The Astrophysical Journal, 985:132 (11pp), 2025 May 20 https://doi.org/10.3847/1538-4357/adc99d

Cosmology from Point Clouds with Dark Matter Halos from the Quijote Simulations

Atrideb Chatterjee[1,2] and Francisco Villaescusa-Navarro[3,4]
[1] Inter-University Centre for Astronomy and Astrophysics, Post Bag 4, Ganeshkhind, Pune 411007, India; atrideb.chatterjee@iucaa.in
[2] Kapteyn Astronomical Institute, University of Groningen, PO Box 800, 9700 AV Groningen, The Netherlands
[3] Center for Computational Astrophysics, Flatiron Institute, 162 5th Avenue, New York, NY 10010, USA
[4] Department of Astrophysical Sciences, Princeton University, 4 Ivy Lane, Princeton, NJ 08544, USA
Received 2024 May 21; revised 2025 March 28; accepted 2025 April 4; published 2025 May 20

Abstract

We train a novel deep learning architecture to perform likelihood-free inference on the value of the cosmological parameters from halo catalogs of the `Quijote` *N*-body simulations. Our model takes as input a halo catalog where each halo is characterized by its position, mass, and velocity modulus. By construction, our model is E(3) invariant and is designed to extract information hierarchically. Unlike graph neural networks, it does not require the transformation of the input halo (or galaxy) catalog into a graph. Given its simplicity, our model can process point clouds with large numbers of points. We discuss the advantages of this class of methods but also point out their limitations and potential ways to improve them for cosmological data.

Unified Astronomy Thesaurus concepts: N-body simulations (1083); Cosmological parameters from large-scale structure (340)

1. Introduction

Our ability to characterize and understand our Universe depends on the level of accuracy to which the value of the different cosmological parameters can be constrained. The spatial distribution of galaxies in the sky is influenced by various cosmological parameters. As a result, galaxy clustering has become a common means by which to study the laws and components of the Universe (V. Ajani et al. 2020; C. Uhlemann et al. 2020; F. Villaescusa-Navarro et al. 2020; D. Gualdi et al. 2021; G. Valogiannis & C. Dvorkin 2022; W. Liu et al. 2022). Upcoming sky surveys—such as CMB-S4 (K. Abazajian et al. 2022), the Roman Space Telescope (C. Y. Lam et al. 2023), the Dark Energy Spectroscopic Instrument (DESI; DESI Collaboration et al. 2016), Euclid (R. Laureijs et al. 2011; G. D. Racca et al. 2016; Euclid Collaboration et al. 2023), the Large Synoptic Survey Telescope (Ž. Ivezić et al. 2019), and J-PAS (N. Benitez et al. 2014)—will gather data with unprecedented precision. This will make galaxy clustering an especially powerful method for advancing precision cosmology.

In view of this, a large number of efforts using different machine learning techniques (S. Ravanbakhsh et al. 2017; J. Fluri et al. 2019; N. Gillet et al. 2019; N. Jeffrey et al. 2021; H. J. Hortua 2021; P. Villanueva-Domingo et al. 2021; S. Anagnostidis et al. 2022; S. Hassan et al. 2022; T. L. Makinen et al. 2022; P. Villanueva-Domingo & F. Villaescusa-Navarro 2022; F. Villaescusa-Navarro et al. 2022c; C. Cuesta-Lazaro & S. Mishra-Sharma 2024; A. Roncoli et al. 2023; M. Ho et al. 2024) have been put forward to make optimal use of these upcoming data. The unique ability of these techniques to learn extremely complicated features from a given data set has given them a decisive advantage over traditional methods that employ summary statistics (G. A. Marques et al. 2019; O. Friedrich et al. 2020; U. Giri & K. M. Smith 2020; D. Gualdi et al. 2021; C. Hahn et al. 2020; F. Villaescusa-Navarro et al. 2020; A. Banerjee & T. Abel 2021; A. E. Bayer et al. 2021; J. Harnois-Déraps et al. 2021; K. Naidoo et al. 2021; L. Samushia et al. 2021; M. Eickenberg et al. 2022). On top of that, these methods can potentially address complex problems, such as marginalizing over baryonic effects at the field level, that are quite challenging with standard methods (P. Villanueva-Domingo & F. Villaescusa-Navarro 2022; N. S. M. de Santi et al. 2023).

Data collected from telescopes are often released as catalogs of sources containing their positions and different properties rather than images of a particular area of the sky. These galaxy catalogs, with their inherent sparse and irregular nature, are difficult/suboptimal to use as the input for any image-based network.[5] To address this issue, a new method for sparse data, called point cloud analysis, has emerged as a new avenue for 3D data analysis (C. R. Qi et al. 2016; S. Shi et al. 2018; C. Xu et al. 2020; P. Villanueva-Domingo et al. 2021; T. L. Makinen et al. 2022; V. Delle Rose et al. 2023; S. Hordan et al. 2023, 2024). Further, with its rapid improvement, point cloud analysis has become very popular, performing better than or at par with other state-of-the-art image-based analysis methods (D. Maturana & S. Scherer 2015; C. R. Qi et al. 2017; S. Fan et al. 2021).

In this work, we employ a novel architecture that takes a point cloud as input and processes it efficiently using multilayer perceptrons (MLPs), whereas graph neural networks (GNNs) rely on graph-based structures and message-passing. A key advantage of this model is its ability to extract information hierarchically, enabling the efficient processing of relatively large point clouds. We note that some previous works exploited complicated geometrical feature extractors to process point clouds (G. Li et al. 2019; M.-H. Guo et al. 2020). Recently, X. Ma et al. (2022) proposed a novel method that takes point clouds as input and processes them in a very similar fashion to

[5] On the one hand, one can place the observed catalog in a regular 2D or 3D grid at the expense of throwing away information on scales below the grid size. On the other hand, if the grid is sufficiently fine (i.e., containing at most one point per pixel), it will be a big grid filled with zeros, given the sparse nature of the data.

GNNs (i.e., with simple, flexible, and learnable geometrical kernels) without having to predefine a graph from the point cloud. They have shown that their method either outperforms or performs at par with other state-of-the-art point cloud networks. Further, due to employing a simple feature extractor, this network is lightweight and fast compared to other 3D data analysis methods.

In this work, we perform parameter inference from point clouds using the method proposed by X. Ma et al. (2022) but enhancing it to make it E(3) invariant. Our data set is composed of halo catalogs from *N*-body simulations of the `Quijote` suite (F. Villaescusa-Navarro et al. 2020). We show that this network performs well when constraining the value of some parameters but fails on others. Besides, its performance against classical summary statistics is not good. We discuss in detail the limitations of the model and potential avenues to improve it.

The outline of the paper is as follows. We describe the data we use in Section 2. In Section 3, we describe the model architecture, the way we impose the symmetries, the loss function used, the training procedure, and the validation metrics. We show the results we achieve from training the networks in Section 4. We then discuss the limitations and future directions in Section 5. We draw the main conclusions of this work in Section 6.

2. Data

In this paper, we work with 3D point clouds. A point cloud contains N points, and each point is characterized by its 3D position (x, y, z). The points can also have other properties, such as masses and peculiar velocity moduli.[6] Each point cloud is fully described by five labels: the values of the cosmological parameters. In this work, we will, however, restrict ourselves to just two of these parameters: $\Omega_{\rm m}$ and σ_8.

A point cloud is constructed as follows. First, all halos from an *N*-body simulation are read. Next, we take the N more massive halos, and for each halo, we store its position, mass, and velocity modulus. The labels of the point cloud are the value of the cosmological parameters of that halo catalog.

The halo catalogs are obtained from the `Quijote` simulations (F. Villaescusa-Navarro et al. 2020) at $z = 0$. In particular, we use the fiducial-resolution[7] latin-hypercubes that contain 2000 *N*-body simulations whose parameters are varied within:

$$0.1 \leqslant \Omega_{\rm m} \leqslant 0.5 \tag{1}$$

$$0.03 \leqslant \Omega_{\rm b} \leqslant 0.07 \tag{2}$$

$$0.5 \leqslant h \leqslant 0.9 \tag{3}$$

$$0.8 \leqslant n_s \leqslant 1.2 \tag{4}$$

$$0.6 \leqslant \sigma_8 \leqslant 1.0 \tag{5}$$

Thus, our data set contains 2000 point clouds, and each point cloud can have $N = 1024$, 4096, or 8192 points.

3. Methodology

In this section, we describe our model's architecture, how we make it E(3) invariant, the loss function we employ, the training and validating procedure, and the validation metrics we use to quantify the model's performance.

3.1. Architecture

This work aims to infer the value of the cosmological parameters from the spatial distribution of dark matter halos of *N*-body simulations, represented as point clouds.

For each simulation, we have a collection of N points $\{p_1, p_2, \ldots, p_n\}$ where each point represents a dark matter halo. We will denote the properties of the point i by $f_i \in \mathbb{R}^d$. The properties of the jth closest neighbor of the point i are instead represented by $f_{i,j} \in \mathbb{R}^d$. The dimensionality of the input feature vector is

1. $d = 3$ when only considering halo positions,
2. $d = 4$ when using positions and masses,
3. $d = 5$ when using halo positions, masses, and velocity moduli.

Our model takes a point cloud as input and returns two values for each cosmological parameter i—μ_i and σ_i—representing the marginal posterior mean and standard deviation:

$$g\colon \mathbb{R}^{N\times d} \rightarrow \mathbb{R}^{2\times N_\theta}, \tag{6}$$

where N_θ is the number of cosmological parameters. The architecture of our models closely follows[8] the `PointMLP-elite`[9] model recently introduced by X. Ma et al. (2022), and consists of (1) an initial embedding, (2) a series of four *stages*, and (3) an inference module. We show a scheme of the model architecture in Figure 1. We now describe each of these elements.

1. *Embedding*. A 1D convolution that is applied to every point in the catalog and whose purpose is to embed the catalog in a higher dimension (32 in our case).
2. *Stage*. At the beginning of a stage, a subset of n points is selected using the farthest point sampling (FPS) algorithm. In our case, we select half of the input points. Next, the features of the selected point are updated as follows. Given a point i at stage l with features $f_i^l \in \mathbb{R}^m$, the features of its k nearest neighbors $f_{i,j}^l$ are first transformed using a *geometric affine module*:

$$\hat{f}_{i,j}^l = \alpha \odot \frac{f_{i,j}^l - f_i^l}{\sigma + \delta} + \beta \tag{7}$$

where $\alpha, \beta \in \mathbb{R}^m$ are learnable parameters, $\odot$ is the Hadamard product and σ is a normalization defined as

$$\sigma = \sqrt{\frac{1}{k \times n \times m}\sum_{i=1}^{n}\sum_{j=1}^{k}(f_{i,j}^l - f_i^l)^2}\ . \tag{8}$$

$\delta = 10^{-5}$ is added for numerical stability. As mentioned by X. Ma et al. (2022) the affine transformation is introduced to transform the local points to a normal

[6] This is defined as $v = \sqrt{v_x^2 + v_y^2 + v_z^2}$ where v_x, v_y, and v_z are the Cartesian components of the peculiar velocity.

[7] Using a higher-resolution latin-hypercube will not have an impact on the results of this paper as we only use the most massive halos that are well resolved at this resolution.

[8] While the model architecture is the same as given by X. Ma et al. (2022), the loss function and the E(3)-invariant point features are modifications introduced in this work.

[9] https://github.com/ma-xu/pointMLP-pytorch

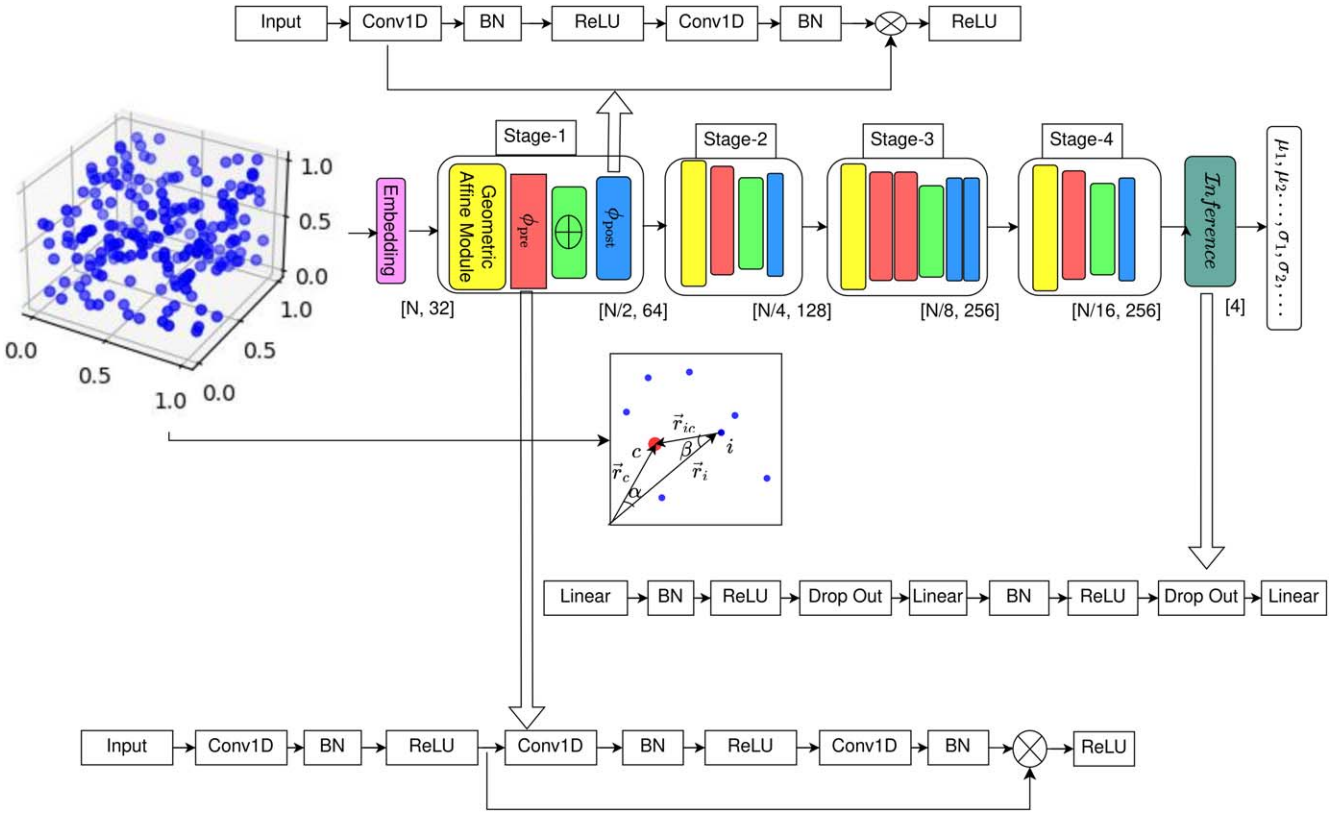

Figure 1. Structure of the network we use in this work. The model takes a point cloud consisting of N points, each of them characterized by 3D positions (and masses and velocity moduli). The 3D position of each point is expressed as a distance and two angles (see central panel), making our model E(3) invariant. The embedding layer performs a 1D convolution and transforms the dimensionality of every point to 32. Next, the point cloud is passed through four *stages* that capture the geometrical information on different scales. Finally, an inference layer consisting of MLPs will transform the data into a vector containing the posterior mean and standard deviation of each considered parameter.

distribution which in turn makes the network more robust, stable, and accurate. Next, we update the features of point i using the transformed properties of its k nearest neighbors via

$$f_i^{l+1} = \Phi_{\rm pos} \bigoplus_{j \in \mathcal{N}_i} \Phi_{\rm pre}(\hat{f}_{i,j}^{l}) \,, \tag{9}$$

where $\Phi_{\rm pos}$ and $\Phi_{\rm pre}$ are the so-called preblock and postblock modules. Those contain 1D convolutions, batch norm, activation, and residual layers (see Figure 1 for details). $\bigoplus$ is a permutation-invariant operation (the maximum in our case) that operates over the neighbors of point i, $\mathcal{N}_i$. After one stage, we increase the dimensionality of f_i^{l+1} by a factor of 2 with respect to f_i^l.

The features of the points are updated through four different stages as illustrated in Figure 1. Note that in the third stage, we use two preblocks and two postblocks instead of one. We emphasize that due to the hierarchical structure of the network, if we start with N points, we will have only $N/2^4$ points available after the four stages. We also note that the number of points to subsample at each stage can vary and does not need to be set to one-half.

3. *Inference*. This part contains several MLP layers (linear + batch norms + activations + dropout) (see Figure 1) that will transform the remaining $N/2^4$ points from the last stage into $2 \times N_\theta$ numbers representing the first two moments of the posterior for each parameter.

3.2. Symmetries

The values of the cosmological parameters of a set of dark matter halos are not affected if we translate and/or rotate their positions. These symmetries can be learned in our model through data augmentation or imposed directly into the model. In our case, we decide to impose them using invariant input features of the point cloud. We checked that the data augmentation (random rotation, translation, etc.) increases the training time of the model significantly, whereas making the input features E(3) invariant bypasses the need for any data augmentation and makes the training much faster.

To achieve this, we take inspiration from P. Villanueva-Domingo et al. (2022) and we transform the Cartesian position coordinate of each halo into a new E(3)-invariant coordinate system (see Figure 1) following these steps:

1. We first calculate the position of the centroid of the distribution of the halos[10] (red point denoted by c in the bottom panel of Figure 1). Next, we define a vector $\boldsymbol{r}_{ic}$ joining the ith halo to the centroid, i.e., $\boldsymbol{r}_{ic} = \boldsymbol{r}_i - \boldsymbol{r}_c$, where $\boldsymbol{r}_i$ and $\boldsymbol{r}_c$ are respectively the positions of ith halo and the centroid of the point cloud system.
2. Next, the distance $d_{ic} = |\boldsymbol{r}_{ic}|$ between the centroid and the ith halo is computed.
3. We then determine the cosines of the angles α and β (as shown in the bottom panel of Figure 1) by taking the scalar product between the vectors ($\boldsymbol{r}_i$, $\boldsymbol{r}_c$) and ($\boldsymbol{r}_i$, $\boldsymbol{r}_{ic}$)

[10] We note that in principle we can take any other point inside the box.

respectively, i.e., $\cos\alpha = \boldsymbol{r}_i \cdot \boldsymbol{r}_c/|\boldsymbol{r}_i||\boldsymbol{r}_c|$ and $\cos\beta = \boldsymbol{r}_i \cdot \boldsymbol{r}_{ic}/|\boldsymbol{r}_i||\boldsymbol{r}_{ic}|$.

4. Finally, we represent the position of the ith halo by $(d_{ic}, \cos\alpha, \cos\beta)$ and employ these coordinates as the features for the 3D position rather than the Cartesian coordinates (x, y, z).
5. We emphasize that we take into account periodic boundary conditions when computing distances and angles between vectors.

One may wonder if the affine transformation of Equation (7) may not make sense in the new coordinate system, given the fact that point properties will no longer be zero-normalized. We have checked that our results are pretty insensitive to this.

3.3. Loss Function

Our model is trained to perform likelihood-free inference on the value of the cosmological parameters. Given a set of N points, the above architecture outputs $2 \times N_\theta$ numbers. For each parameter, the model predicts the marginal posterior mean (μ_i) and standard deviation (σ_i), defined as

$$\mu_i(\mathcal{P}) = \int_{\theta_i} p(\theta_i|\mathcal{P})\theta_i d\theta_i\,, \tag{10}$$

$$\sigma_i^2(\mathcal{P}) = \int_{\theta_i} p(\theta_i|\mathcal{P})(\theta_i - \mu_i)^2 d\theta_i\,, \tag{11}$$

where $\mathcal{P}$ represents a point cloud. This is accomplished by minimizing the following loss function (N. Jeffrey & B. D. Wandelt 2020; F. Villaescusa-Navarro et al. 2022a):

$$\mathcal{L} = \sum_{i=1}^{N_\theta} \log\left(\sum_{j\in\text{batch}} (\theta_{i,j} - \mu_{i,j})^2\right) + \sum_{i=1}^{N_\theta} \log\left(\sum_{j\in\text{batch}} ((\theta_{i,j} - \mu_{i,j})^2 - \sigma_{i,j}^2)^2\right), \tag{12}$$

where $\theta_{i,j}$, $\mu_{i,j}$, and $\sigma_{i,j}$ represent the true value, inferred mean, and inferred standard deviation of the parameter i for the sample j. The reason for using this loss function is that it bypasses the need to compute the posterior density of each parameter, and we directly obtain the estimates of moments of the marginalized distribution of all the parameters.

3.4. Training Procedure

We split the 2000 point clouds available into training (80%), validation (10%), and testing (10%). Each point cloud can have $N = \{1024, 4096, 8192\}$ points and the number of neighbors is set to either $k = \{32, 64\}$. The features of the points can be (i) their positions $f_i \in \mathbb{R}^3$ (referred to as Pos-only hereafter), (ii) their positions and masses $f_i \in \mathbb{R}^4$ (hereafter Pos+M), or (iii) their positions, masses, and velocity moduli $f_i \in \mathbb{R}^5$ (Pos+M +V hereafter). All features are normalized to have their values between 0 and 1. In the case of the masses, we first compute their $\log_{10}$, and then normalize them with min–max. We also normalize the value of the output features representing the cosmological parameters to be between 0 and 1.

Next, the point clouds in the training set are passed through the architecture depicted in Section 3.1, and the loss function of Equation (12) is minimized through gradient descent. The network weights are optimized using the SGD optimizer (S. Ruder 2016) with a Nesterov momentum of 0.9. We also used the cosine annealing scheduler (I. Loshchilov & F. Hutter 2016), which modulates the learning rate during training. We use a batch size of 32 and train the model for 250 epochs.

We consider three hyperparameters for our model: (i) the maximum learning rate, (ii) the minimum learning rate, and (iii) the value of the weight decay. We optimize the value of these hyperparameters for each configuration (i.e., for a given total number of points and neighbors) using the hyperparameter optimization software `OPTUNA` (T. Akiba et al. 2019). The optimization is performed by minimizing the validation loss, and we carry out at least 100 trials.

We have studied the impact of the total number of halos considered, N, and the number of closest neighbors, k, on our results. From now on, we will refer to a configuration containing N halos with k neighbors as N_k. For example, 1024_{64} denotes point clouds containing 1024 halos and considering 64 neighbors.

3.5. Validation Metrics

We employ four statistics to quantify the accuracy and precision of our models and compare our results with other recent studies performed by different groups. Let us consider a point cloud from the test set $\mathcal{P}_i$, which contains N point clouds, each of them characterized by the true value of the j cosmological parameters θ_{ij}. When $\mathcal{P}_i$ is passed through the model, we will obtain a posterior mean and standard deviation for each parameter μ_{ij} and σ_{ij}.

We can use the four metrics below to quantify the accuracy and precision of the model.

1. The mean relative error, ϵ_i defined as
$$\epsilon_i = \frac{1}{N}\sum_{j=1}^{N} \frac{|\theta_{ij} - \mu_{ij}|}{\theta_{ij}}. \tag{13}$$
The smaller the value of ϵ_i, the more precise the network is.
2. The coefficient of determination, R_i^2, defined as
$$R_i^2 = 1 - \frac{\sum_{j=1}^{N}(\theta_{ij} - \mu_{ij})^2}{\sum_{j=1}^{N}(\theta_{ij} - \overline{\theta}_i)^2}, \tag{14}$$
where $\overline{\theta}_i = \sum_{j=1}^{N}\theta_{ij}$. The closer the value of R_i^2 is to 1, the more accurate the network is.
3. The mean squared error MSE_i is defined as
$$\text{MSE}_i = \frac{1}{N}\sum_{j=1}^{N}(\theta_{ij} - \mu_{ij})^2. \tag{15}$$
The smaller the value of the mean squared error, the more accurate the network is.
4. The χ^2 value, χ_i^2, which is defined as
$$\chi_i^2 = \frac{1}{N}\sum_{j=1}^{N}\frac{(\theta_{ij} - \mu_{ij})^2}{\sigma_{ij}^2}. \tag{16}$$
We use this statistic to quantify the quality of the network errors. Values close to 1 indicate that network errors are correctly calibrated.

Table 1
Summary of the Main Results of This Work

Configuration	Features	Ω_m				σ_8			
		ϵ (%)	R^2	χ^2	MSE	ϵ (%)	R^2	χ^2	MSE
1024	2ptCF	17.4	0.75	1.05	3.4×10^{-3}	11.4	0.08	1.10	1.4×10^{-2}
1024_{32}	Pos-only	21.8	0.69	0.95	4.4×10^{-3}	13.4	0.00	1.10	1.4×10^{-2}
1024_{32}	Pos+M	12.6	0.87	0.97	1.9×10^{-3}	5.2	0.81	1.06	2.6×10^{-3}
1024_{32}	Pos+M+V	10.7	0.87	0.91	1.6×10^{-3}	4.4	0.87	0.94	1.9×10^{-3}
1024	Pos+M+V (XgBoost)	14.8	0.68	⋯	3.19×10^{-3}	5.5	0.74	⋯	2.84×10^{-3}
4096	2ptCF	16.2	0.81	1.04	2.6×10^{-3}	8.7	0.43	1.12	7.1×10^{-3}
4096_{32}	Pos+M+V	7.1	0.94	0.51	5.7×10^{-4}	2.7	0.95	0.52	7.2×10^{-4}
4096_{64}	Pos+M+V	8.7	0.95	0.59	7.7×10^{-4}	3.2	0.93	0.60	1.0×10^{-3}
4096	Pos+M+V (XgBoost)	12.6	0.78	⋯	2.36×10^{-3}	5.4	0.74	⋯	2.8×10^{-3}
8192	2ptCF	16.4	0.78	1.75	2.9×10^{-3}	8.5	0.46	1.43	6.7×10^{-3}
8192_{32}	Pos-only	15.6	0.80	0.81	2.8×10^{-3}	13.4	0.01	1.25	1.4×10^{-2}
8192_{32}	Pos+M	11.2	0.89	1.12	1.6×10^{-3}	5.1	0.83	1.04	2.5×10^{-3}
8192_{32}	Pos+M+V	7.1	0.97	0.65	4.6×10^{-4}	2.6	0.95	0.92	6.7×10^{-4}
8192	Pos+M+V (XgBoost)	11.4	0.83	⋯	1.8×10^{-3}	5.2	0.76	⋯	2.6×10^{-3}

Note. The first column states the configuration used with the notation N_k, where N is the number of halos in the catalog and k is the number of neighbors. The second column shows the features used for the halos, where pos, M, and V stand for positions, masses, and velocity moduli. While the 2ptCF denotes the usage of the two-point correlation function using just the positions of the dark matter halos, XGBoost indicates the cases where the XGBoost model is used. The other columns show the values of the different validation metrics for Ω_m and σ_8.

4. Results

We now present the results we obtained after training the models. Table 1 summarizes the results for the different configurations. In Figure 2, we show the constraints we derive on Ω_m and σ_8 when considering 8192 halos with 32 neighbors when using different features for the dark matter halos. We now describe the main features we observe.

1. *Pos-only*. From Table 1, we can see that when considering point clouds that only contain the positions of the dark matter halos, the models can infer Ω_m but not σ_8, even in the case of 8192 halos. We identify this as one of the main limitations of this model and discuss in the next section (and in the Appendix) that it may be due to the inability of the network to extract information from small scales.
2. *Pos+M*. When the point clouds contain both halo positions and masses, we find that our models outperform the results of the ones trained with only positions, as expected. Furthermore, in this case, our models can also infer the value of σ_8. This is likely because the information may come from the halo mass function rather than clustering. Also, in this case, the larger the number of points, the better the performance of the model.
3. *Pos+M+V*. We find that the models trained on point clouds that include positions, masses, and velocities perform the best among all the ones studied in this work. As expected, the more points we include, the better the model's performance. In the case of 8132_{32}, the model is able to constrain Ω_m and σ_8 with mean relative errors of 7.1% and 2.6%, respectively.

4.1. Comparison with the Two-point Correlation Function

As a benchmark for our analysis, we have quantified how well we can constrain the value of the cosmological parameters if we employ the most standard summary statistics in cosmology: the two-point correlation function (2ptCF). For this, we have computed the two-point correlation function of the point clouds (using only halo positions) for different configurations (1024, 4096, and 8192 points) using `Pylians` (F. Villaescusa-Navarro 2018).

We have then trained MLPs to perform parameter inference using these summary statistics. In this case, we use the same loss function as Equation (12), but the architecture is composed of fully connected layers with a LeakyReLU nonlinear activation function. The numbers of layers and neurons, the learning rate, weight decay, and dropout are hyperparameters that we optimize with `OPTUNA`. The results are displayed in Table 1.

We find that the MLP trained on the two-point correlation function outperforms our point cloud model. For 1024 points, both the MLP and the point cloud model cannot infer σ_8, while for 8192 points, the MLP is able to get some loose constraints while the point cloud is not. We note that this result depends on the minimum scale used in the two-point correlation function. In the Appendix, we show that using the correlation function on scales larger than $10h^{-1}$ Mpc prevents the MLP model from constraining σ_8.

Thus, we conclude that when using halo positions alone, our model underperforms the classical two-point correlation function. This could be due to multiple reasons, from the lack of training data to our model being suboptimal. We will discuss this in more detail in the next subsection.

4.2. Comparison with `XGBoost`

We also benchmark our study with `XGBoost`[11] (T. Chen & C. Guestrin 2016), a simple non-neural random-forest-based algorithm. The aim of this exercise is to highlight how a random-forest-based regression algorithm with Pos+M+V

[11] https://xgboost.readthedocs.io/en/stable/index.html

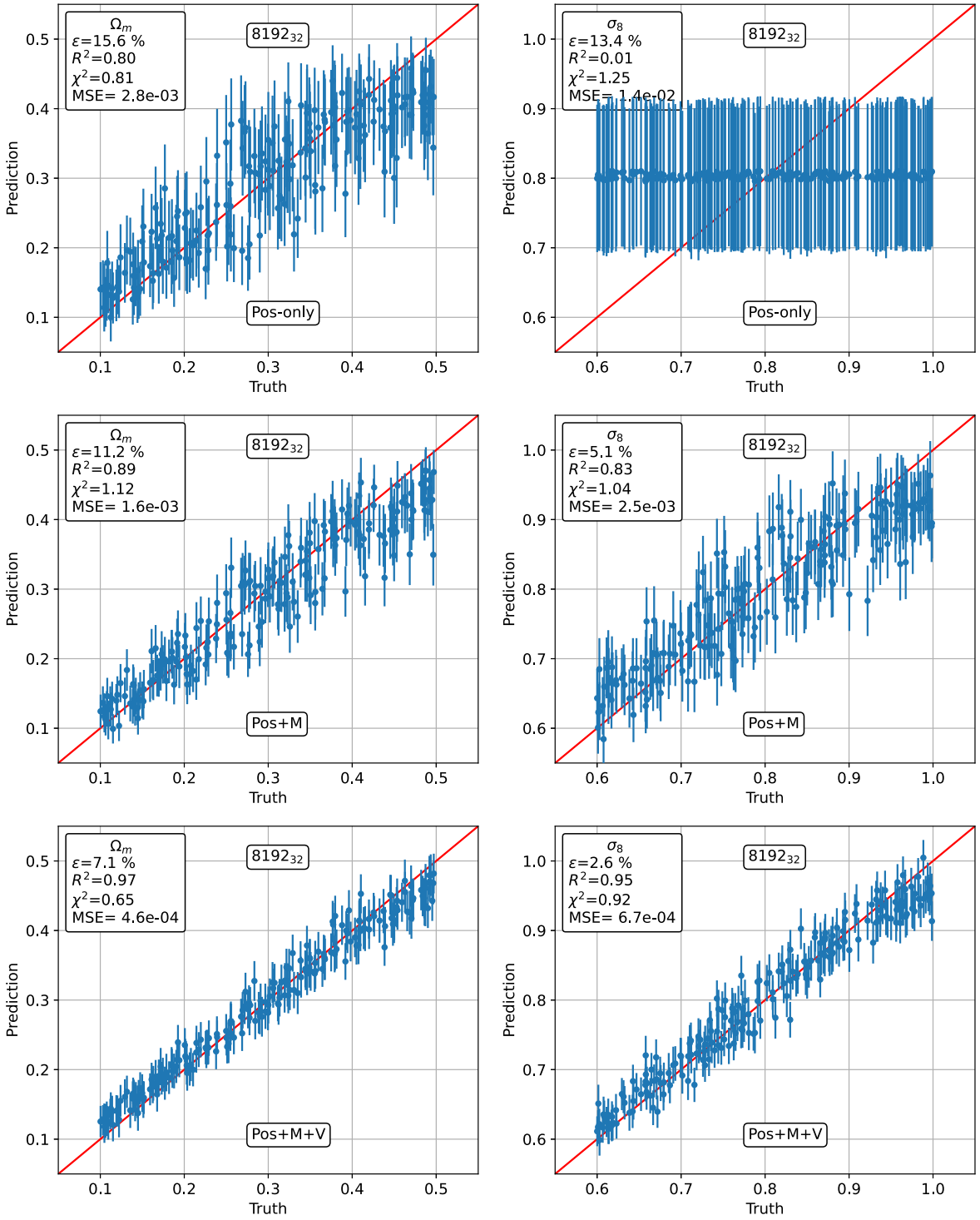

Figure 2. Our model is trained to perform likelihood-free inference on the values of both Ω_m (left column) and σ_8 (right column) from point clouds containing: (1) positions (top row), (2) positions and masses (middle row), and (3) positions, masses, and velocity moduli (bottom row). The notation N_k indicates the number of points considered (N) and the number of neighbors (k). The numbers in the legend indicate the values of different validation metrics.

features performs compared to `PointMLP-elite`. A key distinction between `XGBoost` and `PointMLP-elite` is that the former relies solely on the Pos+M+V features of the selected halos, without incorporating any information about the underlying topology from the k nearest neighbors. We also note that XGBoost can only estimate the mean values of the cosmological parameters and not the standard deviation, therefore we could not provide the χ^2 values in this case.

We have trained the `XGBoost` model with $n_{\rm estimators} = 400$ (size of the forest to be trained), `max_depth` = 10 (maximum depth of a tree), and `tree_method = 'hist'` (tree construction algorithm used in XGBoost) and presented the results in Table 1.

We find that the `XGBoost` model performs significantly worse than the `PointMLP-elite network`. Especially with 8192 halos, the mean squared errors obtained from

Table 2
Summary of the Results Obtained by Varying Different Hyperparameters/Parts of the Network

Fixed Part	Varying Part	Features	Ω_m			σ_8		
			ϵ (%)	R^2	MSE	ϵ (%)	R^2	MSE
Max-pool, FPS	Number of Layers	3 layers	8.3	0.89	2×10^{-3}	6.2	0.75	8.9×10^{-3}
		4 layers	7.1	0.97	4.6×10^{-4}	5.2	0.81	2.6×10^{-3}
		5 layers	6.9	0.95	4.9×10^{-4}	5.9	0.85	2.4×10^{-3}
		6 layers	7.5	0.98	4.7×10^{-4}	5.5	0.81	2.3×10^{-3}
4 layers, FPS	Aggregation	Max-pool	7.1	0.97	4.6×10^{-4}	5.2	0.81	2.6×10^{-3}
		Mean	7.5	0.94	4.8×10^{-4}	5.0	0.79	2.5×10^{-3}
		Sum	7.7	0.95	5.0×10^{-4}	5.4	0.83	2.7×10^{-3}
4 layers, Max-pool	Subsampling	FPS	7.1	0.97	4.6×10^{-4}	5.2	0.81	2.6×10^{-3}
		Random	7.2	0.95	4.2×10^{-3}	5.3	0.83	2.9×10^{-3}

Note. The first column states the parts of the architecture that were kept fixed, while the second column shows the varying parts of the network. The third column mentions the details of the varying features. For all the scenarios, we use 8192 halos with 32 nearest neighbors, i.e., 8192_{32}, with Pos+M+V as the input feature.

`XGBoost` are nearly an order of magnitude higher than those from `PointMLP-elite`. Additionally, the mean relative errors from `XGBoost` are twice as large as those achieved by `PointMLP-elite`. Therefore, it is evident that the `PointMLP-elite` model has gained significant insight by learning the topology around a halo through feature extraction from its nearest neighbors.

4.2.1. Component Ablation Study

In this subsection, we describe in detail the ablation studies demonstrating how the model performs when we change individual components in the `PointMLP-elite` architecture. We summarize the results of the ablation study in Table 2 and describe them as follows.

In the last row of Table 1, we mention the values of different validation matrices for both Ω_m and σ_8. To reiterate, we have obtained that result with (a) configuration 8192_{32}, (b) features Pos +M+V, (c) FPS subsampling, (d) max-pool aggregation, and (e) four layers. For the purpose of the ablation study, we kept that configuration as our benchmark and then varied different individual parts of the `PointMLP-elite` network as follows:

1. *Network depth.* We varied the number of stages from three to six. As shown in Table 2, the accuracy of the model declined with three stages. The accuracy significantly increased with four stages, but then we found that the performance of the model did not improve after four stages; only the time required for execution of the codes increased.
2. *Max-pooling.* We then tried changing the aggregation method from max-pooling to the mean and sum aggregation but found no significant change in the model performance.
3. *Subsampling method.* We have also experimented with the subsampling method. We tried a random subsampling technique rather than the FPS, but the accuracy of the model remains unchanged.

4.3. Comparison with Other Works

Now, we will compare our results with those obtained by other groups using different models. We emphasize that a proper comparison can only be made in a few cases where the data set is the same.

T. L. Makinen et al. (2022) used a GNN along with an information maximizing neural network on halo catalogs from the `Quijote` simulations and reported the mean relative errors of Ω_m and σ_8 to be 7.5% and 0.89% when using the positions and masses of ∼100 halos. Our model for 8192_{32} with Pos+M performs worse for both parameters. Interestingly, they have also compared their GNN result with the 2ptCF for their halo catalogs and found that the 2ptCF outperforms their GNN models (with a fixed number of halos) for the Pos-only scenario.

M. Ho et al. (2024) trained a GNN architecture on catalogs containing only the position of ∼10,000 halos from the `Quijote` simulations to infer cosmological parameters. Although the authors did not report any values of the accuracy metrics, our visual estimate (of Figure 7 in their paper) suggests that they recover Ω_m with ∼20%, which is slightly poorer than our Pos-only study of 8192 halos. We note that they have been able to infer σ_8 with ∼4% accuracy, which our Pos-only study fails to do.

C. Cuesta-Lazaro & S. Mishra-Sharma (2024) used a diffusion-based generative model for generating point clouds that resemble the spatial distribution of dark matter halos from an *N*-body simulation. These authors used the 5000 most massive halos of the latin-hypercube `Quijote` simulations to train their model. Once trained, their model can also be used to infer the likelihood of the data. They quantified the performance of their model in two scenarios: (i) Pos-only and (ii) Pos+M+Vel (Vel here being the 3D velocity). In the case of Pos-only, they achieved mean relative errors of ∼5% and ∼3% on Ω_m and σ_8, respectively. However, the learned likelihood from their model is not well calibrated for Ω_m, so comparison with that cosmological parameter should be taken with a pinch of salt.

P. Villanueva-Domingo & F. Villaescusa-Navarro (2022) use a GNN-based approach to infer the value of cosmological parameters Ω_m and σ_8 based on the galaxy catalogs from the CAMELS simulations (F. Villaescusa-Navarro et al. 2021). Further, they study three scenarios based on the input feature used: (i) only position of the galaxy, (ii) galaxy positions + stellar masses + stellar radii + stellar metallicities, and (iii) galaxy positions + stellar masses + stellar radii + stellar

metallicities + V_{max}. In this case, the authors were able to infer the value of Ω_m with different degrees of confidence. We note that the catalogs, volume, and nature of the simulations are very different from the ones used in this work, so a comparison is not possible.

H. Shao et al. (2022) used a similar GNN-based model to infer Ω_m and σ_8 separately with $\leqslant$5000 halos from halo catalogs obtained from Gadget *N*-body simulations in a periodic volume of $(25h^{-1}\,\mathrm{Mpc})^3$. Two cases from their study that are relevant for comparison with the current works are: (i) Pos-only for inferring Ω_m and σ_8, (ii) Pos+M for inferring σ_8. For (i) Pos-only, they obtained mean relative errors of 10% and 11.8% for Ω_m and σ_8, respectively. In our case for Pos-only and using 8192 halos, we obtained a mean relative error of 15.6% for Ω_m. For Pos+M, they obtained a mean relative error of 5.9% for σ_8, whereas we obtained a mean relative error of 7.1% for σ_8 with 8192 halos. Here, also note that their networks infer only σ_8, unlike ours, which infers both Ω_m and σ_8 at the same time. We would like to remind the readers that they used a box size of $(25h^{-1}\,\mathrm{Mpc})^3$, but in our case the box size is $(1000h^{-1}\,\mathrm{Mpc})^3$, so a proper comparison is not possible and this should be seen as a qualitative statement.

S. Anagnostidis et al. (2022) employ a `PointNeXt` network using halo catalogs from COSMO-GRIDV1 simulations (T. Kacprzak et al. 2023) with (i) Pos-only and (ii) Pos +M as the input features to infer the values of Ω_m and σ_8. Their MSE for Ω_m with 8000 halos using Pos-only is 3.6×10^{-3}, which is slightly poorer than that for our 8192_{32} Pos-only scenario (2.8×10^{-3}). Further, with the same number of halos, their Pos+M model yields an MSE of 1.6×10^{-3}, which is slightly better than the MSE obtained from our Pos+M scenario, 2.5×10^{-3}. Note that, unlike us, their Pos-only case outperforms the 2ptCF result only in the case of σ_8 (as seen from Figure 2(b) of their paper). We note that also, in this case, a proper comparison is not possible given the different setups (lightcone versus periodic boxes) and volumes/resolutions of the catalogs.

5. Model Limitations and Improvements

We now discuss some of the limitations of our model and potential directions to overcome them.

Perhaps the most important limitation of the model studied here is its inability to infer σ_8 from the point clouds when using only the positions of the halos. Other works using different architectures have been shown to provide relatively good constraints on σ_8 using halos from the `Quijote` simulations (see C. Cuesta-Lazaro & S. Mishra-Sharma 2024; M. Ho et al. 2024). This may be because the model cannot properly extract information from small scales, where the constraints on σ_8 most likely arise (see the Appendix). Below, we discuss a few potential changes in architecture that could help improve this behavior.

Another limitation is that standard summary statistics such as the two-point correlation function seem to contain more information than what our model can extract when using point clouds with only the positions of the halos. This may be related to the previous point (i.e., the inability of our model to extract small-scale information). On the other hand, the same behavior has been seen by T. L. Makinen et al. (2021) using the same data set but with more expressive GNN architecture. Thus, perhaps the limitation in this case is the limited training data set (only 1600 point clouds). In the future, it would be interesting to train our models with larger data sets to see whether the constraints on the parameters improve. Another way to overcome this problem would be to concatenate the two-point correlation function measured on the halo catalogs with the latent space before passing it into the inference model (see, e.g., M. Ntampaka et al. 2020).

Another potential limitation is that the input point clouds in the current model must have the same fixed number of points. While this may not be a problem if one has enough halos and performs a cut in number density, there may be applications where a selection function may yield different numbers of halos in different catalogs (e.g., a cut in halo mass). Modifications to our model can enable working with point clouds of different sizes. For instance, in the first stage, one may sample a fixed number of points using the farthest point sampling algorithm rather than just taking half of the points to the input layer.

As a proof-of-concept, we have carried out this study using dark matter halos in real space together with their masses and velocity moduli. To get closer to the data, we will need to work with galaxies in redshift space, or galaxies in real space but with radial peculiar velocities (see, e.g., N. S. M. de Santi et al. 2023). Besides, we need to place the halos/galaxies in a lightcone rather than in periodic cubic boxes. It will be interesting to repeat the work done in this paper with these more realistic setups.

Another potential improvement would be to consider different neighborhood definitions. For instance, in this work, we use the k nearest neighbors of points in each stage, following X. Ma et al. (2022). However, P. Villanueva-Domingo (2022) showed that using all points within a given distance yields better results than using the k closest neighbors when training GNNs. Thus, it would be interesting to try that neighborhood definition and investigate the results.

6. Conclusions

In this paper, we have trained neural networks to perform likelihood-free inferences on the value of the cosmological parameter from halo catalogs, which are represented as point clouds. This task has been carried out before using different approaches such as graph neural networks (see, e.g., T. L. Makinen et al. 2021; S. Anagnostidis et al. 2022; H. Shao et al. 2022; P. Villanueva-Domingo et al. 2022; C. Cuesta-Lazaro & S. Mishra-Sharma 2024; N. S. M. de Santi et al. 2023; M. Ho et al. 2024). The main difference in our work is that we use a recently proposed architecture, pointMLP (X. Ma et al. 2022), that can work directly with point clouds and is able to extract information hierarchically for large data sets. As with GNNs, the advantage of using these models is that they can extract information at the field level without relying on summary statistics. Besides, they do not impose a minimum scale where the field can be sampled, as happens when the point clouds are transformed into regular grids. Furthermore, they can automatically deal with complex geometric patterns, such as selection effects and masks, which are more difficult to treat with standard approaches.

The model takes a point cloud containing N points as input, and reduces that number hierarchically while increasing the dimensionality of their feature vector. Finally, an MLP is used to infer the parameters from the points in the last hierarchical layer. The features of the points in the hierarchy are updated from their neighbors using MLPs and max-pooling

aggregation. This model performs similar operations to GNNs but without requiring an input graph.

We find that our models perform well and can accurately infer the value of the cosmological parameters under different combinations of the input features (e.g., positions only, positions plus masses, etc.). A comparison with other works in the literature is difficult due to intrinsic differences in the training data and methodology. Perhaps the closest analyses are the ones from M. Ho et al. (2024), T. L. Makinen et al. (2022), and C. Cuesta-Lazaro & S. Mishra-Sharma (2024). Our model yields better and worse results for $\Omega_{\rm m}$ depending on setup and parameter. However, all these works perform better when inferring σ_8 than our model.

The model explored in this work presents several advantages with respect to previous methods (e.g., message-passing GNNs). First, its simplicity and lack of sophisticated geometric kernels make the model fast to evaluate. Second, the model can work with large point clouds. In this work, we have explored models that contain up to 8192 points, but this number can easily be made much larger if needed (e.g., decreasing the number of points to sample during the stages).[12] This short computation time for PointMLP is extremely important in the context of future sky surveys. In the upcoming sky surveys, the number of galaxies will be very large, and dealing with a large number of galaxies using GNNs is not possible even with state-of-the-art computers. Therefore, a model such as PointMLP with a computational time less than that of a GNN will be very important for future galaxy surveys. Third, its simplicity makes it very easy to adapt to different tasks. Fourth, this model works with the natural representation of the data (a point cloud), while GNNs need a graph to operate (it is unclear how to construct the optimal graph from a point cloud). We note that deep sets are deep learning architectures designed to work with data that can be represented as sets, but in general, they do not exploit the spatial information (see, e.g., B. Y. Wang et al. 2023, for an example of using deep sets to infer cosmology from void catalogs).

On the other hand, we find that in the simplest case of a point cloud containing only the positions of the halos, the classical two-point correlation function yields better results. While this can be due to multiple reasons, such as lack of training data and a low number of halos (high shot noise), it can also be due to the architecture not being able to exploit the clustering information fully. We note that this result has also been obtained using a GNN architecture by T. L. Makinen et al. (2022), perhaps pointing toward the low-data directions rather than the expressivity of the model. We have identified some areas that could enhance the expressivity of our model and leave this for future work.

Finally, we note that, while in this work we have worked with halo catalogs from *N*-body simulations, the method can be applied to other point cloud data, such as galaxy catalogs from hydrodynamic simulations or semianalytic models (L. A. Perez et al. 2022; F. Villaescusa-Navarro et al. 2022b) or void catalogs (C. D. Kreisch et al. 2022). The main reason for choosing halo catalogs rather than galaxy catalogs is the unavailability of training data sets with volumes comparable to the upcoming sky surveys. The state-of-the-art hydrodynamical data set available to us is CAMELS, which covers a volume of $(25h^{-1}\,{\rm Mpc})^3$, much smaller than the volume of future sky surveys. As this work is a precursor to the data from those sky surveys, we chose Quijote simulation with a box size of $1h^{-1}$ Gpc, comparable to the upcoming sky surveys.

[12] We would like to emphasize that we dealt with 80,000 halos (when the periodic boundary condition is ignored) in this work, which is not even feasible for GNNs. Thus, for lightcone data sets, our model can easily deal with hundreds of thousands of points if several GPUs are used.

Acknowledgments

We thank Teresa Huang, Natali de Santi, and Matthew Ho for their comments and discussion on this work. The calculations performed in this work have been carried out in both the IUCAA cluster and Rusty cluster at the Flatiron Institute. The work of F.V.N. is supported by the Simons Foundation.

Appendix
Two-point Correlation Function

In this Appendix, we study the dependence on the constraints we obtain using the two-point correlation function as a function of the minimum scale used. We consider two different scenarios: (1) we use the correlation function on all available scales, and (2) we only use the correlation function on scales larger than $10h^{-1}$ Mpc. For each of these cases, we train MLPs from scratch and perform hyperparameter tuning as described in Section 3.

We show the results of this analysis in Figure 3. We can reach two important conclusions. First, the larger the number of points, the more accurate the network becomes. This is clearly expected, given the fact that the shot noise will be lower. Second, we find that if we do not use scales smaller than $10h^{-1}$ Mpc, the network cannot infer the value of σ_8. This is perhaps expected as σ_8 measures the variance of the linear perturbations in the underlying matter field on scales of $8h^{-1}$ Mpc.

This behavior may shed some light on why our point cloud model does not yield good results for σ_8. Perhaps the above finding points toward the point cloud model not being able to exploit the small-scale information optimally. We speculate that this may be due to the hierarchical nature of the model, the pooling layer, or the subsampling method. We note that other works using the same data managed to get good constraints on σ_8 using a similar number of halos to the ones considered in this paper (see C. Cuesta-Lazaro & S. Mishra-Sharma 2024; M. Ho et al. 2024).

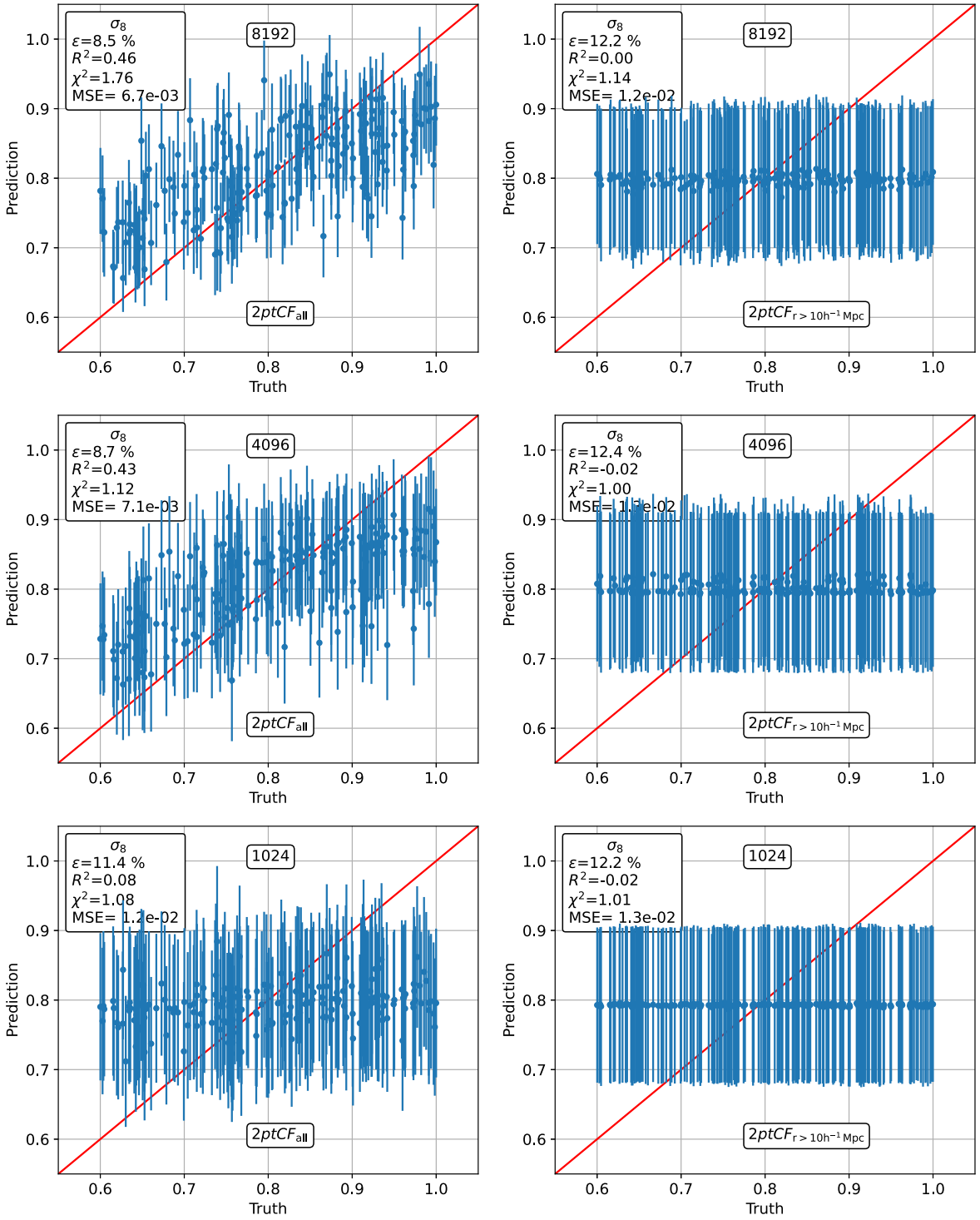

Figure 3. We have measured the two-point correlation functions of point clouds that only contain the positions of the dark matter halos. We have then trained MLPs to infer the values of Ω_m and σ_8 from them. This plot shows the results of this analysis when using 8192 (top), 4096 (middle), and 1024 (bottom) halos employing all scales from the correlation function (left column) and only scales larger than $10h^{-1}$ Mpc (right column). As can be seen, to infer σ_8 we need to use scales larger than $10h^{-1}$ Mpc.

ORCID iDs

Atrideb Chatterjee https://orcid.org/0000-0002-0159-8708
Francisco Villaescusa-Navarro https://orcid.org/0000-0002-4816-0455

References

Abazajian, K., Abdulghafour, A., Addison, G. E., et al. 2022, arXiv:2203.08024
Ajani, V., Peel, A., Pettorino, V., et al. 2020, PhRvD, 102, 103531
Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. 2019, arXiv:1907.10902
Anagnostidis, S., Thomsen, A., Kacprzak, T., et al. 2022, arXiv:2211.12346
Banerjee, A., & Abel, T. 2021, MNRAS, 500, 5479
Bayer, A. E., Villaescusa-Navarro, F., Massara, E., et al. 2021, ApJ, 919, 24
Benitez, N., Dupke, R., Moles, M., et al. 2014, arXiv:1403.5237
Chen, T., & Guestrin, C. 2016, arXiv:1603.02754
Cuesta-Lazaro, C., & Mishra-Sharma, S. 2024, PhRvD, 109, 123531
de Santi, N. S. M., Shao, H., Villaescusa-Navarro, F., et al. 2023, ApJ, 952, 69
Delle Rose, V., Kozachinskiy, A., Rojas, C., Petrache, M., & Barceló, P. 2023, arXiv:2303.12853

DESI Collaboration, Aghamousa, A., Aguilar, J., et al. 2016, arXiv:1611.00036
Eickenberg, M., Allys, E., Moradinezhad Dizgah, A., et al. 2022, arXiv:2204.07646
Euclid Collaboration, Castro, T., Fumagalli, A., Angulo, R. E., et al. 2023, A&A, 671, A100
Fan, S., Dong, Q., Zhu, F., et al. 2021, in 2021 IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR), 14499
Fluri, J., Kacprzak, T., Lucchi, A., et al. 2019, PhRvD, 100, 063514
Friedrich, O., Uhlemann, C., Villaescusa-Navarro, F., et al. 2020, MNRAS, 498, 464
Gillet, N., Mesinger, A., Greig, B., Liu, A., & Ucci, G. 2019, MNRAS, 484, 282
Giri, U., & Smith, K. M. 2020, JCAP, 2022, 028
Gualdi, D., Gil-Marin, H., & Verde, L. 2021, JCAP, 2021, 008
Gualdi, D., Novell, S., Gil-Marìn, H., & Verde, L. 2021, JCAP, 2021, 015
Guo, M.-H., Cai, J.-X., Liu, Z.-N., et al. 2020, arXiv:2012.09688
Hahn, C., Villaescusa-Navarro, F., Castorina, E., & Scoccimarro, R. 2020, JCAP, 2020, 040
Harnois-Déraps, J., Martinet, N., Castro, T., et al. 2021, MNRAS, 506, 1623
Hassan, S., Villaescusa-Navarro, F., Wandelt, B., et al. 2022, ApJ, 937, 83
Ho, M., Bartlett, D. J., Chartier, N., et al. 2024, OJAp, 7, 54
Hordan, S., Amir, T., & Dym, N. 2024, arXiv:2402.02484
Hordan, S., Amir, T., Gortler, S. J., & Dym, N. 2023, arXiv:2301.13821
Hortua, H. J. 2021, arXiv:2112.11865
Ivezić, Ž., Kahn, S. M., Tyson, J. A., et al. 2019, ApJ, 873, 111
Jeffrey, N., Alsing, J., & Lanusse, F. 2021, MNRAS, 501, 954
Jeffrey, N., & Wandelt, B. D. 2020, arXiv:2011.05991
Kacprzak, T., Fluri, J., Schneider, A., Refregier, A., & Stadel, J. 2023, JCAP, 2023, 050
Kreisch, C. D., Pisani, A., Villaescusa-Navarro, F., et al. 2022, ApJ, 935, 100
Lam, C. Y., Abrams, N., Andrews, J., et al. 2023, arXiv:2306.12514
Laureijs, R., Amiaux, J., Arduini, S., et al. 2011, arXiv:1110.3193
Li, G., Müller, M., Qian, G., et al. 2019, arXiv:1910.06849
Liu, W., Jiang, A., & Fang, W. 2022, JCAP, 2022, 045
Loshchilov, I., & Hutter, F. 2016, arXiv:1608.03983
Ma, X., Qin, C., You, H., Ran, H., & Fu, Y. 2022, arXiv:2202.07123
Makinen, T. L., Charnock, T., Alsing, J., & Wandelt, B. D. 2021, JCAP, 2021, 049
Makinen, T. L., Charnock, T., Lemos, P., et al. 2022, OJAp, 5, 18
Marques, G. A., Liu, J., Zorrilla Matilla, J. M., et al. 2019, JCAP, 2019, 019
Maturana, D., & Scherer, S. 2015, in 2015 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS), ed. Christian Laugier (Pisctaway, NJ: IEEE), 922
Naidoo, K., Massara, E., & Lahav, O. 2021, MNRAS, 513, 3596
Ntampaka, M., Eisenstein, D. J., Yuan, S., & Garrison, L. H. 2020, ApJ, 889, 151
Perez, L. A., Genel, S., Villaescusa-Navarro, F., et al. 2022, ApJ, 954, 11
Qi, C. R., Su, H., Mo, K., & Guibas, L. J. 2016, arXiv:1612.00593
Qi, C. R., Yi, L., Su, H., & Guibas, L. J. 2017, arXiv:1706.02413
Racca, G. D., Laureijs, R., Stagnaro, L., et al. 2016, Proc. SPIE, 9904, 99040O
Ravanbakhsh, S., Oliva, J., Fromenteau, S., et al. 2017, arXiv:1711.02033
Roncoli, A., Ćiprijanović, A., Voetberg, M., Villaescusa-Navarro, F., & Nord, B. 2023, arXiv:2311.01588
Ruder, S. 2016, arXiv:1609.04747
Samushia, L., Slepian, Z., & Villaescusa-Navarro, F. 2021, MNRAS, 505, 628
Shao, H., Villaescusa-Navarro, F., Villanueva-Domingo, P., et al. 2022, arXiv:2209.06843
Shi, S., Wang, X., & Li, H. 2018, arXiv:1812.04244
Uhlemann, C., Friedrich, O., Villaescusa-Navarro, F., Banerjee, A., & Codis, S. 2020, MNRAS, 495, 4006
Valogiannis, G., & Dvorkin, C. 2022, PhRvD, 105, 103534
Villaescusa-Navarro, F. 2018, Pylians: Python Libraries for the Analysis of Numerical Simulations, Astrophysics Source Code Library, ascl:1811.008
Villaescusa-Navarro, F., Anglés-Alcázar, D., Genel, S., et al. 2021, ApJ, 915, 71
Villaescusa-Navarro, F., Ding, J., Genel, S., et al. 2022c, ApJ, 929, 132
Villaescusa-Navarro, F., Genel, S., Anglés-Alcázar, D., et al. 2022a, ApJS, 259, 61
Villaescusa-Navarro, F., Genel, S., Anglés-Alcázar, D., et al. 2022b, ApJS, 265, 103534
Villaescusa-Navarro, F., Hahn, C., Massara, E., et al. 2020, ApJS, 250, 2
Villanueva-Domingo, P. 2022, PabloVD/CosmoGraphNet, v1.0, Zenodo, doi:10.5281/zenodo.6485804
Villanueva-Domingo, P., & Villaescusa-Navarro, F. 2022, ApJ, 937, 115
Villanueva-Domingo, P., Villaescusa-Navarro, F., Anglés-Alcázar, D., et al. 2022, ApJ, 935, 30
Villanueva-Domingo, P., Villaescusa-Navarro, F., Genel, S., et al. 2021, PhRvD, 107, 103003
Wang, B. Y., Pisani, A., Villaescusa-Navarro, F., & Wandelt, B. D. 2023, ApJ, 955, 131
Xu, C., Wu, B., Wang, Z., et al. 2020, arXiv:2004.01803