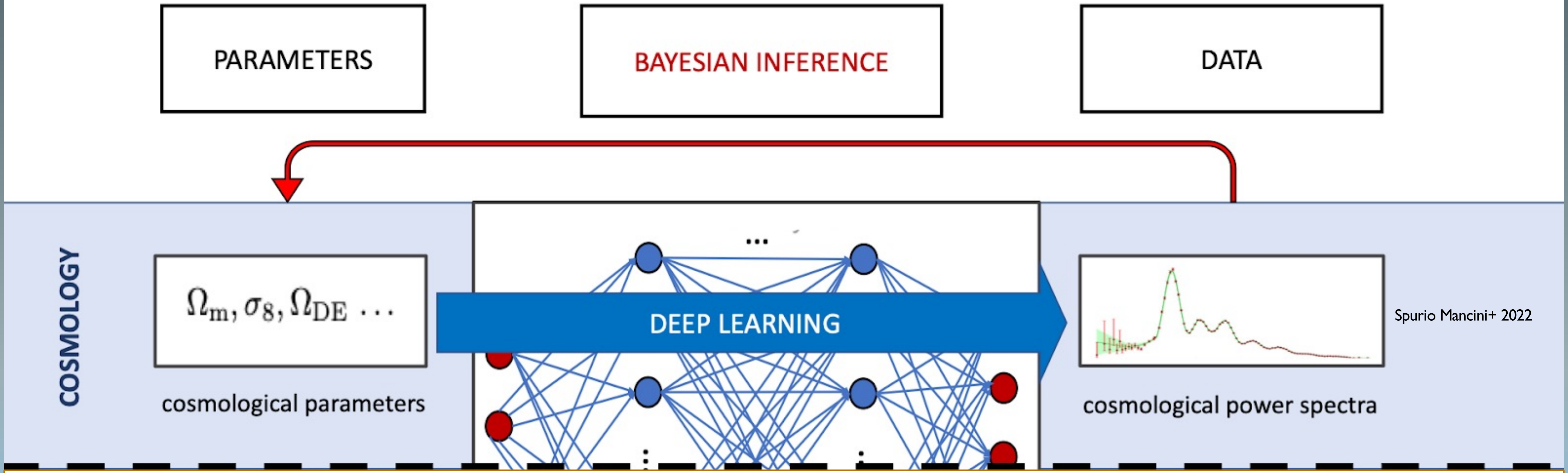
COSMOPOWER:
fully-differentiable Bayesian cosmology with neural emulators

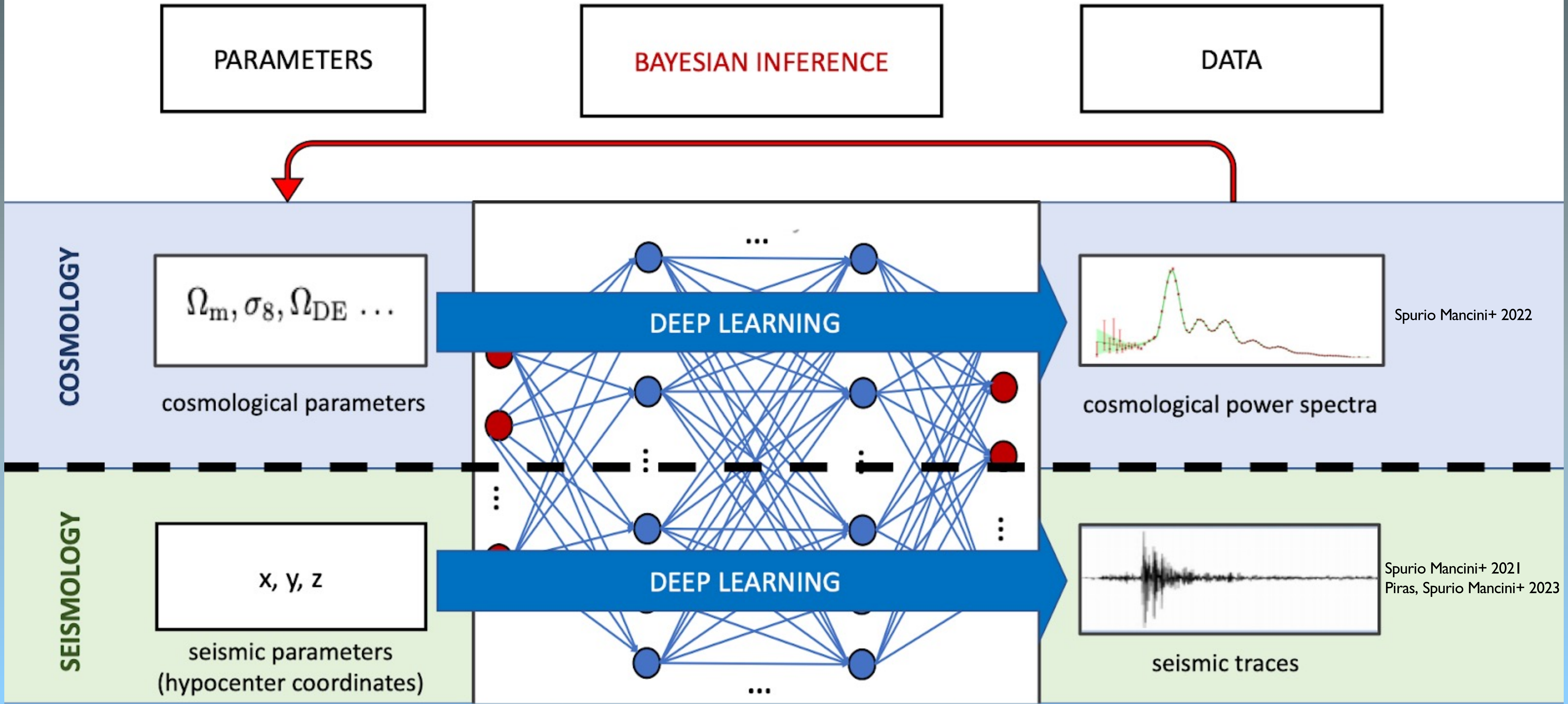
Alessio Spurio Mancini

ROYAL HOLLOWAY UNIVERSITY OF LONDON

COSMOPOWER: DEEP LEARNING FOR ACCELERATED BAYESIAN INFERENCE



COSMOPOWER: DEEP LEARNING FOR ACCELERATED BAYESIAN INFERENCE



DEEP LEARNING FOR ACCELERATED BAYESIAN INFERENCE

IN GEOPHYSICS

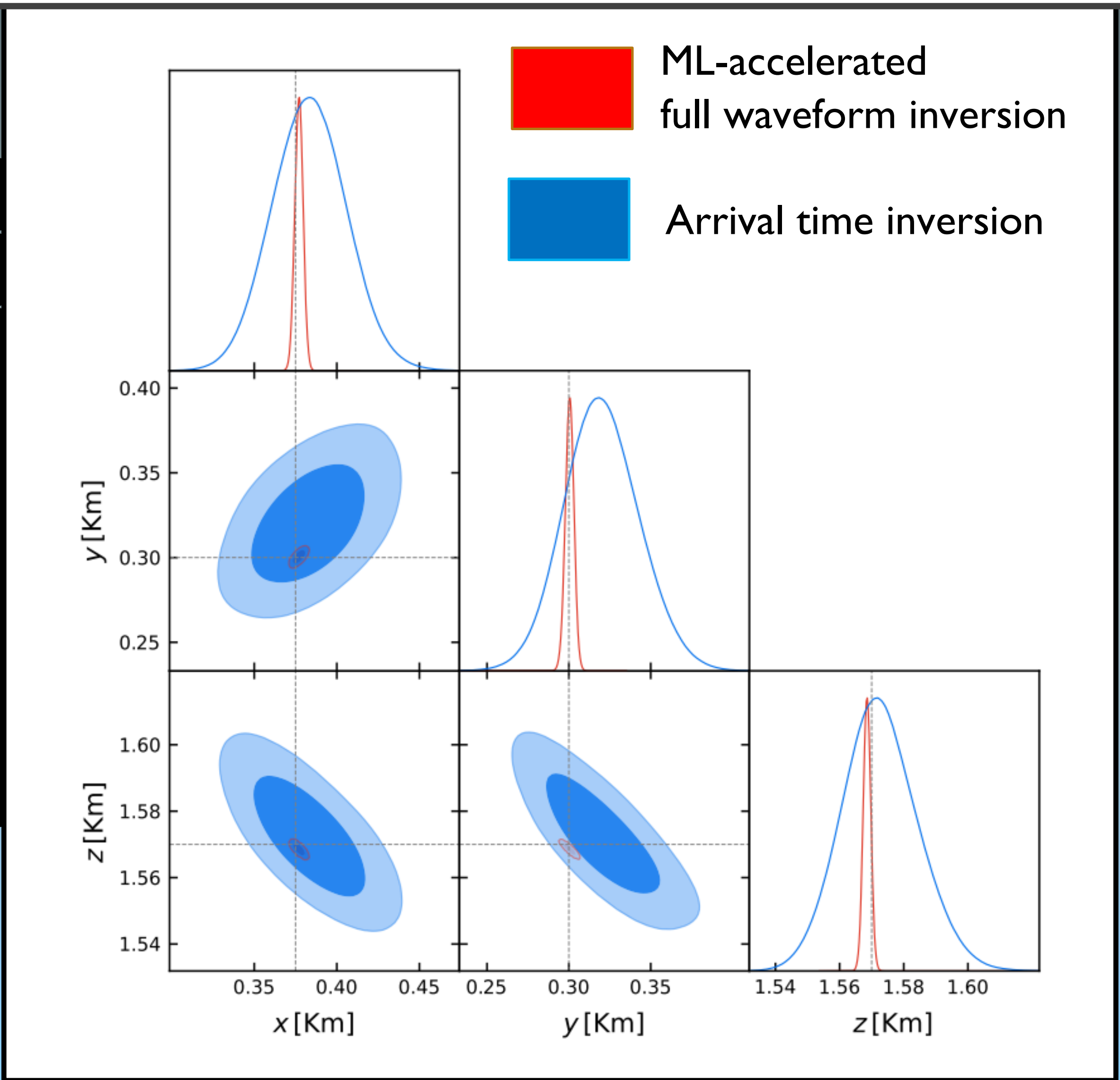
Spurio Mancini+ 2021

Piras, Spurio Mancini+ 2023

From ~ 1h per seismic trace on high-end GPU

→ To ~10 ms on a laptop CPU

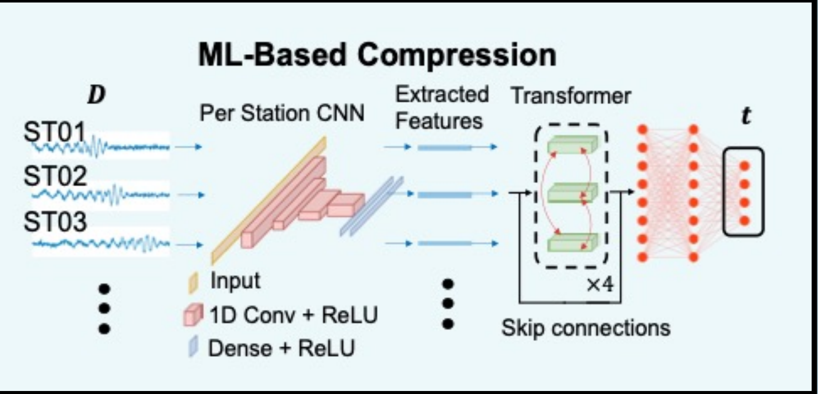
(need ~ $10^4 – 10^6$ evaluations in MCMC)

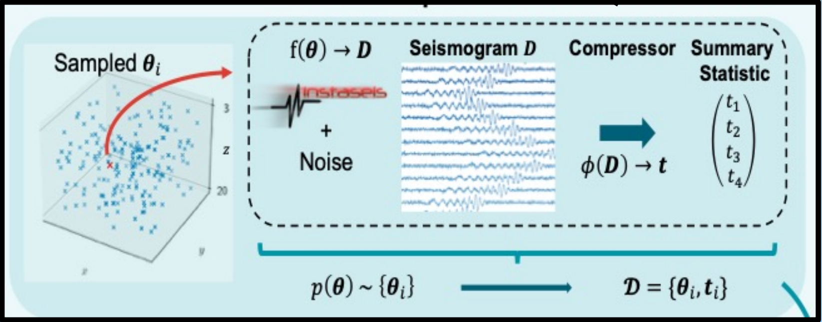


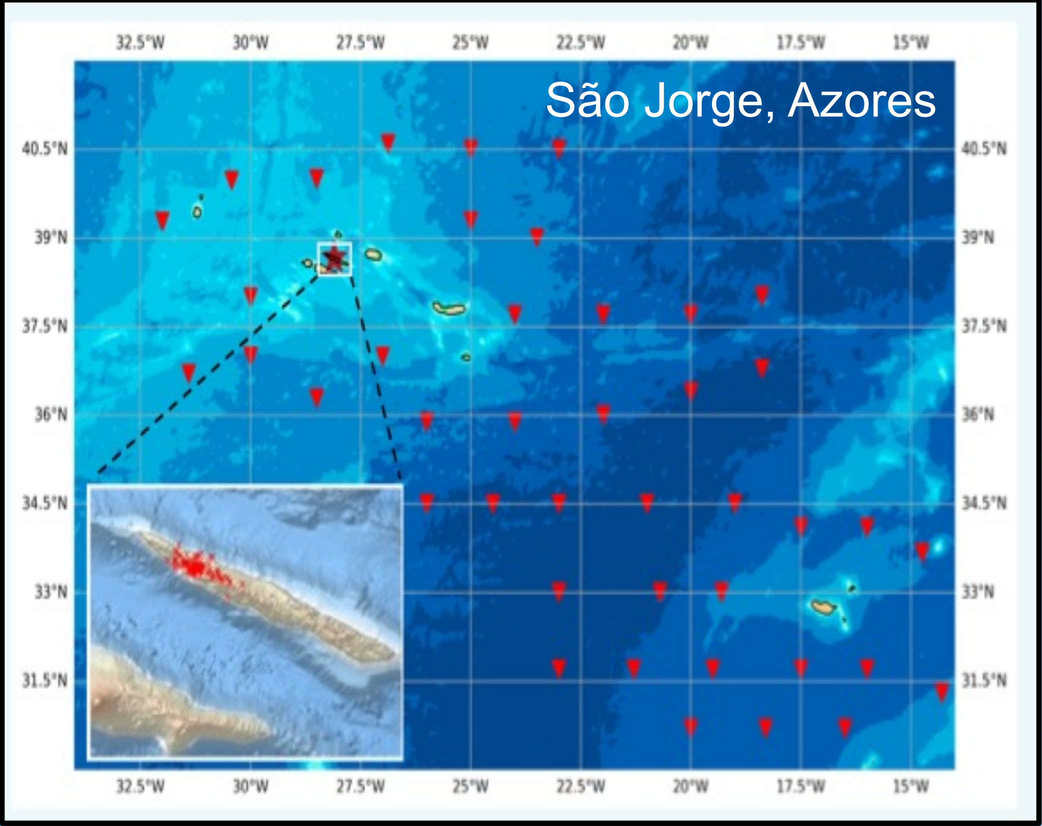
SIMULATION-BASED INFERENCE FOR SEISMIC INVERSION

A. Saoulis, PhD student at UCL

with D. Piras, A. Ferreira, B. Joachimi







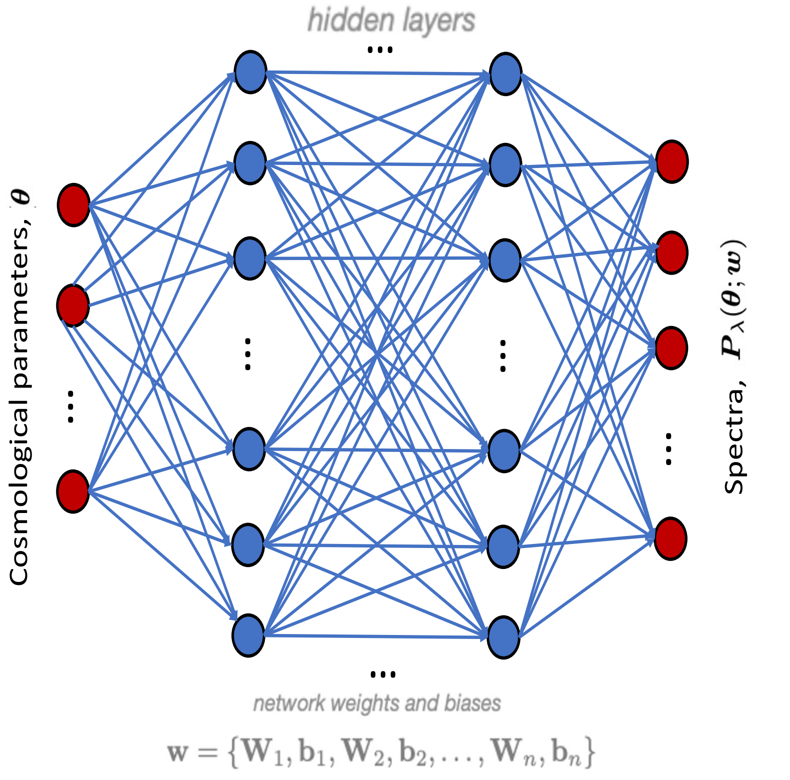
COSMOPOWER

Spurio Mancini+ 2022

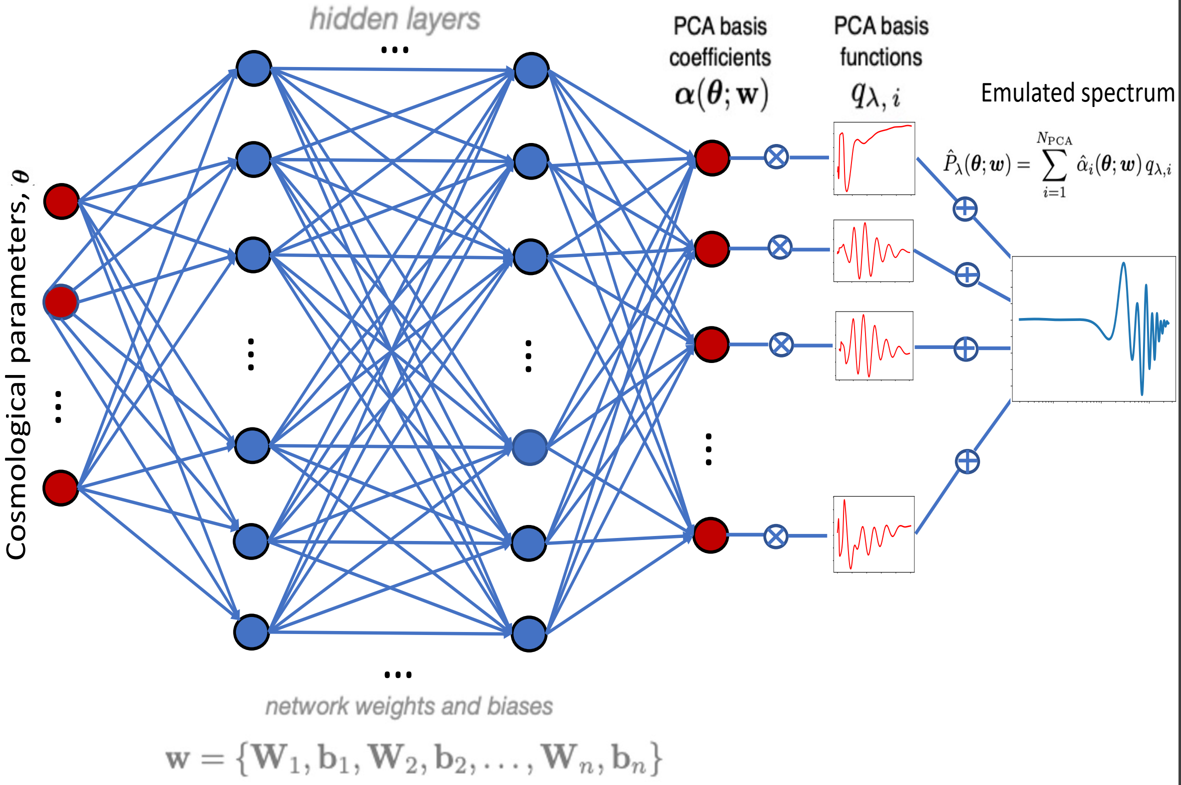
We introduce a suite of
neural cosmological power spectrum emulators
covering both CMB (temperature, polarization and lensing),
and large-scale structure power spectra

COSMOPOWER

Spurio Mancini+ 2022



NEURAL NETWORK



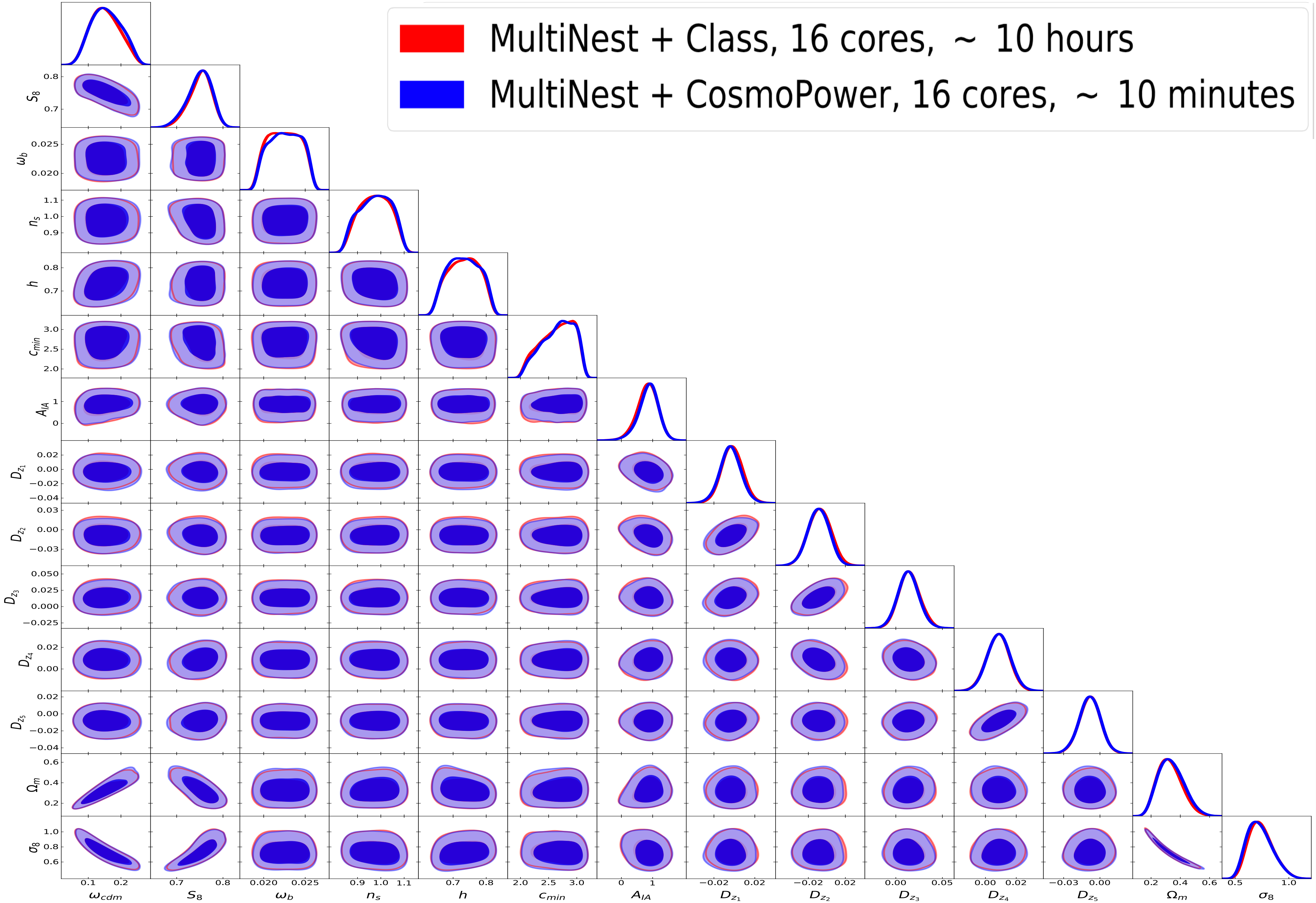
NEURAL NETWORK + PCA

python

GitHub

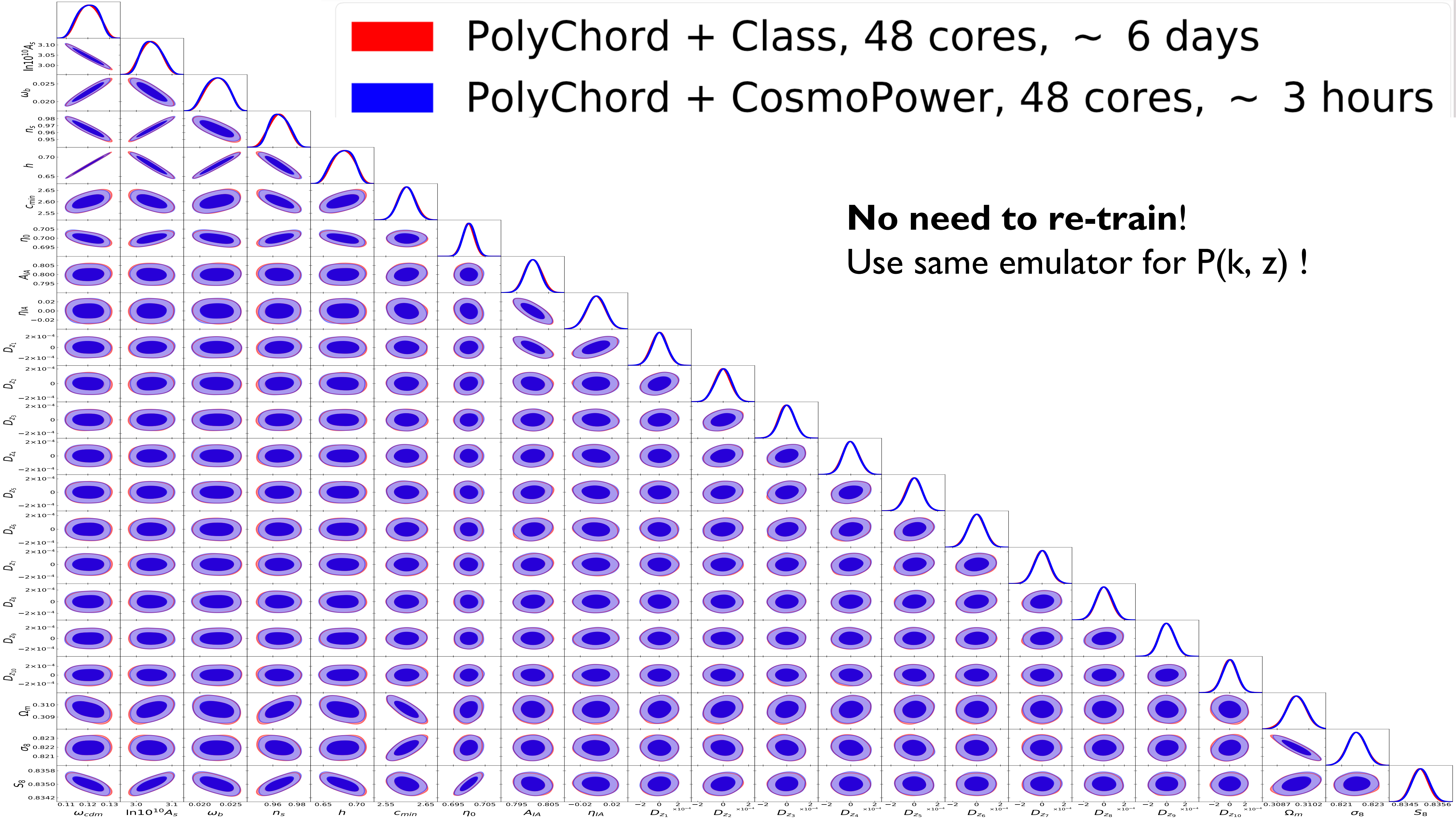
alessiospuriomancini/cosmopower

KIDS-1000 COSMIC SHEAR



MultiNest + Class, 16 cores, ~ 10 hours
MultiNest + CosmoPower, 16 cores, ~ 10 minutes
ω_{cdm}
S_8
ω_b
n_s
h
c_{min}
A_{IA}
D_{z_1}
D_{z_2}
D_{z_3}
D_{z_4}
D_{z_5}
Ω_m
σ_8

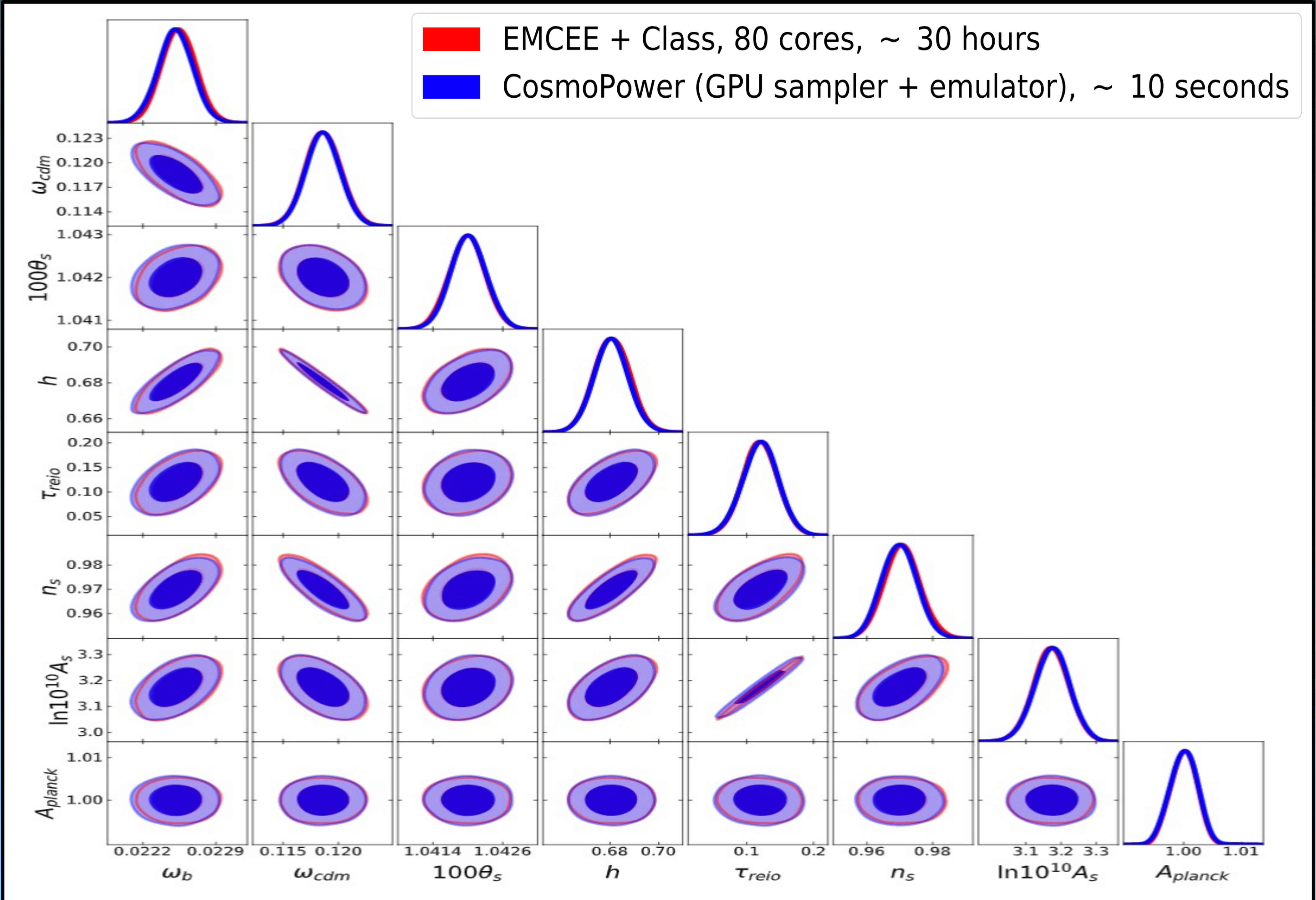
EUCLID-LIKE COSMIC SHEAR



No need to re-train!

Use same emulator for P(k, z) !

PLANCK 2018 TTTEEE


EMCEE + Class, 80 cores, ~ 30 hours
CosmoPower (GPU sampler + emulator), ~ 10 seconds
ω_b
ω_{cdm}
$100\theta_s$
h
τ_{reio}
n_s
$\ln 10^{10} A_s$
A_{planck}
0.0222 0.0229
0.115 0.120
1.0414 1.0426
0.68 0.70
0.1 0.2
0.96 0.98
3.1 3.2 3.3
1.00 1.01
0.123 0.120 0.117 0.114
1.043 1.042 1.041
0.70 0.68 0.66
0.20 0.15 0.10 0.05
0.98 0.97 0.96
3.3 3.2 3.1 3.0
1.01 1.00

COSMOPOWER + JAX =

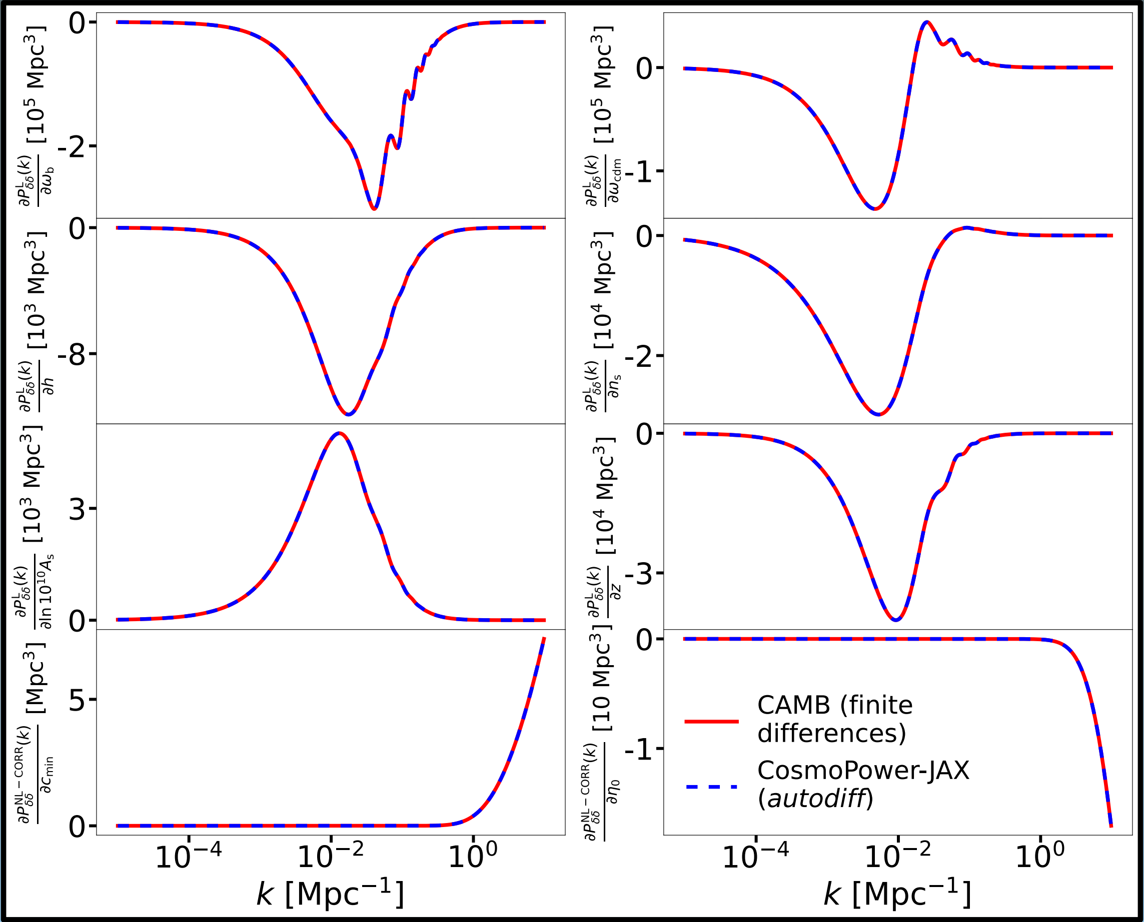
COSMOPOWER-JAX

Piras & Spurio Mancini 2023

python GitHub dpiras/cosmopower-jax

COSMOPOWER-JAX

Piras & Spurio Mancini 2023

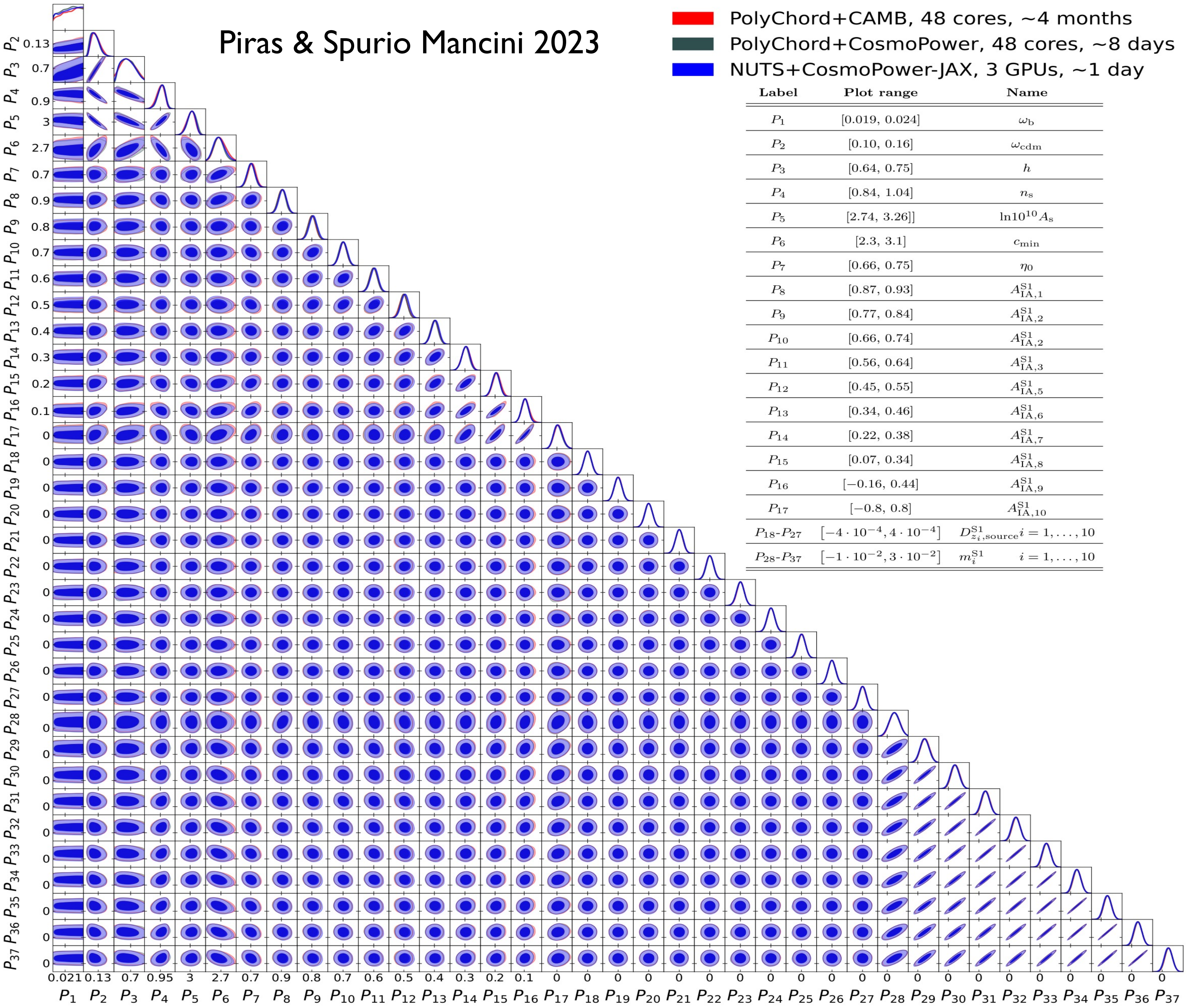


python

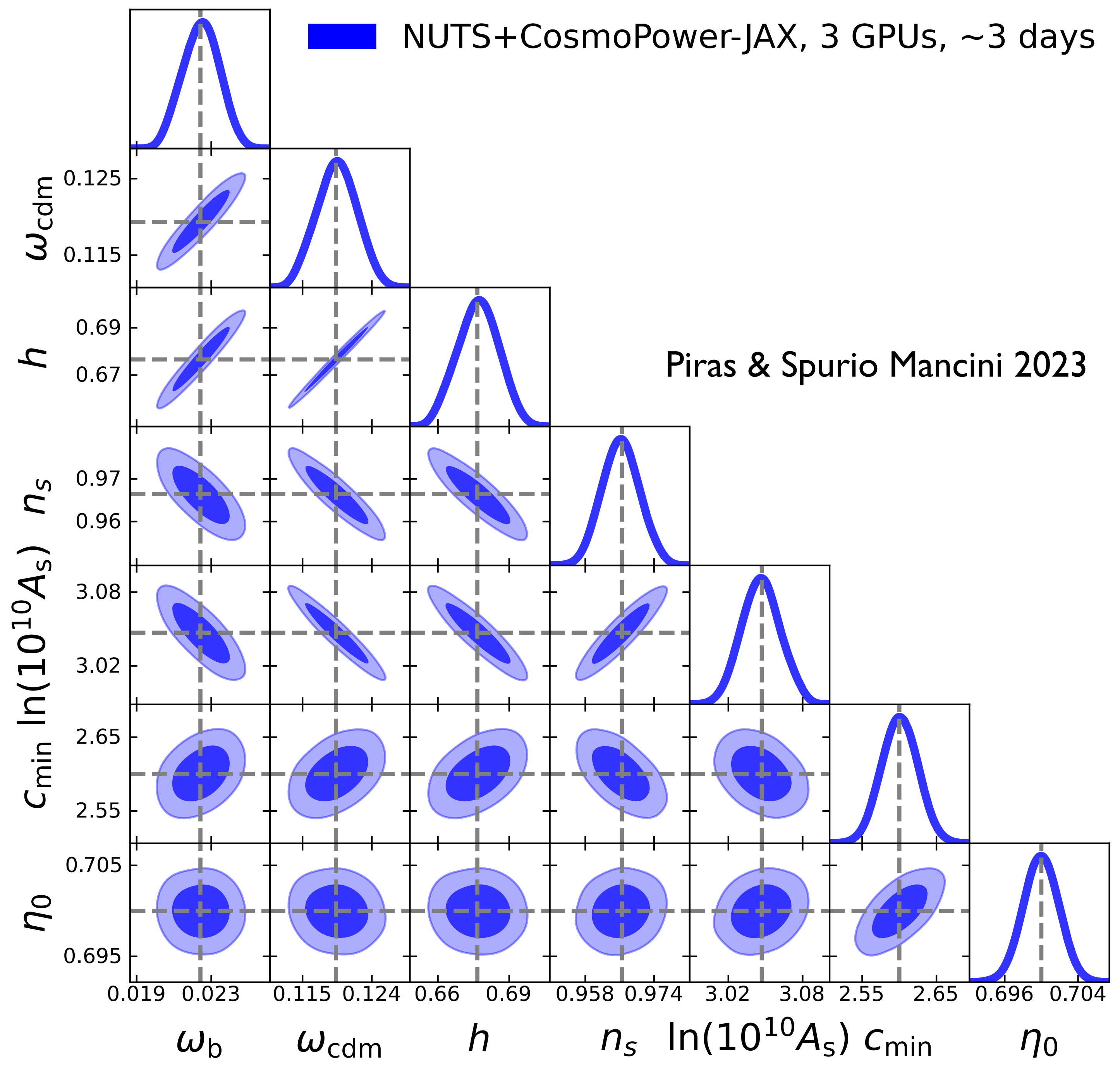
GitHub

dpiras/cosmopower-jax

Piras & Spurio Mancini 2023

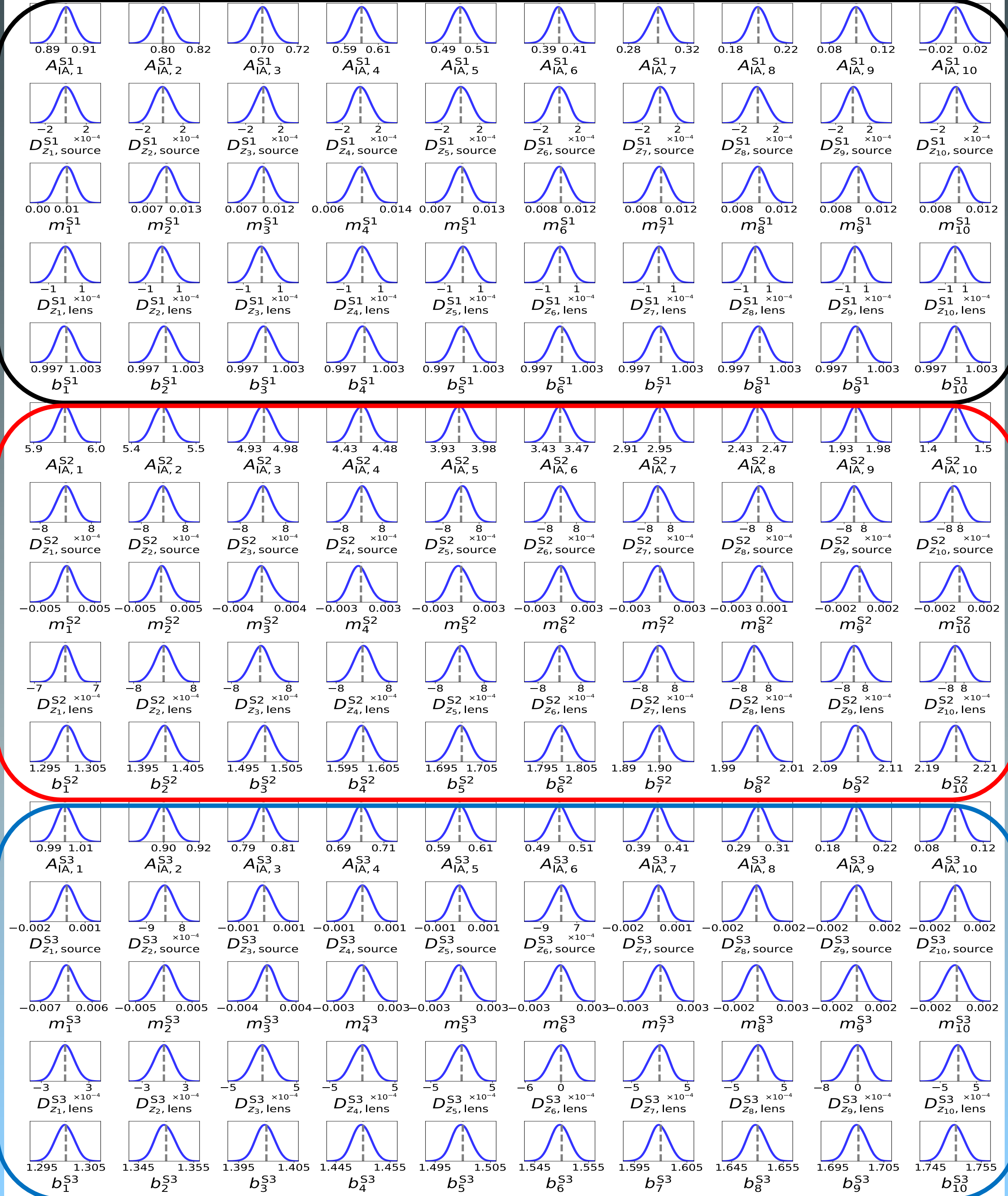


Label	Plot range	Name
P_1	[0.019, 0.024]	$\omega_{\rm b}$
P_2	[0.10, 0.16]	$\omega_{\rm cdm}$
P_3	[0.64, 0.75]	h
P_4	[0.84, 1.04]	$n_{\rm s}$
P_5	[2.74, 3.26]]	$\ln 10^{10} A_{\rm s}$
P_6	[2.3, 3.1]	$c_{\rm min}$
P_7	[0.66, 0.75]	η_0
P_8	[0.87, 0.93]	$A^{\rm S1}_{\rm IA,1}$
P_9	[0.77, 0.84]	$A^{\rm S1}_{\rm IA,2}$
P_{10}	[0.66, 0.74]	$A^{\rm S1}_{\rm IA,2}$
P_{11}	[0.56, 0.64]	$A^{\rm S1}_{\rm IA,3}$
P_{12}	[0.45, 0.55]	$A^{\rm S1}_{\rm IA,5}$
P_{13}	[0.34, 0.46]	$A^{\rm S1}_{\rm IA,6}$
P_{14}	[0.22, 0.38]	$A^{\rm S1}_{\rm IA,7}$
P_{15}	[0.07, 0.34]	$A^{\rm S1}_{\rm IA,8}$
P_{16}	[−0.16, 0.44]	$A^{\rm S1}_{\rm IA,9}$
P_{17}	[−0.8, 0.8]	$A^{\rm S1}_{\rm IA,10}$
P_{18}-P_{27}	$[-4\cdot10^{-4}, 4\cdot10^{-4}]$	$D^{\rm S1}_{z_i,\rm source} i = 1,\ldots,10$
P_{28}-P_{37}	$[-1\cdot10^{-2}, 3\cdot10^{-2}]$	$m^{\rm S1}_i \quad i = 1,\ldots,10$


NUTS+CosmoPower-JAX, 3 GPUs, ~3 days
Piras & Spurio Mancini 2023
$\omega_{\rm cdm}$
h
n_s
$\ln(10^{10}A_{\rm s})$
$c_{\rm min}$
η_0
$\omega_{\rm b}$
0.125
0.115
0.69
0.67
0.97
0.96
3.08
3.02
2.65
2.55
0.705
0.695
0.019 0.023
0.115 0.124
0.66 0.69
0.958 0.974
3.02 3.08
2.55 2.65
0.696 0.704

Piras & Spurio Mancini 2023

- COSMOPOWER-JAX (Piras & Spurio Mancini 2023) + JAX-COSMO (Campagne+ 2023)
- 3 Stage IV surveys → 157 (!) parameters
- 3 days on 3 GPUs with NUTS
- (Optimistic) estimate: 6 years (!!) on 48 CPUs with PolyChord



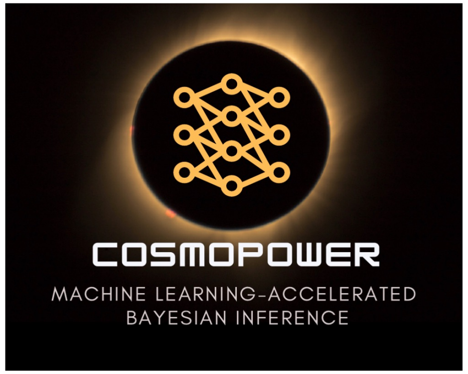
Survey 1

Survey 2

Survey 3

https://github.com/alessiospuriomancini/cosmopower

README.md



Python | Tensorflow | License GPLv3 | Author Alessio Spurio Mancini | Installation pip install cosmopower

Overview • Documentation • Installation • Getting Started • Training • Trained Models • Likelihoods • Support • Citation

```
pip install cosmopower
```

```python
import cosmopower as cp

# load pre-trained NN model: maps cosmological parameters to CMB TT log-C_ell
cp_nn = cp.cosmopower_NN(restore=True,
                         restore_filename='/path/to/cosmopower'\
                         +'/cosmopower/trained_models/CP_paper/CMB/cmb_TT_NN')

# create a dict of cosmological parameters
params = {'omega_b': [0.0225],
          'omega_cdm': [0.113],
          'h': [0.7],
          'tau_reio': [0.055],
          'n_s': [0.96],
          'ln10^{10}A_s': [3.07],
          }

# predictions (= forward pass through the network) -> 10^predictions
spectra = cp_nn.ten_to_predictions_np(params)
```

getting_started_with_cosmopower_NN.ipynb

File Edit View Insert Runtime Tools Help Changes will not be saved

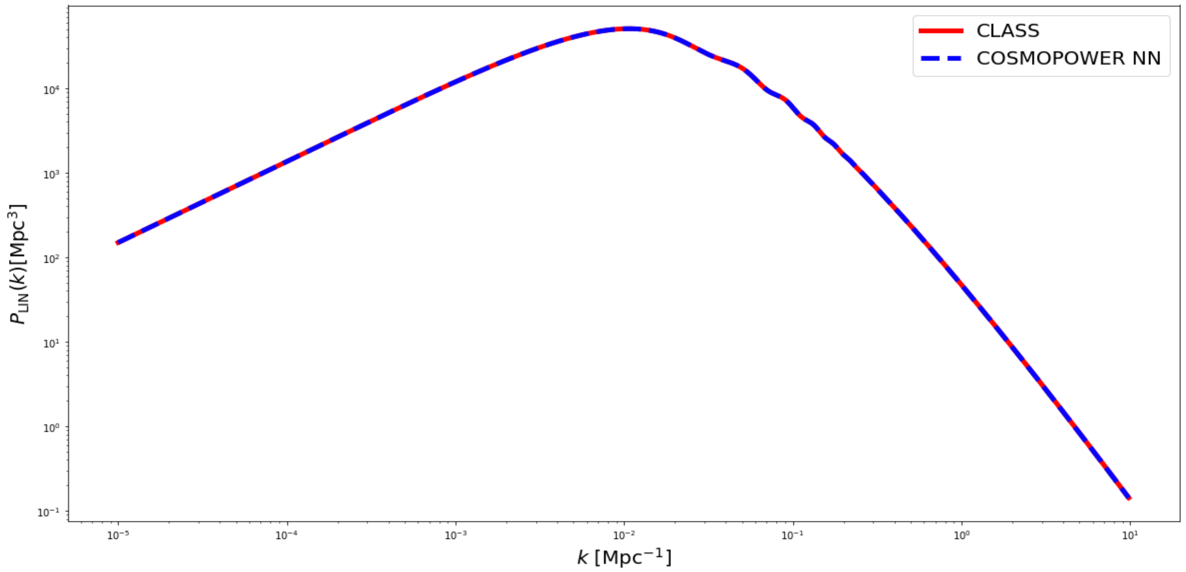
+ Code + Text Copy to Drive

COMPARING with CLASS

```python
k_modes = cp_nn.modes

cosmo = Class()

# Define your cosmology (what is not specified will be set to CLASS default parameters)
params = {'output': 'tCl mPk',
          'z_max_pk': 5,
          'P_k_max_1/Mpc': 10.,
          'nonlinear_min_k_max': 100.,
          'N_ncdm' : 0,
          'N_eff' : 3.046,
          'omega_b': 0.0225,
          'omega_cdm': 0.113,
          'h': 0.7,
          'n_s': 0.96,
          'ln10^{10}A_s': 3.07,
          }

# Set the parameters to the cosmological code
cosmo.set(params)
cosmo.compute()

z = 0.5

spectrum_class = np.array([cosmo.pk(ki, z) for ki in k_modes])
```

```python
pred = spectrum_cosmopower_NN
true = spectrum_class
fig = plt.figure(figsize=(20,10))
plt.loglog(k_modes, true, 'red', linewidth=5, label = 'CLASS')
plt.loglog(k_modes, pred, 'blue', label = 'COSMOPOWER NN', linewidth=5, linestyle='--')
plt.xlabel('$k$ [Mpc$^{-1}]$', fontsize=20)
plt.ylabel('$P_{\mathrm{LIN}}(k) [\mathrm{Mpc}^3]$', fontsize=20)
plt.legend(fontsize=20)
```

```
<matplotlib.legend.Legend at 0x7f25dd360fd0>
```



COLLABORATIONS USING COSMOPOWER

Kilo-Degree Survey

Burger+ 24
Burger+ 23

Dark Energy Survey

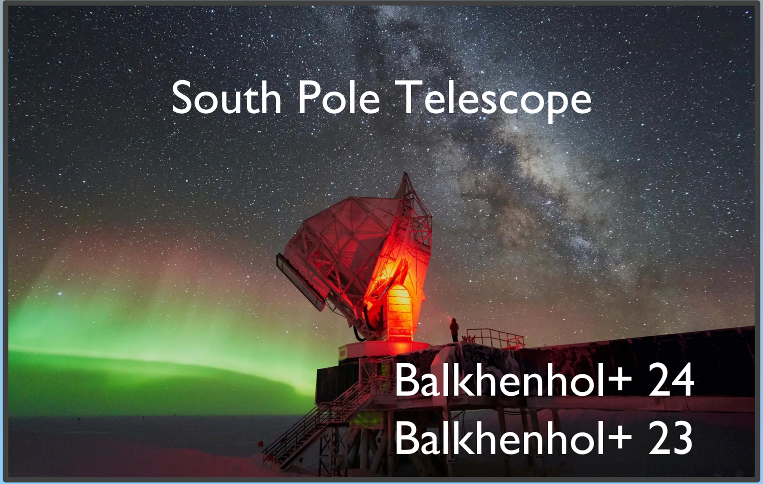
Halder+ 24
Gong+ 23

Euclid

Tsedrik+ 24
Burger+ 23
Linke+ 23
Bose+ 23
Heydenreich+ 22

Atacama Cosmology Telescope

Qu+ 24a,b
Farren+ 23
Bolliet+ 23



South Pole Telescope

Balkhenhol+ 24
Balkhenhol+ 23

Simons Observatory

Zubeldia+ 24
Giardiello+ 24
Patki+ 23
Bolliet+ 23

HARMONIC:
BAYESIAN MODEL COMPARISON
FOR SIMULATION-BASED INFERENCE

LEARNT HARMONIC MEAN ESTIMATOR

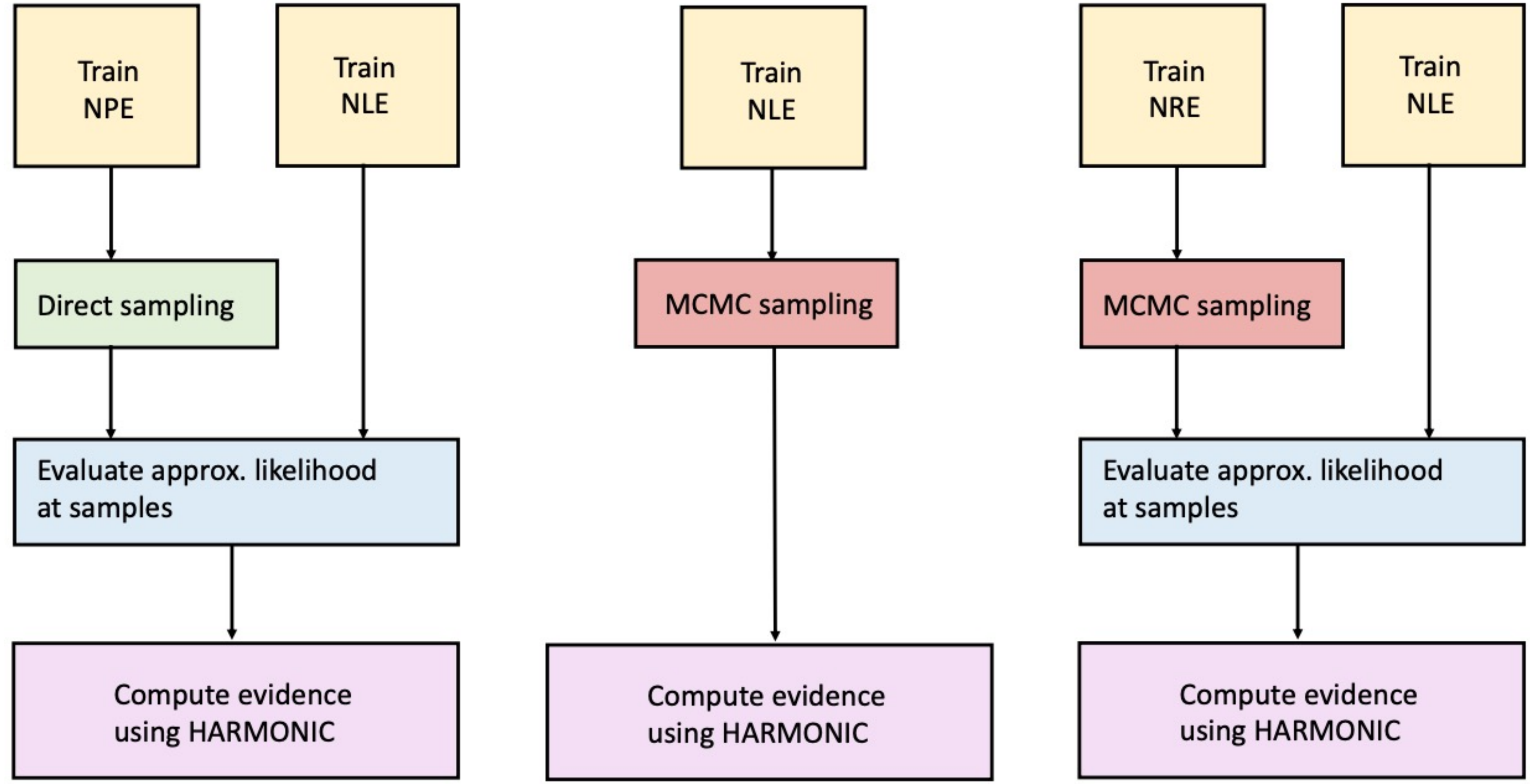
Learn a (mildly accurate) approximation of the optimal target distribution: (McEwen, Wallis, Price, Spurio Mancini 21)

$$\boxed{\varphi(\theta) \overset{\mathsf{ML}}{\simeq} \varphi^{\mathsf{optimal}}(\theta) = \frac{\mathcal{L}(\theta)\pi(\theta)}{z}}$$

$\longrightarrow$ ***agnostic* to sampling strategy !**

`astro-informatics/harmonic`

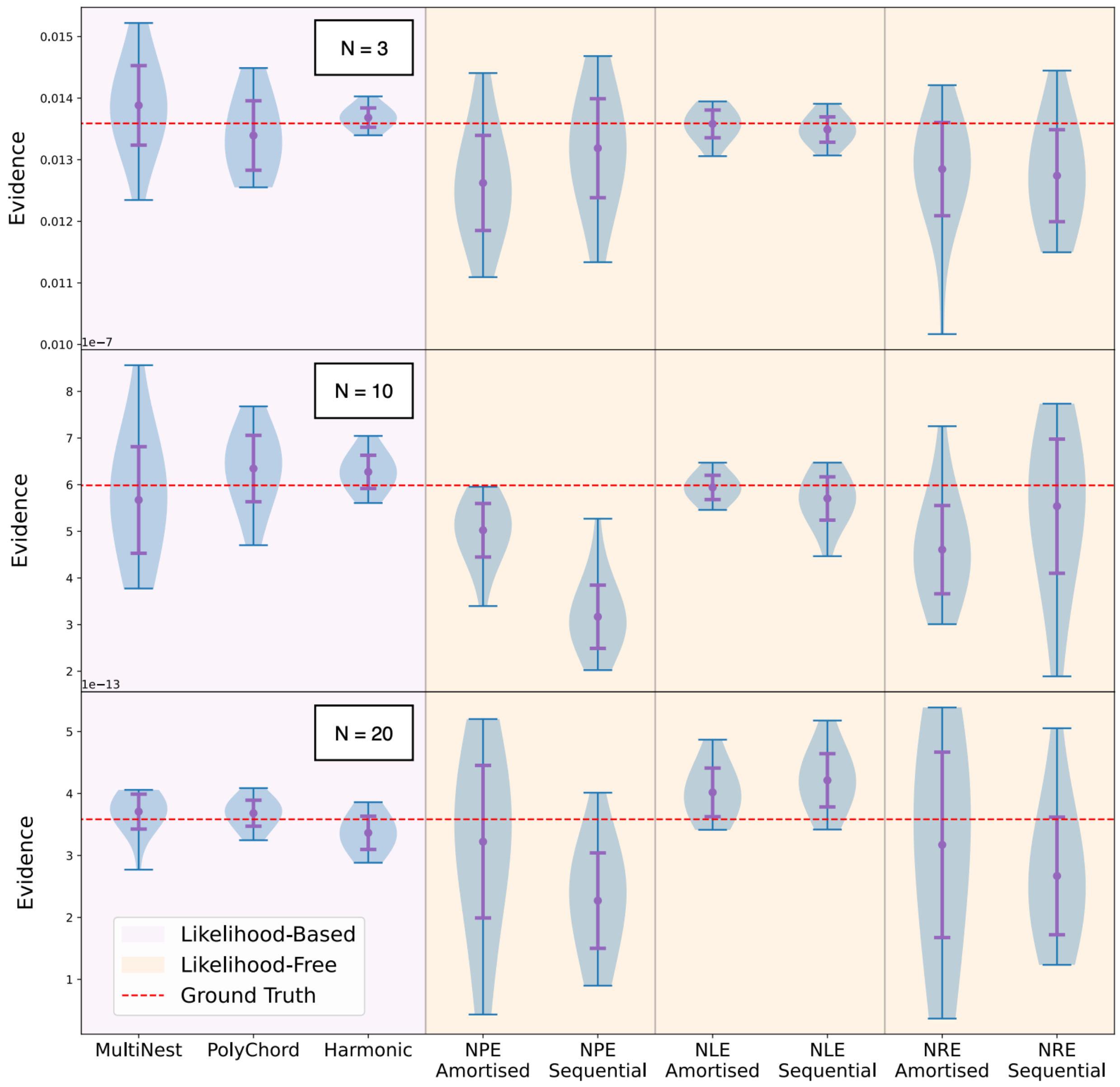
HARMONIC for SBI: Spurio Mancini+ 23



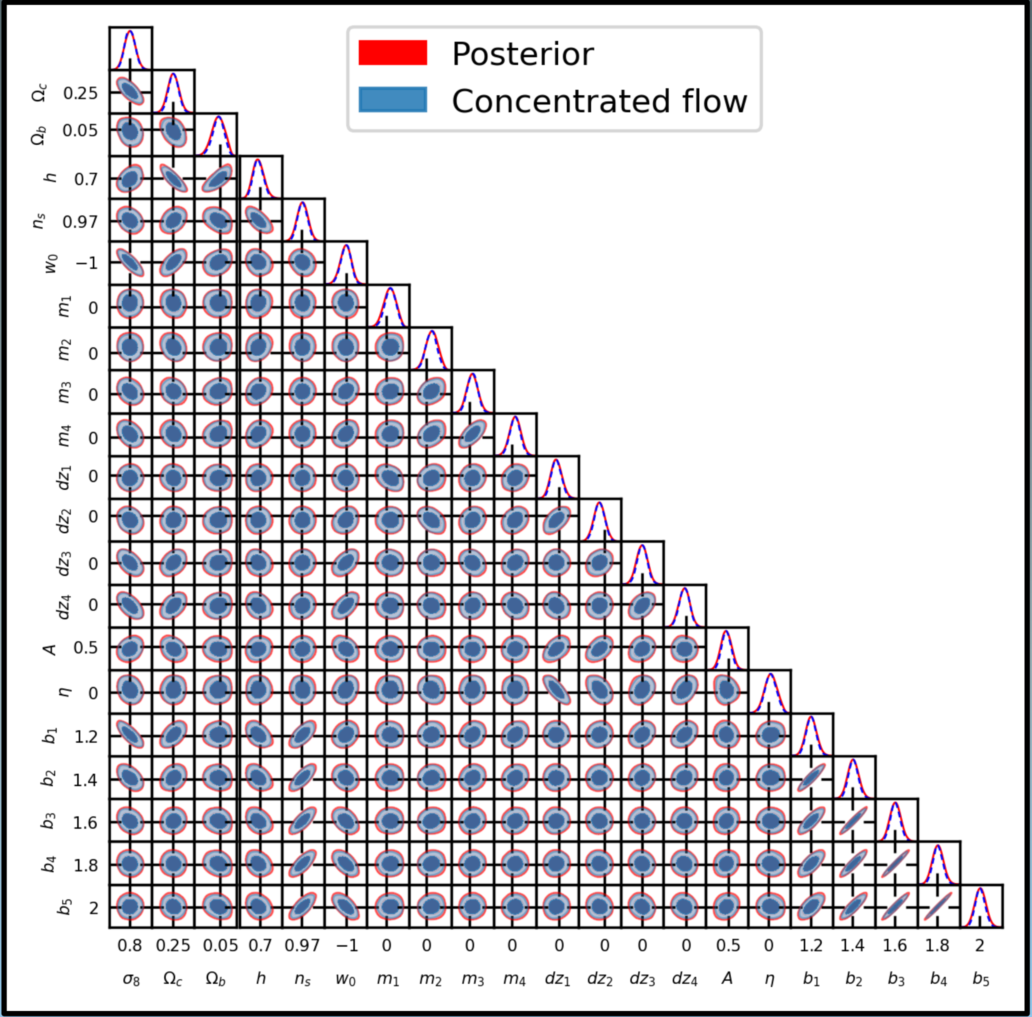
(a) Neural posterior estimation (NPE) (b) Neural likelihood estimation (NLE) (c) Neural ratio estimation (NRE)

LINEAR GAUSSIAN

$$d_i = \theta_i + \mathcal{N}(0,1), \quad i = 1 \ldots D$$



DES Y1 3x2pt



LCDM vs wCDM

Our estimate:

$\log \mathrm{BF} = 2.15 \pm 0.02$ (8h on 128 CPUs)

Nested sampling:

$\log \mathrm{BF} = 2.23 \pm 0.64$ (46h on 128 CPUs)

Polanska, Price, Spurio Mancini, McEwen 23
Polanska, Price, Piras, Spurio Mancini, McEwen, in prep.

Use Normalising Flows
(real NVP, Rational Quadratic Spline)
to learn target distribution

A DIFFERENTIABLE FUTURE FOR COSMOLOGY

- COSMOPOWER: orders-of-magnitude speed up to parameter estimation pipeline

→ All major international CMB and LSS collaborations are using it

- HARMONIC: sampling-agnostic method for LBI & LFI evidence estimation

- COSMOPOWER + HARMONIC: high-dimensional, differentiable parameter estimation + model comparison

GitHub

alessiospuriomancini/cosmopower
dpiras/cosmopower-jax
astro-informatics/harmonic

alessio.spuriomancini@rhul.ac.uk