Eur. Phys. J. C manuscript No.
(will be inserted by the editor)

Estimating Cosmological Parameters and Reconstructing Hubble Constant with Artificial Neural Networks: A Test with covariance matrix and mock H(z)

Jie-feng Chen[1,2,3], **Tong-Jie Zhang**[⋆,a,2,3], **Peng He**[4], **Tingting Zhang**[5] **and Jie Zhang**[b,1]

[1]School of arts and sciences, Shanghai Dianji University, Shanghai, 200240, China
[2]Institute for Frontiers in Astronomy and Astrophysics, Beijing Normal University, Beijing 102206, China
[3]School of Physics and Astronomy, Beijing Normal University, Beijing 100875, China
[4]Bureau of Frontier Sciences and Basic Research, Chinese Academy of Sciences, Beijing 100190, China
[5]College of Command and Control Engineering, PLA Army Engineering University, Nanjing 210000, China
Received: date / Accepted: date

Abstract In this work, we reconstruct the $H(z)$ based on observational Hubble data with Artificial Neural Network, then estimate the cosmological parameters and the Hubble constant. The training data we used are covariance matrix and mock $H(z)$, which are generated based on the real OHD data and Gaussian Process(GP). The use of the covariance matrix propagates the correlated uncertainties and improves training efficiency. Using the reconstructed H(z) data, we first determine the Hubble constant and compare it with CMB-based measurements. To constrain cosmological parameters, we sample on the reconstructed data and calculate the corresponding posterior distributions with Markov Chain Monte Carlo (MCMC). Through comprehensive statistical comparisons, we demonstrate that the parameter estimation using reconstructed samples achieves comparable statistical accuracy to the result derived from real OHD data.

1 Introduction

The accelerating expansion of the universe is one of the most important discoveries in modern cosmology research. In order to explain this phenomenon, constraining cosmological parameters and estimating the Hubble constant have long been fundamental tasks in cosmology.

To achieve these goals, researchers rely on a variety of observational datasets that probe different aspects of the universe. For instance, current datasets include Damped Lyman - α Absorber (DLA) of HI 21 cm system [34, 35], observational Hubble parameter data (OHD, Jesus *et al.* [27]), type Ia supernovae (SNe Ia, [56]), cosmic microwave background ([56]), and large-scale structures [50]. Bseides constraining the cosmological parameters [47, 72], with the observational data, we can understand the neutrino condensation [70], searching for extraterrestrial intelligence [36, 60] and so forth. However, in some occasions, because of the

⋆Corresponding Author
[a]e-mail: tjzhang@bnu.edu.cn
[b]e-mail: zhangjie@qlnu.edu.cn

arXiv:2410.08369v4 [astro-ph.CO] 3 Sep 2025

insufficiency of the existing data, we need to reconstruct or interpolate the observational data, according to the real observational datas and hypotheses.

At the same time, in the past few decades, the artificial neural networks (ANN) developed rapidly [29]. The introduction of deep learning architectures enabled training of very deep neural networks with numerous layers. Deep learning models like Transformer networks, GANs (Generative Adversarial Networks, [20]), and BERT (Bidirectional Encoder Representations from Transformers [31]) have demonstrated remarkable performance in different applications. Therefore, more and more astrophysics tasks started to apply ANNs, such as processing large scale astronomical data [7], constraining the cosmological parameters [9, 65] and data reconstruction [73].

Reconstruction methods are essential in cosmology, where observational data are often sparse and unevenly distributed. By reconstructing quantities such as the Hubble parameter, we are able to obtain a continuous representation of the Universe's evolution, enabling more reliable inference of cosmological parameters. Neural networks have recently been explored as powerful non-parametric tools for cosmological reconstructions, particularly in handling sparse or noisy data. As highlighted in the Cosmoverse White Paper[15], they offer a flexible alternative to traditional methods, and have been applied to reconstruct cosmological functions. In this work, we attempt to reconstruct the Hubble parameter in the redshift range of $z \in [0, 2]$, and then estimate the cosmological parameters and reconstruct the Hubble constant with the reconstructed data. We proposed a new method to complete reconstruction in our work, combining the mock data and the covariance matrix. In particular, [74], [63], and [22] have previously reconstructed the same H(z) data. However, covariance matrix was not considered in the reconstruction process in [63]. Though [22] and [74] took the covariance matrix into account, they just generated covariance matrices based on the existing OHD data points. In our work, the covariance matrix we constructed yields non-zero entries in regions lacking observational data. One of the advantage of our method is the use of the covariance matrix ensures that correlated uncertainties are properly propagated, which could improve the robustness of parameter estimation and reduce training time. Besides, the framework (such as the generation of the covariance matrix) is readily extendable and expected to become more powerful when larger and more precise OHD data points (e.g., CSST, Euclid) become available.

This paper is organized as follows: In section 2, firstly we briefly introduce the basic knowledge of the artificial neural network, and how we realize the reconstruction with ANN. In section 3, we introduce the OHD data we use in this project, and how we generate the mock OHD data. Besides, we also show the reason why we added an additional covariance matrix to the neural network, and how we built it. In section 4, we show the reconstructed Hubble data generated by the neural network, and we calculate the Hubble constant [18, 26] with the reconstructed data. Meanwhile, we constrain the cosmological parameters and do quantitative analysis (Kullback-Leibler divergence and Figure of Merit). Finally in Section 5, we discuss and conclude.

2 Methodology

We first review basic principles of ANNs and CNNs in this section, then introduce the specific neural network architecture we employed in our work.

2.1 Neural Network

Artificial Neural Networks (ANNs) [68] are computational models inspired by biological neural systems. They are typically used to approximate complex functions (mappings) between inputs and outputs through a set of interconnected layers.

2.1.1 Network Structure and Activation Functions

A basic ANN typically consists of an input layer, one or more hidden layers, and an output layer. The input layer receives input data, which is then transformed through the hidden layers using weighted connections and nonlinear activation functions, ultimately the output layer produces the prediction (output).

Activation functions [53] introduce nonlinearity into the network, enabling it to learn complex patterns. Common activation functions include the Sigmoid [32], ReLU [10], and Tanh [16]. For example, given an input x, a weight ω_1, and bias b, the activation output is:

$$h = f(\omega_1 x + b), \tag{1}$$

where f is the activation function. A basic structure of ANN showed as 1a, and the activation function shown as fig. 1b.

2.1.2 Backpropagation and Optimization

Neural networks are trained using the backpropagation algorithm [25], which adjusts weights by minimizing a loss function via gradient descent. A common loss function is the Mean Squared Error:

$$L = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2. \tag{2}$$

The weights are updated according to:

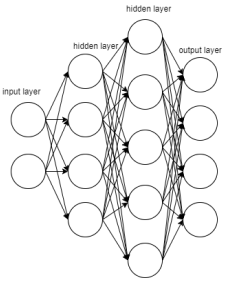
$$x = x - \eta \frac{\partial L}{\partial x}, \tag{3}$$

where η is the learning rate. Selecting an appropriate learning rate is crucial for efficient convergence [24]. Fig. 1c shows how back propagation works.

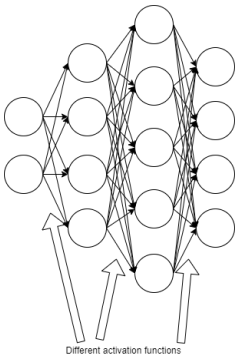
2.1.3 Convolutional Neural Networks (CNNs)

In this work, we employ Convolutional Neural Networks (CNNs) [67], which are particularly well-suited for image processing tasks due to their ability to exploit spatial hierarchies in data. A basic structure of the CNN shown in Fig. 2.

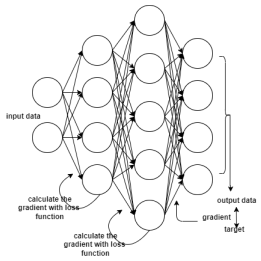
CNNs typically consist of convolutional layers, pooling layers, and fully connected layers. The convolutional layer applies learnable filters (kernels) that slide over the input to extract local features [13]. Pooling layers (e.g., max or average pooling) downsample the feature maps to reduce dimensionality while preserving key features [57]. Finally, fully connected layers convert the processed feature maps into a flat vector for final prediction.



(a) Basic structure of an artificial neural network (ANN), including input layer, hidden layer(s), and output layer. Practical ANNs typically have more complex architectures.



(b) As shown above, activation functions are applied to the connections between units in adjacent layers.



(c) During backpropagation, the algorithm first computes the loss between the network's output and the target values, then calculates the gradient of the output layer. This process is repeated recursively to compute the loss and gradients for each preceding layer in the network.

Fig. 1: The structure of the ANN.

2.1.4 Batch Size and Training

We use large batch sizes in training process to facilitate computation of the output covariance matrix. Batch training allows for efficient parallel processing, especially on GPUs [69]. In each batch, a forward pass computes predictions, followed by a loss calculation and a backward pass to update weights. One training epoch completes after processing all batches.

2.2 The Artificial Neural Networks in This Work

In our work, we tried to reconstruct the $H(z)$ in the range of $z \in [0,2]$. As shown in the Fig. 3, in the begining of our neural network, we built a CNN to process the two-dimensional covariance matrix. Then we added the z (redshift) to the network with the compressed covariance matrix to a full connected network. Flinally, the network outputs the reconstructed H(z) in redshift range $[0,2]$ as a 200-element array. In our work, to compute the covariance loss efficiently, we let the network directly output reconstructed data with a shape of $(1000,200)$ (An example shown in Fig. 8a). Here, the first dimension (1000) corresponds to 1000 distinct reconstruction of $H(z)$ in the redshift range of $z \in [0,2]$ (note: this sample size can be increased as needed, e.g., to 2000 or much larger). Therefore the network would be simultaneous processing of 1000 covariance matrices, each with dimensions $(200,200)$, resulting in a 3D tensor structure $(1000,200,200)$. Through convolutional layers, these are compressed to an output shape of $[1000, D_{\text{cov}}]$, where D_{cov} denotes the reduced dimensionality of the compressed covariance representation. Consistent with this architecture, the redshift inputs are structured as a $(1000,32)$ matrix, representing 1000 sets of redshift values paired with their corresponding $H(z)$ realizations.

In our neural network, the loss consists of two parts: the loss of the reconstruction and the loss of the covariance matrix. Firstly, we calculated the reconstruction loss between the

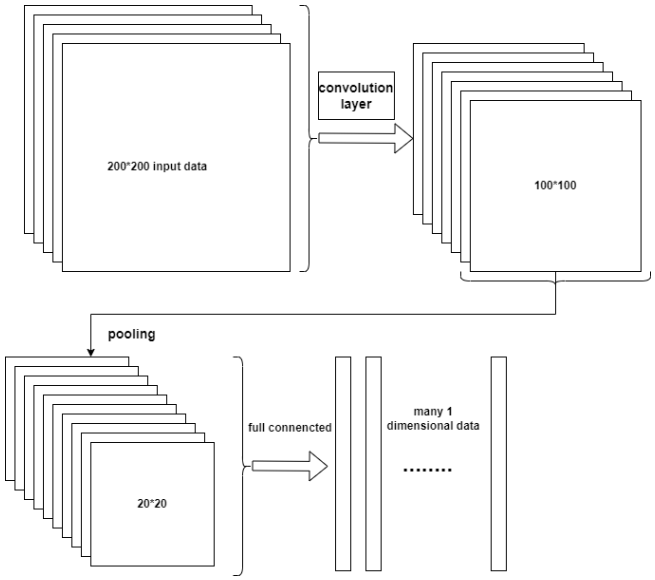


Fig. 2: The basic architecture of CNN. Through convolution and pooling layers, the two-dimensional input data is progressively reduced in size using different operations. Finally, a fully-connected layer transforms the two-dimensional features into one-dimensional output.

output and the training data with MSE (mean squared error) loss:

$$Loss_{reconstruction} = \frac{1}{n}\sum_{i=1}^{n}(H(z)_i - \hat{H}(z)_i). \tag{4}$$

Then we further calculated the covariance matrix of the output data, and thus computed the MSE loss between the training covariance matrix and the covariance matrix of the output data, in our work we define it as covariance loss:

$$Loss_{covariance} = \frac{1}{n}\sum_{i=1}^{n}(cov_i - \hat{cov}_i). \tag{5}$$

3 Data

3.1 The OHD data We Used in This Work

The real OHD is composed of $z_i, H(z_i)$ and σ_i, where z_i is the redshift, and $H(z_i)$ is the corresponding Hubble parameter and σ_i is the corresponding uncertainty. The 32 OHD data points we used in this work are evaluated with the cosmic chronometer method, which are given in [30], [58], [59], [45], [54], [14], [44], [28] and [39], and are shown in Fig. 4.

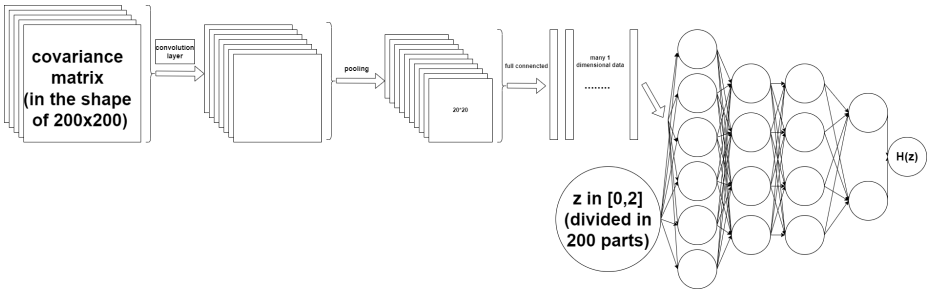


Fig. 3: The network we used in our work. The network architecture integrates both convolutional and fully-connected components. Since the input covariance matrix is two-dimensional, we first process it through a CNN. The CNN part transforms this matrix into a one-dimensional feature vector, enabling concatenation with other one-dimensional input data for subsequent processing in the fully-connected network. In the right part, we built a normal full connected network. Besides the compressed covariance matrix, we also input the redshift in the range $[0,2]$ to the fully-connected network. Finally, the shape of the output is $(200,)$, representing the reconstructed $H(z)$ in the redshift range of $[0,2]$.

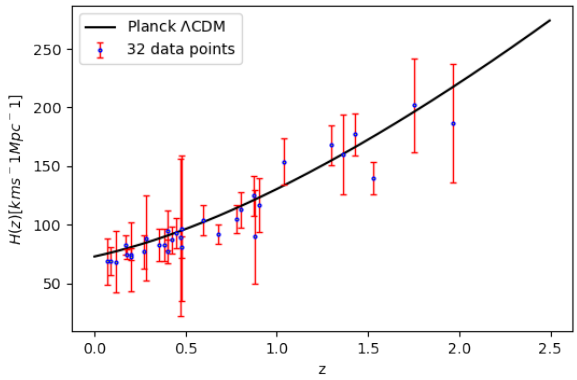


Fig. 4: In this work, we employ 32 observational data points obtained through model-independent methods to avoid potential biases from theoretical assumptions. For comparison, we also include the $H(z)$ prediction from Planck Collaboration [1] with cosmological parameters $\Omega_\Lambda = 0.686$, $\Omega_m = 0.314$, and $H_0 = 67.4$ km s^{-1} Mpc^{-1}. The Planck curve shows general agreement with the distribution of our data points.

3.2 The Mock OHD

According to the flat ΛCDM model, the Hubble parameter is expressed by redshift z with the simple formula:

$$H(z) = H_0\sqrt{\Omega_m(1+z)^3+\Omega_\Lambda}, \tag{6}$$

where the H_0 is the Hubble constant. At the same time, the Hubble parameter can also be given by the non-flat ΛCDM model:

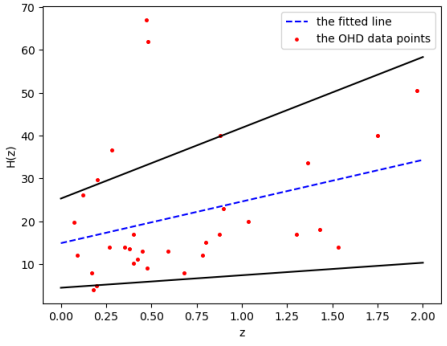
$$H(z) = H_0\sqrt{\Omega_m(1+z)^3 + \Omega_\Lambda + \Omega_k(1+z)^2}. \tag{7}$$

In our work, we generated the mock $H(z)$ with the flat ΛCDM model (Eq. 6), with the fiducial $H_0 = 67.4 kms^{-1}\mathrm{Mpc}^{-1}$ and $\Omega_m = 0.314$ [1]. In the first place, we generate the error of the mock data. With the same error method used by [37], we assumed that the error of $H(z)$ increases linearly with the redshift. We firstly fitted $\sigma_{H(z)}$ with first degree polynomials and obtain $\sigma_0 = 9.72z + 14.87$ (the blue dashed line). Here we assumed that σ_0 is the mean value of $\sigma_{H(z)}$ at a specific redshift. And then, two lines (the black solid lines) are selected symmetrically around the mean value line to ensure that most data points are in the area between them, and these two lines have the functions of $\sigma_- = 2.92z + 4.46$ and $\sigma_+ = 16.52z + 25.28$, which are shown in the Fig. 5a. Finally, the error $\sigma(z)$ was generated (or sampled) randomly according to the Gaussian distribution $\mathcal{N}(\sigma_0(z), \varepsilon(z))$, where $\varepsilon(z) = (\sigma_+ - \sigma_-)/4$ was set to ensure that $\sigma(z)$ falls in the area with a 95% probability.

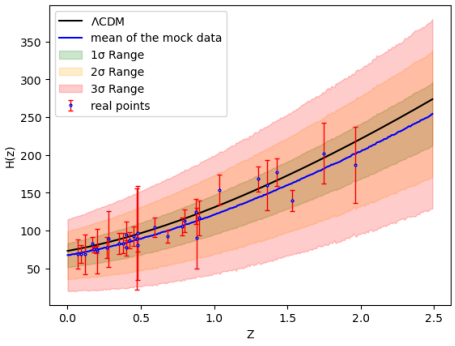
The fiducial values of the Hubble parameter $H_{\mathrm{fid}}(z)$ generated using Eq. 6 are simulated randomly by adding ΔH subject to $\mathcal{N}(0, \varepsilon(z))$. Thus, we can obtain the H_{moc} with the formula:

$$H_{\mathrm{moc},i} = H_{\mathrm{fid}}(z_i) + \Delta H_i, \tag{8}$$

where z_i is in the redshift range $[0,2]$. We show the mock $H(z)$ in Fig. 5b, we generated 10000 data points and plotted the 1σ, 2σ and 3σ of the distribution. In our work, the mock data serves two purposes. First, it is used to train the neural network. Second, we perform Bayesian inference with it, and the result will be a comparison to the result we got with the reconstruction data.



(a) The error model of the mock $H(z)$ [37]. We sample the error (ΔH_i) of the mock $H(z)$ in the region between two blue lines, the redshift range of $[0,2]$. The blue line means the fitted equation of the distribution of the error of the observational data. In this model, we gave up the two above points, for the reason that they are too away from the other data points.



(b) The 1σ, 2σ and 3σ of the mock $H(z)$. We had generated 10000 mock $H(z)$ points. Here we show the redshift in the range of [0,2.5], but the biggest z of the OHD data points is just around 2, so in the work we just tried to reconstruct the $H(z)$ in the redshift range of $[0,2]$.

Fig. 5: The construction of the mock data.

3.3 The Training Covariance Matrix in This Work

Firstly, we will briefly introduce the Gaussian process [38], then we explain how we generate the covariance matrix with the Gaussian process regression [66].

3.3.1 The Gaussian Process Regression

Gaussian Process Regression (GPR) is a non-parametric, probabilistic model that assumes the data is generated from a stochastic process. A Gaussian process (GP) is defined as a collection of random variables, any finite subset of which follows a multivariate normal distribution. In essence, GPR performs regression by modeling data as samples drawn from a GP, where the distribution over functions is specified by a mean function and a covariance (kernel) function.

A one-dimensional Gaussian distribution is characterized by its mean μ and variance σ^2:

$$x \sim \mathcal{N}(\mu, \sigma^2), \tag{9}$$

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right). \tag{10}$$

For a multivariate Gaussian distribution, the distribution is defined by a mean vector and a covariance matrix Σ:

$$X \sim \mathcal{N}(\mu, \Sigma). \tag{11}$$

A Gaussian process generalizes this concept to an infinite-dimensional case, defined over a continuous input space:

$$f(x) \sim \mathcal{GP}(\mu(x), k(x,x')), \tag{12}$$

where $\mu(x)$ is the mean function, often assumed to be zero, and $k(x,x')$ is the covariance function (kernel) that encodes similarity between inputs.

In a regression setting, we model observations $y^{(i)}$ as:

$$y^{(i)} = f(x^{(i)}) + \varepsilon^{(i)}, \quad \varepsilon^{(i)} \sim \mathcal{N}(0, \sigma^2), \tag{13}$$

where $f(x)$ is drawn from a GP.

Given training data (X,Y) and test inputs $X\_$, the joint distribution of training and test outputs is:

$$\left[f(X)\ f(X\_)\right] \sim \mathcal{N}\left(0, \left[K(X,X)\ K(X,X\_)\ K(X\_,X)\ K(X\_,X\_)\right]\right). \tag{14}$$

Applying Bayes theorem yields the posterior predictive distribution:

$$\mu_* = K_*(K + \sigma^2 I)^{-1} Y, \tag{15}$$

$$\Sigma_* = K\_ - K_*(K + \sigma^2 I)^{-1} K\_^{\top}, \tag{16}$$

where $K = K(X,X)$, $K\_ = K(X\_*,X)$, and $K\_ = K(X\_,X\_)$.

The marginal likelihood of the observations is:

$$p(y|x) = \frac{1}{\sqrt{(2\pi)^n |K + \sigma^2 I|}} \exp\left(-\frac{1}{2} y^{\top} (K + \sigma^2 I)^{-1} y\right), \tag{17}$$

and the corresponding log-marginal likelihood is:

$$\log p(y|x,\theta) = -\frac{1}{2}y^{\top}(K+\sigma^2 I)^{-1}y - \frac{1}{2}\log|K+\sigma^2 I| - \frac{n}{2}\log 2\pi, \tag{18}$$

which is typically maximized with respect to the kernel hyperparameters θ.

A commonly used kernel is the Radial Basis Function (RBF) kernel, also known as the Gaussian kernel [6]:

$$k(x_i,x_j) = \sigma_f^2 \exp\left(-\frac{|x_i - x_j|^2}{2l^2}\right), \tag{19}$$

where l is the length-scale and σ_f^2 is the signal variance. The choice of kernel is crucial, as it controls the properties of the GP and the function space being modeled [55].

3.3.2 GPR with The Python Package Scikit-learn

In our work, we used the Python package scikit-learn [51] to perform Gaussian Process Regression. Scikit-learn is a widely-used open-source machine learning library built on top of NumPy [49], SciPy [62], and matplotlib [61].

We employed the RBF kernel [6], which is the default and most commonly used kernel for GPR. It is well-suited for modeling smooth, continuous functions and provides a good balance between model flexibility and generalization performance.

3.3.3 The Generation of The Covariance Matrix

In Gaussian process regression, the covariance matrix encodes the correlations between both observed and reconstructed points. Even when the number of reconstruction points exceeds the number of observations, the predictive distribution is fully determined by the kernel-induced covariance structure, which governs both the mean prediction and the associated uncertainties. Therefore, we generated the covariance matrix with the GP reconstruction. There are some specifically key steps of the process of calculating the covariance matrix:

(1). Sampling 1000 sets of $H(z)$ in every specific redshift from the Gaussian process reconstruction $\mathcal{N}(H(z),\sigma_{H(z)})$. In other word, firstly we will generate an array in shape $(200,1000)$ (Because we have 200 redshift in the range $z\in[0,2]$.).

(2). For two Hubble parameters at the redshift z_1 and z_2, the covariance between them can be calculated by comparing the 1000 $H(z_1)$ and 1000 $H(z_2)$ values using the equation:

$$Cov(H(z_i),H(z_j)) = \frac{1}{N}\sum_{k=1}^{N}[(H(z_i)_k - \bar{H}(z_i))(H(z_j)_k - \bar{H}(z_j))], \tag{20}$$

where $N = 1000$, which means the number of data points at the $H(z_1)$ and $H(z_2)$, and $\bar{H}(z)$ means the average value of the 1000 $H(z)$ data points.

(3). Using the method of **(2)**, we can calculate the covariance between any two Hubble parameters with different redshifts. Therefore we can calculate the whole covariance matrix.

We draw the figure of the covariance matrix in Fig. 6 and randomly chose 15 data points to draw the heatmap in Fig. 7b. Here we quoted the covariance matrix calculated by [42] (we show it in the Fig. 7a), which represents the covariance of the $H(z)$ in the redshift region of $[0,2]$. The covariance matrix calculated by [42] and the covariance matrix calculated with the GPR share similar characteristic: almost only having values on the diagonal line. We can clearly see that the covariance matrix of the mock $H(z)$ is consistent with the result

from [42], almost only having values on the diagonal line as well. In [23], the covariance matrix they generated also only has value on the diagonal line. The purpose of our work is to reconstruct OHD data in the redshift range of $[0,2]$ based on the real OHD data, and we hope the reconstructed data could be used as real data, so the reconstructed covariance matrices should be consistent with the real OHD data. It might be better to avoid adding too many additional components.

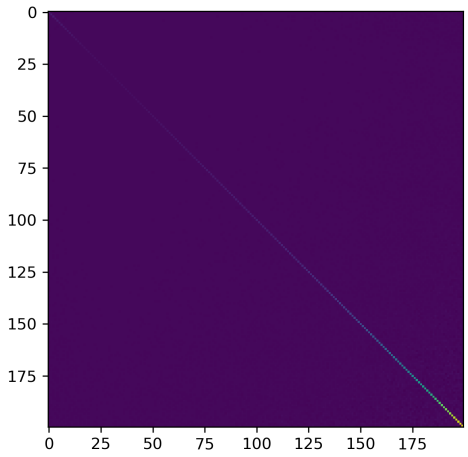
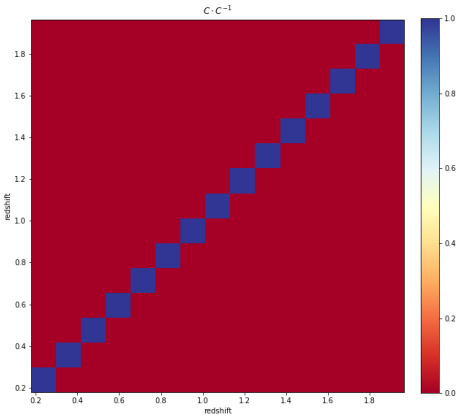
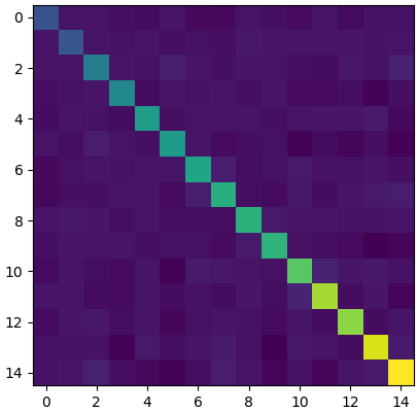


Fig. 6: The full covariance matrix that generate with the $H(z)$ generated with the Gaussian process regression.



(a) The covariance matrix that generate by [42]. There are only values on the diagonal line.



(b) We randomly took 15 data points from the generated covariance matrix.

Fig. 7: The comparison of the covariance matrix.

4 Reconstruction and Parameters Estimation

4.1 MCMC Method

Firstly, in this subsection, we briefly introduce the MCMC method (Markov chain Monte Carlo, [12, 33]), which is the method we used to calculate the posterior distribution of the cosmological parameters.

MCMC method is widely used in various fields, including Bayesian inference [71], physics, and machine learning [2]. The process of MCMC involves simulating a Markov chain [48] in the parameter space, sampling randomly and then satisfying the samples to the target probability distribution function (PDF). It proceeds as follows:

(1). Initialization: Start from a random arbitrary initial state x_0.

(2). Proposal: Propose a new state x' based on a proposal distribution $Q(x'|x_t)$, which defines the probability of transitioning from state x_t to x'.

(3). Acceptance: Calculate the acceptance probability (α) to decide whether to accept or reject the proposed state.

$$\alpha(x_t, x') = min\left(1, \frac{P(x') * Q(x_t|x')}{P(x_t) * Q(x'|x_t)}\right) \tag{21}$$

If $\alpha > 1$ accept x' as the next state.

If $\alpha < 1$ accept x' with probability α and stay at x_t otherwise.

(4). Repeat: Generate a series of states by iterating steps 2 and 3.

This process generates a sequence of states that eventually converge to samples from the target distribution $P(x)$. The samples can then be used to estimate expectations or calculate integrals over the target distribution.

MCMC method has many different kinds of algorithms, such as the Metropolis-Hastings [11], which provides a way to explore complex probability distributions, allowing researchers to draw samples from distributions that are otherwise challenging to sample directly.

The MCMC method we used in our work is Emcee [17]. Emcee is a Python package for performing Markov Chain Monte Carlo (MCMC) simulations, based on the affine-invariant ensemble sampler proposed by Goodman and Weare [21], rather than the traditional Metropolis-Hastings algorithm. It is particularly well-suited for Bayesian inference in high-dimensional and highly correlated parameter spaces. By using an ensemble of walkers that collectively explore the posterior distribution, emcee provides an efficient and user-friendly framework widely adopted in fields such as astrophysics, statistics, and machine learning [5]. Like other MCMC methods, one advantage Emcee has is that it generates 10000 (or we can define the number by ourselves) data points with Markov chain in the end to represent the posterior distribution, at the same time it also provides the uncertainties.

4.2 Reconstruction of $H(z)$

In the Fig. 8a, we show the reconstruction of $H(z)$ generated with our neural network. Besides, we show other 2 different reconstructed datasets. Fig. 8b was generated when the reconstructed dominates, Though it has a normal range, the edge is rough.

We also test the situation that reconstructing the $H(z)$ without covariance loss, showing the result in the Fig. 8c. Without the covariance loss, the edge is much rougher, and it took a longer time to train the network. We drew the training loss curve in the Figs. 9. Fig. 9a shows the reconstruction loss, the covaricance loss, and the rencostruction loss when we didn't use

covariance, all of them decrease rapidly. However, if we expand the epoch and only see the reconstrcution loss (shown in Fig. 9b), we find that it takes more epochs to reach minimum value. Even in the Fig. 9b, the loss was still decreasing and didn't reach the minimum value. We use different methods to see the trend of the losses in Fig. 9c and Fig. 9d, both of them show that the reconstruction loss decreases slowly without covariance loss. Therefore, the covariance loss is necessary.

4.3 Reconstruction of The Hubble Constant

We also calculate the Hubble constant from the reconstructed $H(z)$. The Hubble constant is an important parameter in cosmology. According to the Friedman equations [19]:

$$\dot{a}^2 - \frac{1}{3}(8\pi G\rho + \Lambda)a^2 = -kc^2, \tag{22}$$

and

$$\frac{\ddot{a}}{a} = -\frac{4}{3}\pi G(\rho + 3\frac{p}{c^2}) + \frac{1}{3}\Lambda. \tag{23}$$

Here $a = a(t)$ is the scale factor of the Universe. At any given time, we can define a Hubble parameter:

$$H(t) = \frac{\dot{a}}{a}. \tag{24}$$

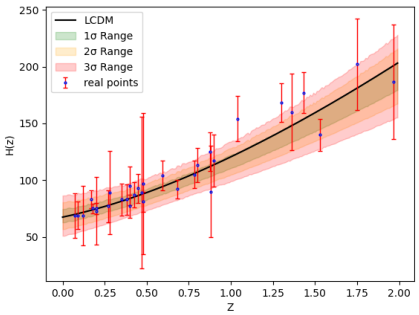
It is obvious that the Hubble constant H_0 is the value of H at the current time. In our reconstruction $H(z)$, we could obtain the H_0 at $z = 0$, which gave us $H_0 = 68.67 \pm 5.903$ km s^{-1} Mpc^{-1}. With CMB-based measurements [3, 4], it gives values of $H_0 = 67 \pm 3.2$ km s^{-1} Mpc^{-1}, to 4.8 per cent precision. The Hubble constant from our reconstructed H_0 is closed to the one from CMB-based calculation.

4.4 Estimation of the Cosmological Parameters with Reconstructed $H(z)$

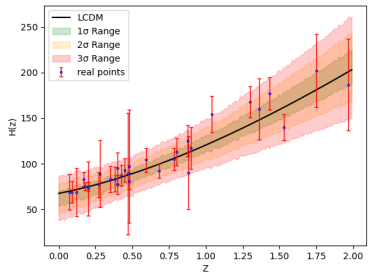
We sampled 32 data points randomly from the reconstructed $H(z)$ and applied them to constrain the cosmological parameters with MCMC method. The posterior distribution estimated by MCMC, which is shown in the Fig. 10a, gives the result of $H_0 = 68.03^{+3.51}_{-3.58}$ km s^{-1} Mpc^{-1}, $\Omega_m = 0.27^{+0.05}_{-0.05}$, $\Omega_\Lambda = 0.73^{+0.05}_{-0.05}$.

4.5 Estimation of the Cosmological Parameters with Mock $H(z)$

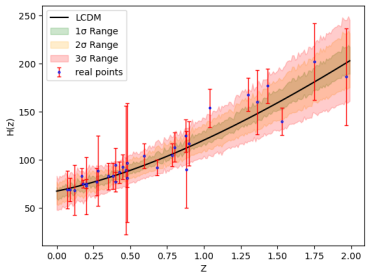
At the same time, in oder to do a better comparison, we also sampled 32 data points randomly from the mock $H(z)$ and applied to constrain the cosmological parameters with MCMC method. The posterior distribution estimated by MCMC, which is shown in the Fig. 10b, gives the result of $H_0 = 62.04^{+15.98}_{-13.7}$ km s^{-1} Mpc^{-1}, $\Omega_m = 0.38^{+0.30}_{-0.17}$, $\Omega_\Lambda = 0.62^{+0.17}_{-0.30}$.



(a) The reconstructed $H(z)$, and the 1σ, 2σ and 3σ. This result looks decent, and is quite similar to the Fig. 5b. It basically matches the ΛCDM. The shape is smooth.



(b) The reconstruction with loss dominates. The edge is not smooth.



(c) The reconstructed $H(z)$ reconstructed without covariance matrix. The shape of the $H(z)$ is mucher rougher.

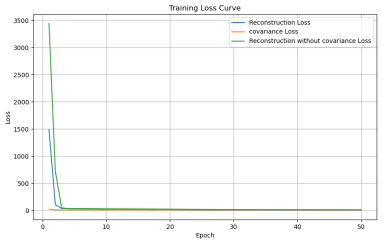
Fig. 8: Different kinds of reconstructions.

4.6 Comparision and Analysis of the Results

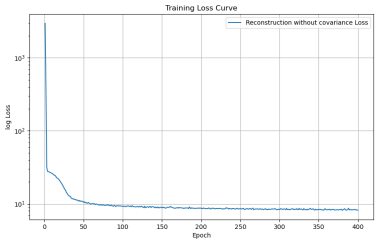
Besides, we also calculated the posterior distribution with the real 32 OHD points, we show the result in the Fig. 10c. The real 32 OHD points gives the result of $H_0 = 67.73^{+3.04}_{-3.10}$ km s^{-1} Mpc^{-1}, $\Omega_m = 0.33^{+0.07}_{-0.06}$, $\Omega_\Lambda = 0.73^{+0.06}_{-0.07}$.

It is obvious that the posterior distribution estimated with the data points from the reconstructed $H(z)$ is more closer to the roughly estimated values ($H_0 \approx 67 kms^{-1}Mpc^{-1}, \Omega_\Lambda \approx 0.7, \Omega_m \approx 0.3$), or more precisely, the ΛCDM model from Planck Collaboration [1], which gives the estimation of $H_0 \approx 67.4 kms^{-1}Mpc^{-1}, \Omega_\Lambda \approx 0.686, \Omega_m \approx 0.314$. At the same time, the confidence interval of the posterior distribution estimated with the data points from the reconstructed $H(z)$ is tighter.

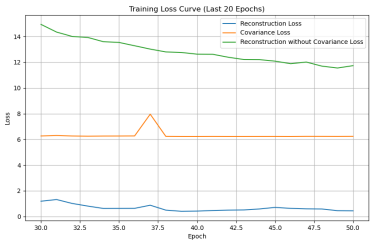
Furthermore, in order to test the quality of the reconstructed $H(z)$, we apply two criteria, KL divergence and figure of merit(FoM).



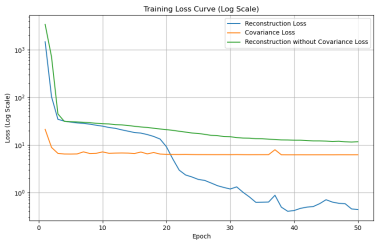
(a) The training loss curve of the reconstruction loss, covariance loss, and the reconstruction loss whitout covariance loss. They decrease rapidly and it is hard to see the difference, we show the detail in the following 3 figures with different methods.



(b) The reconstruction loss whitout covariance loss, we expand the x-axis up to 400. We can see that it took more epochs to obtain a minimum value.



(c) We show the last 20 epochs of the 3 loss, we can see that the the reconstruction loss whitout covariance loss still has a relatively large value.



(d) We use the Log Scale Curve. We can clearly see that the reconstruction loss whitout covariance loss decreases slowly comparing to the other 2 losses.

Fig. 9: The analysis of the training loss curve.

4.6.1 Comparison using KL divergence

Kullback-Leibler divergence (KL divergence). Kullback–Leibler divergence is a statistical feature which can measure how one probability distribution is different from another one. With this characteristic, Kullback–Leibler divergence is widely used to calculate how much information is lost when we try to approximate one distribution with the second one.

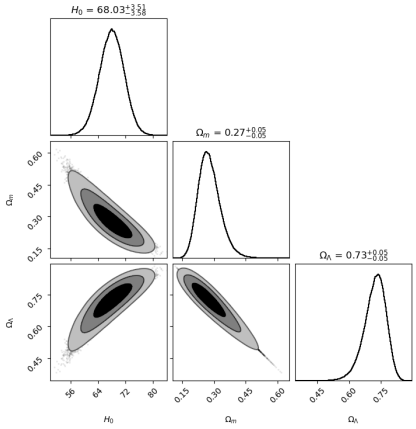
While processing probability and statistics, we generally can simulate the observed data or complex distribution with a simpler approximate distribution, which benefits our following experiment. Suppose that we have two probability density distributions $p_1(\theta)$ and $p_2(\theta)$, where $p_2(\theta)$ is the simulation of the $p_1(\theta)$. In this case, The KL divergence from $p_1(\theta)$ to $p_2(\theta)$ is defined as

$$D_{\mathrm{KL}} = (p_1(\theta)||p_2(\theta)) = \mathbb{E}_{p_1(\theta)}(\log p_1(\theta) - \log p_2(\theta)). \tag{25}$$

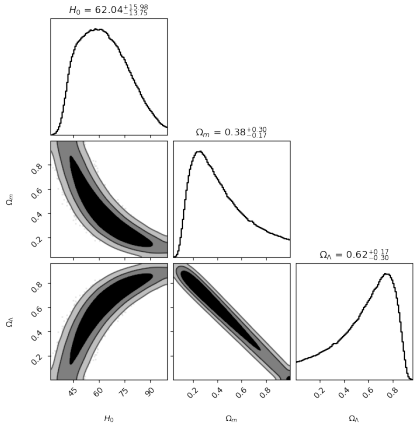
In our work, we sample M samples $\{\theta_i\}$ from the posterior, so the KL divergence is estimated with:

$$D_{\mathrm{KL}}(p_1||p_2) = \frac{1}{M}\sum_{i=1}^{M}(\ln P_1(\theta_i|\boldsymbol{H}_{\mathrm{obs}}) - \ln P_2(\theta_i|\boldsymbol{H}_{\mathrm{obs}})), \tag{26}$$

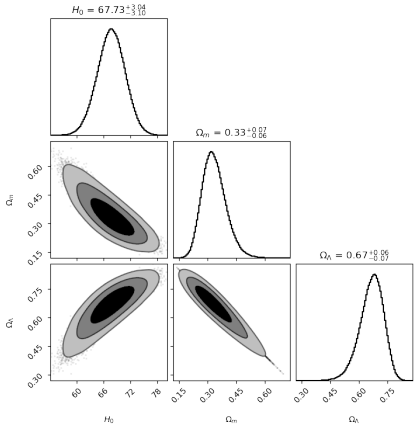
where $P_2(\theta|\boldsymbol{H}_{\mathrm{obs}})$ is the posterior calculated with reconstructed $H(z)$ and $P_1(\theta|\boldsymbol{H}_{\mathrm{obs}})$ is the posterior calculated with real 32 OHD data points. From Eq. (26), it is obvious that the



(a) Posterior distributions for 32 randomly sampled points from the reconstructed $H(z)$, derived with MCMC.



(b) MCMC-computed posterior distributions of 32 randomly selected data points from the mock $H(z)$.



(c) Posterior distributions for 32 OHD data points obtained via MCMC.

Fig. 10: MCMC-derived posterior distributions for: (1) the reconstructed data (Fig. 10a), (2) mock data (Fig. 10b), and (3) real observational data (Fig. 10c).

smaller the KL divergence is, the closer the $P_1(\theta|\boldsymbol{H}_{\rm moc})$ and $P_2(\theta|\boldsymbol{H}_{\rm moc})$ will be. When $D_{\rm KL}(p_1||p_2) = 0$, it means that the two posterior are almost identical.

In our work, we employed KL divergence to compare the posterior distributions of the cosmological parameters cosmological parameters Ω_m, Ω_Λ and H_0, which are inferred from samples based on the reconstructed $H(z)$ and the observed Hubble data (OHD). We sampled 10, 20, 32, 64, 128, data points respectively from the reconstructed $H(z)$ and calculated their posterior distributions. Then we calculated the KL divergence comparing to the posterior distribution calculated by the real 32 data points. As the number of sampled data points

increases, KL divergence decreases, which is reasonable because more data points generally means more accurate measurements. In Fig.11a, we show the KL divergence measurement.

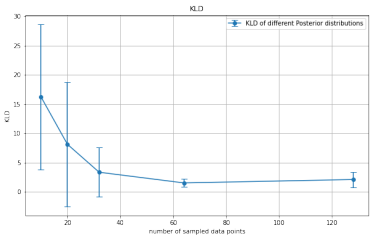
4.6.2 Comparison using FoM

Figure of merit(FoM). The FoM is a statistical feature which can measure the range of the parameters and how tight the constraints are. The FoM used in this work is similar to the one adopted by [37] and [64] in their work, which defined as:

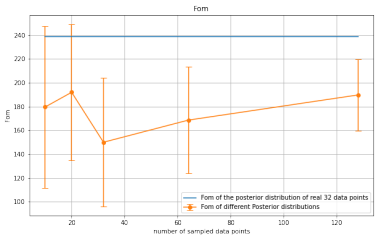
$$P(\theta|\boldsymbol{H}_{\rm obs}) = \text{const.} = \exp(-\Delta\mathscr{X}^2/2)P_{\rm max}, \tag{27}$$

where $P_{\rm max}$ is the maximum possibility density of the posterior, and $\exp(-\Delta\mathscr{X}^2/2)$ is a constant which ensures that $\exp(-\Delta\mathscr{X}^2/2)P_{\rm max}$ is equal to the probability density at the boundary of the 95.44% confidence region of the Gaussian distribution. According to [65], we use the same $\exp(-\Delta\mathscr{X}^2/2)$ here, which is 8.02. The FoM means the reciprocal volume of the confidence region of the posterior distribution, so the larger the FoM is, the tighter the constraint of the parameters is.

In Fig.11b, we show the FoM measurement. Unlike KLD measurement in Fig.11a, there is no obvious trend of change in different sampled data points. But we can see that as the number of sampled data points increases, the error decreases, which is also reasonable because more data points will bring us sRle range of the posterior distribution.



(a) KL divergence measurement. There is a obvious trend that when the number of the sampled data points increases, the KL divergence decreases, which is reasonable. In our work, we calculated the KL divergence based on the joint posterior distribution of the cosmological parameters Ω_m, Ω_Λ and H_0.



(b) FoM measurement doesn't have obvious trend of change in different sampled data points, but the error decreases when the sample data points increases. In our work, we calculated the FoM based on the joint posterior distribution of the cosmological parameters Ω_m, Ω_Λ and H_0.

Fig. 11: The measurement of the KL divergence and FOM.

5 Conclusions and Discussions

5.1 Conclusions

In our work, we had put forward a new method that can be used to reconstruct the $H(z)$. The effect of this method is obvious. One of the main purposes of our work is to estimate the cosmological parameters, so we sampled some data points from the reconstructed $H(z)$ and

constrainded in the ΛCDM model. We found that the posterior distribution calculated by the sampled reconstructed data is consistent with the ΛCDM model with Planck Collaboration [1]. The posterior distribution obtained from the reconstructed H(z) shows confidence intervals comparable in size to those derived from the real 32 OHD data points. Notably, both distributions yield consistent parameter ranges for the three cosmological parameters, and the shapes (Fig. 10a and Fig. 10c) are similar. We further tested the posteriors by using KL divergence and FoM, and the reconstructed data behaved well in the test. On the contrary, the shape and the range of the posterior distribution calculated by the mock $H(z)$ didn't behave well. At the same time, with the reconstructed $H(z)$, we can easily get the Hubble constant H_0, and the result is consistent with the Hubble constant calculated by the CMB-based measurements.

We combined the real observational data and covariance matrix to generate reconstructed $H(z)$ in a specific redshift range in $z \in [0,2]$. More precisely, we hope the reconstructed $H(z)$ can learn the characteristics of both Fig. 5b and GP. To some extent, it is still reasonable because when there has more data points, it is easlier to get a precise estimation. On the contrary, Fig. 5b lose the precision in order to construct the balanced shape of the distribution. For this reason, in our work we decided to combine the 2 distributions in Fig. 5b and GP.

In summary, our analysis demonstrates that the reconstructed $H(z)$ reliably captures the underlying distribution of observational data. As an accurate reconstruction of $H(z)$, it enables the exploration of important cosmological implications, including tighter constraints on the expansion history, the nature of dark energy, and potential deviations from the standard ΛCDM paradigm.

5.2 Discussions

We can see that the Bayesian inference results from the reconstruction and the real data are quite similar, but it doesn't mean that there is no added value in the proposed method. The purpose of our work is to reconstruct OHD data in the redshift range of [0,2] to compensate for the sparse data available in the redshift range, so we hope the Bayesian inference results from the reconstruction and the real data are consistent. Meanwhile, our reconstruction essentially is an extension based on the existing data. In principle, the Bayesian results obtained from the reconstructed data should not outperform those derived from the real data, as that would imply that additional information has been introduced.

Admittedly, we should pay attention to the construction of the covariance matrix we used as training data in our network, and see if we could find a better method to generate the covariance matrix. In the bigining, we tried to reconstruct the $H(z)$ in the redshitf range of $[0,2.5]$. However, when we uesd the GPR, the data diverged in the redshift range of $[2,2.5]$ because of having no data points in that redshift range. Consequently, it is hard for us to calculate the covariance matrix and the posterior distribution. For this reason, we had to limite the redshift in the range of $[0,2]$. In [23], they did a similar work, using VAE to generate the covariance matrix. Unlike their work, we directly input the covariance matrix to the neural network to help to generate reconstruction data. In [40–44], they rigorously computed the covariance matrix through a systematic approach that accounts for key astrophysical parameters, including metallicity, contamination, and Stellar Population Synthesis (SPS) model. While covariance matrix is reliable, it remains limited to a 15×15 dimensionality (shown in Fig. 7a), potentially restricting its applicability. However, this method is still significant because it provides reliable result. In the future, the method could be combined with additional OHD data points from upcoming surveys, such as CSST and Euclid, to

generate a precise and comprehensive covariance matrix, enabling us to generate more reliabe reconstruction.

Besides, ΛCDM is relatively easy to constrain. In the current study, our goal is to demonstrate and validate the proposed methodology within the well-established ΛCDM framework, which serves as a solid baseline. We consider the extension to other models such as CPL , to be a valuable direction for future work, and we plan to explore these in the following studies.

Meanwhile, ANN models usually need careful hyperparameter fine-tuning before the best performance can be achieved. Therefore, future work should be focused on finding some other better models, such as VEA-GAN [8, 46] and AVAE [52]. Exploring efficient hyperparameter tuning strategies, as systematically compared in [22], may further enhance performance. We hope we can find more useful models in the future work.

Acknowledgements We sincerely thank the anonymous reviewers for their thorough review and valuable feedback, which helped us clarify key points and improve the overall presentation of the paper. We thank Shiyu Li, Yunlong Li, Yu Hu, Changzhi Lu, Yuchen Wang, Shulei Cao, Jin Qin, Zhenzhao Tao, Zijian Wang, Xiaohang luan, Jing Niu, guangzai ye, zijian si and Kang Jiao for useful discussions and their kind help. This work is supported by National Key R&D Program of China (2023YFB4503305), National SKA Program of China (2022SKA0110202), the China Manned Space Program with grant No. CMS-CSST-2025-A01, the National Natural Science Foundation of China (Grants No.12373109, 61802428) and China Scholarship Council (File No.2306040042).

References

1. Aghanim, N., Y. Akrami, M. Ashdown, J. Aumont, C. Baccigalupi, M. Ballardini, A. Banday, R. Barreiro, N. Bartolo, S. Basak, *et al.* (2018), arXiv preprint arXiv:1807.06209 .
2. Andrieu, C., N. De Freitas, A. Doucet, and M. I. Jordan (2003), Machine learning **50**, 5.
3. Beutler, F., C. Blake, M. Colless, D. H. Jones, L. Staveley Smith, L. Campbell, Q. Parker, W. Saunders, and F. Watson (2011), Monthly Notices of the Royal Astronomical Society **416** (4), 3017.
4. Blake, C., E. A. Kazin, F. Beutler, T. M. Davis, D. Parkinson, S. Brough, M. Colless, C. Contreras, W. Couch, S. Croom, *et al.* (2011), Monthly Notices of the Royal Astronomical Society **418** (3), 1707.
5. Box, G. E., and G. C. Tiao (2011), *Bayesian inference in statistical analysis* (John Wiley & Sons).
6. Buhmann, M. D. (2000), Acta numerica **9**, 1.
7. Cabayol-Garcia, L., M. Eriksen, A. Alarcón, A. Amara, J. Carretero, R. Casas, F. J. Castander, E. Fernández, J. García-Bellido, E. Gaztanaga, *et al.* (2020), Monthly Notices of the Royal Astronomical Society **491** (4), 5392.
8. Cemgil, T., S. Ghaisas, K. Dvijotham, S. Gowal, and P. Kohli (2020), Advances in Neural Information Processing Systems **33**, 15077.
9. Chen, J.-F., Y.-C. Wang, T. Zhang, and T.-J. Zhang (2023), Physical Review D **107** (6), 063517.
10. Chen, Y., X. Dai, M. Liu, D. Chen, L. Yuan, and Z. Liu (2020), in *European conference on computer vision* (Springer) pp. 351–367.
11. Chib, S., and E. Greenberg (1995), The american statistician **49** (4), 327.
12. Christensen, N., R. Meyer, L. Knox, and B. Luey (2001), Classical and Quantum Gravity **18** (14), 2677.

13. Cong, S., and Y. Zhou (2023), Artificial Intelligence Review **56** (3), 1905.
14. Cong, Z., Z. Han, Y. Shuo, L. Siqi, Z. Tong-Jie, S. Yan-Chun, *et al.* (2014), Research in Astronomy and Astrophysics **14**.
15. Di Valentino, E., J. Levi Said, A. Riess, A. Pollo, V. Poulin, A. Gómez-Valent, A. Weltman, A. Palmese, C. D. Huang, C. van de Bruck, C. Shekhar Saraf, C.-Y. Kuo, C. Uhlemann, D. Grandón, D. Paz, D. Eckert, E. M. Teixeira, E. N. Saridakis, E. Ó. Colgáin, F. Beutler, F. Niedermann, F. Bajardi, G. Barenboim, G. Gubitosi, I. Musella, I. Banik, I. Szapudi, J. Singal, J. Haro Cases, J. Chluba, J. Torrado, J. Mifsud, K. Jedamzik, K. Said, K. Dialektopoulos, L. Herold, L. Perivolaropoulos, L. Zu, L. Galbany, L. Breuval, L. Visinelli, L. A. Escamilla, L. A. Anchordoqui, M. M. Sheikh-Jabbari, M. Lembo, M. G. Dainotti, M. Vincenzi, M. Asgari, M. Gerbino, M. Forconi, M. Cantiello, M. Moresco, M. Benetti, N. Schöneberg, Ö. Akarsu, R. C. Nunes, R. C. Bernardo, R. Chávez, R. I. Anderson, R. Watkins, S. Capozziello, S. Li, S. Vagnozzi, S. Pan, T. Treu, V. Irsic, W. Handley, W. Giarè, Y. Murakami, A. Poudou, A. Heavens, A. Kogut, A. Domi, A. Łukasz Lenart, A. Melchiorri, A. Vadalà, A. Amon, A. Bonilla, A. Reeves, A. Zhuk, A. Bonanno, A. Övgün, A. Pisani, A. Talebian, A. Abebe, A. Aboubrahim, A. L. González Morán, A. Kovács, A. Papatriantafyllou, A. R. Liddle, A. Paliathanasis, A. Borowiec, A. K. Yadav, A. Yadav, A. A. Sen, A. J. W. Mini Latha, A. C. Davis, A. J. Shajib, A. Walters, A. Idicherian Lonappan, A. Chudaykin, A. Capodagli, A. da Silva, A. De Felice, A. Racioppi, A. Soler Oficial, A. Montiel, A. Favale, A. Bernui, A. C. Velasco, A. Heinesen, A. Bakopoulos, A. Chatzistavrakidis, B. Khanpour, B. S. Sathyaprakash, B. Zgirski, B. L'Huillier, B. Famaey, B. Jain, B. Marek, B. Zhang, B. Karmakar, B. Dragovich, B. Thomas, C. Correa, C. G. Boiza, C. Marques, C. Escamilla-Rivera, C. Tzerefos, C. Zhang, C. De Leo, C. Pfeifer, C. Lee, C. Venter, C. Gomes, C. Roque De bom, C. Moreno-Pulido, D. Iosifidis, D. Grin, D. Blixt, D. Scolnic, D. Oriti, D. Dobrycheva, D. Bettoni, D. Benisty, D. Fernández-Arenas, D. L. Wiltshire, D. Sanchez Cid, D. Tamayo, D. Valls-Gabaud, D. Pedrotti, D. Wang, D. Staicova, D. Totolou, D. Rubiera-Garcia, D. Milaković, D. Pesce, D. Sluse, D. Borka, E. Yusofi, E. Giusarma, E. Terlevich, E. Tomasetti, E. C. Vagenas, E. Fazzari, E. G. M. Ferreira, E. Barakovic, E. Dimastrogiovanni, E. Brinch Holm, E. Mottola, E. Özülker, E. Specogna, E. Brocato, E. Jensko, E. Antonette Enriquez, E. Bhatia, F. Bresolin, F. Avila, F. Bouchè, F. Bombacigno, F. K. Anagnostopoulos, F. Pace, F. Sorrenti, F. S. N. Lobo, F. Courbin, F. K. Hansen, G. Sloan, G. Farrugia, G. Lynch, G. Garcia-Arroyo, G. Raimondo, G. Lambiase, G. S. Anand, G. Poulot, G. Leon, G. Kouniatalis, G. Nardini, G. Csörnyei, G. Galloni, and G. Bargiacchi (2025), arXiv e-prints , arXiv:2504.01669arXiv:2504.01669 [astro-ph.CO] .
16. Fan, E. (2002), Physics Letters A **300** (2-3), 243.
17. Foreman-Mackey, D., D. W. Hogg, D. Lang, and J. Goodman (2013), Publications of the Astronomical Society of the Pacific **125** (925), 306.
18. Freedman, W. L., and B. F. Madore (2010), Annual Review of Astronomy and Astrophysics **48** (1), 673.
19. Friedman, A. (1922), Zeitschrift für Physik **10** (1), 377.
20. Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2014), Advances in neural information processing systems **27**.
21. Goodman, J., and J. Weare (2010), Communications in applied mathematics and computational science **5** (1), 65.
22. Gómez-Vargas, I., J. Briones Andrade, and J. A. Vázquez (2023), Physical Review D **107** (4), 10.1103/physrevd.107.043509.

23. Gómez-Vargas, I., R. Medel-Esquivel, R. García-Salcedo, and J. A. Vázquez (2023), The European Physical Journal C **83** (4), 10.1140/epjc/s10052-023-11435-9.
24. He, F., T. Liu, and D. Tao (2019), Advances in neural information processing systems **32**.
25. Hecht-Nielsen, R. (1992), in *Neural networks for perception* (Elsevier) pp. 65–93.
26. Jackson, N. (2015), Living reviews in relativity **18** (1), 2.
27. Jesus, J. F., T. Gregório, F. Andrade-Oliveira, R. Valentim, and C. Matos (2017), Monthly Notices of the Royal Astronomical Society (3), 3.
28. Jiao, K., N. Borghi, M. Moresco, and T.-J. Zhang (2023), The Astrophysical Journal Supplement Series **265** (2), 48.
29. Jimenez, R., L. Verde, T. Treu, and D. Stern (2003), The Astrophysical Journal **593** (2), 622.
30. Jimenez, R., L. Verde, T. Treu, and D. Stern (2003), The Astrophysical Journal **593** (2), 622.
31. Kenton, J. D. M.-W. C., and L. K. Toutanova (2019), in *Proceedings of naacL-HLT*, Vol. 1 (Minneapolis, Minnesota).
32. Kyurkchiev, N., and S. Markov (2015), LAP LAMBERT Academic Publishing, Saarbrucken **4**.
33. Lewis, A., and S. Bridle (2002), Physical Review D **66** (10), 103511.
34. Lu, C.-Z., K. Jiao, T. Zhang, T.-J. Zhang, and M. Zhu (2022), Physics of the Dark Universe **37**, 101088.
35. Lu, C.-Z., T. Zhang, and T.-J. Zhang (2022), arXiv preprint arXiv:2208.05639 .
36. Luan, X.-H., Z.-Z. Tao, H.-C. Zhao, B.-L. Huang, S.-Y. Li, C. Liu, H.-F. Wang, W.-F. Liu, T.-J. Zhang, V. Gajjar, *et al.* (2023), The Astronomical Journal **165** (3), 132.
37. Ma, C., and T.-J. Zhang (2011), The Astrophysical Journal **730** (2), 74.
38. MacKay, D. J., *et al.* (1998), NATO ASI series F computer and systems sciences **168**, 133.
39. Moresco, M. (2015), Monthly Notices of the Royal Astronomical Society: Letters **450** (1), L16, https://academic.oup.com/mnrasl/article-pdf/450/1/L16/3083577/slv037.pdf .
40. Moresco, M. (2024), "Measuring the expansion history of the universe with cosmic chronometers," arXiv:2412.01994 [astro-ph.CO] .
41. Moresco, M., A. Cimatti, R. Jimenez, L. Pozzetti, G. Zamorani, M. Bolzonella, J. Dunlop, F. Lamareille, M. Mignoli, H. Pearce, *et al.* (2012), Journal of Cosmology and Astroparticle Physics **2012** (08), 006.
42. Moresco, M., R. Jimenez, L. Verde, A. Cimatti, and L. Pozzetti (2020), The Astrophysical Journal **898** (1), 82.
43. Moresco, M., R. Jimenez, L. Verde, L. Pozzetti, A. Cimatti, and A. Citro (2018), The Astrophysical Journal **868** (2), 84.
44. Moresco, M., L. Pozzetti, A. Cimatti, R. Jimenez, C. Maraston, L. Verde, D. Thomas, A. Citro, R. Tojeiro, and D. Wilkinson (2016), Journal of Cosmology and Astroparticle Physics **2016** (05), 014.
45. Moresco, M., L. Verde, L. Pozzetti, R. Jimenez, and A. Cimatti (2012), Journal of Cosmology and Astroparticle Physics **2012** (07), 053.
46. Mukesh, K., S. Ippatapu Venkata, S. Chereddy, E. Anbazhagan, and I. Oviya (2022), in *International Conference on Innovative Computing and Communications: Proceedings of ICICC 2022, Volume 1* (Springer) pp. 761–768.
47. Niu, J., and T.-J. Zhang (2023), Physics of the Dark Universe **39**, 101147.
48. Norris, J. R. (1998), *Markov chains*, 2 (Cambridge university press).
49. Oliphant, T. E., *et al.* (2006), *Guide to numpy*, Vol. 1 (Trelgol Publishing USA).

50. Pan, S., M. Liu, J. Forero-Romero, C. G. Sabiu, Z. Li, H. Miao, and X.-D. Li (2020), Science China Physics, Mechanics, and Astronomy **63** (11), 110412, arXiv:1908.10590 [astro-ph.CO] .
51. Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.* (2011), the Journal of machine Learning research **12**, 2825.
52. Plumerault, A., H. Le Borgne, and C. Hudelot (2021), in *2020 25th International Conference on Pattern Recognition (ICPR)* (IEEE) pp. 8687–8694.
53. Ramachandran, P., B. Zoph, and Q. V. Le (2017), arXiv preprint arXiv:1710.05941 .
54. Ratsimbazafy, A., S. Loubser, S. Crawford, C. Cress, B. Bassett, R. Nichol, and P. Väisänen (2017), Monthly Notices of the Royal Astronomical Society **467** (3), 3239.
55. Schulz, E., M. Speekenbrink, and A. Krause (2018), Journal of mathematical psychology **85**, 1.
56. Scolnic, D. M., D. O. Jones, A. Rest, Y. C. Pan, R. Chornock, R. J. Foley, M. E. Huber, R. Kessler, G. Narayan, and A. G. Riess (2017), Astrophysical Journal .
57. Shyam, R. (2021), Journal of Computer Technology & Applications **12** (2), 6.
58. Simon, J., L. Verde, and R. Jimenez (2005), Phys. Rev. D **71**, 123001.
59. Stern, D., R. Jimenez, L. Verde, M. Kamionkowski, and S. A. Stanford (2009), journal of cosmology and astroparticle physics .
60. Tao, Z.-Z., H.-C. Zhao, T.-J. Zhang, V. Gajjar, Y. Zhu, Y.-L. Yue, H.-Y. Zhang, W.-F. Liu, S.-Y. Li, J.-C. Zhang, *et al.* (2022), The Astronomical Journal **164** (4), 160.
61. Tosi, S. (2009), *Matplotlib for Python developers* (Packt Publishing Ltd).
62. Virtanen, P., R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, *et al.* (2020), Nature methods **17** (3), 261.
63. Wang, G.-J., X.-J. Ma, S.-Y. Li, and J.-Q. Xia (2020), The Astrophysical Journal Supplement Series **246** (1), 13.
64. Wang, H., and T. J. Zhang (2011), The Astrophysical Journal **748** (2), 315.
65. Wang, Y. C., Y. B. Xie, T. J. Zhang, H. C. Huang, T. Zhang, and K. Liu (2021), The Astrophysical Journal Supplement Series **254** (2), 43 (16pp).
66. Williams, C., and C. Rasmussen (1995), Advances in neural information processing systems **8**.
67. Yamashita, R., M. Nishio, R. K. G. Do, and K. Togashi (2018), Insights into imaging **9**, 611.
68. Yegnanarayana, B. (2009), *Artificial neural networks* (PHI Learning Pvt. Ltd.).
69. You, Y., I. Gitman, and B. Ginsburg (2017), arXiv preprint arXiv:1708.03888 .
70. Yu, H.-R., J. Emberson, D. Inman, T.-J. Zhang, U.-L. Pen, J. Harnois-Déraps, S. Yuan, H.-Y. Teng, H.-M. Zhu, X. Chen, *et al.* (2017), Nature Astronomy **1** (7), 0143.
71. Yuan, S., and T.-J. Zhang (2015), Journal of Cosmology and Astroparticle Physics **2015** (02), 025.
72. Zhang, H., Y.-C. Wang, T.-J. Zhang, and T. Zhang (2023), The Astrophysical Journal Supplement Series **266** (2), 27.
73. Zhang, J.-C., Y. Hu, K. Jiao, H.-F. Wang, Y.-B. Xie, B. Yu, L.-L. Zhao, and T.-J. Zhang (2023), arXiv preprint arXiv:2311.13938 .
74. Zhang, J.-C., Y. Hu, K. Jiao, H.-F. Wang, Y.-B. Xie, B. Yu, L.-L. Zhao, and T.-J. Zhang (2024), The Astrophysical Journal Supplement Series **270** (2), 23.