Monthly Notices
of the
ROYAL ASTRONOMICAL SOCIETY

MNRAS **499,** 3445–3458 (2020)
Advance Access publication 2020 October 1

doi:10.1093/mnras/staa3030

Galaxy cluster mass estimation with deep learning and hydrodynamical simulations

Z. Yan ,[1★] A. J. Mead ,[1,2] L. Van Waerbeke,[1★] G. Hinshaw[1] and I. G. McCarthy [3]

[1] *Department of Physics and Astronomy, University of British Columbia, 6224 Agricultural Road, Vancouver, BC, V6T 1Z1, Canada*
[2] *Institut de Ciències del Cosmos, Universitat de Barcelona, Martí Franquès 1, E-08028 Barcelona, Spain*
[3] *Astrophysics Research Institute, Liverpool John Moores University, 146 Brownlow Hill, Liverpool L3 5RF, UK*

Accepted 2020 September 28. Received 2020 September 28; in original form 2020 May 27

ABSTRACT

We evaluate the ability of convolutional neural networks (CNNs) to predict galaxy cluster masses in the BAHAMAS hydrodynamical simulations. We train four separate single-channel networks using: stellar mass, soft X-ray flux, bolometric X-ray flux, and the Compton y parameter as observational tracers, respectively. Our training set consists of ∼4800 synthetic cluster images generated from the simulation, while an additional ∼3200 images form a validation set and a test set, each with 1600 images. In order to mimic real observation, these images also contain uncorrelated structures located within 50 Mpc in front and behind clusters and seen in projection, as well as instrumental systematics including noise and smoothing. In addition to CNNs for all the four observables, we also train a 'multichannel' CNN by combining the four observational tracers. The learning curves of all the five CNNs converge within 1000 epochs. The resulting predictions are especially precise for halo masses in the range $10^{13.25}\,\mathrm{M}_{\odot} < M < 10^{14.5}\,\mathrm{M}_{\odot}$, where all five networks produce mean mass biases of order ≈1 per cent with a scatter of $\lesssim$20 per cent. The network trained with Compton y parameter maps yields the most precise predictions. We interpret the network's behaviour using two diagnostic tests to determine which features are used to predict cluster mass. The CNNs trained with stellar mass images detect galaxies (not surprisingly), while CNNs trained with gas-based tracers utilize the shape of the signal to estimate cluster mass.

Key words: hydrodynamics – galaxies: clusters: general – galaxies: groups: general – dark matter – large-scale structure of Universe.

1 INTRODUCTION

Galaxy groups and clusters are collections of several up to thousands of galaxies that are bound by their mutual gravity. With masses in the range of 10^{13}–$10^{15}\,\mathrm{M}_{\odot}$, they are the most massive collapsed objects in the Universe. Their abundance, distribution, and morphology depend both on local physical processes and the underlying cosmological model. Stars typically comprise about 1 per cent of a cluster's mass (e.g. Leauthaud et al. 2011; Zu & Mandelbaum 2015), while hot gas contributes anywhere from ≈7 to 13 per cent (depending on cluster mass; e.g. Allen, Schmidt & Fabian 2002; Pratt et al. 2009; Sun et al. 2009), with the remainder residing in a dark matter halo.

The cluster mass function is a particularly sensitive probe of cosmological parameters and the evolutionary history of large-scale structure (e.g. Voit 2005; Allen, Evrard & Mantz 2011; Planck Collaboration XXIV 2016). However, it is difficult to precisely and accurately measure cluster masses directly because they are dominated by dark matter. Masses can be inferred from weak gravitational lensing data (e.g. Umetsu 2010; Shan et al. 2012; von der Linden et al. 2014; Hoekstra et al. 2015), but the current signal-to-noise ratio of such observations limits the precision of individual cluster masses to typically (at least) tens of per cent. This is not sufficiently precise to be used directly for precision cosmology (via the mass function), but weak lensing is still a very important probe because it can be used to calibrate the mean bias[1] of other tracers whose system-to-system scatter is lower. Examples of such tracers include the total stellar mass or cluster richness, X-ray emission in the form of thermal bremsstrahlung and recombination lines from the hot intracluster medium (ICM), and the tSZ effect (i.e. the inverse Compton scattering of cosmic microwave background photons off hot ICM electrons as they pass through clusters).

Note that X-ray emission itself can be used to infer mass by combining spectroscopic measurements of the temperature profile with surface brightness measurements that strongly constrain the density profile, allowing one to infer a mass under the assumption of hydrostatic equilibrium. How well this assumption holds is currently a subject of strong debate, with the level of deviation from hydrostatic equilibrium having been estimated to be anywhere from 40 per cent (i.e. the hydrostatic mass underestimates the true mass by this amount; e.g. von der Linden et al. 2014) to only $\lesssim$ 5 per cent (e.g. Melin & Bartlett 2015; Smith et al.

[1] It has been shown from mock analyses of weak lensing observations of simulated clusters that weak lensing mass measurements yield a nearly unbiased mean mass estimate (e.g. Becker & Kravtsov 2011; Bahé, McCarthy & King 2012).

★ E-mail: yanza15@phas.ubc.ca (ZY); waerbeke@phas.ubc.ca (LVW)

Published by Oxford University Press on behalf of the Royal Astronomical Society

2016). The Halo Occupation Distribution (Peacock & Smith 2000; Seljak 2000) model links halo mass with galaxy properties. Moster et al. (2010) study relations between stellar mass and halo mass. The thermal Sunyaev-Zel'dovich (tSZ) effect is also known to be related to cluster masses through $Y_{500}-M_{500}$ relation where Y_{500} is the Compton y parameter within r_{500}(Melin et al. 2011). These studies link stellar mass, X-ray luminosity, and tSZ with cluster mass, which suggests that they are an important supplement to gravitational lensing to estimate cluster masses and probe cluster physics.

Large-scale hydrodynamical simulations are playing an increasingly important role in calibrating the inference of cluster masses from observational data. These simulations are now capable of capturing gravitational and gas dynamics on cosmological scales and can therefore provide large samples of realistic clusters in order to assess mass inferences statistically. They also aid in understanding systematic effects that may hinder these inferences (see Borgani & Kravtsov 2011 for a review of hydrodynamical simulations). However, even with these simulations, the complexities of substructure, morphology, and small-scale physical processes, such as active galactic nuclei (AGN) feedback (Gitti, Brighenti & McNamara 2012) and gas clumping (Nagai & Lau 2011), hinder the accuracy of many cluster mass estimates. Yan et al. (2020) used hydrodynamical simulations to assess the effect of cluster miscentring on mass determinations.

Machine learning (ML) is a technique in which computer systems learn to analyse data without using explicit instructions or model parametrizations but instead are 'trained' to make decisions based on properties of the data itself. In astronomy, ML algorithms such as linear regression, decision tree, random forest, and Principal Component Analysis have been widely used in model fitting and feature extraction [see Baron (2019) for a review].

Artificial Neural Networks (ANNs) are a popular class of ML tools inspired by the way in which biological nervous systems, such as the brain, process information. ANNs use a hierarchy of simple functions, called activation functions, to construct a highly non-linear function. Given their ability to mimic complicated functions, ANNs form the basis of many voice recognition and image identification tools. A Convolutional Neural Network (CNN) is a category of ANN that is particularly useful in the field of object identification and image classification. CNNs have been used by astronomers to classify galaxy morphology (Banerji et al. 2010), to identify lensing shear (Lanusse et al. 2017), to generate cosmic webs (Rodríguez et al. 2018), and to directly constrain cosmological parameters (Ribli, Pataki & Csabai 2019).

Cohn & Battaglia (2020) and Armitage, Kay & Barnes (2019) use multiple machine-learning algorithms to estimate cluster mass from a set of observable quantities. Green et al. (2019) use X-ray observational parameters, Ntampaka et al. (2019) use mock X-ray images, and Gupta & Reichardt (2020) use simulated microwave sky to train neural networks to predict cluster mass. Here, we extend their work and utilize CNNs to predict cluster masses from stellar mass data, X-ray data, Compton y data, and from combinations of them. The test data are the BAHAMAS hydrodynamical simulations (McCarthy et al. 2017). We choose M_{200} as the proxy for cluster mass. This is the total mass within the characteristic radius r_{200}, the radius at which the cluster density falls to 200 times the critical density of the (simulated) universe.

The structure of this paper is as follows: Section 2 describes the simulation data and the setup of our CNN; Section 3 presents our results; Section 4 describes a test to understand the behaviour of the CNN; and Section 5 presents our conclusions.

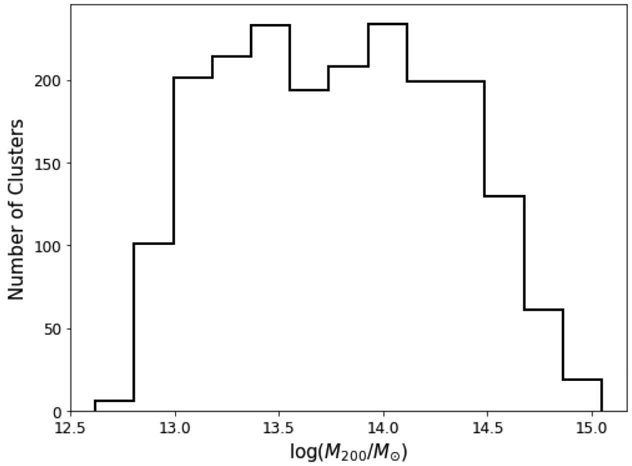


Figure 1. The mass distribution of galaxy clusters that we analyse from the BAHAMAS simulation.

2 DATA AND METHOD

2.1 The BAHAMAS simulation

We employ data from the BAHAMAS (BAryons and Haloes of MAssive Systems, McCarthy et al. 2017, 2018) simulations. BAHAMAS is a suite of cosmological, hydrodynamical simulations run using a modified version of the TreePM SPH code GADGET3. The simulations consist of 400 cMpc/h periodic boxes containing 2 × 1024^3 particles (with equal numbers of dark matter and baryonic particles). The run we use adopts the WMAP 9-yr best-fitting cosmology with massless neutrinos (Hinshaw et al. 2013).

BAHAMAS includes subgrid treatments of important physical processes that cannot be directly resolved in the simulations, including metal-dependent radiative cooling, star formation, stellar evolution and mass loss, black hole formation and growth, and stellar and AGN feedback. The subgrid models were developed as part of the OWLS project (Schaye et al. 2010). The parameters governing the efficiencies of AGN and stellar feedback were adjusted so that the simulations approximately reproduce the observed galaxy stellar mass function for $M_* \geq 10^{10}$ $M_{\odot}$ and the hot gas fraction–halo mass relation of groups and clusters, as determined from high-resolution X-ray observations of local systems. As shown in McCarthy et al. (2017), the simulations match the galaxy–halo–tSZ–X-ray scaling relations of galaxies and groups and clusters.

For the present study, friends-of-friends haloes are selected from the dark matter-only simulation that accompanies the BAHAMAS hydrodynamical simulations.[2] We select up to 200 haloes in each of 10 mass bins of width of 0.25 dex, spanning the range $M_{200} = 10^{13} - 10^{15}$ $M_{\odot}$, resulting in a sample of almost 2000 haloes (some bins have slightly fewer than 200 haloes). This sample is then matched to the BAHAMAS hydrodynamical simulation. The resulting number distribution of clusters as a function of mass is shown in Fig. 1. The distribution is not perfectly flat because the hydrodynamical masses are different from the underlying dark matter-only masses.

We tag all particles (gas, dark matter, and stellar) within $2r_{200}$ of the most bound particle for analysis. We also generate a catalogue

[2] We select haloes from a dark matter-only simulation so as to facilitate comparisons with hydrodynamical runs that vary feedback and the cosmological model.

of simulated galaxies within this radius that have $M_{\rm gal} > 10^{10}$ M$_\odot$. (Simulated galaxies are defined as the stellar component of self-gravitating substructures identified with the SUBFIND algorithm.)

The soft and bolometric X-ray luminosity of each gas particle is provided with the simulation. For the tSZ signal, a quantity Υ is calculated for each gas particle (McCarthy et al. 2018),

$$\Upsilon \equiv \sigma_{\rm T}\frac{k_{\rm b}T}{m_{\rm e}c^2}\frac{m}{\mu_{\rm e}m_{\rm H}}, \tag{1}$$

where T is the gas particle's temperature, m is the gas particle's mass, $\mu_{\rm e}$ is the mean molecular weight per free electron of each gas particle, and $m_{\rm H}$ is the atomic mass of hydrogen.

2.2 Data set and image generation

The data sets used to train the neural networks are images of each of the four observables derived from the simulated cluster sample. The simulated clusters are provided at redshift 0, but we place them at random redshifts between 0.03 and 0.07 (with a uniform distribution) when we produce images. The cluster catalogue contains ~2000 clusters, but this is insufficient to train the neural network to the desired level of precision. To overcome this, we generate four images of each cluster by projecting along four different directions: x-, y-, z-axis and along $(\sqrt{2}/2, -\sqrt{2}/2, 0)$. To make the images of same cluster look more different, we rotate it with a random azimuthal angle before projecting. To properly include the correlated structure and foreground contamination, which are difficult to remove in real observation, we also project all the particles within 50 Mpc in front and 50 Mpc behind each cluster to the images. To make the images of the same cluster look more different from each other, all the clusters are rotated with a random angle around the line of sight before projection. In the end, we have ~8000 clusters at different redshifts with which to train the neural networks.

The image of each cluster is made by projecting it on to the x–y plane and binning the particles on to a 120 × 120 grid with an overall angular size of 20 arcmin. For a cluster at redshift z, the signal in pixel (i, j) for each observable is obtained as follows.

2.2.1 *Stellar density*

We evaluate the stellar surface density in each pixel as

$$I_{\rm ij} = \sum_{p\in({\rm i,j})} M_{\rm s}(\boldsymbol{r}_{\rm p})/S, \tag{2}$$

where the sum is over all stellar particles that project into pixel (i, j), $M_{\rm s}(\boldsymbol{r}_{\rm p})$ is the stellar mass of particle p (located at position $\boldsymbol{r}_{\rm p}$ with respect to the cluster centre), and S is the physical area of pixel (i, j). The angular coordinates of pixel (i, j) are

$$\boldsymbol{\theta}_p = (x_p, y_p)/d_{\rm A}(z), \tag{3}$$

where $(x_{\rm p}, y_{\rm p})$ are the x and y components of $\boldsymbol{r}_{\rm p}$ and $d_{\rm A}(z)$ is the angular diameter distance to redsfhit z.

2.2.2 *X-ray emission*

We convert luminosity into flux using $F = L/4\pi d_{\rm L}(z)$, where $d_{\rm L}(z)$ is the luminosity distance to redshift z. The signal in pixel (i, j) is the flux due to all gas particles that project into that pixel,

$$I_{\rm ij} = \sum_{p\in({\rm i,j})} F(\boldsymbol{r}_{\rm p}) = \sum_{p\in({\rm i,j})} \frac{L(\boldsymbol{r}_{\rm p})}{4\pi\, d_{\rm L}(z)}. \tag{4}$$

Table 1. Simulated data set properties. The labels are used throughout this paper.

Signal	Label	Units	Noise (*rms*)	Beam (FWHM)
Stellar mass	Star	M$_\odot$	2.14×10^{11}	–
Soft X-ray	Fxs	erg/s/cm^2	9×10^{-16}	4″
Bolometric X-ray	Fxb	erg/s/cm^2	9×10^{-16}	4″
Compton y	Ypar	–	10^{-8}	1.4′

2.2.3 *Compton* y *parameter*

The signal in pixel (i, j) is obtained by summing Υ/S (McCarthy et al. 2018) over all gas particles that project into that pixel,

$$I_{\rm ij} = \sum_{p\in({\rm i,j})} \Upsilon(\boldsymbol{r}_{\rm p})/S. \tag{5}$$

For low-mass clusters (those with $2\theta_{200} < 20'$), all cluster particles reside within the image, while for high-mass clusters, some particles extend outside the image and are lost. Our choice of image size strikes a balance between performance and computation time.

In order to mimic realistic data, we add noise to our images and smooth them to mimic a telescope point spread function. For stellar images, we take the rms to be 1/10 the mean mass across the whole sample giving a signal-to-noise ratio roughly 10, which mimics an SDSS-like observation(Abazajian et al. 2009). No smoothing is applied since most optical telescopes have a beam size smaller than our pixel size. The gas-based images have Gaussian random noise added and are then smoothed with a Gaussian beam. For the X-ray images, the rms noise and beam size are chosen to match the Chandra HRI sensitivity and full width at half-maximum (FWHM), respectively (Abazajian et al. 2009). For the Compton y image, the rms noise is taken to be 10^{-8} per pixel and the beam FWHM is 1.4 arcmin, corresponding to an *ACT*-like experiment (Hasselfield et al. 2013). We have also considered a Planck-like experiment with rms noise 10^{-6} and an FWHM of 9.66 arcmin (Aghanim et al. 2016), but the CNN performed quite poorly in this case. The parameters discussed above are summarized in Table 1. Images of each observable, in four clusters selected to have different masses, are shown in Fig. 2.

2.3 Artificial Neural Network

An ANN is a function that maps inputs to outputs,

$$\mathrm{ANN}\,(I) = O, \tag{6}$$

where I is the input and O is the output. In practice, the inputs can be images, sounds, text, etc., and the output can be a parameter to measure, a classification, and so on. A typical ANN is a sequential nest of functions that are defined on each layer of the network. In the simplest case of a feed-forward neural network, the neuron layers are evaluated in sequence, passing information from layer to layer. The output, a^l_k, of the k-th neuron in the l-th layer may be written as

$$a^l_k = f\left(\sum_j W^l_{jk}a^{l-1}_j + b^l_k\right), \tag{7}$$

where f is called an activation function (our choice of f is defined in the following section), W^l_{jk} is a matrix of weights, and b^l_k is a vector of additive biases. a^0_k is the input data, I, and a in the last layer is the output, O. Schematically, W^l_{jk} connects the j-th neuron

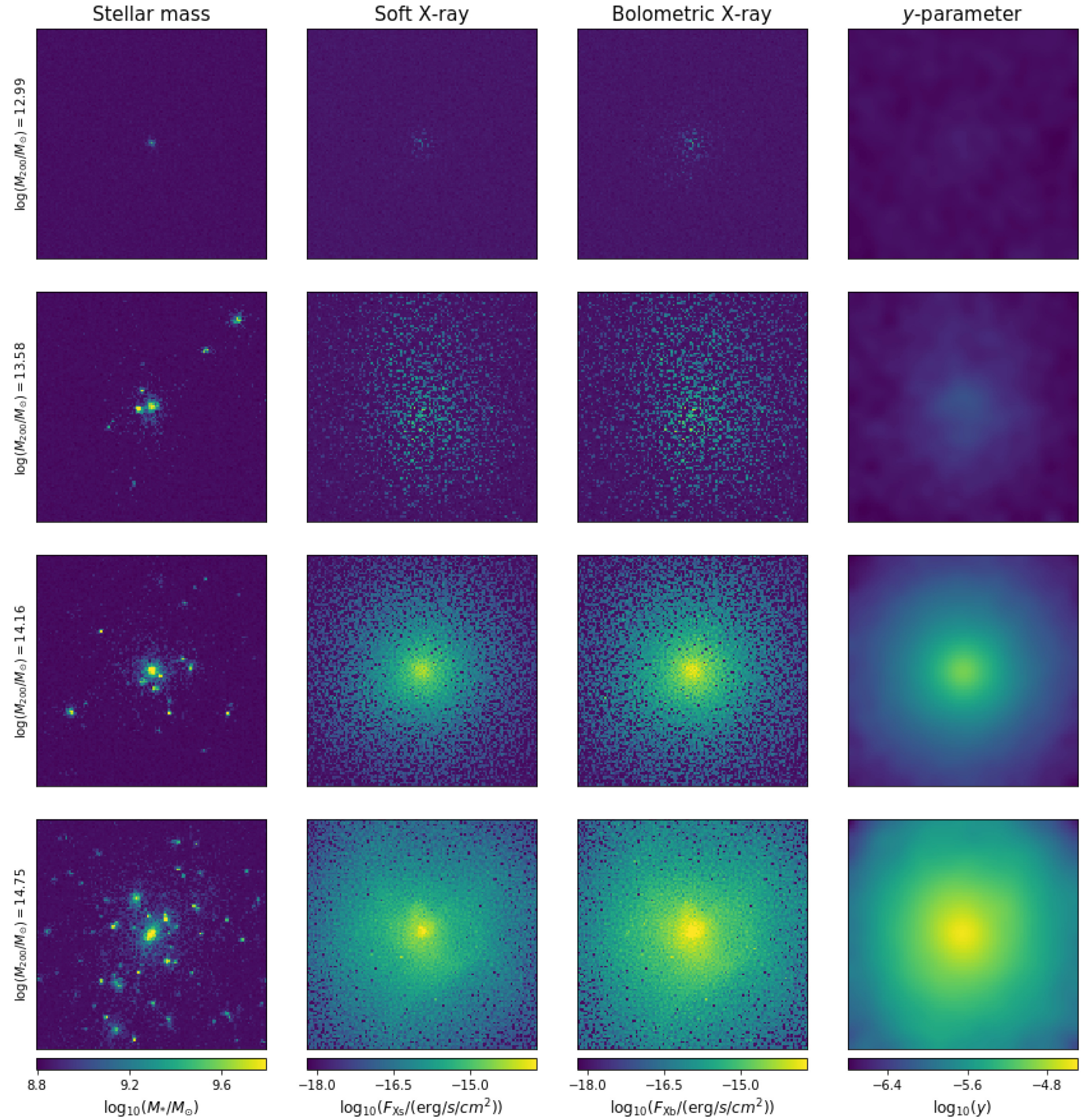


Figure 2. Selected cluster images from the BAHAMAS simulation. Each row is a cluster drawn from a different mass range, as indicated, and each column is a different observable: stellar density, soft X-ray luminosity, bolometric X-ray luminosity, and Compton y parameter. The colour scales are the same across the mass range (rows). The angular size of all the images is 20 arcmin.

in layer $(l-1)$ to the k-th neuron in layer l. The number of neurons and layers, or equivalently, the dimensions of W^l_{jk} and b^l_k are called the architecture of the ANN. Given the architecture and activation functions, the ANN is completely specified W^l_{jk} and b^l_k.

ANN training is a fitting procedure to determine the parameters W and b required to reproduce known information (so-called 'labels') from data. The labels can be categories (for a classification task) or quantities (for a regression task), and so on. For example, an ANN designed to recognize hand-written numbers is a classifier that takes hand-written images of numbers as input and generates numbers as output labels.

ANN training involves iteratively optimizing the weights so as to minimize the difference between the output labels and the known labels, as quantified by the loss function. The ANN is initialized with random weights and biases and then, during each iteration ('epoch'), the training data are provided to the ANN and outputs are predicted from them. The weights and biases are updated to reduce the loss function by an algorithm called an optimizer. The training is complete when the loss function converges. In order to validate the model, a 'validation set' (whose labels are also known) is needed. The loss function is calculated on the validation set during each epoch to monitor the training. The training is considered to be finished when the validation loss converges. After that, a 'test set' is then supplied to the ANN. If the ANN gives accurate predictions for the test set, then one can safely use it to predict labels from data whose labels are not known.

2.4 Convolutional Neural Network

In our analysis, we use a category of ANN called a Convolutional Neural Network (CNN) (see, for example, Aloysius & Geetha 2017 for a review on CNN). The typical input of a CNN is a two-dimensional image, and the CNN uses convolution layers to extract features from them (for example, textures, edges, gradual changes, and so on). Unlike fully connected layers, in which each neuron is connected to each of the previous neurons, convolutional layers pass forward information from a small neighbourhood around each neuron. A convolution layer comprises several filters, which are smaller than the input image. The filtered image is given by

$$I^F_{ij} = \sum_{i'j'} F_{i'j'} I_{i+i',j+j'}, \tag{8}$$

where the sum runs over the filter pixels, centred on $(i', j') = (0, 0)$. The output is called a feature image, and within the same convolutional layer, different filters extract different kinds of features (for example, horizontal and vertical textures). The parameters in a filter define a set of weights that are optimized during training. The feature images are downsized into a 'pooling layer', so the feature images get smaller as they pass through convolution-pooling layers. Different convolution layers can be designed to extract information on different scales by tuning the filter size and the feature image size. In our analysis, we take 3 × 3 square filters, which means in equation (8), i' and j' take the value $\{-1, 0, 1\}$. The pooling filter is 2 × 2 with a stride of 2 pixels. This means that each pixel in the pooling layer is the average of a 2 × 2 patch in the previous feature image with a stride of 2 pixels. This process is called 'average pooling', which downsizes the feature image by a factor of 2. As the feature images are downsized from layer to layer, the deeper convolution layers extract larger scale features. By using convolution and pooling layers, one can also reduce the computational cost and make the result easier to interpret. A sequence of convolution-pooling layers is flattened into a one-dimensional layer, followed by fully connected layers to further parametrize the features.

In our application, we utilize a CNN to predict cluster masses, so the output layer is a single neuron: the cluster mass. In the training set, we label each cluster with the M_{200} value calculated by summing the masses of all simulated particles within r_{200} of the cluster centre. In the rest of this paper, we denote this value as $M_{\rm true}$, and we denote the CNN-predicted mass by $M_{\rm pred}$. For each training run, we randomly select images of ∼ 4800 (60 per cent) simulated clusters as the training set, ∼ 1600 (20 per cent) as the validation set and the remaining ∼ 1600 (20 per cent) as the test set. The training and testing sets are carefully split so that we never train on a simulated cluster and then test on the same cluster as viewed from a different angle.

We train the four 'single-channel' CNNs with the four data sets described in Table 1. We can write

$$\texttt{CNN}^c\left(I^c_{ij}\right) = M_{\rm pred} \tag{9}$$

where I^c_{ij} is the image of tracer $c \in \{\text{Star, Fxs, Fxb, Ypar}\}$, (i, j) is the 2D pixel index, and $M_{\rm pred}$ is the predicted M_{200}. To assess the advantage of multiple tracers, we also train a 'multichannel' CNN, denoted $\texttt{CNN}^{mc}$, by simultaneously feeding all four data sets into one neural network,

$$\texttt{CNN}^{mc}\left(I^{\rm Star}_{ij}, I^{\rm Fxs}_{ij}, I^{\rm Fxb}_{ij}, I^{\rm Ypar}_{ij}\right) = M_{\rm pred}. \tag{10}$$

For each layer except the output layer, we use the Rectified Linear Unit as our activation function (Nair & Hinton 2010). This is defined as $f(x) \equiv \max\{0, x\}$. To prevent over-fitting, we force a 20 per cent dropout between fully connected layers (Srivastava et al. 2014), which means that, for each training epoch, 20 per cent of the weights (randomly selected) between those layers are set to zero. A dropout fraction slows down the training, so we chose this value to prevent over-fitting while keeping the training fairly fast. For the output layer, we use the mean-squared logarithmic error as our loss function, defined as

$$\delta \equiv \left\langle \left(\log M_{\rm pred} - \log M_{\rm true}\right)^2 \right\rangle. \tag{11}$$

During training, the loss function of the validation set (the so-called validation loss) is calculated in each epoch to monitor the progress of the training. Our convergence criterion is discussed below.

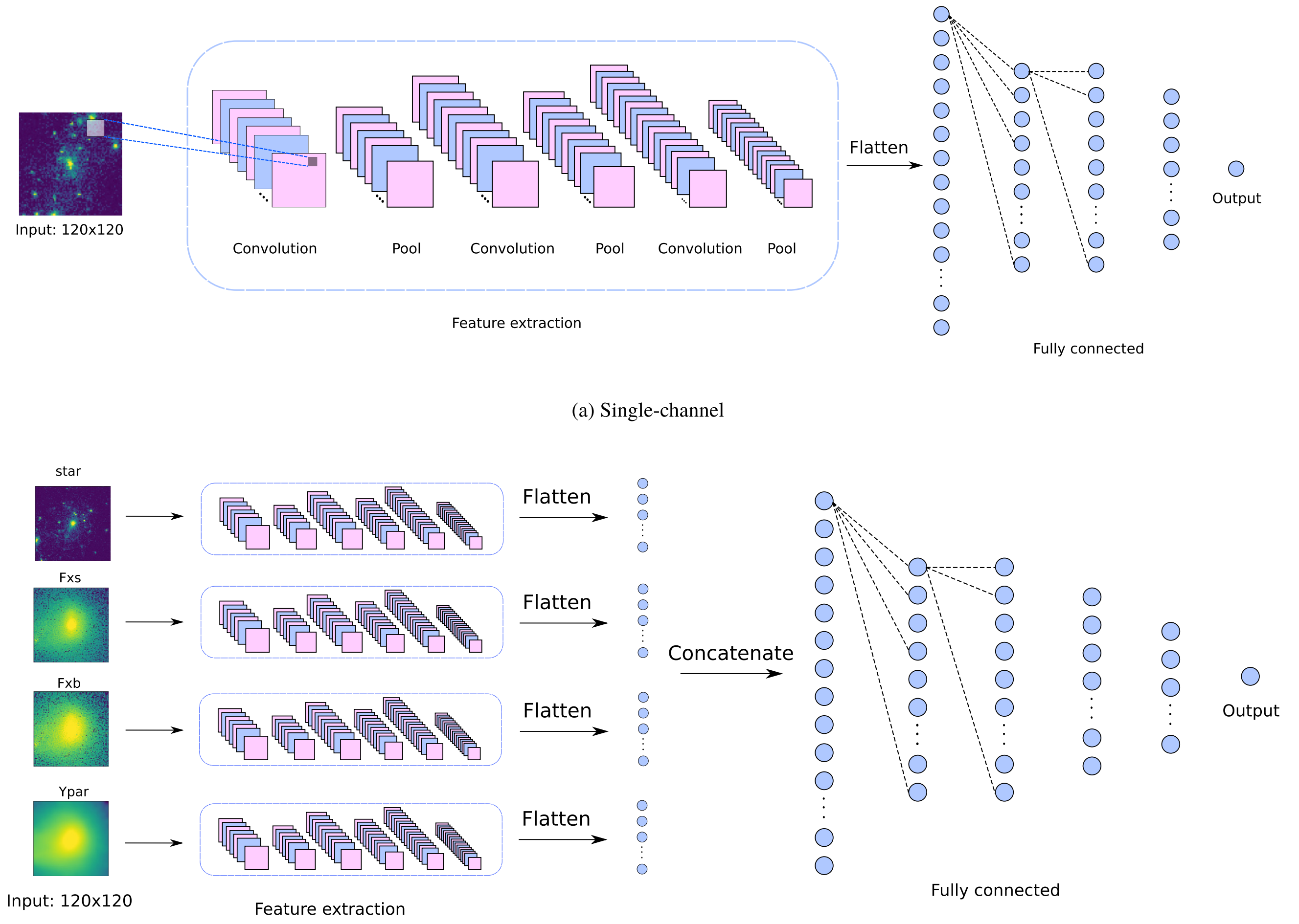
Our CNN is implemented using the `keras` package with a `Tensorflow` back end written in PYTHON. Our network architecture is similar to that used by Ntampaka et al. (2019), which is a simplified version of that used by Simonyan & Zisserman (2014):

(1) 3 × 3 convolution with 16 filters
(2) 2 × 2, stride-2 average pooling
(3) 3 × 3 convolution with 32 filters
(4) 2 × 2, stride-2 average pooling
(5) 3 × 3 convolution with 64 filters
(6) 2 × 2, stride-2 average pooling
(7) Flatten
(8) Fully connected with 200 neurons
(9) 10 per cent dropout
(10) Fully connected with 100 neurons
(11) 10 per cent dropout
(12) Fully connected with 100 neurons
(13) 10 per cent dropout
(14) Fully connected with 20 neurons
(15) Output neuron

For our multichannel network, each data channel is convolved, pooled, and flattened separately, using the same architecture as the single-channel network. As shown in Fig. 3(b), the flattened layers from each channel are concatenated into one flattened layer, followed by fully connected layers with the same architecture as the single-channel network.

We use `RMSprop` (Hinton, Srivastava & Swersky 2012) as our optimizer because it converges quickly in this application. We set the learning rate (the step size in the parameter space) to be 0.01 with decay rate of 10^{-4}. We tested other combinations of optimizers and learning rates, but this choice gave the best performance. The training data are divided into batches of 50 images each. In one training epoch, the network is trained through each batch separately, and the CNN weights are obtained by averaging over all batches. Each network was trained for 1000 epochs on 2 GPUs with 6 CPUs. The training took about 20 min for a single-channel network, and 45 min for the multichannel network.

Fig. 4 shows the 'learning curve' (validation loss as a function of training epoch) for each of our CNNs. During training, the validation



(a) Single-channel

(b) Multi-channel

Figure 3. Upper panel: Architecture of the single-channel CNN used in this analysis. Our network utilizes three convolutional and pooling layers for feature extraction and four fully connected layers for parameter estimation. Lower panel: Architecture of the multichannel CNN. The four channels take images: Star, Fxs, Fxb, and Ypar, respectively, and perform feature extraction independently. The feature extraction layers have the same structure as the single-channel portion outlined in the upper panel.

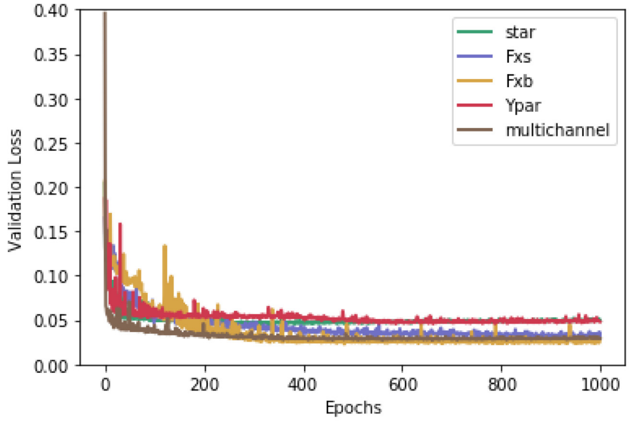


Figure 4. The learning curves for each of our five CNNs as a function of training epoch. The *y*-axis is the loss function on the validation set defined in equation (11).

loss drops quickly at first and then converges after ∼600 epochs. The final CNN weights are taken to be those that gave the minimum validation loss during training.

3 RESULTS

Our cluster mass predictions are shown in Fig. 5. For each cluster in the test set, we show the CNN-predicted mass versus the true M_{200} measured in the simulation. In this rendition, all five data sets produce similar results. Fig. 6 shows the fractional mass bias for each tracer as a function of the true mass. From Fig. 6, we see a clear tendency that the CNN generally *over*-predicts the mass by ∼20 per cent in the lowest mass bin, while it *under*-predicts the mass by ∼10 per cent in the highest mass bin. The is due to the fact that for these extreme masses, there are not enough samples, so the CNNs tend to predict towards the mean mass of the whole sample. To mitigate this ‘towards-the-mean’ bias, one needs to extend the mass range of training set than the test set, or alternatively, only trust the results of test clusters with masses close to the mean.

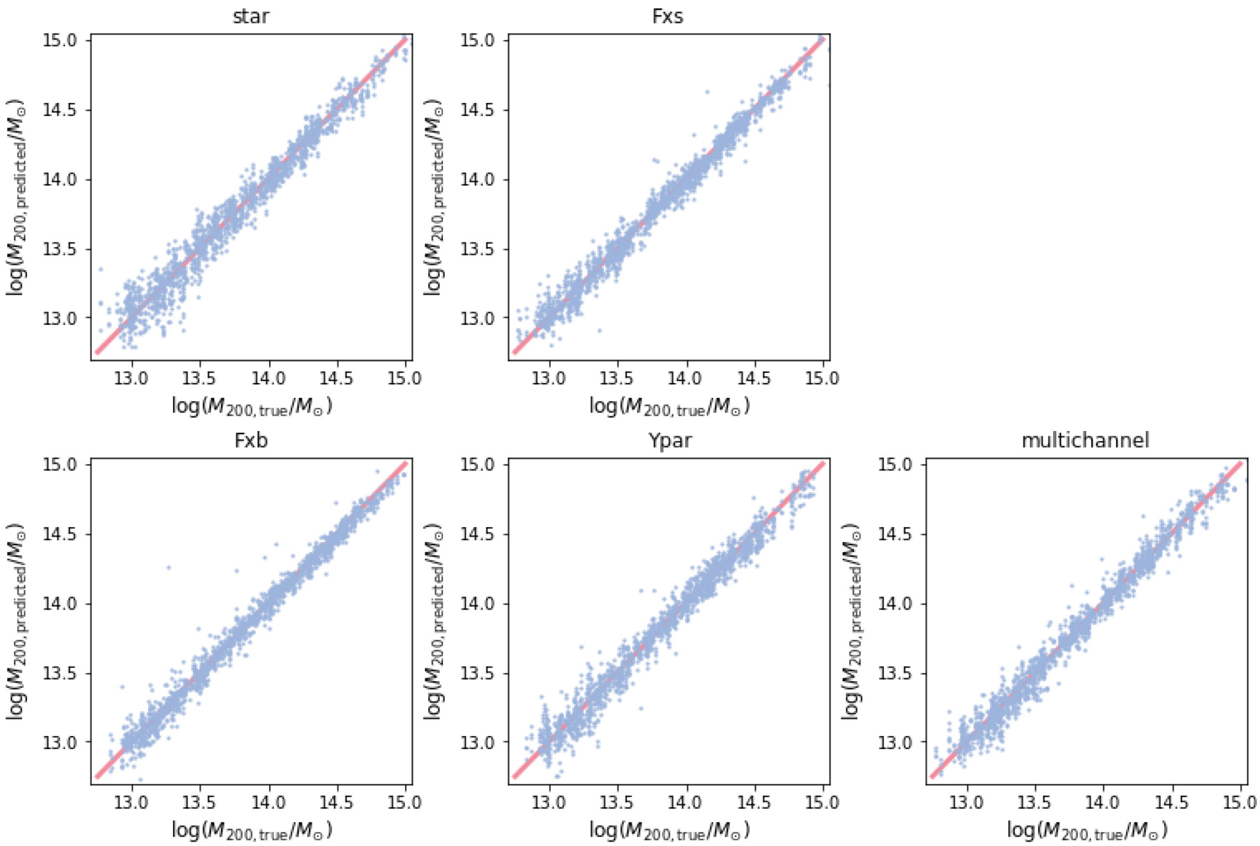


Figure 5. The CNN-predicted cluster mass versus the true mass for each cluster in the test set. Each tracer is shown separately for clarity.

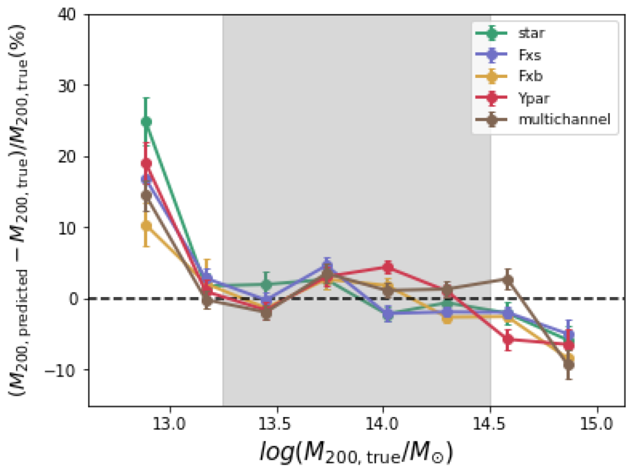


Figure 6. The bias in our CNN-predicted cluster masses as a function of the true mass for each tracer. The plotted uncertainties show the standard deviation of the bias in each bin. The bias is small within the central mass range of $13.25 < \log{(M_{200,\,\mathrm{true}}/\mathrm{M}_\odot)} < 14.5$ (the shaded region).

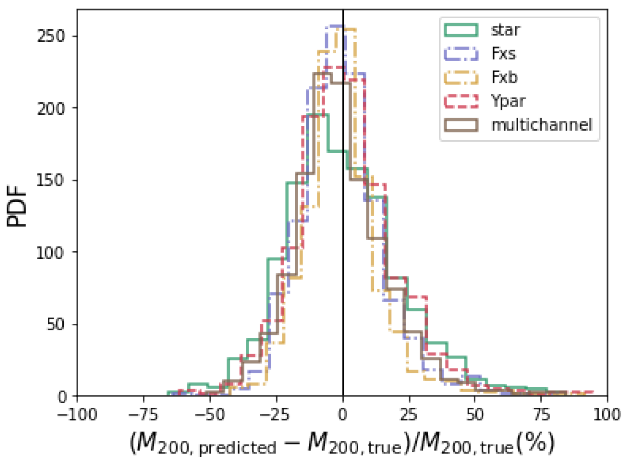


Figure 7. The probability distribution of the mass bias for each tracer for mass bins in the range $13.25 < \log{(M_{200,\,\mathrm{true}}/\mathrm{M}_\odot)} < 14.5$ (the shaded region in Fig. 6).

Within the central mass range of $13.25 < \log{(M_{200,\,\mathrm{true}}/\mathrm{M}_\odot)} < 14.5$ (the shaded region in Fig. 6), the mass bias is quite small. Histograms of the mass bias in these central bins are shown in Fig. 7. Each tracer is plotted as a separate colour, with the gas-based tracers plotted as dashed curves, for clarity. A summary of our numerical results, both the average bias and the rms scatter, is given in Table 2.

The average mass bias, $\Delta M/M_{\mathrm{true}}$, in the central mass bins is on the order of 1 per cent with an uncertainty of ∼0.5 per cent. The uncertainty per individual cluster is of order 15 per cent (Table 2). Somewhat surprisingly, the multichannel network is not the most precise. We assume that this is due to limitations in the CNN architecture to synthesize information across all four tracers.

Armitage et al. (2019) apply an ML method on cluster masses and report a 7 per cent mass scatter. However, they use multiple observables derived from simulations to train their model. These observables may suffer from uncertainty in real observation. In addition, they do not include observational effects such as instrument

Table 2. The mean mass bias ($\Delta M \equiv M_{\rm pred} - M_{\rm true}$) and scatter obtained from the test set for $13.25 < \log{(M_{200,\,{\rm true}}/{\rm M}_\odot)} < 14.5$.

Data set	$\left\langle \log \frac{M_{\rm predict}}{M_{\rm true}} \right\rangle$	$\left\langle \frac{M_{\rm predict}}{M_{\rm true}} \right\rangle - 1(\%)$	$\langle$rms$\rangle$
Star	− 0.01 ± 0.003	− 0.516 ± 0.621	19.028
Fxs	− 0.007 ± 0.002	− 0.349 ± 0.517	16.49
Fxb	− 0.004 ± 0.002	0.094 ± 0.524	16.036
Ypar	0.002 ± 0.002	1.814 ± 0.559	17.662
Multichannel	− 0.001 ± 0.002	1.075 ± 0.575	17.693

noise or beams, which will degrade the performance of the mass estimation. Henson et al. (2016) evaluate the performance of conventional mass estimation techniques applied to the BAHAMAS hydrodynamical simulations. They fit azimuthally averaged shear profile of each simulated cluster with both NFW and Einasto models with the cluster mass as a free parameter. By comparing best-fitting mass with the true cluster mass, they find mass biases of $\Delta M/M_{\rm true} = -8.9^{+0.3}_{-0.2}$ per cent, and $-6.4^{+0.3}_{-0.2}$ per cent for the NFW and Einasto profile, respectively. Yan et al. (2020) analyse the same BAHAMAS catalogue by fitting an NFW model to the density profile of all particles in a cluster and found a mean mass bias of -10 per cent. These studies are based on weak lensing profiles, which is an unbiased tracer of the cluster masses, while the present study uses biased tracers like galaxy or gas. Moreover, the previous studies do not include observational effects such as noise and smoothing. We conclude that our CNN-based results are more accurate than these profile-based analyses performed on the same hydrodynamical simulation even with biased tracers, possibly due to limitations in the profile models they use. As we will see in the next section, CNN is capable of extracting shapes, orientations, and substructures from 2D cluster images, which contains more information about the cluster masses.

As a reference to real observation, Zhang et al. (2008) use scale relations of X-ray observations to evaluate real cluster masses and get an individual mass uncertainty of ∼ 30 per cent; Bleem et al. (2015) also use scale relation of tSZ signal to estimate mass for South Pole Telescope (SPT) galaxy clusters and get a mass uncertainty of ∼ 24 per cent for each cluster. Hoekstra et al. (2015) use weak lensing techniques to evaluate the masses of clusters. They estimate an uncertainty of about 20 per cent which, if correct, indicates that the precision of our CNN-based method is not significantly better than weak lensing analysis. However, the weak lensing analysis is generally performed on more massive clusters that are not readily available in our simulation, so the comparison is not completely apt.

There are three causes of the mass bias of our CNN prediction: (1) the underlying scatter between cluster mass and morphology correlation; (2) the observational biases caused by smoothing and noise; and (3) the imperfection of our CNN algorithm. The first cause is not able to overcome by observation; the second one could be suppressed by carefully handling instrument systematics; the third one could be suppressed by improving the CNN architecture and training setup. In addition, to apply this method on real observation, one needs to take care of the difference between simulation and observation. This can be done by either comparing simulated galaxy clusters with real clusters or introducing real data in the testing set, which are left for future work.

In order to evaluate the impact of the foreground and background interlopers, we also train a set of CNNs with images that do not have fore- and background. The scatter of mass bias is lower than our fiducial results by ∼ 2 per cent. This indicates that the presence of uncorrelated structure along the line of sight has only a marginal impact on our results.

4 INTERPRETING THE CNN PERFORMANCE

Cluster masses are traditionally estimated using scaling relations based on known physics. For example, in relaxed clusters, X-ray luminosity is related to cluster mass via the virial theorem. In contrast, neural networks contain a large number of parameters, which makes their behaviour difficult to interpret. What makes the network predict a particular value of mass? What cluster feature(s) is it sensitive to? In this section, we attempt to interpret our single-channel networks in two ways.

4.1 Deep Dream

Google's Deep Dream (DD) (Mordvintsev, Olah & Tyka 2015) is an iterative, gradient ascent algorithm that is applied to an input image to determine which image pixels affect a particular output neuron the most. In our application, we have one output neuron, $M_{\rm pred}$, so DD may be expressed in the form

$$I_{ij}^{(p)} = I_{ij}^{(p-1)} + \alpha \left. \frac{\partial M_{\rm pred}}{\partial I_{ij}} \right|_{I^{(p-1)}}, \qquad (12)$$

where $I_{ij}^{(p)}$ is the image at the p-th iteration of the algorithm, and α is the step size. For small α, the difference between successive image iterations is proportional to the gradient of $M_{\rm pred}$ with respect to the image.

We ran one iteration of DD for each data tracer, selecting images from a range of true cluster masses. (We also ran two iterations, as did Ntampaka et al. (2019), and found similar results.) The gradient images for these examples are shown in the middle columns of Figs 8–11. In the stellar mass examples, the pixels that affect the predicted mass the most (rendered in yellow) appear to lie mostly *adjacent* to the galaxy locations. This suggests that $\texttt{CNN}^{\rm star}$ is mainly triggering on the number and the size of galaxies in the image.

The gas-based tracers are more diffuse and symmetric, and this is reflected in the DD gradient images. For the X-ray tracers, the gradient images are quite granular, reflecting the granularity of the input images. But the critical information captured by the CNNs appears to be the shape of a cluster. For example, in the third row of Fig. 9, we see that the $F_{\rm xs}$ image has substructure at the top-right, which is also seen in the gradient image. The central regions appear to be relatively uninformative, in agreement with the conclusions of Ntampaka et al. (2019). In addition, the lack of sensitivity of the central region with the DD images generally mimics the shape of the cluster itself (it is clear, for instance, in the lowest two rows of Fig. 10). For the Compton y images, the DD gradient images show two contours at different radius for massive clusters, both sketching the outskirt of the cluster without the fine granularity seen in the X-ray tracers.

4.2 Gaussian mask

A somewhat complimentary approach to interpreting the CNN performance is to examine the predicted mass when selected regions of the image are masked. For this study, we define a Gaussian mask in the image plane as

$${\rm Mask}_{ij} = 1 - \exp\left[-\frac{(i-a)^2 + (j-b)^2}{2\sigma^2} \right], \qquad (13)$$

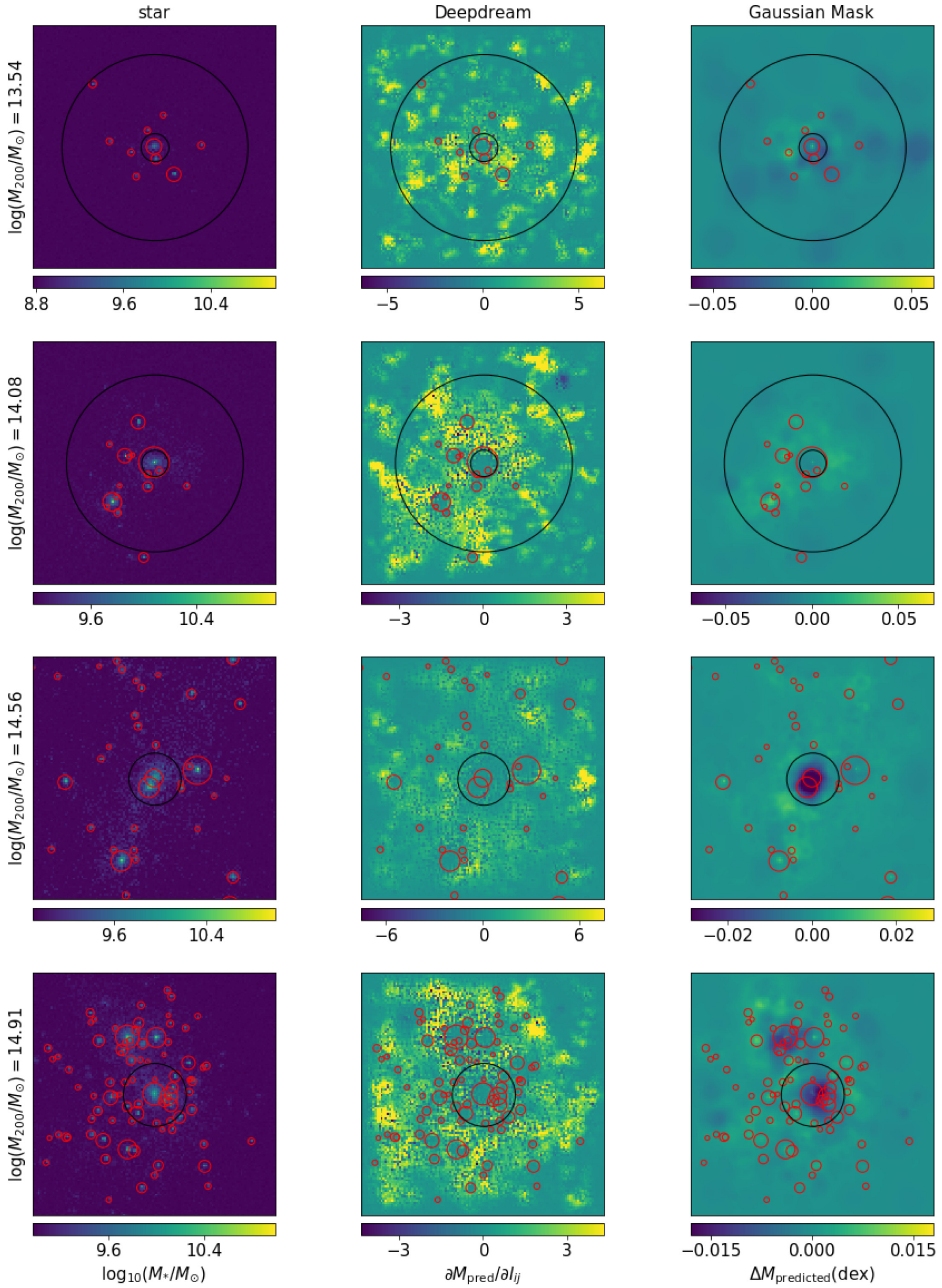


Figure 8. *Left column*: stellar mass images of four galaxy clusters selected to cover our mass range; *middle column*: the relative signal change, $\propto \Delta M_{\rm pred}$, after two Deep Dream iterations; *right column*: the signal change, $\Delta M_{\rm pred}$, when masking the image with a Gaussian mask centred, in turn, on each image pixel (see text for details). The inner black circles indicate $0.15R_{200}$, while the outer circles indicate R_{200}. The red circles highlight galaxy positions, with radii that are proportional to the galaxy's mass.

where a and b define the centre of the mask in pixel coordinates, and we take $\sigma = 5$ pixels, corresponding to 1.25 arcmin for our images. For each a and b in the image plane, we multiply the original image by this mask and then use the pre-trained CNN to (re)predict the cluster mass, $M_{\rm pred}(a, b)$. The results of this test are presented in the right column of Figs 8–11, in the form of images of $\Delta M_{\rm pred}(a, b)$, the change in predicted mass when pixels in the neighbourhood of pixel (a, b) of the corresponding cluster image are masked. Pixels that contribute significantly to the original mass estimate will produce a negative ΔM when masked.

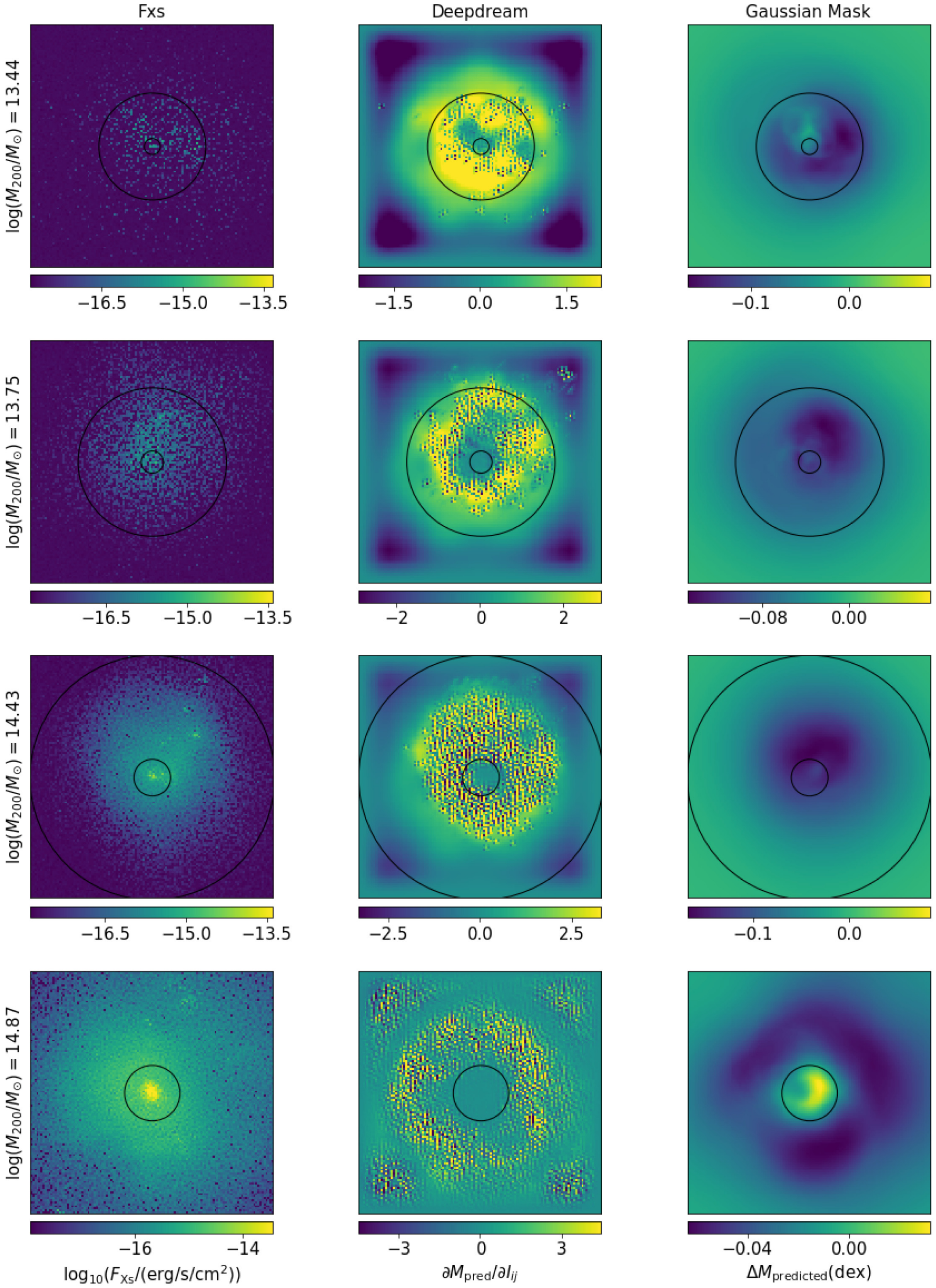


Figure 9. Left column: soft X-ray images of four galaxy clusters; middle column: change of signal after two Deep Dream iterations; right column: change of mass prediction when masking the image with Gaussian masks centred at each pixel. The inner circles show the radius of $0.15R_{200}$ and the outer circles R_{200}.

For the stellar images, masking the central galaxy reduces the predicted mass dramatically, as we might expect. Beyond the central galaxy, the effects are much less clear. There is some mild anticorrelation between the masked image and the DD image, as we might expect, but these are lower level effects compared to the central region.

For the X-ray-based tracers, the mask analysis shows that the outskirt of the X-ray data, where the signal gradient is largest, appears to be the most decisive feature the CNN triggers on. For Compton y images, the mask analysis shows that the central region plays an important role but it fails to capture the details of the cluster. In summary, the mask method generally agrees with the DD analysis but is less informative, probably because the mask removes a fairly large region with details about cluster structure.

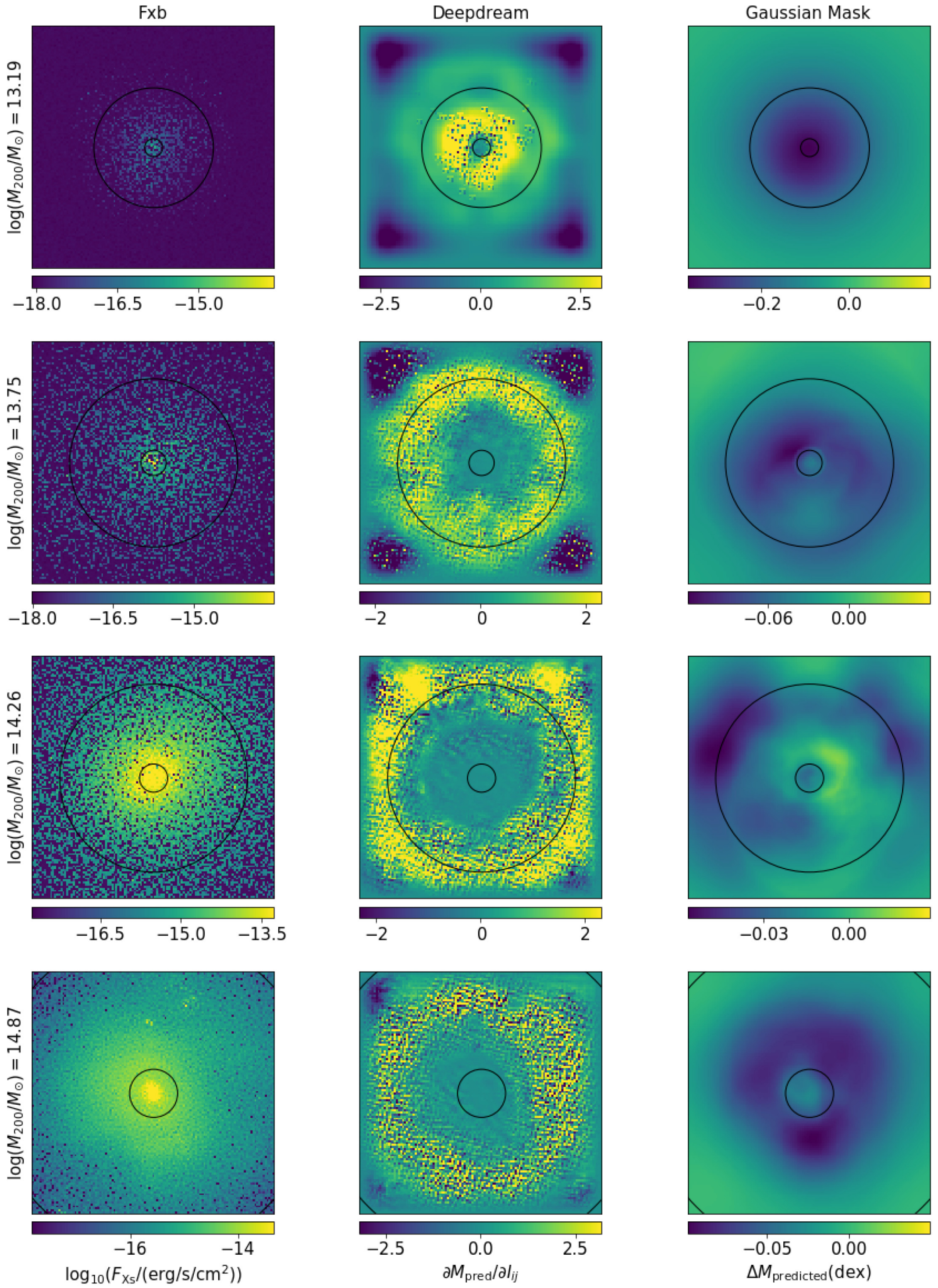


Figure 10. Left column: bolometric X-ray images of four galaxy clusters; middle column: change of signal after two Deep Dream iterations; right column: change of mass prediction when masking the image with Gaussian masks centred at each pixel. The inner circles show the radius of $0.15R_{200}$ and the outer circles R_{200}.

As with the DD analysis, the CNN seems to be relatively insensitive to central regions. We attribute this to the observation that the signal in the central region is more scattered with respect to cluster mass (Mantz et al. (Maughan 2007; Mantz et al. 2018). We quantify this by calculating the correlation between the true mass and $F_{\rm cent}$ on the one hand, and $F_{\rm ring}$ on the other, where $F_{\rm cent}$ is the integrated X-ray signal in the range $r < 0.15R_{200}$ and $F_{\rm ring}$ is the integrated signal in the range $0.15R_{200} < r < R_{200}$. The former has a correlation coefficient of 0.58 while the latter is 0.94 (for both $F_{\rm xs}$ and $F_{\rm xb}$).

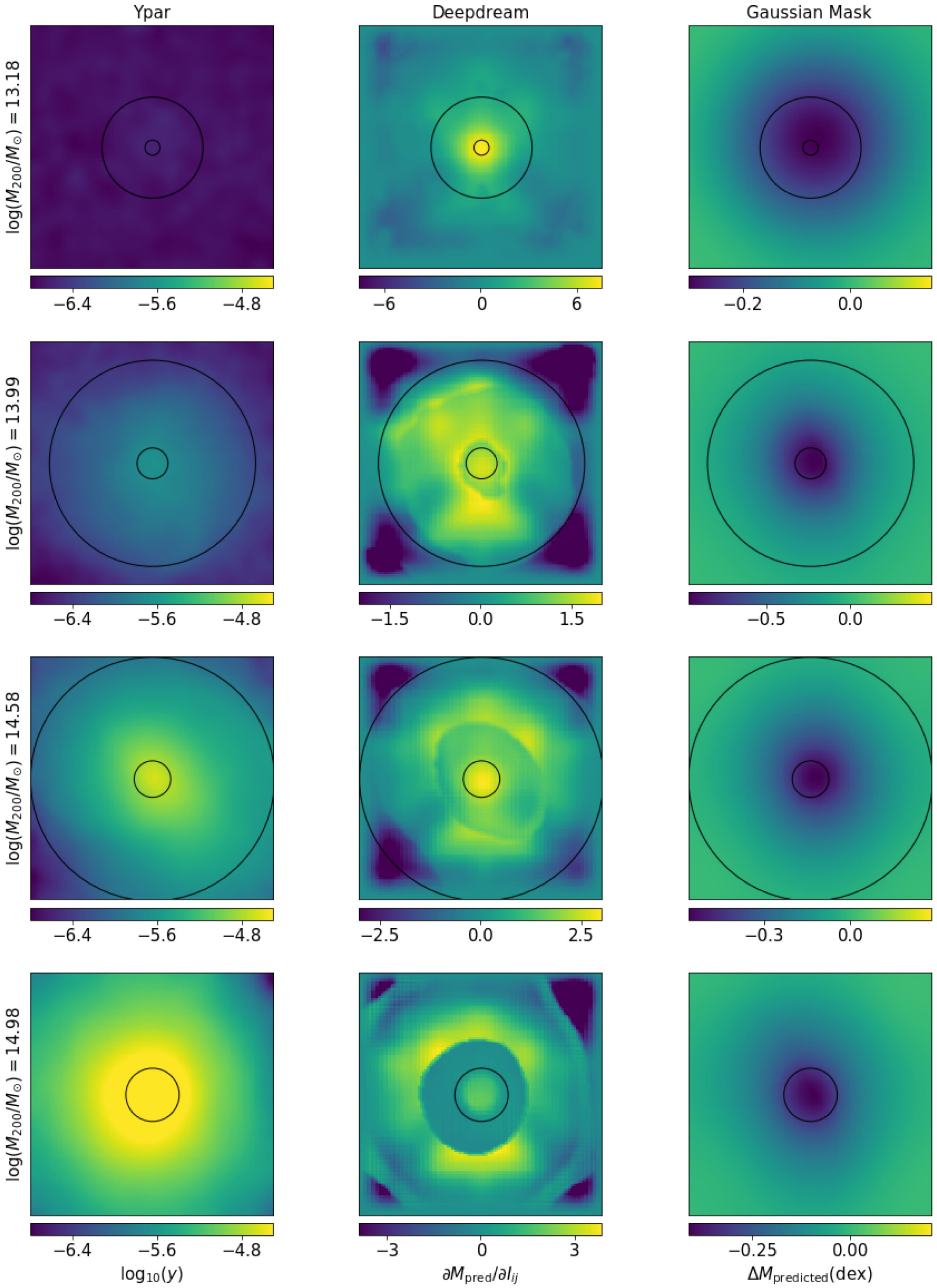


Figure 11. Left column: y parameter images of four galaxy clusters; middle column: change of signal after two Deep Dream iterations; right column: change of mass prediction when masking the image with Gaussian masks centred at each pixel. The inner circles show the radius of $0.15R_{200}$ and the outer circles R_{200}.

5 CONCLUSION

We construct and train a set of CNNs to predict galaxy cluster masses and test the network using cluster catalogues derived from the BAHAMAS hydrodynamical simulations. The clusters used in our study range in mass from $10^{12.7}$ to $10^{14.8}$ $\mathrm{M}_{\odot}$. Using the simulation data base, we generate mock data sets of stellar mass, soft X-ray flux, bolometric X-ray flux, and Compton y-parameter images as input, and train four single-channel networks on each of these observables independently. Each network has three convolutional layers and three pooling layers for feature extraction, followed by five fully connected layers. We also construct a multichannel network that takes all four data sets as simultaneous input. The multichannel network is configured to run the four single-channel feature extraction sections independently. The output is then concatenated and processed by six fully connected layers. We train each network with 4800 randomly selected cluster images and validate our training using ∼1600 validation images. The pre-trained network is then tested with ∼1600 test images.

Our results are presented in Section 3. All five of the networks successfully learn to predict cluster masses from mock data images. In the mass range $10^{13.25}\,\mathrm{M}_\odot < M < 10^{14.5}\,\mathrm{M}_\odot$, our networks predict the true mass with a mean bias that is of order of 1 per cent. Outside of this range, our networks tend to over-predict the mass of low-mass clusters and under-predict the mass of high-mass clusters, which reflects a tendency towards mean. The per-cluster rms scatter is ∼15 per cent, with the Compton y parameter and soft X-ray networks giving modestly lower scatter than the rest. This performance is better than X-ray and tSZ-based analysis like Zhang et al. (2008) and Bleem et al. (2015) while comparable to the weak lensing analysis of real data reported by Umetsu et al. (2014). However, we note that weak lensing profiles bear richer mass information than the tracers we study, and current weak lensing studies focus on higher mass clusters. Future work applying CNNs to simulated weak lensing images would be needed to make a fairer comparison.

Henson et al. (2016) estimate cluster masses in the BAHAMAS simulation by fitting the weak lensing profiles of all particles with empirical models. They find a comparable mass bias to ours; however, they do not include noise, so the results are not directly comparable. So, we conclude that our method is more accurate than previous methods that used the same hydrodynamical simulation, although the overall precision does not improve significantly. Although we have considered realistic systematics that could affect the performance of CNN, including beam smoothing, instrumental noise, and additional structure along the line of sight, systematics in real observations are generally more complicated than what we have included in our analysis. Particularly, by introducing fore- and background correlated signals, the mass scatter gets higher by ∼ 2 per cent. We want to emphasize that it is important to include such systematics in future deep learning-related studies concerned with galaxy clusters mass estimates. We also note that the data set in our analysis is still idealized compared to real observational data.

We use two diagnostics to interpret the performance of our trained networks. Both of them aim to identify image features that ‘trigger’ the network to reach a particular conclusion. The stellar mass CNN clearly detects galaxies and takes them into account when predicting cluster masses. The gas-based CNNs apparently trigger on the shape and alignment of the gas, but the details are elusive. For example, the X-ray-based CNNs treat the cluster outskirts more importantly than the central region [in agreement with Ntampaka et al. (2019)]. The reason might be that cluster cores are known to be significantly scattered with mass, so the neural networks choose to ignore the central region for an optimal prediction. Similar future ML work could take this fact into account and down weight the central part by hand (by cutting out central region in pre-processing, for example).

This paper demonstrates a new approach to measuring galaxy cluster masses, a key parameter for understanding the origin and evolution of large-scale structure in the universe. We show that a CNN is capable of recovering cluster masses directly from images of observable signals, despite the presence of substructure and noise. Our method does not require a physical model; however, it does require that one be able to simulate realistic clusters and systematic measurement errors reliably. Future work might aim to train networks by combing data from different simulations, or even from real data. We caution that neural networks are notoriously difficult to interpret, so future work should aim to better understand the behaviour of hidden layers of the network.

ACKNOWLEDGEMENTS

We thank Dr. Shirley Ho, Nayyer Raza, Riley Hill, Dr. Tilman Tröster, and Qiufan Lin for fruitful discussions. The calculations were performed on Compute Canada nodes (www.computecanada.ca). AJM received funding from the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement no. 702971. This work is financially supported by University of British Columbia and NSERC. This project has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement no. 769130).

DATA AVAILABILITY

The galaxy cluster data from BAHAMAS simulation used in this paper will be shared on reasonable request to the corresponding author.

REFERENCES

Abazajian K. N. et al., 2009, ApJS, 182, 543
Aghanim N. et al., 2016, A&A, 594, A22
Allen S. W., Schmidt R. W., Fabian A. C., 2002, MNRAS, 334, L11
Allen S. W., Evrard A. E., Mantz A. B., 2011, ARA&A, 49, 409
Aloysius N., Geetha M., 2017, in 2017 International Conference on Communication and Signal Processing (ICCSP). IEEE, Chennai, p. 0588
Armitage T. J., Kay S. T., Barnes D. J., 2019, MNRAS, 484, 1526
Bahé Y. M., McCarthy I. G., King L. J., 2012, MNRAS, 421, 1073
Banerji M. et al., 2010, MNRAS, 406, 342
Baron D., 2019, preprint (arXiv:1904.07248)
Becker M. R., Kravtsov A. V., 2011, ApJ, 740, 25
Bleem L. et al., 2015, ApJS, 216, 27
Borgani S., Kravtsov A., 2011, Adv. Sci. Lett., 4, 204
Cohn J., Battaglia N., 2020, MNRAS, 491, 1575
Gitti M., Brighenti F., McNamara B. R., 2012, Adv. Astron., 2012, 950641
Green S. B., Ntampaka M., Nagai D., Lovisari L., Dolag K., Eckert D., ZuHone J. A., 2019, ApJ, 884, 33
Gupta N., Reichardt C. L., 2020, ApJ, 900, 110
Hasselfield M. et al., 2013, ApJS, 209, 17
Henson M. A., Barnes D. J., Kay S. T., McCarthy I. G., Schaye J., 2016, MNRAS, 465, 3361
Hinshaw G. et al., 2013, ApJS, 208, 19
Hinton G., Srivastava N., Swersky K., 2012, Cited on, 14, 8
Hoekstra H., Herbonnet R., Muzzin A., Babul A., Mahdavi A., Viola M., Cacciato M., 2015, MNRAS, 449, 685
Lanusse F., Ma Q., Li N., Collett T. E., Li C.-L., Ravanbakhsh S., Mandelbaum R., Póczos B., 2017, MNRAS, 473, 3895
Leauthaud A. et al., 2011, ApJ, 744, 159
McCarthy I. G., Schaye J., Bird S., Le Brun A. M. C., 2017, MNRAS, 465, 2936
McCarthy I. G., Bird S., Schaye J., Harnois-Deraps J., Font A. S., van Waerbeke L., 2018, MNRAS, 476, 2999
Mantz A. B., Allen S. W., Morris R. G., von der Linden A., 2018, MNRAS, 473, 3072
Maughan B., 2007, ApJ, 668, 772
Melin J.-B., Bartlett J. G., 2015, A&A, 578, A21
Melin J.-B., Bartlett J., Delabrouille J., Arnaud M., Piffaretti R., Pratt G., 2011, A&A, 525, A139
Mordvintsev A., Olah C., Tyka M., 2015, Google Res., 2, 5
Moster B. P., Somerville R. S., Maulbetsch C., Van Den Bosch F. C., Macciò A. V., Naab T., Oser L., 2010, ApJ, 710, 903
Nagai D., Lau E. T., 2011, ApJ, 731, L10
Nair V., Hinton G., 2010, Proceedings of the 27th international conference on machine learning, Vol. 27, ICML-10, Haifa, p. 807
Ntampaka M. et al., 2019, ApJ, 876, 82

Peacock J., Smith R., 2000, MNRAS, 318, 1144

Planck Collaboration XXIV, 2016, A&A, 594, A24

Pratt G. W., Croston J. H., Arnaud M., Böhringer H., 2009, A&A, 498, 361

Ribli D., Pataki B. Á., Csabai I., 2019, Nature Astron., 3, 93

Rodríguez A. C., Kacprzak T., Lucchi A., Amara A., Sgier R., Fluri J., Hofmann T., Réfrégier A., 2018, Comput. Astrophys. Cosmology, 5, 4

Schaye J. et al., 2010, MNRAS, 402, 1536

Seljak U., 2000, MNRAS, 318, 203

Shan H. et al., 2012, ApJ, 748, 56

Simonyan K., Zisserman A., 2014, preprint (arXiv:1409.1556)

Smith G. P. et al., 2016, MNRAS, 456, L74

Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R., 2014, J. Mach. Learn. Res., 15, 1929

Sun M., Voit G. M., Donahue M., Jones C., Forman W., Vikhlinin A., 2009, ApJ, 693, 1142

Umetsu K., 2010, preprint (arXiv:1002.3952)

Umetsu K. et al., 2014, ApJ, 795, 163

Voit G. M., 2005, Rev. Mod. Phys., 77, 207

von der Linden A. et al., 2014, MNRAS, 439, 2

Yan Z., Raza N., Van Waerbeke L., Mead A., McCarthy I., Tröster T., Hinshaw G., 2020, MNRAS, 493, 1120

Zhang Y.-Y., Finoguenov A., Böhringer H., Kneib J.-P., Smith G., Kneissl R., Okabe N., Dahle H., 2008, A&A, 482, 451

Zu Y., Mandelbaum R., 2015, MNRAS, 454, 1161

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