The Astrophysical Journal, 887:25 (17pp), 2019 December 10 https://doi.org/10.3847/1538-4357/ab4f82

A Robust and Efficient Deep Learning Method for Dynamical Mass Measurements of Galaxy Clusters

Matthew Ho[1], Markus Michael Rau[1], Michelle Ntampaka[2,3], Arya Farahi[1], Hy Trac[1], and Barnabás Póczos[4]
[1] McWilliams Center for Cosmology, Department of Physics, Carnegie Mellon University, Pittsburgh, PA 15213, USA; mho1@andrew.cmu.edu
[2] Center for Astrophysics, Harvard & Smithsonian, Cambridge, MA 02138, USA
[3] Harvard Data Science Initiative, Harvard University, Cambridge, MA 02138, USA
[4] School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA
Received 2019 March 12; revised 2019 October 15; accepted 2019 October 18; published 2019 December 6

Abstract

We demonstrate the ability of convolutional neural networks (CNNs) to mitigate systematics in the virial scaling relation and produce dynamical mass estimates of galaxy clusters with remarkably low bias and scatter. We present two models, $\mathrm{CNN_{1D}}$ and $\mathrm{CNN_{2D}}$, which leverage this deep learning tool to infer cluster masses from distributions of member galaxy dynamics. Our first model, $\mathrm{CNN_{1D}}$, infers cluster mass directly from the distribution of member galaxy line-of-sight velocities. Our second model, $\mathrm{CNN_{2D}}$, extends the input space of $\mathrm{CNN_{1D}}$ to learn on the joint distribution of galaxy line-of-sight velocities and projected radial distances. We train each model as a regression over cluster mass using a labeled catalog of realistic mock cluster observations generated from the MultiDark simulation and UniverseMachine catalog. We then evaluate the performance of each model on an independent set of mock observations selected from the same simulated catalog. The CNN models produce cluster mass predictions with lognormal residuals of scatter as low as 0.132 dex, greater than a factor of 2 improvement over the classical M–σ power-law estimator. Furthermore, the CNN model reduces prediction scatter relative to similar machine-learning approaches by up to 17% while executing in drastically shorter training and evaluation times (by a factor of 30) and producing considerably more robust mass predictions (improving prediction stability under variations in galaxy sampling rate by 30%).

Key words: cosmology: theory – galaxies: clusters: general – galaxies: kinematics and dynamics – methods: statistical

1. Introduction

Galaxy clusters are the most massive gravitationally bound structures in the universe. Clusters are complex, dark-matter-dominated systems of mass $\gtrsim 10^{14}h^{-1}\,M_{\odot}$. Galaxy clusters dominate the high-mass tail of the halo mass function (HMF), and cluster number density is a highly sensitive probe of the growth of structure. Because of this distinction, measurements of galaxy cluster abundance as a function of mass and redshift are a major method to test cosmological models (e.g., Voit 2005; Allen et al. 2011; Mantz et al. 2015; Planck Collaboration et al. 2016).

Utilizing cluster abundance in precision cosmology requires a large, well-defined cluster sample and robust mass measurement methods. Furthermore, modern cluster measurement techniques are expected to place a strong emphasis on efficiency and automation, as the wealth of detailed cluster data is expected to greatly increase with current and upcoming surveys such as DES, LSST, *WFIRST*, and *Euclid* (Dodelson et al. 2016). Current methods infer cluster masses from one of several mass-dependent observables, which occur at a variety of wavelengths, including the emission of X-rays by hot intracluster gas (e.g., Mantz et al. 2016; Giles et al. 2017), the scattering of CMB photons on intracluster plasma (e.g., Sunyaev & Zeldovich 1972; Planck Collaboration et al. 2016), the gravitational lensing of background light (e.g., Applegate et al. 2014; McClintock et al. 2019), and the properties of luminous member galaxies (e.g., Old et al. 2014). Galaxy-based techniques probe clusters using multiband and spectroscopic measurements, relating mass to cluster features such as richness (e.g., Yee & Ellingson 2003; Old et al. 2014; Baxter et al. 2016), escape velocity profile (e.g., Diaferio & Geller 1997; Diaferio 1999; Gifford & Miller 2013), and member dynamics (e.g., Gerke et al. 2005; Old et al. 2014). For an extensive review and comparison of galaxy-based techniques, see Old et al. (2014).

Dynamical mass measurements are a broad classification of galaxy-based techniques which infer cluster mass from the line-of-sight (LOS) velocity distribution of galaxies. The classical approach for dynamical measurements is the M–σ scaling relation, which connects a virialized cluster's total mass to the velocity dispersion of its galaxies via a power law (e.g., Evrard et al. 2008). Dynamical measurements of this nature were famously used to infer the existence of dark matter in the Coma cluster (Zwicky 1933). While historically significant, the M–σ relation makes several assumptions about clusters which are unreliable in practice, including spherical symmetry, gravitational equilibrium, and perfect member selection. In reality, proper modeling of clusters requires careful consideration of systematics such as dynamical substructure (e.g., Saro et al. 2013; Wojtak 2013; Old et al. 2018), halo environment (e.g., White et al. 2010), triaxiality (e.g., Skielboe et al. 2012; Saro et al. 2013; Svensmark et al. 2015), and mergers (e.g., Evrard et al. 2008; Ribeiro et al. 2011). In addition, galaxy selection effects are a primary source of scatter in dynamical mass predictions, as the member sample can be incomplete or otherwise contaminated by unbound interloper galaxies (Saro et al. 2013; Old et al. 2015; Wojtak et al. 2018). Modern applications of the M–σ relation mitigate these effects using complex membership modeling and interloper removal schemes (e.g., Wojtak et al. 2007; Mamon et al. 2013; Farahi et al. 2016, 2018; Abdullah et al. 2018).

Recently, a suite of machine-learning (ML) algorithms have been used to reconstruct dynamical cluster masses. This class of methods often involves training an ML model on a large data set of simulation-generated mock observations to then produce inference on unlabeled observations. Ntampaka et al. (2015, 2016) introduced an ML method to infer mass from the full LOS velocity distribution of cluster members. This method attempts to capture higher-order features of the velocity distribution using a support distribution machine (SDM; Sutherland et al. 2012) and has been shown to reduce the scatter of traditional dynamical mass predictions (M–σ) by a factor of 2. Armitage et al. (2019a) applied a variety of simple regression models on a hand-built feature set of dynamics observables to achieve similar error margins. Calderon & Berlind (2019) regressed mass on a list of cluster properties via several more complex ML models (XGBoost, Random Forests, and neural networks) to ultimately achieve prediction improvements comparable to previous ML approaches. Calderon & Berlind briefly discussed the impacts of simulation assumptions on ML model fitting and produced preliminary predictions on cluster observations from SDSS.

In this paper, we introduce a novel deep learning methodology for measuring cluster masses from galaxy dynamics. The core of our model is a convolutional neural network (CNN), a deep learning tool which has received considerable attention for its applications in image recognition. We utilize kernel density estimators (KDEs) to create phase-space mappings of each cluster's galaxy dynamics distribution which serve as "image" inputs to our CNNs. We train CNNs as a regression over logarithmic cluster mass using a catalog of realistic mock observations. We then use the trained CNN models to perform inference on unseen mock test data to evaluate model performance. This paper is organized into the following sections: in Section 2, we discuss our simulation, galaxy labeling, and mock observation procedures. In Section 3, we discuss the background and methodology surrounding the application of our ML algorithm. In Section 4, we describe details of several comparative methods which will serve as a baseline for evaluating the performance of our model. In Section 5, we discuss performance metrics and evaluate the performance of our model. We summarize conclusions in Section 6. Lastly, we provide an appendix describing the explicit calculations of our mock observables (Appendix). Upon publication of this manuscript, the code developed for this analysis will be made publicly available on Github.[5]

2. Data Set

In this section, we discuss the creation of our data set, namely the calculation of mock cluster observations. Clusters and galaxies are modeled as dark matter halos present in a $z = 0.117$ snapshot of the MultiDark Planck 2 N-body simulation (Klypin et al. 2016). Simulated clusters are converted to realistic mock observables in agreement with the simulation's original cosmology. Mock cluster observations are designed to include realistic systematics which would impact dynamical mass estimates.

2.1. Simulation and Galaxy Assembly

The mock observations were created using data from the MultiDark Planck 2 simulation (MDPL2; Klypin et al. 2016). MDPL2 is a large N-body dark matter simulation which evolves 3840^3 particles from $z = 120$ to $z = 0$ within a box length of 1 h^{-1} Gpc and at a mass resolution of $1.51 \times 10^9\ h^{-1}\,M_\odot$. The force resolution varies from 13 h^{-1} kpc at high z to 5 h^{-1} kpc at low z. The simulation is executed using the publicly available L-GADGET-2 code (Springel 2005) and uses a ΛCDM cosmology consistent with 2013 *Planck* data (Planck Collaboration et al. 2014): $\Omega_\Lambda = 0.693$, $\Omega_m = 0.307$, $h = 0.678$, $n = 0.96$, and $\sigma_8 = 0.8228$.

We model clusters and their member galaxies as host halos and subhalos, respectively. We utilize a halo catalog generated from MDPL2 simulation data using the ROCKSTAR halo finder (MDPL2 Rockstar; Behroozi et al. 2013). The MDPL2 Rockstar catalog identifies a hierarchy of host halos and subhalos within the MDPL2 simulation at sequential redshift snapshots throughout the simulation evolution. Clusters are painted onto host halos, inheriting properties such as mass, radius, position, and velocity. The mass definition applied for our simulated clusters is M_{200c}, calculated via spherical overdensities of 200 times the critical density of the MDPL2 simulation. Galaxies are painted onto subhalos through the galaxy assignment procedure UniverseMachine (Behroozi et al. 2019). By tracking the gravitational evolution of disrupted halos below the resolution limit of ROCKSTAR, UniverseMachine produces a rich and detailed catalog of simulated galaxies ideal for our data set. UniverseMachine determines stellar formation rates and masses for each galaxy, which are consistent with observational constraints. UniverseMachine galaxies inherit position and velocity from their associated subhalos.

We conduct this analysis on a publicly available $z = 0.117$ snapshot of the MDPL2 simulation.[6] The MDPL2 Rockstar and UniverseMachine catalogs provide mass, comoving position, and proper velocity information for host halos and subhalos. Host halos included in our sample are constrained to $M_{200c} \geqslant 10^{13.5}\ h^{-1}\,M_\odot$. Galaxy subhalos in our sample are restricted to a stellar mass limit of $M_{\rm stellar} \geqslant 10^{9.5}\ h^{-1}\,M_\odot$.

2.2. Contaminated Mock Observations

The mock observations are designed to model physical and selection effects inherent in real cluster measurements. The physical effects (cluster mergers, triaxiality) are encoded in the distributions of cluster members and surrounding material. The selection effects (interlopers) arise from nonmember galaxies positioned along the LOS and with similar perceived LOS velocities to the host cluster. To account for these effects, the mock observations select samples of member galaxies by taking large, fixed-size cylindrical cuts positioned at the cluster center and oriented along the LOS axis. This cut allows information regarding interlopers and cluster shape to contaminate the sample. We will refer to the realistic mock observations as the contaminated catalog. A previous version of the mock observation procedure used in this paper is described in Ntampaka et al. (2016).

In creating this set of mock cluster observations, we make the following assumptions: (1) all subhalos tracked by UniverseMachine above $M_{\rm stellar} \geqslant 10^{9.5}\ h^{-1}\,M_\odot$ are assumed to represent a galaxy, with the galaxy inheriting its subhalo's position and velocity. (2) Host halos with mass $M_{200c} \geqslant 10^{13.5}\ h^{-1}\,M_\odot$ are considered to be cluster candidates. (3) The cluster center is

[5] https://github.com/McWilliamsCenter/halo_cnn

[6] https://www.cosmosim.org/

assumed to be known and consistent with the host halo's position and velocity. (4) Each cluster observation considers a unique observer assumed to lie at $z = 0$ along the chosen LOS. Obstructions, lensing, and other observational artifacts are not accounted for.

Before observational cuts are calculated, the simulation snapshot box is padded on each side to account for periodic boundary conditions. At each box face and edge, a slice of galaxy data is duplicated from across the periodic boundary. The padding width is calculated from simulation data and overestimated so as to not exclude any galaxies that might be captured in a cluster's cylinder cut. This analysis used a padding width of $112\,h^{-1}$ Mpc. This creates a final padded cube of side length $1.224\,h^{-1}$ Gpc.

For a given LOS axis (Section 2.3), we determine cluster membership by first calculating the position and velocity observables for each cluster–galaxy pair. We calculate $x_{\rm proj}$, $y_{\rm proj}$, and $v_{\rm los}$ for all galaxies around a cluster center, where $x_{\rm proj}$ and $y_{\rm proj}$ are projected plane-of-sky x' and y' positions, and $v_{\rm los}$ is the net LOS velocity. The net velocity, $v_{\rm los}$, is given by the sum of the object's relative peculiar velocity and Hubble flow along the LOS. The quantities $x_{\rm proj}$, $y_{\rm proj}$, and $v_{\rm los}$ are expressed as relative values to the cluster candidate's center. We also calculate the projected plane-of-sky radial distance $R_{\rm proj}$, defined as the Euclidean distance to the cluster center. For a full description of the calculation of these mock observables, see the Appendix.

The cylindrical cuts are characterized by three fixed parameters, $R_{\rm aperture}$, $v_{\rm cut}$, and $N_{\rm min}$, which correspond to the cylinder's radial aperture in the $x_{\rm proj}$–$y_{\rm proj}$ plane, the half-length along the $v_{\rm los}$-axis, and the minimum cluster richness, respectively. Galaxy subhalos that fall between the bounds $R_{\rm proj} \leqslant R_{\rm aperture}$ and $|v_{\rm los}| \leqslant v_{\rm cut}$ are included in the mock observation of the host cluster, whether or not they are truly gravitationally bound to the system. Following the cylindrical cut, cluster candidates that have less than $N_{\rm min}$ galaxy subhalos are discarded from our sample. In this analysis, the cylinder parameters are chosen to be $R_{\rm aperture} = 1.6\,h^{-1}$ Mpc and $v_{\rm cut} = 2200$ km s^{-1}, corresponding to the typical radius and $2\sigma_v$ of a $10^{15}\,h^{-1}\,M_\odot$ massive halo. We use a richness cut of $N_{\rm min} = 10$. The cylindrical cut procedure is symmetric for azimuthal rotations about the LOS axis. This symmetry is taken into account when augmenting training data in Section 3.3. An example contaminated mock observation is shown in Figure 1.

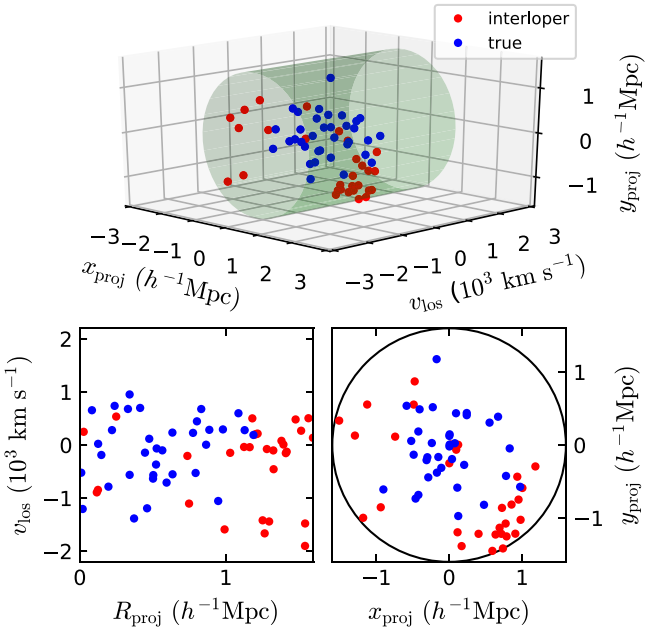


Figure 1. An example contaminated cluster member distribution showing both true members (blue) and interlopers (red), with a total $\log_{10}[M_{200c}\ (h^{-1}\,M_\odot)] = 14.27$. True members correspond to galaxies that fall within the cluster's MDPL2 Rockstar FOF group. It is important to note that in our model (and in reality), we cannot distinguish between true members and interlopers. Top: cluster members extracted from a cylinder cut in the mock light cone. Bottom left: traditional $v_{\rm los}$ vs. $R_{\rm proj}$ showing cluster member distribution in projected phase space. Bottom right: projected plane-of-sky perspective.

2.3. Train/Test Split

We build a training set of mock cluster observations with a flat mock cluster mass function across all masses so as not to introduce a bias in mass predictions via an imposed prior on cluster abundance. Due to a scarcity of simulated halos above $M_{200c} \geqslant 10^{14.6}\ h^{-1}\,M_\odot$ (Figure 2), we create an evenly distributed training set by upsampling clusters at high masses and downsampling clusters at low masses. We execute our sampling procedure by generating new mock cluster projections from various LOS. First, we choose a number density of clusters which provides a sufficient number of cluster examples to effectively train our model without overfitting. Here, we choose a flat training cluster number density of $10^{-5.2}\ h^3\,{\rm Mpc}^{-3}\,{\rm dex}^{-1}$. Next, each cluster in our catalog is evaluated at three orthogonal LOS projections. Then, clusters in an abundant mass region are downsampled to our chosen cluster number density. Clusters in scarce mass regions are upsampled by taking additional LOS projections. Any additional LOS projections aside from the initial three are distributed with roughly even spacing on the unit sphere, according to a Fibonacci lattice (González 2010). The average number of LOS samplings per cluster for the full training catalog is 2.91. The training set mock cluster mass function is shown in Figure 2.

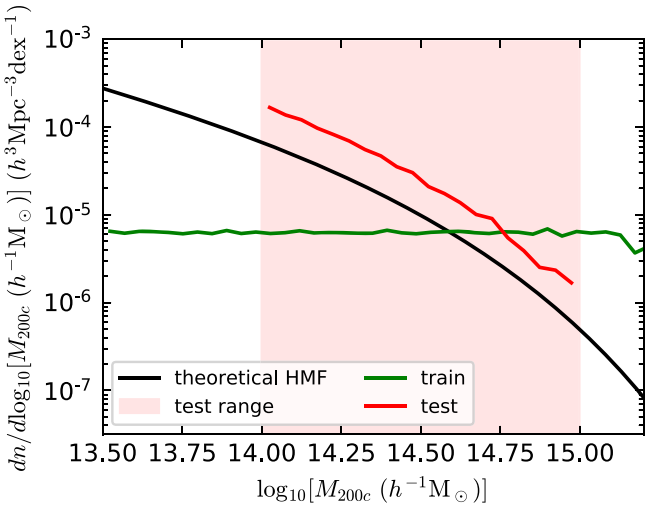


Figure 2. Mock cluster mass function for training and test samples in the contaminated catalog relative to the theoretical HMF for MDPL2 cosmology. The test sample cluster mass function is equivalent to three times the theoretical HMF, for the three orthogonal LOS perspectives taken of each cluster. The training sample has a flat cluster mass function, to eliminate selection bias during training. Note, to create the flat mass function training set, clusters are downsampled at low masses and upsampled at high masses.

To evaluate our model under realistic measurement conditions, the test catalog cluster mass function is weighted to follow the theoretical HMF, i.e., the exact distribution of cluster masses that is present in our base simulation. The test set solely consists of three orthogonal LOS projections of each cluster. The testing mass range is restricted to $14 \leqslant \log_{10}[M_{200c}\ (h^{-1}\,M_{\odot})] \leqslant 15$ so as to avoid unreliable mean-reversion edge effects. The test set cluster mass function is shown in Figure 2.

2.4. Summary

The data set generation can be summarized with the following procedure:

1. MDPL2 and UniverseMachine provide position, velocity, and mass information for dark matter halos and subhalos at a chosen redshift $z = 0.117$. Host halos are considered to be cluster candidates if $M_{200c} \geqslant 10^{13.5}\ h^{-1}\,M_{\odot}$. Subhalos represent galaxies if they have a mass accretion of $M_{\rm acc} \geqslant 10^{11}\ h^{-1}\,M_{\odot}$. Cluster centers are assumed to be known and consistent with the host halo's position and velocity.
2. The simulation box is padded along each side to account for periodic boundaries. The padding width used in this analysis is overestimated at 112 h^{-1} Mpc.
3. Each cluster candidate's center is placed at $z = 0.117$, and an observer is placed at $z = 0$. The quantities $x_{\rm proj}$, $y_{\rm proj}$, and $v_{\rm los}$ are calculated for each cluster-member pair using the procedure described in Equations (15)–(22).
4. For the contaminated catalog, the mock observations consist of all galaxies within a cylinder cut of fixed radius $R_{\rm aperture}$ and length $2v_{\rm cut}$ centered at each cluster center in $\{x_{\rm proj}, y_{\rm proj}, v_{\rm los}\}$ space. For the pure catalog, all galaxies within the virial radius of a given cluster are included in its mock observation. For both the pure and contaminated catalogs, all cluster candidates below a minimum richness of 10 galaxies are discarded.
5. Training and test sets are created from the mock observation catalogs. The training set is constructed with a flat cluster mass function in an effort to mitigate prediction bias. The test set follows the simulation's theoretical HMF. We sample the catalog to match these cluster mass function trends accordingly. Upsampling involves repeating steps 3–4 from multiple projected LOSs.

3. Method

In this section, we present the deep learning methodology used to infer masses from cluster member galaxy dynamics. Our first model, $\rm CNN_{1D}$, uses the distribution of galaxy LOS velocities $\{v_{\rm los}\}$ to infer cluster mass. This model is then extended to $\rm CNN_{2D}$ by incorporating the projected plane-of-sky radius $R_{\rm proj}$ as an additional input dimension. In Section 3.1, we discuss how catalog data are preprocessed to serve as input to our deep learning architectures. We then describe our ML model in Section 3.2 and our training/evaluation procedures in Section 3.3.

3.1. Preprocessing

For each cluster, we map the distribution of member galaxies in projected phase space using a KDE. KDEs effectively smooth the distribution of discrete galaxy positions into a continuous PDF according to some prescribed length scale (bandwidth). This smoothed distribution is then sampled at regular intervals to form a pixelated mapping over the cylinder cut. PDF mappings generated with KDEs can sufficiently encapsulate features of the underlying member distribution while remaining relatively invariant to variations in the sampling rate. These mappings serve as direct input to our ML model.

3.1.1. Kernel Density Estimation

Given a univariate, independent, and identically distributed sample $\{x_i\}$ of length n drawn from some unknown distribution with density f, we can derive an expression for the estimated PDF $\hat{f}$ using a KDE,

$$\hat{f}\,(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right), \tag{1}$$

where K is a kernel function and h is the kernel bandwidth. The kernel function is nonnegative, integrates to unity, and is often chosen to be the standard normal distribution (Gaussian KDE). The kernel bandwidth is a smoothing parameter, which we will assign to scale linearly with the sample standard deviation $h = h_0\hat{\sigma}_x$.

Product kernel estimators are used to estimate multivariate PDFs. Product kernels use the same univariate kernel in each dimension, but with a possibly different smoothing bandwidth for each dimension. Given a multivariate, independent, and identically distributed sample $\{(x_i^0, \ldots, x_i^d)\}$ of length n and dimension d drawn from some unknown distribution with density f, a product kernel $\hat{f}$ can be written as

$$\hat{f}\,(\boldsymbol{x}) = \frac{1}{nh_1 \cdots h_d}\sum_{i=1}^{n}\left\{\prod_{j=1}^{d} K\left(\frac{x^j - x_i^j}{h_j}\right)\right\}, \tag{2}$$

where K is the kernel function (like the standard normal), $\boldsymbol{x} = (x^0, \ldots, x^d)$ is the evaluation point, and $\{h_i\}$ is the set of smoothing bandwidths. The smoothing bandwidths scale with the sample's standard deviation along their respective dimension $h_i = h_0\sigma_i$. The bandwidth scaling factor h_0 is a constant coefficient applied to all smoothing bandwidths. For a comprehensive discussion of univariate and product kernels, see Scott & Houston (2015, chap. 6).

3.1.2. Model Input

The $\rm CNN_{1D}$ model learns on cluster $\{v_{\rm los}\}$ distributions estimated using a univariate Gaussian KDE. We know from the M–σ relation that the shape of the $\{v_{\rm los}\}$ distribution contains information regarding the cluster mass. The set of cluster PDFs are generated at a fixed bandwidth scaling factor of $h_0 = 0.25$. We sample each $\{v_{\rm los}\}$ PDF at 48 evenly spaced points across the cylinder cut, producing a fixed-length vector describing the distribution. Normalizing this vector to unity produces our input for the $\rm CNN_{1D}$ model. Examples of the normalized $\{v_{\rm los}\}$ PDF vector are shown in Figure 3.

The $\rm CNN_{2D}$ model uses a bivariate product kernel estimator to form a joint $\{R_{\rm proj}, v_{\rm los}\}$ distribution. Similar to the M–σ relation, the $R_{\rm proj}$ distribution is descriptive of cluster mass (Ntampaka et al. 2016; Armitage et al. 2019a). In addition, the

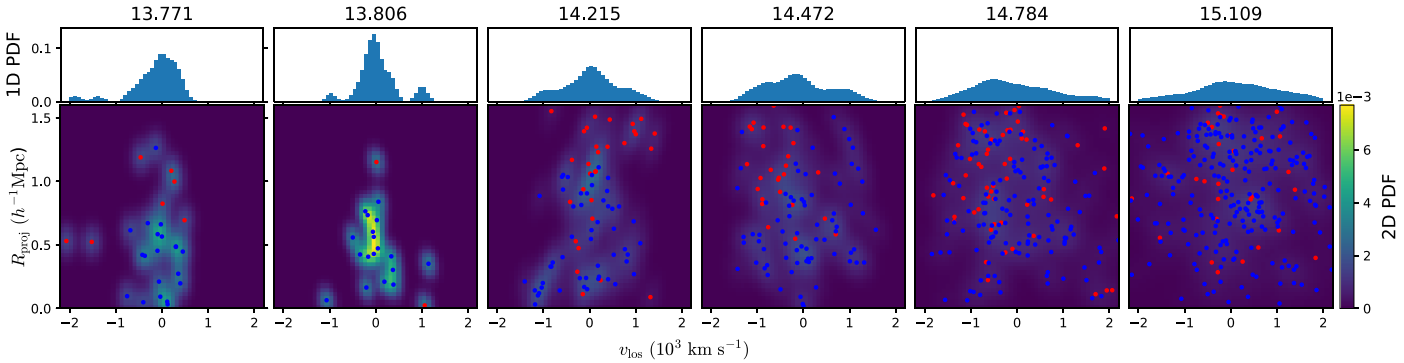


Figure 3. Six example contaminated clusters randomly selected from evenly spaced log mass bins. Each column shows the 1D and 2D normalized PDFs generated from each cluster's member distribution using a Gaussian KDE. The title of each plot gives the true $\log_{10}[M_{200c}\ (h^{-1}\,M_{\odot})]$ value assigned to each cluster. The populations of true members (blue) and interlopers (red) are superimposed on the 2D PDFs, though it is important to note that this information is not passed to the CNN models. The 1D and 2D PDFs shown here are estimated using Gaussian KDEs with a bandwidth factor of 0.25. The 1D PDFs are equivalent to the 2D PDFs marginalized over $R_{\rm proj}$.

joint $\{R_{\rm proj}, v_{\rm los}\}$ shows clustering behavior of true member and interloper populations (Figures 1 and 3). We create a bivariate product kernel estimator for each clusters $\{R_{\rm proj}, v_{\rm los}\}$ distribution with a fixed bandwidth scaling factor $h_0 = 0.25$. We sample the PDF at 48×48 points regularly spaced across the $\{R_{\rm proj}, v_{\rm los}\}$ phase space. This produces a 48×48 array which we then normalize to unity. This array serves as input to the $\rm CNN_{2D}$ model and is demonstrated in Figure 3.

3.2. Models

The foundations of our mass estimators are CNNs. CNNs are a class of feed-forward deep neural networks (DNNs) which have garnered considerable attention recently for their applications in computer vision. CNNs have convolutional layers that learn patterns on subsets of data. The objective of this approach is to allow convolutional layers to learn and correct for observational constraints such as interlopers and cluster mergers.

3.2.1. Deep Learning

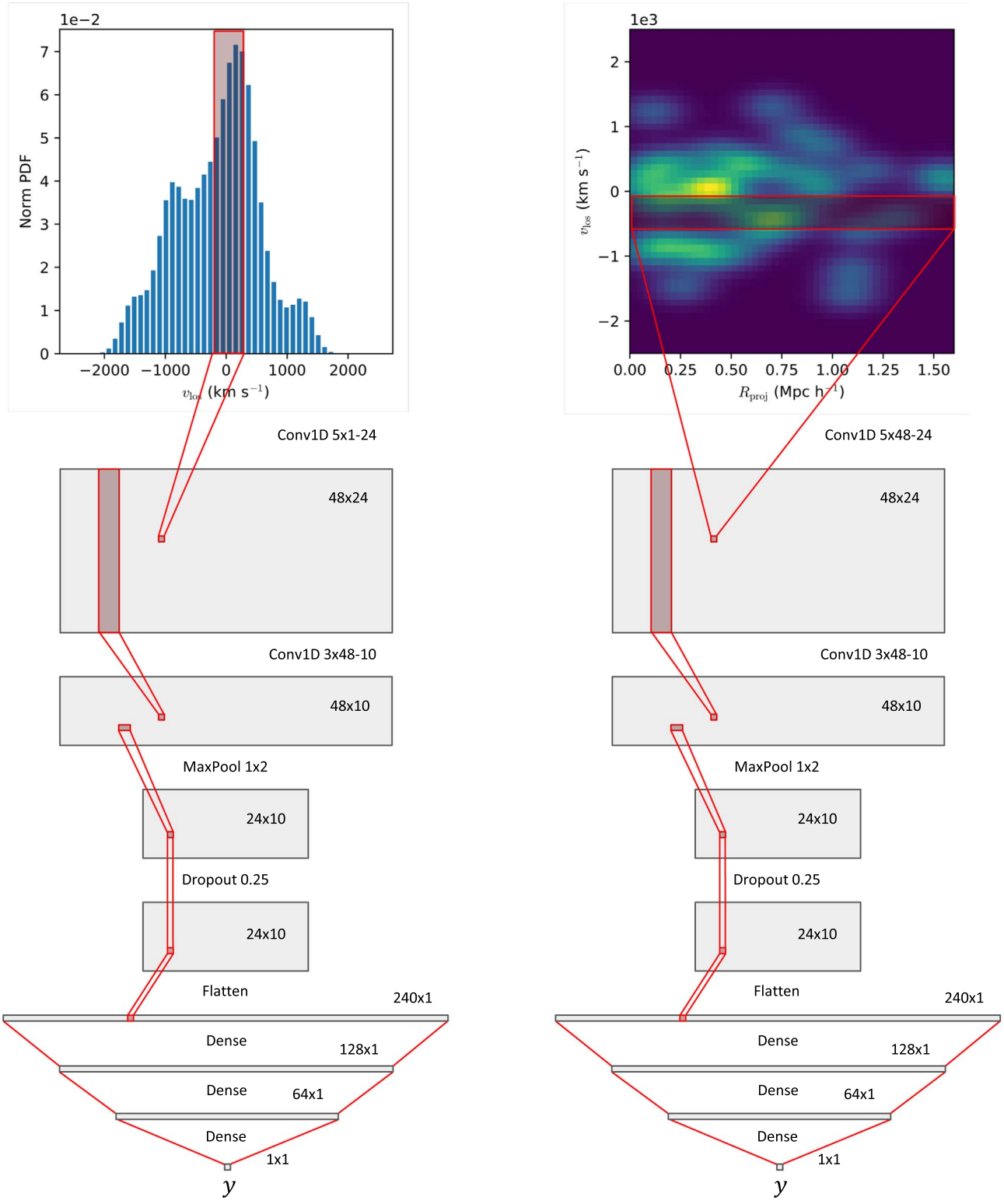
DNNs are a group of supervised ML methods which encompass CNNs. DNNs have been shown to be able to learn complex, nonlinear relationships between fixed-length input and output arrays (LeCun et al. 2015) and have been met with a plethora of applications in observational cosmology (e.g., Dieleman et al. 2015; Hoyle 2016; Lanusse et al. 2018; Ntampaka et al. 2018). Within a DNN, input and output are related through a sequence of connected neuron layers. The neurons of each layer are linked to neurons of adjacent layers through a multitude of directed, weighted connections. During evaluation, each neuron produces a numerical output by taking a linear combination of values from its incoming connections and subjecting the result to a nonlinear activation function. In the simplest case of a feed-forward DNN, the neuron layers are evaluated in sequence, passing information from layer to layer without recurrence. Stated in tensor notation, the output $\boldsymbol{h}^{(l)}$ of the lth layer of a feed-forward neural network can be described by the following:

$$\boldsymbol{h}^{(l)} = f(\boldsymbol{W}^{(l)} \cdot \boldsymbol{h}^{(l-1)} + \boldsymbol{b}^{(l)}), \tag{3}$$

where $\boldsymbol{W}^{(l)}$ is a matrix of connection weights, $\boldsymbol{b}^{(l)}$ is a vector of additive biases, and f is the element-wise nonlinear activation function (e.g., sigmoid). The set $\{\boldsymbol{W}, \boldsymbol{b}\}$ constitutes the model parameters for the DNN. We consider the layers $1 \leqslant l \leqslant L-1$ as part of the DNN architecture, whereas $\boldsymbol{h}^{(0)}$ is the input vector and $\boldsymbol{h}^{(L)}$ is the final, output vector. Neural layers of this form (Equation (3)), wherein every neuron is connected to every neuron of the previous layer (i.e., $\boldsymbol{W}^{(l)}$ is dense), are often referred to as dense or fully connected layers.

DNNs are trained to relate input and output by optimizing connection weights between neuron layers. During model training, we evaluate the network on a set of inputs for which the true, desired output is known. We then calculate the model's prediction error by comparing the model's output to the true values using a loss function. We seek to minimize this prediction loss by exploring the parameter space of all connection weights using an iterative parameter optimization algorithm such as stochastic gradient descent (SGD; Robbins & Monro 1951). SGD repeats its update procedure for many small, randomly selected sets of training data until the loss function stops decreasing. At this point, one might evaluate the performance of the now-optimized network on a set of independent test data.

CNNs (LeCun et al. 1998) are a subset of DNNs, which mainly benefit from, and are named for, their use of convolutional layers. Unlike dense connections, convolutional connections restrict neurons in one layer to receive information only from neurons within a small neighborhood of the previous layer, called a receptive field. This local receptive field method allows neurons to extract simple features from subsets of the input layer, the information from which can be combined to form higher-order features in subsequent layers. The input receptive fields of adjacent neurons within a convolutional layer often overlap, forming a contiguous transformation from input to output, akin to a convolution. The filter or feature extractor, the set of weights and biases that connect the small region of inputs to the output node, is shared across the entire input layer. This allows the same feature to be detected in different receptive fields across the input while also reducing the complexity of the connection. The output of a filter applied to all regions of an input is called a feature map. A full convolutional layer often consists of multiple feature maps, each with different filters. A physical depiction of convolutional layers and their filters can be seen in Figure 4.



(a) $\mathrm{CNN_{1D}}$ architecture (b) $\mathrm{CNN_{2D}}$ architecture

Figure 4. CNN architecture for each model. The architectures for each case are identical except for the input array and the first convolutional layers. The output of each model y (Equation (4)) varies linearly with the predicted logarithmic cluster mass and is restricted to the range $y \in [0, 1]$. Each layer is subject to a ReLU activation function, and the weight vectors are constrained to a maximum L2 norm of 3.

Convolutional layers within a CNN are often followed by a pooling layer. Pooling layers perform a downsampling operation intended to reduce the dimensionality of the convoluted feature maps. The downsampling operation functions in a similar manner to the convolutional filters, in that they execute on local receptive fields across the input. A

common downsampling operation is max pooling, in which only the maximum activation from the local receptive field is passed to the next layer.

CNNs, and DNNs in general, use dropout layers as a type of stochastic regularization to avoid overfitting. Dropout layers randomly set some fraction of neurons from the previous layers equal to 0 during training. This forces the network to learn feature relationships through multiple neuron paths, reducing training time and preventing overfitting.

Typical simple CNN architectures consist of alternating convolutional and pooling layers followed by several dense layers. Each successive convolutional layer produces coarser, higher-order feature maps of the original input. The final dense layers relate the highest-order features to an output vector. CNNs use the same training procedure as discussed for DNNs.

3.2.2. Architecture

The CNN models used in this analysis (Figure 4) were designed to incorporate layering patterns common to image-recognition applications while minimizing architectural complexity. Both models use two convolutional layers followed by a max pooling layer, a dropout layer, and three dense layers. The inputs to the models are generated from KDEs as discussed in Section 3.1.2. Each model outputs a single variable y, which ranges from $0 \leqslant y \leqslant 1$ and relates linearly to a mass prediction $\log_{10}[\hat{M}_{\rm pred}\ (h^{-1}\,M_{\odot})]$,

$$\log_{10}[\hat{M}_{\rm pred}] = \log_{10}[M_{\rm min}] + y\log_{10}\left[\frac{M_{\rm max}}{M_{\rm min}}\right], \qquad (4)$$

where $M_{\rm min}$ and $M_{\rm max}$ are the minimum and maximum values for M_{200c} in our sample. All masses are expressed in units of $h^{-1}\,M_{\odot}$.

The convolutional and dense layers in both architectures use a kernel normalization constraint and a rectified linear unit (ReLU) activation function. The kernel constraint normalizes the weighting vector for the input of a given neuron to a constant. The ReLU function is given by the simple form $f(x) = \max(x,\ 0)$. The ReLU activation function has been shown to not saturate as much as conventional sigmoid functions (Nair & Hinton 2010).

The architectures for $\rm CNN_{1D}$ and $\rm CNN_{2D}$ are nearly identical, with an exception made for the first convolutional layer. In the first layer, both models use 1D convolution filters of width 5, which pass over receptive fields along the $v_{\rm los}$ axis with a stride of 1. The difference between these architectures is that the $\rm CNN_{1D}$ model's filters are of shape 5×1 while the $\rm CNN_{2D}$ model's filters are of shape 5×48. This is done to account for the difference in input shape between the two models. In the first neural layer, both models use 24 filters to create 24 feature maps of length 48. As a result, the outputs of the initial convolutional layers of both $\rm CNN_{1D}$ and $\rm CNN_{2D}$ are of shape 48×24.

3.3. Training and Evaluation

We train each model as a regression over the single output variable y (Equation (4)) using a mean squared error loss function. This is equivalent to minimizing the sum of squared residuals of the output variable y, i.e., $\sum(y_{\rm pred} - y_{\rm true})^2$, over the space of our model parameters. For our optimization procedure, we use the Adam protocol (Kingma & Ba 2014), a variant of SGD that accounts for momentum and adaptive learning rates in a straightforward, computationally efficient manner. We parameterize the Adam optimizer with a learning rate of 10^{-3} and a decay rate of 10^{-6}. We use a batch size of 100 samples and achieve loss convergence within 20 epochs.

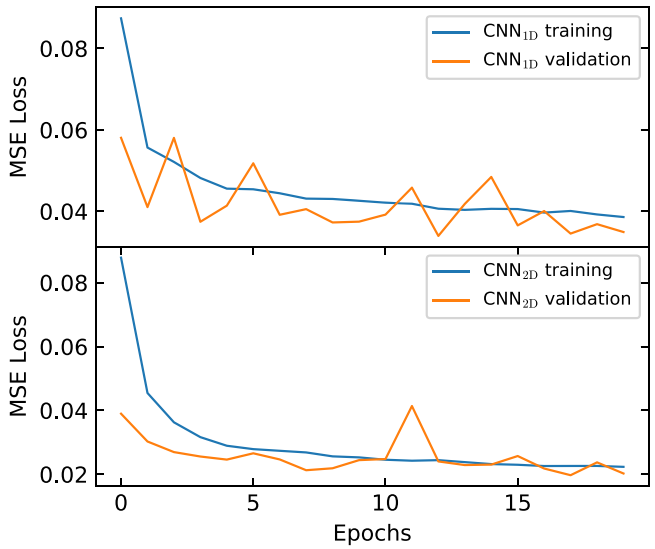


Figure 5. Evolution of mean squared error (MSE) loss for a single fold during $\rm CNN_{1D}$ and $\rm CNN_{2D}$ training. Training progression exhibits gradual improvement in both the training and validation sets. The validation loss appears less than the training loss due to the introduction of dropout layers.

We use a 10-fold cross-validation scheme to evaluate our model. For a given fold, we train on $9/10$ of the cluster candidates in our catalog and test on the remaining, independent $1/10$. This process cycles for 10 folds until predictions have been made for the entire test set. Cluster candidates are grouped along with their rotated LOS duplicates in the training-test split, such that we are never training and testing on the same cluster from different LOSs. This ensures independence of training and testing data for each fold. On average, there are $\sim$10,000 training and $\sim$7000 test cluster candidates for a given fold.

A validation set is constructed from a disjoint 10% random sampling of the independent test data. Figure 5 shows training and validation loss curves for a single fold during the 20 epoch training procedure. The loss curves from both the $\rm CNN_{1D}$ and $\rm CNN_{2D}$ show gradual improvement throughout training evolution, indicating that neither model is overfitting.

The CNN models and training procedure are implemented using the Keras[7] library with a Tensorflow[8] backend. Each ML analysis was run on two Intel Haswell (E5-2695 v3) CPU nodes with 14 cores and 128 GB of total RAM. The full 10-fold training procedure is executed to convergence in $\sim$10 minutes for both CNN architectures. The KDE generation and sampling process takes $\sim$73 μs and $\sim$410 μs per input, for $\rm CNN_{1D}$ and $\rm CNN_{2D}$ respectively. Once the models are trained and the KDEs are sampled, the evaluation time of either CNN neural architecture lasts $\sim$44 μs per input.

4. Comparative Methods

In our comparative analysis, we discuss the performance of $\rm CNN_{1D}$ and $\rm CNN_{2D}$ relative to other dynamical mass

[7] https://keras.io/
[8] https://www.tensorflow.org/

estimation techniques, namely the classical M–σ and SDM (Ntampaka et al. 2015, 2016). Each of these models are evaluated in the context of the mock catalog described in Section 2.

4.1. M–σ

The M–σ scaling relation infers cluster mass from a single summary statistic, the galaxy velocity dispersion σ_v. If we assume clusters to be stable, spherically symmetric, and purely evolving with gravity, we can derive the classical form of the M–σ relation from the kinetic–potential energy equivalence described in the virial theorem. Stated with the appropriate normalization for galaxy clusters, the M–σ relation is as follows:

$$\sigma_v = \sigma_{v,15}\left[\frac{h(z)M_{200c}}{10^{15}\,M_\odot}\right]^{\alpha} \tag{5}$$

where M_{200c} is our cluster mass definition of a spherical region of density $200\rho_c$, $\sigma_{v,15}$ is a scaling factor parameterizing the velocity dispersion of a galaxy cluster of $M_{200c} = 10^{15}h^{-1}\,M_\odot$, $h(z)$ is the dimensionless Hubble parameter, and α is the power-law scaling parameter. Assuming spherical symmetry, the velocity dispersion σ_v can be conveniently taken to be the standard deviation of galaxy velocities projected along a single LOS. The parameter α captures information about the spatial distribution of mass in the spherical cluster and is generally fit with simulation (Evrard et al. 2008).

We perform an M–σ analysis on both the contaminated catalog described in Section 2.2 and a comparative, idealized pure catalog. Mock observations in the pure catalog are designed to ignore all member selection effects by assuming pure and complete cluster membership. Cluster member samples are constructed from all galaxies which are associated with the cluster's MDPL2 Rockstar FOF group. From this pure member sample, mock observables are calculated in the familiar manner (see the Appendix). The pure cluster catalog is designed to mimic data products of optimal interloper removal strategies, producing a lower limit on M–σ measurement scatter for modern dynamical mass estimation techniques. Conversely, the cylindrical cuts taken in the contaminated catalog are decidedly simpler than modern methods and thereby produce an upper limit on M–σ scatter.

We find best-fit parameters $\sigma_{v,15}$ and α for both the pure and contaminated mock catalogs. We use the unbiased standard deviation (Equation (6)) to estimate velocity dispersions for each cluster sample.

$$\sigma_v = \sqrt{\frac{1}{N_{\rm gal}-1}\sum_{i=1}^{N_{\rm gal}}(v_{\rm los,i}-\bar{v}_{\rm los})^2}\,, \tag{6}$$

where $N_{\rm gal}$ is the number of galaxies in a given cluster sample, $v_{\rm los,i}$ is the LOS velocity of the ith cluster, and $\bar{v}_{\rm los}$ is the average LOS velocity for the cluster. We use an ordinary least-squares linear regression model to fit the power law in log-space: $\log_{10}(\sigma_v) = A\log_{10}(M_{200c}) + B$. As demonstrated in Figure 6, the contaminated cluster's M–σ relationship exhibits a departure from log-linear dependence at low masses, due primarily to the saturation of mock observations with unbound galaxies. This is a direct result of the fixed-size cylindrical cuts and was explored in detail in Ntampaka et al. (2016). When fitting the M–σ, we choose to take a linear regression above a mass cut of $10^{14.5}\ h^{-1}\,M_\odot$ and subsequently extrapolate to lower masses. This mass cut is implemented for both the pure and contaminated M–σ regressions. In addition, both regressions use the flat cluster mass function training set described in Section 2.3.

The M–σ distribution for pure and contaminated catalogs is shown in Figure 6. Best-fit parameters are calculated for $\sigma_{v,15}$ and α, and are tabulated in Table 1. We evaluate the lognormal scatter by taking the standard deviation of the residual δ for clusters above the mass cut,

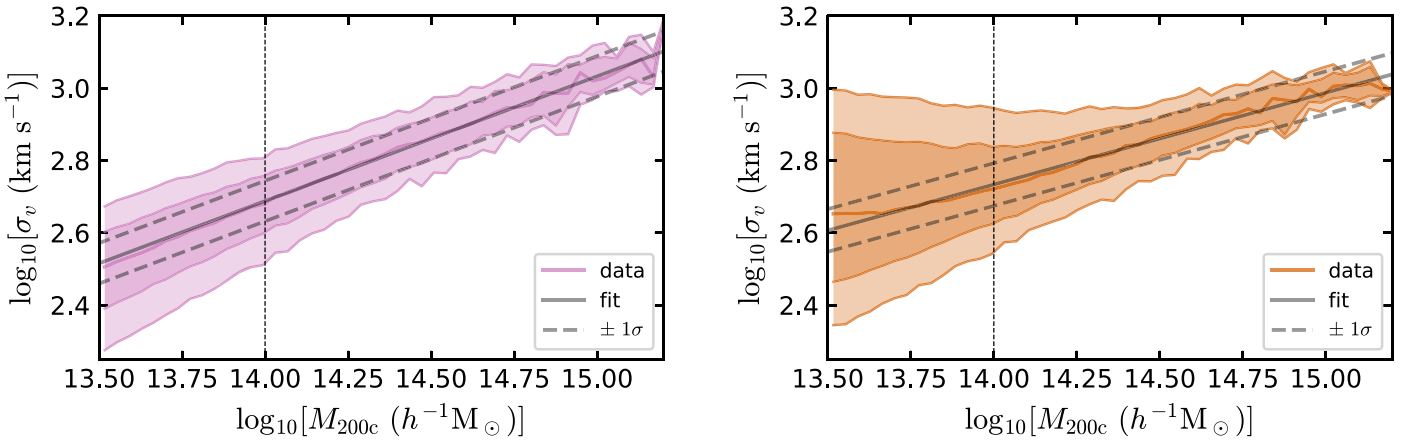
$$\delta = \log_{10}\left[\frac{\sigma_{v,\rm pred}}{\sigma_{v,\rm true}}\right], \tag{7}$$

where $\sigma_{v,\rm true}$ is the true velocity dispersion for a given cluster and $\sigma_{v,\rm pred}$ is its predicted velocity dispersion from its true mass and best-fit parameters $\sigma_{v,15}$ and α. The parameter values presented in Table 1 are representative of values previously derived from simulation (Evrard et al. 2008), but also exhibit variation due to differences in mock observation strategy.

The M–σ predictions for both the pure and contaminated catalogs exhibit significant scatter. In the pure case, this scatter can be attributed to physical effects which distort cluster shape or mass distribution. Clusters are highly complex systems in which assumptions of gravitational equilibrium or spherical symmetry are unreliable. In practice, features such as dynamical substructure (Old et al. 2018), halo environment (White et al. 2010), cluster triaxiality (Svensmark et al. 2015), and mergers (Ribeiro et al. 2011) act to increase the scatter of M–σ predictions. In the contaminated case, the prediction scatter is higher than the pure catalog due to the introduction of selection effects (Wojtak et al. 2018). Realistic cluster observations may be incomplete or otherwise contaminated by interloper galaxies. In modern applications of the M–σ, complex membership modeling and interloper removal schemes may be applied to reduce the impact of selection effects (e.g., Wojtak et al. 2007; Mamon et al. 2013; Farahi et al. 2016, 2018; Abdullah et al. 2018), ideally producing predictions equivalent to our pure catalog. Our pure and contaminated predictions therefore define lower and upper bounds, respectively, of the scatter apparent in real M–σ predictions.

4.2. Support Distribution Machines

SDMs (Sutherland et al. 2012) are a class of ML algorithms that perform scalar regression over a set of probability distributions. SDMs effectively function as an extension of kernel support vector machine (SVM; Schölkopf & Smola 2002) regression, where nonlinear input is mapped to a space of linear features via some kernel function. Each input to SDM is a variable-length set of i.i.d. samples chosen from an underlying probability distribution. The output is some continuous, scalar value quantifying something about the base probability distribution. SDMs are nonparametric and trained transductively, meaning the complexity of the model is directly proportional to the size of the data set (train + test). The first application of SDMs to dynamical mass measurements was made in Ntampaka et al. (2015, 2016), where SDMs were used to directly infer the cluster mass from lists of galaxy velocities and positions. The SDM approach was effective in reducing M–σ prediction scatter by a



(a) Pure M-σ (b) Contaminated M-σ

Figure 6. M–σ relationship for (a) pure and (b) contaminated mock observation cluster catalogs derived from MDPL2 data. Each distribution is plotted at its median (solid line), 16th–84th percentile range (dark region), and 3rd–97th percentile range (light region). The log-linear regression lines are shown along with their $\pm 1\sigma$ lognormal scatter. The dotted black line at $M_{200c} = 10^{14.5}\,h^{-1}\,M_\odot$ signifies the lower-bound mass cut used to perform the log-linear regression. Selection effects in the contaminated catalog introduce significant scatter and bias at low masses.

Table 1
Best-fit Parameters for Log-linear Regression of M–σ in the Pure and Contaminated Catalogs

Catalog	$\sigma_{v,15}$ (km s^{-1})	α	Scatter (dex)
Pure	1078	0.345	0.056
Contaminated	971	0.254	0.059

Note. Parameters are defined in the formalization of the M–σ given in Equation (5). The lognormal scatter is defined as the standard deviation of prediction residuals for clusters above the mass cut, $M_{200c} \geqslant 10^{14.5}\,h^{-1}\,M_\odot$.

factor of 2. Here, we evaluate SDM performance in the context of our catalog to serve as a baseline with which to compare our ML model.

Replicating our treatment of CNN models, we train SDMs on two types of cluster descriptions, the member $\{v_{\rm los}\}$ distribution and the joint member $\{R_{\rm proj},\ v_{\rm los}\}$ distribution. We will appropriately refer to these as $\mathrm{SDM_{1D}}$ and $\mathrm{SDM_{2D}}$, respectively. Each individual input to the SDM is a list of univariate or bivariate galaxy properties (velocities and/or radial positions). The length of each input list is variable and equal to the cluster richness. In this application of SDMs, we assume this list of galaxies is representative of some underlying probability distribution which varies with cluster mass.

Our implementation of SDM mirrors that of Ntampaka et al. (2016). The kernel function employed in our SDM model is a Kullback–Leibler divergence, estimated using the k-nearest-neighbor method (Wang et al. 2009) with $k = 3$. We use three-fold cross-validation to find optimal values for SDM parameters C and σ, the loss function parameter and Gaussian kernel parameter, respectively. We evaluate the SDM models with ten-fold cross-validation, and the training and test sets described in Section 2.3.

Analysis of each SDM model was run on two Intel Haswell (E5-2695 v3) CPU nodes with 14 cores each and 128 GB of total RAM. Using the mock catalog described in Section 2, the full 10-fold transductive training and evaluation procedure executed in ∼6 hr for each SDM model.

5. Results

The results presented in this section analyze the performance of our CNN models when evaluated on a catalog of mock cluster observations (Section 2). Model performance is quantified in terms of predictive scatter, bias, lognormality, robustness, and application time. We describe these metrics in the context of observational studies and discuss their implications in precision cosmology. Using these metrics, we perform comparative analyses with respect to the dynamical mass estimators described in Section 4. The complete list of investigated models presented in this section is summarized in Table 2. We find that the CNN models produce more accurate and robust mass estimates than all other investigated methods, with considerably shorter implementation times than SDM.

5.1. Predictive Performance

Figure 7 shows the multifold predicted-versus-true mass distribution of the $\mathrm{CNN_{1D}}$ and $\mathrm{CNN_{2D}}$ models when performing inference on the test data set (Section 2.3). For each model, we describe the distribution of mass predictions via the logarithmic residual ϵ, defined as

$$\epsilon = \log_{10}\left[\frac{M_{\rm pred}}{M_{\rm true}}\right] \tag{8}$$

for a cluster of mass $M_{\rm true}$ whose predicted mass is $M_{\rm pred}$. This metric is commonly employed in other observational studies (e.g., Armitage et al. 2019a, 2019b; Calderon & Berlind 2019) and conveniently scales linearly with our model output y (Equation (4)). The mass definition used in this analysis is $M_{\rm true} = M_{200c}$. We further characterize model predictions by calculating cumulative statistics of the ϵ distribution, namely the median ($\tilde{\epsilon}$), 16th–84th percentile range ($\Delta\epsilon$), and the standard deviation scatter (σ_ϵ). The values of these statistics for $\mathrm{CNN_{1D}}$ and $\mathrm{CNN_{2D}}$ are tabulated in Table 2. Note that these cumulative statistics are constructed from the test catalog and marginalized over true mass and are thereby weighted by the shape of the test catalog cluster mass function (Figure 2).

Table 2
Summary of Investigated Models

Model	This Paper?	Data	Color	Catalog	$\tilde{\epsilon} \pm \Delta\epsilon$ [a]	σ_ϵ [b]	γ [b]	κ [b]
CNN_{1D}	✓	$\{v_{los}\}$	green	Contaminated	$-0.003^{+0.173}_{-0.163}$	0.174	0.419	0.826
CNN_{2D}	✓	$\{R_{proj}, v_{los}\}$	blue	Contaminated	$-0.003^{+0.119}_{-0.125}$	0.132	0.221	1.600
M–σ_{pure}		$\{v_{los}\}$	violet	Pure	$+0.006^{+0.179}_{-0.195}$	0.193	−0.262	0.417
M–σ_{contam}		$\{v_{los}\}$	orange	Contaminated	$-0.016^{+0.300}_{-0.290}$	0.316	0.225	0.647
SDM_{1D} [c]		$\{v_{los}\}$	yellow	Contaminated	$-0.039^{+0.229}_{-0.187}$	0.226	0.646	1.183
SDM_{2D} [c]		$\{R_{proj}, v_{los}\}$	pink	Contaminated	$-0.018^{+0.149}_{-0.148}$	0.159	0.309	1.459

Notes. In addition to the CNN models presented in this paper, we include other comparative dynamical mass estimates, including the traditional M–σ and a modern ML approach (SDM; Ntampaka et al. 2016). We analyze the M–σ method under both a pure and contaminated catalog in order to provide lower and upper bounds on the scatter of general interloper removal strategies. We include several cumulative statistics describing the error (Section 5.1) and lognormality (Section 5.2) of each model's mass predictions.
[a] Residual median and 16th–84th percentile range (dex).
[b] Residual standard deviation scatter (dex), skewness, and excess kurtosis, respectively.
[c] Ntampaka et al. (2016).

As seen in Figure 7, the CNN model predictions exhibit low scatter and bias across the test mass range. The residual scatter σ_ϵ for CNN_{2D} predictions, 0.132 dex (≃30%), is considerably lower than that for CNN_{1D} predictions, 0.174 dex (≃40%), indicating that the supplementary information about underlying galaxy distributions provided by R_{proj} reduces scatter by 24% under the CNN framework. Each model's ϵ distribution shows a marginal trend toward higher scatter at low true mass, which we attribute to a reduction of true members and a saturation of interlopers in the fixed cylindrical membership cut.

Figure 8 plots the median and 16th–84th percentile range of prediction residuals as a function of true mass for each investigated model listed in Table 2. Each model is evaluated on the same contaminated mock catalog (Section 2) except for M–σ_{pure}, which is evaluated on a catalog with perfect membership selection (Section 4.1). The SDM and M–σ models serve as baselines for modern ML and interloper removal schemes, respectively. Cumulative statistics for these comparative methods are listed in Table 2. The prediction scatter measured for the M–σ and SDM methods is consistent with literature (Evrard et al. 2008; Ntampaka et al. 2016).

CNN models produce the equivalent or better predictive performance than either pure or contaminated M–σ measurements. The simple M–σ_{contam} model exhibits high bias and scatter, with exceptionally high deviation at low masses, resulting from interloper saturation. The ϵ distribution of CNN_{1D} is virtually equivalent to that of M–σ_{pure}, suggesting that CNN_{1D} is capable of achieving the same scatter as optimal interloper removal algorithms. Whereas M–σ_{pure} improves upon M–σ_{contam} by eliminating selection systematics, the prediction improvements made by CNN_{1D} likely stem from a mitigation of both selection and physical effects. CNN_{2D}'s low scatter and bias relative to the pure and contaminated M–σ can be attributed to its use of R_{proj} information. These results imply that the CNN models presented here may be preferable over modern M–σ-based interloper removal methods.

According to Table 2, the SDM_{1D} and SDM_{2D} models are effective in reducing prediction scatter to below that of M–σ_{pure}, but produce strong prediction biases. Both SDM models observe significant deviations in median prediction $\tilde{\epsilon}$ at various regions in the testing mass range. This is visible in Figure 8, where SDM_{1D} and SDM_{2D} underpredict medium- to high-mass clusters. This behavior may complicate applications in precision cosmology. The SDM biases measured here are consistent with results shown in Ntampaka et al. (2016). Aside from these biases, both SDM_{1D} and SDM_{2D} produce lower prediction scatter σ_ϵ than CNN_{1D}. This outcome is intuitive, considering that the KDE step in the CNN approach "smooths out" distribution information, which is potentially informative of cluster mass. However, CNN_{2D} is capable of overcoming this hindrance to produce a prediction scatter that is lower than both SDM models. The improved complexity of CNN_{2D} is therefore capable of capturing mass-dependent features of cluster dynamics at least as well as applications of SDM.

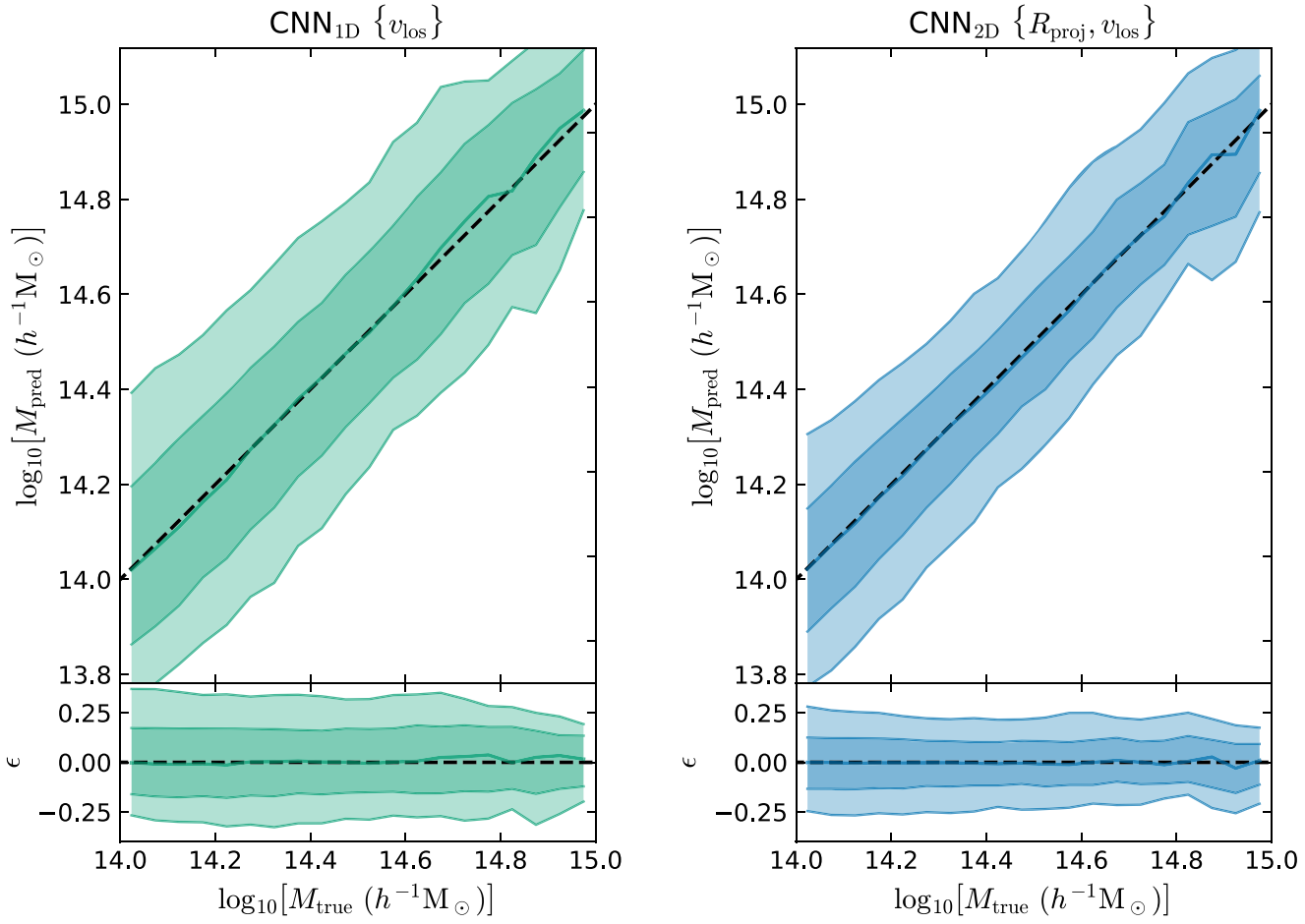
CNN_{1D} and CNN_{2D} reduce the prediction scatter σ_ϵ of the contaminated M–σ measurements by 45% and 58%. When compared to the idealized M–σ, these models show 10% and 32% improvement respectively. CNN_{2D} shows lower scatter than the best SDM model, producing 17% lower scatter than SDM_{2D}. The prediction improvements of CNN are comparable to those noted in other ML approaches (e.g., Armitage et al. 2019a; Calderon & Berlind 2019). This analysis suggests that CNN_{1D} and CNN_{2D} are capable of capturing mass-dependent input features and are effective models of cluster dynamics distributions. Under the assumptions made by the simulated catalog listed in Section 2.2, CNN_{2D} is the most accurate predictor of dynamical cluster masses among the above investigated models.

5.2. Lognormality

Mass estimators with non-Gaussian prediction likelihoods can introduce bias in cosmological analyses based on cluster counts (Erickson et al. 2011; Weinberg et al. 2013). We seek to characterize the non-Gaussianity of predictions made by CNN and other comparative methods in order to estimate their impact on halo abundance calculations. We follow a formalism introduced by Shaw et al. (2010) whereby we model the observable-mass relation for a fixed redshift by an Edgeworth expansion,

$$P(M_{pred}|M_{true}) \approx G(x) - \frac{\gamma}{6}\frac{d^3G}{dx^3} + \frac{\kappa}{24}\frac{d^4G}{dx^4}, \quad (9)$$

where $x = (\epsilon - \langle\epsilon\rangle)/\sigma_\epsilon$ is the normalized logarithmic residual, G is the standard normal distribution, and γ and κ are the skewness and excess kurtosis of the x distribution, respectively. For a power-law mass function $[dn/d\ln M] \propto M^{-\alpha}$, cluster



(a) $\mathrm{CNN_{1D}}$ Predictive Performance (b) $\mathrm{CNN_{2D}}$ Predictive Performance

Figure 7. Predicted-vs.-true mass distributions for our CNN models when predicting a realistic sample of mock cluster observations. Panel (a) shows the binned distribution of predicted masses and residuals (Equation (8)) using $\mathrm{CNN_{1D}}$. Each distribution is plotted at its median (solid line), 16th–84th percentile range (dark region), and 3rd–97th percentile range (light region). Panel (b) shows the same prediction and residual distributions for $\mathrm{CNN_{2D}}$. The mass definition applied in this analysis is $M_{\mathrm{true}} = M_{200c}$.

abundance measurements can be expressed as

$$\frac{dn}{d\ln M_{\mathrm{pred}}} \approx \left(\frac{dn}{d\ln M_{\mathrm{pred}}}\right)_0 \times \left[1 + \frac{\alpha^3\sigma^3}{6}\gamma + \frac{\alpha^4\sigma^4}{24}\kappa\right], \quad (10)$$

where M_{pred} is defined in terms of $h^{-1}\,M_\odot$, σ is the logarithmic prediction scatter (in percent), and $(dn/d\ln M_{\mathrm{pred}})_0$ is the abundance for a purely lognormal x distribution (Weinberg et al. 2013). From Equation (10), we can estimate the systematic uncertainty in cluster abundance measurements from the mass estimator cumulants σ, γ, and κ.

Table 2 lists the lognormality descriptors for each model's mass predictions. Figure 9 draws the PDF of the normalized residual distribution for each investigated model. From these statistics, we see that the PDF of each model's prediction residuals is roughly Gaussian. For a typical power-law mass distribution of slope $\alpha = 2$, the impact of non-Gaussian uncertainty on abundance measurements is $\leqslant$5% for all models except M–σ_{contam} and $\mathrm{SDM_{1D}}$. M–σ_{contam}'s high systematic uncertainty (23%) is primarily driven by its large scatter $\sigma_\epsilon = 0.316$ dex. $\mathrm{SDM_{1D}}$'s uncertainty (17.8%) is a result of its biased high mass cluster predictions and resulting residual skewness ($\gamma = 0.646$). $\mathrm{CNN_{1D}}$ predictions produce a low systematic uncertainty of 5.0%. $\mathrm{CNN_{2D}}$ produces the lowest non-Gaussian systematic uncertainty of all investigated models at 1.7%, slightly below that of the idealized M–σ_{pure} at 2.0% and $\mathrm{SDM_{2D}}$ at 3.8%.

5.3. Mass and Richness Dependence

We adopt the formalism introduced in Wojtak et al. (2018) to characterize the dependence of our models' bias and scatter on cluster mass and richness. Following this formalism, we assume that the distribution of our residuals ϵ (Equation (8)) is Gaussian with mass-dependent mean μ and richness-dependent scatter σ. We describe our residual distribution according to the following likelihood:

$$L \propto \prod_i \left[(1 - w_c)G(\epsilon_i;\, \mu,\, \sigma) + w_c G(\epsilon_i;\, \mu,\, \sigma_c)\right], \quad (11)$$

where $G(\epsilon;\, \mu,\, \sigma)$ is a Gaussian function of ϵ with mean μ and variance σ^2 and where the product is over the full contaminated

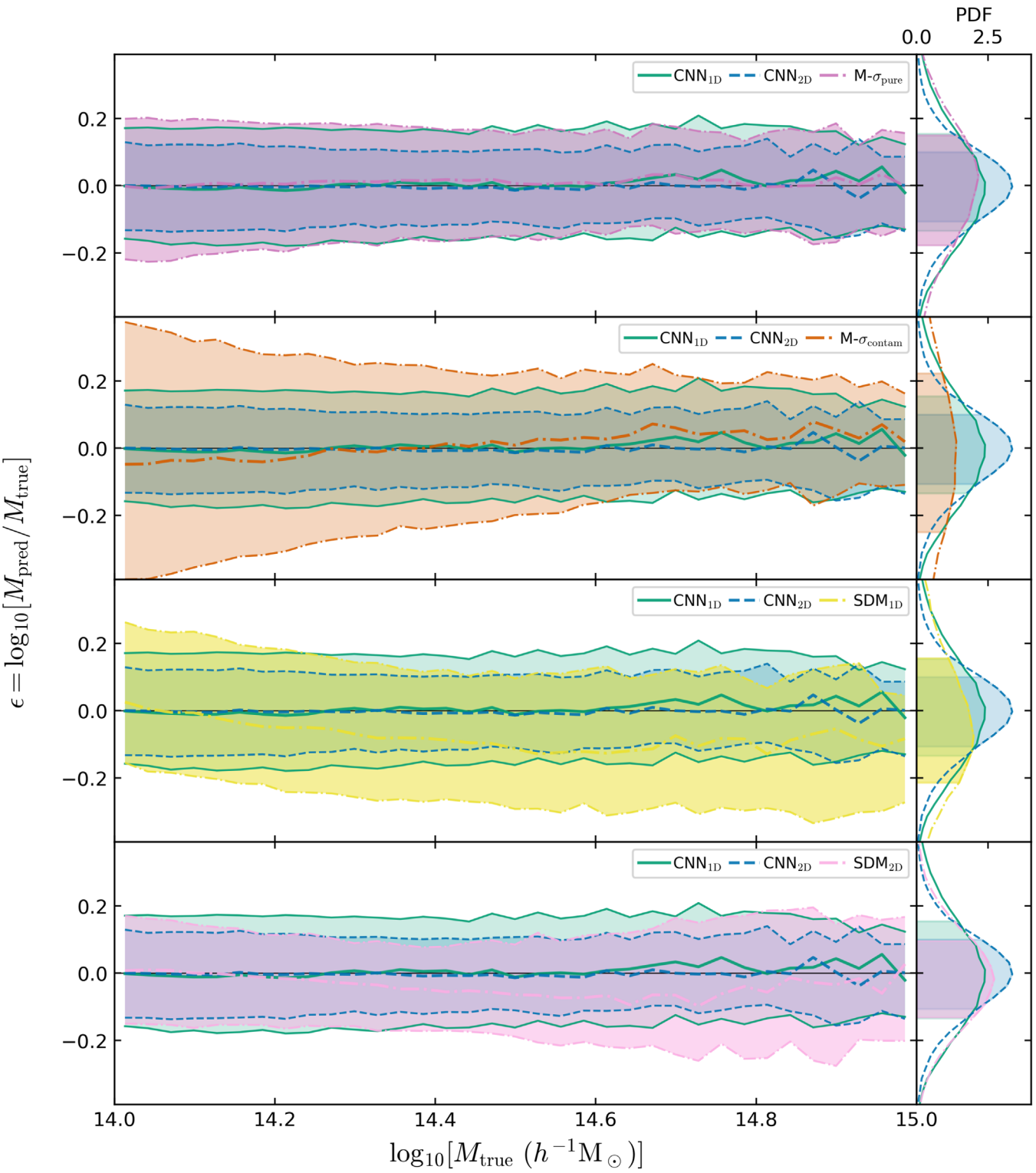


Figure 8. Prediction residuals ϵ (Equation (8)) for $\mathrm{CNN_{1D}}$ and $\mathrm{CNN_{2D}}$ relative to comparative models (Table 2), including the traditional M–σ and a modern ML approach (SDM; Ntampaka et al. 2016). For clarity, comparisons with various models are shown on separate rows, in the order of Table 2. Left column: residual distributions are binned along true mass and shown at their median and 16th–84th percentile range. Right column: residual distributions marginalized over true mass and plotted as PDFs. The highlighted region corresponds to the marginalized 16th–84th percentile range.

catalog test set. The second term in the likelihood accounts for a flat distribution of outliers and is parameterized by the nuisance parameter w_c and scatter σ_c, the latter of which is fixed to a large value, $\sigma_c = 10^{10}$ dex. The mean μ of our residual distribution is modeled as a linear function of logarithmic cluster mass:

$$\mu = \mu_0 + (\alpha_0 - 1)\log_{10}(M_{\rm true}/M_0), \tag{12}$$

where μ_0 and α_0 are free parameters, and the pivot mass M_0 is fixed to the median of our cluster sample, $M_0 = 10^{14.17}\ h^{-1}\ M_\odot$.

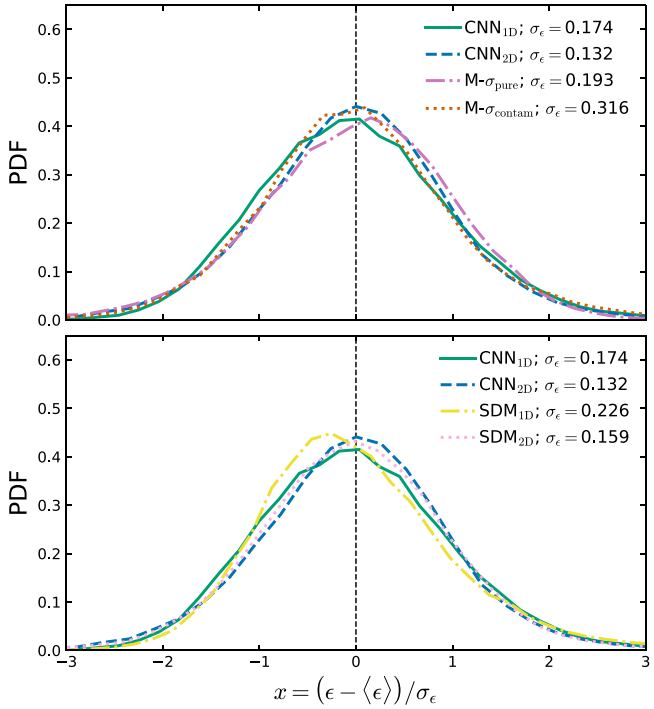


Figure 9. Distribution of normalized prediction residuals marginalized over true mass for each investigated model (Table 2). Each subfigure plots the PDF of residuals normalized by their mean $\langle\epsilon\rangle$ and scatter σ_ϵ. For context, residual scatter σ_ϵ (in dex) is listed in the legend for each model. For clarity, model comparisons with M–σ (upper) and SDM (lower) are shown on separate plots.

The residual scatter, σ, is related to cluster richness through the following parameterization:

$$\sigma^2 = \sigma_0^2 + \left(\frac{100}{N_{\rm true}}\right)\sigma_1^2, \tag{13}$$

where σ_0 and σ_1 are free parameters describing the intrinsic and richness-dependent scatter, respectively, and $N_{\rm true}$ denotes the true cluster richness as reported by the UniverseMachine catalog, ignoring sample contamination and incompleteness.

We use a Metropolis–Hastings algorithm to sample the likelihood (Equation (11)) and report the best-fit values in Table 3, marginalizing over the nuisance parameter w_c. For both models, our results indicate mass biases consistent with $\mu_0 = 0$ and a well-constrained log-linear $M_{\rm pred}$–$M_{\rm true}$ relation ($\alpha_0 = 1$). As expected, residual scatter for each model scales with cluster richness, with higher cluster richness (more information) leading to a reduced residual scatter. At our median richness of $\bar{N}_{\rm true} = 40$, about 55% and 47% of residual scatter can be explained by the intrinsic scatter for $\rm CNN_{1D}$ and $\rm CNN_{2D}$, respectively.

The mass and richness dependencies of CNN models are comparable to those of the 25 commonly used cluster mass estimation techniques analyzed as part of the Galaxy Cluster Mass Reconstruction Project (GCMRP; Wojtak et al. 2018). Table 3 indicates that the log-linear $M_{\rm pred}$–$M_{\rm true}$ relations recovered by CNN models show the least intrinsic and mass-dependent biases of all values reported by the GCMRP. In addition, our results suggest that the richness-corrected scatters of CNN models are among the lowest of all GCMRP models. However, we caution the reader against using the values listed in Table 3 as a direct, rigorous comparison with those published in Wojtak et al., on account of differences between our data set and that of the GCMRP. Most notably, the catalogs of mock clusters used in the GCMRP analyses (Old et al. 2015) were populated with galaxies using halo occupation distribution (HOD) or semi-analytic models (SAMs), each of which employs procedures considerably different from UniverseMachine. A future joint analysis of the CNN and GCMRP models conducted on the same data set would provide more reliable comparisons.

Table 3
Best-fit Model Parameters Characterizing the Dependence of Prediction Residuals on Cluster Mass and Richness (Wojtak et al. 2018)

Model	μ_0	σ_0	σ_1	α_0
$\rm CNN_{1D}$	$0.01^{+0.03}_{-0.02}$	$0.15^{+0.00}_{-0.00}$	$0.05^{+0.00}_{-0.00}$	$1.00^{+0.00}_{-0.00}$
$\rm CNN_{2D}$	$0.03^{+0.03}_{-0.02}$	$0.11^{+0.00}_{-0.00}$	$0.05^{+0.00}_{-0.00}$	$1.00^{+0.00}_{-0.00}$

Note. Each entry shows the median and 16th–84th percentile range for a Metropolis–Hastings sampling over the parameter space of the likelihood given in Equation (11).

5.4. Sampling Variation

We seek to quantify the robustness of our model predictions under variations in galaxy sampling rate. In practice, this is a measure of the reliability of our mass estimates when some fraction of galaxies are indistinguishable or otherwise not spectroscopically observed, as is common in astronomical observations. We construct the subsampled mass deviation $\epsilon^{(r)}$ as a measurement of prediction stability. For each model, we define $M_{\rm pred}^{(r)}$ as the mass prediction for a given cluster when its set of member galaxies is randomly subsampled at a rate of r without replacement. We choose to subsample randomly so as not to introduce new selection effects. The number of possible subsampled galaxy combinations can be intractably high, so we use the average subsampled mass prediction $\bar{M}_{\rm pred}^{(r)}$, calculated from a fixed number of subsampled combinations. For each cluster, we average mass predictions from 10 different galaxy subsamplings to assign a single measurement of $\bar{M}_{\rm pred}^{(r)}$. Following from this definition, the subsampled mass deviation $\epsilon^{(r)}$ (Equation (14)) is the logarithmic difference between the average subsampled mass prediction $\bar{M}_{\rm pred}^{(r)}$ and the fully sampled prediction $M_{\rm pred}^{(1.0)}$:

$$\epsilon^{(r)} = \log_{10}\left[\frac{\bar{M}_{\rm pred}^{(r)}}{M_{\rm pred}^{(1.0)}}\right]. \tag{14}$$

The subsampled mass deviation measures how much a model's predictions "drift" on average under fluctuations in sampling rate. Mass measurements that have a high reliance on cluster richness will show a strong correlation between r and $\epsilon^{(r)}$. While accurate, these models may fail when the sampling rate is not well constrained. We construct a cumulative statistic $\tilde{\epsilon}^{(6-8)} \pm \Delta\epsilon^{(6-8)}$, which describes the median and 16th–84th percentile scatter of all $\epsilon^{(r)}$ measurements within $0.6 \leqslant r \leqslant 0.8$. This measurement aims to characterize the bias and scatter involved with using each of the investigated models when the sampling rate is allowed to vary uniformly between 60% and 80%. In doing so, we capture the effects of both intrinsic scatter and richness dependence within our models' predictions. Ideal model performance involves producing low values of $|\tilde{\epsilon}^{(6-8)}|$ and $\Delta\epsilon^{(6-8)}$. Regardless of model performance, we expect

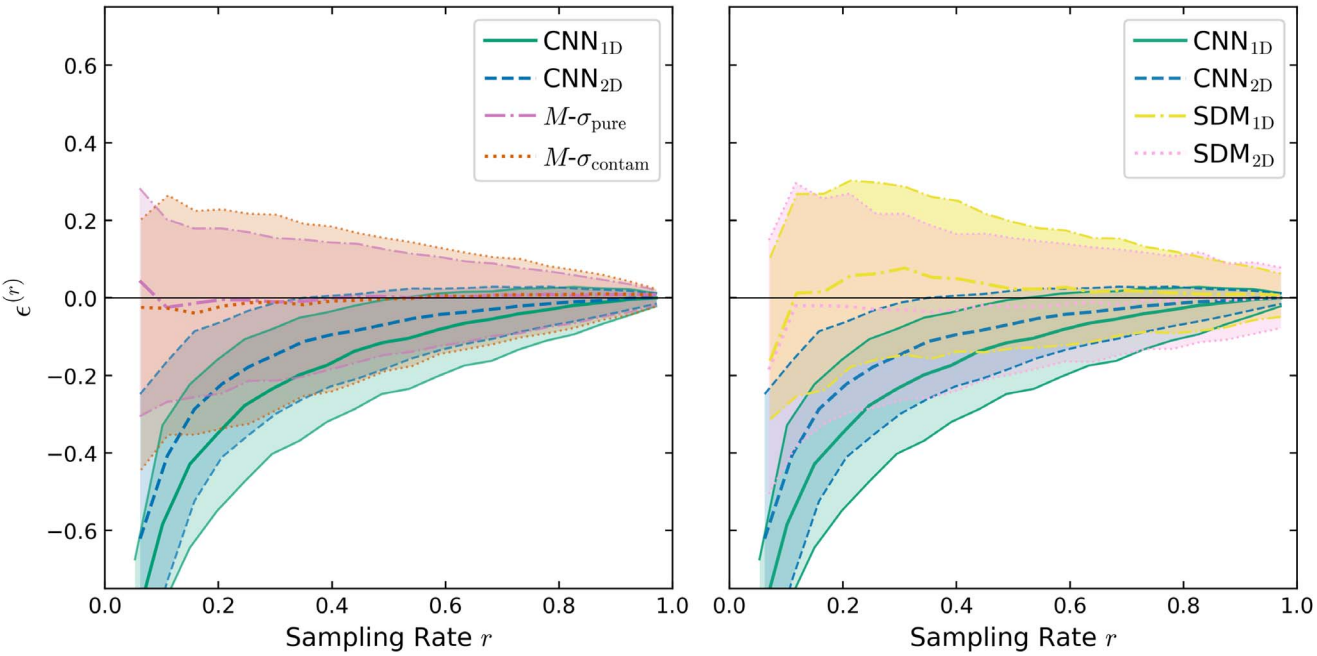


Figure 10. Subsampled mass deviation $\epsilon^{(r)}$ (Equation (14)) at a range of sampling rates $0 \leqslant r \leqslant 1$ for CNN_{1D}, CNN_{2D}, and comparative models (Table 2). Subsampled mass deviation is a measure of how model predictions "drift" when galaxies are randomly removed from the input. The CNN models as plotted here show low prediction drift under variations in galaxy sampling rate relative to other models. These deviation trends are independent of original cluster mass and richness. Distributions are binned and shown at their median and 16th–84th percentile range. For clarity, model comparisons with M–σ (left) and SDM (right) are shown on separate plots.

sampling mass fraction to deviate strongly from $\epsilon^{(r)} = 0$ as $r \to 0^+$ due to loss of input information.

Measurements of $\epsilon^{(r)}$ for the CNN and M–σ models are constructed via inductive learning, i.e., by optimizing model parameters on fully sampled training data and subsequently inferring masses for subsampled test data. Due to the transductive nature of SDM, both train and test data have an impact on SDM model fitting and must be used jointly in the learning procedure. We fit numerous iterations of SDM models, each trained on the same fully sampled training data and evaluated on a unique set of sampled test data. For a single iteration, each cluster in the test data set is subsampled by the same fraction r. This is done to mimic realistic observation conditions; each iteration corresponds to observation conditions where the galaxy observation rate is fixed at r. Consolidating the mass predictions made by each SDM iteration and comparing them to the fully sampled $r = 1$ predictions produce estimations of $\epsilon^{(r)}$ for the range of possible sampling rates.

Figure 10 shows the subsampled mass deviation distribution for the investigated ML models and the traditional M–σ as a function of sampling rate r. These sampling variation trends are independent of true sample richness or mass. Cumulative statistics for these distributions are calculated in Table 4. As expected, each model tends to deviate strongly from $\epsilon^{(r)} = 0$ at low r due to loss of input information. At sampling rates $r < 0.2$, we see sharp changes in $\epsilon^{(r)}$, suggesting that $r = 0.2$ marks a considerable loss of cluster structure information. Below this threshold, dynamical mass measurements may encounter considerable difficulty in resolving the necessary information to make accurate mass predictions.

CNN_{1D} and CNN_{2D} have similar sampling variation curves, with CNN_{2D} exhibiting slightly less sensitivity to the sampling rate. The $\epsilon^{(r)}$ in the CNN models show a slight correlation with sampling fraction, suggesting that the CNNs derive some information from sample richness. For a sampling rate chosen uniformly between 0.6 and 0.8, the sampled mass predictions for CNN_{1D} or CNN_{2D} can be expected to vary within a $\pm 1\sigma$ interval of 85%–102% or 90%–103% of their fully sampled prediction, respectively. Both CNN models converge to negative values of $\epsilon^{(0)}$, demonstrative of a model output of $y = 0$ (Equation (4)).

Figure 10 infers the argument that the CNN models are less sensitive to sampling variation than either the M–σ or SDM approaches. The width of the $\epsilon^{(r)}$ scatter for the M–σ models increases considerably as more cluster members are randomly removed. The M–σ models do not bias away from $\epsilon^{(r)} = 0$ as a result of the richness-corrected velocity dispersion estimator (Equation (6)). The SDM models produce a higher sampling

Table 4
Cumulative Statistics of Robustness Measurements for All Investigated Models Listed in Table 2

Model	Color	$\tilde{\epsilon}^{(6-8)} \pm \Delta\epsilon^{(6-8)}$[a]	$\Delta\epsilon^{(6-8)}$[b]
CNN_{1D}	green	$-0.049^{+0.069}_{-0.100}$	0.168
CNN_{2D}	blue	$-0.026^{+0.053}_{-0.073}$	0.126
M–σ_{pure}	violet	$0.005^{+0.077}_{-0.102}$	0.179
M–σ_{contam}	orange	$0.006^{+0.095}_{-0.121}$	0.216
SDM_{1D}[c]	yellow	$0.017^{+0.126}_{-0.118}$	0.244
SDM_{2D}[c]	pink	$-0.013^{+0.133}_{-0.135}$	0.268

Notes.
[a] Subsampled mass deviation median and 16th–84th percentile range marginalized over sampling rates within $0.6 \leqslant r \leqslant 0.8$ in dex.
[b] 16th–84th percentile width in dex.
[c] Ntampaka et al. (2016).

variation scatter than the M–σ and also do not bias considerably from $\epsilon^{(r)} = 0$.

The CNN models display the lowest six to eight sampling deviation scatter of all investigated models, reducing the six to eight residual ranges of the best M–σ and SDM models by up to 30%. This robust behavior is primarily driven by the KDEs used to normalize the CNN model input, which are relatively insensitive to variations in sample number count. The CNN estimators presented in this paper are shown to be robust under fluctuations in the sampling rate.

5.5. Training and Evaluation Time

The final performance metrics we will consider are estimations of training and evaluation time. While of secondary importance to prediction error, fast implementation and execution are advantageous qualities of cluster measurements, especially when analyzing large data sets. As the abundance of high-quality data continues to increase (Dodelson et al. 2016), mass modeling methods are expected to improve computational efficiency.

In the analysis presented here, we have seen that implementation of the CNN approach is significantly faster than SDM. The full cross-validation training-and-evaluation procedure run on the catalog described in Section 2.2 lasts approximately 10 minutes with CNN models and 6 hr with SDM. This CNN speedup is important, especially given that the practical data sets may be orders of magnitude larger than those discussed here.

In general, CNN models are more computationally efficient than SDMs. SDMs are nonparametric and transductive (Sutherland et al. 2012), meaning that the model complexity and evaluation procedure scale as the number of train+test points. The training and evaluation steps for SDM influence one another, implying that fitted SDM models need to be retrained upon encountering new unlabeled test data. These attributes may be undesirable in practice, where the test examples may scale up to terabytes of data. In comparison, CNN models undergo supervised, inductive learning procedures, where training and evaluation are independent calculations. The complexity of CNNs is fixed by the chosen neural architecture. In recent years, deep neural models such as CNNs have benefited from the increased use of GPUs, which speed up evaluations of neural architecture considerably (LeCun et al. 2015). CNNs find use in applications where data are overwhelmingly abundant. Under these conditions, other models such as SDM may be intractable.

6. Conclusion

We present a novel ML method for inferring dynamical masses of galaxy clusters. Our method leverages the use of CNNs to model complex cluster substructure and to mediate systematics of traditional dynamical mass measurements. We learn cluster mass directly from distributions of galaxy kinematics, namely LOS velocity ($v_{\rm los}$) and projected radial distance to the cluster center ($R_{\rm proj}$). We employ KDEs to create normalized heatmap "images" of these distributions, which serve as input to our deep neural architecture. Using this set of inputs, we train CNNs as a regression over a single output variable, the logarithmic cluster mass ($\log_{10}[M_{200c}\ (h^{-1}\,M_{\odot})]$). We then assign cluster mass predictions to unseen test data via inductive inference. This paper discusses two versions of this method, named $\rm CNN_{1D}$ and $\rm CNN_{2D}$ for their respective learned input spaces $\{v_{\rm los}\}$ and $\{R_{\rm proj}, v_{\rm los}\}$.

We train and evaluate our model using a catalog of realistic mock cluster observations constructed from dark matter simulation at a single redshift snapshot of $z = 0.117$. The mock observations determine cluster membership via a simplistic cylindrical cut of fixed aperture ($R_{\rm aperture} = 1.6\,h^{-1}$ Mpc) and velocity cut ($v_{\rm cut} = 2200$ km s^{-1}). We use a 10-fold cross-validation scheme to rigorously test our models on independent mock observations. We perform a comparative analysis of our models' performances with respect to several baselines including the realistic and idealized M–σ and a similar ML method (SDM; Ntampaka et al. 2015, 2016). The findings of our analysis are summarized as follows:

1. $\rm CNN_{1D}$ and $\rm CNN_{2D}$ produce mass predictions with low scatter and bias in the mass range $14 \leqslant \log_{10}[M_{200c}\ (h^{-1}\,M_{\odot})] \leqslant 15$. We see that $\rm CNN_{2D}$ reduces the error margin of $\rm CNN_{1D}$ by 24%, suggesting that the supplemental $R_{\rm proj}$ input is informative of cluster mass. Training and validation loss curves do not indicate overfitting.
2. $\rm CNN_{1D}$ and $\rm CNN_{2D}$ reduce the error margin of simplistic, contaminated M–σ measurements by 45% and 58%, respectively. We compare our models to an M–σ measurement with perfect member selection (pure and complete) and observe that $\rm CNN_{1D}$ and $\rm CNN_{2D}$ reduce prediction error by 10% and 32%, respectively.
3. CNN methods show improved predictive performance relative to SDM (Ntampaka et al. 2015, 2016) and other ML approaches (Armitage et al. 2019a). In our comparison, $\rm CNN_{2D}$ reduces the error of SDM by 17%.
4. Mass predictions from CNN models have lognormal residuals. The effects of non-Gaussianity in $\rm CNN_{2D}$ residuals result in a lower systematic uncertainty in cluster abundance measurements than all other investigated models (1.7%).
5. In the context of our test catalog, the CNN models recover log-linear $M_{\rm pred}$–$M_{\rm true}$ relations with biases and scatter among the lowest measured for modern galaxy-based cluster mass estimators (Wojtak et al. 2018). However, we present this conclusion with the caveat that analyses of our models and those of other recorded methods were conducted on different data sets.
6. CNN methods are robust under input sampling variation. Relative to M–σ and SDM, predictions made by CNN models show the lowest prediction variation when inputs are randomly subsampled. This is a desirable model property, especially under conditions where some unknown fraction of galaxies are indistinguishable or otherwise not observable.
7. For either CNN model, the 20 epoch training procedure of a single fold with ∼10,000 labeled inputs lasts about one minute. For each test input, average evaluation time can be broken down into KDE generation time (73 μs for $\rm CNN_{1D}$ and 410 μs for $\rm CNN_{2D}$) and network evaluation time (44 μs for either model). The entire training and evaluation procedure for CNN models is considerably faster than that of SDM (∼6 hr).

We remark that the results described in this manuscript are presented in the context of the assumptions listed in Section 2.2. The cluster observations used to train our model are independent from, but constructed identically to, our

evaluation catalog. Our catalog construction procedure does not account for a variety of observational systematics such as obstruction, lensing, miscentering, or galaxy dark matter bias. Mass estimates produced by our model are only reliable for clusters at redshifts near that of our training catalog, as a result of redshift-dependent factors such as the definition of M_{200c} and the distribution of interloping galaxies. Lastly, the CNN models produce singular point estimates of cluster mass and their cross-validation scatter σ_ϵ should not be interpreted as a Bayesian posterior. We seek to perform the above further analyses as part of a later publication.

In conclusion, mass predictions produced by CNN methods have low, lognormal error relative to other dynamical mass estimates, are stable under input sampling variation, and are computationally efficient to implement and evaluate. The CNN approach presented here may be a preferred dynamical mass estimator under conditions where high-quality simulated data is abundant or where richness measurements are uncertain or expensive. Future work involving this approach would investigate CNN modeling with more complex data inputs and deeper neural architectures. These models could potentially consolidate information from a variety of measurements (spectroscopic, X-ray, microwave, etc.) to produce a complete, precise, and unbiased prediction of cluster mass.

We thank the reviewer as well as Rachel Mandelbaum and Yizhou He for helpful input while developing this project. We thank Andrew Hearin and Peter Behroozi for preparing UniverseMachine catalogs of MDPL2 simulation data. This work is supported in part by DOE DE-SC0011114 and NSF 1563887. The computing resources necessary to complete this analysis were provided by the Pittsburgh Supercomputing Center. The CosmoSim database used in this paper is a service by the Leibniz-Institute for Astrophysics Potsdam (AIP). The MultiDark database was developed in cooperation with the Spanish MultiDark Consolider Project CSD2009-00064.

Appendix

We describe the procedure for calculating $x_{\rm proj}$, $y_{\rm proj}$, $R_{\rm proj}$, and $v_{\rm los}$ for an arbitrary cluster–galaxy pair under the assumptions stated in Section 2.2. Let $\boldsymbol{r}_{\rm clu}^{\rm (CM)}$ and $\boldsymbol{r}_{\rm gal}^{\rm (CM)}$ represent the comoving simulation positions of the cluster and galaxy, respectively. Furthermore, define $\boldsymbol{r} = \boldsymbol{r}_{\rm gal}^{\rm (CM)} - \boldsymbol{r}_{\rm clu}^{\rm (CM)}$ to be the comoving distance vector between the objects. Let $\{\hat{\boldsymbol{x}}_{\rm los}, \hat{\boldsymbol{y}}_{\rm los}, \hat{\boldsymbol{z}}_{\rm los}\}$ be an orthonormal basis representation of the chosen LOS, where $\hat{\boldsymbol{z}}_{\rm los}$ is oriented along the LOS axis, and $\hat{\boldsymbol{x}}_{\rm los}$ and $\hat{\boldsymbol{y}}_{\rm los}$ dictate the azimuthal orientation of the observer. Under these conditions, we can write $x_{\rm proj}$ and $y_{\rm proj}$ for a given cluster–galaxy relationship as follows:

$$x_{\rm proj} = [\boldsymbol{r} - (\boldsymbol{r}\cdot\hat{\boldsymbol{z}}_{\rm los})\hat{\boldsymbol{z}}_{\rm los}]\cdot\hat{\boldsymbol{x}}_{\rm los}, \tag{15}$$

$$y_{\rm proj} = [\boldsymbol{r} - (\boldsymbol{r}\cdot\hat{\boldsymbol{z}}_{\rm los})\hat{\boldsymbol{z}}_{\rm los}]\cdot\hat{\boldsymbol{y}}_{\rm los}. \tag{16}$$

We will also often use an additional quantity called the projected radius $R_{\rm proj} = (x_{\rm proj}^2 + y_{\rm proj}^2)^{1/2}$, which is invariant to azimuthal rotations.

To calculate $v_{\rm los}$, we first find the comoving distance from the observer to the galaxy:

$$d_{\rm clu}^{\rm (CM)} = \int_0^{z_{\rm clu}} \frac{c}{H(z)}\,dz \tag{17}$$

$$\begin{aligned} d_{\rm gal}^{\rm (CM)} &= |d_{\rm clu}^{\rm (CM)}\hat{\boldsymbol{z}}_{\rm los} + \boldsymbol{r}| \\ &\approx d_{\rm clu}^{\rm (CM)} + \boldsymbol{r}\cdot\hat{\boldsymbol{z}}_{\rm los}, \end{aligned} \tag{18}$$

where $z_{\rm clu}$ is the redshift of the cluster center. $H(z)$ is the Hubble parameter as a function of redshift and is dependent on the chosen cosmology (see Section 2.1). Because Equation (17) generally has no analytical solution, we use a numerical quadrature interpolation scheme to generate a function for $d^{\rm (CM)}(z)$ and the corresponding inverse $z(d^{\rm (CM)})$. The latter allows us to calculate $z_{\rm gal} = z(d_{\rm gal}^{\rm (CM)})$, which is necessary for determining Hubble flow velocities $v^{\rm (H)}$.

$$v^{\rm (H)}(z) = \left[\frac{(1+z)^2 - 1}{(1+z)^2 + 1}\right]c. \tag{19}$$

Let $\boldsymbol{v} = \boldsymbol{v}_{\rm gal}^{\rm (P)} - \boldsymbol{v}_{\rm clu}^{\rm (P)}$ represent the relative comoving peculiar velocity between the cluster candidate and the galaxy member. We apply the small-angle approximation again to calculate peculiar velocities along the LOS $v^{\rm (P,los)}$,

$$v_{\rm clu}^{\rm (P,los)} = \boldsymbol{v}^{\rm (P)}\cdot\hat{\boldsymbol{z}}_{\rm los} \tag{20}$$

$$\begin{aligned} v_{\rm gal}^{\rm (P,los)} &= |\boldsymbol{v}_{\rm clu}^{(P)} + \boldsymbol{v}| \\ &\approx v_{\rm clu}^{\rm (P,los)} + \boldsymbol{v}\cdot\hat{\boldsymbol{z}}_{\rm los}. \end{aligned} \tag{21}$$

Equipped with these peculiar velocities and the following Hubble velocities $v_{\rm clu}^{\rm (H)} = v^{\rm (H)}(z_{\rm clu})$ and $v_{\rm gal}^{\rm (H)} = v^{\rm (H)}(z_{\rm gal})$, we can finally write an expression for $v_{\rm los}$,

$$v_{\rm los} = (v_{\rm gal}^{\rm (P,los)} + v_{\rm gal}^{\rm (H)}) - (v_{\rm clu}^{\rm (P,los)} + v_{\rm clu}^{\rm (H)}), \tag{22}$$

where $\pm$ are the relativistic linear velocity addition/subtraction operators.

ORCID iDs

Matthew Ho https://orcid.org/0000-0003-3207-8868
Markus Michael Rau https://orcid.org/0000-0003-3709-1324
Michelle Ntampaka https://orcid.org/0000-0002-0144-387X
Arya Farahi https://orcid.org/0000-0003-0777-4618
Hy Trac https://orcid.org/0000-0001-6778-3861

References

Abdullah, M. H., Wilson, G., & Klypin, A. 2018, ApJ, 861, 22
Allen, S. W., Evrard, A. E., & Mantz, A. B. 2011, ARA&A, 49, 409
Applegate, D. E., von der Linden, A., Kelly, P. L., et al. 2014, MNRAS, 439, 48
Armitage, T. J., Kay, S. T., & Barnes, D. J. 2019a, MNRAS, 484, 1526
Armitage, T. J., Kay, S. T., Barnes, D. J., Bahé, Y. M., & Dalla Vecchia, C. 2019b, MNRAS, 482, 3308
Baxter, E. J., Rozo, E., Jain, B., et al. 2016, MNRAS, 463, 205
Behroozi, P., Wechsler, R. H., Hearin, A. P., et al. 2019, MNRAS, 488, 3143
Behroozi, P. S., Wechsler, R. H., & Wu, H. 2013, ApJ, 762, 109
Calderon, V. F., & Berlind, A. A. 2019, MNRAS, 490, 2367
Diaferio, A. 1999, MNRAS, 309, 610
Diaferio, A., & Geller, M. J. 1997, ApJ, 481, 633
Dieleman, S., Willett, K. W., & Dambre, J. 2015, MNRAS, 450, 1441
Dodelson, S., Heitmann, K., Hirata, C., et al. 2016, arXiv:1604.07626
Erickson, B. M. S., Cunha, C. E., & Evrard, A. E. 2011, PhRvD, 84, 103506
Evrard, A. E., Bialek, J., Busha, M., et al. 2008, ApJ, 672, 122
Farahi, A., Evrard, A. E., Rozo, E., Rykoff, E. S., & Wechsler, R. H. 2016, MNRAS, 460, 3900
Farahi, A., Guglielmo, V., Evrard, A. E., et al. 2018, A&A, 620, A8
Gerke, B. F., Newman, J. A., Davis, M., et al. 2005, ApJ, 625, 6

Gifford, D., & Miller, C. J. 2013, ApJL, 768, L32
Giles, P. A., Maughan, B. J., Dahle, H., et al. 2017, MNRAS, 465, 858
González, Á 2010, Math. Geosci., 42, 49
Hoyle, B. 2016, A&C, 16, 34
Kingma, D. P., & Ba, J. 2014, arXiv:1412.6980
Klypin, A., Yepes, G., Gottlöber, S., et al. 2016, MNRAS, 457, 4340
Lanusse, F., Ma, Q., Li, N., et al. 2018, MNRAS, 473, 3895
LeCun, Y., Bengio, Y., & Hinton, G. 2015, Natur, 521, 436
LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. 1998, Proc. IEEE, 86, 2278
Mamon, G. A., Biviano, A., & Boué, G. 2013, MNRAS, 429, 3079
Mantz, A. B., Allen, S. W., Morris, R. G., et al. 2016, MNRAS, 463, 3582
Mantz, A. B., von der Linden, A., Allen, S. W., et al. 2015, MNRAS, 446, 2205
McClintock, T., Varga, T. N., Gruen, D., et al. 2019, MNRAS, 482, 1352
Nair, V., & Hinton, G. E. 2010, in Proc. 27th Int. Conf. on Machine Learning, ICML '10 (USA: Omnipress), 807, https://dl.acm.org/citation.cfm?id=3104322.3104425
Ntampaka, M., Trac, H., Sutherland, D. J., et al. 2015, ApJ, 803, 50
Ntampaka, M., Trac, H., Sutherland, D. J., et al. 2016, ApJ, 831, 135
Ntampaka, M., ZuHone, J., Eisenstein, D., et al. 2018, ApJ, 876, 82
Old, L., Skibba, R. A., Pearce, F. R., et al. 2014, MNRAS, 441, 1513
Old, L., Wojtak, R., Mamon, G. A., et al. 2015, MNRAS, 449, 1897
Old, L., Wojtak, R., Pearce, F. R., et al. 2018, MNRAS, 475, 853
Planck Collaboration, Ade, P. A. R., Aghanim, N., et al. 2014, A&A, 571, A16
Planck Collaboration, Ade, P. A. R., Aghanim, N., et al. 2016, A&A, 594, A24
Ribeiro, A. L. B., Lopes, P. A. A., & Trevisan, M. 2011, MNRAS, 413, L81
Robbins, H., & Monro, S. 1951, Ann. Math. Stat., 22, 400
Saro, A., Mohr, J. J., Bazin, G., & Dolag, K. 2013, ApJ, 772, 47
Schölkopf, B., & Smola, A. J. 2002, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Cambridge, MA: MIT Press)
Scott, D. W. 2015, Multivariate Density Estimation: Theory, Practice, and Visualization (2nd ed.; Houston, TX: Wiley)
Shaw, L. D., Holder, G. P., & Dudley, J. 2010, ApJ, 716, 281
Skielboe, A., Wojtak, R., Pedersen, K., Rozo, E., & Rykoff, E. S. 2012, ApJL, 758, L16
Springel, V. 2005, MNRAS, 364, 1105
Sunyaev, R. A., & Zeldovich, Y. B. 1972, CoASP, 4, 173
Sutherland, D. J., Xiong, L., Póczos, B., & Schneider, J. 2012, arXiv:1202.0302
Svensmark, J., Wojtak, R., & Hansen, S. H. 2015, MNRAS, 448, 1644
Voit, G. M. 2005, RvMP, 77, 207
Wang, Q., Kulkarni, S. R., & Verdu, S. 2009, ITIT, 55, 2392
Weinberg, D. H., Mortonson, M. J., Eisenstein, D. J., et al. 2013, PhR, 530, 87
White, M., Cohn, J. D., & Smit, R. 2010, MNRAS, 408, 1818
Wojtak, R. 2013, A&A, 559, A89
Wojtak, R., Łokas, E. L., Mamon, G. A., et al. 2007, A&A, 466, 437
Wojtak, R., Old, L., Mamon, G. A., et al. 2018, MNRAS, 481, 324
Yee, H. K. C., & Ellingson, E. 2003, ApJ, 585, 215
Zwicky, F. 1933, AcHPh, 6, 110