Progress report: Deep learning halos

Luisa Lucie-Smith

February 25, 2020

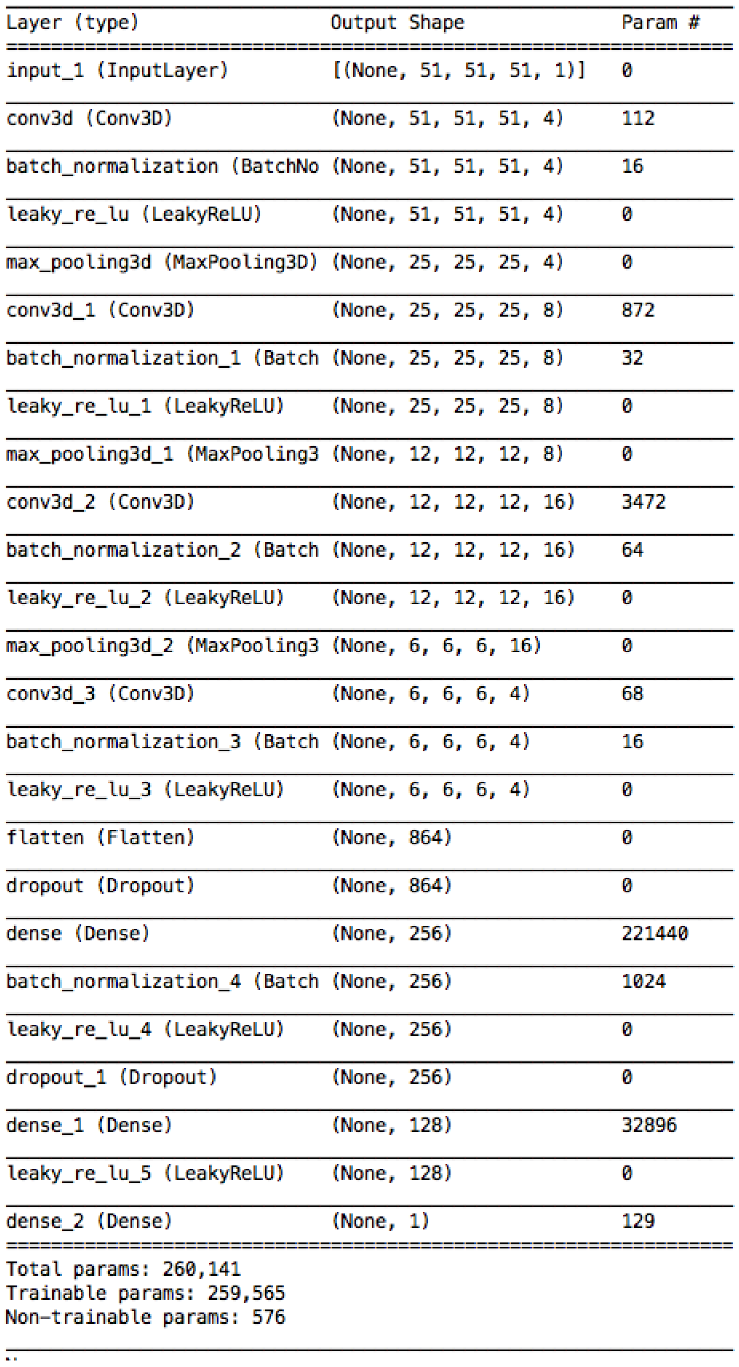
1 Predicting halo mass from the initial conditions

1.1 The training set, validation set and training procedure

The training set is made of $100,000$ particles consisting of $20,000$ particles from each of five simulations based on different initial conditions realizations. The validation set is made of $20,000$ particles from one independent simulation from those used for training.

The input to the CNN is given by the initial density field sampled in a 3D box of comoving length $L_{\rm box} \sim 10\,{\rm Mpc}\,h^{-1}\,a$ and made of $N = 51^3$ voxels, centred on each particle's initial position. The resolution of the 3D input is set to be equal to the resolution of the simulation's grid, meaning that the length of each voxel, $l_{\rm voxel}$, is the same as the initial grid spacing in the simulation i.e., $l_{\rm voxel} = 0.2{\rm Mpc}\,h^{-1}\,a$. Since the length of the voxel is fixed, the choice of $N = 51^3$ sets the volume of the sub-box. This choice can be straightforwardly changed to be a larger value (e.g. $N = 75^3$) since the sub-boxes are computed at the time each particle is called for training. The output of the CNN is given by the logarithmic mass of the halo to which each particle belongs at $z = 0$. Both the inputs and outputs are rescaled in the training set such that their distributions have zero mean and unit variance. The validation and test sets will then also be rescaled by the same means and variances.

A summary of the CNN architecture is illustrated in Fig. 1. The "layer" column shows the sequence of layers in the network, the "output shape" column shows the dimensions of the layer's output in the form (batch size, x-dimension, y-dimension, z-dimension, channel-dimension). A batch size of None means that the algorithm can take as input any batch size. The first layer is a convolutional layer, where the stride and the padding are chosen such that the spatial dimensions of the outputs are equal to the spatial dimensions of the inputs. For example after the first conv layer, the spatial dimensions of the output (51^3) match those of the inputs. The channel dimension increased from 1 to 4 because I chose to convolve the input with four kernels. Note that the number of kernels in any convolutional layer is a hyperparameter that is set a priori by the user. The number of kernels represents the number of features that the CNN can learn from the input. Each convolutional layer is followed by a batch normalization layer, a non-linear activation function and a pooling layer. The pooling layer reduces each spatial dimension by a factor of two. The forth convolutional layer (called "conv3d-3") uses a kernel size of $1 \times 1 \times 1$ – its purpose is simply to reduce the channel dimension from 16 to 4. In general, convolutional layers of $1 \times 1 \times 1$ kernels are used for dimensionality reduction in the channel dimension. Before the fully-connected (FCC) layers, the inputs are flattened to



```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 51, 51, 51, 1)]   0
_________________________________________________________________
conv3d (Conv3D)              (None, 51, 51, 51, 4)     112
_________________________________________________________________
batch_normalization (BatchNo (None, 51, 51, 51, 4)     16
_________________________________________________________________
leaky_re_lu (LeakyReLU)      (None, 51, 51, 51, 4)     0
_________________________________________________________________
max_pooling3d (MaxPooling3D) (None, 25, 25, 25, 4)     0
_________________________________________________________________
conv3d_1 (Conv3D)            (None, 25, 25, 25, 8)     872
_________________________________________________________________
batch_normalization_1 (Batch (None, 25, 25, 25, 8)     32
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 25, 25, 25, 8)     0
_________________________________________________________________
max_pooling3d_1 (MaxPooling3 (None, 12, 12, 12, 8)     0
_________________________________________________________________
conv3d_2 (Conv3D)            (None, 12, 12, 12, 16)    3472
_________________________________________________________________
batch_normalization_2 (Batch (None, 12, 12, 12, 16)    64
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 12, 12, 12, 16)    0
_________________________________________________________________
max_pooling3d_2 (MaxPooling3 (None, 6, 6, 6, 16)       0
_________________________________________________________________
conv3d_3 (Conv3D)            (None, 6, 6, 6, 4)        68
_________________________________________________________________
batch_normalization_3 (Batch (None, 6, 6, 6, 4)        16
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 6, 6, 6, 4)        0
_________________________________________________________________
flatten (Flatten)            (None, 864)               0
_________________________________________________________________
dropout (Dropout)            (None, 864)               0
_________________________________________________________________
dense (Dense)                (None, 256)               221440
_________________________________________________________________
batch_normalization_4 (Batch (None, 256)               1024
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 256)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               32896
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 129
=================================================================
Total params: 260,141
Trainable params: 259,565
Non-trainable params: 576
_________________________________________________________________
```

Figure 1: An example architecture made of 3 convolutional layers with kernels of size $3 \times 3 \times 3$, one convolutional layer of kernel size $1 \times 1 \times 1$ and two fully connected layers. After each convolutional layer, one has a batch normalization layer, a non-linear activation function and a pooling layer. The layer column shows the sequence of layers in the network, the "output shape" column shows the dimensions of the layer's output in the form (batch size, x-dimension, y-dimension, z-dimension, channel-dimension). See the text for further details on the network.

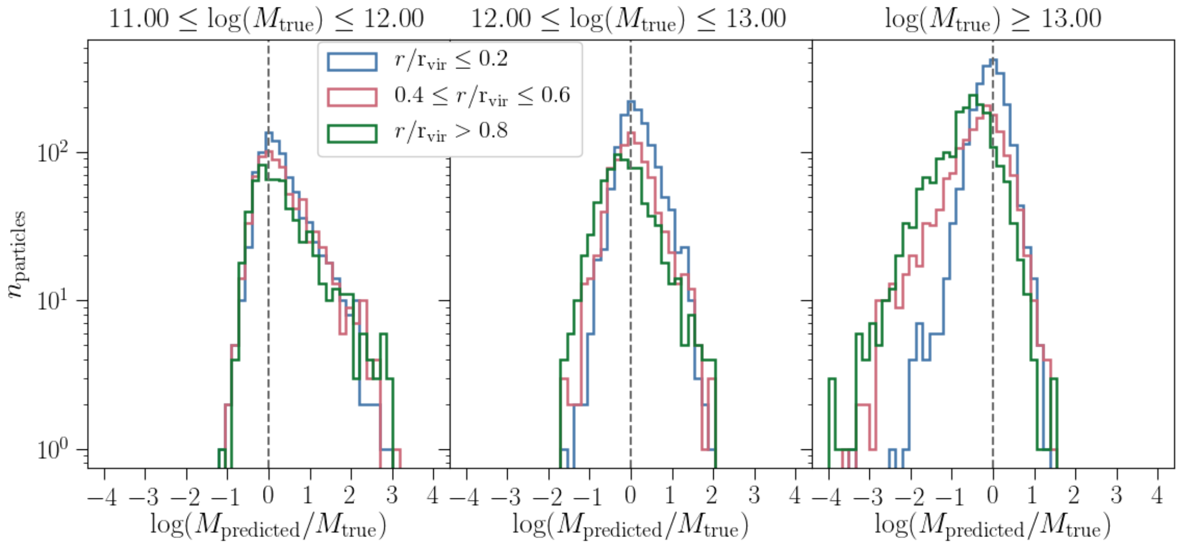


Figure 2: Difference between predicted and true halo mass in three different mass bins for three subsets of particle, divided according to the particles' distance from the centre of the halo as described in the legend. The halo mass predictions come from a CNN trained on the initial conditions density field at $z = 99$, sampled on a 51^3 grid centred at each particle's initial position. The predictions are shown for an independent simulation from the 5 simulations used for training.

a one-dimensional vector of 864 neurons. The dimensions of the FCC layers are in the form (batch size, number of neurons). Similar to the number of kernels for convolutional layers, the number of neurons in FCC layers are hyperparameters. Here I chose 256 and 238 for the two fully connected layers. Each FCC layer adopts a dropout of 0.4 of the neurons.

Fig. 2 shows the difference between predicted and true halo mass in three different mass bins for three subsets of particle, divided according to the particles' distance from the centre of the halo as described in the legend. For high-mass halos, the predictions become worse for particles that are further away from the centre of the halo. The large skew in the green histogram of the right panel indicates that the CNN tends to underestimate the halo mass of outskirts particles. On the other hand, particles that are close to the halos' centre-of-mass have a more symmetric distribution of $\log(M_{\rm predicted}/M_{\rm true})$. For mid-mass halos, the distributions are approximately symmetric around 0 for all radial categories of particles. We also find no radial dependence in the distributions for the case of low-mass halos. In this regime, the CNN tends to overestimate halo masses for all particles, independently of their final radial positions inside the halos. The aim is to understand the origin of the skews in the $\log(M_{\rm predicted}/M_{\rm true})$ distributions of high-mass and low-mass halos and see whether we can achieve better predictions in these regimes.

1.2 Underestimating halo masses for high-mass halos

The CNN underestimates halo masses for particles in the outskirts of high-mass halos. This behaviour is similar to that found in the $z = 0$ test (Section 2.1), as shown in the left panel of Fig. 8. In the $z = 0$ case, we found that the reason for this was that the input sub-boxes was too small to capture the full extent of the halo. Since the CNN was not able to extrapolate beyond the boundary of the input box, it would not have enough information to yield a reliable mass estimate for particles in the outskirts of halos. By adopting a larger input sub-box, we were able to correct for this effect and achieve a similar mean and variance in the $\log(M_{\text{predicted}}/M_{\text{true}})$ distributions for all radial categories of particles. A similar issue could be happening in the initial conditions. We will therefore test whether by adopting a larger input sub-box, we will able to improve the predictions of outskirts particles and yield similar distributions in the right panel of Fig. 2 for all three radial categories.

1.3 Overestimating halo masses for low-mass halos

At the low-mass end, the CNN overestimates halo masses. This effect is independent of the final position of particles inside the halo, given that the three distributions in the right panel of Fig.8 show similar skews. One hypothesis for this is that in the first convolutional layer we are taking a convolution between the input and a 3^3 kernel. This may lead to a loss of information below the 3-pixel scale which then affects the predictions of the CNN for low-mass halos. In other words, the resolution is insufficient to resolve the protohalos of low-mass halos when taking convolutions of the input with a 3^3 kernel.

We roughly estimated the lowest mass scale which can be resolved when convolving the input with a 3^3 kernel. The simulations have a particle mass resolution of $M_p \sim 8 \times 10^8 M_\odot$. With this resolution, the lowest mass halo that is fully resolved in the simulation is about $M_h \sim 10^{11} M_\odot$. Since in a $3 \times 3 \times 3$ kernel there are ~ 27 particles in the initial conditions, the estimated loss of resolution after the convolution is of about one order of magnitude. We therefore expect the algorithm to make reliable predictions down to $M_h \sim 10^{12} M_\odot$, which is indeed what we find from Fig. 2. To test this hypothesis, we train and test the algorithm only on particles in halos of mass $M \geq 10^{12} M_\odot$. If our hypothesis is correct, we would expect to find identical predictions on scales $M \geq 10^{12} M_\odot$ to the case of training on all particles. The predictions are shown in Fig. 3. The orange points are the case of training/testing only on particles in halos of mass $M \geq 10^{12} M_\odot$, whereas the blue points are the case of training/testing on particles from the full mass range of halos. The bias of the blue points at scales $11 \leq \log M \leq 12$ is present at scales $12 \leq \log M \leq 13$ for the orange points. The bias is even more evident in Fig. 4 where one can see the skew in the distributions in the middle panel, which were not present in the middle panel of Fig. 2. The improvement in the right panel when comparing Fig. 4 to Fig. 2 is only apparent: the error can only be up to two orders of magnitude since the range of halo mass in the training/test set has changed to cover only the range $12 \leq \log M \leq 14$. The skew in the outskirts particles compared to inner particles has not changed. This test demonstrates that the origin of the skew in the predictions of low-mass halos cannot be predominantly due to the loss of information induced by the $3 \times 3 \times 3$ kernel in the first convolutional layer.

Hiranya and I discussed with Justin Alsing the point about the potential loss of information when taking convolutions with a kernel. A number of interesting points came out of the discussion:

1. Justin agreed that there should be some loss of information if the kernel size is larger than the pixel size. Other machine learning experts (John Wu and Tom Charnock) did not agree on this – they said that the answer depends on (a) how much structure the input contains and (b) the length of correlations in the input data. The learned kernels do not need to be smooth so they can actually enhance small

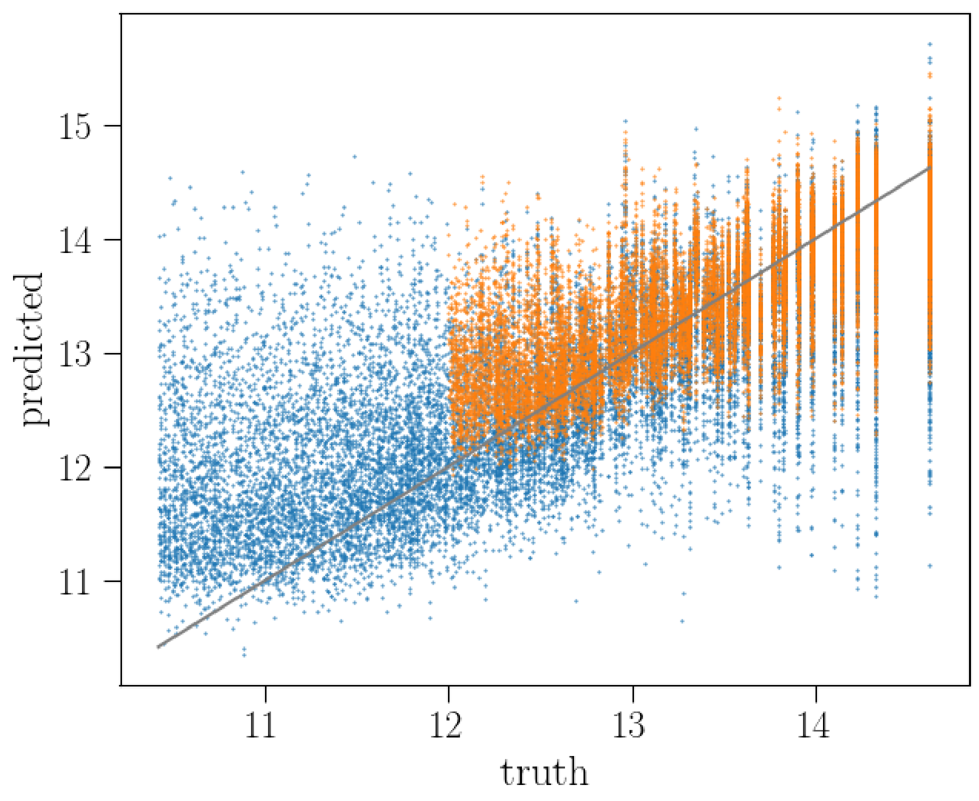


Figure 3: Predicted vs truth of the validation set for the case of training/testing only on particles in halos of mass $M \geq 10^{12} M_{\odot}$ (orange points) and the case of training/testing on particles from the full mass range of halos (blue points). The bias of the blue points at scales $11 \leq \log M \leq 12$ is present at scales $12 \leq \log M \leq 13$ for the orange points.

features (below the $3 \times 3 \times$ scale). If the small feature is dominant for the thing one wants the network to do then the training will provide a kernel which does this enhancing. So on any scales below $3{\times}3{\times}3$ one could potentially get zero information loss. In my opinion, there will be *some* information loss and we should find a way to quantify this. One thing that Hiranya suggested is to look at the change in predictions when adopting a kernel of size $5 \times 5 \times 5$.

2. We should visualize the kernels in the first layer. Are the weights all similar values or are they peaked? Fig. 5 shows the kernels of the first layer. The four rows are the different four kernels and the three columns are the three z-slices of each 3D kernel. The values do not seem smooth or statistically isotropic but rather quite random.

3. Justin suggested to try to initialize the weights in such a way that the distribution is peaked on the central voxel of the kernel. By weighting the central voxel of the kernel more than boundary voxels, we could pick up more information at the pixel-level than uniformly weighting all voxels in the kernel.

4. Justin also asked about how feasible it would be to remove the convolutions and train only with dense layers. This is basically equivalent to the case where you have a convolutional kernel that has a single

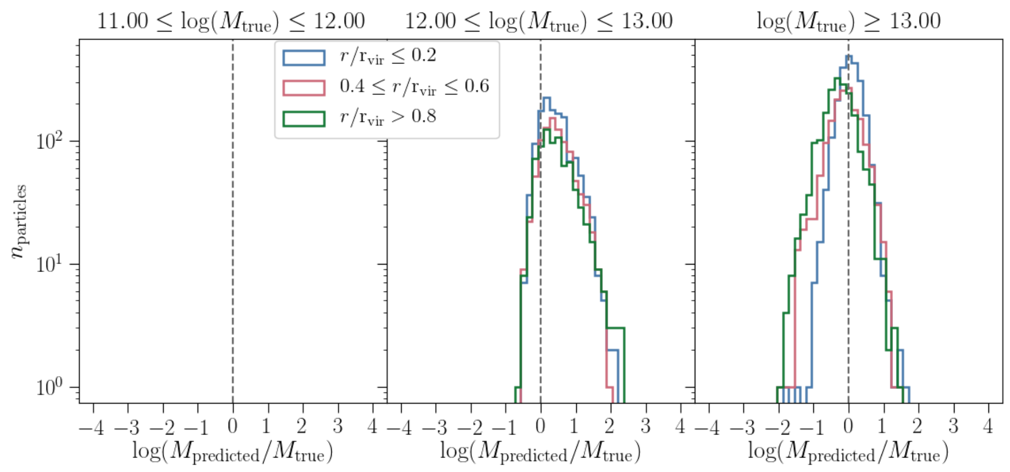


Figure 4: Same as Fig. 2 but for the case of training/testing only on particles in halos of mass $M \geq 10^{12} M_{\odot}$. The middle panel now shows a skew in the distributions which was not present in the middle panel of Fig. 2.

non-zero value at the central voxel of the kernel and all other voxels set to zero.

5. Fig. 6 shows the loss function as a function of number of epochs for the training set and the validation set. The training set is made of particles from 5 simulations, whereas the validation set is made of a random subset of particles from an independent simulation to those used for training. The loss function as a function of number of epochs is decreasing for the training set but is constant for the validation set after 10/15 epochs. Justin wouldnt necessarily call this overfitting because in the case of overfitting one would generally see the validation loss increase as the training loss decreases. He found it surprising that instead we see that the validation loss remains constant. He suggested to take all six simulations and randomly choose one for validation at each epoch. In this way, all simulations are used for training and every time we cross-validate on the simulation not used for training in that epoch. He thinks this a more robust test of overfitting than training on 5 sims and testing on the same independent simulation at each epoch.

6. Justing also suggested re-weighting the training samples by the halo mass function. He said that if you think of it like a (Bayesian) inference task, one would want to include the prior on the halo mass in there to get sensible answers out. Alternatively, if you think about wanting to get an estimator that has the same fidelity across all halo masses, youd also want to reweight your samples so its not trading off doing very well on the most typical samples at the expense of rarer ones. Hiranya and I thought that by randomly selecting the training particles from the simulation, the training set should naturally reflect the HMF. Moreover, by explicitly including a prior given by the HMF we may be feeding the algorithm with the truth.

1.4 List of next steps/ things to think about

1. Figure out the most time efficient and memory efficient way to compute and feed the data to the CNN.

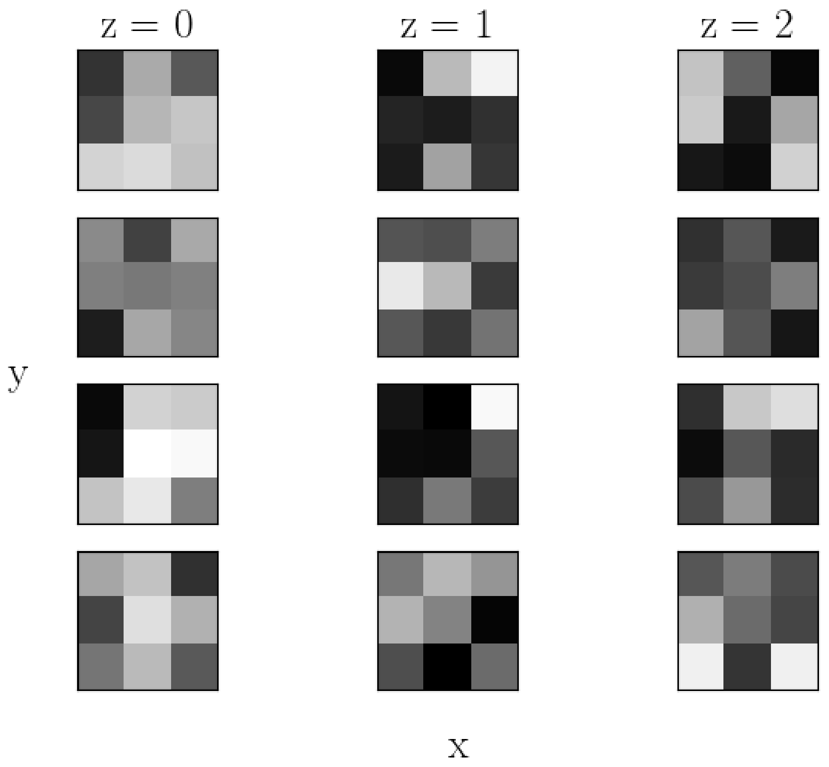


Figure 5: Kernels of the first layer. The four rows are the different four kernels and the three columns are the three z-slices of each 3D kernel. The weights are scaled such that white voxels have higher values than black voxels.

2. Train the algorithm using 75^3 boxes to test our hypothesis for the skew in the high-mass regime.
3. Revisit some of the hyperparameters like the number of kernels in the convolutional layers, the choice of activation function and the choice of weights initializer.
4. Try changing the kernel size in the first convolutional layer to $5 \times 5 \times 5$.
5. Revisit the training set size.
6. Randomly set aside one out of six simulations for validation at each epoch.

2 A test of the CNN architecture

We perform a test to verify whether or not the CNN architecture we adopt is suited for our problem of learning final halo mass starting from a 3D density field. The test is specifically designed to test whether there are certain choices in the CNN architecture that are not appropriate for predicting halo mass given as input a density field sampled on a grid. To do this, we test the performance of the model in simple scenarios where we can compare the predictions of the CNN against our expectations.

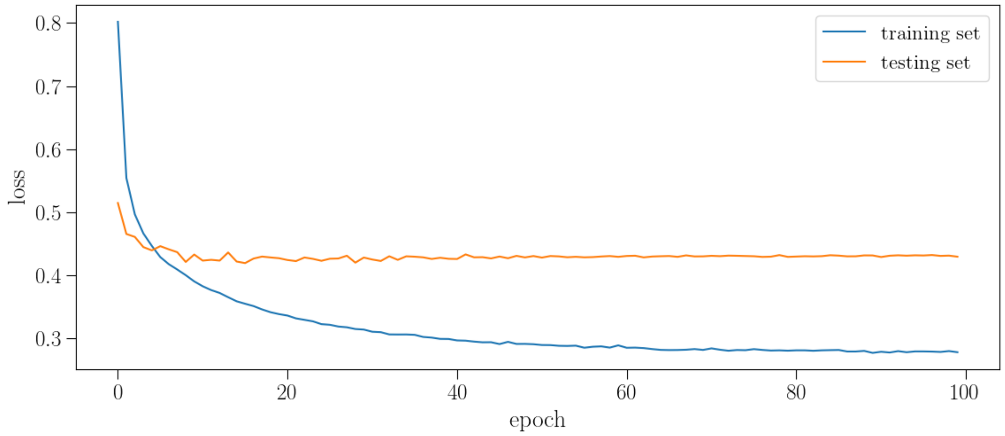


Figure 6: The loss function of the training set, made of particles from 5 simulations, and the validation set, made of a random subset of particles from an independent simulation to those used for training.

2.1 Predicting final halo mass given the non-linear density field at $z = 0$

We trained the CNN to learn the mapping between the non-linear density field at $z = 0$ and the mass of the resulting haloes. This mapping is effectively given by an algorithm which first identifies the boundary of a halo based on a fixed density threshold, similar to a friends-of-friends algorithm, and then computes the mass enclosed within such halo. As this is a much simpler mapping than that between the initial conditions density field and the final halo masses, we expect the CNN to return near-perfect predictions.

The input is given by the non-linear density field at $z = 0$ in a 3D box centred at each particle's position. We fixed the resolution of the 3D box to that used for the $z = 99$ case, $N = 51^3$, and chose a box size of $L = 1.5\,\mathrm{Mpc}\,h^{-1}$, which approximately corresponds to the virial radius of a halo with mass $M = 10^{14}\,\mathrm{M}_\odot$. These choices resulted in a voxel length $l_\mathrm{voxel} \sim 30\,\mathrm{kpc}\,h^{-1}$, which is approximately equivalent to half the virial radius of a $M = 10^{10}\,\mathrm{M}_\odot$ halo.

The CNN architecture is given by 3 convolutional layers and 3 fully-connected layers. Each convolutional layer performs first convolutions with a $3 \times 3 \times 3$ filter of stride 1 and no zero padding, then applies batch normalization (BN), then a leaky ReLU non-linear activation ($\alpha = 0.3$) and finally average pooling layer. The number of convolutions in each layer are 4, 8 and 16 respectively. The fully-connected layers have 256, 128, 1 neurons each, 0.2 dropout, a leaky ReLU non-linear activation ($\alpha = 0.3$) and batch normalization. The optimizer is AMSGrad and the learning rate is fixed at 0.0001. The training is done on $20,000$ randomly selected particles from 5 simulations of different initial random seed, resulting in a total of $100,000$ training particles. The testing is done on an independent simulation with different initial conditions realization.

The predictions for the independent simulation are shown in Fig. 7. The predictions are in good agreement with their respective ground truth halo masses, confirming that the CNN is able to learn the relevant information from the density field about halo mass. However, the tails of the violin plots (and scatter points along those tails) indicate a small degree of inaccuracy in the predictions. This raises a number of questions:

1. Is a convolutional neural network able to reproduce a "watershed" algorithm that allows it to infer the

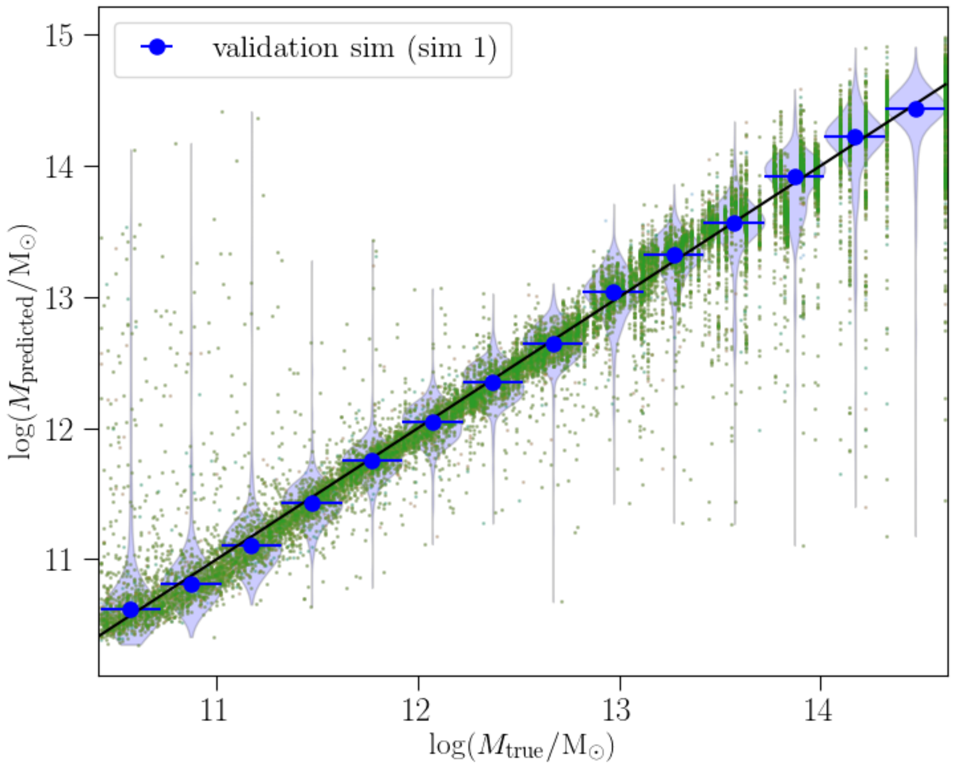


Figure 7: Halo mass predictions returned by a CNN trained on the non-linear density field at $z = 0$, sampled on a 51^3 grid. The predictions are shown in the form of violin plots i.e., distributions (and their medians) of predicted halo masses of particles within evenly-spaced bins of true logarithmic halo mass. The distributions are shown for an independent simulation from those used for training.

mass of the halo given the density field on a grid?

2. If so, do the predictions returned by the CNN match our expectations given the size/resolution of the input data? For example, does the input sub-box have (i) enough resolution to return the correct predictions for low-mass halos and (ii) a large enough volume to fully enclose high-mass halos?

We investigate further the origin of the tails of the distributions in Fig. 7 by answering the above questions as follows. Fig. 8 shows the difference between predicted and true halo mass in three different mass bins for three subsets of particle, divided according to the particles' distance from the centre of the halo. For the case of high-mass halos, we find that the predictions are worse for outskirts particles and very very good for inner and mid-radii particles. For low-mass halos, the variance in the predictions seems to come mostly from particles that are mistakenly predicted to be in larger halos, which is however uncorrelated with being in the inner or outer region of a halo. Mid-mass halos generally show a smaller variance in the predictions than high-mass and low-mass halos, where in particular outskirts particles are mildly under-predicted.

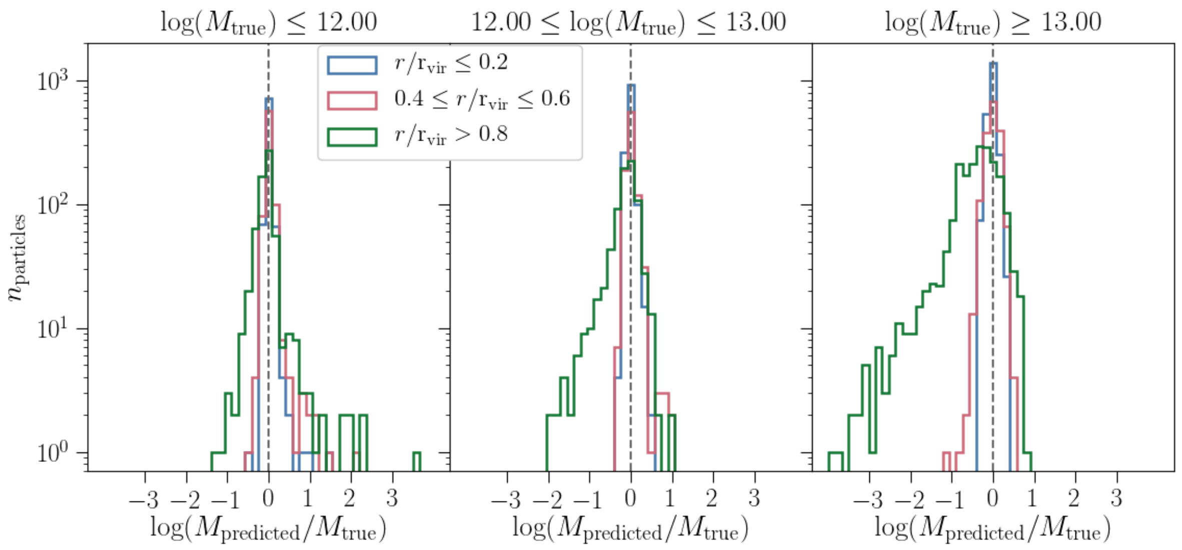
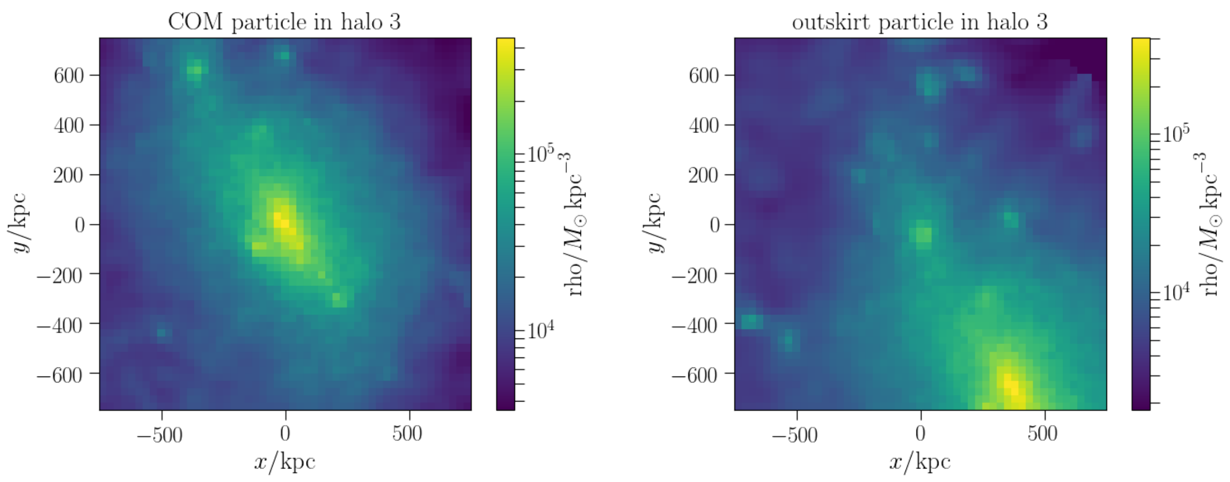


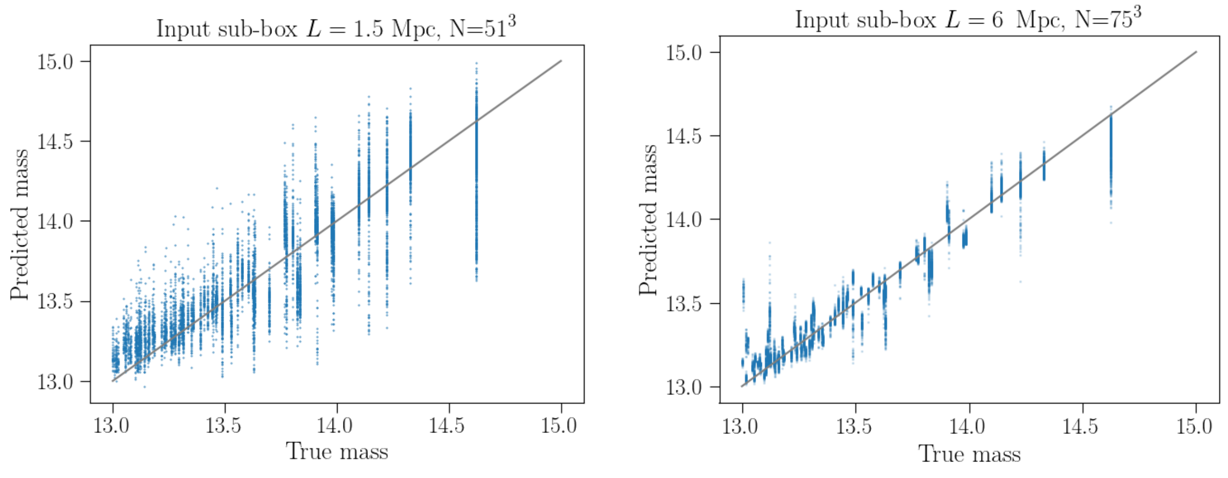
Figure 8: Difference between predicted and true halo mass in three different mass bins for three subsets of particle, divided according to the particles' distance from the centre of the halo as described in the legend.



(a) Centre-of-mass particle

(b) Outskirt particle

Figure 9: Two examples of input sub-boxes (averaged along the z-axis) for two particles in a high-mass halo, one near the centre-of mass of the halo and the other in the outskirts of the same halo.



(a) Input sub-box of size $L = 1.5$ Mpc and resolution $N = 51^3$. (b) Input sub-box of size $L = 6$ Mpc and resolution $N = 75^3$.

Figure 10: Predictions for the validation set from two models only trained and tested on particles in halos of mass $M \geq 10^{13} M_\odot$. The two models differ by the inputs used to training the algorithm: the left panel shows the case where the input is given by the $z = 0$ density field sampled in a box of size $L = 1.5$ Mpc and resolution $N = 51^3$and the right panel shows the case of a box of size $L = 6$ Mpc and resolution $N = 75^3$.

2.1.1 High-mass halos

There are two hypothesis that would explain why the CNN returns unreliable halo mass estimates for outskirts particles:

1. The first hypothesis is that the sub-box that we provide to the CNN is not large enough to capture the full halo. Fig. 9 shows the input sub-boxes of a centre-of-mass (COM) particle (left panel) and an outskirt particle (right panel) in the same cluster-sized halo; the sub-box of the COM particle encloses most of the halo meaning that the CNN is able to return a good estimate of the mass of that halo, whereas the sub-box of the outskirt particle misses most of the halo meaning that the algorithm does not have sufficient information to infer the total mass of that halo. If the deep learning algorithm is not able to extrapolate beyond the boundaries of the sub-box (as a human would do), then it would not be able to return the correct halo mass predictions in cases where the sub-box is centred at the edge of the halos. This would therefore explain the poor predictions of outskirts particles shown in Fig. 7.

2. A second hypothesis is that outskirts particles have the wrong predictions because they belong to sub-halos within the larger parent halo. Given that the sub-boxes are too small to capture the full extent of the parent halo, the algorithm may wrongly consider sub-halos to be distinct halos and assign a predicted halo mass that is correlated with the mass of the subhalo.

Is the algorithm failing to extrapolate beyond the boundaries of the input sub-boxes?

To test this hypothesis, we change the resolution/size of the input sub-box to $N = 75^3$ and $L = 6$ Mpc and then train and test the algorithm only on particles in halos of mass $M \geq 10^{13} M_\odot$. This ensures that

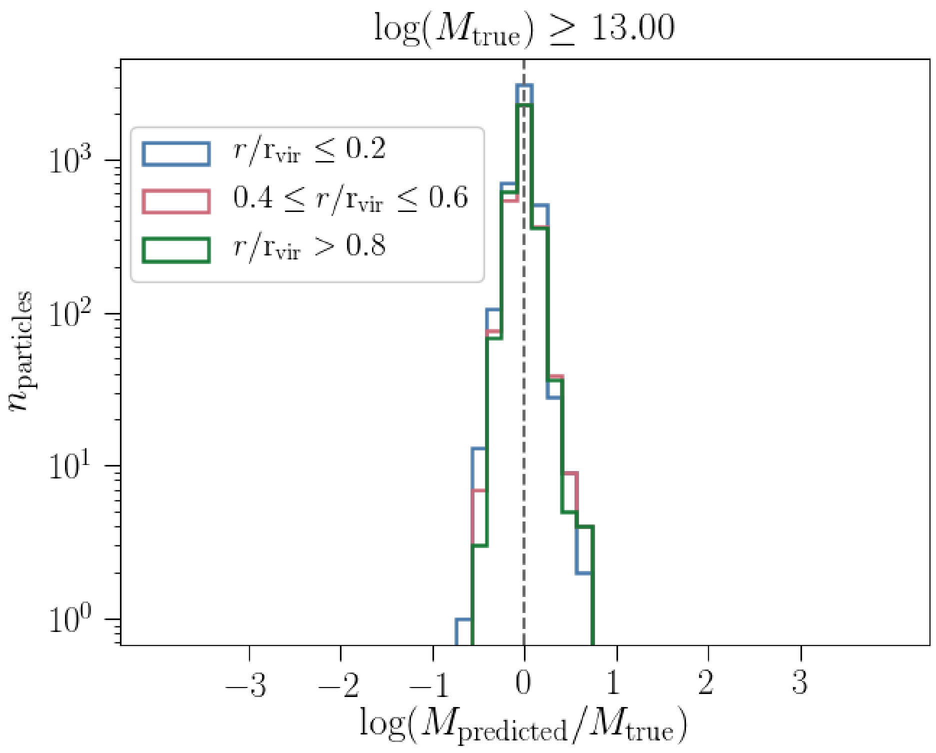


Figure 11: Difference between predicted and true halo mass for high-mass halos for the case where the input sub-box has size $L = 6\,$Mpc and resolution $N = 75^3$. The plot shows that outskirts particles have now the same variance as the other radial categories, thanks to the larger volume of the input sub-box compared to Fig.8.

the sub-box is sufficiently large to enclose the largest halos in the simulation, as well as still resolving the lowest-mass halo in the training set. The training and testing procedure is done as described in 2.1. We compare this to the case of training and testing only on particles in halos of mass $M \geq 10^{13} M_\odot$ but using as inputs the original sub-boxes of $L = 1.5\,$Mpc and $N = 51^3$.

Fig. 10 shows the predictions on the validation set when using sub-boxes of $L = 1.5\,$Mpc and $N = 51^3$ (left panel) and sub-boxes of $L = 6\,$Mpc and $N = 75^3$ (right panel); the test set in the left panel shows quite a large degree of variance in the predictions compared to that of the right panel, which instead shows very good predictions up to the largest halo mass. This indicates that by providing the algorithm with sub-boxes that are sufficiently large to enclose the full size of halos, the algorithm yields improved predictions. This also confirms that it is difficult for the algorithm to extrapolate if it is provided with parts of the full 3D structure given a limited training set. One possibility to overcome this issue would be to augment the training data such that it can learn from many examples how inputs enclosing only parts of the full halo link to the same ground truth mass. This can be done via data augmentation, where one rotates and translates the existing data to yield synthetic new training samples. The residual noise in the predictions of the right panel may be due to the small number of halos present in the simulations in the mass range $\log M \geq 13.5$. On these scales, since there are only a limited number of halos per simulation, the distribution of true halo

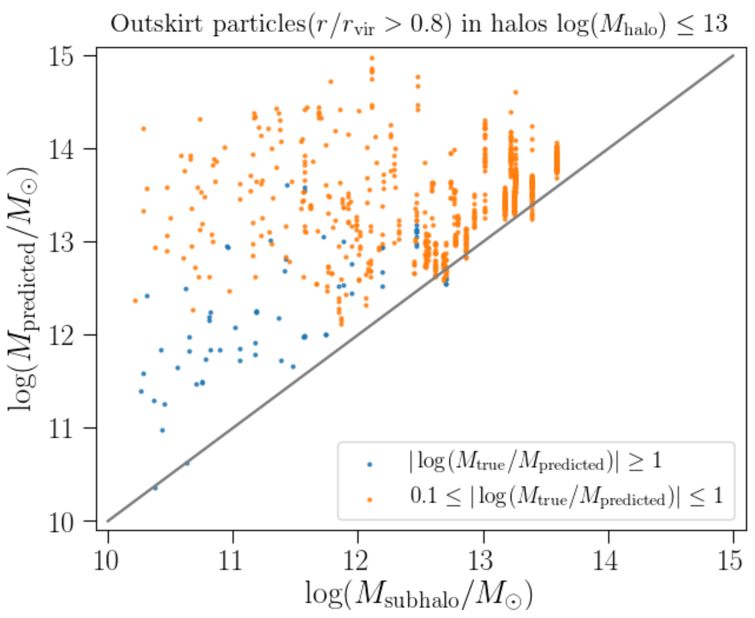


Figure 12: Subhalo mass vs the predicted halo mass for particles that are in the outskirts of high-mass halos and have (i) $|\log(M_{\rm predicted}/M_{\rm true})| \geq 1$ and (ii) $0.1 \leq |\log(M_{\rm predicted}/M_{\rm true})| \leq 1$. The plot shows that there is a weak correlation between the predicted halo mass of these particles and the mass of the subhalo to which they belong for particles that have a predicted halo mass that is at least an order of magnitude off the true mass. However, for many particles that belong to small subhalos, their predicted mass is closer to the true halo mass, i.e. in the range $13 \leq \log(M_{\rm predicted}) \leq 14$ and uncorrelated with their subhalo mass.

mass will suffer from discreteness issues. The distribution of true mass in the validation set is more discrete than that of the training set since the validation set is made of particles from a single simulation, whereas the training set combines particles from five different simulations. Nevertheless, this discreteness feature may lead to additional noise in the mapping between density field and halo mass learnt by the CNN.

In conclusion, we have found that by increasing the input sub-box to be large enough that it encloses the full spatial extend of the protohalos, the distribution of $\log(M_{\rm predicted}/M_{\rm true}$ for outskirts particles becomes the same as that for particles near the centre-of-mass of halos. This is demonstrated explicitly in Fig.11.

Do the outskirt particles in high-mass halos with wrong halo mass predictions belong to subhalos?

We then wish to test our second hypothesis of why the algorithm underpredicts the halo mass of outskirt particles in high-mass halos in the original case of a 51^3 box of size $L = 1.5\,{\rm Mpc}\,h^{-1}$. The hypothesis is that the algorithm underestimates the halo mass of outskirt particles in high-mass halos because these particles belong to sub-halos.

We first test whether or not there is a correlation between the predicted mass of outskirt particles in high-mass halos and the mass of the subhalo in which the belong, for those particles that do belong to subhalos. Fig. 12 shows the subhalo mass vs the predicted halo mass for particles that are in the outskirts of high-mass halos and have (i) $|\log(M_{\rm predicted}/M_{\rm true})| \geq 1$ and (ii) $0.1 \leq |\log(M_{\rm predicted}/M_{\rm true})| \leq 1$. The

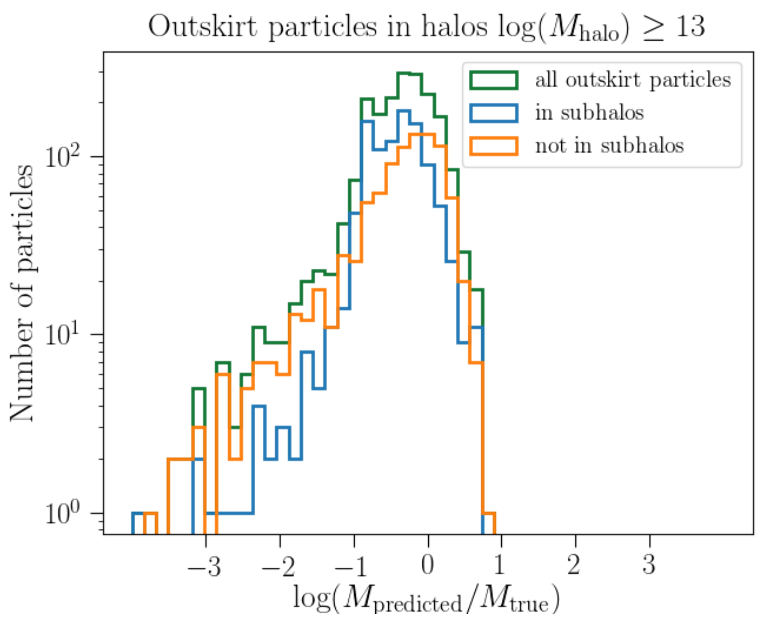


Figure 13: Difference between predicted and truth for all outskirts particles in high-mass halos (green histogram), together with the distributions for particles that belong to sub-haloes inside the larger parent halo and those that do not. The histograms are comparable, meaning that not all particles that are being incorrectly classified by the algorithm also belong to subhalos. A significant fraction of incorrectly classified particles is given by those that do not belong to halos.

former set of particles (i) are those that have a predicted halo mass that is at least an order of magnitude away from the true mass. These are essentially the particles that are in the tails of the green histogram in the right panel of Fig. 7, or in other words, those that have the worst predictions. The latter set of particles (ii) are those that form the bulk of the green histogram and that are up to 0.1 in log mass away from the true mass. We find that there is a (weak) correlation between subhalo mass and predicted halo mass, for the (i) set of particles. Unfortunately, there are too few points to have a robust statistical measure of the correlation ($\sim$ 200 particles) but the visible correlation in the plot is good enough for the purpose of this test. This means that when outskirt particles in subhalos are assigned a predicted halo mass that is very far off the true mass, that predicted mass is correlated with the mass of the subhalo is correct. However, we find that for many particles that belong to small subhalos, their predicted mass is uncorrelated with their subhalo mass. In fact, their predicted mass is closer to the true halo mass, i.e. in the range $13 \leq \log(M_{\rm predicted}) \leq 14$, and has very little correlation with the mass of the subhalo to which they belong. This demonstrates that the fact that outskirts particles live in subhalos can at most be a secondary reason why the algorithm infers a wrong halo mass estimate for those particles.

We then test what fraction of outskirt particles that are misclassified by the algorithm belongs to subhalos. Fig. 13 shows the distribution of $\log(M_{\rm predicted}/M_{\rm true})$ for three sets of particles:

1. all outskirts particles in halos of $\log(M) \geq 13$ (this is the same as the green histogram of the right panel in Fig. 7)),

2. outskirts particles which are in subhalos,
3. outskirts particles which are not in subhalos.

We find that the tail of the green histogram is not dominated by particles in subhalos, as we had previously thought. Quantitatively, the green histogram has mean=-0.42, standard deviation=0.58; the blue histogram has mean=-0.45, standard deviation=0.48; the orange histogram has mean=-0.401, standard deviation=0.67. This means that the origin of the underpredictions is not solely associated to particles in subhalos; the distributions of particles in subhalos and not in subhalos are comparable.

In conclusion, not all particles that are being incorrectly classified by the algorithm also belong to subhalos. We find that the distribution of incorrectly classified particles that do not belong to subhalos is similar to that of particles that do belong to subhalos. On the other hand, out of those particles that do belong to subhalos and have a predicted mass that is very far off the true halo mass, we find that their predicted mass does (weakly) correlate with the mass of the subhalo to which they belong. This demonstrates the ability of the deep learning algorithm to recognize a halo-like structure when presented with an overdense region and to infer its mass. However, the impact of belonging to a subhalo can only be a secondary effect. The main source of error for outskirts particles in high-mass halos comes from the fact that a sub-box of $L = 1.5$ Mpc can only capture part of the full halo size and the algorithm is not able to extrapolate beyond the sub-box and infer the correct halo mass when presented with only partial data. This is demonstrated by the fact that as we provide to the algorithm a sufficiently large box to enclose the full extent of high-mass halos, the algorithm returns an improved set of predictions (Fig.10 &11).

2.1.2 Low-mass halos

The issues in the low-mass regime may be multiple. Firstly, we tested whether the resolution of the input sub-boxes is sufficient to resolve halos in the low-mass regime and if by increasing the resolution of the input sub-box we would be able to improve the tails in the predicted distributions of low-mass halos. Fig. 14 shows the comparison between the predicted distributions given as input the $z = 0$ density field sampled on a 51^3 and a 75^3 grid. We find no significant difference in the predictions of low-mass halo when increasing the resolution of the input sub-box.

Secondly, the input sub-box of a particle in a low-mass halo will contain many different halos since the volume of the sub-box is much larger than the size of a typical low-mass halo. Examples of input sub-boxes for an outskirt and an inner particle of the same low-mass halo are shown in Fig. 15. The different halos within each sub-box may only be separated by a few pixels due to the limited resolution of the box. Moreover, if a neighbouring halo happens to be a large cluster (as in Fig. 15), the CNN may not be able to distinguish whether the particle (i) belongs to the outskirts of the cluster (and possibly in a sub-halo of that cluster) or (ii) belongs to the smaller distinct halo at the centre of the box. Fig. 15 also shows the very small difference between the inputs of an inner and an outskirt particle in low-mass halos, explaining why the predictions in the left panel of Fig. 7 do not differ significantly for particles in the different radii categories. In conclusion, the proximity of a larger-sized halo may be the primary cause of ambiguity in estimating the final halo mass for those particles. The issue of neighbouring halos separated by only a few pixels is related

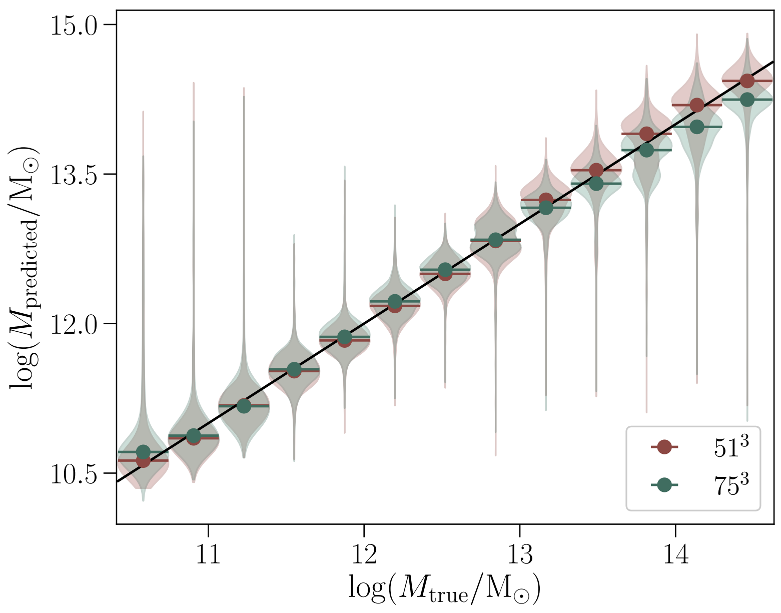


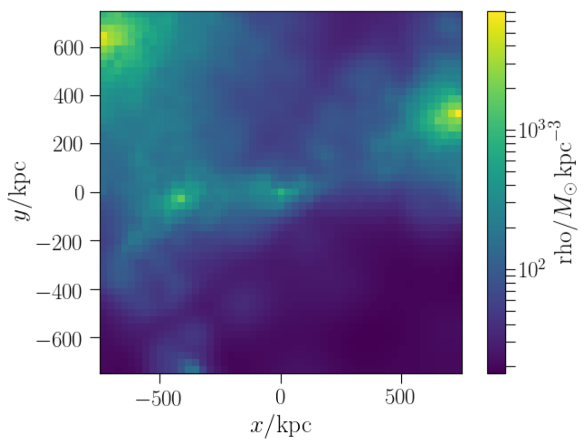
Figure 14: Comparison between the predicted distributions given the $z = 0$ density field sampled on a 51^3 and a 75^3 grid. Increasing the resolution of the input sub-box does not improve the tails of the distributions for low-mass halos.

to the reason why Bernardini et al. (2019) choose to ignore halos below quite a high mass threshold for the size/resolution of their simulation in their deep learning framework.

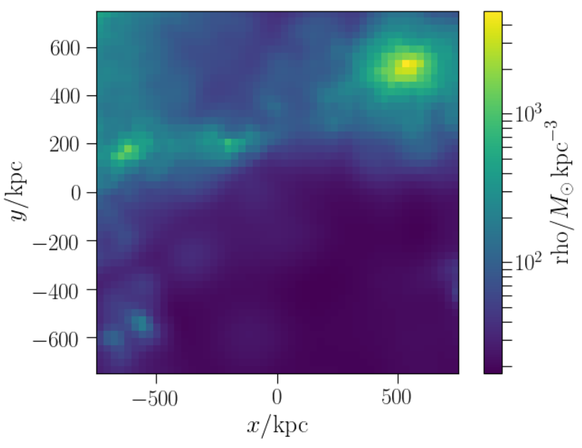
2.1.3 Conclusions

As we provide as input the $z = 0$ non-linear density field sampled in a box of size $L = 1.5$ Mpc and resolution 51^3, the CNN is able to return halo mass predictions that match our expectations, despite a small fraction of particles with errors that are larger than expected. We found that the worst predictions come from particles in the outskirts of high-mass halos. One reason why outskirts particles show the worst performance is that an input sub-box of size $L = 1.5$ Mpc is too small to capture the full extent of cluster-sized halos. Therefore, the CNN is not able to estimate the mass of the halo given a sub-box centred on the edges of that halo. When increasing the size of the input box from $L = 1.5$ Mpc to $L = 6$ Mpc, the predictions of high-mass halos are very much improved, when training and testing only on cluster-size halos. In particular, the distribution of $\log(M_{\mathrm{predicted}}/M_{\mathrm{true}}$ for outskirts particles is the same as that for particles near the centre-of-mass of halos.

Moreover, for those outskirt particles that belong to subhalos, we find that their predicted mass correlates with the mass of the subhalo only for particles whose predicted mass is at least one order of magnitude away from the true halo mass. This shows the algorithm's ability of the algorithm to estimate a mass when presented with a halo-like structure. On the other hand, we also find particles in subhalos whose predicted mass is uncorrelated with the subhalo mass. This means that the impact of subhalos on the predictive performance of the algorithm can at most be secondary to other effects. A small fraction of particles in low-mass halos also show larger errors than expected. The error in this regime most likely originates from



(a) Centre-of-mass particle



(b) Outskirt particle

Figure 15: Two examples of input sub-boxes (averaged along the z-axis) for two particles in a low-mass halo, one near the centre-of mass of the halo and the other in the outskirts of the same halo. Since the size of the halo is small compared to the size of the input box, there is no significant difference between an outskirt and an inner particle of the same halo. The primary source of error in this regime may be the proximity of other (possibly larger-sized) halos which cause ambiguity in estimating the final halo mass for those particles.

the fact that low-mass halos live in the proximity of other halos of larger mass, thus causing ambiguity for the algorithm in estimating the mass of the halo to which those particles belong.

3 Increasing the size of the training set

So far, when training the algorithm to predict final halo masses starting from the initial conditions density field, we have been using $20,000$ randomly selected particles from 5 independent simulations for training and one additional independent simulation for validation. We found that a network that is more complex than three convolutional layers and three fully-connected layers, i.e. with either an additional convolutional layer or an additional fully connected layer or one skip-connection layer, starts overfitting the training data after about 20 epochs. One possibility for this may be that the training data is too small and therefore any additional complexity to the network leads to overfitting. Consequently, by adding more independent training samples it may be possible to train a more complex network and improve the halo mass predictions.

To test this, I ran five additional DM-only simulations with different initial conditions random seeds. I first plan to use as training data $10,000$ randomly selected particles from 10 independent simulations and train a network with one additional convolutional layer and/or one skip-connection layer. If this also results into overfitting, I will try with a training set made of $20,000$ randomly selected particles from 10 independent simulations.

Figure 16 shows the power spectra at $z = 0$ for all the simulations used for training and validation. They all follow a similar trend in power, meaning that the simulations should be reliable to use. Sim-3 is the

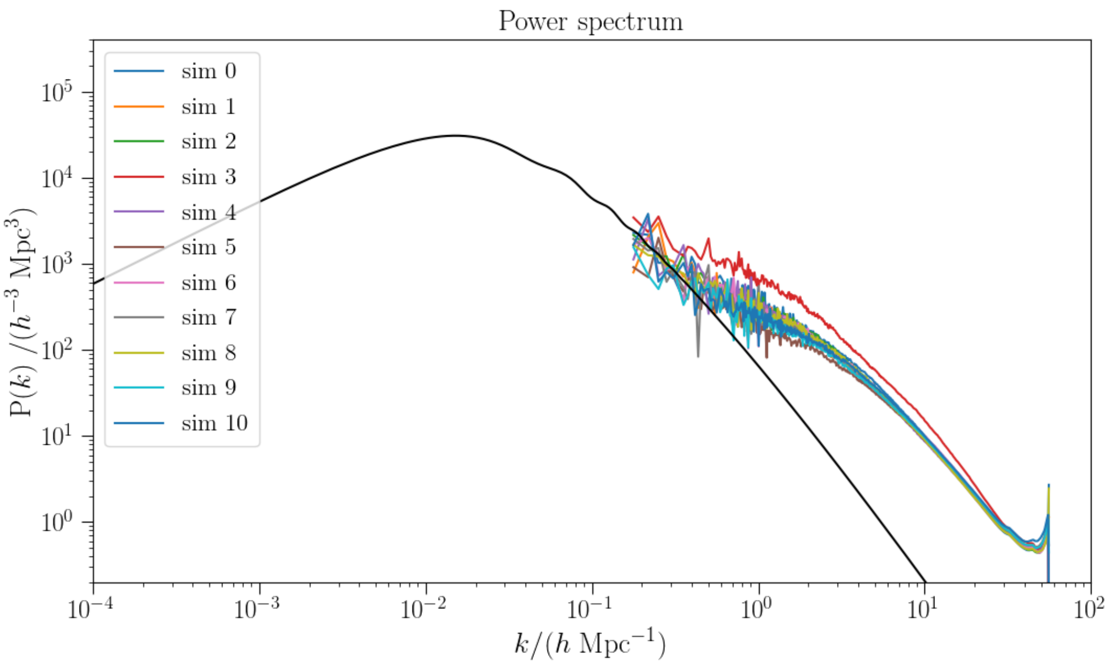


Figure 16: Power spectra at $z = 0$ for all ten simulations used for training and validating the DL algorithm. The black line shows the theoretical linear power spectrum from CAMB.

only simulations that shows a systematic offset at all k-modes but converges to similar values at the largest k-scales. Fig. 18 confirms that sim-3 has the correct power spectrum in the initial conditions, meaning that the mild offset in the power spectrum at $z = 0$ is simply the result of an "outlier" realization that picks up more power than most other simulations. Given the small number of independent realizations that we feed to the algorithm, such an outlier may affect the learning of the algorithm in an unexpected way. We may therefore want to consider ignoring simulation 3 and replacing that with a new realization that has a more typical power spectrum.

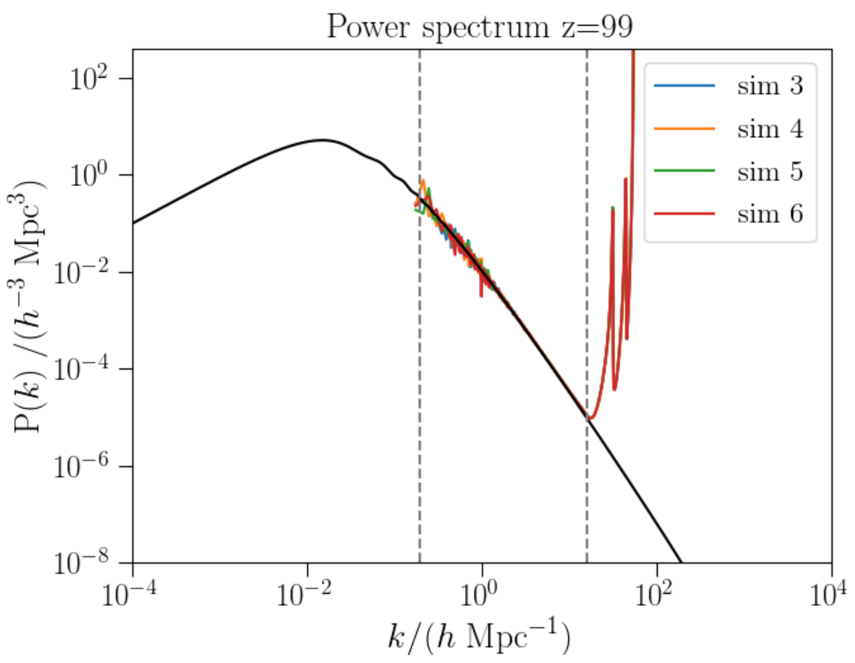


Figure 17: Power spectra at $z = 99$. The black line shows the theoretical linear power spectrum from CAMB and the dashed grey vertical lines show the smallest and largest k-scales probed by the simulation.

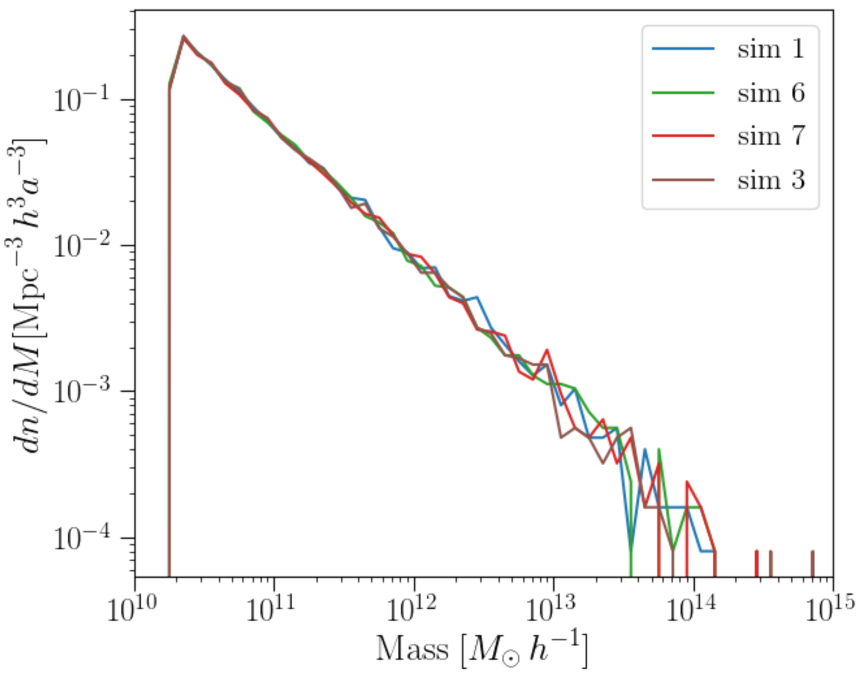


Figure 18: Halo mass function for some of the simulations. Sim-3 does not show any significant difference compared to the other simulations.