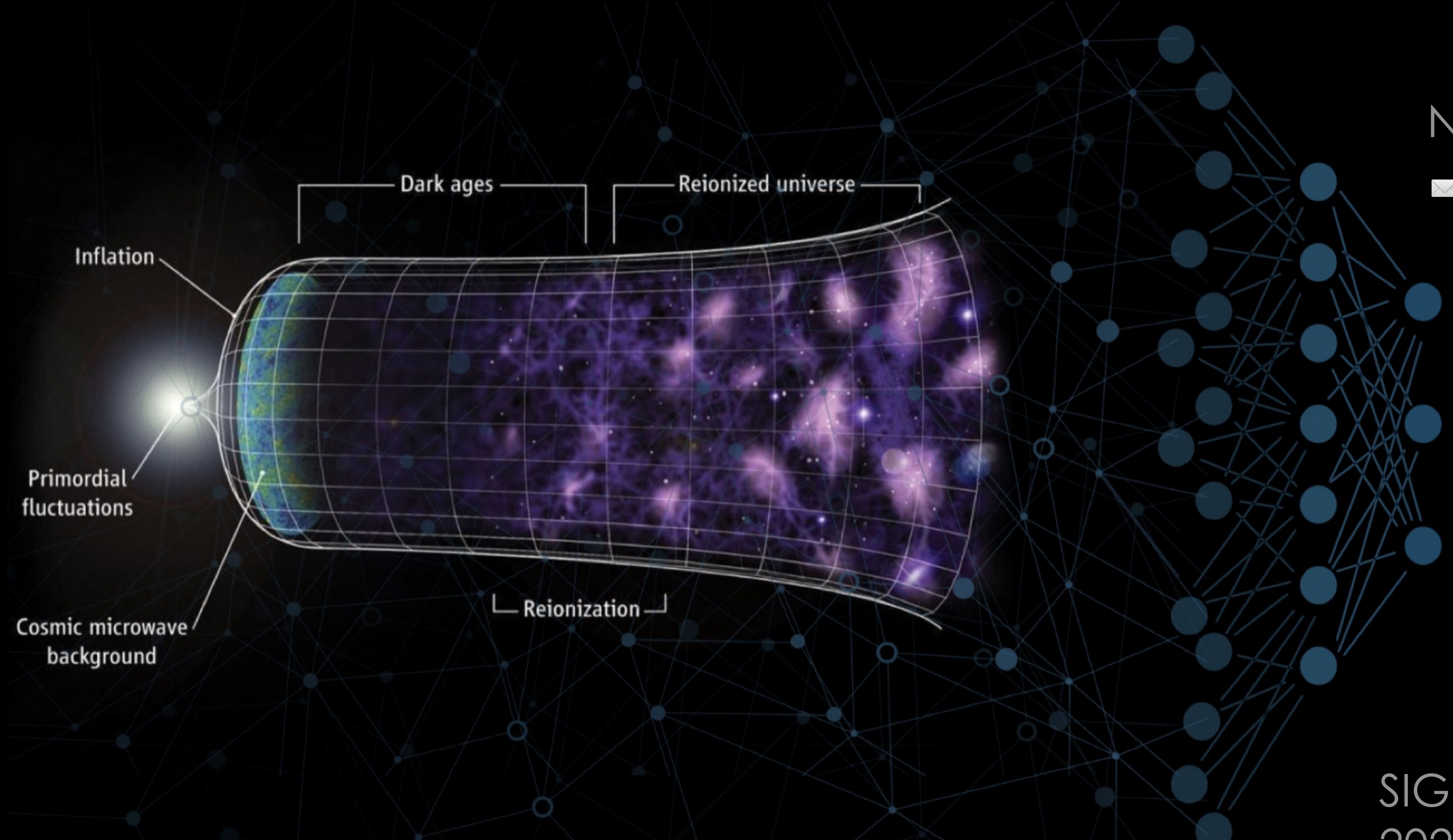
Mining the Universe

Machine Learning in Cosmology



Credits: C. FAUCHER-GIGUÈRE, A. LIDZ, AND L. HERNQUIST, SCIENCE 319, 5859 (47) modified by Nicoletta Krachmalnicoff

Nicoletta Krachmalnicoff
nkrach@sissa.it
SISSA

SIGRAV international School 2022 on Cosmology

Outline:

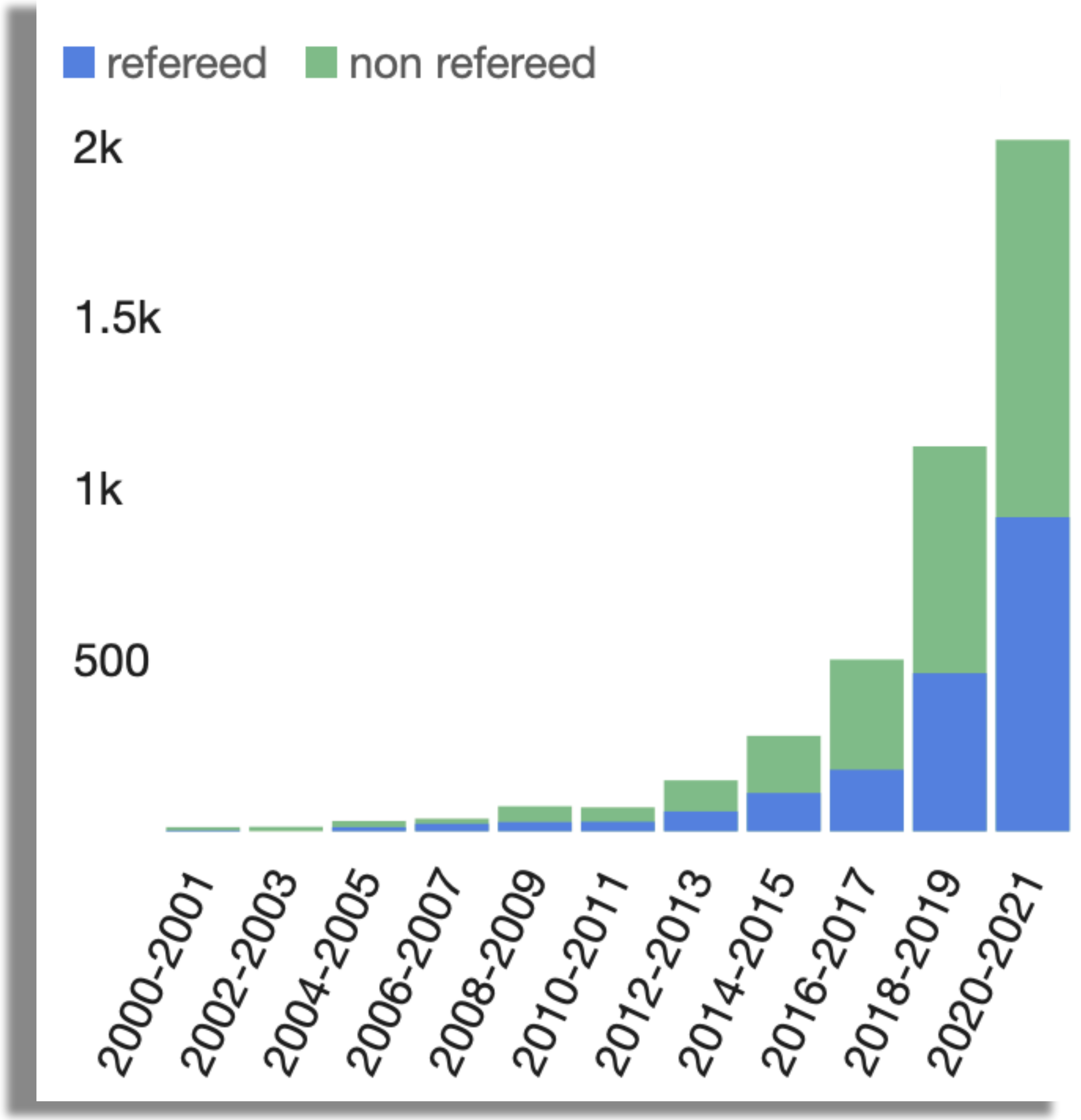
i. Intro: numbers about ML in cosmology
ii. Open questions in cosmology
iii. NN applications on LSS simulations
iv. NN applications on CMB

- Basics of Neural Networks
- Convolutional Neural Networks
- Generative adversarial Networks

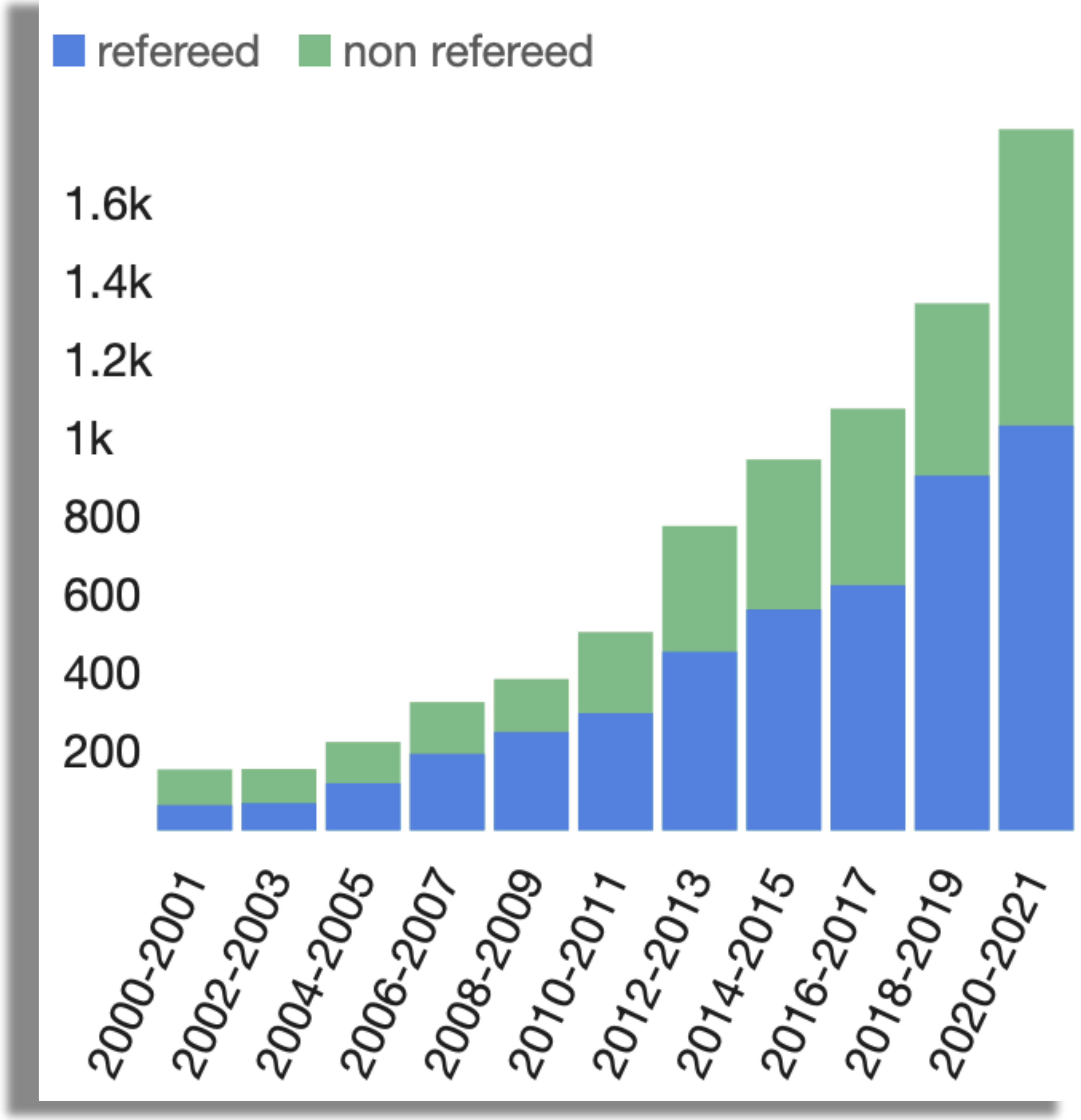
Some numbers

Machine learning (**ML**) is the study of computer algorithms that can improve automatically through experience and by the use of data.

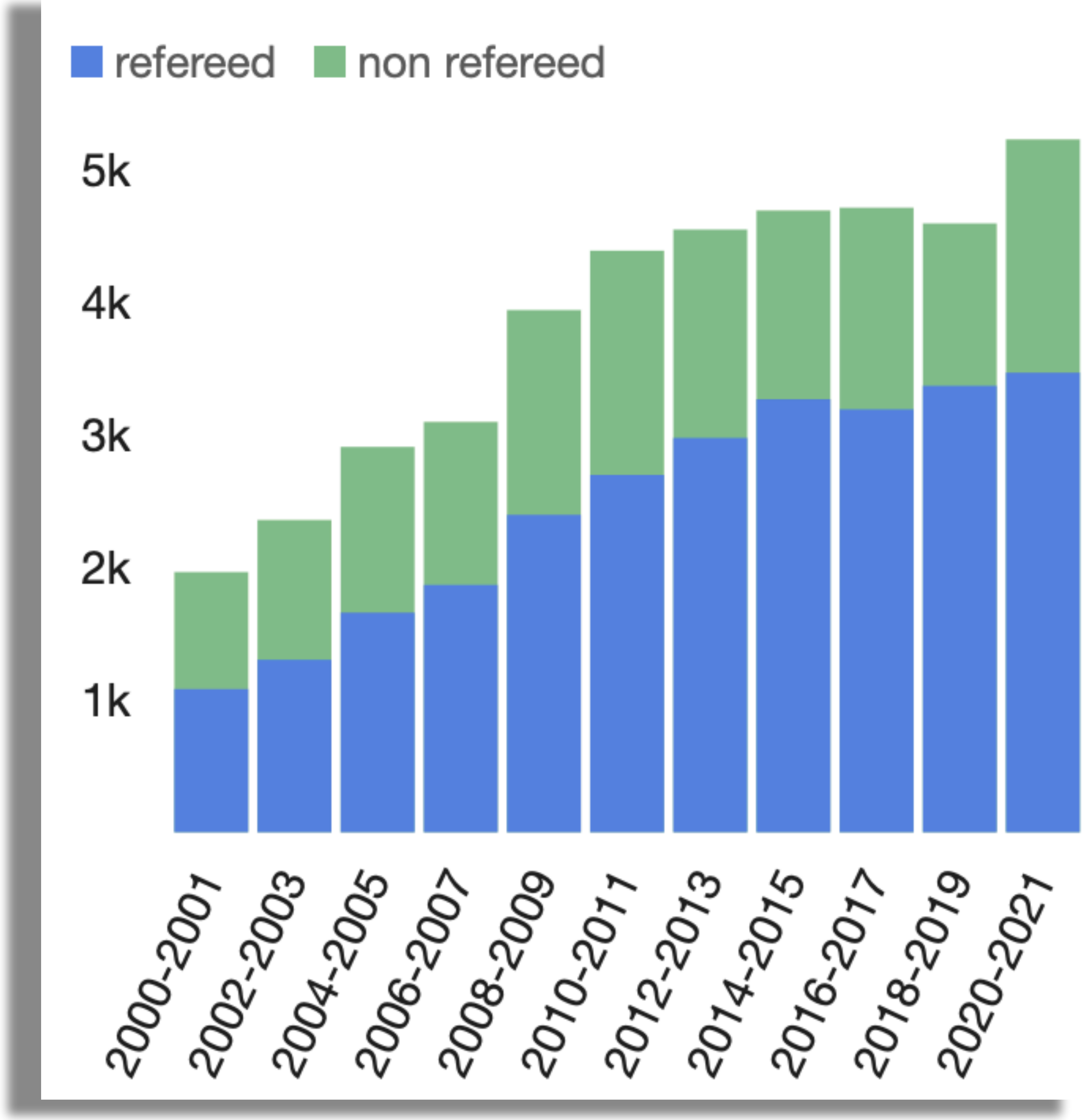
"Neural Networks"



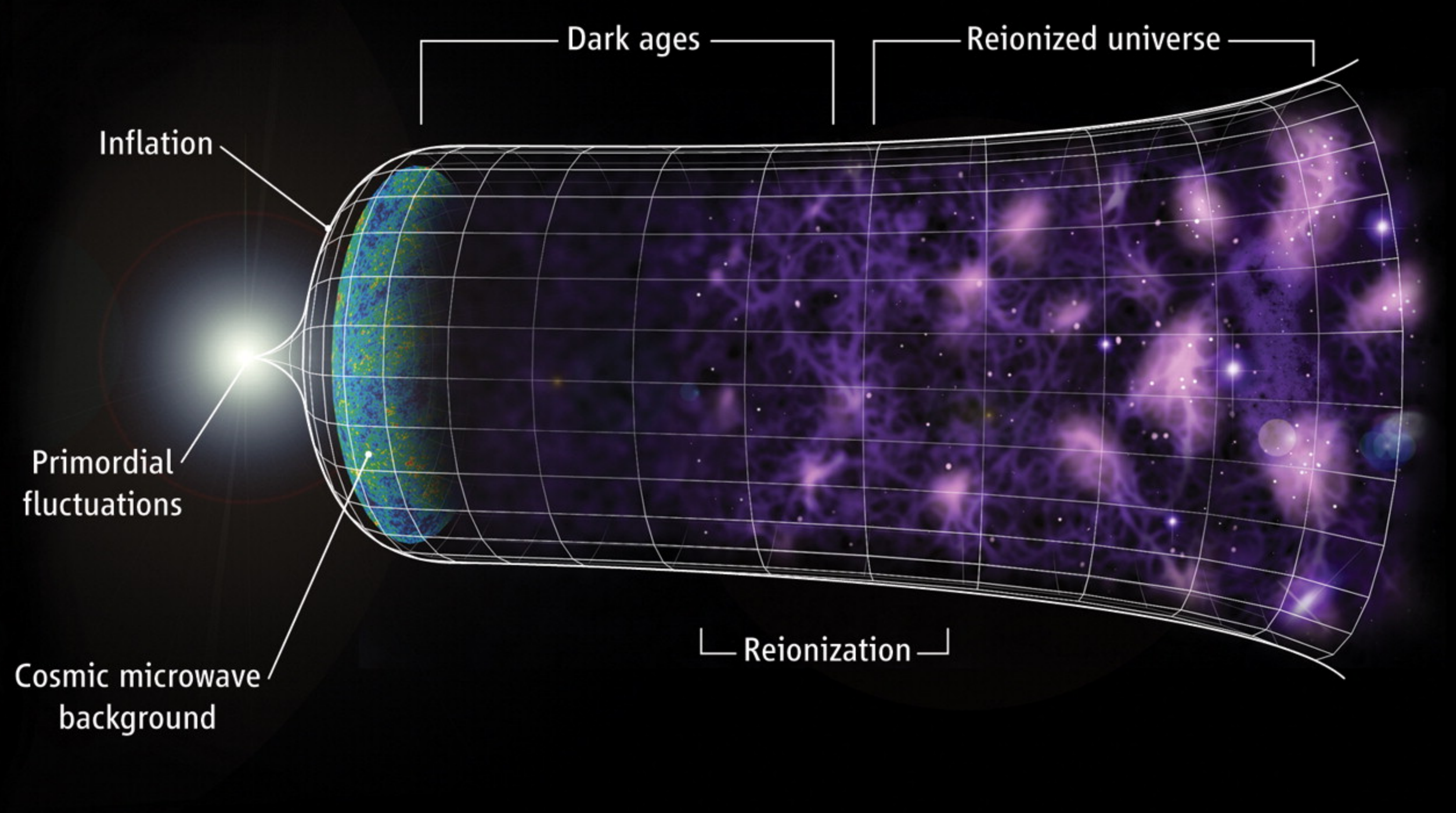
"Bayesian"



"Dark Matter"



source: NASA/ADS



Many open questions:

- What is Dark Matter?
- What is the nature of Dark Energy?
- What is the correct theory of Inflation?
- Which are the neutrino masses?
- ….

CMB:

"simple", almost perfectly Gaussian, signal
…but faint and highly contaminated
(foregrounds and instrumental systematics)

Large Scale Structure:

Complex signal, involving highly non linear physical process

CMB experiments

Early Universe - **faint signal**

SIMONS OBSERVATORY

Simons Observatory
2023

CMB-S4
Next Generation CMB Experiment

~ 2028

LiteBIRD

LiteBIRD
~ 2029

Galaxy surveys

Large Scale Structure - **complex signal**

DARK ENERGY SPECTROSCOPIC INSTRUMENT

DESI
ongoing

VERA C. RUBIN OBSERVATORY

~ 2023

euclid

~ 2025

How to fully exploit data?

Are current methodologies sufficient, given the amount of data, the signal complexity and the precision we want to achieve?

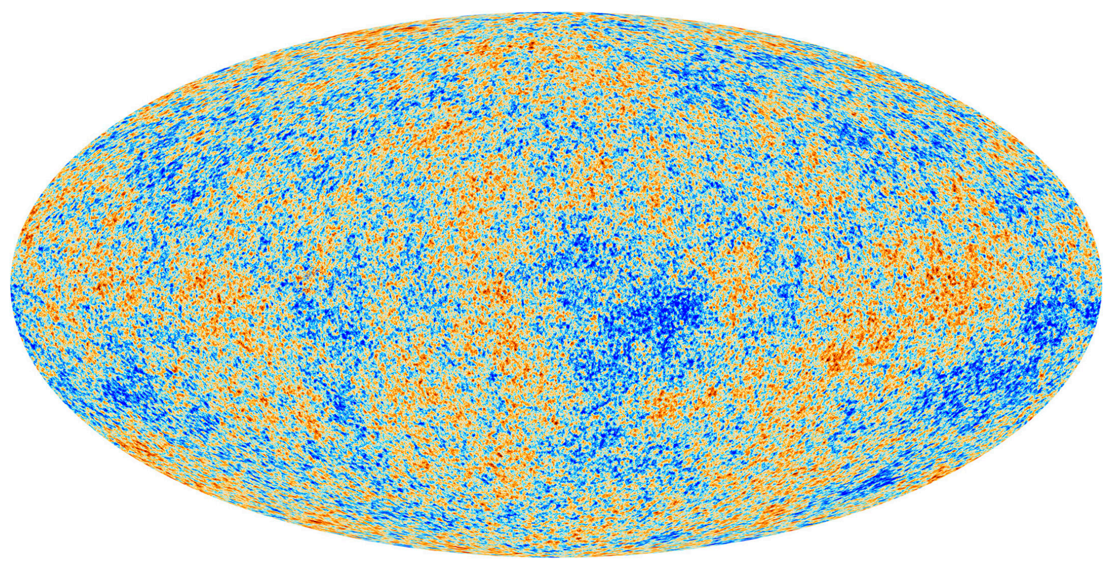
- ➡ The game we are playing in cosmology is to **find the value of few cosmological parameters**, with the highest possible precision

- ➡ Parameter estimation is a **huge data compression**: from many TB of data to few numbers

- ➡ To do that we typically use **summary statistics** computed from data and compared with **theoretical expectations**

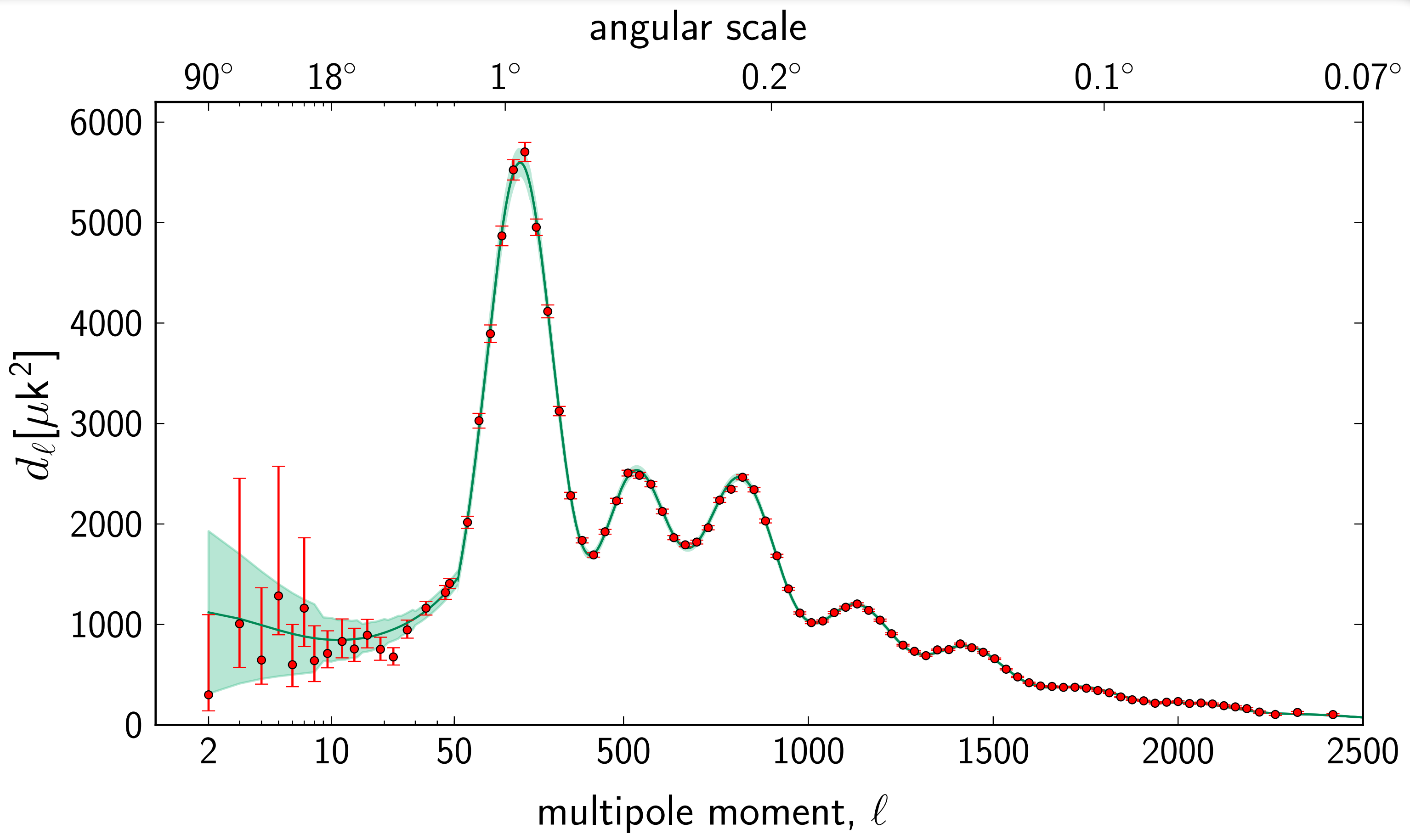
parameters from CMB:

Calibration
Map Making
Component separation



clean CMB map

summary statistic

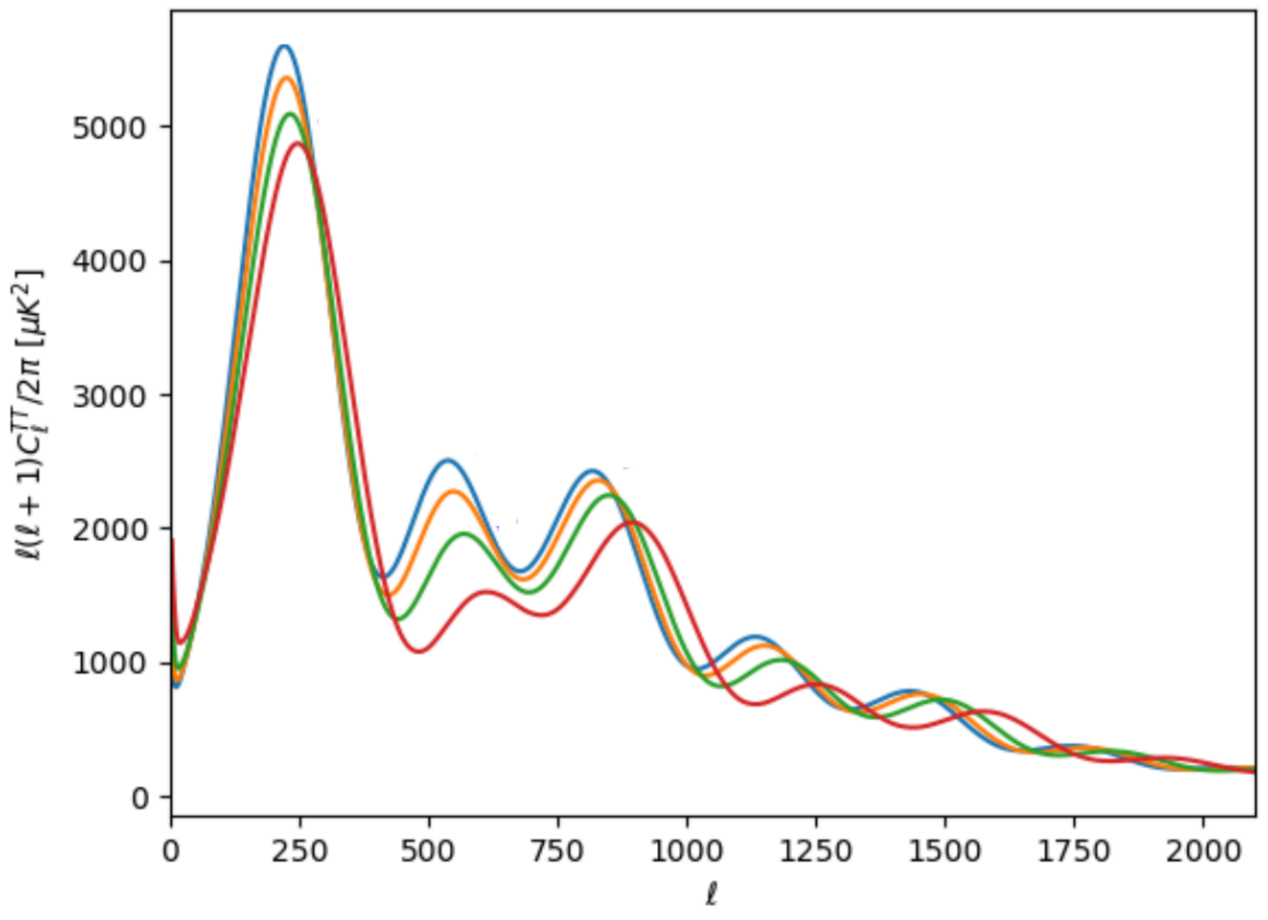


Power spectra

Inference

$$P(H\,|\,\bar{\theta}) \propto P(\bar{\theta}\,|\,H)P(\bar{\theta})$$

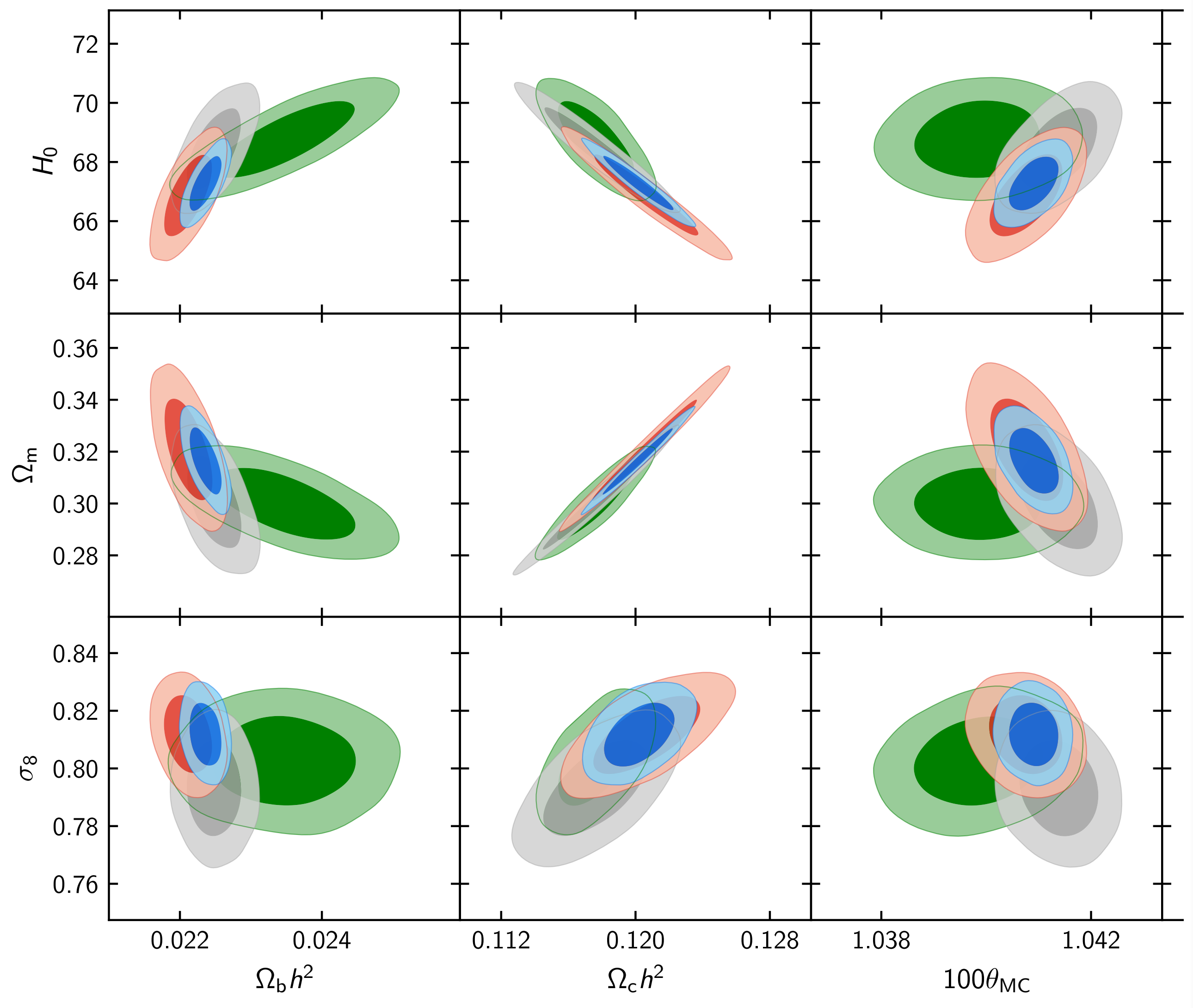
Theoretical prediction



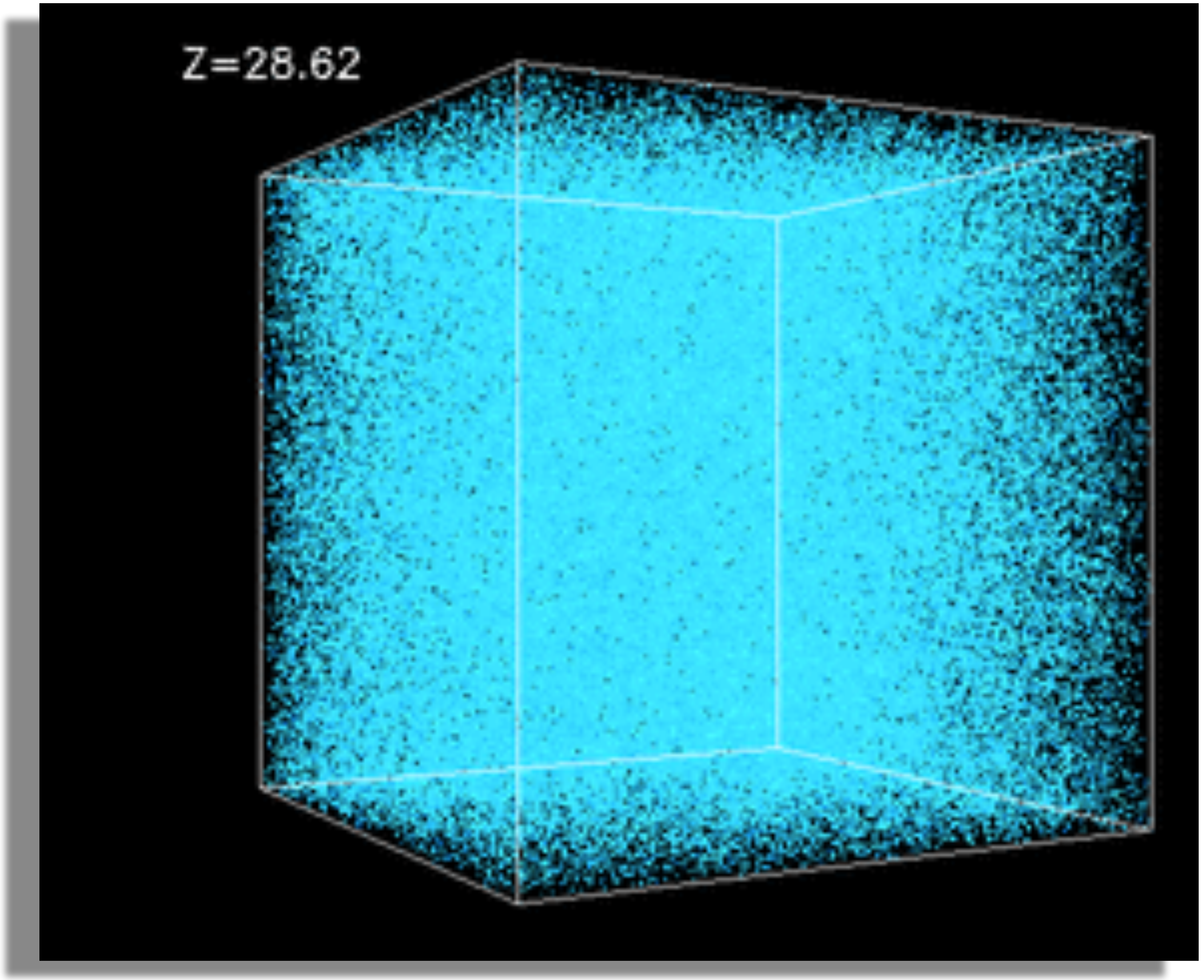
Likelihood

$$P(\bar{\theta}\,|\,H)$$

Parameters



Parameters from LSS:



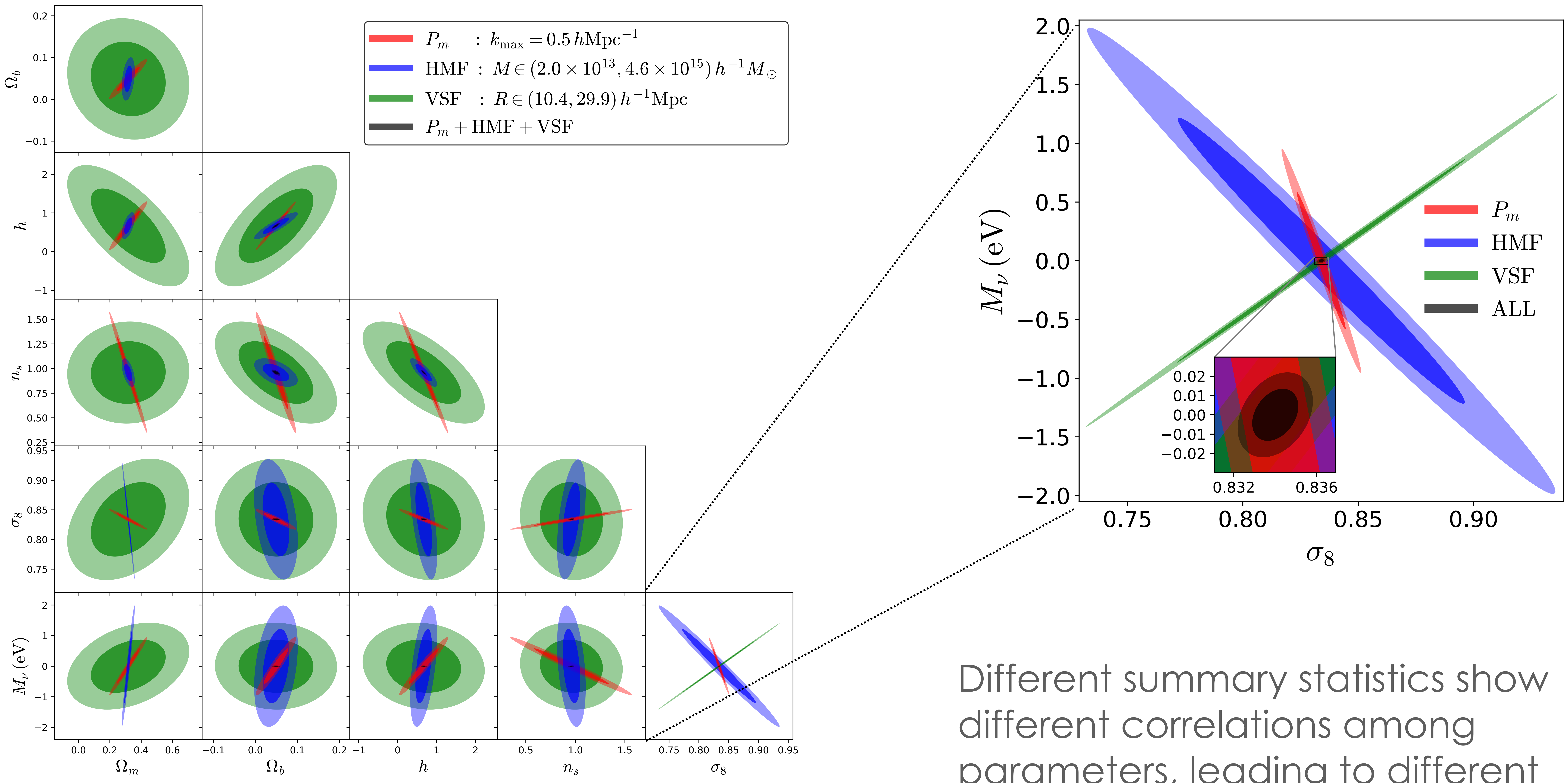
Even if the early Universe was a Gaussian random field, non-linear gravitational evolution leads to a non-Gaussian density field on small scales and at low redshift

The main challenge is not in the amount and quality of data (as for CMB) but in the signal complexity!

1. What is the optimal summary statistic?
2. How to efficiently compute numerical simulations for theoretical prediction?
3. How to marginalize over unknown physical processes?

Just one example….

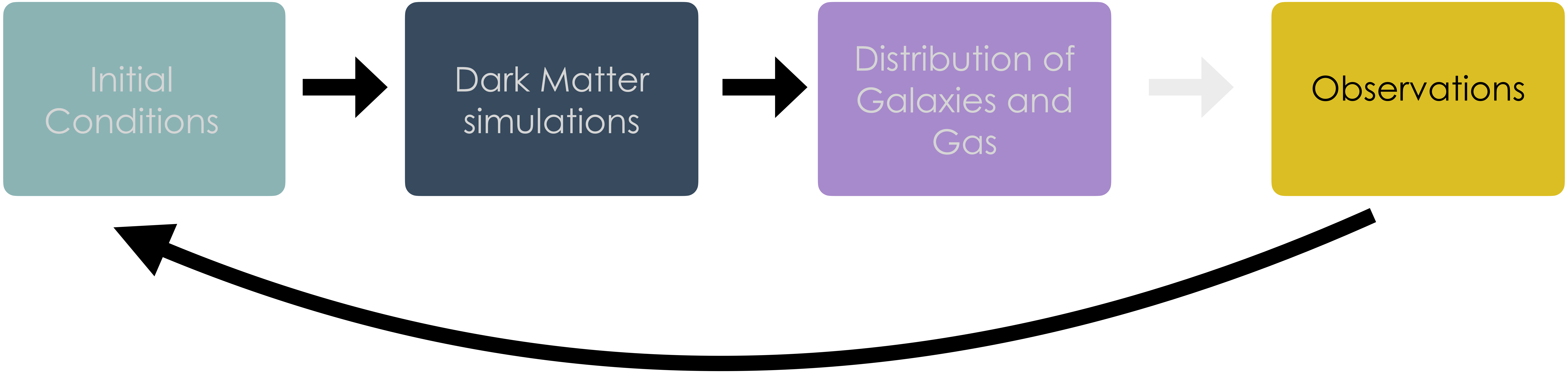
Bayer, A.E. et al., ApJ 2021 (arXiv:2102.05049)



Different summary statistics show different correlations among parameters, leading to different constraints

Credits: D. Spergel, ML x Cosmo group at CCA

Mapping from initial conditions to simulations



…and from observations to initial conditions

Basics of Neural Networks

Basics of NNs

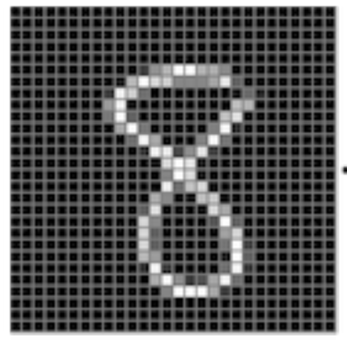
$$y = f(\mathbf{x})$$

- The goal of a **feed-forward Neural Network** is to find a good enough approximation of the function f that maps inputs into outputs

- The Neural Network defines a mapping $f^* = f^*(x;\theta)$ and finds the value of the parameters θ that results in the best approximation $f^* \sim f$

Basics of NNs

$f($  $) = 8$ $\qquad$ $f($  $) = 2$



28 x 28
784 pixels

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	4	62	146	182	254	254	181	176	139	15	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	34	186	253	217	208	136	136	136	166	232	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	61	242	208	111	3	0	0	0	0	0	18	32	107	43	0	0	0	0	0	0
0	0	0	0	0	0	0	0	156	242	23	0	0	0	0	0	0	0	13	191	181	6	0	0	0	0	0	0
0	0	0	0	0	0	0	0	121	255	98	3	0	0	0	0	0	8	194	225	12	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	169	253	120	3	0	0	0	0	128	247	51	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	3	111	244	169	19	0	14	131	249	117	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	59	241	235	72	142	229	66	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	25	218	254	231	36	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	133	253	221	33	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	19	237	111	196	217	19	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	138	0	23	193	204	18	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	96	224	0	0	0	25	218	169	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	138	0	0	0	0	86	253	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	3	162	214	11	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	0	118	253	68	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	185	157	0	0	0	0	0	40	254	98	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	50	244	61	0	0	0	0	112	244	58	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	251	142	59	83	167	244	111	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	6	133	253	253	253	169	61	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

$$f : \mathbb{R}^{n \times m} \to \mathbb{N}$$

Basics of NNs

- Neural networks can, in principle, be used every time there is a unique relation between a input and output
- very powerful tool, especially when this relation is unknown
- it must exist a set of data “large enough” for which the output associated at each input is known (training set)

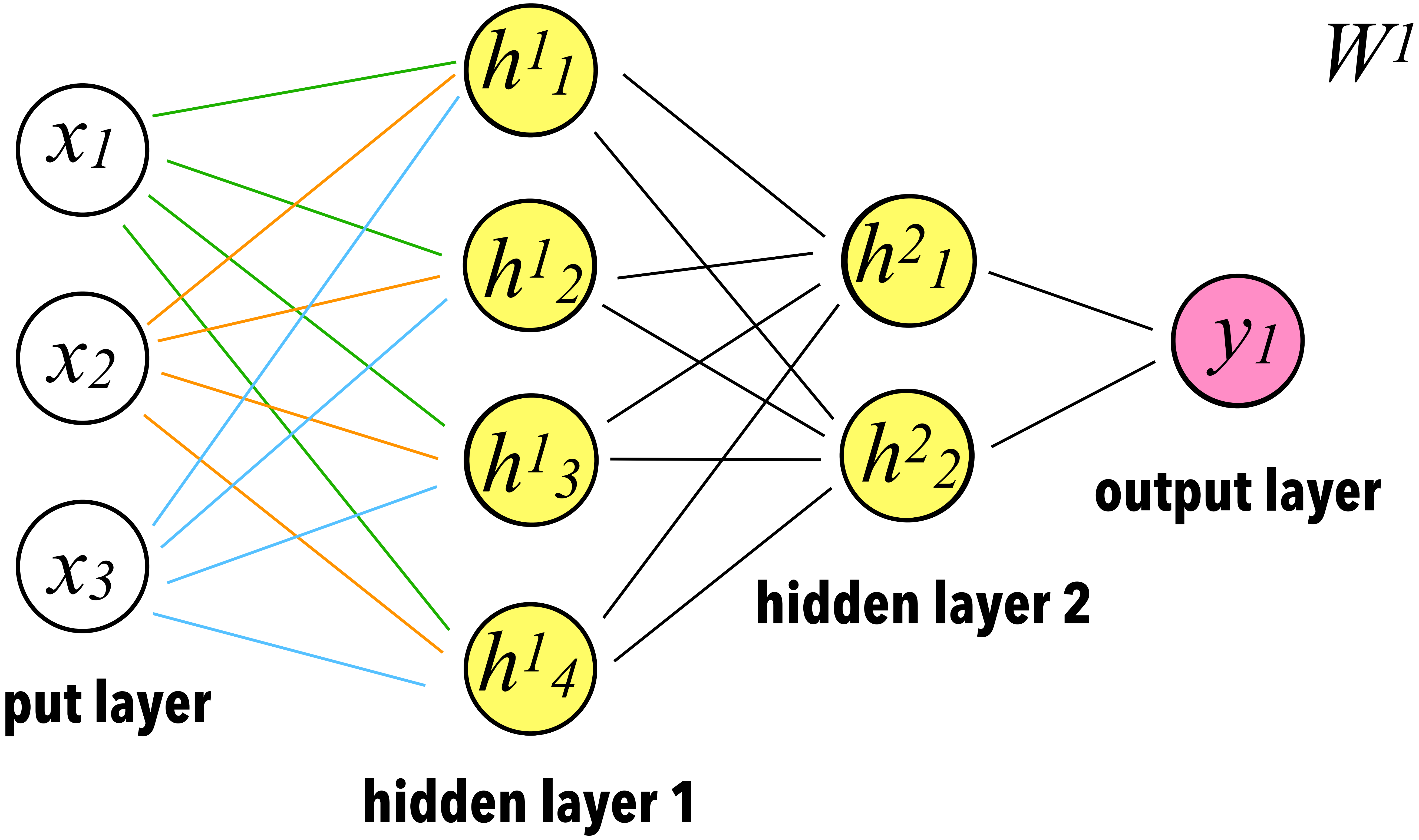
Supervised Learning

Basics of NNs

Q: How can a NN approximate very complex unknown functions?

A: By recursively apply **non-linear** activation functions to a linear combination of input elements

Fully connected NN



$$W^1 = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \end{bmatrix} \qquad b^1 = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

$$\tilde{h}^{[1]} = W^{[1]T} \cdot x + b^{[1]}$$

$$h^{[1]} = \sigma(\tilde{h}^{[1]})$$

$$\tilde{h}^{[2]} = W^{[2]T} \cdot h^{[1]} + b^{[2]}$$

$$h^{[2]} = \sigma(\tilde{h}^{[2]})$$

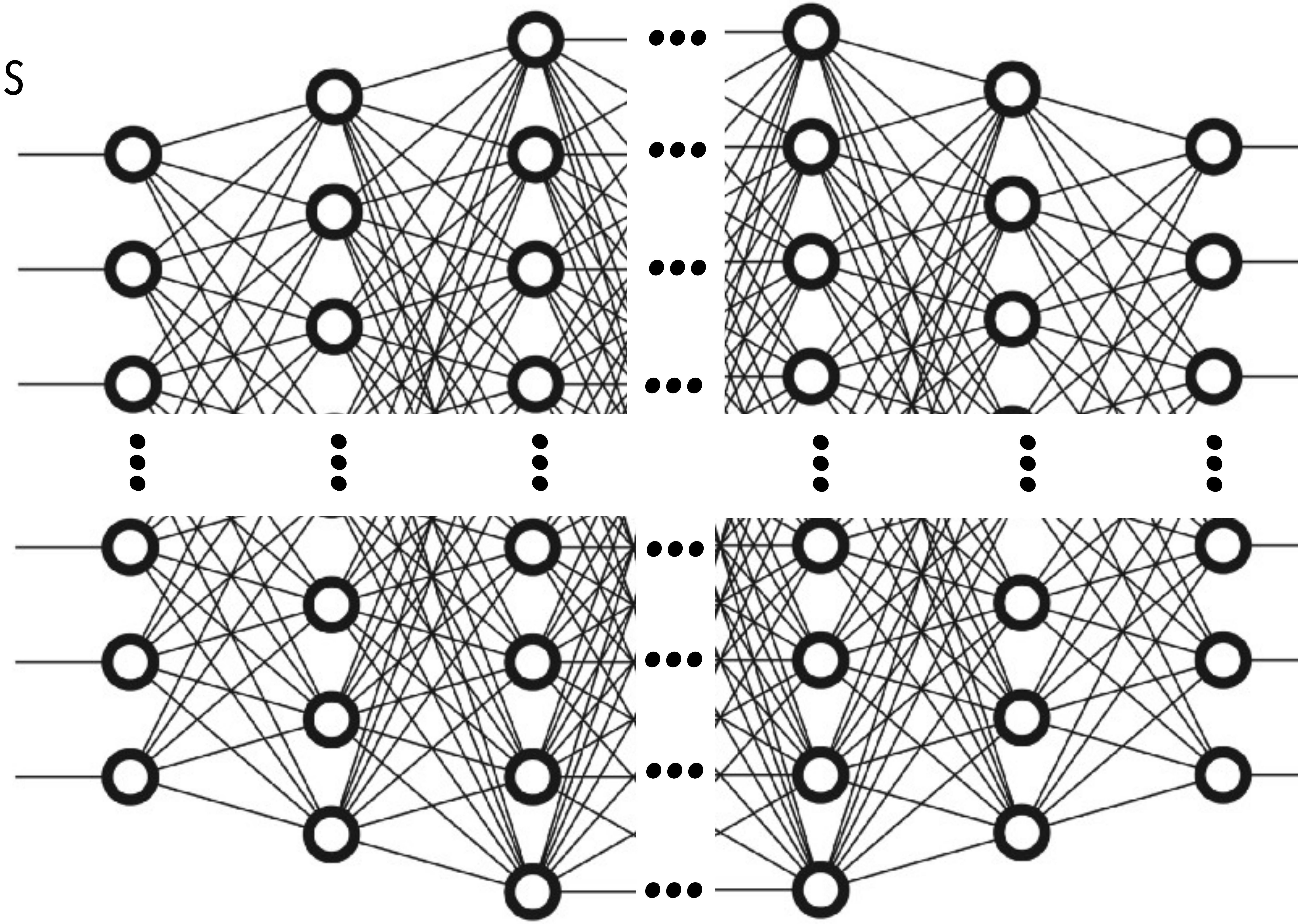
$$\tilde{y} = W^{[3]T} \cdot h^{[2]} + b^{[3]}$$

$$y = \sigma^*(\tilde{y})$$

- Each line represents a weight w
- In each neuron a linear combination of the inputs is computed
- The result is then activated with a non-linearity and become one of the inputs of the following layer

Fully connected NN

- This type of NN are called **dense** or **fully connected**
- very deep NNs can have thousands of layers
- and $O(10^7)$ parameters
- The number of layers and neurons in each layer define the architecture of the NN and are called **hyperparameters**



Loss function

- Through the feed-forward propagation the NN produces the output (values of the neurons in the output layer)
- During training the **output is compared with the ground truth**
- This is done by computing a **loss function**

$$\mathcal{L}(\mathbf{y^i}, \hat{\mathbf{y}}^\mathbf{i})$$

NN output   Ground truth

"Distance" between true output and NN output

Cost function

- The **cost function** is the average of the loss over the training set

$$\mathscr{J} = \frac{1}{N}\sum_{i=0}^{N} \mathscr{L}(\mathbf{y^i}, \hat{\mathbf{y}}^{\mathbf{i}})$$

- it is a function of all the NN's parameters (weights and biases):

$$\mathscr{J} = \mathscr{J}(\mathbf{w}, \mathbf{b})$$

The goal of training is to find the parameters $\mathbf{w}$ and $\mathbf{b}$ that minimize $\mathscr{J}$

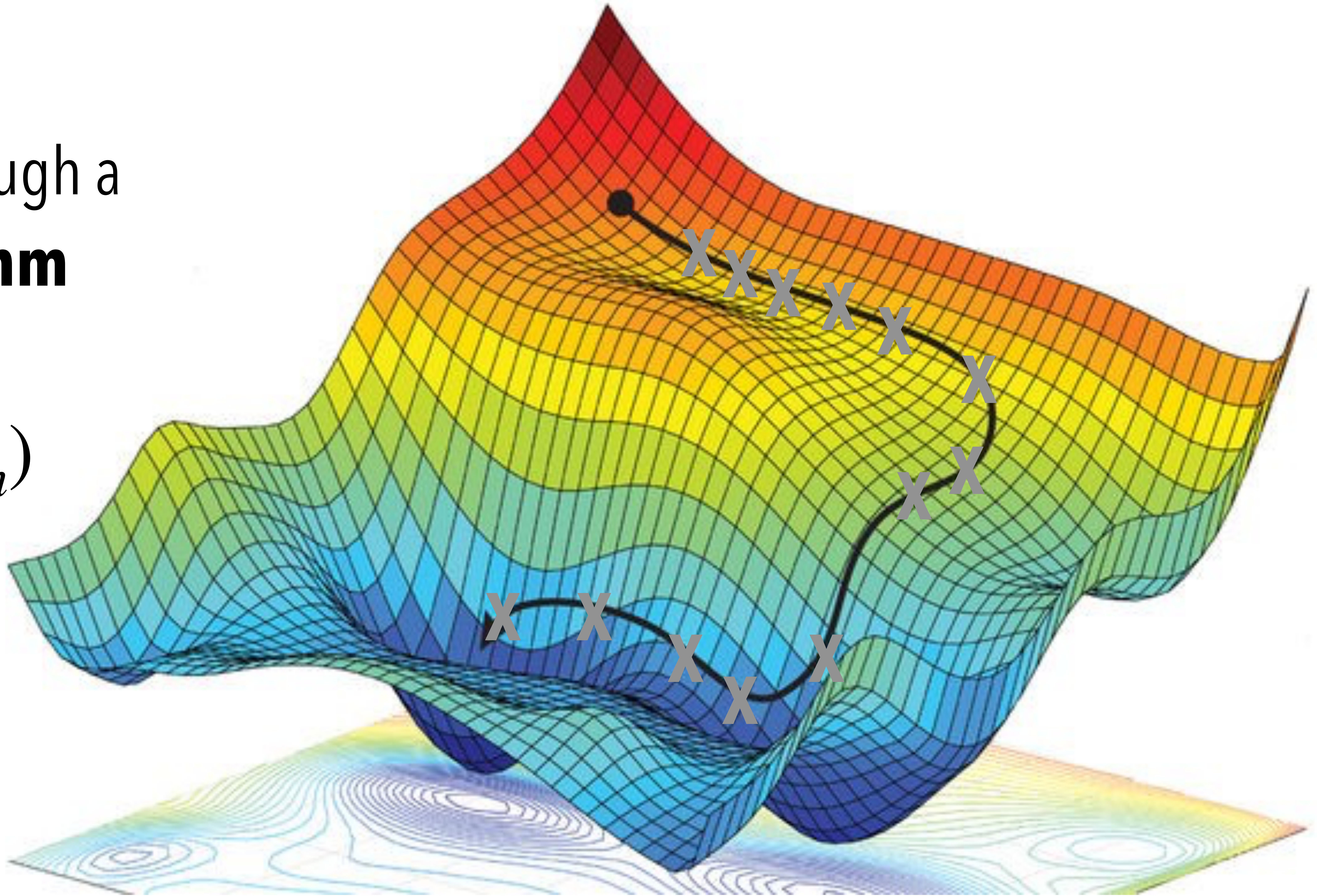
Gradient descent

We need to find the minimum of the cost function $\mathscr{J} = \mathscr{J}(\mathbf{w}, \mathbf{b}) = \mathscr{J}(\boldsymbol{\Theta})$

Which could be a very complicated function

The minimum is achieved through a **gradient descent algorithm**

$$\boldsymbol{\Theta}_{n+1} = \boldsymbol{\Theta}_n - \alpha \nabla \mathscr{J}(\boldsymbol{\Theta}_n)$$



Gradient descent

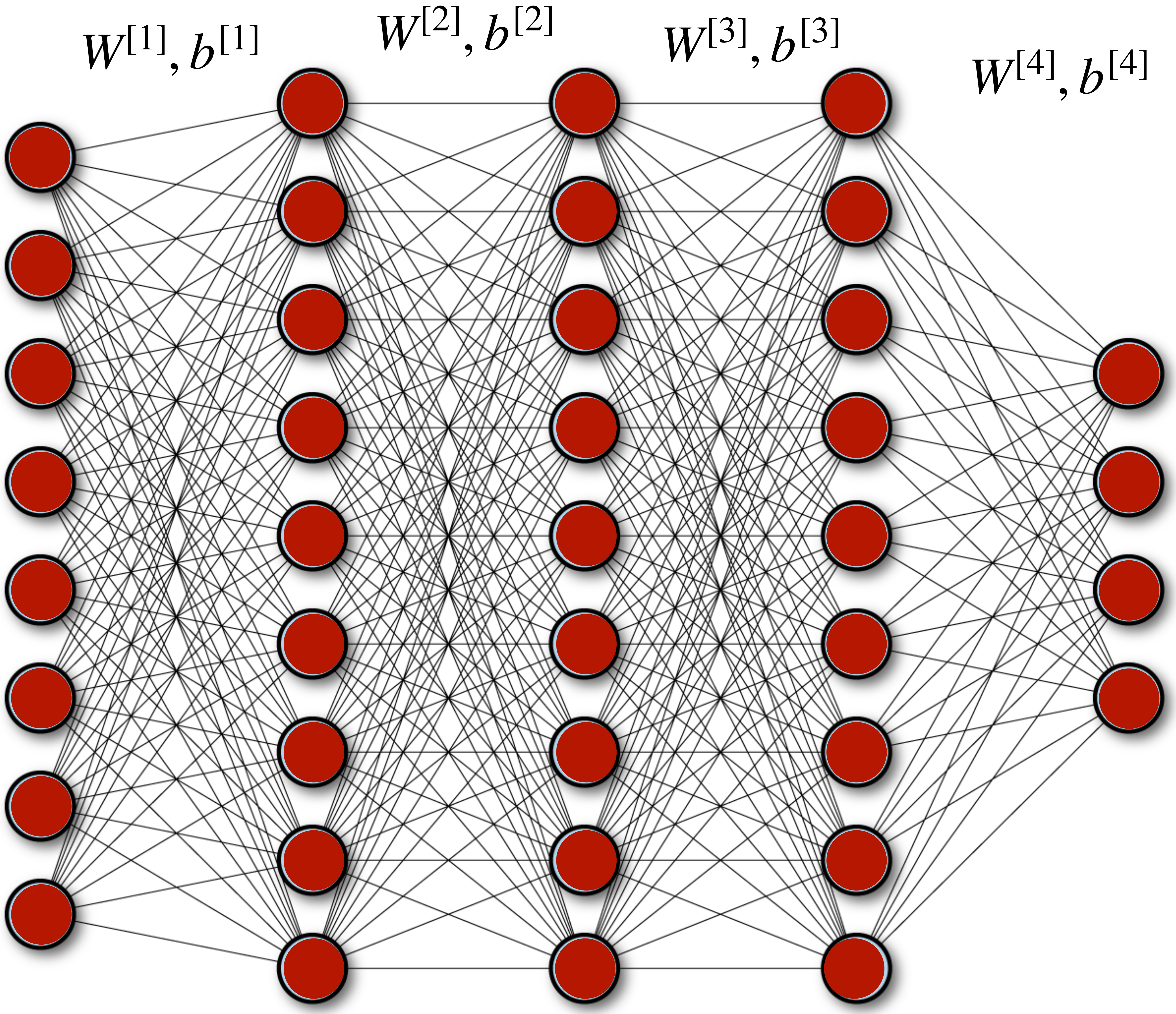
- at each iteration the parameters are updated as:

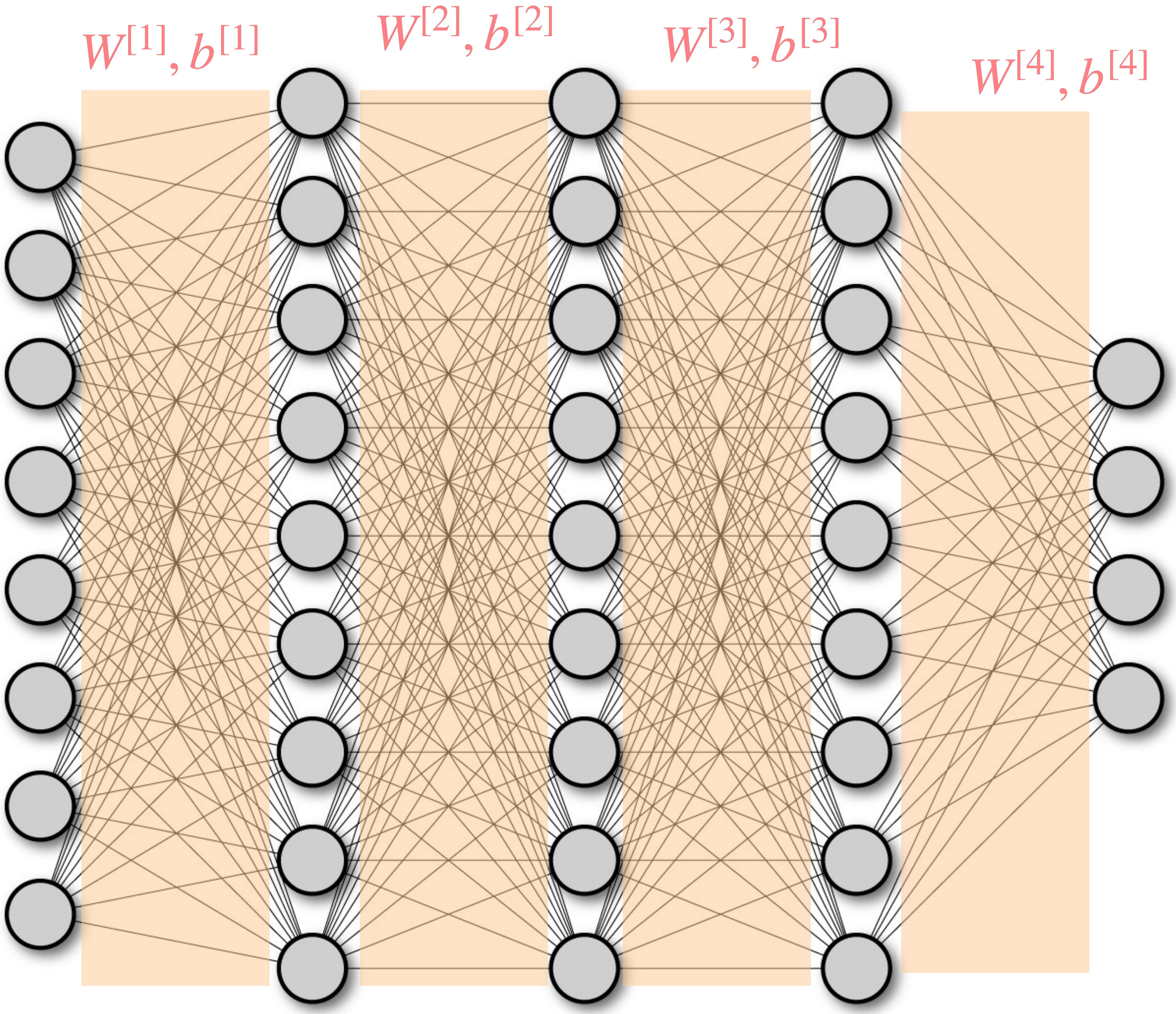
$$w := w - \alpha \frac{\partial \mathcal{J}}{\partial w} \qquad\qquad b := b - \alpha \frac{\partial \mathcal{J}}{\partial b}$$

- α is the **learning rate** and is an **hyper-parameter** of the NN

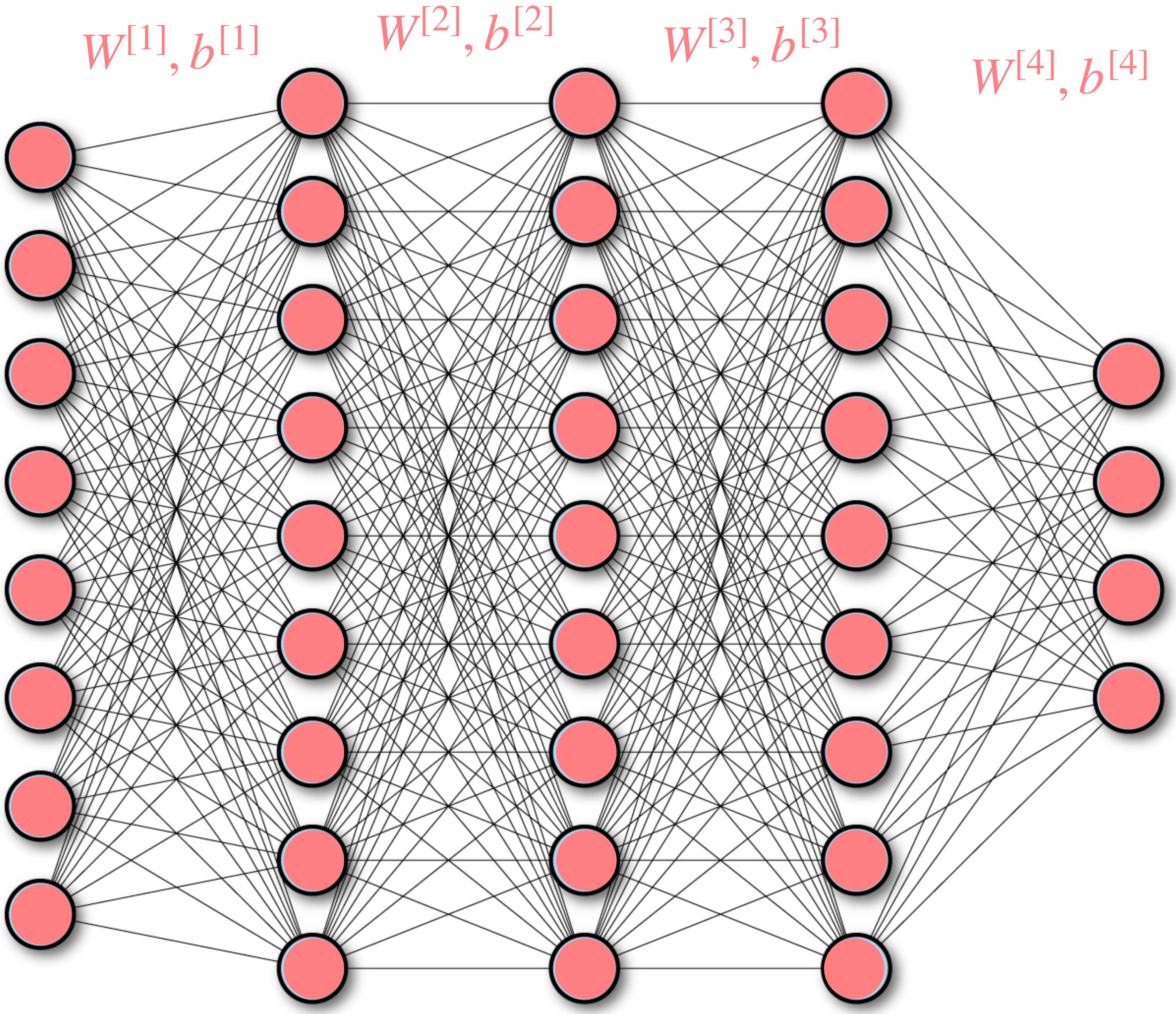
(hyperparameter are set before training the network and are not optimize during training)

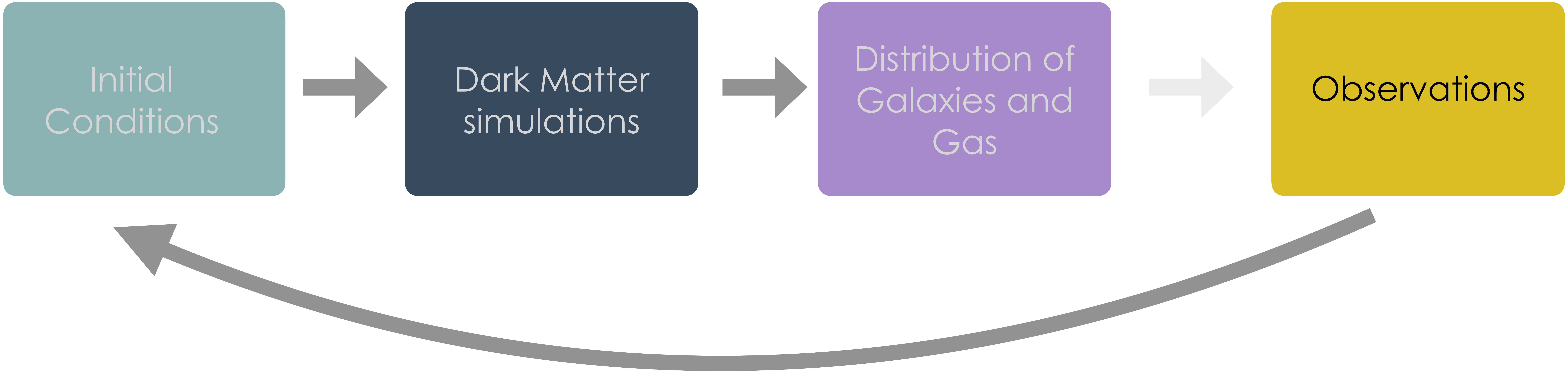
Feedforward & Backpropagation




$W^{[1]}, b^{[1]}$
$W^{[2]}, b^{[2]}$
$W^{[3]}, b^{[3]}$
$W^{[4]}, b^{[4]}$

Feedforward & Backpropagation

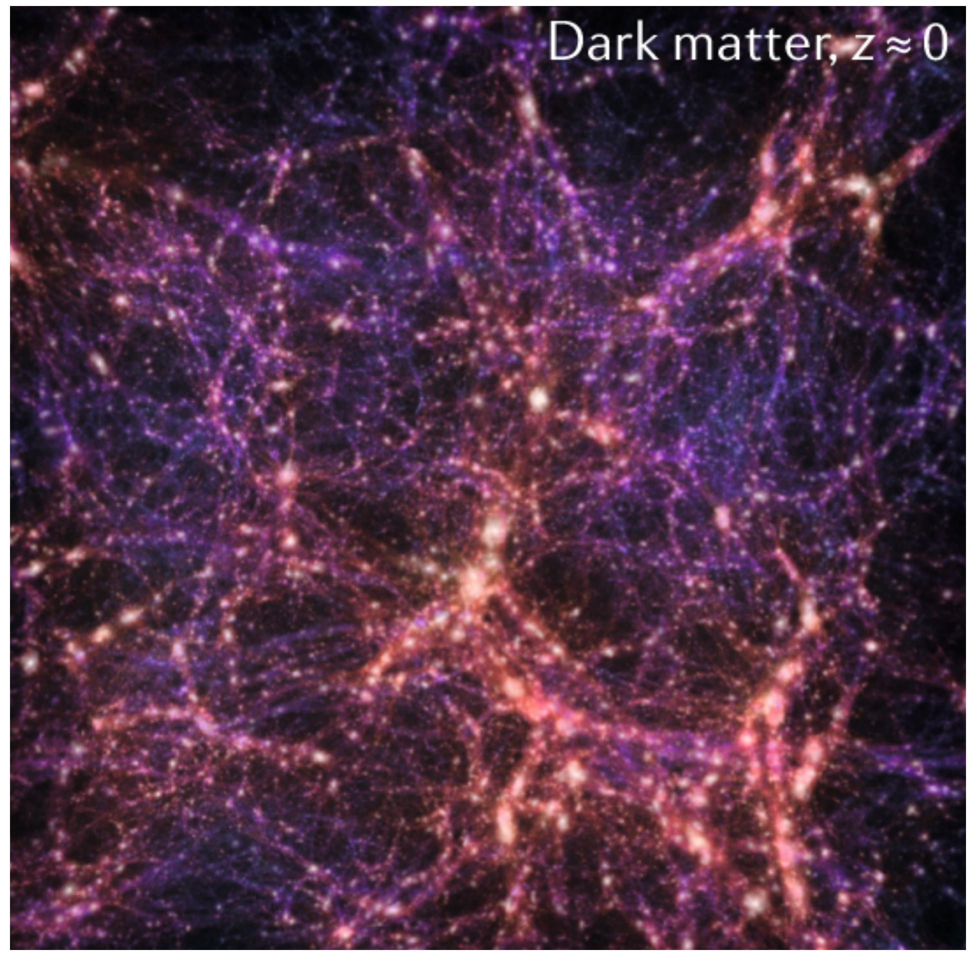


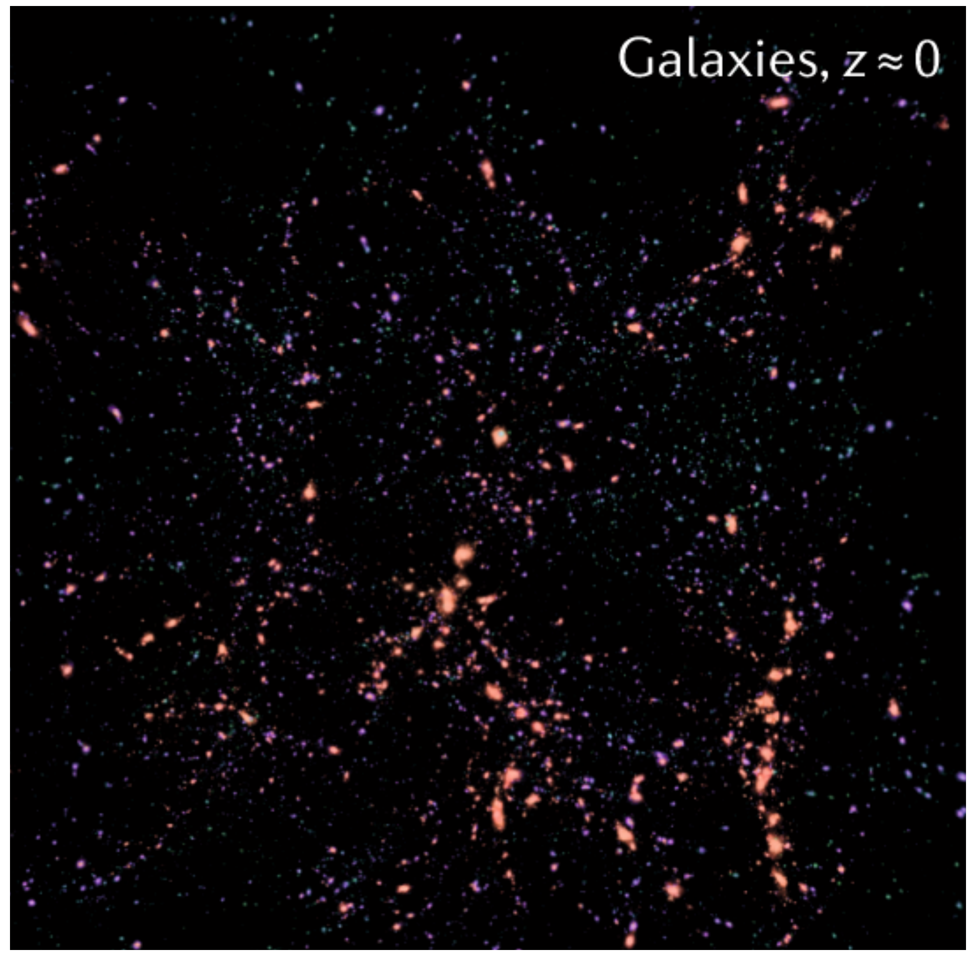


1. How to efficiently compute numerical simulations for theoretical prediction?
2. Which is the optimal summary statistic?
3. How to marginalize over unknown physical processes?

1. **How to efficiently compute numerical simulations for theoretical prediction?**
2. Which is the optimal summary statistic?
3. How to marginalize over unknown physical processes?

From Dark Matter to Galaxies





Gravity only N-body simulations evolving position and velocity of massive particles over time

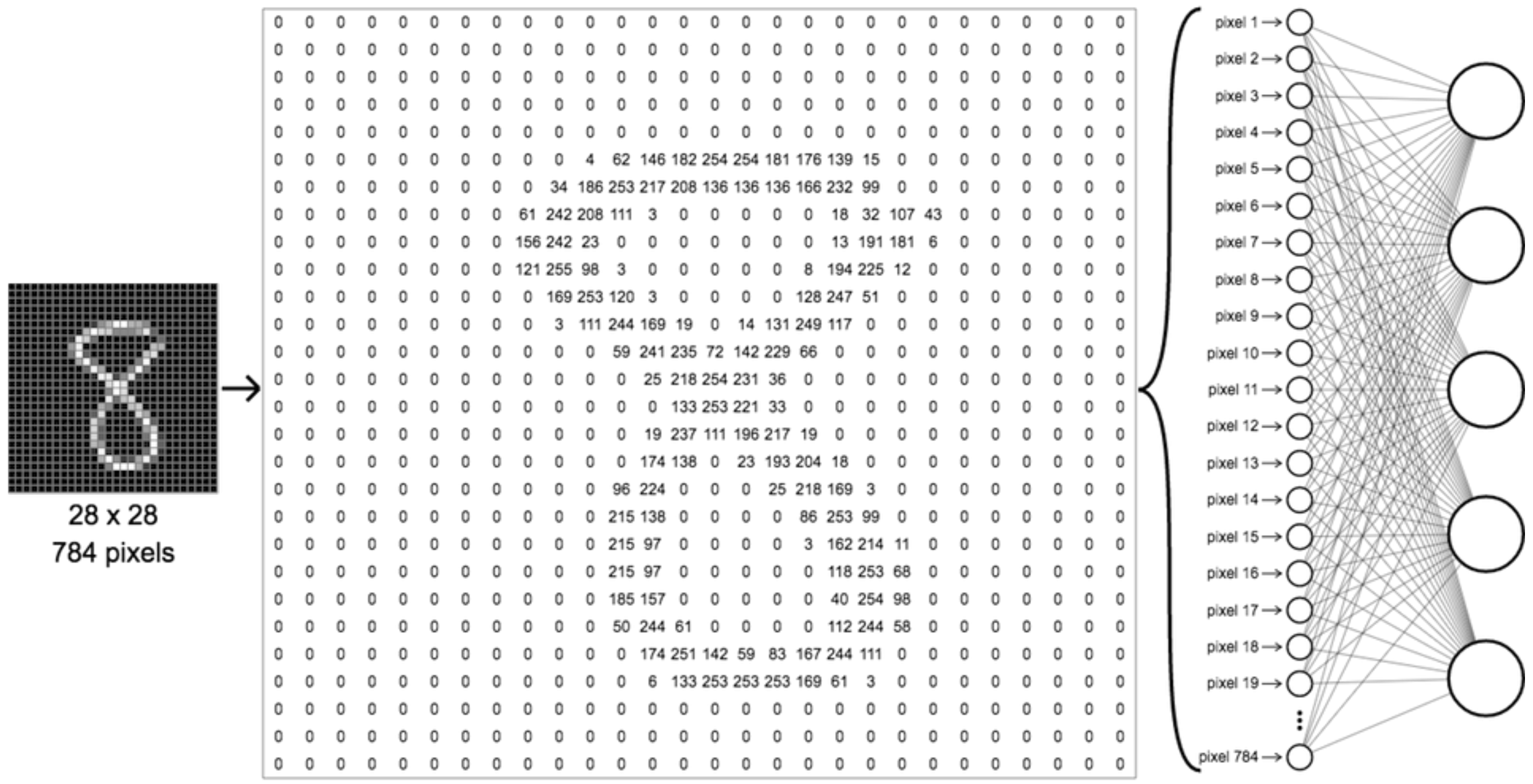
Full cosmological simulations including gravity, electromagnetism and hydrodynamics to evolve different species of particles.

Needed to compare theory with data

..but extremely computational expensive

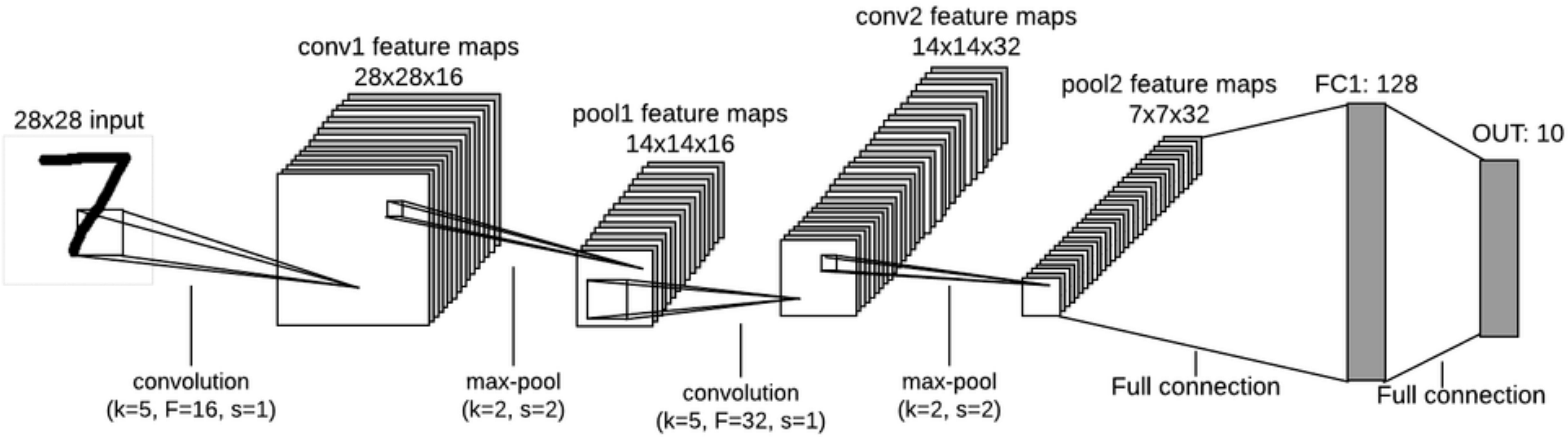
Can we use NN to map DM into Galaxy distribution?

Convolutional Neural Networks

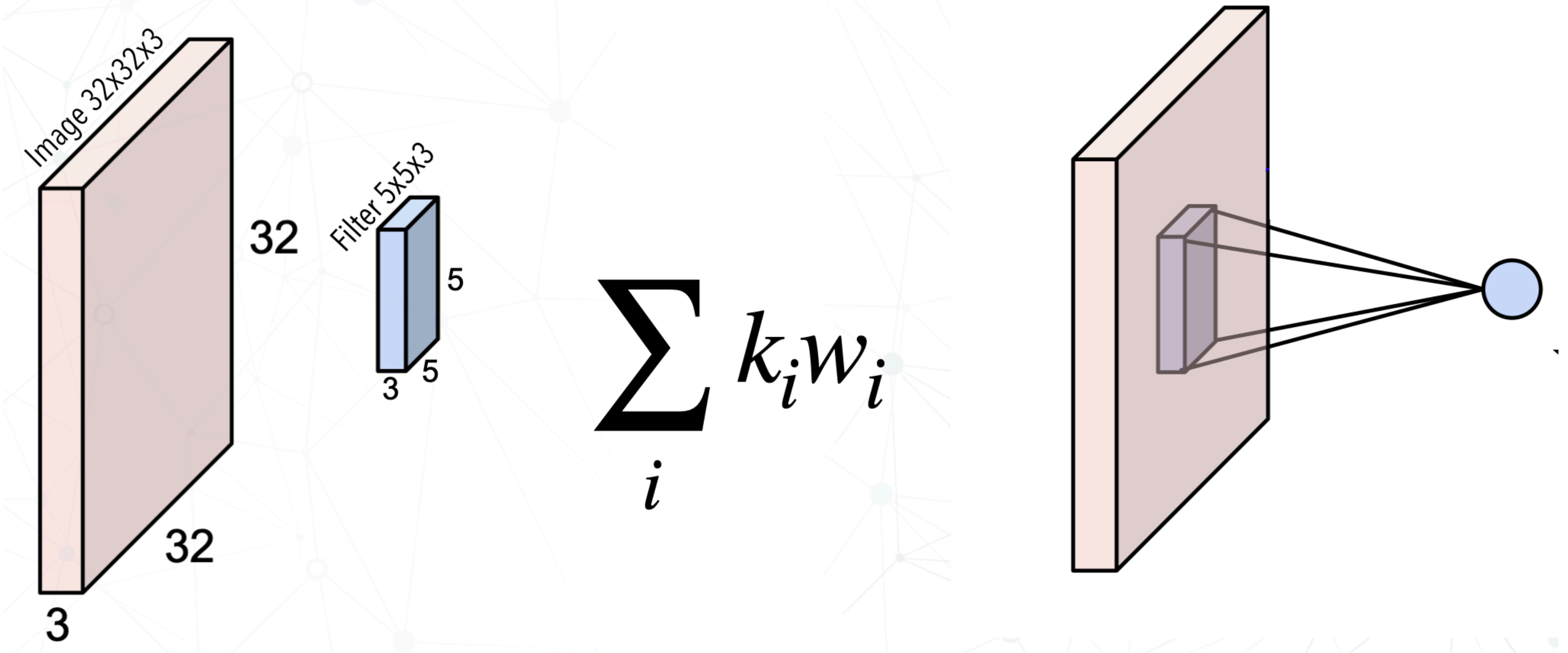


Fully connect NN works with one-dimensional layers, destroying the spatial information of the input

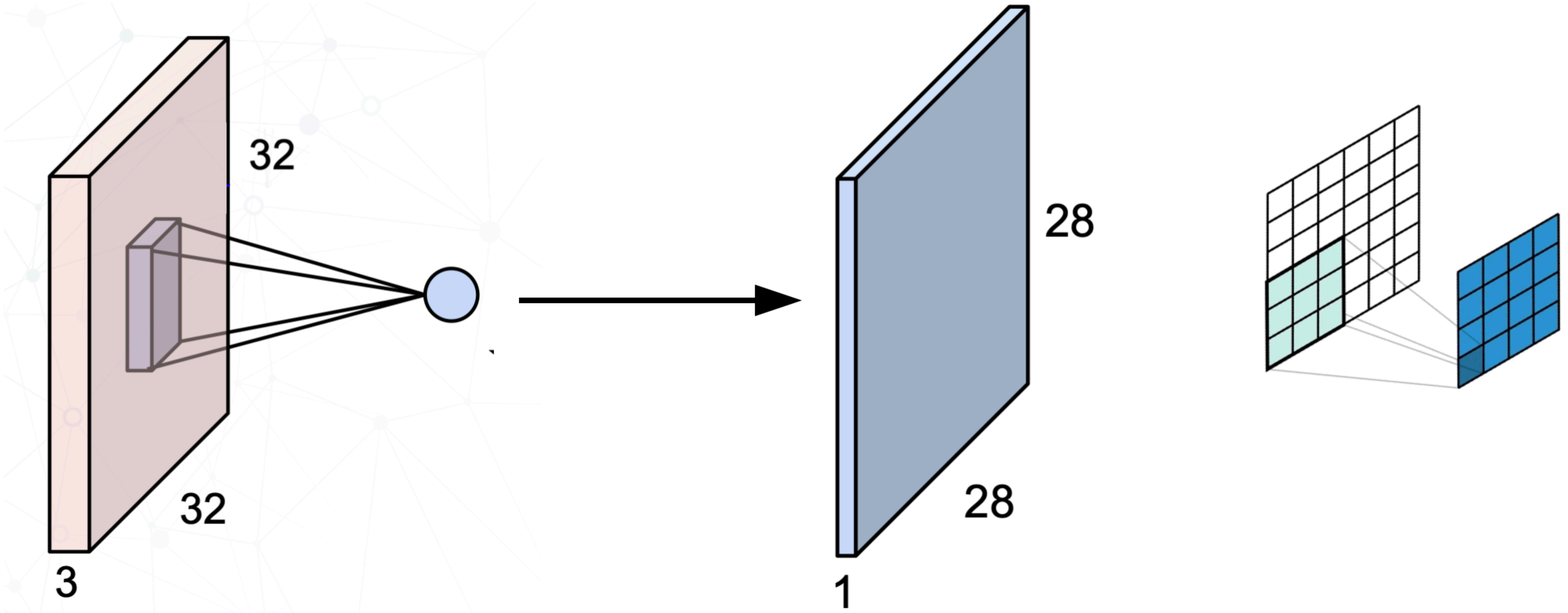
Convolutional neural networks preserve the spatial information by using filters instead of neurons



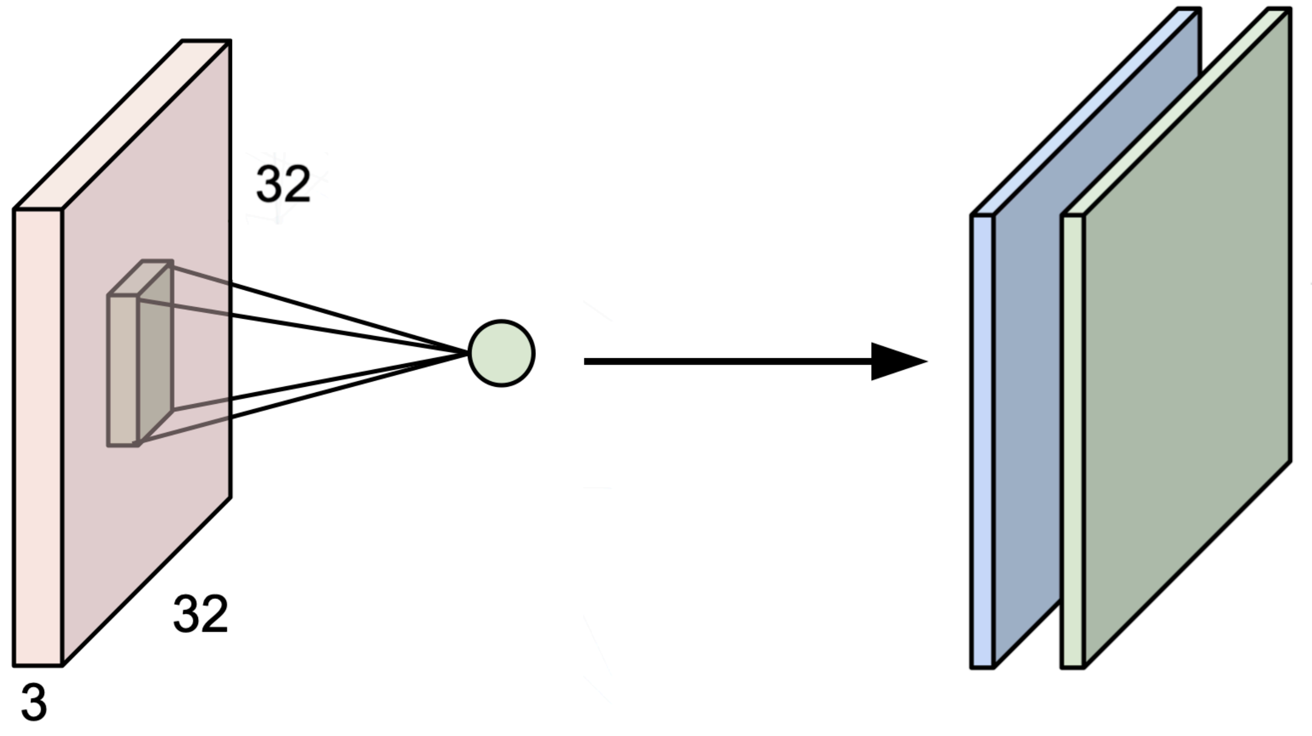
CNN: basic concepts



The convolution between sub-image (5x5x3) and filter is computed as the sum of the element wise product between the two



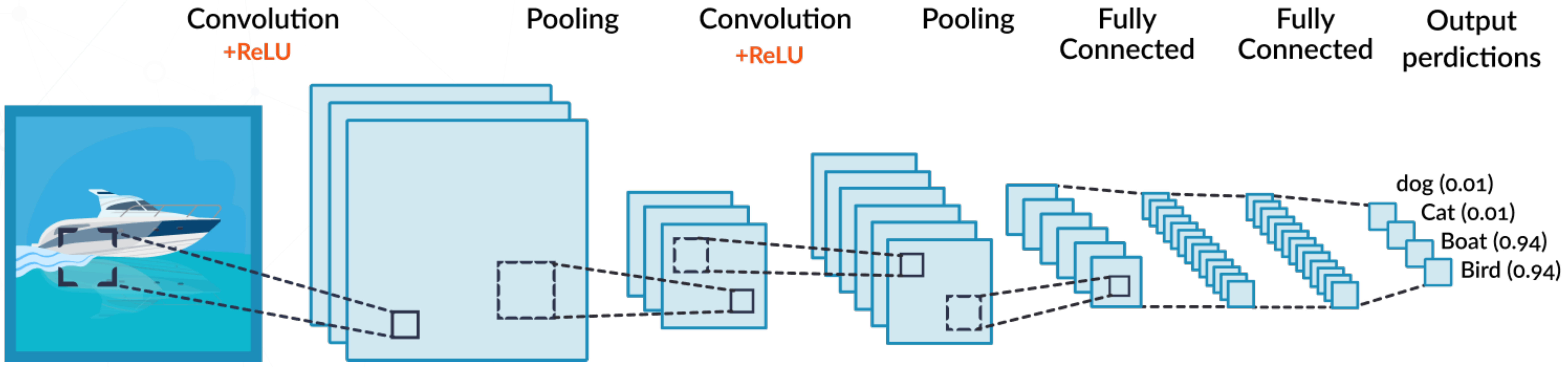
The full convolution is done by slicing the filter over the whole input image



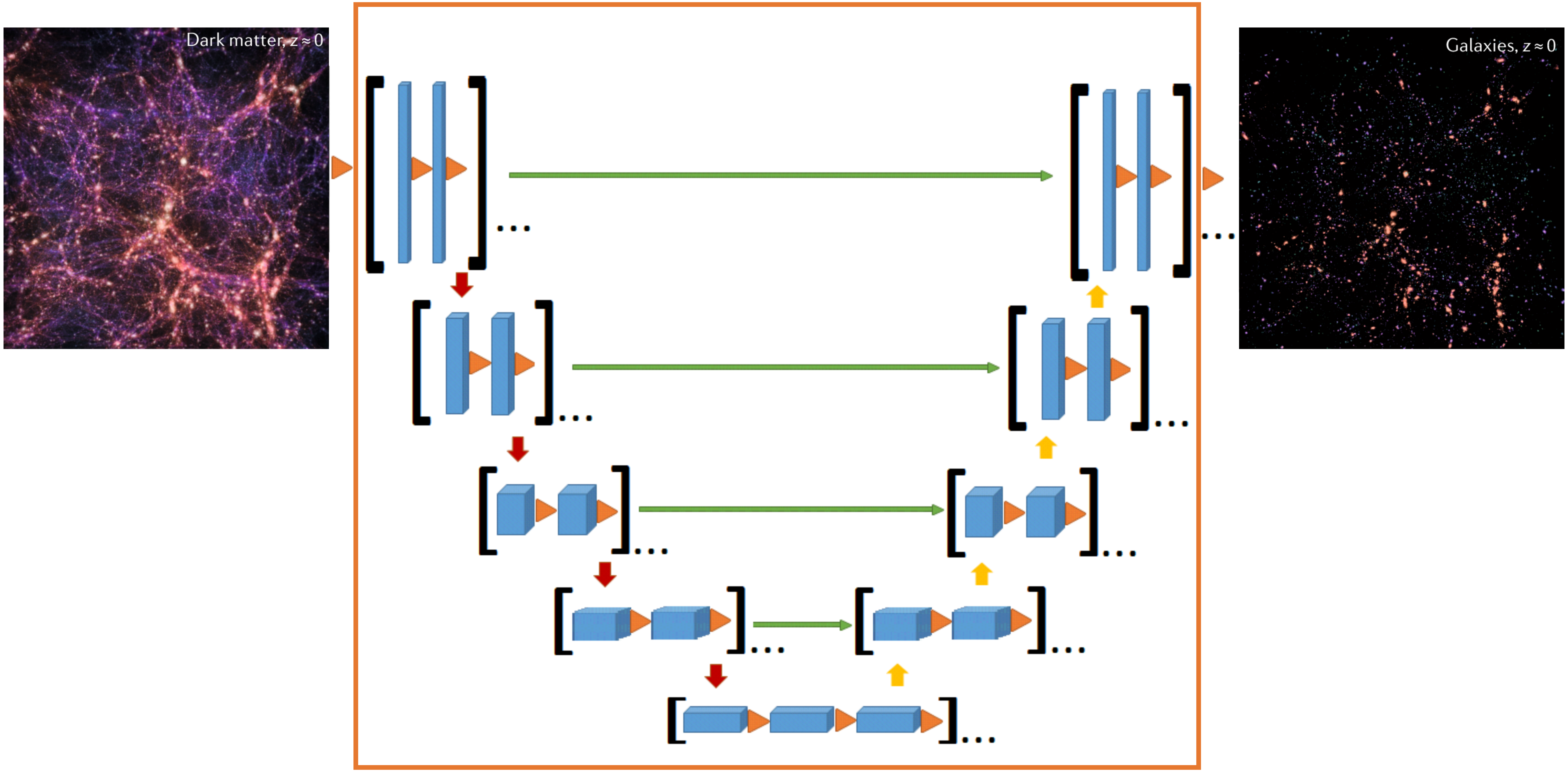
Different filters can be used in the same convolutional layer

CNN: complete architecture

- A deep CNN is built by stacking together several **convolution+pooling layers**
- **Pooling layers are used to reduce dimensionality** in (width x height) and usually they consist in taking the maximum or average values of pixels in the pooling reception field
- Typically while going deeper in the network (width x height) dimension is reduced while depth of the volume grows



From Dark Matter to Galaxies

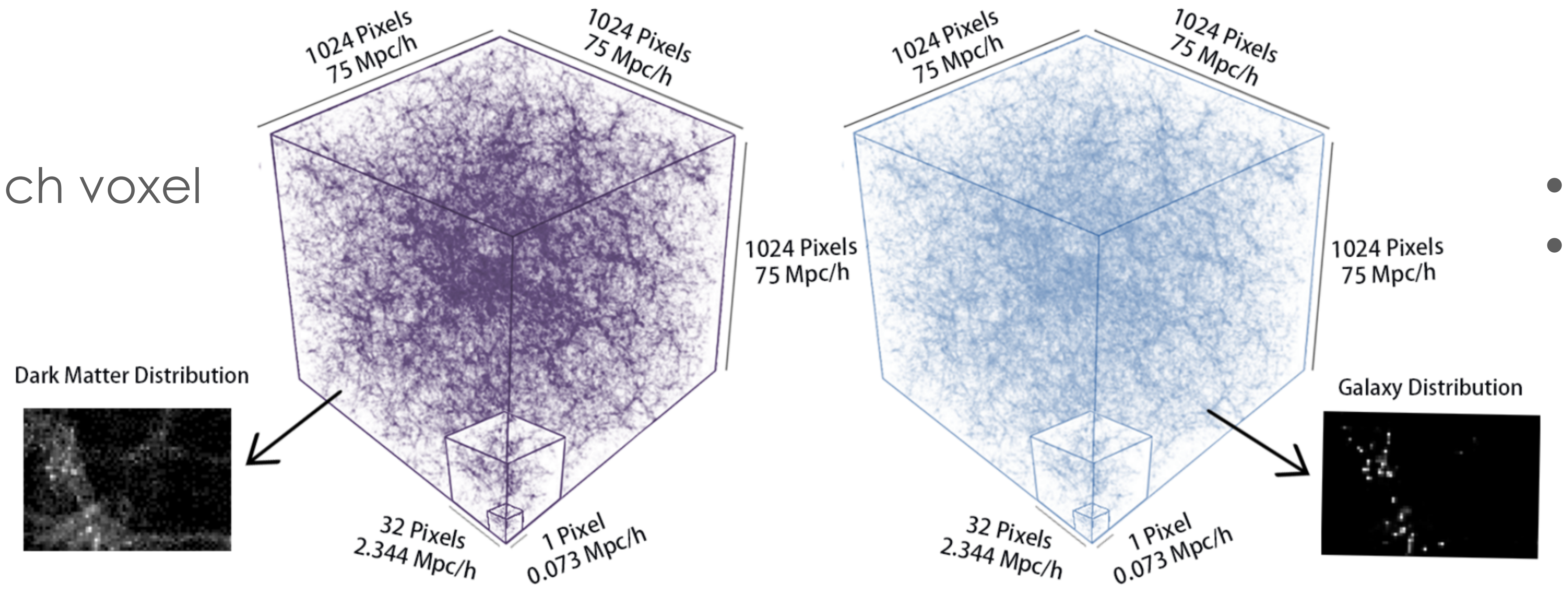


Mapping with Convolutional Neural Networks

Yip et al. 2019, Zhang et al. 2019 (arXiv:1902.05965, arXiv:1910.07813)

Dark matter:

- 0 to 7.5×10^5 particles in each voxel
- 45% non zero cells



Galaxies:

- 0 to 10 particles in each voxel
- 0.4% non zero cells

To solve the sparsity problem the task of going from Dark Matter to Galaxy distribution is divided in two steps:

1. Use a fully connected NN that output the probability that in each voxel there is at least one Galaxy

$$\mathbb{L}_{\text{CrossEnt}}(\hat{p}, \mathbf{y}) = -(w \cdot \mathbf{y} \cdot \log(\hat{p}) + (1 - \mathbf{y}) \cdot \log(1 - \hat{p}))$$

2. Use a CNN to make prediction on Galaxy distribution for voxel selected in the first step

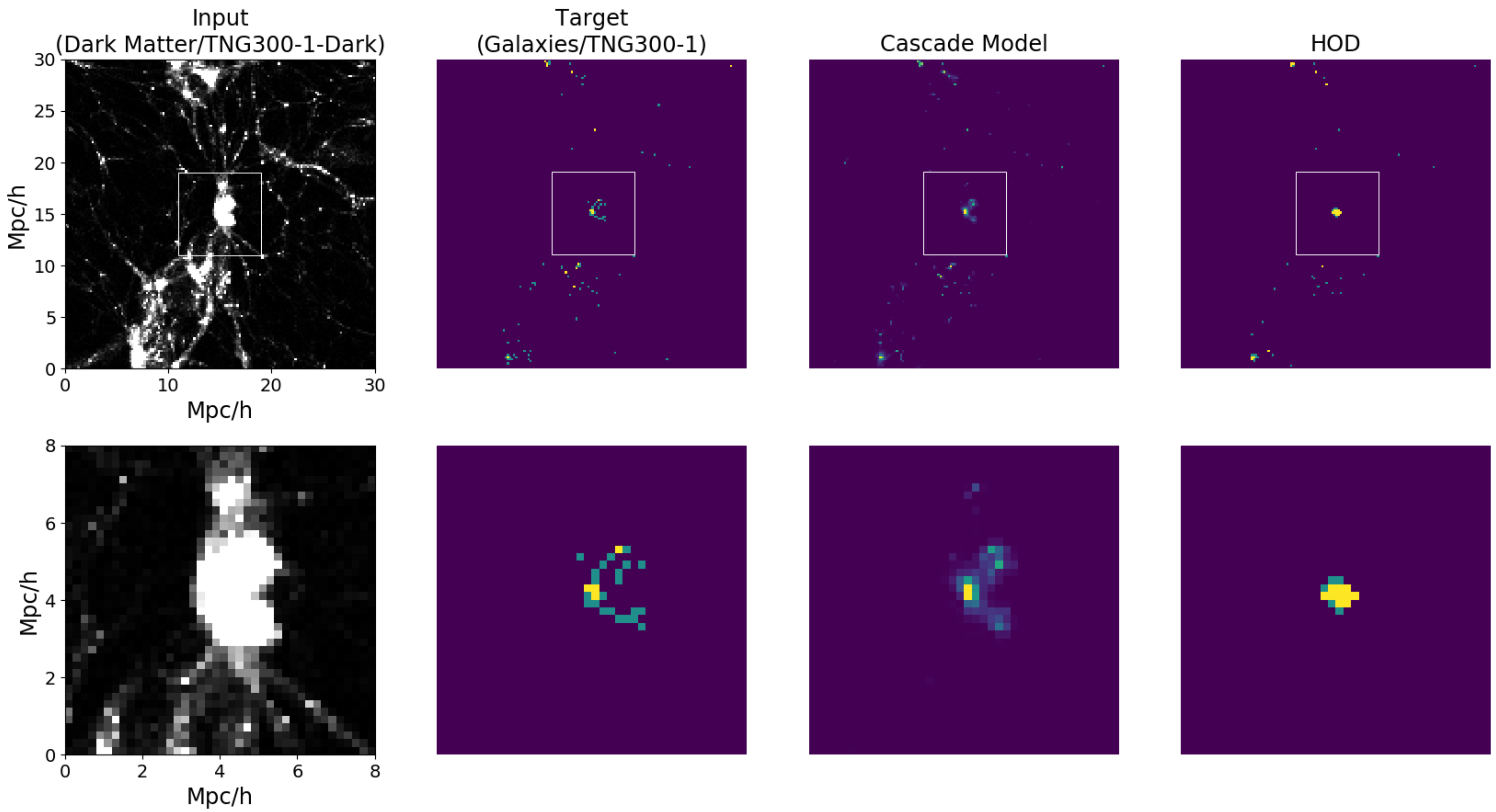
$$\mathbb{L}(n_g, \hat{p}, n_t) = M(\hat{p})(n_g - n_t)^2$$

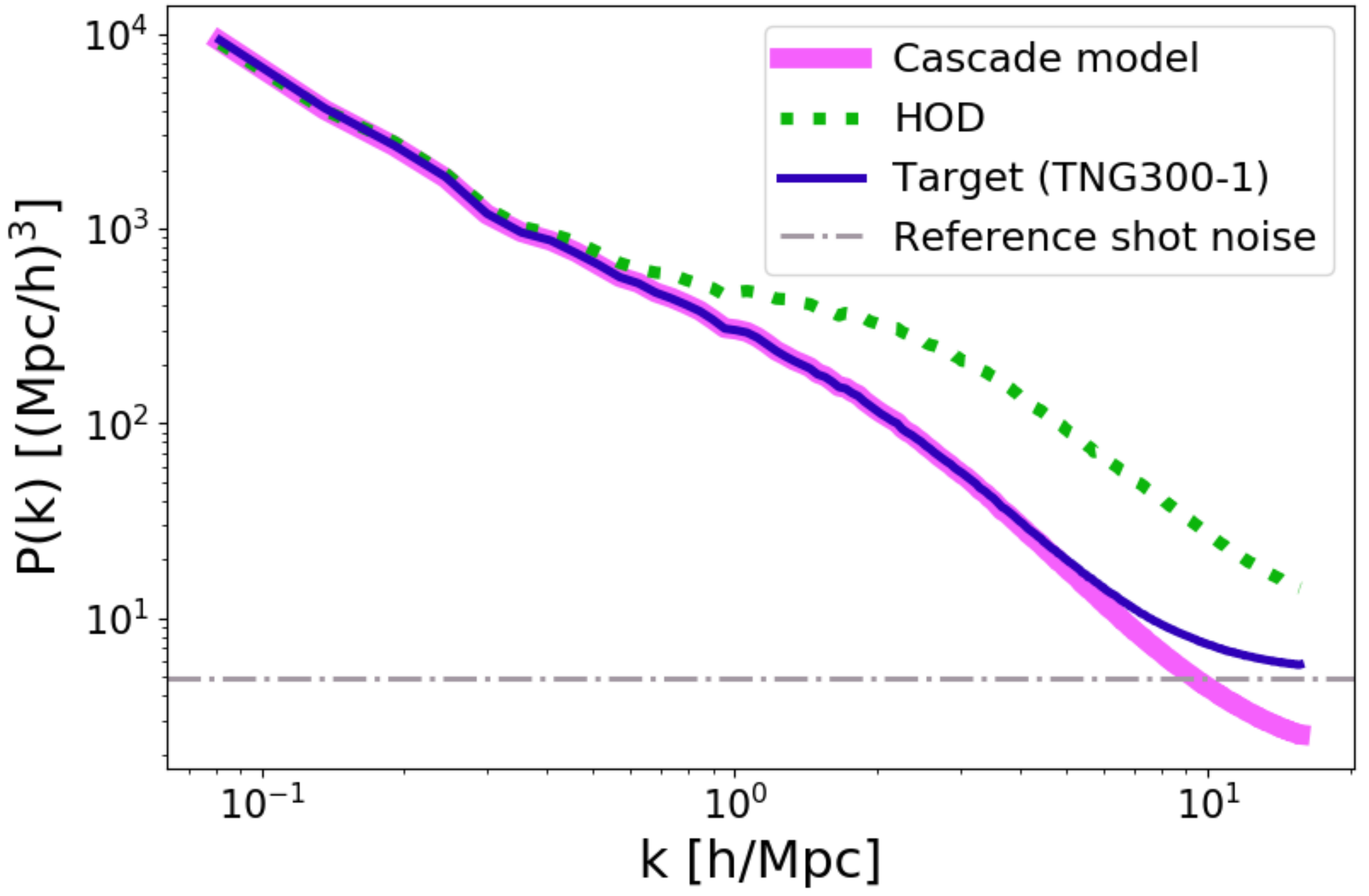
$$M(\hat{p}) = \begin{cases} 1 & \hat{p} > 0.5 \\ 0 & Otherwise \end{cases}$$

Yip et al. 2019, Zhang et al. 2019 (arXiv:1902.05965, arXiv:1910.07813)

Target: full hydrodynamical simulations

HOD: Halo Occupation Distribution model





Credits: D. Spergel, ML x Cosmo group at CCA

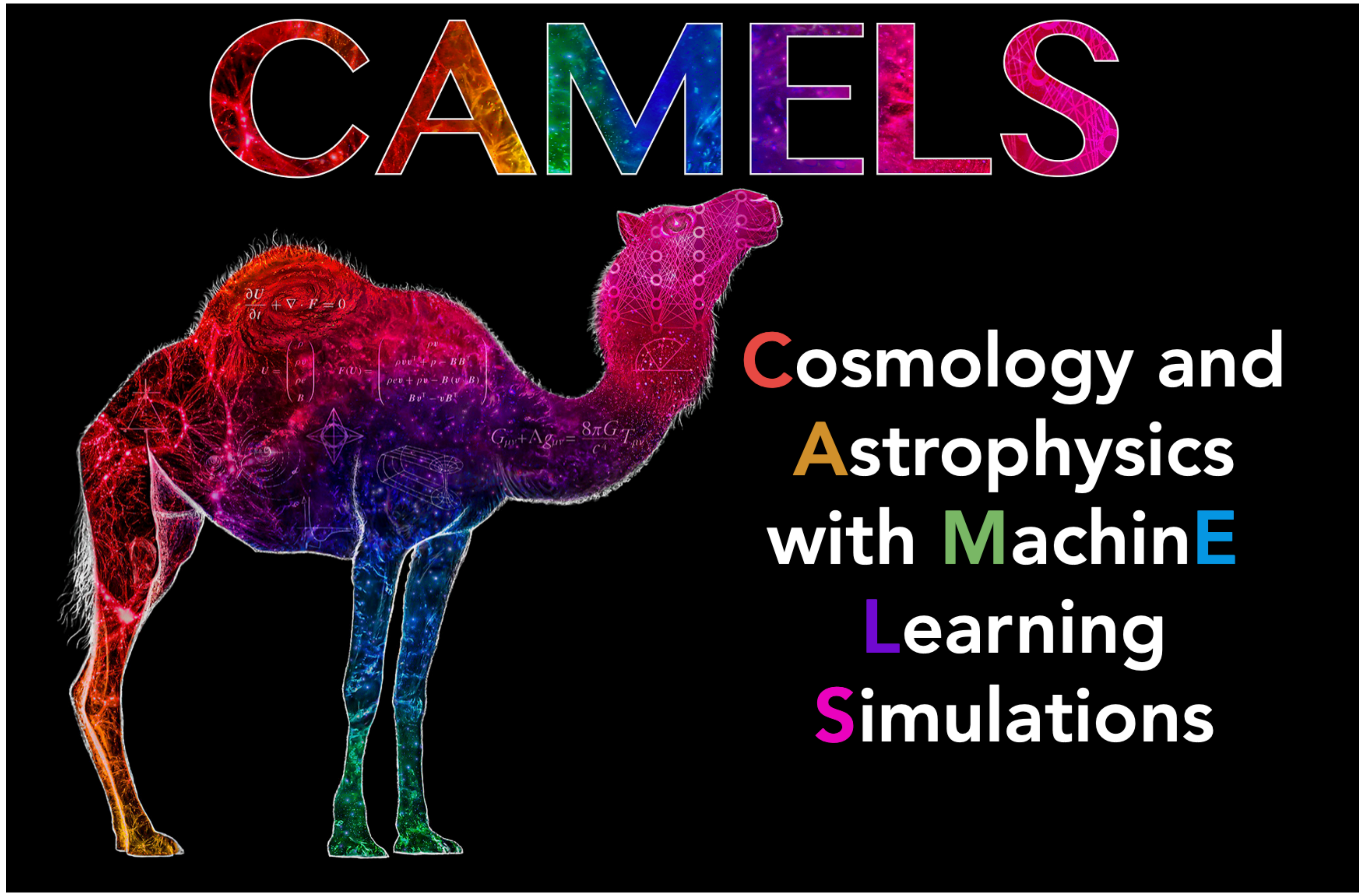
1. How to efficiently compute numerical simulations for theoretical prediction?
2. **Which is the optimal summary statistic?**
3. **How to marginalize over unknown physical processes?**

Credits: D. Spergel, ML x Cosmo group at CCA

1. How to efficiently compute numerical simulations for theoretical prediction?
2. Which is the optimal summary statistic?
3. How to marginalize over unknown physical processes?

We can use NNs to infer parameters directly from our observations.

- No need of selecting a summary statistic
- No need to build a likelihood model (likelihood-free inference)
- Need of large number of (realistic) simulations to train the NN



https://www.camel-simulations.org/science

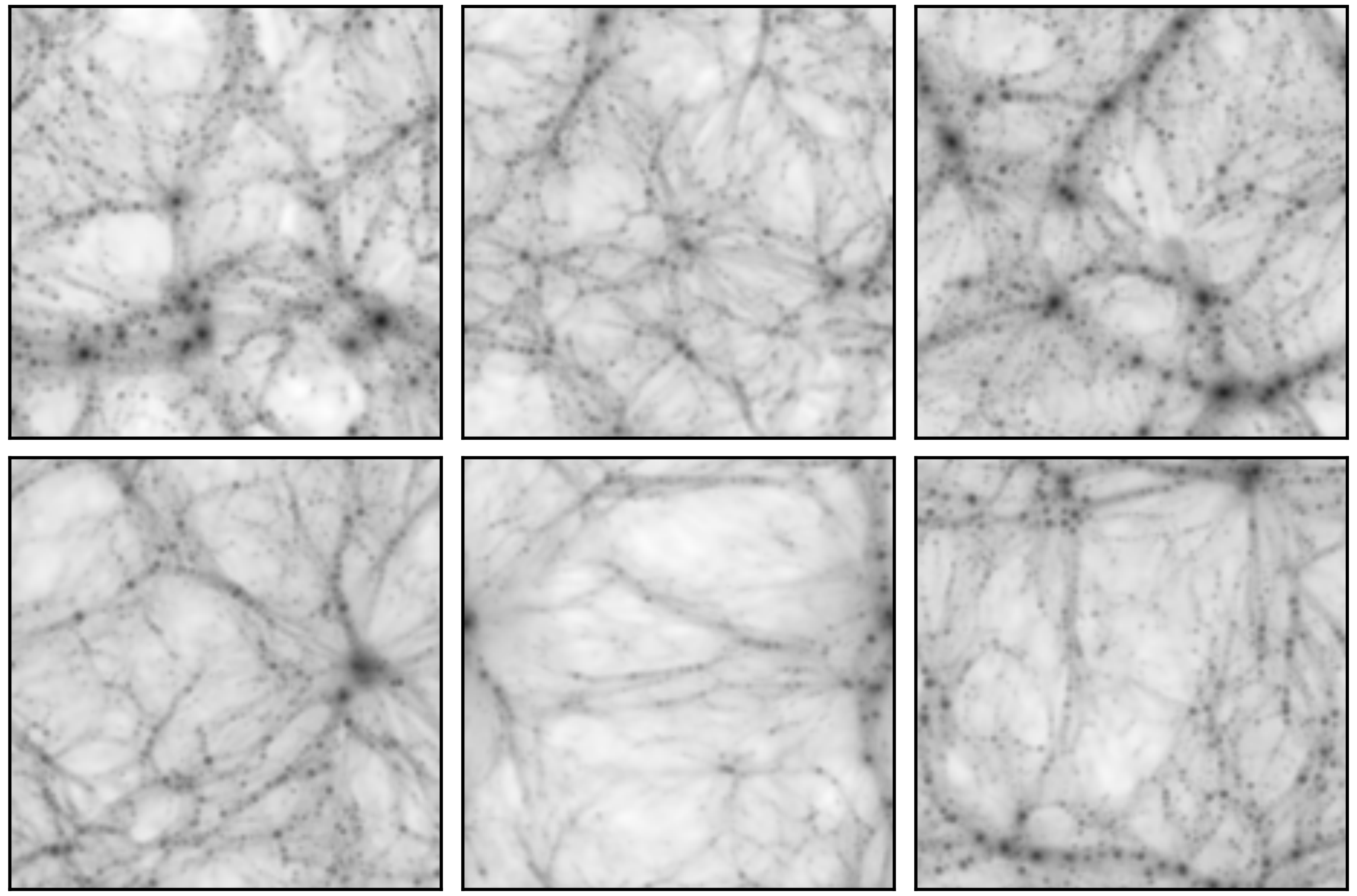
Villaescusa-Navarro et. al 2021 (arXiv:2201.01300)

- More than 4,000 numerical simulation (both N-body and magneto-hydrodynamic)
- Includes thousands of different cosmological and astrophysical models
- Large Dataset to train Machine Learning models

Likelihood-free inference with CNNs

Villaescusa-Navarro et. al 2021
[arXiv:2109.10360]

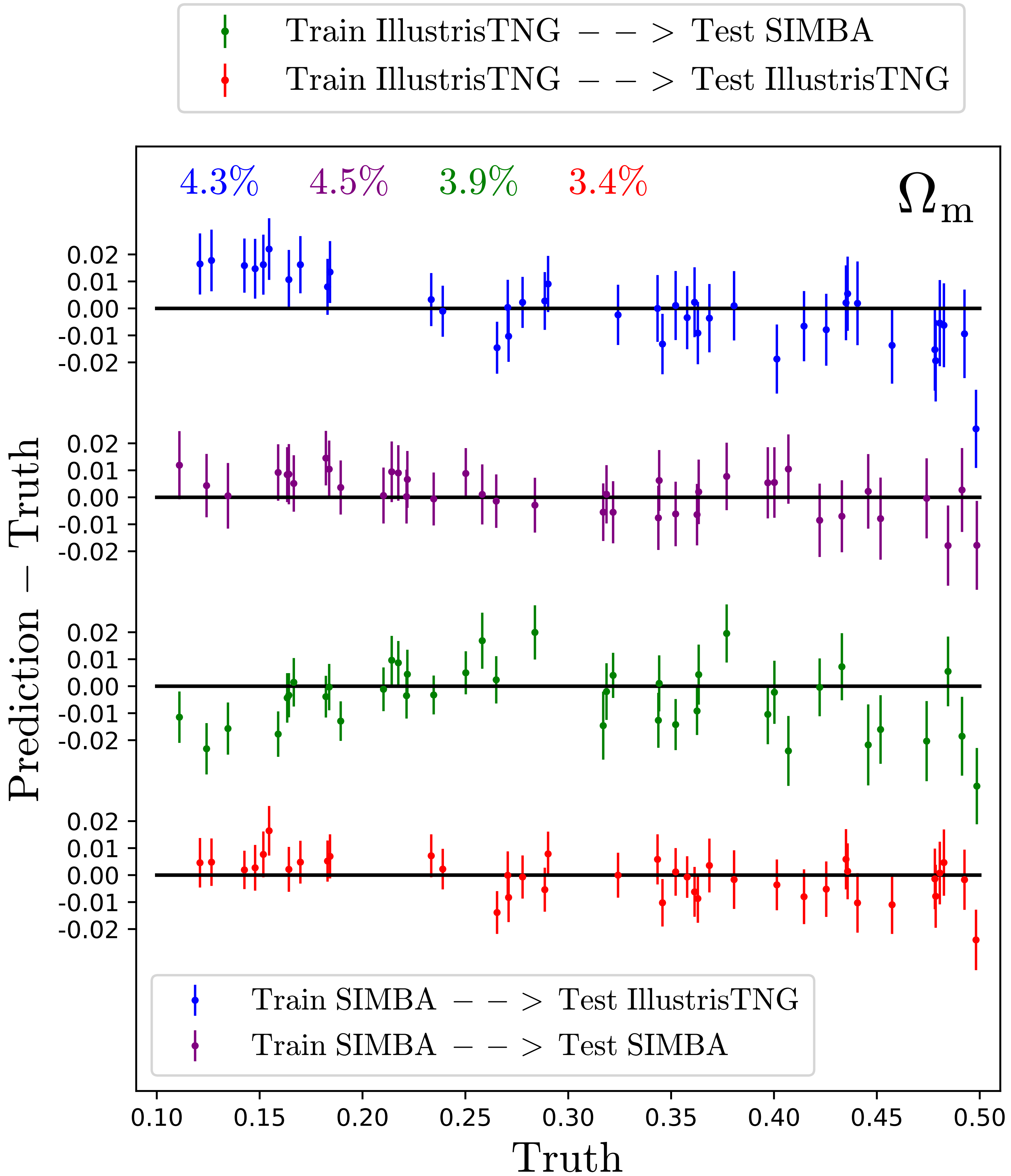
- Convolutional NN to infer cosmological parameters (Ω_m) from density maps
- Simulations are produced with two different codes with different treatment of baryonic effects
- Training is done on one set of simulations and test on the other in order to understand if the NN can marginalize over baryonic effects
- Output of CNN are Ω_m and $\sigma(\Omega_m)$



25x25 $(h^{-1}\,\mathrm{Mpc})^2$
256 x 256 pixels

CNN

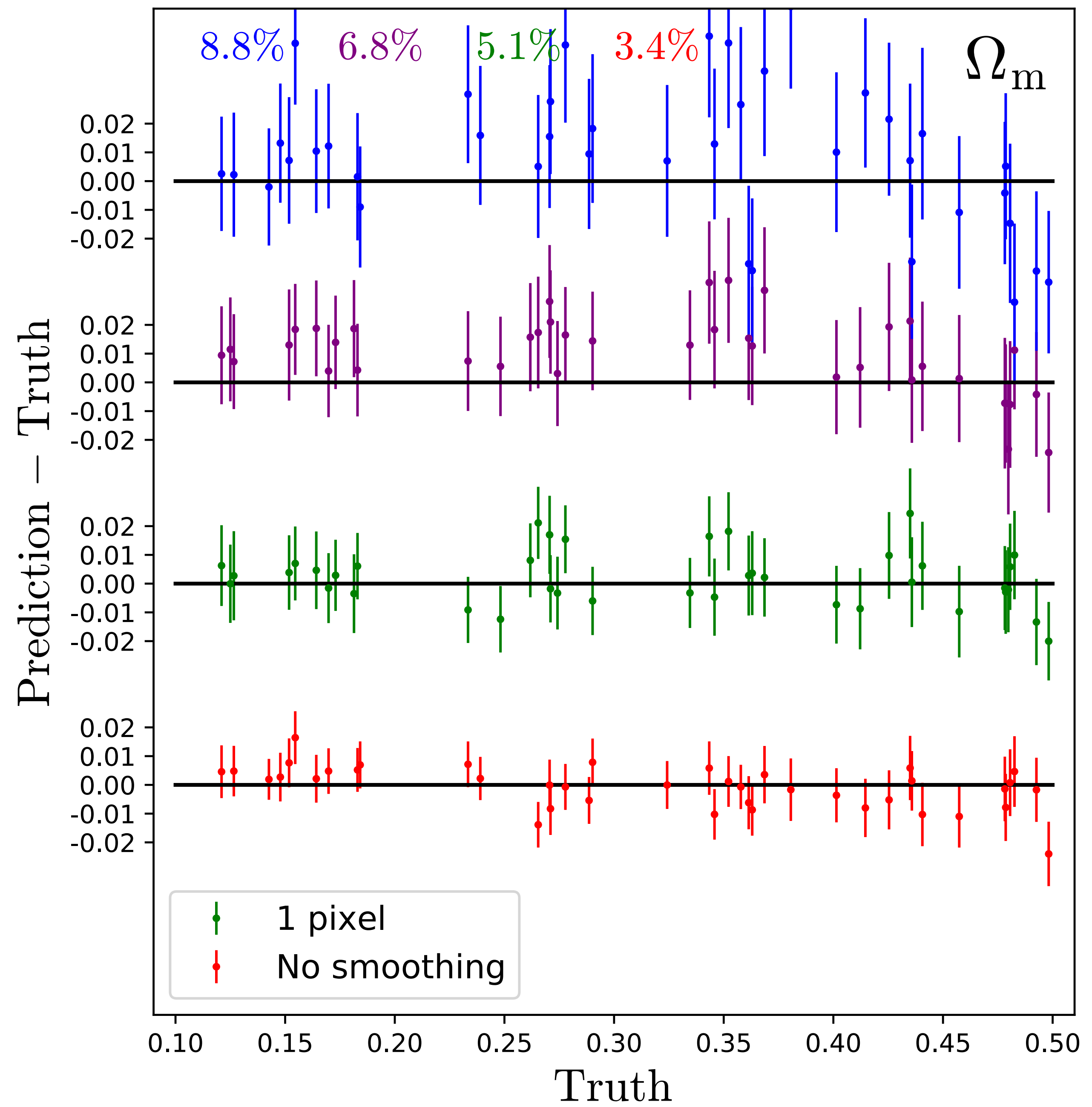
$$\mathcal{L} = \sum_{i=1}^{N_{\rm par}} \log\left(\sum_{j\in \rm batch} (\theta_{i,j}-\mu_{i,j})^2 \right) + \sum_{i=1}^{N_{\rm par}} \log\left(\sum_{j\in \rm batch} \left((\theta_{i,j}-\mu_{i,j})^2-\sigma_{i,j}^2\right) \right)$$



Likelihood-free inference with CNNs

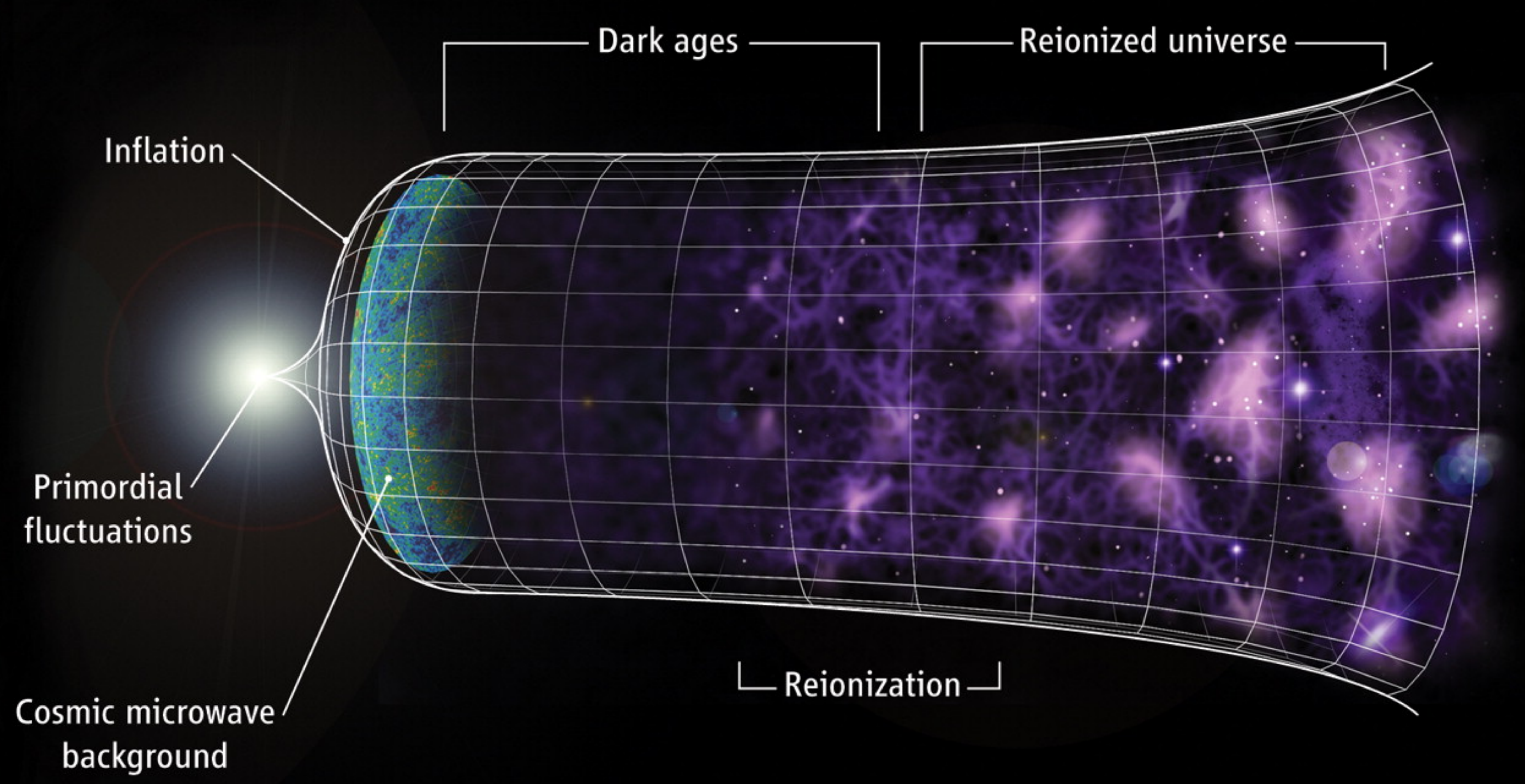
Villaescusa-Navarro et. al 2021
arXiv:2109.10360

- Is the NN only learning information from the total mass in the maps?
- No. The correlation of total mass and Ω_m is not enough to explain the small error bars that the NN is getting
- Is the NN only learning the information that is encoded in the two point correlation function?
- No. If you estimate Ω_m from the the power spectrum you get relative errors ~20%
- Is contamination from baryonic effects negligible?
- No. If you train a CNN from N-body only simulations it is unable to retrieve Ω_m
- Is the NN ignoring the smallest angular scales which are more effected by baryonic effects?
- No. Errors on Ω_m are larger if input maps are smoothed



Conclusions part 1:

- Many explorative works are being conducted to understand the applicability of NNs in the context of Cosmology from LSS
- Interesting results have been obtained for computing fast simulations and likelihood free inference
- Still at the level of “toy models” applied on simulated data



Many open questions:

- What is Dark Matter?
- What is the nature of Dark Energy?
- What is the correct theory of Inflation?
- What are neutrino masses?
- Is General Relativity correct on large scales?

CMB:

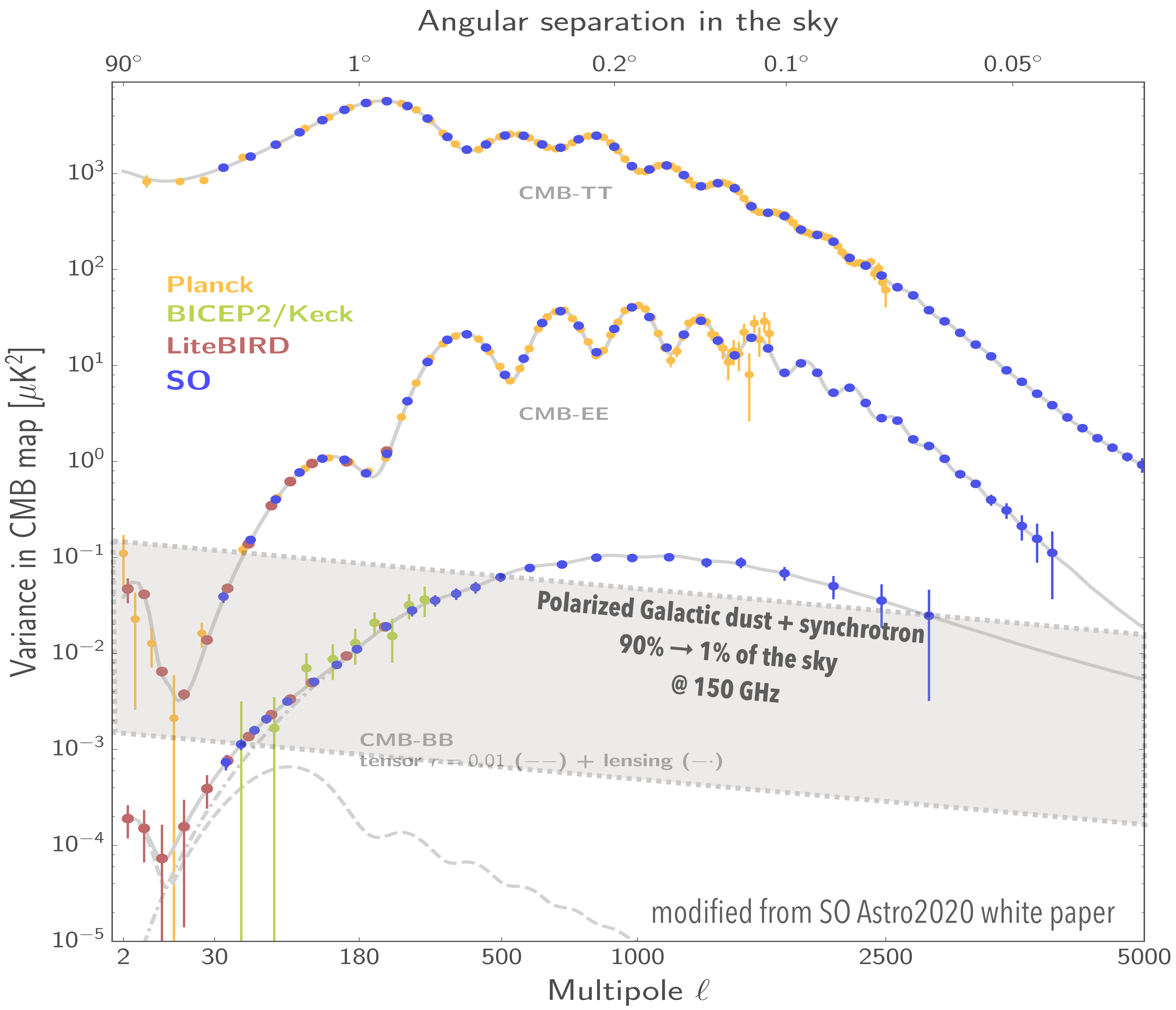
"simple", almost perfectly Gaussian, signal
…but faint and highly contaminated
(foregrounds and instrumental systematics)

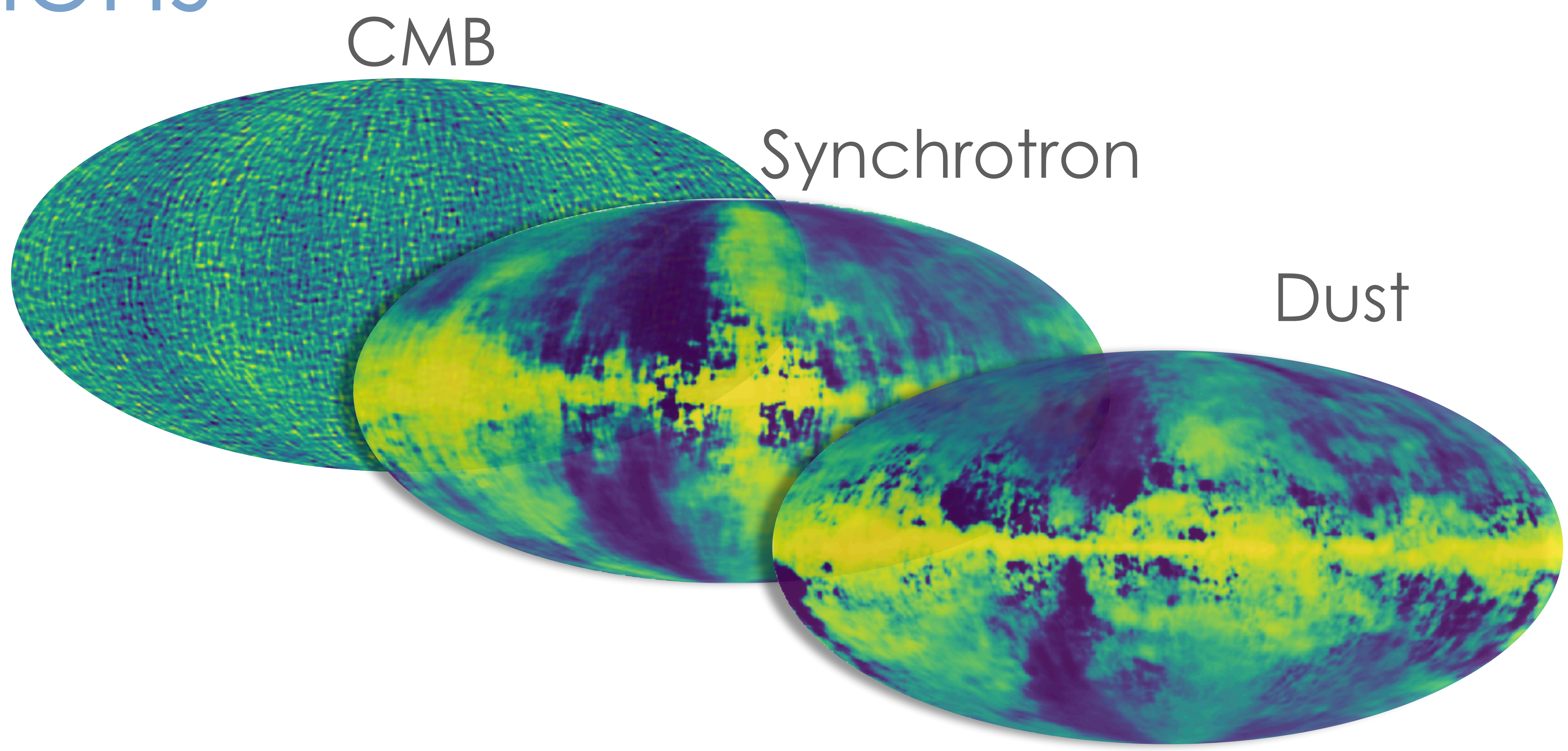
SIMONS OBSERVATORY

LiteBIRD

CMB-S4
Next Generation CMB Experiment

Challenges in CMB observations





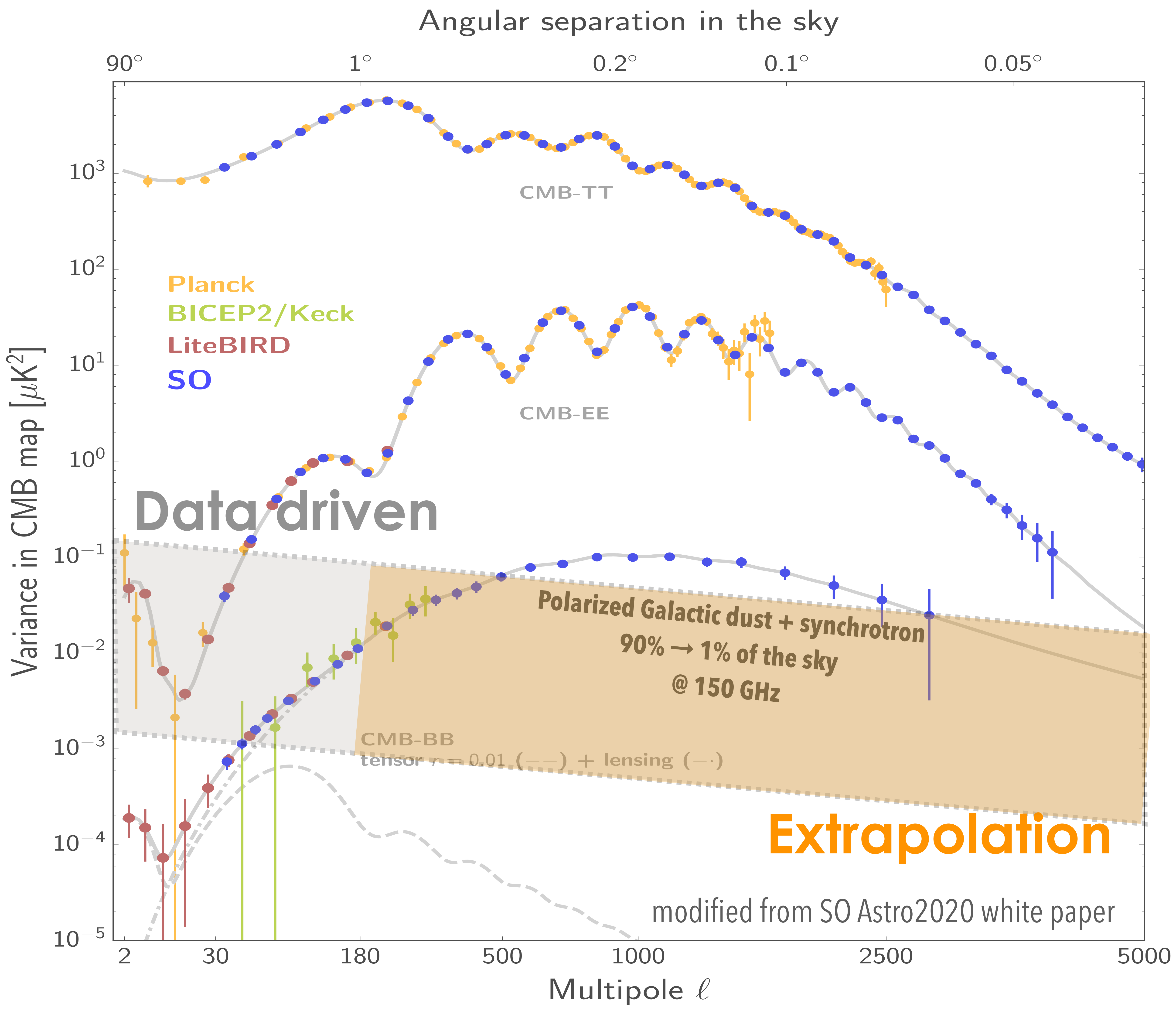
Main Goal: detection of primordial Gravitational waves (B-modes)

Challenges: Instrumental systematics + contamination by foreground emission

How can machine learning help?

1. Solution to specific, unsolved, problems/tasks: e.g. foreground modeling, instrumental systematic treatment, optimal masking etc…
2. Complement and support classical data analysis techniques: e.g. component separation, parameter estimation

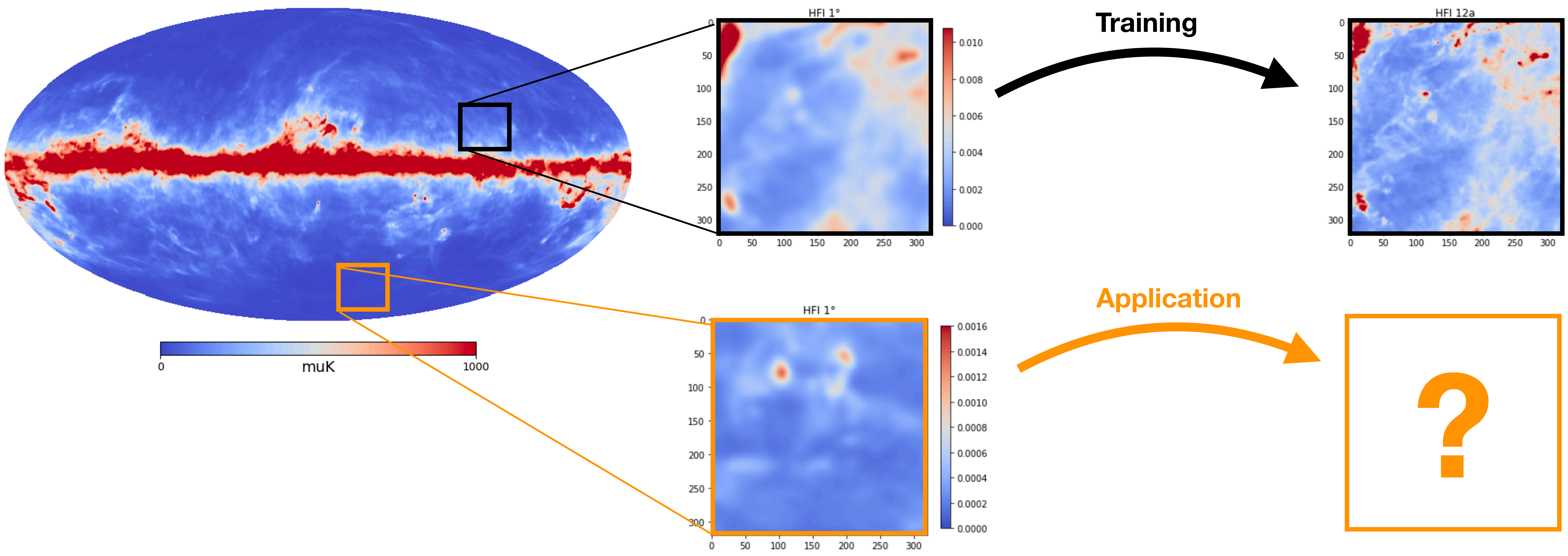
Foreground modeling



- Small scales (< 1°) added as extrapolation realizations of power law spectra
- Few data available!
- foregrounds are highly non-Gaussian, and we must understand their impact especially on lensing reconstruction

GANs to simulate small scale foregrounds

i. Train **Neural Networks to learn the statistics of foregrounds at the sub-degree scale** in total intensity (in the regions where we have enough sensitivity)



ii. Reproduce the same statistics starting from large scales in other regions of the sky and in polarization

Generative Adversarial Networks (GAN)

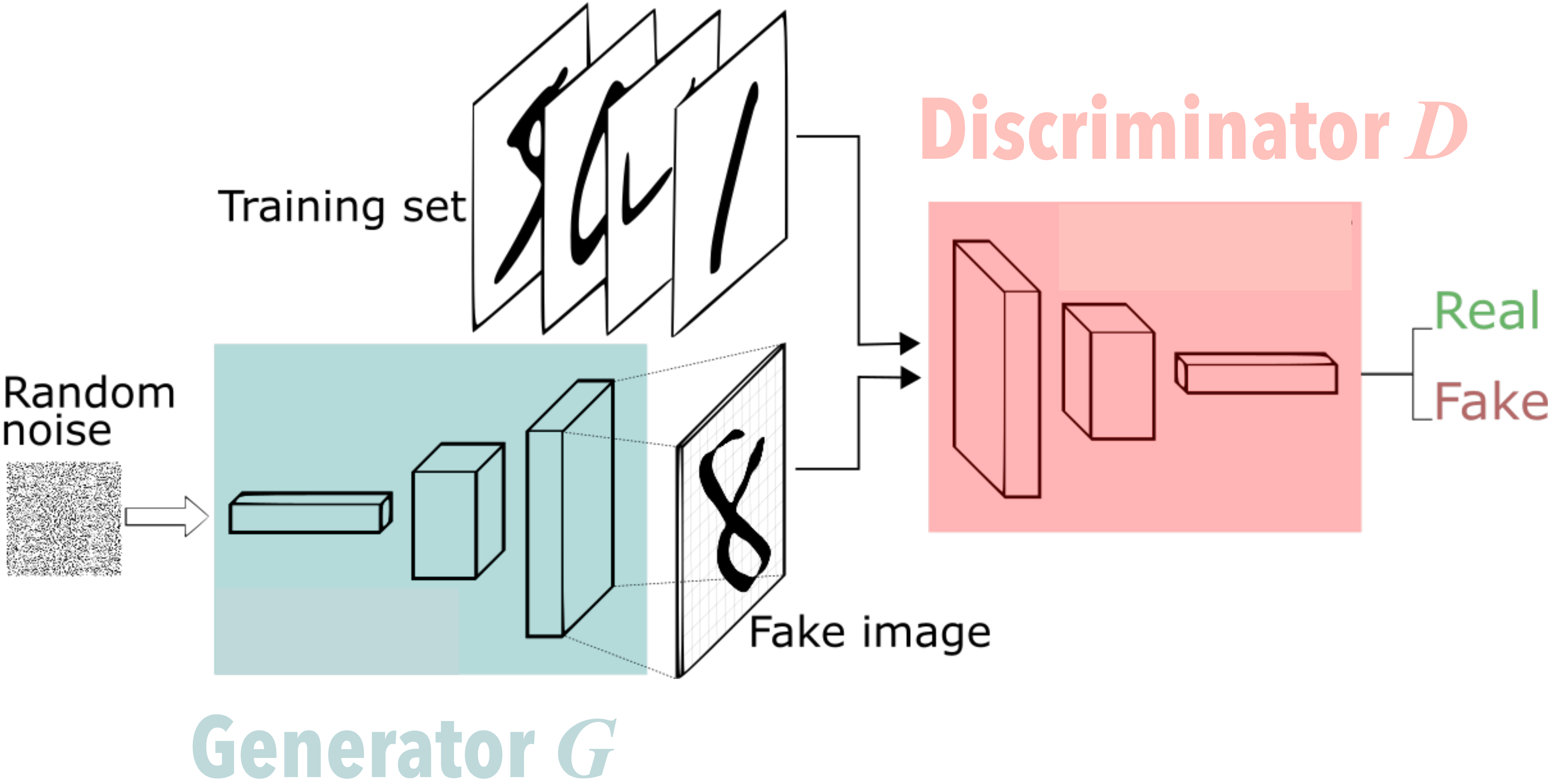
- A generative adversarial NN is a **system of two separate NNs that compete against each other**
- Given a training set **a GAN learns to generate new sets of data with the same “properties" as the training set**
- The original paper (Goodfellow et al. 2014, https://arxiv.org/abs/1406.2661) has more than 40.000+ citations

Abstract

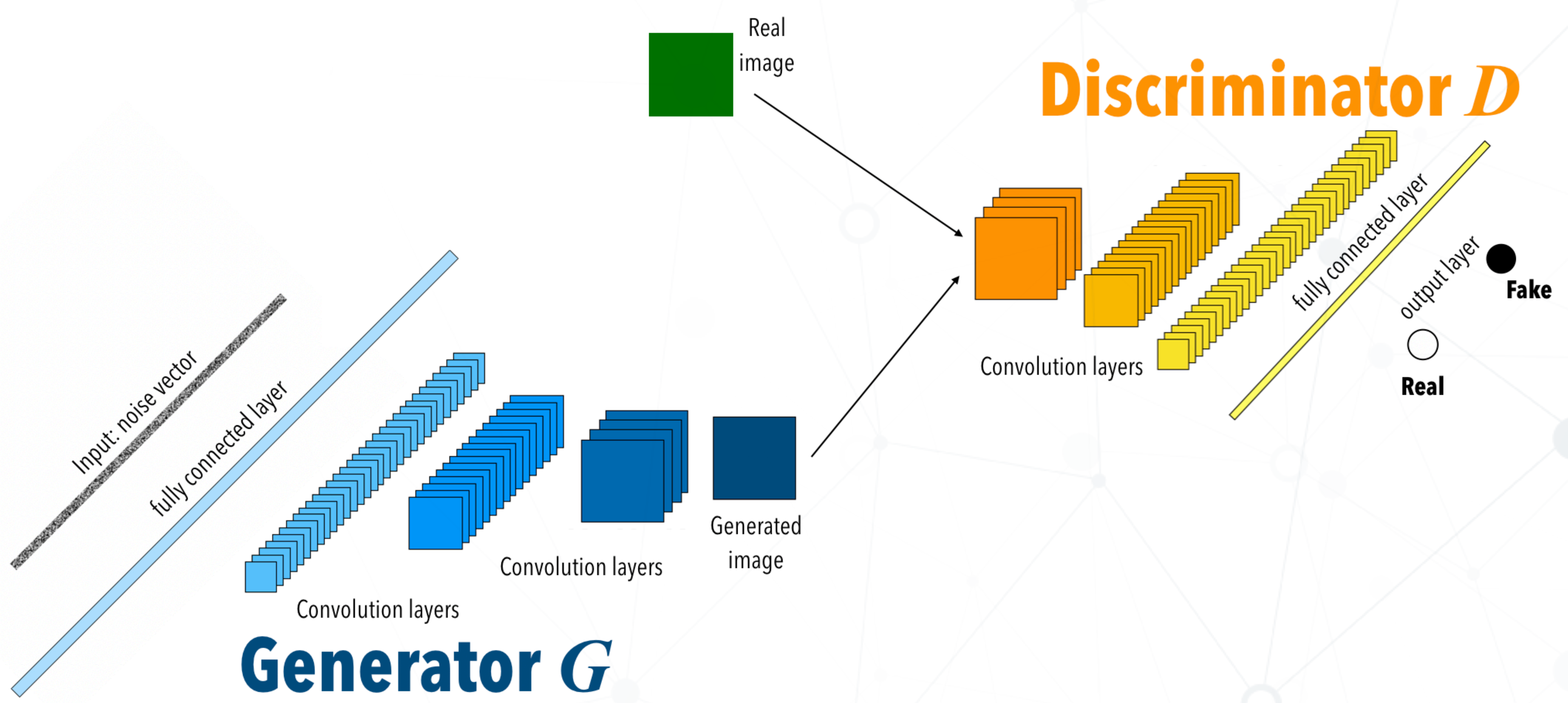
We propose a new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to $\frac{1}{2}$ everywhere. In the case where G and D are defined by multilayer perceptrons, the entire system can be trained with backpropagation. There is no need for any Markov chains or unrolled approximate inference networks during either training or generation of samples. Experiments demonstrate the potential of the framework through qualitative and quantitative evaluation of the generated samples.

GAN: architecture

Deep Convolutional GAN (DCGAN): Radford et al. 2015 https://arxiv.org/abs/1511.06434



GAN: training



N : number of elements in the mini-batch

z^i : i-th noise vector which is the input of G

x^i : i-th real image

$G(z^i)$: output of the generator for z^i

Discriminator D:

- D receives as input **real images** (labeled as **1**) and **fake images** generated by G (labeled as **0**)
- its goal is therefore to output 1 when input is x and 0 when input is $G(z)$
- during training it aims at minimizing the following cost function:

$$\mathcal{J}_D = -\frac{1}{N}\sum_i^N \log\left[D(x^i)\right] + \log\left[1 - D\left(G(z^i)\right)\right]$$

in minimizing this cost function weights of G are fixed while the training optimizes the weights of D

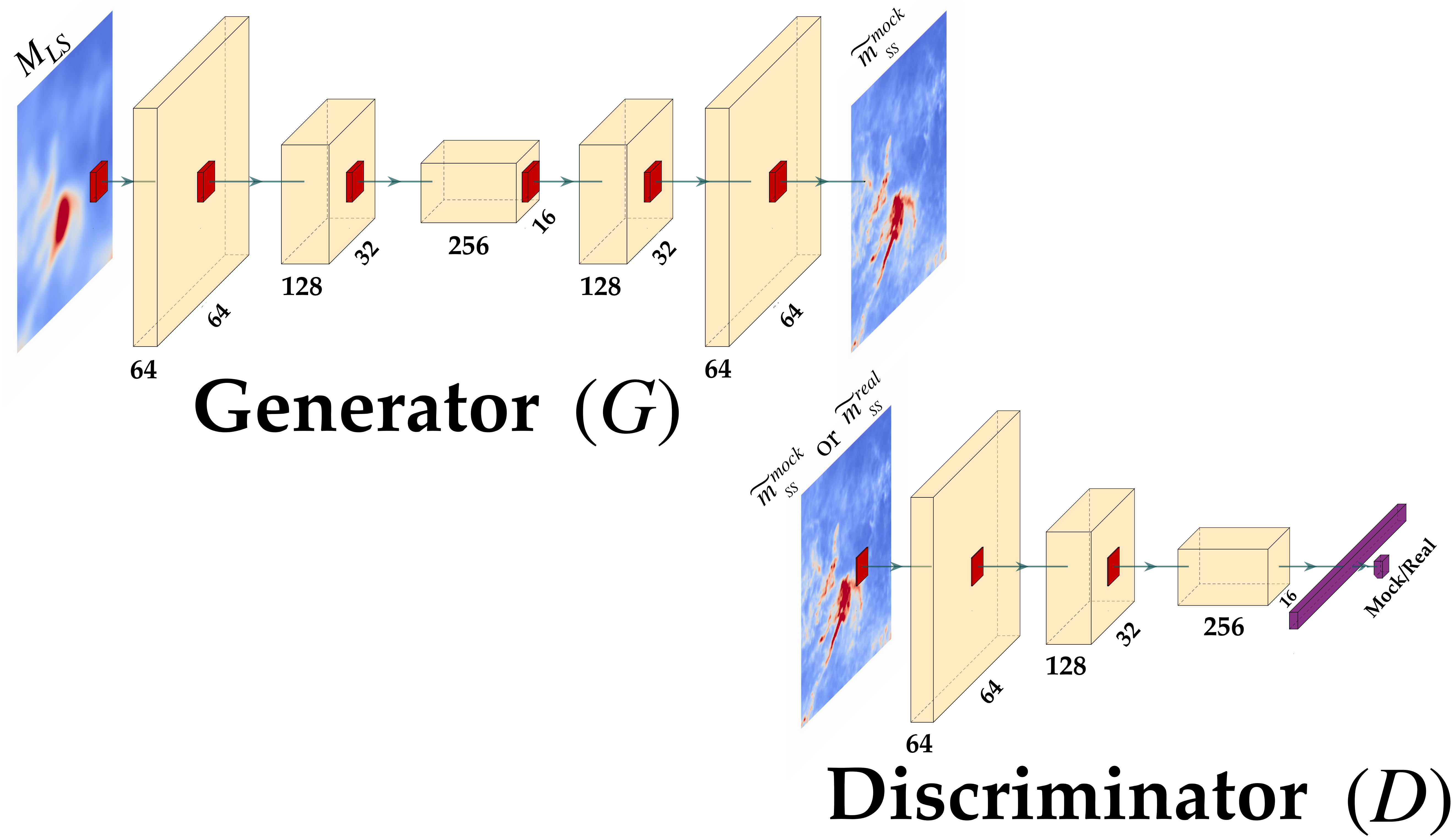
Generator G:

- its goal is to produce images that mislead D
- it aims at minimizing

$$\mathcal{J}_G = -\frac{1}{N}\sum_i^N \log\left[D\left(G(z^i)\right)\right]$$

in minimizing this cost function the weights of G are optimized while those of D are fixed

GANs to simulate small scale foregrounds

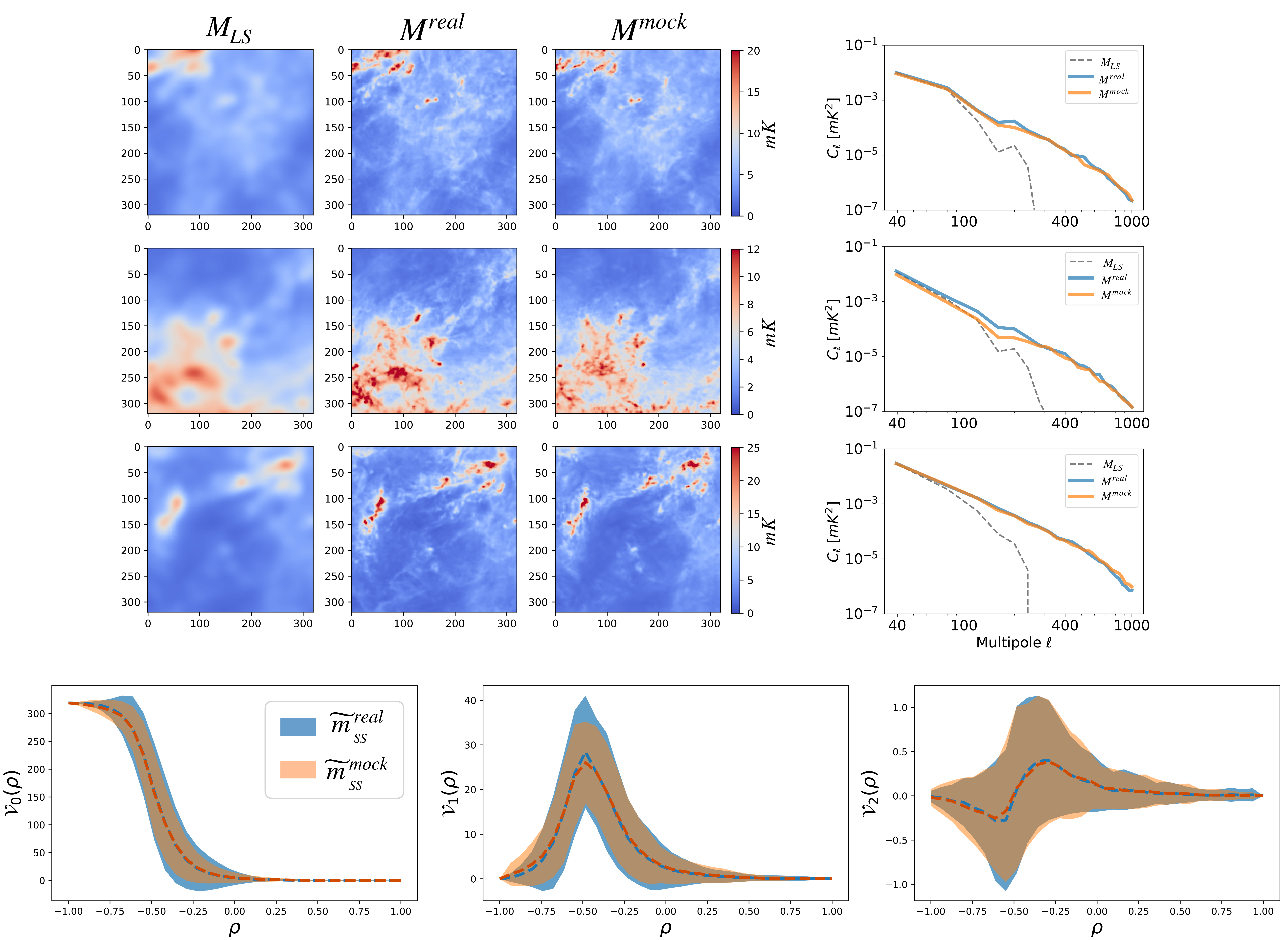


- ✦ Input to the NN are patches of the sky (20°x20°) at low resolution (80 arcmin): M_{LS}
- ✦ Output are small scale features at 12 arcmin: m_{SS}

$$M = M_{LS} + M_{SS}$$

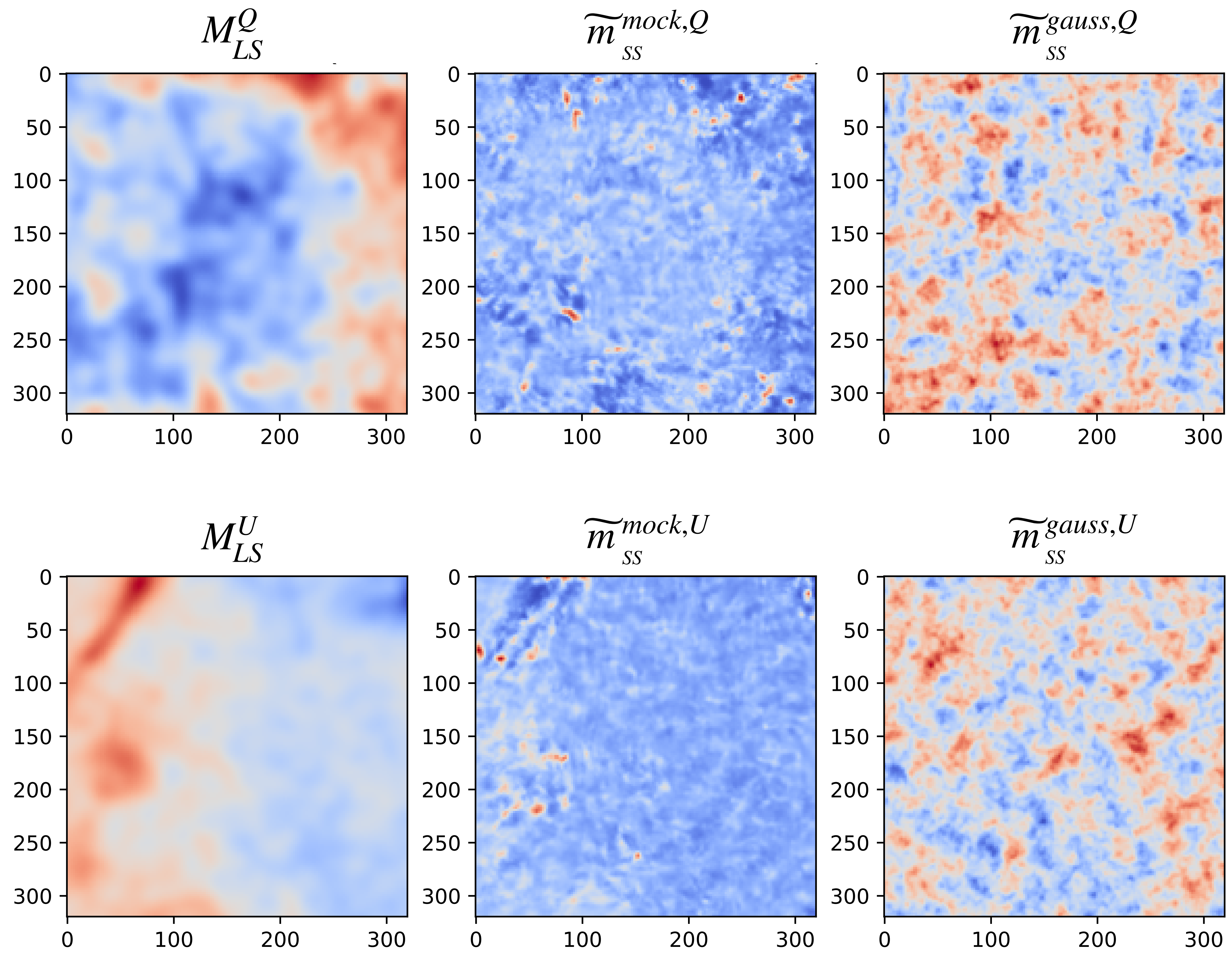
with small scales modulated by the large ones: $M_{SS} = M_{LS} \cdot m_{ss}$

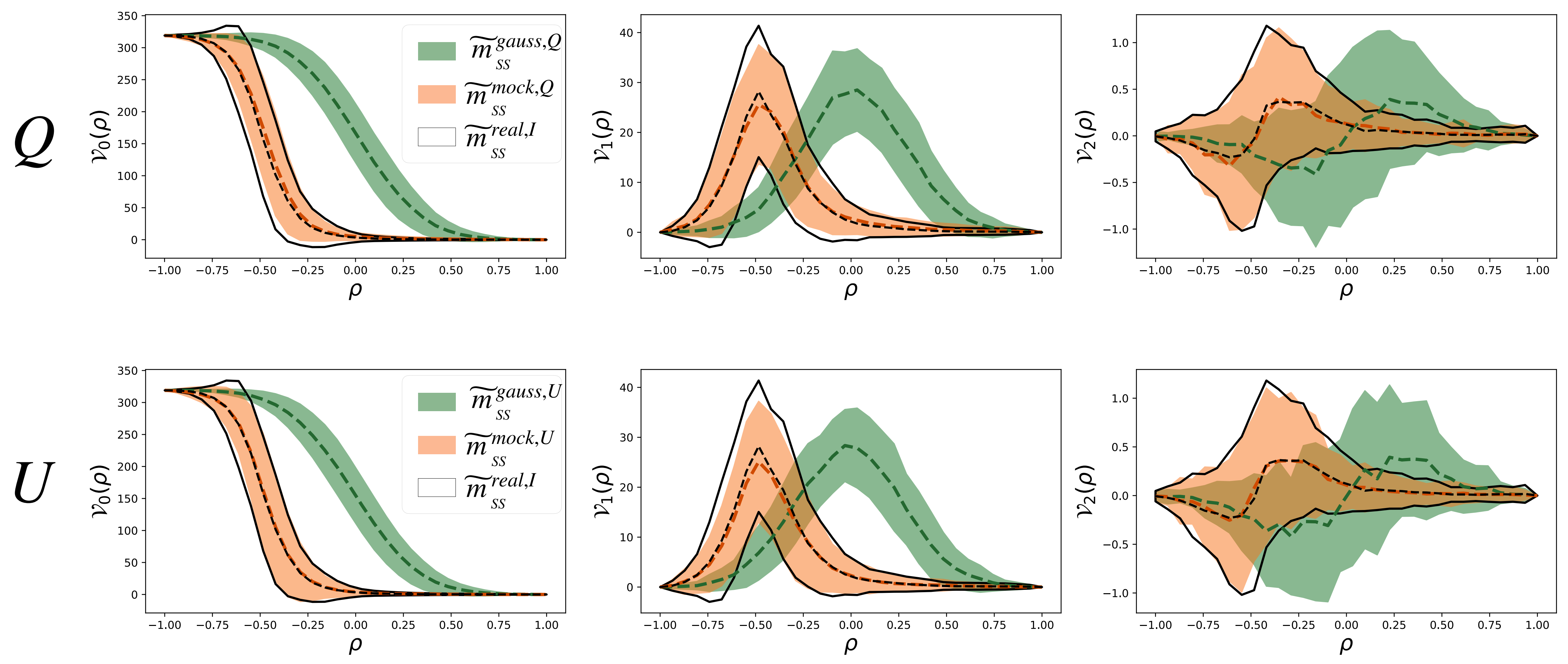
Total Intensity test case



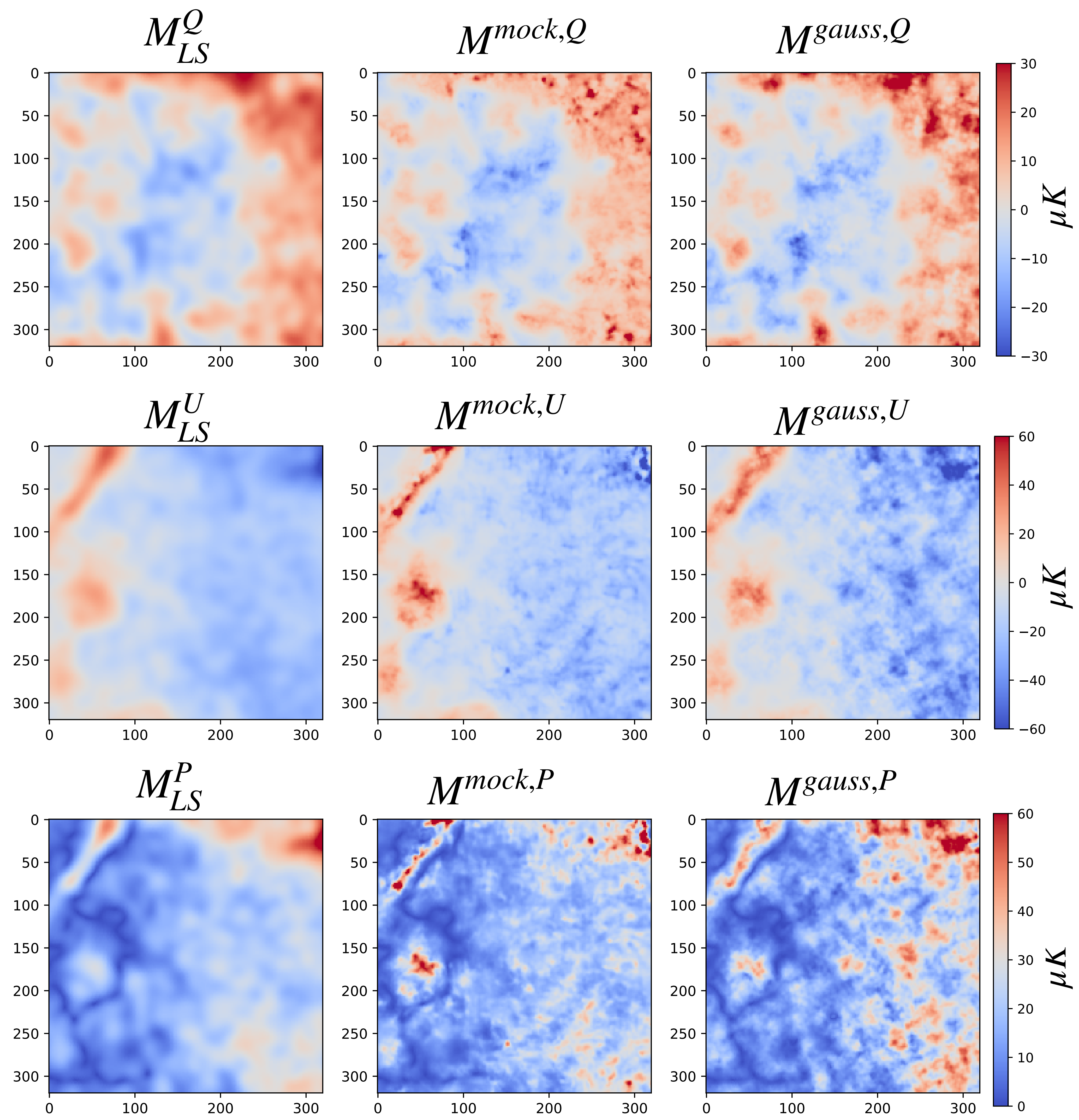
Polarization application

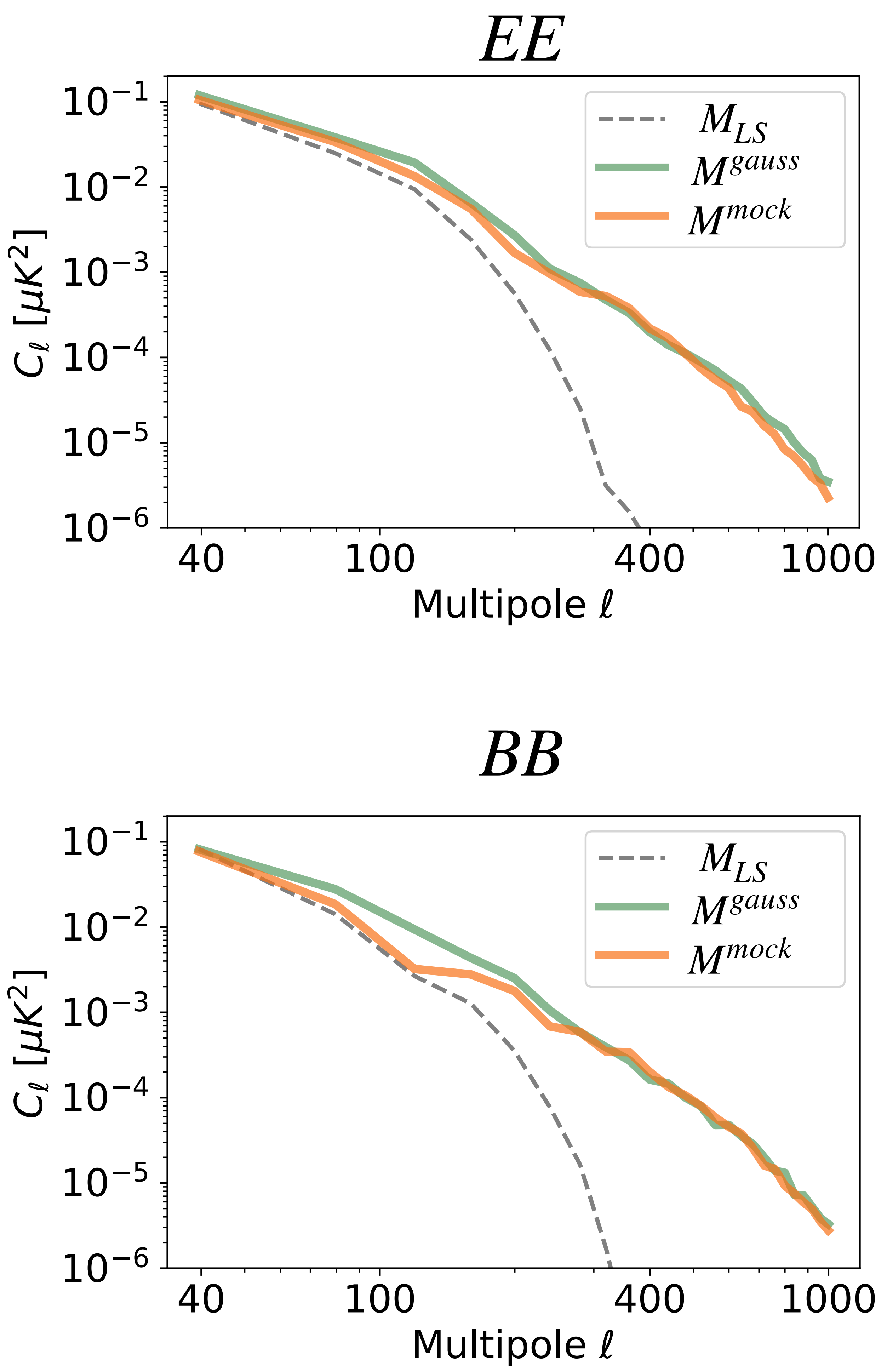
- **No data to train the network directly in polarization**
- Assumption that small scale structures on Q and U maps have the same statistical properties as the ones in Total intensity





Polarization application





How can machine learning help?

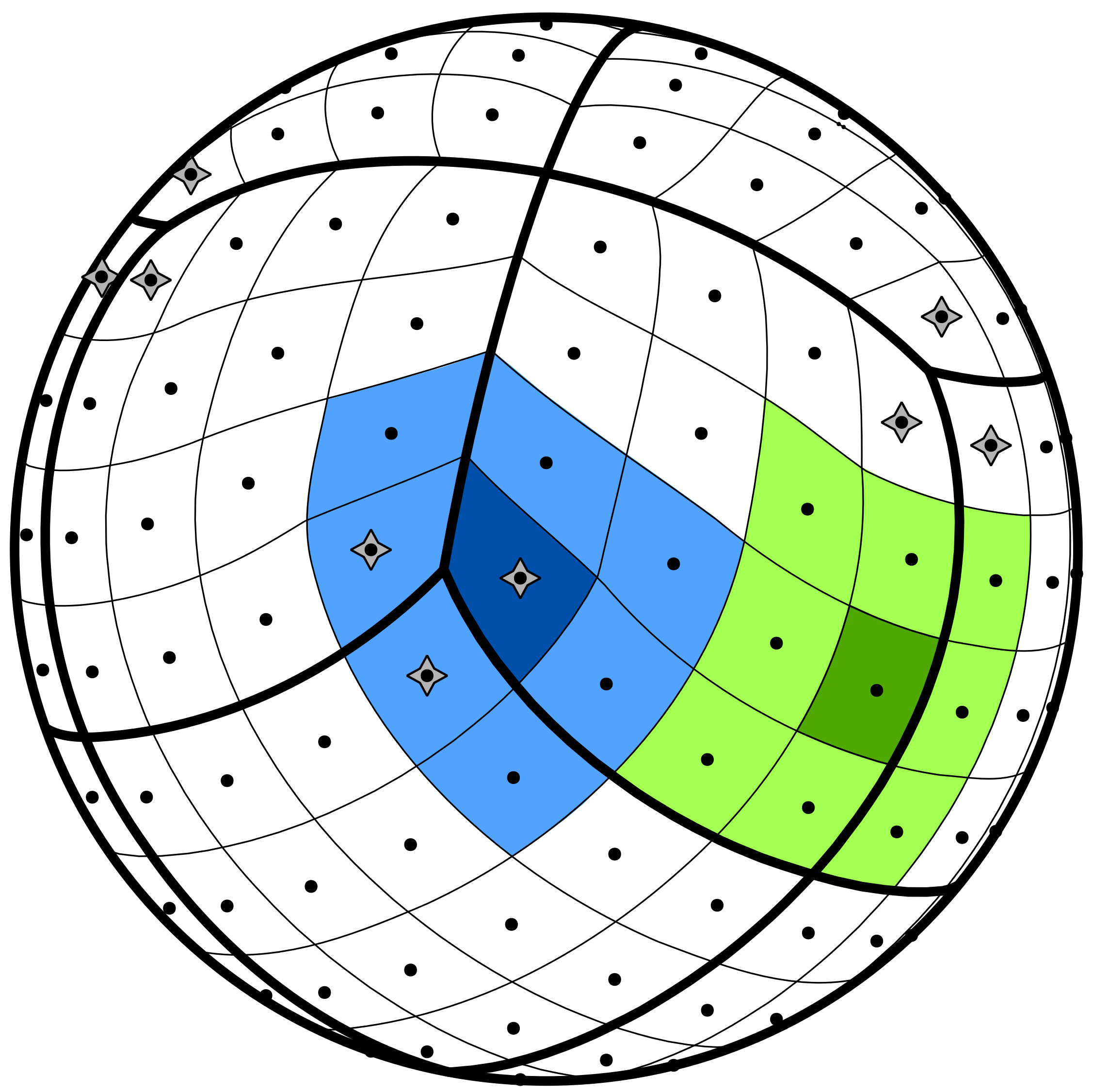
1. Solution to specific, unsolved, problems/tasks: e.g. foreground modeling, instrumental systematic treatment, optimal masking etc…
2. Complement and support classical data analysis techniques: e.g. component separation, **parameter estimation**

Convolutional NN on HEALPix

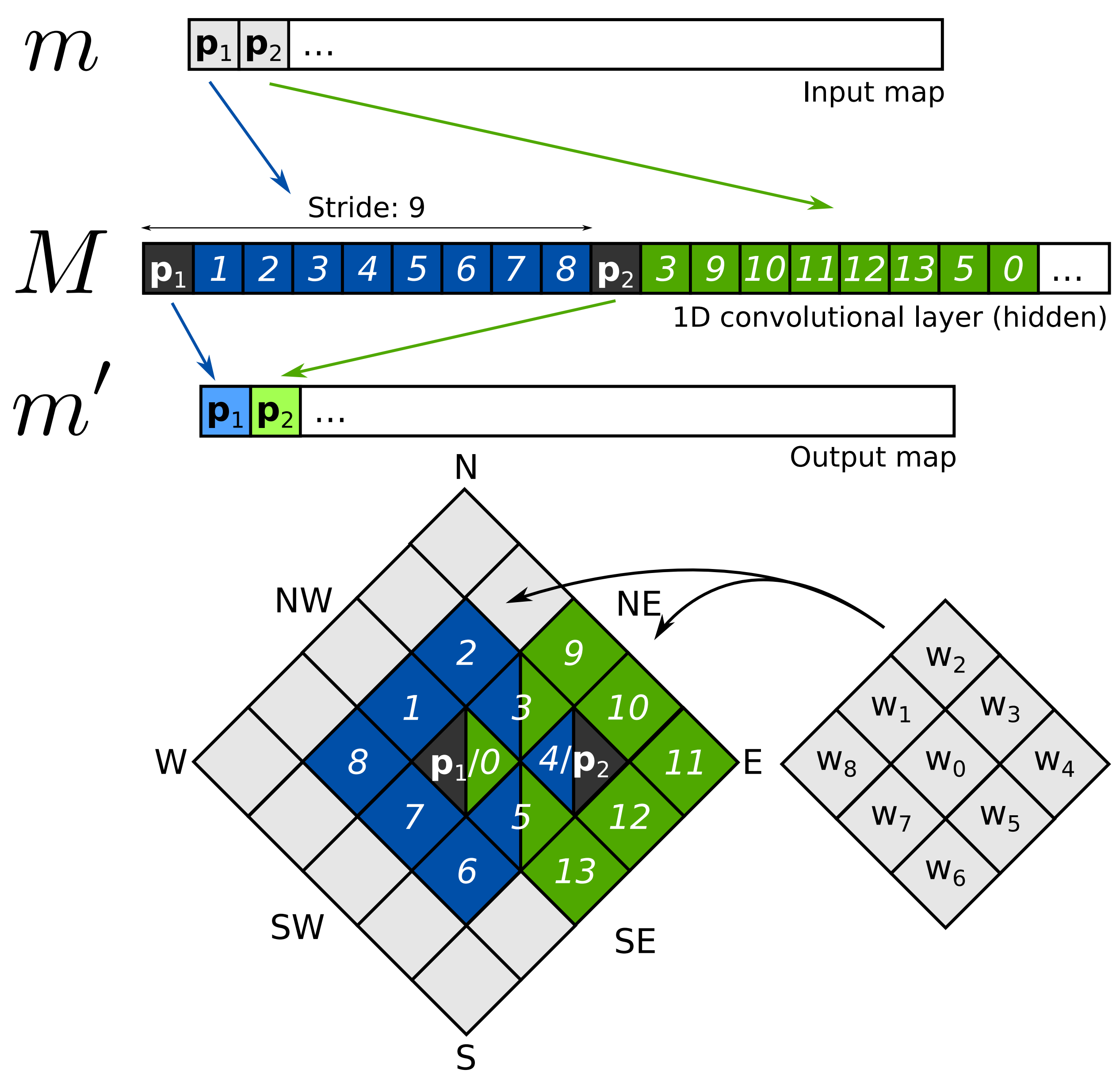
https://github.com/ai4cmb/NNhealpix

Krachmalnicoff, N. & Tomasi, M., A&A, 2019

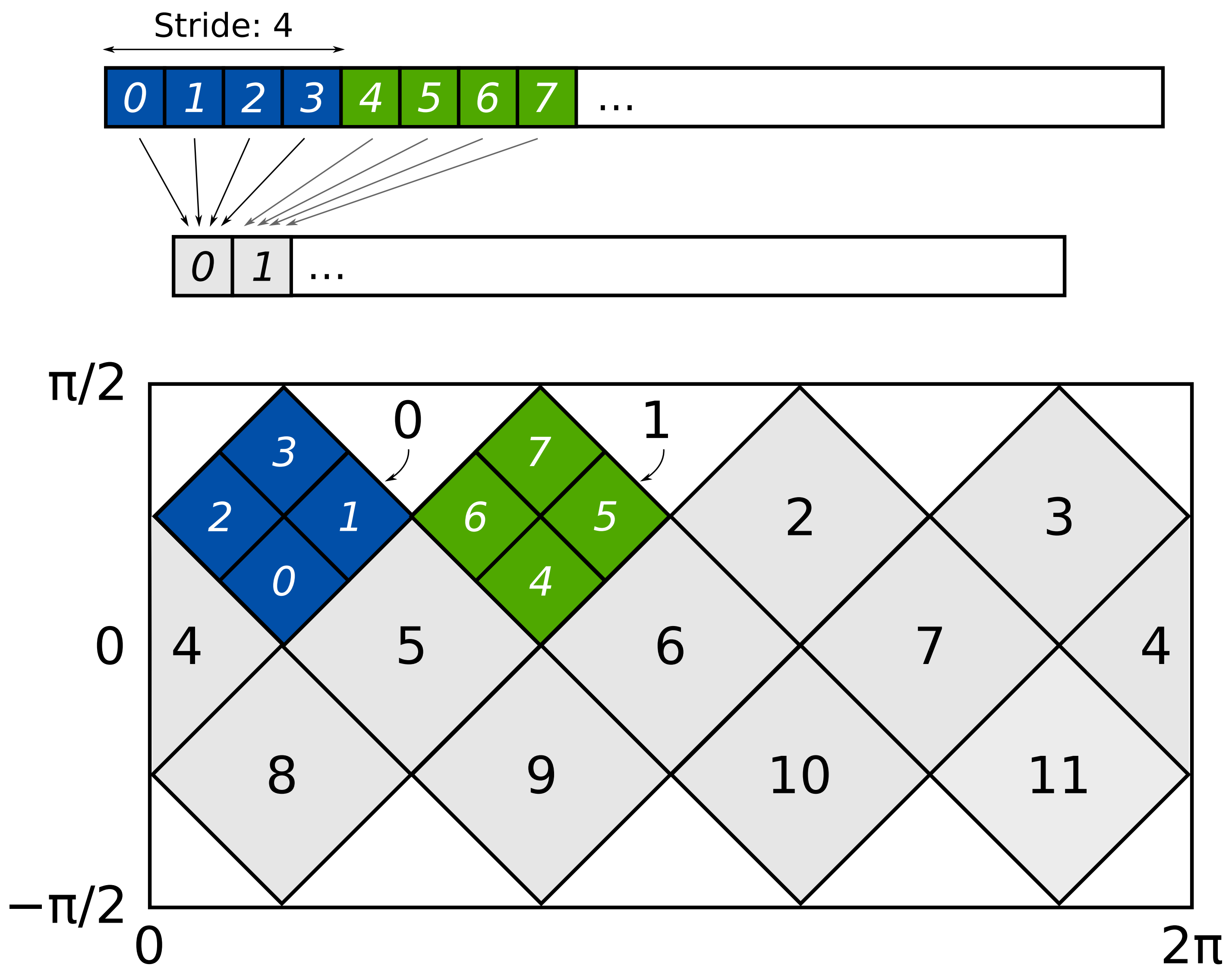
- NN can be used to estimate cosmological parameters from CMB maps
- Valuable tool especially at large angular scales, which are highly contaminated by systematics and foreground. Non Gaussian signal.
- First step: need convolution on the sphere.



CONVOLUTION

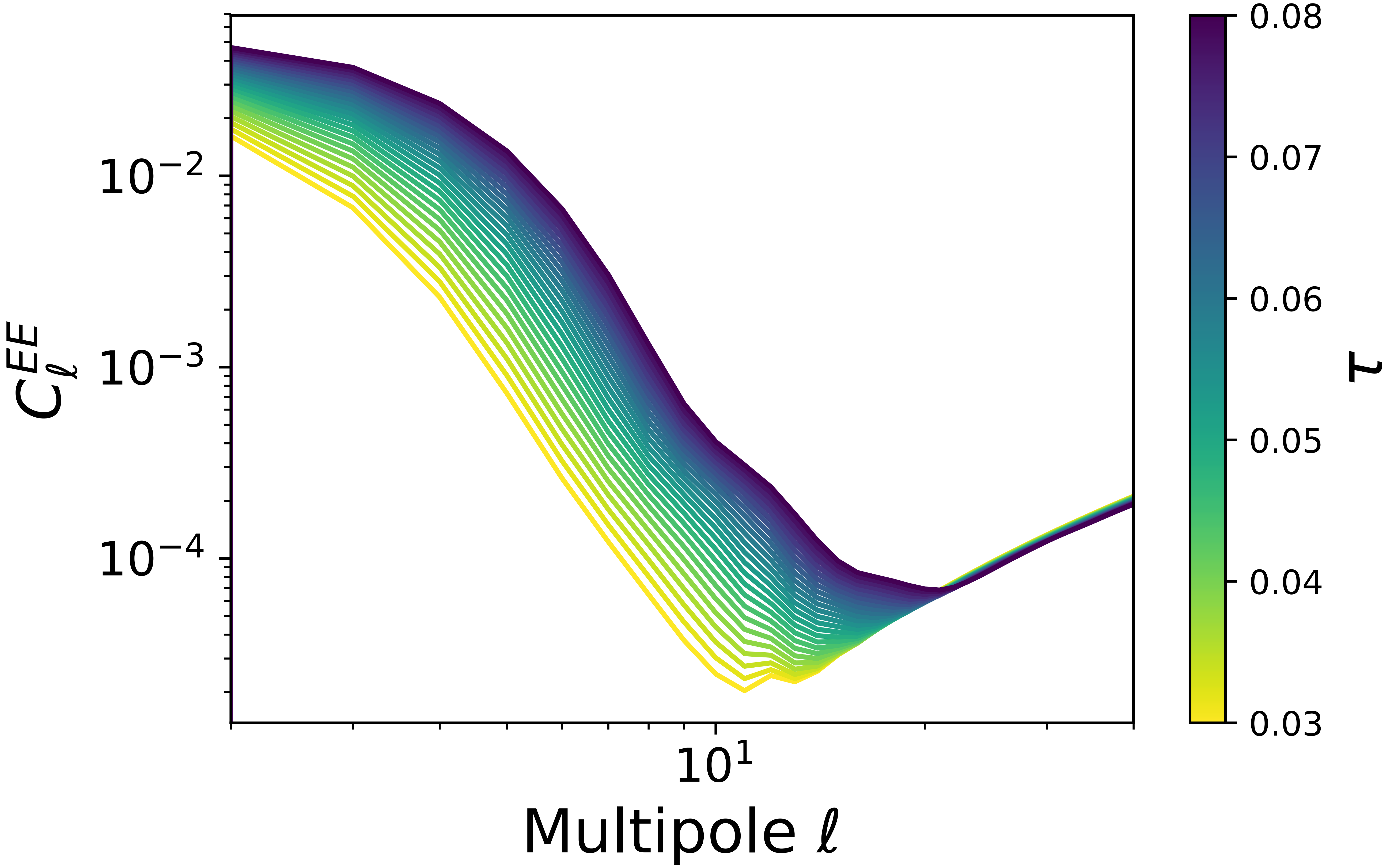


POOLING



Towards τ estimation from Planck data

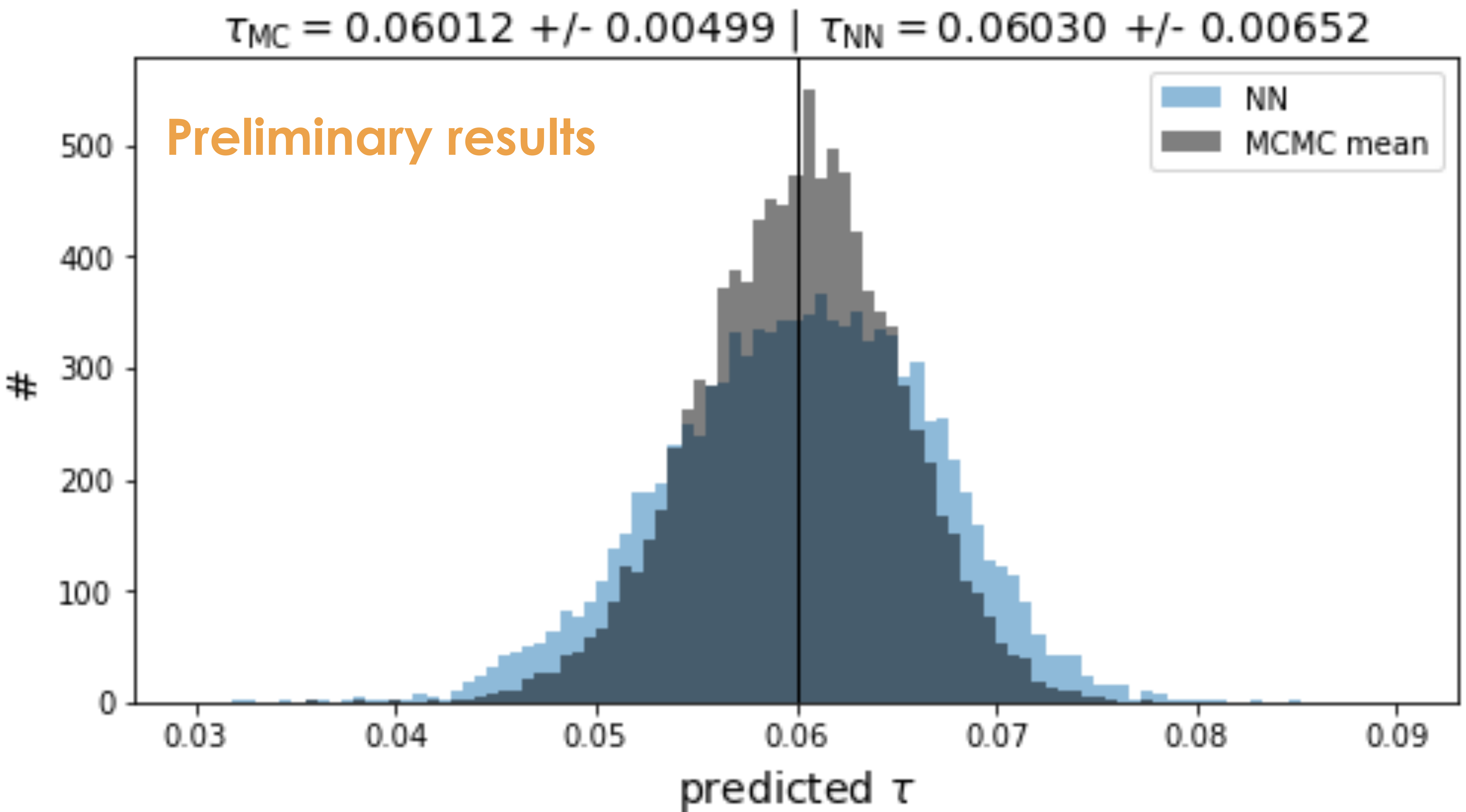
- Optical depth at reionization is one of the most difficult parameter to estimate
- it impacts CMB polarization at very large angular scales (> 20°)
- highly contaminated by foreground and systematics: current constrain $\tau = 0.059 \pm 0.006$ (Pagano et al. 2020)
- Typically a spectrum based likelihood is used
- Can we estimate it directly from maps with CNN?



Towards τ estimation from Planck data

In collaboration with Kevin Wolz (SISSA) and Luca Pagano (UniFe)

- Tested on simulations, with realistic correlated noise and masked sky
- Next step is to train on realistic sims, including foreground residuals and systematics
- **Goal: demonstrate the feasibility of the approach on real and complex data!**



Conclusions:

- The field of Machine Learning (and specifically of Neural Networks) is growing fast…and so does the amount of cosmological data
- Important to understand the role ML can play in the future of Cosmology
- Ideally it has the potential to:
    1. help in computing faster and better simulations
    2. allow a full exploitation of data (summary statistic - free and likelihood - free inference)
- Many preliminary works and proof-of-concepts has been done, with great results
- full end-to-end applications that work on real and complex data are still missing (but I'm confident we will arrive there)