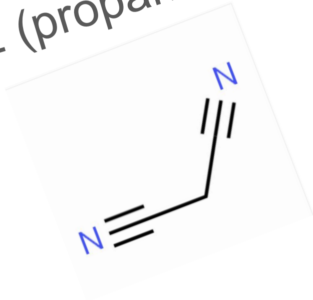
# C3NN:

# Cosmological Correlator Convolutional Neural Network

## an interpretable (explainable?) ML framework for weak lensing analyses

Zhengyangguang (Laurence) Gong (USM, MPE)
with Anik Halder, Annabelle Bohrdt, Stella Seitz and David Gebauer
(https://arxiv.org/abs/2402.09526)

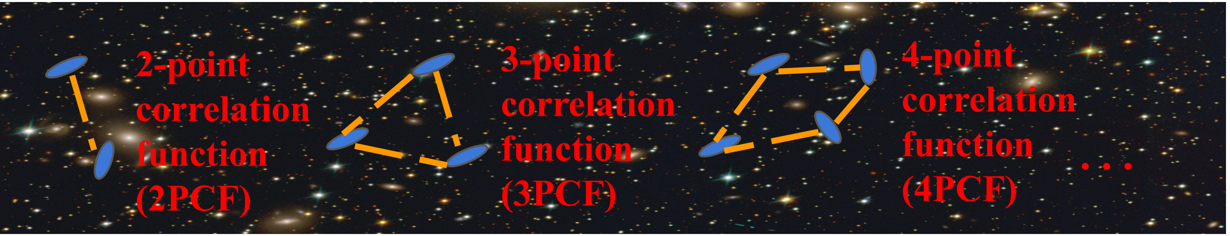
or C3N2 (propanedinitrile)?



New Strategies for Extracting Cosmology from Galaxy Surveys
Sexten Center for Astrophysics

04.07.2024

# Motivation: information in the N-point ladder



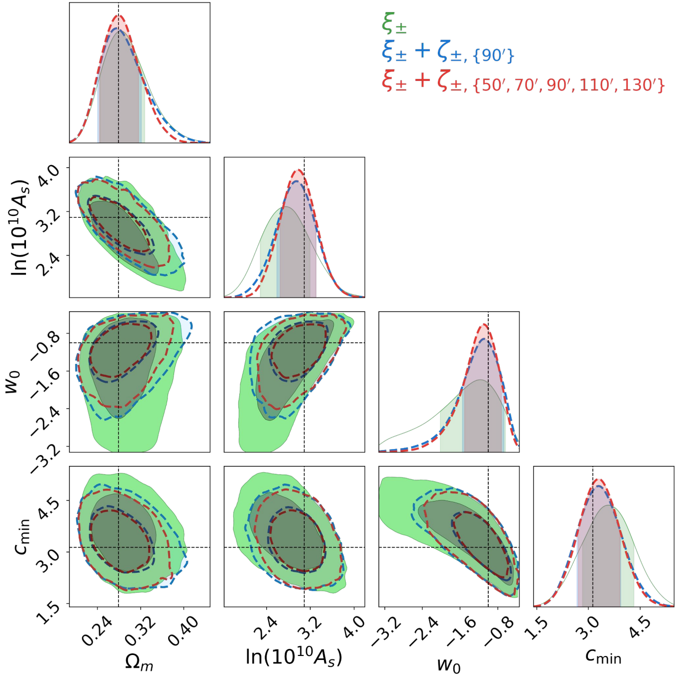
2-point correlation function (2PCF)

3-point correlation function (3PCF)

4-point correlation function (4PCF)

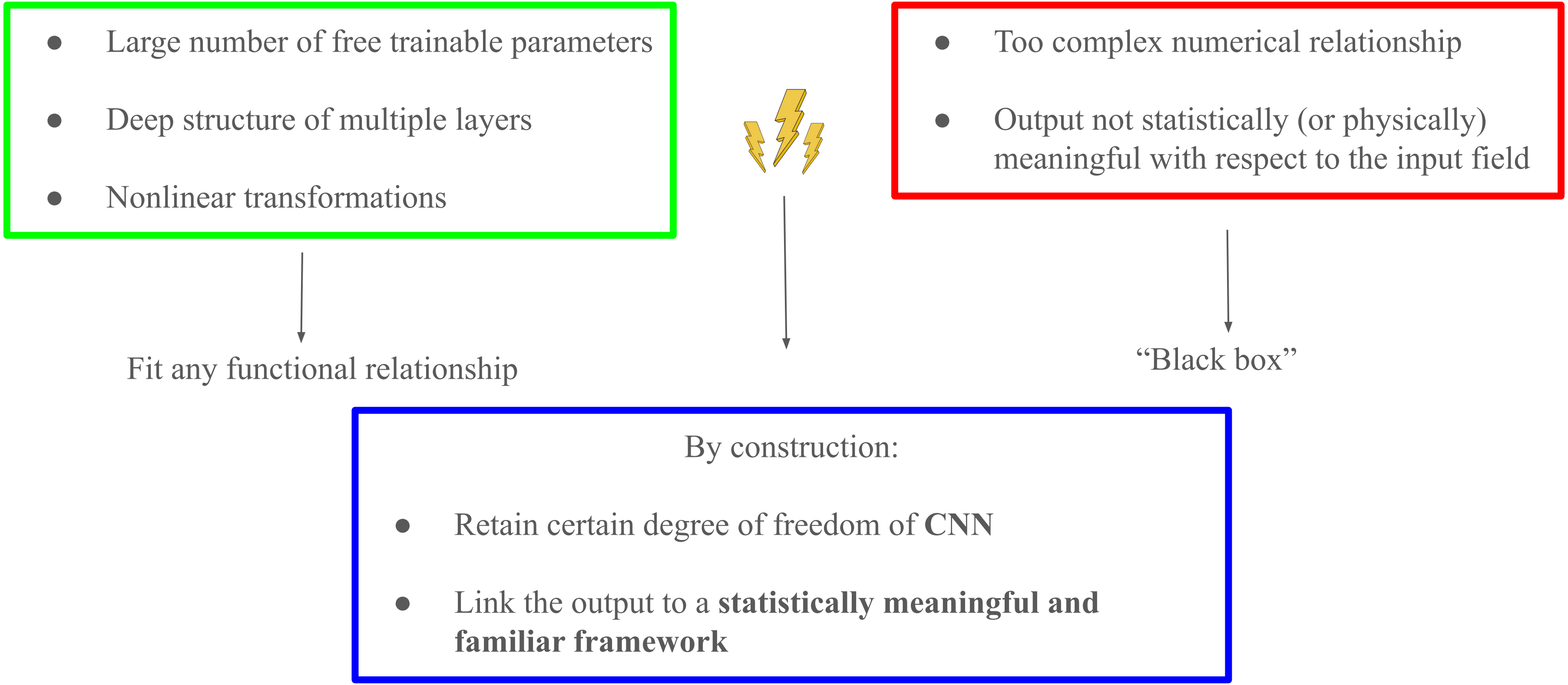
. . .

- 3PCF adds more information to cosmologies compared to 2PCF alone in weak lensing analyses
- What about even higher-orders? How much do they contribute? Can we have any form of quantification of them?
- Lack of both theoretical modelling and efficient estimators



https://arxiv.org/abs/2304.01187

# Motivation: CNN



- Large number of free trainable parameters
- Deep structure of multiple layers
- Nonlinear transformations

Fit any functional relationship

- Too complex numerical relationship
- Output not statistically (or physically) meaningful with respect to the input field

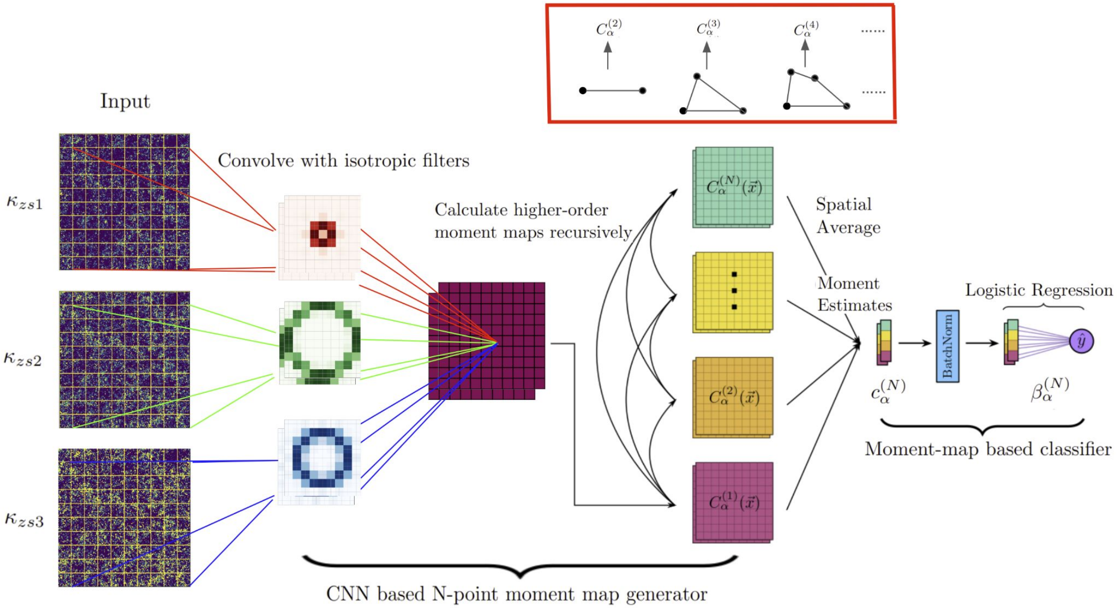
“Black box”

By construction:

- Retain certain degree of freedom of **CNN**
- Link the output to a **statistically meaningful and familiar framework**

# C3NN architecture (https://arxiv.org/abs/2011.03474)

built in pytorch



$C_\alpha^{(2)}$
$C_\alpha^{(3)}$
$C_\alpha^{(4)}$
......
......
Input
$\kappa_{zs1}$
$\kappa_{zs2}$
$\kappa_{zs3}$
Convolve with isotropic filters
Calculate higher-order moment maps recursively
$C_\alpha^{(N)}(\vec{x})$
$C_\alpha^{(2)}(\vec{x})$
$C_\alpha^{(1)}(\vec{x})$
Spatial Average
Moment Estimates
Logistic Regression
BatchNorm
$c_\alpha^{(N)}$
$\beta_\alpha^{(N)}$
$\hat{y}$
Moment-map based classifier
CNN based N-point moment map generator

- Rotationally invariant filter (ESCNN) Cesa et al, 2022 →spatial isotropy
- Single layer of convolution
- Remove nonlinear transformations →an expansion of N-point moment maps

N-point moment maps

$$C_\alpha^{(1)}(\mathbf{x}) = \sum_{\mathbf{a},k} w_{\alpha,k}(\mathbf{a}) S_k(\mathbf{x}+\mathbf{a})$$

$$C_\alpha^{(2)}(\mathbf{x}) = \frac{1}{2!}\left[\sum_{(\mathbf{a},k)\neq(\mathbf{a}_1,k')} w_{\alpha,k}(\mathbf{a}) w_{\alpha,k'}(\mathbf{a}_1) S_k(\mathbf{x}+\mathbf{a}) S_{k'}(\mathbf{x}+\mathbf{a}_1)\right]$$

$$\cdots$$

$$C_\alpha^{(N)}(\mathbf{x}) = \frac{1}{N!}\left[\sum_{(\mathbf{a},k)\neq\ldots\neq(\mathbf{a}_N,k_N)} \prod_{j=1}^{N} w_{\alpha,k_j}(\mathbf{a}_j) S_{k_j}(\mathbf{x}+\mathbf{a}_j)\right]$$

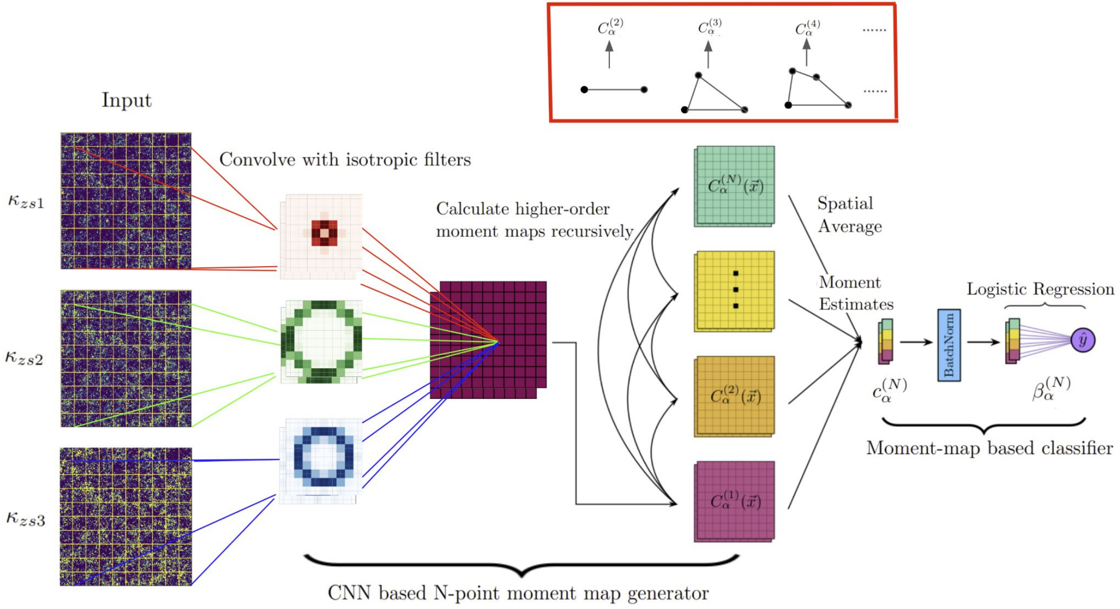
↓

$$C_\alpha^{(N)}(\mathbf{x}) = \frac{1}{N}\sum_{\ell=1}^{N}(-1)^{\ell-1}\left(\sum_{\mathbf{a},k} w_{\alpha,k}^{\ell}(\mathbf{a}) S_k^{\ell}(\mathbf{x}+\mathbf{a})\right) C_\alpha^{(N-\ell)}(\mathbf{x})$$

computational cost up to order N per site:

$\mathcal{O}((KP)^N) \rightarrow \mathcal{O}(N^2 KP)$

# C3NN architecture



Input
$\kappa_{zs1}$
$\kappa_{zs2}$
$\kappa_{zs3}$
Convolve with isotropic filters
Calculate higher-order moment maps recursively
$C_\alpha^{(2)}$
$C_\alpha^{(3)}$
$C_\alpha^{(4)}$
......
......
$C_\alpha^{(N)}(\vec{x})$
$C_\alpha^{(2)}(\vec{x})$
$C_\alpha^{(1)}(\vec{x})$
Spatial Average
Moment Estimates
Logistic Regression
BatchNorm
$c_\alpha^{(N)}$
$\beta_\alpha^{(N)}$
$\hat{y}$
Moment-map based classifier
CNN based N-point moment map generator

N-point moments

$$c_\alpha^{(N)} = \frac{1}{N_{\text{pix}}} \sum_{\mathbf{x}} C_\alpha^{(N)}(\mathbf{x})$$

- Map compression
- Key to the mathematical formulation of correlation functions

- Batch normalization: scale different N-point moments to the same order of magnitude
- Logistic regression (for binary classification):

$$\hat{y} = \frac{1}{1 + \mathrm{e}^{-\boldsymbol{\beta}\cdot\boldsymbol{c}+\epsilon}}$$

# C3NN and correlation functions

## 2-point moments and 2-point correlation function

$$
\begin{aligned}
c_\alpha^{(2)} &= \frac{1}{N_{\rm pix}} \sum_{\mathbf{x}} \frac{1}{2!} \left[ \sum_{(\mathbf{a},k)\neq(\mathbf{a}_1,k')} w_{\alpha,k}(\mathbf{a}) w_{\alpha,k'}(\mathbf{a}_1) S_k(\mathbf{x}+\mathbf{a}) S_{k'}(\mathbf{x}+\mathbf{a}_1) \right] \\
&= \frac{1}{2!} \sum_{(\mathbf{a},k)\neq(\mathbf{a}_1,k')} w_{\alpha,k}(\mathbf{a}) w_{\alpha,k'}(\mathbf{a}_1) \left[ \frac{1}{N_{\rm pix}} \sum_{\mathbf{x}} S_k(\mathbf{x}+\mathbf{a}) S_{k'}(\mathbf{x}+\mathbf{a}_1) \right] \\
&= \frac{1}{2!} \sum_{(\mathbf{a},k)\neq(\mathbf{a}_1,k')} w_{\alpha,k}(\mathbf{a}) w_{\alpha,k'}(\mathbf{a}_1) \hat{\xi}_{kk'}(\mathbf{a}_1-\mathbf{a}) \\
&= \frac{1}{2!} \sum_{(\mathbf{a},k)\neq(\mathbf{a}+\mathbf{r},k')} w_{\alpha,k}(\mathbf{a}) w_{\alpha,k'}(\mathbf{a}+\mathbf{r}) \underbrace{\hat{\xi}_{kk'}(\mathbf{r})}_{\text{2-point correlation function (2PCF) estimator}} \ ,
\end{aligned}
$$

## 3-point moments and 3-point correlation function

$$
c_\alpha^{(3)} = \frac{1}{3!} \left[ \sum_{(\mathbf{a},k)\neq(\mathbf{a}+\mathbf{r},k_1)\neq(\mathbf{a}+\mathbf{r}',k_2)} w_{\alpha,k}(\mathbf{a}) w_{\alpha,k_1}(\mathbf{a}+\mathbf{r}) w_{\alpha,k_2}(\mathbf{a}+\mathbf{r}') \underbrace{\hat{\zeta}_{kk_1k_2}(\mathbf{r},\mathbf{r}',-\mathbf{r}-\mathbf{r}')}_{\text{3-point correlation function (3PCF) estimator}} \right]
$$

# C3NN training procedure (in a binary classification case)

## First round of training:

- trainable parameters: filter weights $w$ (non-negative), logistic regression coefficients $\beta_\alpha^{(N)}$ and bias $\epsilon$
- loss function: $L_{1\mathrm{st}}(y,\hat{y}) = -y\log\hat{y} - (1-y)\log(1-\hat{y}) + \gamma \sum_{\alpha,k,\mathbf{a}} w_{\alpha,k}(\mathbf{a})$
- training product: C3NN model directly usable

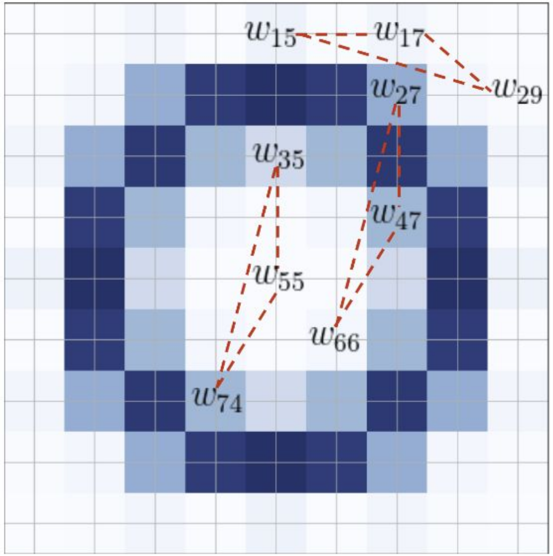
## Second round of training (regularization path analysis):

- trainable parameters: logistic regression coefficients $\beta_\alpha^{(N)}$ and bias $\epsilon$ (filter weights frozen from the first round)
- new loss function: $L_{2\mathrm{nd}}(y,\hat{y}) = -y\log\hat{y} - (1-y)\log(1-\hat{y}) + \lambda \sum_{\alpha,n} |\beta_\alpha^{(n)}|$
- feature selection: the evolution of $\beta_\alpha^{(N)}$ along with changes of regularization strength $\lambda$
- training product: a rank of moments based on their contribution to the classification

# C3NN training procedure (in a binary classification case)

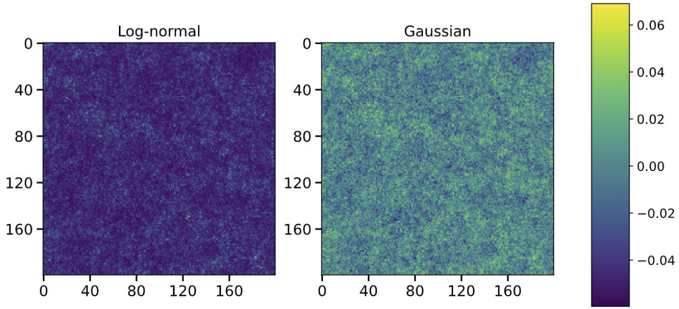
## Filter weights analysis

- At a given order N, rank different configurations of the NPCF based on their total weight $W$ within the filter
- It works in combination with the regularization path analysis
- All NPCF configurations considered here are in real space, limited by the resolution of the input map



$$W = w_{35}w_{55}w_{74} + w_{27}w_{47}w_{66} + w_{15}w_{17}w_{29} + \dots$$

# Proof of concept (Gaussian vs. log-normal random fields)



Input two classes:

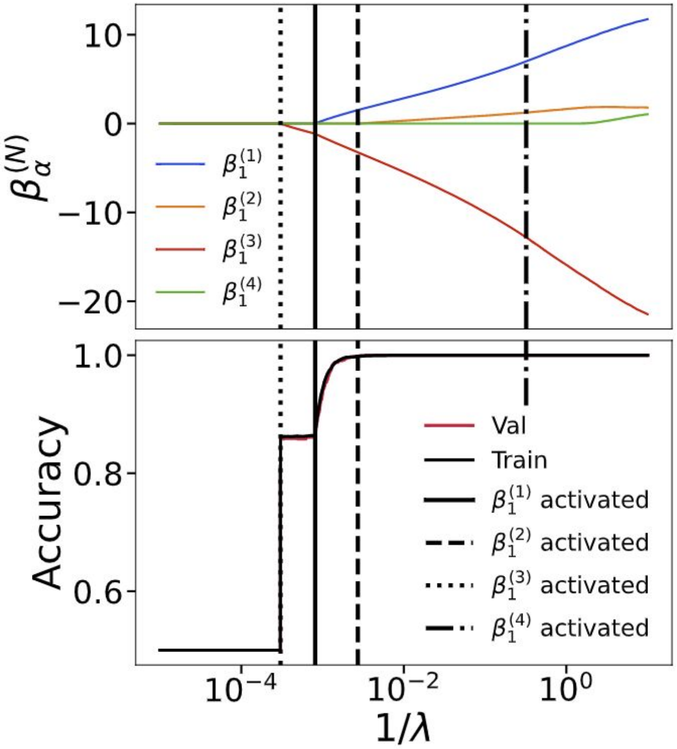
- weak lensing convergence field from FLASK, single channel, pixel resolution 6 arcmin
- one follows a Gaussian distribution, the other a log-normal distribution
- Both have the same underlying cosmological parameters and power spectrum

Model:

- fix model parameters
  (filter number, filter size and highest correlation order) = (1, 31 x 31 pixels, 4)

hyper-parameter tuning

| parameter | $\gamma$ | learning rate (lr) | learning rate decaying ratio ($\phi$) | optimizer |
|---|---|---|---|---|
| value | 0.0026 | 0.15 | 0.66 | "RMSprop" |

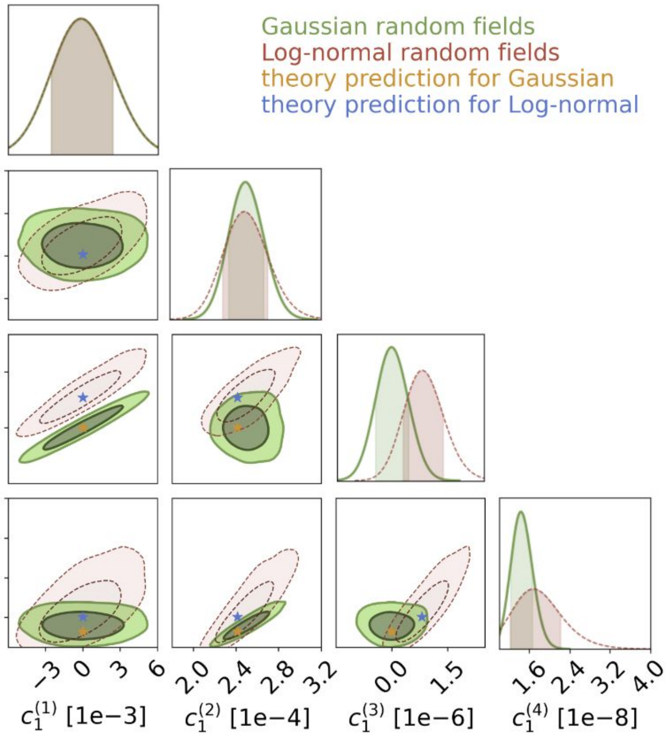


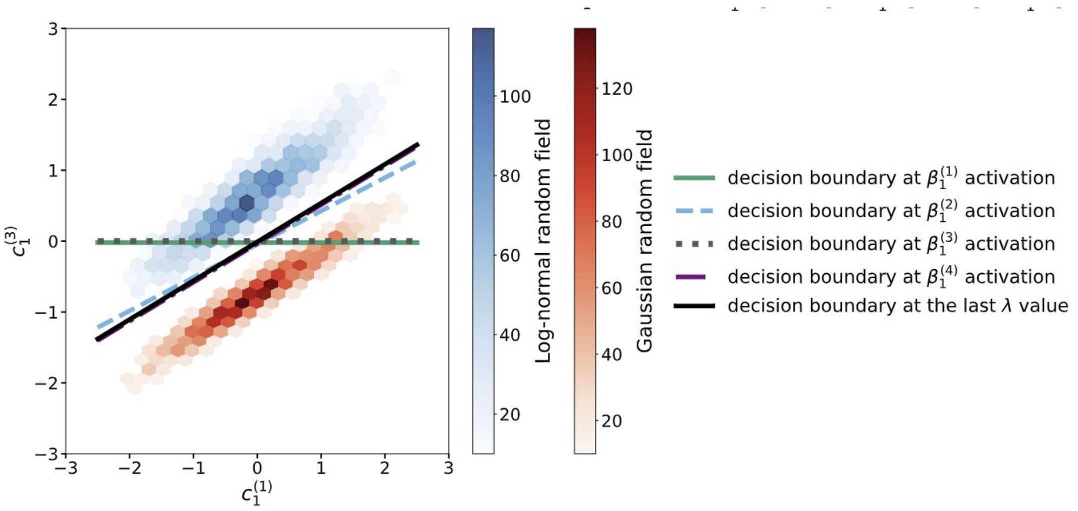
# Regularization path analysis:

- 3rd moment first activated
- 3rd moment combined with average to reach a full classification

$$\hat{y} = \frac{1}{1 + \mathrm{e}^{-\boldsymbol{\beta}\cdot\boldsymbol{c}+\epsilon}}$$

coefficients are important



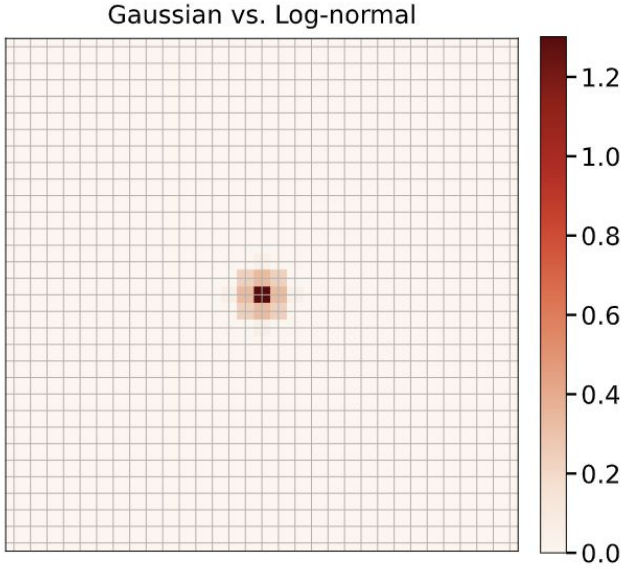


Corner plot:

- no overlapping in the $c_1^{(1)} - c_1^{(3)}$ plane
- agree with the theoretical predictions from analytical log-normal correlation functions (Hilbert et al. 2011) and trained filter weights

Evolution of the decision boundary:

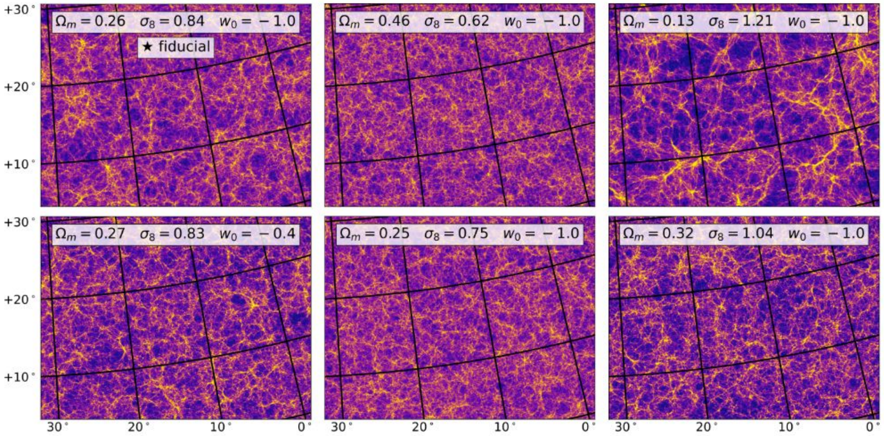
- from 1D to 2D
- linear decision boundary $-\boldsymbol{\beta} \cdot \boldsymbol{c} + \epsilon = 0$
- rotate to the degeneracy direction of distribution $\{c_1^{(1)}, c_1^{(3)}\}$



## Trained filter weights:

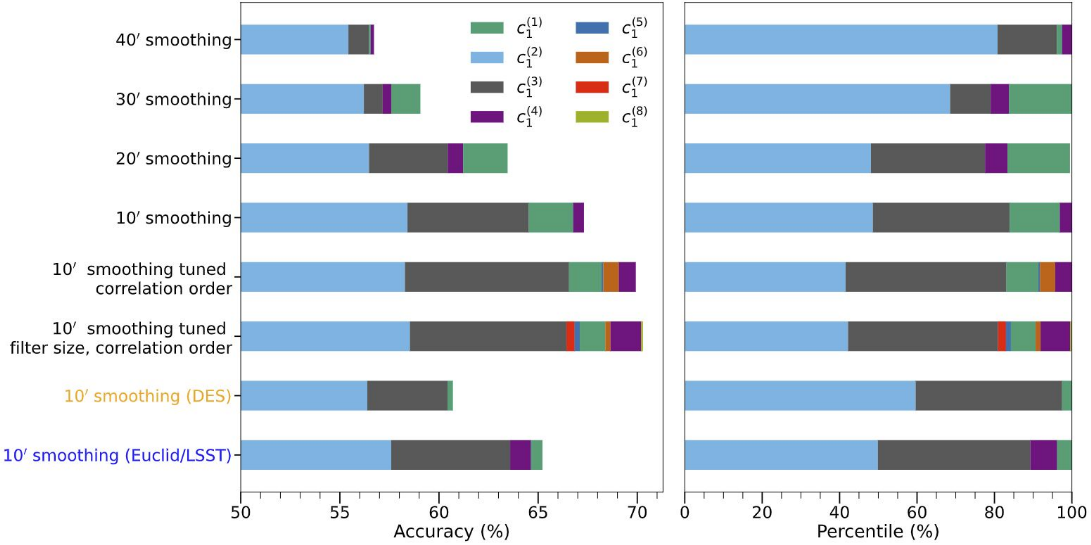
- prominent weights at center of the filter
- most heavily weighted 3-point configuration is the isosceles triangle with side length (6,6, $6\sqrt{2}$ )

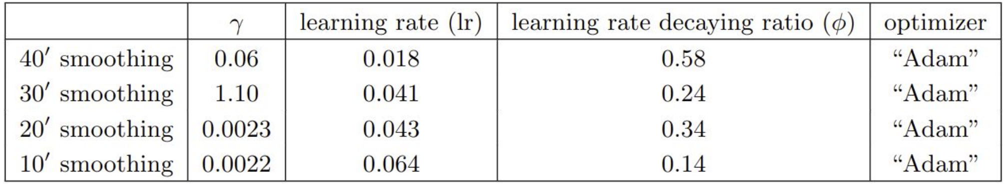
# Test with N-body simulation


$\Omega_m = 0.26$ $\sigma_8 = 0.84$ $w_0 = -1.0$
★ fiducial
$\Omega_m = 0.46$ $\sigma_8 = 0.62$ $w_0 = -1.0$
$\Omega_m = 0.13$ $\sigma_8 = 1.21$ $w_0 = -1.0$
$\Omega_m = 0.27$ $\sigma_8 = 0.83$ $w_0 = -0.4$
$\Omega_m = 0.25$ $\sigma_8 = 0.75$ $w_0 = -1.0$
$\Omega_m = 0.32$ $\sigma_8 = 1.04$ $w_0 = -1.0$
+30°
+20°
+10°
+30°
+20°
+10°
30° 20° 10° 0°
30° 20° 10° 0°
30° 20° 10° 0°

Full-sky weak lensing convergence maps:

- CosmogridV1 simulation suite (http://www.cosmogrid.ai/)
- projected into square maps of 20 by 20 degree with 200 by 200 pixels
- four tomographic redshift bins (Dark Energy Year 3) → tomographic C3NN analysis
- both training classes at fiducial cosmology except the dark energy equation of state $(w_0 = -0.95 \text{ and } -1.05)$
- we implement different smoothing scales and Gaussian shape noise



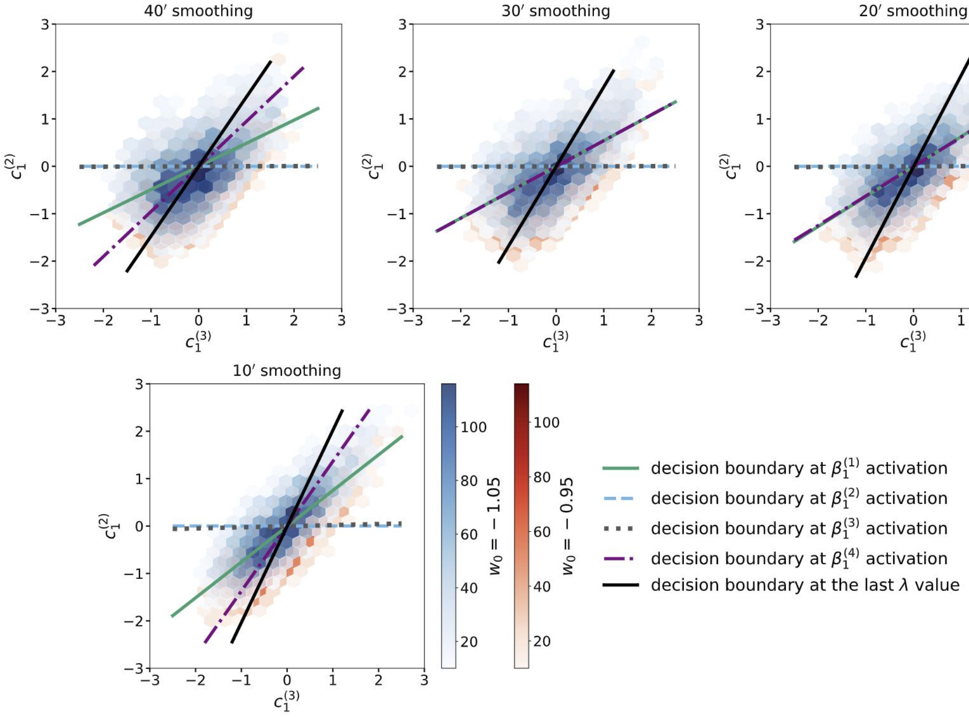


| | $\gamma$ | learning rate (lr) | learning rate decaying ratio ($\phi$) | optimizer |
|---|---|---|---|---|
| 40′ smoothing | 0.06 | 0.018 | 0.58 | “Adam” |
| 30′ smoothing | 1.10 | 0.041 | 0.24 | “Adam” |
| 20′ smoothing | 0.0023 | 0.043 | 0.34 | “Adam” |
| 10′ smoothing | 0.0022 | 0.064 | 0.14 | “Adam” |

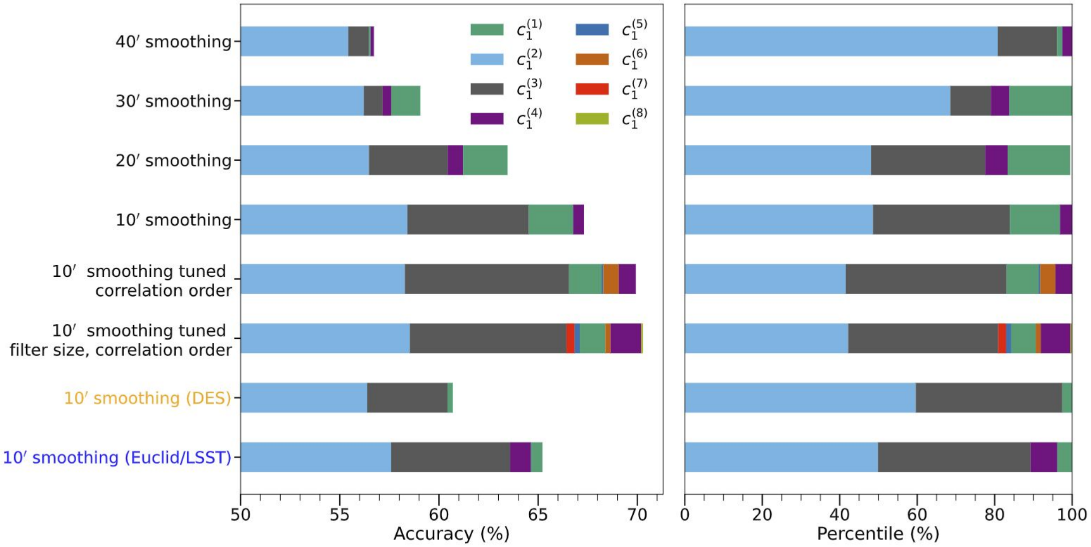
(filter number, filter size and highest correlation order) = (1, 31 x 31 pixels, 4)

**Noiseless with different smoothing scales**:

- 2nd moment activated first, then 3rd moment. The combination of them dominates the classification power
- Increasing smoothing scale reduces the classification accuracy, especially on 3rd order
- Suggesting that moments of convergence beyond 3rd order in total may not contain sufficient information to classify different dark energy equation of state parameters at this desired precision



- Increasing the smoothing scale extends the distribution along the orthogonal direction w.r.t the degeneracy direction of $\{c_1^{(2)}, c_1^{(3)}\}$ joint distribution



| parameter | correlation order | $\gamma$ | learning rate (lr) | learning rate decaying ratio ($\phi$) | optimizer |
|---|---|---|---|---|---|
| value | 6 | 0.0039 | 0.18 | 0.75 | “Adam” |

| parameter | filter size | correlation order | $\gamma$ | learning rate (lr) | learning rate decaying ratio ($\phi$) | optimizer |
|---|---|---|---|---|---|---|
| value | 11 × 11 | 8 | 0.002 | 0.019 | 0.93 | “Adam” |

## Noisy at 10 arcmin smoothing scales with varying C3NN parameters and shape noise:

- Free filter size and correlation order do not significantly help →cross-correlate different observables may help more than increasing the measured correlation order
- Adding shape noise has similar effects on the results as increasing the smoothing scale

# Summary

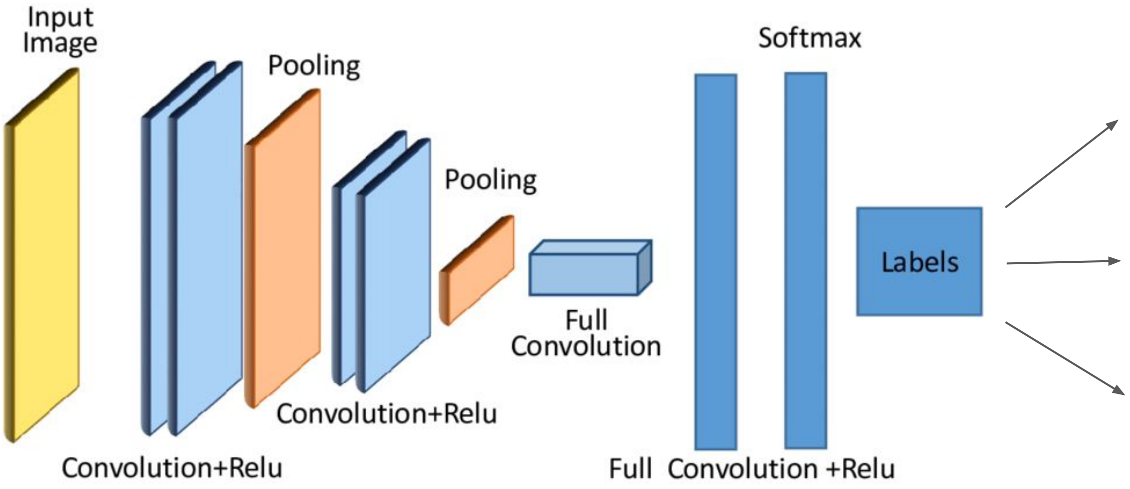
- The architecture of C3NN contains novel features in the application of machine learning to cosmology
    - The output moment at a given order can be mathematically expressed in terms of the correlation function at the same order
    - Through the regularization path analysis, we can have a quantitative understanding of the relative importance of different moments in contributing to the model's predicative power
    - We can investigate the trained filter weights by connecting individual pixels to form the configuration of any given connected NPCF.

- Through multiple tests, we prove its validity and reveal the potential of it to provide us with physical insights

- C3NN can have interesting extensions beyond a classifier

# Additional Slides

# Convolutional Neural Network (CNN)



system classification
e.g. strong gravitational lens detection
Schaefer et al, 2017

parameter inference
e.g. simulation based inference
SIMBIG collaboration, Fluri et al, 2019

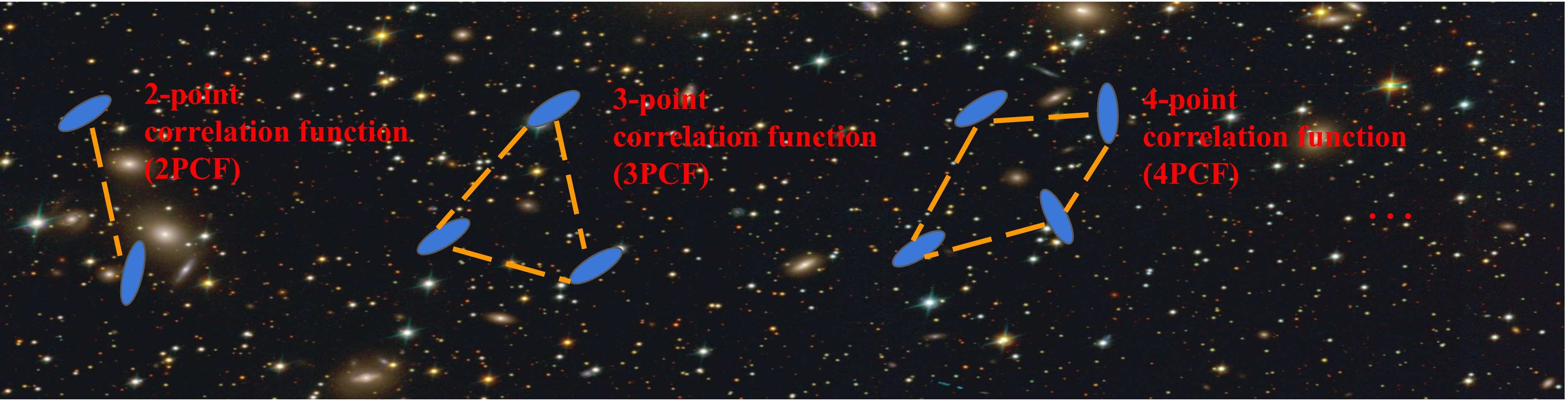
representation learning
e.g. parameter representation of dynamical dark energy
Piras & Lombriser, 2023

…

Linear transformation (convolution): $C_\alpha(\mathbf{x}) = \sum_{\mathbf{a},k} w_{\alpha,k}(\mathbf{a}) S_k(\mathbf{x}+\mathbf{a}) + b_\alpha(\mathbf{x})$

Nonlinear transformation (activation function, e.g. ReLu): $\max\left(0, C_\alpha(\mathbf{x})\right)$

# Correlation function


2-point correlation function (2PCF)
3-point correlation function (3PCF)
4-point correlation function (4PCF)
…

Applications in:

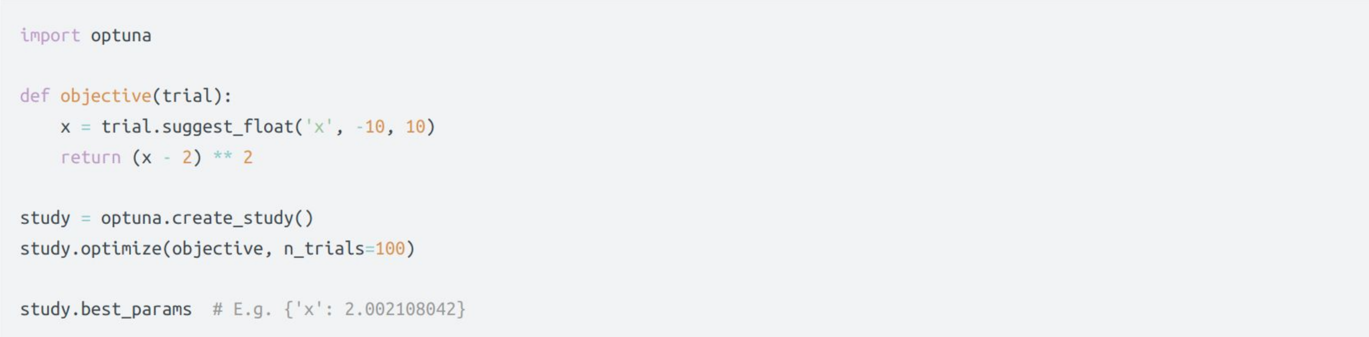
- CMB
- Galaxy clustering
- Weak gravitational lensing

…

# C3NN training procedure (binary classification)

## Hyper-parameter tuning

- hyper-parameters: learning rate, regularization strength…filter number, filter size, highest correlation order
- free to vary, user-defined, can be optimized
- optimization framework: Optuna (https://optuna.org/)
- search algorithm in the hyper-parameter space→maximize or minimize certain metric



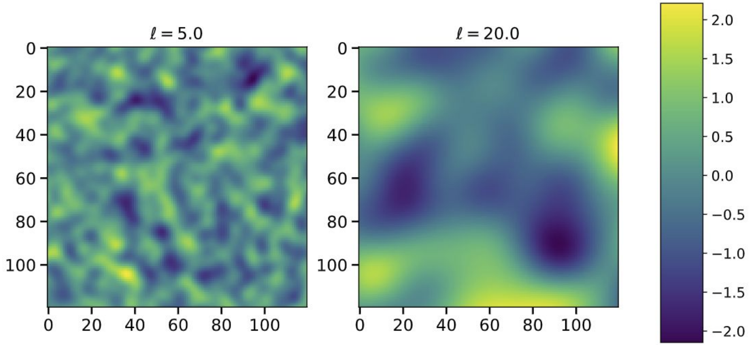
```python
import optuna

def objective(trial):
    x = trial.suggest_float('x', -10, 10)
    return (x - 2) ** 2

study = optuna.create_study()
study.optimize(objective, n_trials=100)

study.best_params  # E.g. {'x': 2.002108042}
```

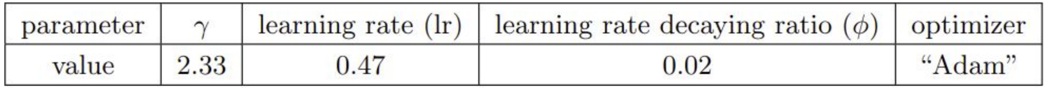
# Test  (Gaussian vs. Gaussian)



Input two classes:

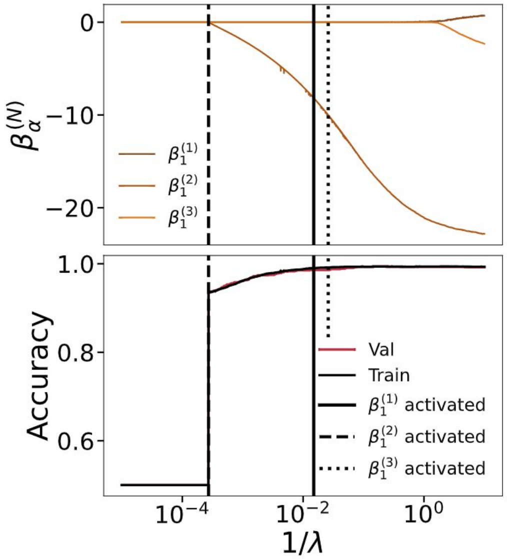
- Both Gaussian random fields, single channel
- Same variance amplitude, but different correlation length

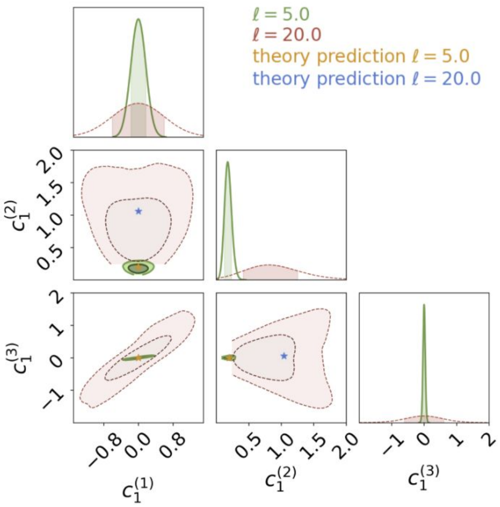
hyper-parameter tuning
(filter number, filter size and highest correlation order) = (1, 31 x 31 pixels, 3)

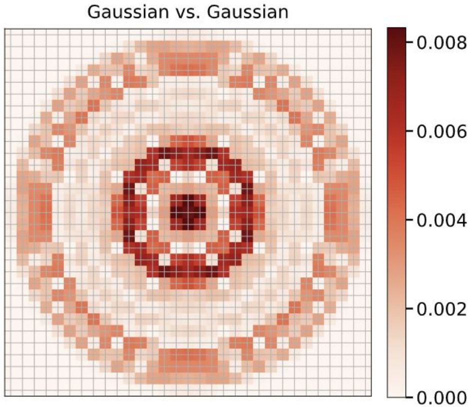


| parameter | $\gamma$ | learning rate (lr) | learning rate decaying ratio ($\phi$) | optimizer |
|---|---|---|---|---|
| value | 2.33 | 0.47 | 0.02 | “Adam” |

the initial learning rate (lr) of each parameter group decays by a factor, the decaying ratio, after every epoch







## Regularization path analysis:

- 2nd moment alone can fully distinguish two classes
- average and 3rd moment are overfitting

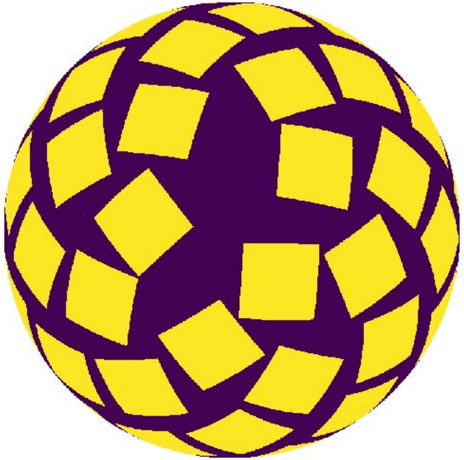
## Corner plot:

- analogy to Bayesian inference corner plot; distributions of moments mapped from the training data
- agree with the theoretical predictions from analytical Gaussian correlation functions and trained filter weights

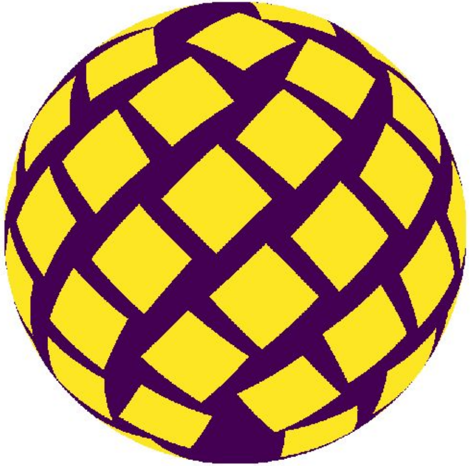
## Trained filter weights:

- two prominent weight annuli
- most heavily weighted 2-point separation is 5 pixels

# Map projection



polar view



side view

- Use the Gnomonic projection from healpy