldsymbol{\theta}\,[p(\boldsymbol{x},\boldsymbol{\vartheta})\ln(d(\mathbf{x},\boldsymbol{\vartheta})) + p(\boldsymbol{x})p(\boldsymbol{\vartheta})\ln(1-d(\boldsymbol{x},\boldsymbol{\vartheta}))]$$

- Minimizing that function w.r.t the network parameters φ yields:

$$d_\varphi(\boldsymbol{x},\boldsymbol{\vartheta}) \approx \frac{p(\boldsymbol{x},\boldsymbol{\vartheta})}{p(\boldsymbol{x},\boldsymbol{\vartheta}) + p(\boldsymbol{x})p(\boldsymbol{\vartheta})}$$

Credit: C. Weniger

Likelihood-to-evidence ratio

Training binary classification networks yield true Bayesian posterior estimates!

- With a bit of math one can show that:

$$r(\boldsymbol{x}, \boldsymbol{\vartheta}) \equiv \frac{d_\varphi(\boldsymbol{x},\boldsymbol{\vartheta})}{d_\varphi(\boldsymbol{x},\boldsymbol{\vartheta})-1} \approx \frac{p(\boldsymbol{x}|\boldsymbol{\vartheta})}{p(\boldsymbol{x})} = \frac{p(\boldsymbol{\vartheta}|\boldsymbol{x})}{p(\boldsymbol{\vartheta})}$$

- Once we have trained the network $d_\varphi(\boldsymbol{x}, \boldsymbol{\vartheta})$, we can **estimate the posterior**:

$$p(\boldsymbol{\vartheta}|\boldsymbol{x}) \simeq r(\boldsymbol{x}, \boldsymbol{\vartheta}) p(\boldsymbol{\vartheta})$$

- Swyft: a flexible and powerful tool for efficient marginal posterior estimation using NN, designed by B. Miller et al. (2020, 2021).

Credit: C. Weniger

Marginal Neural Ratio Estimation (MNRE)

arXiv: 2107.01214

Estimates posteriors through a binary classification problem:

"Given a (parameter: $\boldsymbol{\vartheta}$, image: $\boldsymbol{x}$) pair, is the image, $\boldsymbol{x}$, actually generated by the parameter $\boldsymbol{\vartheta}$?"

Class 1:	**Class 2:**
$(\boldsymbol{x}, \boldsymbol{\vartheta}) \sim p(\boldsymbol{x}\|\boldsymbol{\vartheta})\, p(\boldsymbol{\vartheta})$	$(\boldsymbol{x}, \boldsymbol{\vartheta}') \sim p(\boldsymbol{x}) p(\boldsymbol{\vartheta})$

Train a classifier with mock data to directly estimate:

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

Once we have trained the network, we can **estimate the posterior**:

$$p(\boldsymbol{\vartheta}|\vec{\boldsymbol{x}}) = r(\boldsymbol{x}, \boldsymbol{\vartheta}) p(\boldsymbol{\vartheta})$$