MNRAS **000**, 1–13 (2020) Preprint 9 September 2020 Compiled using MNRAS LATEX style file v3.0

Simulation-based inference of dynamical galaxy cluster masses with 3D convolutional neural networks

Doogesh Kodi Ramanah,[1]* Radosław Wojtak,[1]† Nikki Arendse[1]‡
[1] DARK, Niels Bohr Institute, University of Copenhagen, Jagtvej 128, 2200 Copenhagen, Denmark

Accepted XXX. Received YYY; in original form ZZZ

ABSTRACT
We present a simulation-based inference framework using a convolutional neural network to infer the dynamical mass of galaxy clusters from their observed 3D projected phase-space distribution, which consists of the projected galaxy positions in the sky and their line-of-sight velocities. By formulating the mass estimation problem within this simulation-based inference framework, we are able to quantify the uncertainties on the inferred masses in a straightforward and unbiased way. We generate a realistic mock catalogue emulating the Sloan Digital Sky Survey (SDSS) Legacy spectroscopic observations and explicitly illustrate the challenges posed by interloper (non-member) galaxies for cluster mass estimation from actual observations. Our approach constitutes the first optimal exploitation of the information content of the full projected phase-space distribution for the inference of dynamical cluster masses. We also present, for the first time, the application of a machine learning-based inference machinery to obtain dynamical masses of around 900 galaxy clusters found in the SDSS Legacy Survey and demonstrate that the inferred masses reproduce the cluster mass function, as predicted by Planck ΛCDM cosmology, down to $10^{14.1}h^{-1}\mathrm{M}_{\odot}$ which is nearly a mass completeness limit of the selected cluster sample.

Key words: methods: numerical – methods: statistical – galaxies: clusters: general

1 INTRODUCTION

Galaxy clusters are formed by the collapse of high density regions in the early Universe, and they are important to study the formation and evolution of large-scale cosmic structures. The cluster abundance as a function of mass and its evolution are sensitive to the amplitude of density perturbations and to the properties of dark matter and dark energy. Galaxy clusters can therefore provide competitive cosmological constraints that are complementary to other cosmological probes. As future surveys, such as the Dark Energy Spectroscopic Instrument (DESI, DESI Collaboration et al. 2016), the Vera C. Rubin Observatory (Ivezic et al. 2008), Euclid (Racca et al. 2016) and eROSITA (Merloni et al. 2012), will provide unprecedented volumes of data extending to high redshifts, the efficiency and automation of cluster mass estimation techniques will become crucial. With the ever increasing scale of state-of-the-art cosmological simulations (e.g. Villaescusa-Navarro et al. 2020; Ishiyama et al. 2020) providing considerable volumes of training data, along with the limitations of traditional techniques, the use of machine learning (ML) algorithms to infer cluster masses has become an increasingly attractive and viable option (e.g. Sutherland et al. 2012; Ntampaka et al. 2015, 2016; Armitage et al. 2019; Calderon & Berlind 2019; Ho et al. 2019, 2020; Kodi Ramanah et al. 2020a; Yan et al. 2020). These models are typically trained on a large simulated data set, such that the algorithm learns the connection between the observables and cluster masses. Once optimized, they can subsequently be used to predict masses for unseen data, provided that the simulations used for training are sufficiently realistic.

For observations probing galaxy kinematics in galaxy clusters, ML methods offer a promising alternative to traditional methods of cluster mass estimation which are usually based on scaling relations and are limited by several assumptions, primarily involving dynamical equilibrium and spherical symmetry. Recently, convolutional neural networks (CNNs), by virtue of their sensitivity to visual features, have been applied by Ho et al. (2019) to obtain accurate dynamical mass estimates of galaxy clusters in spectroscopic surveys. The network inputs are images generated by a kernel density estimator from the 2D projected phase-space distributions defined by the cluster-centric projected distance and line-of-sight velocities of galaxies observed in the fields of clusters. The challenge with ML methods is often to not

* ramanah@nbi.ku.dk
† radek.wojtak@nbi.ku.dk
‡ nikki.arendse@nbi.ku.dk

only produce single point estimates, but also a reliable estimate of the associated uncertainties. The most recent attempts to approach the problem of uncertainty estimation used normalizing flows (Kodi Ramanah et al. 2020a) to infer the conditional probability distribution of the dynamical cluster masses and approximate Bayesian inference to assign prior distributions to the neural network weights (Ho et al. 2020).

The primary challenge intrinsic to galaxy cluster mass estimation is posed by interlopers. These are galaxies that are not gravitationally bound to the cluster, but that are located along the line of sight and have similar line-of-sight velocities to the cluster. Distinguishing interlopers from member galaxies is a problematic task, because redshift surveys can only provide information about the positions and velocities of objects along the line of sight, and not perpendicular to it. Finding an effective way of reducing contamination from interlopers, with the limited information available from surveys, is essential to improve the accuracy of galaxy cluster mass estimates.

In this work, we propose to work at the level of 3D projected phase-space distribution, characterized by the sky projected galaxy positions and their line-of-sight velocities, instead of the standard 2D phase-space, to alleviate the interloper contamination and improve the precision of cluster mass estimation, as motivated by the following arguments. Cluster members are distributed more symmetrically around the cluster centre, while interlopers can clump in any place. Moreover, 2D phase-space density is averaged over the position angle, such that the information on any axially asymmetric localization of interlopers is lost, rendering it more difficult for the algorithm to differentiate between interlopers and cluster members. In contrast, 3D phase-space density retrieves the information encoded in the position angle and is, therefore, expected to provide a better separation between cluster members and interlopers. To optimize the information from the 3D dynamical phase-space distribution, we make use of 3D convolutional kernels, naturally designed to extract spatial features, in neural networks.

We opt for a simulation-based inference approach to quantify the uncertainties on the neural network predictions. Simulation-based inference (e.g. Cranmer et al. 2019, and references therein), often referred to as *likelihood-free inference*, encompasses a class of statistical inference methods where simulations are used to estimate the posterior distributions of the parameters of interest conditional on data, without any prior knowledge or assumption of the likelihood distribution. Simulation-based inference has emerged as a viable alternative to perform Bayesian inference under complex generative physical models using only simulations. This framework allows all physical effects encoded in forward simulations to be properly accounted for in the inference pipeline, without having recourse to inadequate or misguided likelihood assumptions, thereby preventing biased inferences and/or misstated uncertainties. As such, simulation-based inference, and variants thereof, have recently garnered significant interest for cosmological data analysis (e.g. Akeret et al. 2015; Lintusaari et al. 2017; Jennings & Madigan 2017; Leclercq 2018; Charnock et al. 2018; Alsing et al. 2018, 2019; Alsing & Wandelt 2019; Wang et al. 2020).

In essence, we present a simulation-based inference framework for the estimation of the dynamical mass of galaxy clusters with 3D convolutional neural networks. The approach presented here is complementary to our previous neural flow (NF) mass estimator (Kodi Ramanah et al. 2020a, hereafter NF2020) in various aspects. This is primarily a conceptually different framework of uncertainty estimation using neural networks. The simulation-based inference machinery, as presented here, allows the inference of the approximate posterior distribution of cluster masses given their 3D projected phase-space distribution, using an ensemble of simulated clusters and a neural network designed to extract summary statistics. In contrast, the NF mass estimator is a sophisticated neural density estimator, where the cluster mass inference problem is formulated within a conditional density estimation framework. The two methods also differ in network architecture and dimensionality of their respective inputs. The approach presented here employs 3D convolutional kernels to fully exploit the information encoded in the 3D phase-space distribution of galaxy clusters, while the NF mass estimator relies on fully connected layers, i.e. multilayer perceptrons, and works at the level of the compressed 2D phase-space dynamics.

The remainder of this manuscript is organized as follows. Section 2 provides an overview of the 3D dynamical phase-space distribution in terms of the key observables used for training the neural network. We also outline the mock generation procedure for cluster catalogues emulating the features of the actual SDSS data set and the preprocessing steps involved in the preparation of the training and test sets. We then describe the simulation-based inference approach utilized in this work, followed by a brief introduction to convolutional neural networks and a description of the neural network architecture and training procedure in Section 3. We subsequently validate and demonstrate the performance of the optimized model on the test cluster catalogue in Section 4 and follow up by inferring cluster masses from the actual SDSS catalogue in Section 5. Finally, we provide a summary of the main aspects and findings of our work in Section 6, and highlight potential future investigations with cosmological applications.

2 DYNAMICAL PHASE-SPACE DISTRIBUTION

We outline the general problem of cluster mass estimation by first introducing the dynamical phase-space distribution. We then describe the generation of the mock SDSS catalogue which will be used to train and evaluate the performance of the neural network in future sections.

2.1 Galaxy cluster observables

The definition of dynamical cluster mass adopted throughout this work is M_{200c}, corresponding to the mass contained in a sphere with mean density equal to 200 times the critical density of the Universe at the cluster's redshift. We obtain an estimate of the mass by employing the full projected phase-space distribution of galaxy clusters. This consists of the positions of each member galaxy projected onto the (x, y) plane

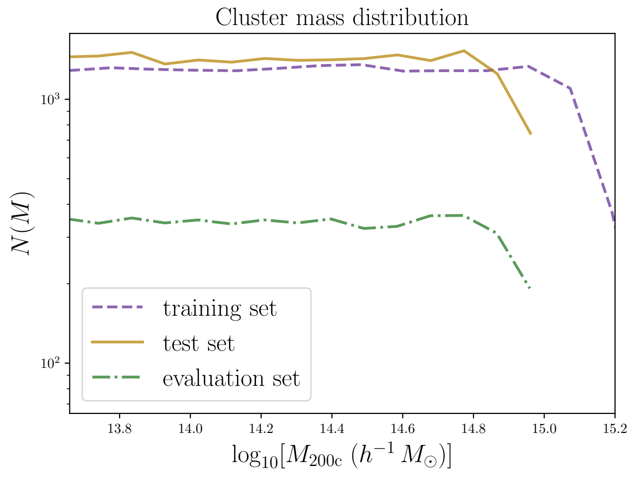


Figure 1. Cluster mass distribution, i.e. variation of number of clusters with logarithmic mass for the training, test and evaluation sets extracted from the mock SDSS catalogue. To ensure we do not induce any cosmological information or selection bias while training the neural network, we upsample the relatively scarce high-mass clusters using independent lines of sight, thereby resulting in an approximately flat mass distribution for the training set.

of the sky, denoted as $(x_{\rm proj}, y_{\rm proj})$, as well as their separate line-of-sight velocities, $v_{\rm los}$, as provided by redshift surveys. In this work, instead of computing the projected radial distance from the cluster centre as $R_{\rm proj} = (x^2_{\rm proj} + y^2_{\rm proj})^{1/2}$ as is typically done, we exploit the information from $x_{\rm proj}$ and $y_{\rm proj}$ separately. This should make our model more sensitive to interlopers, which are often located asymmetrically around the cluster centre.

2.2 Mock cluster catalogues

We generate mock observations of galaxy clusters using publicly available galaxy catalogues derived from the MultiDark simulations (Klypin et al. 2016).[1] Among the three different semi-analytic galaxy formation models applied to the simulation (Knebe et al. 2018), we opted for Semi-Analytic Galaxies (SAG), which includes the most complete implementation of modelling orphan galaxies and, therefore, produces the most realistic distribution of galaxies in the cluster cores (Cora 2006; Cora et al. 2018). For more details regarding implementations of the star formation and feedback processes in SAG as well as a comparison to the remaining two semi-analytic models, i.e. GALACTICUS (Benson 2012) and the Semi-Analytic Galaxy Evolution (SAGE) model (Croton et al. 2006), we refer the interested reader to Knebe et al. (2018). The galaxy catalogues from SAG contain the positions and absolute magnitudes in the SDSS filters at all snapshots of the simulation. The background dark matter simulation (MDPL2) was run for the Planck ΛCDM cosmological model (Planck Collaboration et al. 2014). The simulation box has a size of $1\,h^{-1}$ Gpc and a mass resolution of $1.51 \times 10^9\ h^{-1}\ \mathrm{M}_{\odot}$.

[1] http://skiesanduniverses.org

We select galaxy clusters as massive dark matter halos found in the halo catalogues produced by the ROCKSTAR halo finder (Behroozi et al. 2013). For every cluster, we construct its observable projected phase-space diagram by drawing a line of sight and computing the corresponding projections of the galaxy positions and velocities onto the plane of the sky and the line of sight, respectively. All phase-space coordinates are calculated relative to the central galaxy assigned to the main cluster halo and the observed velocities include the Hubble flow with respect to the cluster centre. The final projected phase-space diagrams are generated by applying the following cuts: ±2200 km s^{-1} in line-of-sight velocities $v_{\rm los}$ and $\pm 1.6\,h^{-1}$Mpc in proper distances $x_{\rm proj}$ and $y_{\rm proj}$.

Aiming at generating mock data which resemble the main spectroscopic galaxy sample of the SDSS Legacy Survey (Strauss et al. 2002), we adopt a flux limit of 18.0 magnitude in r-band. The flux limit is 0.2 magnitude lower than the actual SDSS limiting magnitude in order to compensate the slightly lower counts of galaxies in simulated clusters than in the SDSS ones (see Knebe et al. 2018). The apparent magnitudes of all galaxies in the field of each galaxy cluster are computed by assigning each simulated cluster an observer located at comoving distance randomly drawn from a uniform distribution within a 3D ball. The maximum comoving distance is $250\,h^{-1}$ Mpc, for which galaxy cluster detection in the SDSS main galaxy sample is complete down to a cluster mass of $10^{14.0}h^{-1}\mathrm{M}_{\odot}$ (Abdullah et al. 2020b).

Keeping in mind possible future applications of our dynamical mass estimator for cosmological inference with the cluster abundance, it is instructive to generate a galaxy cluster sample for which the distribution of cluster masses is independent of cosmological model though the mass function. An optimal solution is to consider a set of clusters with a flat distribution in log mass with a possibly wide range of dynamical cluster masses. Aiming at generating a sample with $\sim 10^4$ galaxy clusters, we downsample the actual mass function below halo mass $M_{200c} \approx 10^{14.3}h^{-1}M_{\odot}$ and generate up to 25 projections per cluster at higher masses. In order to minimize correlations between projected phase-space diagrams derived from the same cluster, we use a set of directions (up to 25 lines of sight) found by maximizing angular separations between every two closest sight lines. The adopted maximum number of sight lines per cluster is not sufficient to keep a flat distribution of log cluster masses (cf. Fig. 1). However, further increase of upsampling would introduce strong correlations between phase-space diagrams generated from the same galaxy cluster. The final sample contains 4.3×10^4 galaxy clusters with a minimum halo mass of $10^{13.7}h^{-1}\,\mathrm{M}_{\odot}$.

The overdensity threshold used in the halo mass definition depends on redshift. This leads to a well-known non-physical evolution of halo masses which reflects merely the redshift dependence of the critical density (Diemer et al. 2013). Since phase-space diagrams do not provide any information on cluster redshifts required to adjust the overdensity threshold, mass estimates from neural networks may be consequently affected by an additional noise. For a wide redshift range, the noise may be sufficiently large so that it would be necessary to supplement each phase-space diagram with the information on cluster redshift setting the corresponding overdensity threshold. However, for our mock data the expected uncertainty due to the lack of information on cluster redshifts amounts to only 0.006 dex, which is significantly lower than the level of precision obtained in our work and similar studies, i.e. ∼ 0.1 dex.

We extract the training set, with a flat mass distribution, containing around seventeen thousand clusters by randomly drawing from the mock catalogue. The corresponding validation set, used for early stopping when optimizing the neural network, is designated as 10% of the training set, such that it contains ∼ 1700 clusters and only ∼ 15500 clusters are utilized during training. The test set consisting of twenty thousand clusters is obtained by randomly sampling from the remaining clusters in the mock catalogue. The remaining ∼ 5000 clusters in the catalogue then constitute an evaluation set. The test set is used in the simulation-based inference framework (cf. Section 3.1), whilst the purpose of the evaluation set is to assess the performance of the network (cf. Section 4.2). The mass distributions of the non-overlapping training, test and evaluation sets are depicted in Fig. 1.

2.3 Kernel density estimator

Before the observables ($v_{\rm los}$, $x_{\rm proj}$, $y_{\rm proj}$) are provided as input to the ML model, they are first preprocessed with a kernel density estimator (KDE) to create a smooth PDF mapping in the 3D phase-space distribution. This is done in order to obtain similar-sized arrays as inputs for all clusters, which contain different numbers of member galaxies, as well as to create a visual input (image) for the convolutional neural network. An in-depth review of kernel density estimation is provided in Diggle & Gratton (1984); Wand & Jones (1994); Sheather (2004).

Let our complete set of n observables $\{\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_n\}$, in which each variable $\boldsymbol{X}_i$ is given by ($v_{\rm los}$, $x_{\rm proj}$, $y_{\rm proj}$), be drawn from an unknown distribution with density f. The density evaluated at a point $\boldsymbol{x}$ = ($v_{\rm los}$, $x_{\rm proj}$, $y_{\rm proj}$) can be approximated as

$$\hat{f}(\boldsymbol{x}) = \frac{1}{n|\mathbf{H}|^{1/2}} \sum_{i=1}^{n} K\big[\mathbf{H}^{-1/2}(\boldsymbol{x} - \boldsymbol{X}_i)\big], \tag{1}$$

where K is the kernel function and $\mathbf{H}$ is a 3×3 bandwidth matrix. The KDE sums up the density contributions from the collection of data points $\{\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_n\}$ at the evaluation point $\boldsymbol{x}$. Data points close to $\boldsymbol{x}$ contribute significantly to the total density, while data points further away from $\boldsymbol{x}$ have only a relatively small contribution. The shape of the density contributions is determined by the kernel function, and their size and orientation are dictated by the bandwidth matrix. In this work, we use a 3-dimensional Gaussian kernel given by

$$K(\boldsymbol{u}) = (2\pi)^{-3/2}\, |\mathbf{H}|^{-1/2}\, \exp\left(-\tfrac{1}{2}\, \boldsymbol{u}^\intercal\, \mathbf{H}^{-1}\, \boldsymbol{u}\right), \tag{2}$$

with $\boldsymbol{u} = \boldsymbol{x} - \boldsymbol{X}_i$. For the bandwidth matrix, a scaling factor h_0 is multiplied with the covariance matrix of the data, $\mathbf{H} = h_0 \boldsymbol{\Sigma}$. The scaling factor should be sufficiently small to encapsulate even the more subtle features of the data and small-scale signal expected for low-mass clusters, but large enough that the ML model is robust to changes in galaxy number count, and can easily interpolate between the data sets of discrete and quite scarcely distributed points. Optimizing the bandwidth scaling factor with respect to the performance of the mass estimator, we find $h_0 = 0.175$, which is close to the value proposed by Ho et al. (2019). This is the scaling factor that we adopt in our study.

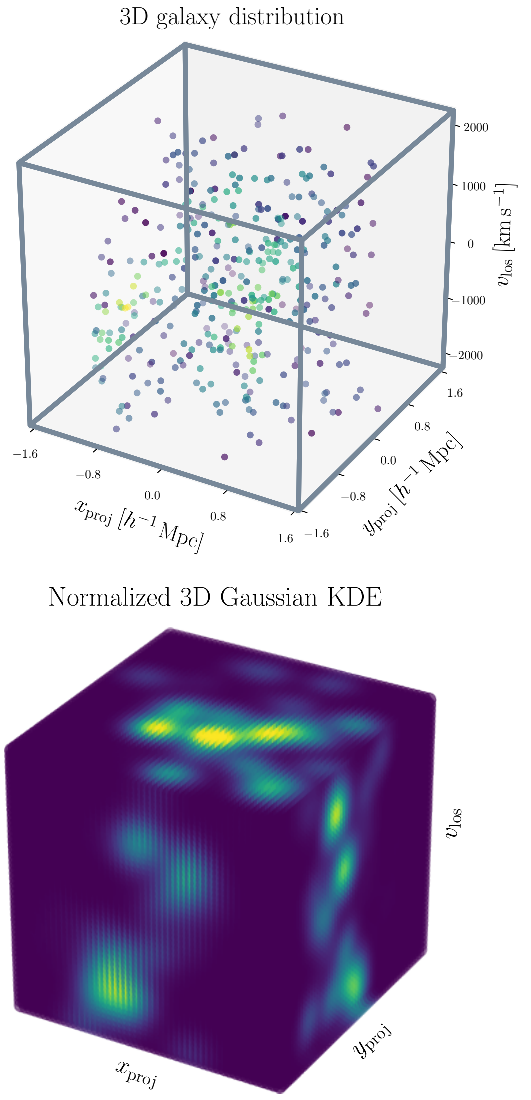


Figure 2. Simulated 3D phase-space distribution of galaxies observed in an example cluster of galaxies, represented via its 3D galaxy distribution (*top panel*) and 3D Gaussian KDE (*bottom panel*), consisting of projected galaxy positions in the sky, $x_{\rm proj}$ and $y_{\rm proj}$, and line-of-sight velocities, $v_{\rm los}$, in dynamical phase space. The KDE representation serves as inputs to our 3D convolutional neural network.

An example of a 3D KDE representation that serves as input to the neural network for one particular cluster is illustrated in Fig. 2. For all clusters, the extents of the observables are as follows: $v_{\rm los} \in [-2200, 2200]\,{\rm km\,s^{-1}}$, $x_{\rm proj} \in [-1.6, 1.6]\,h^{-1}{\rm Mpc}$ and $y_{\rm proj} \in [-1.6, 1.6]\,h^{-1}{\rm Mpc}$, with 50 voxels along each axis, resulting in 3D slices of dimension 50^3.

3 SIMULATION-BASED INFERENCE WITH NEURAL NETWORKS

In this section, we present the rationale underlying simulation-based inference and convolutional neural networks, and outline how a standard neural network may be employed within the simulation-based inference framework to yield unbiased uncertainties. We also describe the implementation of our 3D convolutional neural network in terms of the network architecture and training routine.

3.1 Simulation-based inference

Simulation-based inference (hereafter SBI) encompasses a class of statistical inference techniques employing a simulator, which inherently defines a statistical model, capable of generating high-fidelity simulations for comparison with actual observations (see, for e.g., Cranmer et al. 2019, for an in-depth review of recent developments). However, the probability density for a given observation, i.e. the likelihood, which constitutes a crucial component of any statistical inference framework, is generally intractable. SBI techniques, as relevant to this work, therefore, typically involve an estimation of the likelihood or posterior via informative summary statistics using classical density estimators (Diggle & Gratton 1984) or neural density estimators (e.g. Jimenez Rezende & Mohamed 2015; Germain et al. 2015; Uria et al. 2016; Kingma et al. 2016; Papamakarios & Murray 2016; Papamakarios et al. 2017, 2018; Huang et al. 2018) from recent advances in ML. In essence, these density estimators are used to approximate the distribution of summary statistics of the samples generated from the simulator.

The SBI framework adopted in this work is inspired by the approach presented in Charnock et al. (2018), where they demonstrate that parameter inference is feasible via SBI using summary statistics provided by a neural network. They developed an information maximizing neural network to produce optimal summary statistics, but the approach presented therein may be employed with any neural network predicted summaries. This is because a neural network, by design, performs some form of data compression (or dimensionality reduction) to extract meaningful features from a given input data set to yield informative summaries of the data. Formally, a neural network, $\mathbb{NN}(\boldsymbol{\theta}, \boldsymbol{\gamma}) : \boldsymbol{d} \to \tilde{\boldsymbol{\tau}}$, may be described as a trainable and flexible approximation of a model, $\mathcal{M} : \boldsymbol{d} \to \boldsymbol{\tau}$. The neural network maps some input data $\boldsymbol{d}$ to a prediction or estimate $\tilde{\boldsymbol{\tau}}$ of the desired label or target $\boldsymbol{\tau}$ associated with the data. It is parameterized by a set of trainable weights $\boldsymbol{\theta}$ and a set of hyperparameters $\boldsymbol{\gamma}$, which encompasses the choice of network architecture, initialization of the weights, type of activation and loss functions.

During training, the network weights $\boldsymbol{\theta}$ are optimized via stochastic gradient descent to minimize a particular cost or loss function given a training data set. The loss function is equivalent to the negative logarithm of the likelihood, $-\ln \mathcal{L}(\tilde{\boldsymbol{\tau}}|\boldsymbol{d}, \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\gamma}})$, for a given set of network weights and hyperparameters, $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$ and $\boldsymbol{\gamma} = \hat{\boldsymbol{\gamma}}$, respectively. As such, the loss function provides a measure of how close the network prediction $\tilde{\boldsymbol{\tau}}$ is to the desired target $\boldsymbol{\tau}$. Training the neural network entails finding the maximum likelihood estimates $\hat{\boldsymbol{\theta}}_{\mathrm{MLE}}$ of the network weights with a given training set of data-target pairs $\{\boldsymbol{d}, \boldsymbol{\tau}\}$ at fixed hyperparameters $\hat{\boldsymbol{\gamma}}$. In the ideal scenario, there would be one global minimum in the likelihood surface, but this is generally not the case in practice, with the surface being extremely complex, degenerate and non-convex. Consequently, it is highly probable that the weights will only converge to a local minimum on the likelihood surface, which is dictated to some extent by the initialized values. This is a well-known caveat inherent to the standard training routine for neural networks. SBI provides a means to mitigate this crucial problem with classically trained networks to obtain scientifically rigorous predictions, including reliable uncertainties, of the true targets $\boldsymbol{\tau}$ given the input data $\boldsymbol{d}$.

For the particular mass inference problem studied here, the SBI approach entails the generation of an ensemble of galaxy clusters using a physical model or simulator. We employ some physical priors, in terms of a flat distribution in dynamical mass (as motivated by decoupling from the cosmological model imprinted in the halo mass function) and uniform spatial distribution, in the cluster generation procedure (cf. Section 2.2). Given that our ultimate objective is the inference of cluster masses from the SDSS catalogue, we generate a realistic set of SDSS-like clusters. By feeding the 3D phase-space distributions of this set of generated clusters to our trained neural network, $\mathbb{NN}(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\gamma}}) : \boldsymbol{d} \to \tilde{d}$, we obtain a corresponding set of predicted summaries $\tilde{d}$, which, by design, correspond to the cluster masses. This allows us to characterize the joint probability distribution of data (via the compressed summaries) and parameters, $\mathcal{P}(\tilde{d}, M)$, via a kernel (or neural) density estimator. By slicing this joint distribution at any observed data fed to the network, $\mathbb{NN}(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\gamma}}) : \boldsymbol{d}_{\mathrm{obs}} \to \tilde{d}_{\mathrm{obs}}$, we obtain the approximate posterior as follows:

$$\mathcal{P}(M|\tilde{d}_{\mathrm{obs}}) \approx \mathcal{P}(M|\boldsymbol{d}_{\mathrm{obs}}, \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\gamma}}). \tag{3}$$

Our particular implementation of the SBI pipeline may be summarized via the following steps:

- A convolutional neural network is trained to obtain the desired summary, i.e. cluster mass $\tilde{d}$, from the input data, i.e. the 3D phase-space distribution $\boldsymbol{d} \equiv \{\boldsymbol{x}_{\mathrm{proj}}, \boldsymbol{y}_{\mathrm{proj}}, \boldsymbol{v}_{\mathrm{los}}\}$, using a training set;
- A separate test set of simulated clusters is fed to the trained neural network to obtain the corresponding cluster masses;
- A Gaussian kernel density estimator is used to compute the joint probability distribution of the summary-parameter pairs, i.e. $\mathcal{P}(\tilde{d}, M)$ (cf. Fig. 4);
- A slice through the above distribution at the network summary prediction $\tilde{d}_{\mathrm{obs}}$, for a given input observation $\boldsymbol{d}_{\mathrm{obs}}$, yields the approximate posterior predictive distribution $\mathcal{P}(M|\boldsymbol{d}_{\mathrm{obs}}, \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\gamma}})$.

The above approach has several key advantages. Although the posterior is conditional on the (trained) network weights and choice of hyperparameters, this SBI framework is guaranteed to provide us with a posterior of the parameter of interest with unbiased uncertainties. If the performance and efficacy of the neural network are sub-optimal, then the uncertainties will only be inflated but not erroneously biased. Moreover, the density estimator can be precomputed, such that any slice through the likelihood can be computed almost instantaneously to yield the desired approximate posterior for any given observation. In this work, we make use of a Gaussian KDE (cf. Fig. 4 in Section 4) as our density estimator. The main caveats of this framework are that there is a choice of density estimator with some hyperparameters, such as the bandwidth for the Gaussian KDE, and as for any other SBI approaches, there is some dependence on the total number of simulations used to compute an approximation of the likelihood. Nevertheless, the Gaussian KDE is a fairly robust option as it is not very sensitive to the choice of hyperparameters. In contrast, sophisticated neural density estimators would require further (unsupervised) training and hyperparameter tuning, and would be prone to the shortcoming related to the training of conventional neural networks, i.e. convergence to local minima on the likelihood surface, as outlined above.

3.2 Convolutional neural networks

Convolutional neural networks (hereafter CNNs) (LeCun et al. 1995, 1998) are a particular type of artificial neural network, especially suited for problems where spatial information is crucial. In essence, a CNN is designed as follows: A convolutional kernel, commonly referred to as a *filter*, of a given size, encoding a set of neurons, is applied to each pixel (or voxel for 3D inputs) of the input image and its vicinity as it scans through the whole region. A given pixel in a specific layer is only a function of the pixels in the preceding layer which are enclosed within the window defined by the kernel, known as the *receptive field* of the layer. This yields a *feature map* which encodes high values in the pixels which match the pattern encoded in the weights and biases of the corresponding neurons in the convolutional kernel. These weights and biases are the trainable parameters that are optimized during training.

To extract the series of distinct features of the input image, a convolutional layer generally employs several filters, resulting in a set of feature maps which are then fed as inputs to the subsequent layer. This convolutional operation is typically followed by a *pooling* layer as a subsampling or dimensionality reduction step (Goodfellow et al. 2016). The application of these two types of layers will reduce the initial input image to a compact representation of features, which can be reshaped as a vector. This feature vector is subsequently passed to the final layer which is a fully connected layer to ultimately generate an output (Lecun et al. 2015).

In terms of the mathematical formalism, the convolutional operation may be described as a specialized linear operation, with the discrete convolution implemented via matrix multiplication. As such, a particular convolutional layer, denoted by ℓ, can be computed using

$$x_j^{\ell} = \mathcal{F}\left(\sum_{i \in \mathcal{M}_j} x_i^{\ell-1} \times k_{ij}^{\ell} + b_j^{\ell} \right), \tag{4}$$

where $\mathcal{F}$ denotes the activation function, k represents the convolutional kernel, $\mathcal{M}_j$ corresponds the receptive field and b is the bias parameter (Goodfellow et al. 2016). The role of the activation function is to encode some non-linearity in the convolutional layers, so that a stack of such layers can be used as a generic function approximator. We make use of the rectified linear unit (`ReLU`) activation function (Nair & Hinton 2010) in our neural network, as described in Section 3.3, defined as follows:

$$f(z_i) = \begin{cases} 0, & z_i < 0 \\ z_i, & z_i \geq 0. \end{cases} \tag{5}$$

The `ReLU` activation and its variants are less computationally expensive than other common activation functions, such as the sigmoid and hyperbolic tangent (`tanh`) functions, and mitigates the vanishing gradient issue in training deep neural networks. The latter predicament arises when neurons saturate due to an activation function where $f(z_i) \approx 0$ or $f(z_i) \approx 1$, as in the case of the sigmoid and `tanh` functions, such that the gradient tends to zero. This is inevitably detrimental to the effectiveness of gradient descent during training, resulting in poor training performance. In our neural network implementation, the `ReLU` function also ensures positivity of the final output, i.e. the predicted cluster mass.

The above CNN design allows the neural network to autonomously extract meaningful spatial features from the input image. By stacking several convolutional layers, the network is capable of building an internal hierarchical representation of features encoding the most relevant information from the input image, such that the network is able to identify increasingly complex patterns with the addition of more layers. Hence, convolutional layers provide a natural approach to take spatial context into consideration. A key aspect of such networks is that a stack of convolutional layers increases the sensitivity of subsequent layers to features on increasingly larger scales. In other words, the size of the receptive field becomes larger as we go deeper in the network. Moreover, convolutional layers retain the local information while performing the convolution on adjacent pixels, thereby allowing both local and global information to propagate through the network (Lecun et al. 2015).

CNNs have recently been developed for a range of cosmological applications involving the distribution of cosmic structures on various scales. Deep CNNs, based on the U-Net model (Ronneberger et al. 2015), have been designed to predict the non-linear cosmic structure formation from linear perturbation theory (He et al. 2019) and to include physical effects induced by the presence of massive neutrinos in standard dark matter simulations (Giusarma et al. 2019). Zhang et al. (2019) devised a two-phase CNN architecture to map 3D dark matter fields to their corresponding galaxy distributions in hydrodynamic simulations. 3D deep CNNs have also been used for the generation of mock halo catalogues (Berger & Stein 2019; Bernardini et al. 2020) or for the classification of the distinct features of the cosmic web from N-body simulations (Aragon-Calvo 2019). Physically motivated CNNs, based on the Inception architecture (Szegedy et al. 2017), have been constructed to map 3D dark matter fields to their halo count distributions (Kodi Ramanah et al. 2019) and to augment low-resolution N-body simulations with high-resolution structures (Kodi Ramanah et al. 2020b).

3.3 Neural network architecture

The underlying objective of our SBI framework is to infer the posterior of the dynamical mass of a galaxy cluster,

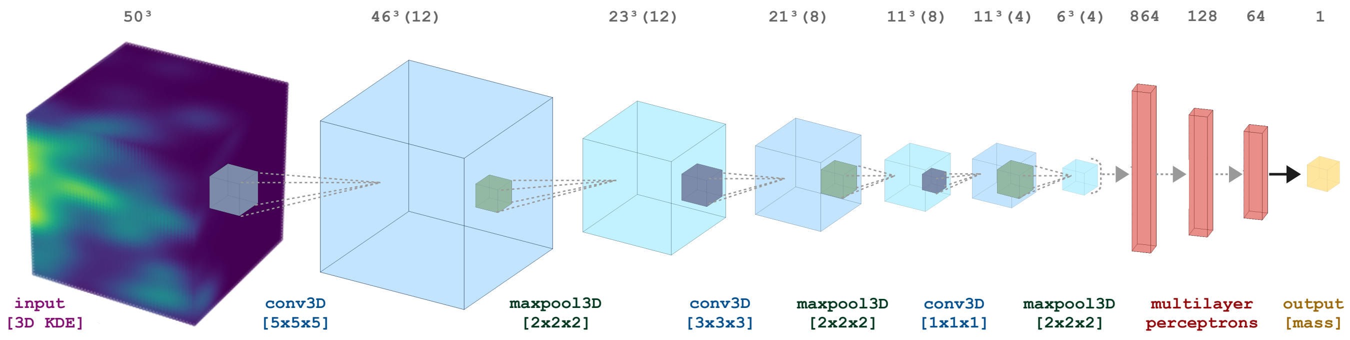


Figure 3. Schematic representation of our CNN$_{\rm 3D}$ architecture to predict cluster masses from their 3D dynamical phase-space distributions. The dimensions of the input 3D slice and those of the subsequent slices, resulting from the convolutional and maxpooling operations, are indicated in the top row, with the number of feature maps per layer given in parentheses. The respective kernel sizes of the latter operations are given in the bottom row, with single strides employed and without use of padding. The CNN extracts the informative spatial features from the 3D phase-space distribution and gradually compresses the high-dimensional space to a single scalar which corresponds to the dynamical cluster mass.

given its 3D phase-space distribution characterized by the projected sky positions and the line-of-sight velocities, i.e. $\mathcal{P}(M|\{\boldsymbol{x}_{\rm proj}, \boldsymbol{y}_{\rm proj}, \boldsymbol{v}_{\rm los}\})$. The neural network takes as input a 3D slice $\tilde{\boldsymbol{\mathcal{D}}}$, which is a 3D array of the Gaussian KDE applied to the phase-space distribution, as described in Section 2, with an example illustrated in Fig. 2. The training data set, therefore, consists of pairs of $\{M, \tilde{\boldsymbol{\mathcal{D}}}\}$.

A schematic of our 3D CNN (hereafter CNN$_{\rm 3D}$) architecture is depicted in Fig. 3. The network extracts spatial features from the input 3D phase-space distribution by performing convolutions with a kernel of size 5×5×5. We employ several such kernels in one layer to probe different aspects of the input 3D slice, yielding a set of feature maps which are subsequently fed to a maxpooling layer for the purpose of dimensionality reduction. We use a $2 \times 2 \times 2$ maxpooling kernel to reduce the slice size by a factor of two. We adopt single strides and no padding for both operations. By repeatedly alternating between these two types of layers, we can reduce the initial 3D distribution to a compact representation of features. At this point, the resulting 3D slice may be flattened to a vector, with this vectorized set of features fed to the final layers which consist of fully connected layers of neurons, i.e. multilayer perceptrons. Finally, the output layer yields the dynamical cluster mass, as desired. We encode `ReLU` (Nair & Hinton 2010) activation functions in the convolutional layers, and linear activations in the final fully connected layers. We highlight the relatively low complexity of the network architecture with $\sim 10^5$ trainable weights.

3.4 Training methodology

We train our CNN$_{\rm 3D}$ model as a regression over the logarithmic cluster mass by minimizing a mean squared error loss function with respect to the network weights. The model and training routine are implemented using the Keras library (Chollet et al. 2015) via a TensorFlow backend (Abadi et al. 2016). We make use of the *Adam* (Kingma & Ba 2014) optimizer, with a learning rate of $\eta = 10^{-4}$ and first and second moment exponential decay rates of $\beta_1 = 0.9$ and $\beta_2 = 0.999$, respectively. The batch size is set to 100. We train the neural network for around 50 epochs, requiring around 10 minutes on an NVIDIA V100 Tensor Core GPU. In order to prevent any overfitting, we adopt the standard regularization technique of early stopping in our training routine. For this purpose, 25% of the original training data set is kept as a separate validation set, with both the training and validation losses monitored during training. We opt for an early stopping criterion of 5 epochs, such that training is halted when the validation loss no longer shows any improvement for 5 consecutive epochs, and the optimized weights of the previously saved best fit model are restored.

4 VALIDATION AND PERFORMANCE

4.1 Uncertainty estimation

We now assess the performance of our optimized CNN$_{\rm 3D}$ model on the evaluation set. As part of the SBI procedure, as described in Section 3.1, we first compute the joint 2D probability density function (PDF), $\mathcal{P}(\tilde{d}, M)$, of the summary statistics $\tilde{d}$ extracted by the neural network and the parameters M obtained using the test set containing around twenty thousand clusters (cf. Section 2.2). Recall that the neural summary statistics in this case are, by design, taken to be point predictions of masses by the CNN$_{\rm 3D}$, while the parameters correspond to the ground truth masses. We make use of a bivariate Gaussian KDE, with a bandwidth scaling of $h_0 = 0.20$, to obtain the 2D PDF depicted in Fig. 4. To infer the posterior PDFs of the dynamical masses of the clusters in the evaluation set, we first obtain the network point predictions using the trained CNN$_{\rm 3D}$ model. We subsequently vertically slice the joint PDF from Fig. 4 at the point estimates to infer the approximate posterior PDFs for the mass of each cluster. From the posteriors, we quantify the 1σ uncertainties by integrating the 68% probability volume, such that the upper and lower 1σ uncertainty limits may be asymmetrical.

4.2 Performance evaluation

Using the inferred posterior mass PDFs for the clusters in the evaluation set, we evaluate the performance of our CNN$_{\rm 3D}$ on the realistic mock catalogue by plotting our model predictions against the ground truth masses of the

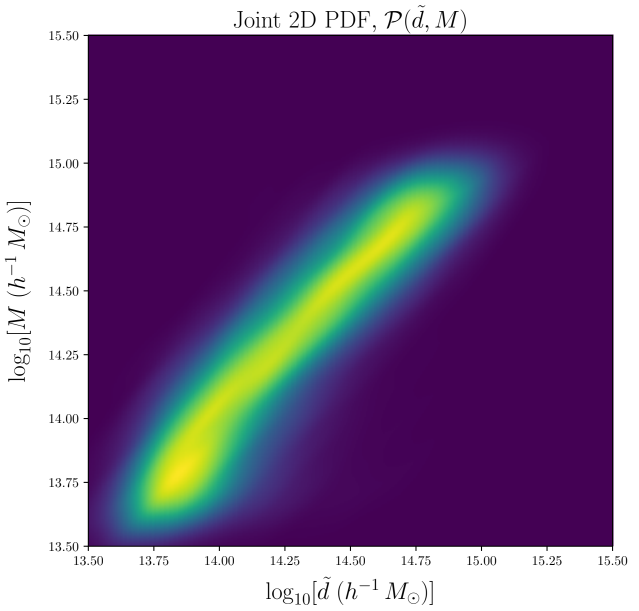


Figure 4. Joint 2D PDF of network predicted summaries $\tilde{d}$ and parameters $\boldsymbol{M}$, i.e. $\mathcal{P}(\tilde{d}, \boldsymbol{M})$, obtained using a bivariate Gaussian KDE with a bandwidth scaling factor of 0.20. A test set consisting of twenty thousand clusters is used to compute this 2D PDF. This is representative of the prediction scatter with respect to the ground truth masses and is employed in our simulation-based inference framework to compute and assign uncertainties associated to point masses predicted by our $\mathrm{CNN_{3D}}$. We obtain the approximate posterior, $\mathcal{P}(\boldsymbol{M}\,|\,\{\boldsymbol{x}_{\mathrm{proj}}, \boldsymbol{y}_{\mathrm{proj}}, \boldsymbol{v}_{\mathrm{los}}\}, \boldsymbol{\theta}, \boldsymbol{\alpha})$, given a set of network weights $\boldsymbol{\theta}$ and hyperparameters $\boldsymbol{\alpha}$, for a particular cluster by making a vertical slice at the neural network predicted value of $\tilde{d}$.

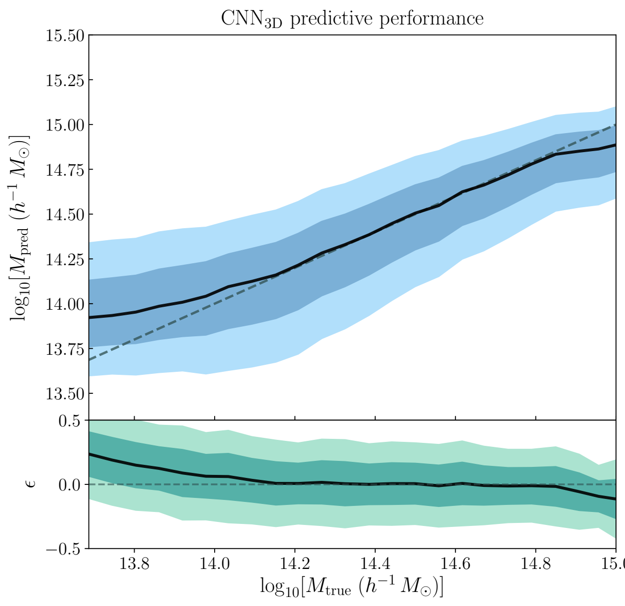


Figure 5. Predictive performance of our $\mathrm{CNN_{3D}}$ model. *Top panel:* $\mathrm{CNN_{3D}}$ predictions against ground truth, depicting the mean prediction (solid line) and the predicted confidence intervals (shaded 1σ and 2σ regions) of the posterior probability density as a function of logarithmic bins of $\boldsymbol{M}_{\mathrm{true}}$ for ~ 5000 galaxy clusters from the evaluation set. The simulation-based inference approach, as expected, yields larger uncertainties for low-mass clusters. *Bottom panel:* Distribution of residual scatter as a function of the logarithmic true cluster mass. The solid line corresponds to the mean logarithmic residual scatter, $\boldsymbol{\epsilon} \equiv \log_{10}(\boldsymbol{M}_{\mathrm{true}}/\boldsymbol{M}_{\mathrm{pred}})$, if we consider only the maximum likelihood predictions (i.e. point mass estimates) from our $\mathrm{CNN_{3D}}$, in logarithmic bins of $\boldsymbol{M}_{\mathrm{true}}$. The shaded bands depict the log-normal scatter (1σ and 2σ regions) about the mean residuals. The $\mathrm{CNN_{3D}}$ tends to overestimate masses of poor clusters below $\log[\boldsymbol{M}_{\mathrm{true}}(h^{-1}\boldsymbol{M}_\odot)] \approx 14.0$ dex. The correspondingly larger uncertainties for clusters in this mass regime demonstrate the reliability of the simulation-based inference framework to provide robust and unbiased uncertainties that are not underestimated.

~ 5000 clusters from the evaluation set in the top panel of Fig. 5. We bin the model predictions in logarithmic mass intervals with the mean prediction and confidence intervals (1σ and 2σ regions) of the posterior probability density depicted via the solid line and shaded regions, respectively. The top panel shows the efficacy of our $\mathrm{CNN_{3D}}$ model to recover the ground truth masses of the clusters from the evaluation set within the 1σ uncertainty limit. The bottom panel displays the distribution of residuals, $\boldsymbol{\epsilon} \equiv \log_{10}(\boldsymbol{M}_{\mathrm{true}}/\boldsymbol{M}_{\mathrm{pred}})$, in the $\mathrm{CNN_{3D}}$ point predictions relative to the ground truth, as a function of the logarithmic cluster mass, with the solid line indicating the mean residual scatter and the shaded bands corresponding to the 1σ and 2σ regions. The $\mathrm{CNN_{3D}}$ predictions have a mean residual and log-normal scatter of $\langle\boldsymbol{\epsilon}\rangle = 0.04$ dex and $\sigma_{\boldsymbol{\epsilon}} = 0.16$ dex.

From the bottom panel of Fig. 5, we observe the tendency of the $\mathrm{CNN_{3D}}$ to overestimate the masses for clusters with masses below $\log[\boldsymbol{M}_{\mathrm{true}}(h^{-1}\boldsymbol{M}_\odot)] \approx 14.1$ dex. This may primarily be attributed to the realistic effects included in our mock catalogue as detailed in Section 4.3 below. Nevertheless, this relatively high residual scatter due to the overprediction of cluster mass is properly accounted for in the network predicted uncertainties, with the lower 1σ and 2σ limits being larger than the upper limits. This demonstrates the capacity of the SBI framework to yield reliable and unbiased uncertainties. Conversely, the network slightly underestimates the masses for the most massive clusters above $\log[\boldsymbol{M}_{\mathrm{true}}(h^{-1}\boldsymbol{M}_\odot)] \approx 14.9$ dex. There is a two-fold plausible explanation for this effect. First, the selection cuts (cf. Section 2.2) to produce the 3D phase-space diagrams may not be sufficiently large to capture all the galaxy members of the massive clusters, resulting in incomplete cluster samples. The second explanation is related to possible mean-reversion edge effects, as also reported by Ntampaka et al. (2016); Ho et al. (2019, 2020), whereby the model predictions of cluster masses at the edge of the mass range considered here are biased towards the average. In general, this systematic bias is related to a neural network's tendency to be more adept at interpolation than extrapolation. To mitigate such biases, we would require more training clusters beyond the edges of the mass regime of the training set, i.e. $\log[\boldsymbol{M}_{\mathrm{true}}(h^{-1}\boldsymbol{M}_\odot)] <$ 13.7 dex and $\log[\boldsymbol{M}_{\mathrm{true}}(h^{-1}\boldsymbol{M}_\odot)] > 15.0$ dex. Note that this mean-reversion effect may also be partially responsible for the overprediction of cluster masses in the low-mass regime.

4.3 Visualization of interloper contamination

Our mock catalogue contains a realistic level of contamination by interloper galaxies, as expected from the actual SDSS observations. In this section, we explicitly highlight how the presence of these spurious galaxies renders the mass estimation extremely challenging.

The interloper contamination principally induces a bias (overestimation) in the neural network predictions which is more significant for the low-mass clusters, substantiating the relatively larger residual scatter for clusters with masses smaller than ∼ 14.1 dex, as depicted in the bottom panel of Fig. 5. This outcome is caused by several low-mass clusters, for which the CNN$_{\rm 3D}$ systematically and significantly overpredicts the mass. To illustrate that interlopers constitute the underlying cause of these inaccurate mass estimates, we compute the contamination per cluster as the mass ratio of interloper clusters to the original cluster. A cluster is considered to be an interloper cluster when it is more massive than the original cluster, is located within a distance of $R_{\rm proj} = (x^2_{\rm proj} + y^2_{\rm proj})^{1/2} = 1.6\ h^{-1}$ Mpc and has $\Delta v_{\rm los} < 2200$ km s^{-1}. An additional factor that exacerbates this problem and renders the task of the CNN$_{\rm 3D}$ more convoluted is the distance between the interloper and original clusters in 3D phase space. The closer the two clusters are together, the more difficult it is to tell them apart. The relative phase-space distance between the two clusters, denoted by c_1 and c_2, respectively, is computed as

$$d(c_1, c_2) = \sqrt{\left(\frac{\Delta x_{\rm proj}}{R_{200c,\rm av}}\right)^2 + \left(\frac{\Delta y_{\rm proj}}{R_{200c,\rm av}}\right)^2 + \left(\frac{\Delta v_{\rm los}}{v_{200c,\rm av}}\right)^2}, \qquad (6)$$

with $R_{200c} = \left(\frac{3M_{200c}}{4\pi 200 \rho_c}\right)^{1/3}$ and $v_{200c} = \sqrt{\frac{GM_{200c}}{R_{200c}}}$,

where $R_{200c,\rm av} = \frac{1}{2}(R^{c_1}_{200c} + R^{c_2}_{200c})$ and $\Delta x_{\rm proj} = x^{c_1}_{\rm proj} - x^{c_2}_{\rm proj}$, with $\Delta y_{\rm proj}$, $\Delta v_{\rm los}$ and $v_{200c,\rm av}$ analogously defined.

Fig. 6 provides a stark illustration of the overwhelming interloper contamination inherent to the individual clusters from the test set, with the colour bar corresponding to the degree of interloper contamination and the marker size corresponding to the inverse of the distance. As can be seen, the clusters with the largest overestimation of their dynamical masses are also the ones whose phase-space diagrams are highly contaminated by interlopers in the form of independent clusters. For some of these clusters, the interloper cluster is around 20 times more massive than the original cluster, which renders the mass estimation extremely challenging. Observational data, such as the SDSS catalogue used in this work, would be similarly plagued by interloper contamination. While the performance of our CNN$_{\rm 3D}$ model with a mean residual and log-normal scatter of $\langle\epsilon\rangle = 0.04$ dex and $\sigma_\epsilon = 0.16$ dex is not as impressive as the recent ML techniques at first glance, this is purely due to the more realistic mock catalogue employed here.

4.4 Information gain with higher dimensionality

In an attempt to illustrate the gain in information by exploiting the full 3D phase-space distribution, we also train our CNN$_{\rm 3D}$ on the mock catalogue from Ho et al. (2019) and compare the performance of our network to their 1D and 2D

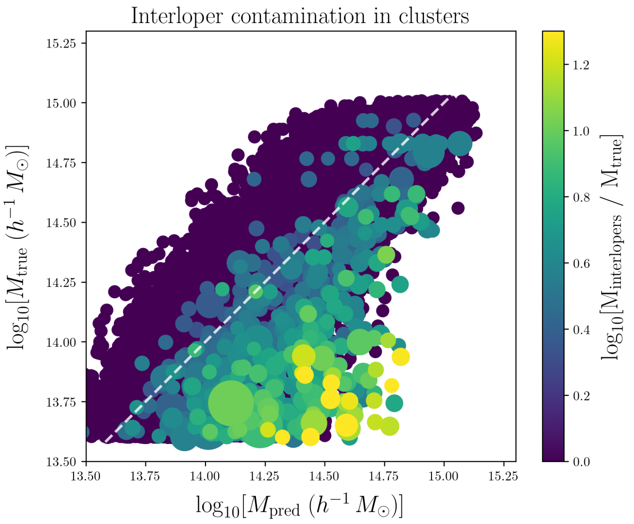


Figure 6. Effect of clustered interlopers on the CNN$_{\rm 3D}$ mass predictions. Each individual mass measurement is coloured according to the relative mass of interloper clusters contaminating the phase-space diagram of the main cluster. In addition, the size of the markers indicates the inverse distance between the interloper and original clusters in the projected phase space, as defined by equation (6). Interlopers residing in relatively massive clusters which overlap closely with the original cluster in the projected phase space can hardly be distinguished from the original cluster members, giving rise to a substantial mass overestimation.

counterparts in terms of the logarithmic residual scatter in Fig. 7. CNN$_{\rm 1D}$ infers cluster masses solely from the univariate distribution of line-of-sight velocities, i.e. $\{v_{\rm los}\}$, while CNN$_{\rm 2D}$ additionally takes as input the sky-projected radial positions given by $R_{\rm proj} = (x^2_{\rm proj} + y^2_{\rm proj})^{1/2}$, such that it relies on the joint distribution of $\{R_{\rm proj}, v_{\rm los}\}$. Note that only the point (maximum *a posteriori*) estimates from our approach are used to produce the comparison plot displayed in Fig. 7. As expected, we find that the precision of the mass estimator, as indicated by the log-normal residual scatter (shaded 1σ and 2σ regions), improves progressively with further information, thereby justifying the development and application of our CNN$_{\rm 3D}$ model in this work.

As in our previous work (NF2020), we quantify the precision of cluster mass estimation in terms of the total scatter about the best-fit power-law relation between the ground truth and predicted cluster masses. Adopting the approach employed in Wojtak et al. (2018), we express the total scatter σ into a richness-dependent component given by σ_N and a richness-independent part denoted by σ_0, as follows:

$$\sigma^2 = \sigma^2_N (N_{\rm mem}/100)^{-1} + \sigma^2_0, \qquad (7)$$

where $N_{\rm mem}$ indicates the number of cluster members. We determine the values of σ_N and σ_0 by fitting the above equation to the logarithmic residuals in the cluster mass predictions for the test set. We carry out this procedure for the three CNN models and also include the results of the neural flow mass estimator (NF2020). Note that the same mock cluster catalogue from Ho et al. (2019) was used for all the methods. The recent ML techniques, illustrated in Fig. 8, all outperform the traditional cluster mass estimators (cf. Fig. 6 in NF2020) extensively tested in the

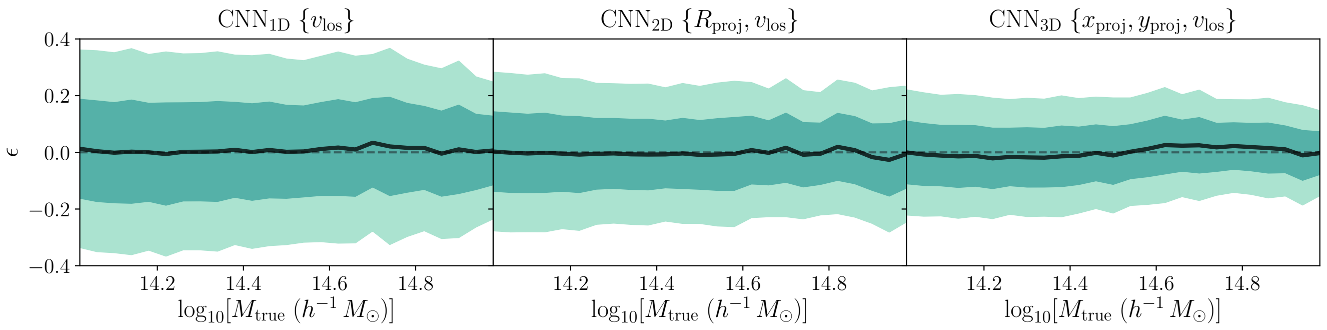


Figure 7. Comparison of the performance of three CNN models with distinct input dimensionality. Performance is quantified in terms of the residual scatter, $\epsilon \equiv \log_{10}(M_{\rm true}/M_{\rm pred})$, in the CNN point predictions relative to the ground truth. The mean residual scatter is depicted via solid dark lines, with the shaded bands corresponding to the log-normal scatter (1σ and 2σ regions). The CNNs are trained with progressively larger dimensionality of the phase-space distribution of the same mock cluster catalogue from Ho et al. (2019). This illustrates the gain in constraining power when the information content of the full 3D phase-space distribution of galaxies is exploited instead of relying merely on the velocity dispersion as in the 1D case displayed in the left panel. The $\mathrm{CNN_{1D}}$ and $\mathrm{CNN_{2D}}$ results are reproduced from Ho et al. (2019).

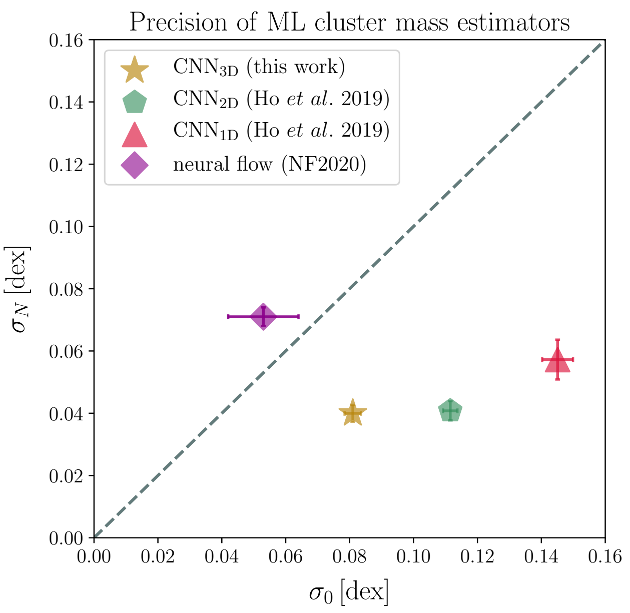


Figure 8. Comparison of the precision of three recently proposed ML cluster mass estimators, along with the $\mathrm{CNN_{3D}}$ model from this work. We quantify the precision in terms of richness-dependent error σ_N and richness-independent systematic error σ_0 (cf. equation (7)). In accordance with Fig. 7, this shows the progressive improvement in the precision of CNN models with increasing input dimensionality. Our $\mathrm{CNN_{3D}}$ model is also less sensitive to the cluster richness than the neural flow mass estimator (NF2020).

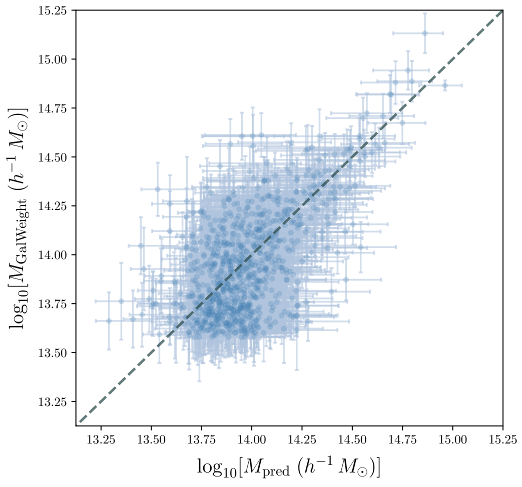
Galaxy Cluster Mass Comparison Project (Old et al. 2015), which are not shown for the sake of clarity. For our $\mathrm{CNN_{3D}}$ model, we find $\sigma_N = 0.04$ dex and $\sigma_0 = 0.08$ dex, with the richness-dependent error smaller by a factor of two relative to $3/(\sqrt{2N_{\rm mem}}\ln 10) = 0.09$ dex expected for the mass estimation based solely on the scaling relation with the velocity dispersion. As expected from Fig. 7, Fig. 8 shows a progressive improvement in precision going from $\mathrm{CNN_{1D}}$ to $\mathrm{CNN_{3D}}$ due to the gain in constraining power when exploiting the full information from the 3D phase-space distribution of galaxies rather than relying only on the velocity dispersion. In general, the CNN mass estimators are less sensitive to cluster richness than the neural flow model. Figs. 7 and 8, therefore, present an adequate depiction of the network performance and a fair comparison with recent ML methods, demonstrating the precision of our $\mathrm{CNN_{3D}}$ model.

5 APPLICATION TO SDSS CATALOGUE

We now apply the trained neural network to redshift data from the SDSS catalogue to infer the dynamical masses of the galaxy clusters and use the bivariate KDE (cf. Fig. 4) to derive their corresponding uncertainties. We subsequently perform a detailed comparison of the inferred dynamical masses to recent measurements from literature.

We use the publicly available *GalWeight* catalogue containing galaxy clusters found in the main galaxy sample of the SDSS with the GALWEIGHT algorithm (Abdullah et al. 2020b).[2] We select 910 galaxy clusters at comoving distances shorter than $250\,h^{-1}$ Mpc for which the catalogue is complete down to its minimum cluster mass, i.e. $\log_{10}[M_{200c}(h^{-1}M_\odot)] \approx 13.7$ dex. For each cluster, we find velocities and positions of all galaxies from the main spectroscopic SDSS sample in its field. We use the same cuts in the projected phase space coordinates as for the mock observations. We also adopt cluster centres and redshifts from the *GalWeight* catalogue which set them at the peak of a smoothed galaxy density in the projected phase space. The cluster catalogue also provides also the measurements of dynamical cluster masses based on the virial theorem with the surface term computed for NFW density profile extrapolated beyond the virial sphere. The mass estimations account for cluster membership by a special scheme of assigning weights to all galaxies observed in the phase-space diagram. The scheme was devised using mock data generated from cosmological simulations (Abdullah et al. 2018).

[2] https://mohamed-elhashash-94.webself.net/galwcat/



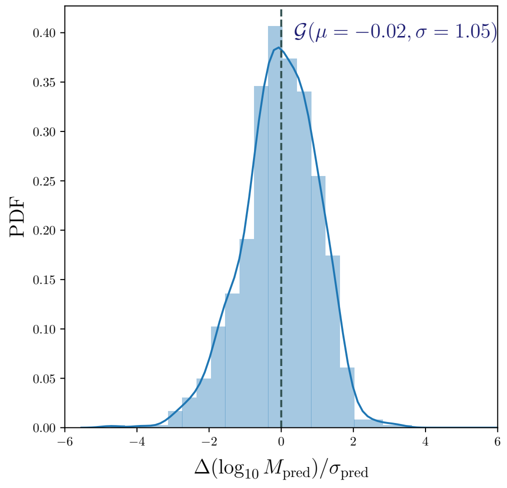


Figure 9. Comparison of our SDSS cluster mass predictions with the recent estimates from the *GalWeight* galaxy cluster catalogue (Abdullah et al. 2020b), illustrated via a qualitative visual depiction (*top panel*) and a 1D PDF (*bottom panel*) of the difference between the predictions, normalized by the corresponding uncertainties in our predictions. The resulting distribution is approximately characterized by a normalized Gaussian distribution, quantitatively indicating the overall consistency between our cluster mass predictions and those from Abdullah et al. (2020b).

The galaxy clusters from the catalogue are subjected to an initial preprocessing step similar to the preparation of the training set. We compute the 3D Gaussian KDE of their respective phase-space distributions, as outlined in Section 2.3, with the resulting 3D slices subsequently provided as inputs to our CNN$_{\rm 3D}$ model. The point estimates are then fed to the SBI pipeline to obtain their respective uncertainties, resulting in the inferred dynamical masses for the SDSS clusters. To compare our predictions with the recent results from Abdullah et al. (2020b), we compute the 1D PDF of the difference between the predictions, normalized by the uncertainty $\sigma_{\rm pred}$ in our prediction, i.e. $\Delta(\log_{10} M_{\rm pred})/\sigma_{\rm pred}$, where $\Delta(\log_{10} M_{\rm pred}) = \log_{10}(M_{\rm pred}/M_{\rm GalWeight})$ and $M_{\rm GalWeight}$ is the cluster mass estimate from the *GalWeight* galaxy cluster catalogue (Abdullah et al. 2020b). We compute the 1D PDF by binning

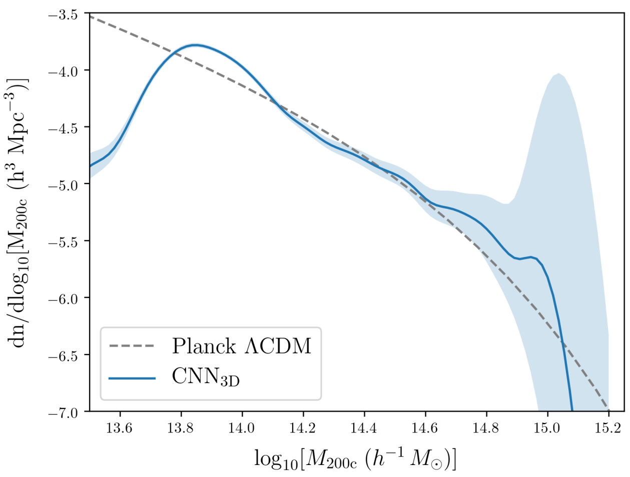


Figure 10. Cluster mass function derived from the dynamical mass estimates of SDSS clusters obtained in this work (CNN$_{\rm 3D}$) and the theoretical halo mass function as predicted for Planck ΛCDM cosmology. The CNN$_{\rm 3D}$ mass function is computed by means of a kernel density estimator, weighted inversely by the uncertainties of the mass estimates. Bootstrapping was used to obtain the predicted 1σ confidence interval indicated by the shaded band.

this mass contrast, with the resulting distribution illustrated in Fig. 9. The latter distribution has a mean and standard deviation of $\mu = -0.02$ and $\sigma = 1.05$, respectively, which approximately corresponds to a normalized Gaussian distribution. This highlights the overall consistency of our mass predictions with those from Abdullah et al. (2020b), with the absence of kurtosis implying a negligible bias or error underestimation/overestimation with respect to the former literature estimates, which would otherwise render the 1D PDF leptokurtic or platykurtic.

Fig. 10 shows the mass function derived from our measurements of dynamical masses adopting the total area (8250 square degrees) of the SDSS Legacy Survey. We compare our results to the theoretical mass function computed for Planck cosmology (Planck Collaboration et al. 2018) using universal fitting formula from Tinker et al. (2008). It is readily apparent that our measurements recover the mass function of the standard cosmological model down to a mass of $\sim 10^{14.1} h^{-1} M_\odot$ which is nearly a mass completeness limit of the *GalWeight* cluster catalogue (Abdullah et al. 2020b).

6 CONCLUSIONS AND OUTLOOK

We have presented a simulation-based inference framework, based on 3D convolutional feature extractors, to infer the galaxy cluster masses from their 3D dynamical phase-space distributions, which consist of the projected positions in the sky and the galaxy line-of-sight velocities, i.e. $\{x_{\rm proj}, y_{\rm proj}, v_{\rm los}\}$. The simulation-based inference framework allows us to quantify the uncertainties on the inferred masses in a straightforward and unbiased way. By optimally exploiting the information content of the full projected phase-space distribution, the network yields remarkable constraints on the dynamical cluster masses. As such, this fast and robust tool is a novel and complementary addition to the state-of-the-art machine learning techniques in the cluster mass estimation toolbox.

We train our CNN$_{3D}$ model using a realistic mock cluster catalogue emulating the properties of the actual SDSS catalogue. Once optimized on the training set, we use our CNN$_{3D}$ model within a simulation-based inference framework to infer the dynamical masses of a set of SDSS clusters and their associated uncertainties, which allow us to recover, for the first time, the theoretical ΛCDM mass function down to a mass of $\sim 10^{14.1} h^{-1} M_{\odot}$ using a machine learning-based cluster mass estimator. The primary advantage of simulation-based inference, as employed in this work, is that it yields robust and unbiased uncertainties. If the neural network used to perform the feature extraction to derive summary statistics is sub-optimal, the uncertainties will only be inflated and not biased. Moreover, we clearly illustrate the difficulties related to the presence of interlopers close to the cluster centre when dealing with actual observations. In practice, there exists no effective solution to this predicament inherent to the mass estimation problem. As a consequence, our simulation-based inference framework yields correspondingly larger uncertainties for such problematic clusters. This conservative approach ensures that the uncertainties of highly contaminated clusters are not underestimated.

The design of our network architecture, based on the use of 3D convolutional kernels, is justified by the gain in constraining power with progressively larger dimensionality of the input phase-space distribution, as substantiated by smaller log-normal residual scatter and improved precision (cf. Figs. 7 and 8, respectively). Compared to our recently proposed neural flow mass estimator (Kodi Ramanah et al. 2020a), our CNN$_{3D}$ model is more robust to the size of galaxy samples with spectroscopic redshifts, i.e. galaxy selection effects. The former method employs normalizing flows, implemented via a stack of multilayer perceptrons, to predict the posterior cluster mass PDFs from 2D phase-space distributions $\{R_{\rm proj}, v_{\rm los}\}$, thereby deriving uncertainties in a conceptually distinct approach.

The performance of our CNN$_{3D}$ mass estimator, along with that of our recent neural flow mass estimator and the variational inference approach by Ho et al. (2020), provides exciting avenues to infer cosmological constraints from the SDSS catalogue using cluster abundances (Abdullah et al. 2020a). These three novel machine learning algorithms yield robust and reliable cluster masses with complementary ways of deriving uncertainties and, therefore, may be utilized to constrain the cluster mass function to complement standard approaches based on traditional mass estimators.

ACKNOWLEDGEMENTS

We acknowledge Tom Charnock for providing us with the motivation to make use of simulation-based inference using neural networks. We also thank Matthew Ho for his insights pertaining to the use of KDE for preprocessing the phase-space distribution. DKR is a DARK fellow supported by a Semper Ardens grant from the Carlsberg Foundation (reference CF15-0384). This work was supported by a VILLUM FONDEN Investigator grant (project number 16599). This work has made use of the Horizon and Henon Clusters hosted by Institut d'Astrophysique de Paris. This work has been done within the activities of the Domaine d'Intérêt Majeur (DIM) Astrophysique et Conditions d'Apparition de la Vie (ACAV), and received financial support from Région Ile-de-France.

DATA AVAILABILITY

The data underlying this article will be shared on reasonable request to the corresponding author.

REFERENCES

Abadi M., et al., 2016, preprint, (arXiv:1603.04467)
Abdullah M. H., Wilson G., Klypin A., 2018, ApJ, 861, 22
Abdullah M. H., Klypin A., Wilson G., 2020a, preprint, (arXiv:2002.11907)
Abdullah M. H., Wilson G., Klypin A., Old L., Praton E., Ali G. B., 2020b, ApJS, 246, 2
Akeret J., Refregier A., Amara A., Seehars S., Hasner C., 2015, J. Cosmology Astropart. Phys., 2015, 043
Alsing J., Wandelt B., 2019, MNRAS, 488, 5093
Alsing J., Wandelt B., Feeney S., 2018, MNRAS, 477, 2874
Alsing J., Charnock T., Feeney S., Wandelt B., 2019, MNRAS, 488, 4440
Aragon-Calvo M. A., 2019, MNRAS, 484, 5771
Armitage T. J., Kay S. T., Barnes D. J., 2019, MNRAS, 484, 1526
Behroozi P. S., Wechsler R. H., Wu H.-Y., 2013, ApJ, 762, 109
Benson A. J., 2012, New Astron., 17, 175
Berger P., Stein G., 2019, MNRAS, 482, 2861
Bernardini M., Mayer L., Reed D., Feldmann R., 2020, MNRAS,
Calderon V. F., Berlind A. A., 2019, MNRAS, 490, 2367
Charnock T., Lavaux G., Wandelt B. D., 2018, Phys. Rev. D, 97, 083004
Chollet F., et al., 2015, Keras, https://keras.io
Cora S. A., 2006, MNRAS, 368, 1540
Cora S. A., et al., 2018, MNRAS, 479, 2
Cranmer K., Brehmer J., Louppe G., 2019, preprint, (arXiv:1911.01429)
Croton D. J., et al., 2006, MNRAS, 365, 11
DESI Collaboration et al., 2016, preprint, (arXiv:1611.00036)
Diemer B., More S., Kravtsov A. V., 2013, ApJ, 766, 25
Diggle P. J., Gratton R. J., 1984, Journal of the Royal Statistical Society: Series B (Methodological), 46, 193
Germain M., Gregor K., Murray I., Larochelle H., 2015, preprint, (arXiv:1502.03509)
Giusarma E., Reyes Hurtado M., Villaescusa-Navarro F., He S., Ho S., Hahn C., 2019, preprint, (arXiv:1910.04255)
Goodfellow I., Bengio Y., Courville A., 2016, Deep learning. MIT press
He S., Li Y., Feng Y., Ho S., Ravanbakhsh S., Chen W., Póczos B., 2019, Proceedings of the National Academy of Science, 116, 13825
Ho M., Rau M. M., Ntampaka M., Farahi A., Trac H., Póczos B., 2019, ApJ, 887, 25
Ho M., Farahi A., Rau M. M., Trac H., 2020, preprint, (arXiv:2006.13231)
Huang C.-W., Krueger D., Lacoste A., Courville A., 2018, preprint, (arXiv:1804.00779)
Ishiyama T., et al., 2020, preprint, (arXiv:2007.14720)
Ivezic Z., et al., 2008, preprint, (arXiv:0805.2366)

Jennings E., Madigan M., 2017, Astronomy and Computing, 19, 16

Jimenez Rezende D., Mohamed S., 2015, preprint, (arXiv:1505.05770)

Kingma D. P., Ba J., 2014, preprint, (arXiv:1412.6980)

Kingma D. P., Salimans T., Jozefowicz R., Chen X., Sutskever I., Welling M., 2016, preprint, (arXiv:1606.04934)

Klypin A., Yepes G., Gottlöber S., Prada F., Heß S., 2016, MNRAS, 457, 4340

Knebe A., et al., 2018, MNRAS, 474, 5206

Kodi Ramanah D., Charnock T., Lavaux G., 2019, Phys. Rev. D, 100, 043515

Kodi Ramanah D., Wojtak R., Ansari Z., Gall C., Hjorth J., 2020a, preprint, (arXiv:2003.05951)

Kodi Ramanah D., Charnock T., Villaescusa-Navarro F., Wandelt B. D., 2020b, MNRAS, 495, 4227

LeCun Y., Bengio Y., et al., 1995, The handbook of brain theory and neural networks, 3361, 1995

LeCun Y., Bottou L., Bengio Y., Haffner P., et al., 1998, Proceedings of the IEEE, 86, 2278

Leclercq F., 2018, Phys. Rev. D, 98, 063511

Lecun Y., Bengio Y., Hinton G., 2015, Nature, 521, 436

Lintusaari J., et al., 2017, preprint, (arXiv:1708.00707)

Merloni A., et al., 2012, preprint, (arXiv:1209.3114)

Nair V., Hinton G. E., 2010, in Proceedings of the 27th International Conference on Machine Learning. ICML'10. Omnipress, USA, pp 807–814, http://dl.acm.org/citation.cfm?id=3104322.3104425

Ntampaka M., Trac H., Sutherland D. J., Battaglia N., Póczos B., Schneider J., 2015, ApJ, 803, 50

Ntampaka M., Trac H., Sutherland D. J., Fromenteau S., Póczos B., Schneider J., 2016, ApJ, 831, 135

Old L., et al., 2015, MNRAS, 449, 1897

Papamakarios G., Murray I., 2016, preprint, (arXiv:1605.06376)

Papamakarios G., Pavlakou T., Murray I., 2017, preprint, (arXiv:1705.07057)

Papamakarios G., Sterratt D. C., Murray I., 2018, preprint, (arXiv:1805.07226)

Planck Collaboration et al., 2014, A&A, 571, A16

Planck Collaboration et al., 2018, preprint, (arXiv:1807.06209)

Racca G. D., et al., 2016, in Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series. p. 99040O (arXiv:1610.05508), doi:10.1117/12.2230762

Ronneberger O., Fischer P., Brox T., 2015, in International Conference on Medical image computing and computer-assisted intervention. pp 234–241

Sheather S. J., 2004, Statistical Science, 19, 588

Strauss M. A., et al., 2002, AJ, 124, 1810

Sutherland D. J., Xiong L., Póczos B., Schneider J., 2012, preprint, (arXiv:1202.0302)

Szegedy C., Ioffe S., Vanhoucke V., Alemi A. A., 2017, in AAAI Conference on Artificial Intelligence. p. 12

Tinker J., Kravtsov A. V., Klypin A., Abazajian K., Warren M., Yepes G., Gottlöber S., Holz D. E., 2008, ApJ, 688, 709

Uria B., Côté M.-A., Gregor K., Murray I., Larochelle H., 2016, preprint, (arXiv:1605.02226)

Villaescusa-Navarro F., et al., 2020, ApJS, 250, 2

Wand M. P., Jones M. C., 1994, Kernel smoothing. CRC press

Wang Y.-C., Xie Y.-B., Zhang T.-J., Huang H.-C., Zhang T., Liu K., 2020, preprint, (arXiv:2005.10628)

Wojtak R., et al., 2018, MNRAS, 481, 324

Yan Z., Mead A. J., Van Waerbeke L., Hinshaw G., McCarthy I. G., 2020, preprint, (arXiv:2005.11819)

Zhang X., Wang Y., Zhang W., Sun Y., He S., Contardo G., Villaescusa-Navarro F., Ho S., 2019, preprint, (arXiv:1902.05965)

This paper has been typeset from a TEX/LATEX file prepared by the author.