

Deep learning insights into cosmological structure formation

Luisa Lucie-Smith^{1, 2, *}, Hiranya V. Peiris^{1, 3}, Andrew Pontzen¹, Brian Nord^{4, 5, 6}, Jeyan Thiyagalingam^{7, 8}

¹*Department of Physics & Astronomy, University College London, Gower Street, London WC1E 6BT, UK.*

²*Max-Planck-Institut für Astrophysik, Karl-Schwarzschild-Str. 1, 85748 Garching, Germany.*

³*The Oskar Klein Centre for Cosmoparticle Physics, Department of Physics, Stockholm University, AlbaNova, Stockholm, SE-106 91, Sweden.*

⁴*Fermi National Accelerator Laboratory, P.O. Box 500, Batavia, IL 60510, USA.*

⁵*Department of Astronomy and Astrophysics, University of Chicago, Chicago, IL 60637, USA.*

⁶*Kavli Institute for Cosmological Physics, University of Chicago, Chicago, IL 60637, USA.*

⁷*Scientific Computing Department, Rutherford Appleton Laboratory, Science and Technology Facilities Council, Harwell Campus, Didcot, OX11 0QX.*

⁸*Department of Engineering Sciences, University of Oxford, Parks Road, Oxford OX1 3PJ.*

*luisals@MPA-Garching.MPG.DE

24 November 2020

While the evolution of linear initial conditions present in the early universe into extended halos of dark matter at late times can be computed using cosmological simulations, a theoretical understanding of this complex process remains elusive. Here, we build a deep learning framework to learn this non-linear relationship, and develop techniques to physically interpret the learnt mapping. A three-dimensional convolutional neural network (CNN) is trained to predict the mass of dark matter halos from the initial conditions. We find no change in the predictive accuracy of the model if we retrain the model removing anisotropic information from the inputs. This suggests that the features learnt by the CNN are equivalent to spherical averages over the initial conditions. Our results indicate that interpretable deep learning frameworks can provide a powerful tool for extracting insight into cosmological structure formation.

INTRODUCTION

The standard model of cosmology successfully describes the evolution of the Universe from its near-uniform early state to the present-day clustered distribution of matter [1, 2]. Within this paradigm, the formation of cosmic structures in the Universe is driven by the gravitational collapse of initially small perturbations in the density of matter. These perturbations grow over cosmic time into extended halos of dark matter, which are interconnected along walls and filaments and surrounded by empty voids, together forming the so-called cosmic web. Dark matter halos are the fundamental building blocks of cosmic large-scale structure, within which galaxy formation proceeds by accreting gas and turning it into stars [3]. The properties of the initial density fluctuations are well-constrained by measurements of the cosmic microwave background [4]; a major contemporary challenge involves developing a physical understanding of how complex, non-linear late-time cosmic structures emerge from these linear initial conditions.

Computer simulations are currently the most accurate method to compute the non-linear evolution of dark matter over cosmic time [5–8]. Given the initial conditions and a cosmological model, N -body simulations follow the evolution of “particles”, representing small-scale chunks of matter, governed by the laws of gravity. In this context, particles are not hypothesized subatomic particles thought to comprise the dark matter, but rather regions of the Universe enclosing a set amount of dark matter. N -body particles are many orders of magnitude more massive than fundamental particles; for example, for state-of the-art cosmological simulations such as the Millennium-II [9] and the Bolshoi simulations [10], a single particle has $M \sim 10^6 M_\odot$ and $M \sim 10^8 M_\odot$, respectively. Despite being able to compute the evolution of matter in the Universe, simulations alone do not provide a straightforward

answer to how dark matter halos acquire their characteristic properties – such as mass, shape, inner profile and spin – from the initial density perturbations.

On the other hand, analytic theories of structure formation can provide a qualitative understanding of the connection between the early- and late-time Universe. Analytic frameworks broadly fall into two categories: those that approximate halo formation as the gravitational collapse of spherical overdense regions [11, 12] and those that attempt to account for external tidal shear effects by modelling the collapse of ellipsoidal, rather than spherical, regions [13–17]. Due to the simplifying assumptions required to make any analytic solution possible, these models can at most provide a qualitative picture of halo collapse.

In previous work [18, 19], we proposed a novel approach based on machine learning to gain new insights into physical aspects of the early Universe responsible for halo collapse, without the need to introduce approximate halo collapse models. The approach consists of training a machine learning algorithm to learn the relationship between the early universe and late-time halo masses directly from numerical simulations. The learning of the algorithm is based on a set of inputs, known as *features*, describing pre-selected physical aspects about the linear density field in the initial conditions. Our choice of inputs was motivated by existing analytic approximations of halo collapse; we provided the algorithm with spherical overdensities (motivated by spherical collapse models) and tidal shear information (motivated by ellipsoidal collapse models) in the local environment surrounding each dark matter particle in the initial conditions. Contrary to existing interpretations of the Sheth-Tormen ellipsoidal collapse model [16, 17], we found that the addition of tidal shear information does not yield an improved model of halo collapse compared to a model based on density infor-

mation alone [18, 19]. This approach is limited by the need for feature extraction, a step required by most standard machine learning algorithms; to propose a set of informative features, we must rely on our current understanding of halo formation based on simplified and incomplete analytic approximations of halo collapse.

In this work, we extend our approach to a deep learning framework based on convolutional neural networks (CNNs) [20, 21]. The aim is to check whether there exists information in the initial conditions beyond that of spherical overdensities captured by analytic models, that yields improved predictions for the final dark matter halo masses. CNNs are capable of extracting spatially-local information directly from raw data. Unlike standard machine learning algorithms, they do not require manual extraction of pre-selected features from the data. The majority of CNN-based architectures have, at least, two main components: a feature extraction part and a predictive part. The feature extraction part consists of a series of convolutional layers, in which the algorithm learns to extract relevant features from the input data. This process is hierarchical: the first layers learn local, low-level features, which are then combined by subsequent layers into more global, higher-level features [22]. In each layer, the input is repeatedly convolved with a number of kernels, each thought to detect a specific type of feature present in the input. Convolutional layers are stacked onto each other by using the output of one layer as the input to the next; it is this hierarchical structure that enables the network to learn complex features. The second component of the CNN is the predictive part; the features assembled in the convolutional layers are combined to return the final prediction. For this, we use fully-connected layers; these are made of neurons, each returning a single output by applying a non-linear function to a weighted sum of the inputs.

The ability of CNNs to extract features makes them an attractive method to gain new insights into non-linear structure formation; they can learn directly from the initial conditions which physical aspects are most useful to predict final halo masses. On the other hand, training a machine learning algorithm with a pre-defined set of features carrying physical meaning has the advantage of being more readily interpretable. Interpretability in deep learning remains a challenge and is an active area of research in the machine learning community [23–25]. Tools based on feature visualization [26–28] or attribution techniques [29–31], such as saliency maps [30, 32], have been proposed as approaches for interpretability, but have found limited utility in extracting new knowledge about underlying datasets in physics problems. Representation learning is an area of machine learning devoted to turning high-dimensional data into more tractable representations [33–35]; however, limitations to these techniques make it difficult to turn the representations into meaningful physical parameters. Steps have been taken towards interpretability techniques that facilitate knowledge extraction about the physics of a given problem, using modified variational auto-encoders [36, 37]; these methods have so far only been applied to problems with already known representations in physics.

In cosmology, deep learning has recently become a popular method to learn mappings that require computationally-expensive N -body simulations. Examples of CNN applications to simulations include estimating cosmological parame-

ters from the three-dimensional dark matter distribution [38–41] or from simulated galaxy maps [42, 43], and constructing mock dark matter halo catalogues [44, 45]. CNNs have also been employed to learn from simulations of non-linear mappings, such as that between the Zel'dovich-displaced and non-linear density fields [46], that between the non-linear density field and the halo distribution [47–49], or that between the dark matter and galaxy distributions [43, 50]. Our work differs from such applications primarily in the aim: it is not to develop fast N -body surrogates, but rather to make use of deep learning to gain physical insight into the formation of cosmic structures within the simulations.

In this work, we combine the ability of deep learning algorithms to model non-linear relationships with techniques that enable us to physically interpret the resulting mapping between inputs and outputs. We train a CNN to model the relationship between the initial conditions and the final dark matter halos in cosmological simulations. We compare the performance of the algorithm when given the raw density field in the initial conditions of the Universe as inputs, with the case where certain physical aspects of the initial conditions are intentionally removed from the inputs. In this way, we are able to learn about which aspects of the early Universe carry relevant information about the final dark matter halos' mass.

RESULTS

We begin with an overview of our deep learning framework, which takes sub-regions of the initial conditions of an N -body simulation and predicts the final mass of dark matter halos. We present the halo mass predictions returned by the model, comparing to expectations from previous work. We then move on to interpreting the learnt mapping between initial conditions and halo mass, by testing the impact on the model's performance as we remove from the inputs certain physical aspects of the initial conditions. Finally, we test the robustness of our model in extracting features across the wide range of scales probed by the initial sub-region boxes, and conclude with a discussion on the implications of our work.

Overview of the deep learning framework

N -body simulations are the most general and effective available approach to follow the clustering evolution of matter at all scales (Fig. 1). Simulations start from early times, when the Universe was filled with small matter density perturbations that are described by a Gaussian random field. The simulations then follow the evolution of these fluctuations as they enter the non-linear regime, through the emergence of self-gravitating dark matter halos wherein galaxies form. The final product of the simulation resembles the Universe we observe today, characterized by dark matter halos embedded in a web of filamentary large-scale structure.

Our aim is to develop an interpretable deep learning framework that can be used to learn about the physical connection between the early Universe and the mass of the final dark matter halos (Fig. 1). We focus on the mass of dark matter halos as it is the halos' primary characteristic, but our framework can similarly be applied to other halo properties. The CNN is trained to predict the final mass of the halo to which any given dark matter particle in the simulation belongs at the present time. Twenty cosmological simulations of $N = 256^3$ dark

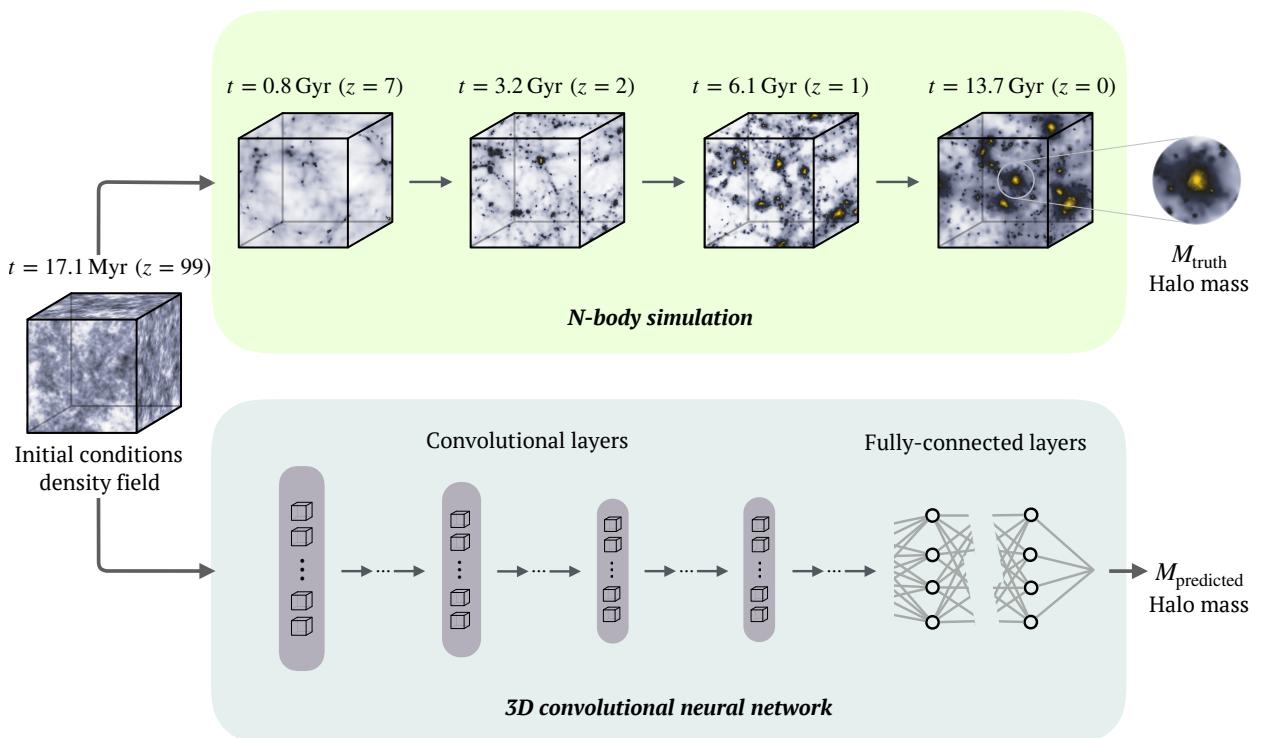


Figure 1 | *N*-body simulations of cosmological structure formation can accurately compute the gravitational evolution of dark matter over cosmic time, but do not provide a straightforward physical understanding of how cosmic structures arise from the initial conditions. We train a CNN model to learn the relationship between the initial density field and the final dark matter halos, given examples from *N*-body simulations. The inputs to the CNN are given by the initial density field surrounding each dark matter particle and the outputs are the mass of the dark matter halos to which each particle belongs at $z = 0$. The aim is to interpret the mapping learnt by the CNN in order to gain physical insights into dark matter halo formation.

matter simulation particles in a $(50 \text{ Mpc } h^{-1})^3$ volume were used to generate the training data for the CNN. The input to the CNN for a given particle is given by the initial density field in a cubic sub-region of the initial conditions of the simulation, centred on the particle's initial position. The sub-region covers a $(15 \text{ Mpc } h^{-1})^3$ sub-volume of resolution $N = 75^3$.

The deep learning model consists of a 3D CNN, made of six convolutional layers and three fully-connected layers. Although CNNs are generally applied to two-dimensional images, we use three-dimensional kernels in the convolutional layers that can be applied to the 3D initial density field of the *N*-body simulation. The convolutions were performed with 32, 32, 64, 128, 128, 128 kernels for the six convolutional layers, respectively. The kernels have size $3 \times 3 \times 3$. All convolutional layers (but the first one) are followed by max-pooling layers; their output is then used as input to the non-linear leaky rectified linear unit (LeakyReLU) [51] activation function. We refer the reader to the Methods for more details on the CNN architecture. By training the network across many examples of particles across many simulations, the model learns to identify the aspects of the initial density field which impact the final mass of the resulting halos.

Training the deep learning model requires solving an optimization problem. The parameters of the model, \mathbf{w} , are op-

timized to minimize the loss function, $\mathcal{L}_w(\mathbf{M}_{\text{true}}, \mathbf{M}_{\text{predicted}})$, which measures how closely the predictions, $\mathbf{M}_{\text{predicted}}$, are to their respective ground truths, \mathbf{M}_{true} , for the training data. The model consists of a large number of parameters, thus making it extremely flexible. As a result, CNNs are often prone to overfitting the training data, without generalizing well to unobserved test data. To overcome this, regularization techniques are employed by incorporating an additional penalty term into the loss function. We briefly describe how we design a custom loss function that is tailored to our specific problem, referring the reader to the Methods section for more details. The loss function is given by

$$\mathcal{L}_w(\mathbf{M}_{\text{true}}, \mathbf{M}_{\text{predicted}}) = \mathcal{L}_{\text{pred}}(\mathbf{M}_{\text{true}}, \mathbf{M}_{\text{predicted}}) + \mathcal{L}_{\text{reg}}(\mathbf{w}), \quad (1)$$

where $\mathcal{L}_{\text{pred}}(\mathbf{M}_{\text{true}}, \mathbf{M}_{\text{predicted}})$ is the predictive term, measuring how well the predicted values match the true target values, and $\mathcal{L}_{\text{reg}}(\mathbf{w})$ is the regularization term. The predictive term can be re-expressed as $\mathcal{L}_{\text{pred}} = -\ln [p(\mathbf{M}_{\text{true}} | \mathbf{w}, \mathcal{M})]$, where $p(\mathbf{M}_{\text{true}} | \mathbf{w}, \mathcal{M})$ describes the probability distribution of ground truth values \mathbf{M}_{true} , given the predicted values $\mathbf{M}_{\text{predicted}}$ of the training data returned by the 3D CNN model \mathcal{M} with parameters \mathbf{w} . A common choice in the community is that of a Gaussian or Laplacian distribution, yield-

ing the popular mean-squared-error or mean-absolute-error losses for regression. We found that a Cauchy distribution provides a better description of the data, as it contains broader tails than those of a Gaussian distribution. A Cauchy distribution is characterized by the scale parameter γ which specifies the half-width at half-maximum. Since we do not know a priori the appropriate value for γ , we optimized this parameter via back-propagation [52] during the training procedure of the CNN, similar to the way the model parameters w are optimized. The Cauchy loss is further modified to account for selection effects in the training data.

The regularization term in Eq. (1), $\mathcal{L}_{\text{reg}}(w)$, is intended to simultaneously (i) improve the optimization during training by preventing the algorithm from overfitting the training data and (ii) compress the neural network model into the smallest number of parameters without loss in performance. To prevent overfitting, we imposed L2 regularization in the convolutional layers, penalizing the sum over the squared values of the layers' parameters, and L1 regularization in the fully-connected layers, penalizing the sum over the absolute values of the layers' parameters. Both choices of regularizers promote small values for the weights; the choice of L1 regularization has the additional benefit that it induces sparsity on the weights by driving most weights to be zero. To compress the model, we used an L1 regularizer over groups of parameters in the fully-connected layers, such that all the parameters in a group are either simultaneously set to 0, or none of them are. This is known as a group Lasso regularizer [53]. By defining a group to be all parameters related to a single neuron, we were able to induce sparsity over the neurons of the fully-connected layers, thus yielding model compression.

The parameters are optimized during training via back-propagation, which consists of the chain rule for partial differentiation applied to the gradient of the loss with respect to the parameters. The training proceeds for thousands of iterations, each consisting of a forward and a backward pass. In the forward pass, the input runs through the network and reaches the output layer, and in the backward pass, the parameters of the network are updated to minimize the loss function evaluated for the training data. We split the training samples into sub-sets, called batches, which are forward- and backward-propagated through the network independently. The performance of the model is monitored at the end of each iteration by evaluating the loss function on an independent set of particles kept aside for validation. The training proceeds until the validation loss reaches its minimum.

Predicting the mass of dark matter halos from the initial conditions

We applied our trained deep learning framework to particles from independent simulations which were not used for training. The CNN predicts the mass of the halo to which the particles will belong at $z = 0$. The test set contains particles belonging to randomly-selected dark matter halos with mass $\log(M/M_{\odot}) \in [11, 13.4]$ (Fig. 2A). This mass range is set by the resolution and volume of our simulations; halos of mass $\log(M/M_{\odot}) \lesssim 11$ are not well resolved and those with mass $\log(M/M_{\odot}) \gtrsim 13.4$ are rare (and therefore under-represented) in the small volume of our simulations. The halos in the test set do not only differ in mass; they also differ by

factors such as their formation history and their large-scale environment, which contribute significantly to making the mapping between initial conditions and halo masses challenging. For example, halos of the same mass may have assembled smoothly through small accretion events or violently through mergers with other massive structures; they may have formed in isolated regions of the Universe or close to filaments and other massive objects. Particles that belong to the same halo may even have experienced significantly different dynamical histories: those in the inner region of halos are more likely to have been bound to the proto-halo patch from very early times, whereas those in the outskirts may have been more recently accreted onto the halo through late-time halo mergers, tidal stripping or accretion events. All this variability in the formation process of dark matter halos is not explicitly presented to the deep learning model; the CNN is faced with the task of finding features in the initial conditions which contain information about the complex, non-linear evolution of halos.

We compare the predictions made by the CNN to the true halo masses of the test set particles (Fig. 2). The predictions are shown as violin plots i.e., distributions obtained from the predicted halo masses of particles within bins defined by their true logarithmic halo mass. The black solid line shows $y = x$ and represents the idealized case of 100% accuracy. The predictions' distributions are characterized by large variances and skewness throughout the whole mass range of halos, although the maxima of the posterior distributions are in the correct location. The variance in the distributions is larger for low-mass halos compared to high-mass halos.

We next compare the accuracy of the predictions across particles that reside in different locations inside the halos. We split the halos into three separate mass ranges and compare the predictions for particles in the innermost, mid-region, and outskirts of halos for all three mass bins. We find that the particles with the most accurate predictions are those in the innermost regions of the highest-mass halos. Particles that live in the outskirts of high-mass halos have a larger variance in their predictions: this is partially due to the fact that sometimes the input sub-box volume does not include the full extent of the region that will later collapse into a high-mass halo. Therefore, when the input sub-box is centred on an outskirt particle, its volume does not cover a large fraction of the proto-halo region, making it difficult for the CNN to infer the correct final halo mass. This explains why outskirts particles tend to yield larger errors than inner particles. For halos of smaller mass, the distribution of predictions for particles in different locations inside the halos share similar variances. One way to improve the predictions of outskirt particles in high-mass halos would be to use larger sub-box volumes as inputs. However, adopting a larger sub-box at the same resolution was not possible due to computational limitations in memory consumption. Similarly, decreasing the resolution to accomodate for a larger volume would cause the predictions to worsen for particles in low-mass halos.

We now compare our results with those from our previous work [19], where a gradient boosted tree (GBT) was trained on the same regression task. The fundamental difference between the CNN and the GBT lies in the inputs provided to the algorithms: the former learns from the initial density field, containing all the information required to describe the

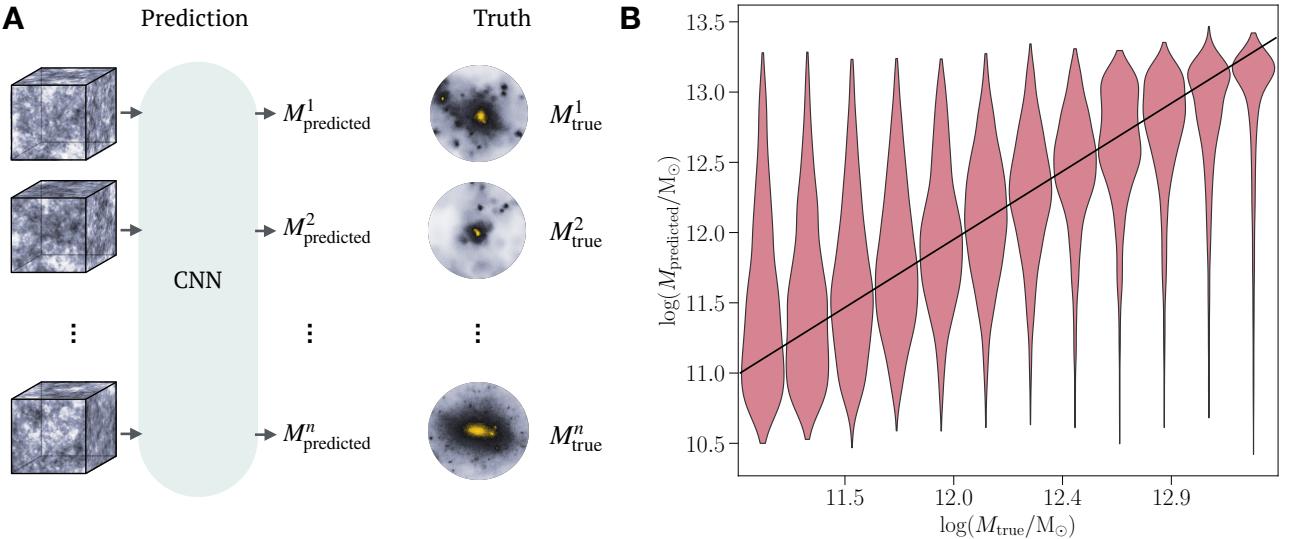


Figure 2 | (A) The CNN makes predictions for simulation particles that occupy different regions of the initial conditions of the simulation. These particles end up in halos which differ not only in their mass, but also in their formation history, large-scale environment, and amount of sub-structure within the halos. The CNN must identify from the initial density field the features that impact the final mass of the resulting halos. **(B)** Halo mass predictions returned by a CNN trained on the initial density field surrounding each dark matter particle's initial position. The predictions are shown as violin plots i.e., distributions of predicted halo masses of particles within evenly-spaced bins of true logarithmic halo mass.

initial conditions, whilst the latter learns from a more limited set of information describing only some specific physical aspect about the density field. More specifically, the features used to train the GBT were the averaged density within spheres of different radii centred at each dark matter particle's initial position. We find that the CNN returns qualitatively similar predictions to those from the GBT in the mass range $11.4 \leq \log(M/M_\odot) \leq 13.4$.

To quantify the similarity between the predictions of the two machine learning models, we used the Kullback-Leibler (K-L) divergence [54], a measure of the distance between probability distribution functions. Given the residual, $\Delta = \log(M_{predicted}/M_{true})$, we computed the K-L divergence between the distribution of residuals returned by the CNN model, $n_{CNN}(\Delta)$, and that returned by the GBT model, $n_{GBT}(\Delta)$,

$$D_{KL}(n_{CNN} \parallel n_{GBT}) = \int_{\Delta_{min}}^{\Delta_{max}} n_{CNN}(\Delta) \ln \left[\frac{n_{CNN}(\Delta)}{n_{GBT}(\Delta)} \right] d\Delta, \quad (2)$$

where Δ_{min} and Δ_{max} are given by the minimum and maximum values of Δ where $n_{CNN}(\Delta) \neq 0$ and $n_{GBT}(\Delta) \neq 0$. It is a non-negative quantity and takes the value $D_{KL}(n_{CNN} \parallel n_{GBT}) = 0$ if and only if the two distributions are fully identical, $n_{CNN}(\Delta) = n_{GBT}(\Delta)$. On the other hand, comparing the CNN to a hypothetical model \mathcal{H} that returns random halo mass predictions yields $D_{KL}(n_{CNN} \parallel n_{\mathcal{H}}) = 0.40$. We find $D_{KL}(n_{CNN} \parallel n_{GBT}) = 0.03$, indicating good agreement between the distribution of residuals of the CNN and the GBT models.

Despite the fact that the CNN learns directly from the initial density field, containing all the information needed to describe the initial conditions of the Universe, its final halo mass predictions are consistent with those of a model based on infor-

mation about spherical overdensities alone. This result suggests that the CNN may be extracting similar features to those provided to the GBT, and that these saturate the most relevant information in the initial conditions about final halo masses. To verify this hypothesis, we require tools that allow us to interpret the features learnt by the deep learning model with respect to physical aspects of the initial conditions.

Interpreting the information learnt by the CNN

The comparison between the GBT and the CNN revealed that the CNN may be learning features that carry similar information to spherical overdensities, as the two models return consistent predictions. We therefore wish to interpret the features learnt by the deep learning model in relation to physical aspects of the initial conditions. This interpretation would then allow us to provide insight into whether or not there exists information in the initial conditions beyond spherical overdensities that is useful for describing halo collapse.

To do this, we intentionally modified the inputs to the CNN to remove any anisotropic information about the 3D density field and re-trained the CNN. Given 20 concentric shells around the centre of the box, the inputs were constructed by assigning to each voxel within a given shell the average density within that shell (Fig. 3A). In this way, each voxel of the 3D input sub-box only carries information about the spherically-averaged density, similar to the features used for training the GBT. We call this model the *averaged-density* training set model, whereas the original one which uses the full initial density field as input is denoted as the *raw-density* training set model. The two separately-trained models were used to return their own halo mass predictions for the same set of test particles.

Figure 3B shows the comparison between the predictions of the two models, one trained on the raw initial density field

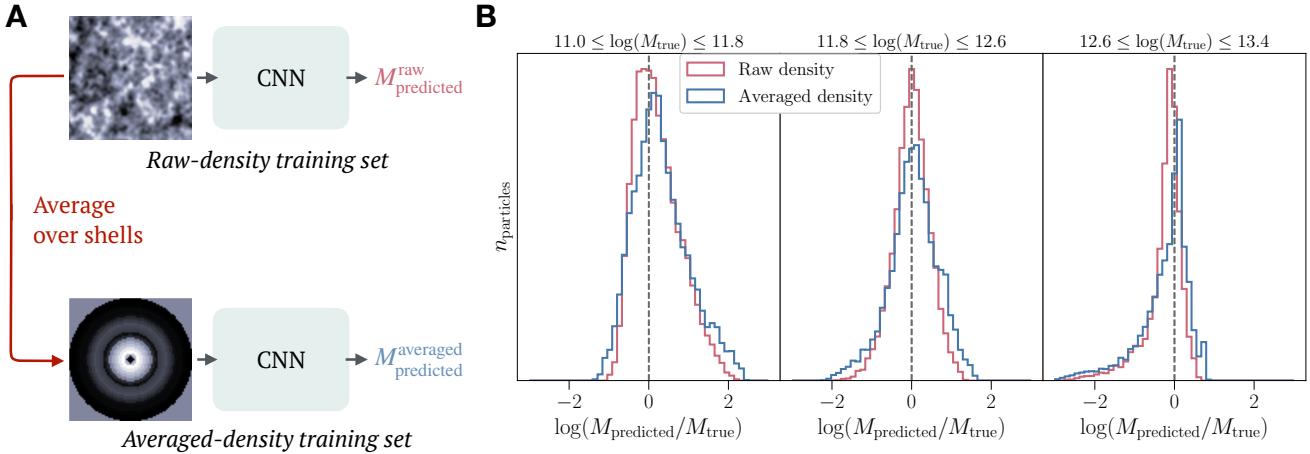


Figure 3 | (A) We re-train the model on inputs where the density in the initial conditions is averaged over shells so that any anisotropic information is removed. The two models each return a set of predictions for the test set particles. **(B)** We compare the predictions returned by the *raw-density* training set model and the *averaged-density* training set model. The histograms show the difference between the predicted and the true log halo mass for particles split into three mass bins of halos. The two models show similar predictive accuracy, meaning that the CNN identifies features which carry no more information than spherical averages over the initial density field.

and one trained only on spherically-averaged information. We find that the two models return qualitatively similar predictions in the halo mass range $11 \leq \log(M/M_\odot) \leq 13.4$, for the same set of test particles. As before, we quantify the difference between the two CNN models using the K-L divergence between the distribution of residuals predicted by the *raw-density* training set model, $n_{\text{raw}}(\Delta)$ and that predicted by the *averaged-density* training set model, $n_{\text{avg}}(\Delta)$, as in Eq. (2). We find $D_{\text{K-L}}(n_{\text{raw}} \parallel n_{\text{avg}}) = 0.03$, similar to the value of the K-L divergence found when comparing the CNN to the GBT model. This level of similarity also applies when comparing the predictions for particles in the innermost region, in the middle and in the outskirts of halos in three different mass bins. However, we note a slightly larger variance in the residuals of the *averaged-density* predictions for particles in the outskirts of low-mass halos, when compared to the predictions for the same particles in the *raw-density* model. These results confirm that the performance of the CNN does not significantly degrade if one removes from the inputs the anisotropic information about the density field around the particle. Therefore, the features learnt by the CNN when given the raw-density initial field carry no more information than spherically averaged densities.

Demonstrating the ability of the CNN to extract features

One fundamental assumption behind this interpretation of our results is that the CNN is capable of capturing features across the range of different scales present in the input sub-box. If instead the CNN's ability to learn were limited, then we could not exclude the possibility that there exists additional information in the inputs which affects halo collapse, but which the CNN is unable to learn. This motivated us to perform tests showing that the same CNN model can return highly accurate predictions, when presented with the information to do so. In particular, we wanted to create a test as closely related to the real initial conditions-to-halo mass problem, which would

specifically demonstrate the ability of the CNN to extract features from the density field, on all scales probed by the input sub-boxes, and return halo mass predictions that are consistent with expectations.

To do this, we tested the performance of the model in a scenario where we could compare the predictions of the CNN to our expectations. We trained the CNN to learn the mapping between the non-linear density field at the present time ($z = 0$) and the mass of the resulting halos. This mapping is effectively given by an algorithm which first identifies the boundary of a halo based on a fixed density threshold, similar to the friends-of-friends algorithm used to identify halos in the simulation, and then computes the mass enclosed within such halo. To do this, the CNN must be able to simultaneously extract features at a number of different scales; from that of the boundary of the lowest mass halos up to that of the most massive ones. As this is a more straightforward mapping than that between the initial conditions density field and the final halo masses, we expect the CNN to return near-perfect predictions.

Similar to the $z = 99$ case, we provided the CNN with the non-linear density field in cubic sub-regions of the simulation, centred at each particle's position. The $z = 99$ and $z = 0$ settings use as inputs the density fields at $t = 17.1$ Myr and $t = 13.7$ Gyr after the Big Bang respectively, while keeping all other choices about the model's architecture and its hyperparameters identical. Fig. 4 shows the predictions of the CNN when trained on the $z = 0$ non-linear density field. Just like in Fig. 2B, the predictions are illustrated in the form of violin plots, showing the distributions of predicted halo masses in bins of true mass. As expected, the predictions show good agreement with the true halo mass labels, yielding a correlation coefficient $r = 0.96$, where $r = 1$ implies an exact linear relationship. The presence of a very low number of outliers, that make up the visible tails of the violin plots, is expected from any machine learning model trained from a finite dataset. Since the predictions are highly accurate throughout the full

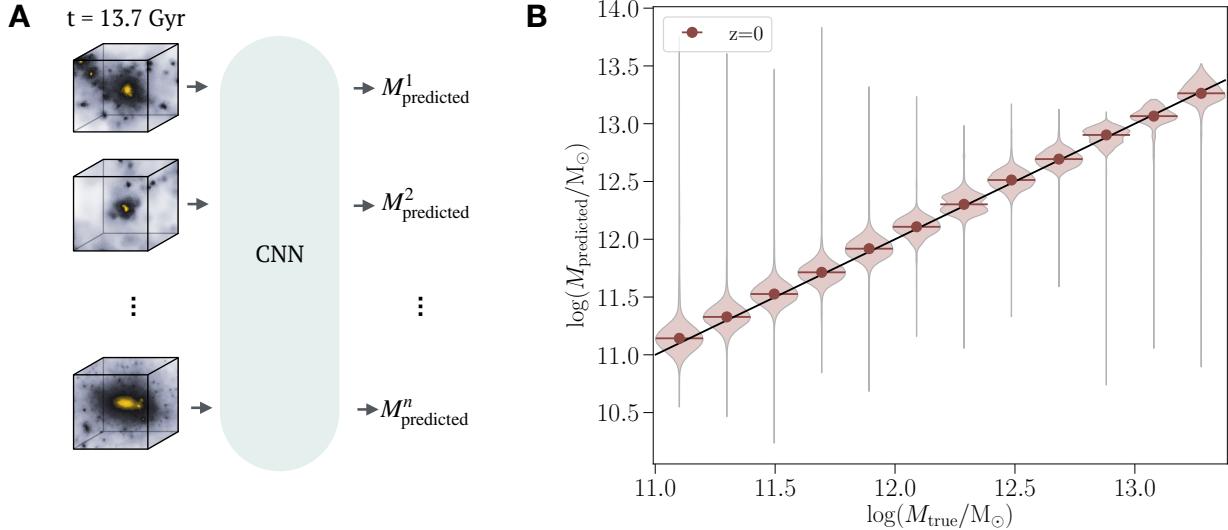


Figure 4 | Halo mass predictions returned by a CNN trained on the non-linear density field at $z = 0$. The predictions are shown in the form of violin plots i.e., distributions (and their medians) of predicted halo masses of particles within evenly-spaced bins of true logarithmic halo mass. The predictions are in excellent agreement with their respective ground truth halo masses, yielding a correlation coefficient $r = 0.96$.

mass range of halos, the CNN must be able to identify the relevant features from the density field on all scales within the input sub-box and yield predictions within expected accuracy. Consequently, we expect the same architecture to also have the capability to extract features on multiple scales within the $z = 99$ linear density field. However, since the features in the initial conditions have a much more complex relationship with halo mass, the predictions will naturally be less accurate than the $z = 0$ case.

Our CNN architecture does not contain constraints on the nature of the kernels in the convolutional layers; this means that there is no reason why the kernels would prefer spherically-symmetric features over non-spherical ones. If the CNN were to preferentially identify spherical features, then it would only predict the correct mass for halos that are spherical, leading to a larger number of outliers than those observed in Fig. 4.

DISCUSSION

We have presented a deep learning framework, capable of learning final halo masses directly from the linear density field in the initial conditions of an N -body simulation. The overall goal of our work is to use deep learning for *knowledge extraction*: that is, we would like to learn about physical aspects of the early Universe which impact the formation of late-time halos using the results of deep learning, without the need to featurize the initial conditions. To do this, we require a deep learning framework that allows for the interpretability of its learning; for example, in understanding the features assembled by the convolutional layers and how these map onto the final predictions. In this work, we intentionally removed part of the information from the inputs and re-trained the CNN to test the impact of this on the accuracy of the final predictions. We found that the performance of the deep learning model does not change if we remove anisotropic information about the initial density field from the model’s inputs. This result implies that the information in the initial conditions gleaned by the

deep learning model to infer halo masses at $z = 0$ is equivalent to spherical averages over the initial density field. A crucial test of robustness of our framework was to demonstrate that the deep learning model can effectively extract spatially-local features on all scales probed by the input sub-boxes and yield robust halo mass predictions that match expectations for a simpler test-case scenario.

Our framework differs from other applications of 3D CNNs to cosmological simulations primarily in three regards. First, most applications aim to construct models that can be used as fast alternatives to expensive computational simulations [44, 46–49]. Our aim is to make use of deep learning to gain physical insight into the emergence of cosmic structures within the simulations. Second, our CNN model returns particle-specific predictions. This yields a halo collapse model that can describe the non-linear evolution of the density field from any initial location in the simulation. By contrast, prior work often trains 3D CNNs to infer global quantities from the simulation, such as cosmological parameters [38–40, 42], the abundance of galaxies [50, 55] or the $z = 0$ displacement field across the entire simulation [46]. Typically, predicting global quantities requires a large number of training simulations since each simulation represents one single training example; in our work, since the training data consist of simulation particles, we only require 20 cosmological simulations, as each contains millions of particles. The third difference is in the evaluation step. We evaluate the performance of the model using localized particle-based metrics, which directly test how well the outputs of the CNN match their respective ground truths. On the other hand, existing works often evaluate the performance of their model on global summary statistics such as two-point or three-point correlation functions [38, 39, 42, 48, 50]. Although these are observables commonly used in cosmology, they only contain limited information about the performance of the deep learning algorithm and may therefore wash out informative aspects about the model’s predictions or hide limitations of the model.

In future work, we will extend the current deep learning framework to one which exploits synergies between convolutional neural networks and variational auto-encoders (VAEs) [36, 56, 57]. VAEs can be used to reduce the 3D initial density field into a lower-dimensional representation known as a *latent space*. The latent variables will be used as input to a feed-forward neural network, trained to predict the mass (or any other property) of the final dark matter halos. The latent variables provide us with an automatically generated set of features, containing all the relevant information about final halos, which can be interpreted in relation to physical aspects of the initial density field. Further extensions of the work also involve providing the deep learning model with multiple linear fields, such as the linear density field, the velocity field or the gravitational potential; this is straightforwardly achievable by exploiting the existing multiple ‘channels’ infrastructure already established for RGB channels in images. Although the density, velocity, and gravitational fields have a one-to-one correspondence with one another in the whole simulation box, they carry different information when evaluated in sub-boxes which cover only a sub-region of the simulation. The inputs to a CNN model trained on 2D images are effectively 3D, where the third dimension represents colour space; similarly, our 3D inputs will become 4D in cases where multiple linear fields sampled within the initial box are provided as inputs. Finally, our framework can be extended to gain physical insights into the origin of other properties of dark matter halos – such as their spin, density profile, or their assembly history – or into the formation of other cosmic structures such as voids. Our results illustrate the promise of interpretable deep learning frameworks as powerful tools for extracting new insights into cosmological structure formation.

DATA AVAILABILITY

The software used to generate the simulations are available at <https://github.com/pynbody/genetIC> to generate the initial conditions, and at <https://wwwmpa.mpa-garching.mpg.de/gadget/> to run the N -body simulations. The parameter files for generating the training data can be made available from the authors upon reasonable request.

CODE AVAILABILITY

The code used to conduct the analysis can be made available from the authors upon reasonable request.

ACKNOWLEDGEMENTS

LLS thanks Corentin Cadiou, Joao Caldeira, Andres Felipe Alba Hernandez, Michael Maire, Manuel Valentin and Samuel Witte for useful discussions. LLS thanks Stephen Stopyra for help in generating the simulations. HVP thanks Justin Alsing and Daniel Mortlock for useful discussions. LLS acknowledges the hospitality of the Fermi National Accelerator Laboratory, where part of this work was completed, and thanks Josh Frieman and Brian Nord for kindly hosting. HVP acknowledges the hospitality of the Aspen Center for Physics, which is supported by National Science Foundation grant PHY1607611. This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 818085 GMGalaxies. LLS was also supported by the Science and Technology Fa-

cilities Council. HVP was partially supported by the research project grant “Fundamental Physics from Cosmological Surveys” funded by the Swedish Research Council (VR) under Dnr 2017-04212. This project has received support from the research project grant “Understanding the Dynamic Universe” funded by the Knut and Alice Wallenberg Foundation under Dnr KAW 2018.0067. AP was supported by the Royal Society. This work was partially supported by the Visiting Scholars Award Program of the Universities Research Association, and by funding from the UCL Cosmoparticle Initiative. This work used computing facilities provided by the UCL Cosmoparticle Initiative; and we thank the HPC systems manager Edd Edmondson for his committed support. This work was also partially supported by Wave 1 of The UKRI Strategic Priorities Fund under the EPSRC Grant EP/T001569/1, particularly the “AI for Science” theme within that grant & The Alan Turing Institute, and by the Benchmarking for AI for Science at Exascale (BASE) project under the EPSRC Grant EP/V001310/1. Work supported by the Fermi National Accelerator Laboratory, managed and operated by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for U.S. Government purposes.

AUTHOR CONTRIBUTIONS

L.L.-S.: led the project; project conceptualisation; methodology; software; obtained, validated, and interpreted results; writing - original draft, editing, final; funding acquisition. **H.V.P & A.P.:** project conceptualisation; methodology; investigation, validation & interpretation; supervision; writing - editing; funding acquisition. **B.N.:** methodology; investigation; writing - editing; provided resources. **J.T.:** validation; writing - editing; provided resources.

COMPETING INTERESTS

The authors declare no competing interests.

REFERENCES

- [1] Blumenthal, G. R., Faber, S. M., Primack, J. R. & Rees, M. J. Formation of galaxies and large-scale structure with cold dark matter. *Nature* **311**, 517–525 (1984).
- [2] Percival, W. J. *et al.* The 2dF Galaxy Redshift Survey: the power spectrum and the matter content of the Universe. *MNRAS* **327**, 1297–1306 (2001).
- [3] White, S. D. M. & Rees, M. J. Core condensation in heavy halos: a two-stage theory for galaxy formation and clustering. *MNRAS* **183**, 341–358 (1978).
- [4] Planck Collaboration *et al.* Planck 2018 results. VI. Cosmological parameters. *A&A* **641**, A6 (2020).
- [5] Efstathiou, G., Davis, M., White, S. D. M. & Frenk, C. S. Numerical techniques for large cosmological N-body simulations. *ApJS* **57**, 241–260 (1985).
- [6] Jenkins, A. *et al.* The mass function of dark matter haloes. *MNRAS* **321**, 372–384 (2001).

- [7] Navarro, J. F., Frenk, C. S. & White, S. D. M. A Universal Density Profile from Hierarchical Clustering. *ApJ* **490**, 493–508 (1997).
- [8] Springel, V., Yoshida, N. & White, S. D. M. GADGET: a code for collisionless and gasdynamical cosmological simulations. *New Astronomy* **6**, 79–117 (2001).
- [9] Boylan-Kolchin, M., Springel, V., White, S. D. M., Jenkins, A. & Lemson, G. Resolving cosmic structure formation with the Millennium-II Simulation. *MNRAS* **398**, 1150–1164 (2009).
- [10] Klypin, A. A., Trujillo-Gomez, S. & Primack, J. Dark Matter Halos in the Standard Cosmological Model: Results from the Bolshoi Simulation. *ApJ* **740**, 102 (2011).
- [11] Press, W. H. & Schechter, P. Formation of Galaxies and Clusters of Galaxies by Self-Similar Gravitational Condensation. *ApJ* **187**, 425–438 (1974).
- [12] Bond, J. R., Cole, S., Efstathiou, G. & Kaiser, N. Excursion set mass functions for hierarchical Gaussian fluctuations. *ApJ* **379**, 440–460 (1991).
- [13] Doroshkevich, A. G. The space structure of perturbations and the origin of rotation of galaxies in the theory of fluctuation. *Astrofizika* **6**, 581–600 (1970).
- [14] Bond, J. R. & Myers, S. T. The Peak-Patch Picture of Cosmic Catalogs. I. Algorithms. *ApJS* **103**, 1 (1996).
- [15] Sheth, R. K. & Tormen, G. Large-scale bias and the peak background split. *MNRAS* **308**, 119–126 (1999).
- [16] Sheth, R. K., Mo, H. J. & Tormen, G. Ellipsoidal collapse and an improved model for the number and spatial distribution of dark matter haloes. *MNRAS* **323**, 1–12 (2001).
- [17] Sheth, R. K. & Tormen, G. An excursion set model of hierarchical clustering: ellipsoidal collapse and the moving barrier. *MNRAS* **329**, 61–75 (2002).
- [18] Lucie-Smith, L., Peiris, H. V., Pontzen, A. & Lochner, M. Machine learning cosmological structure formation. *MNRAS* **479**, 3405–3414 (2018).
- [19] Lucie-Smith, L., Peiris, H. V. & Pontzen, A. An interpretable machine-learning framework for dark matter halo formation. *MNRAS* **490**, 331–342 (2019).
- [20] LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).
- [21] Bengio, Y. Learning deep architectures for ai. *Foundations and Trends® in Machine Learning* **2**, 1–127 (2009). URL <http://dx.doi.org/10.1561/2200000006>.
- [22] Le, Q. V. *et al.* Building high-level features using large scale unsupervised learning. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* 8595–8598 (2011).
- [23] Olah, C. *et al.* The building blocks of interpretability. *Distill* (2018). URL <https://distill.pub/2018/building-blocks/>.
- [24] Chakraborty, S. *et al.* Interpretability of deep learning models: A survey of results. In *2017 IEEE Smart-World, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation*, 1–6 (2017).
- [25] Murdoch, W. J., Singh, C., Kumbier, K., Abbasi-Asl, R. & Yu, B. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences* **116**, 22071–22080 (2019). URL <http://dx.doi.org/10.1073/pnas.1900654116>.
- [26] Olah, C., Mordvintsev, A. & Schubert, L. Feature visualization. *Distill* **2** (2017).
- [27] Springenberg, J. T., Dosovitskiy, A., Brox, T. & Riedmiller, M. Striving for simplicity: The all convolutional net. *CoRR* **abs/1412.6806** (2014).
- [28] Zeiler, M. D. & Fergus, R. Visualizing and understanding convolutional networks. In Fleet, D., Pajdla, T., Schiele, B. & Tuytelaars, T. (eds.) *Computer Vision – ECCV 2014*, 818–833 (Springer International Publishing, Cham, 2014).
- [29] Fong, R. & Vedaldi, A. Interpretable explanations of black boxes by meaningful perturbation. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 3449–3457 (2017).
- [30] Selvaraju, R. R. *et al.* Grad-cam: Visual explanations from deep networks via gradient-based localization. *2017 IEEE International Conference on Computer Vision (ICCV)* 618–626 (2016).
- [31] Simonyan, K., Vedaldi, A. & Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR* **abs/1312.6034** (2013).
- [32] Zhou, B., Khosla, A., Lapedriza, Á., Oliva, A. & Torralba, A. Learning deep features for discriminative localization. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 2921–2929 (2015).
- [33] Bengio, Y., Courville, A. & Vincent, P. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* **35**, 1798–1828 (2013).
- [34] Hinton, G. E. & Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *Science* **313**, 504–507 (2006). URL <https://science.sciencemag.org/content/313/5786/504>.
- [35] Higgins, I. *et al.* beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR* (2017).
- [36] Iten, R., Metger, T., Wilming, H., del Rio, L. & Renner, R. Discovering Physical Concepts with Neural Networks. *Phys. Rev. Lett.* **124**, 010508 (2020).
- [37] Nautrup, H. P. *et al.* Operationally meaningful representations of physical systems in neural networks. Preprint at arXiv:2001.00593 (2020).
- [38] Ravanbakhsh, S. *et al.* Estimating cosmological parameters from the dark matter distribution. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML’16, 2407–2416 (JMLR.org, 2016).
- [39] Mathuriya, A. *et al.* Cosmoflow: Using deep learning to learn the universe at scale. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis, SC ’18* (IEEE Press, 2018). URL <https://doi.org/10.1109/SC.2018.00068>.

- [40] Pan, S. *et al.* Cosmological parameter estimation from large-scale structure deep learning. *Science China Physics, Mechanics, and Astronomy* **63**, 110412 (2020).
- [41] Villaescusa-Navarro, F. *et al.* Neural networks as optimal estimators to marginalize over baryonic effects. Preprint at arXiv:2011.05992 (2020).
- [42] Ntampaka, M., Eisenstein, D. J., Yuan, S. & Garrison, L. H. A Hybrid Deep Learning Approach to Cosmological Constraints from Galaxy Redshift Surveys. *ApJ* **889**, 151 (2020).
- [43] Moster, B. P., Naab, T., Lindström, M. & O’Leary, J. A. GalaxyNet: Connecting galaxies and dark matter haloes with deep neural networks and reinforcement learning in large volumes. Preprint at arXiv:2005.12276 (2020).
- [44] Berger, P. & Stein, G. A volumetric deep Convolutional Neural Network for simulation of mock dark matter halo catalogues. *MNRAS* **482**, 2861–2871 (2019).
- [45] Bernardini, M., Mayer, L., Reed, D. & Feldmann, R. Predicting dark matter halo formation in N-body simulations with deep regression networks. *MNRAS* **496**, 5116–5125 (2020).
- [46] He, S. *et al.* Learning to predict the cosmological structure formation. *Proceedings of the National Academy of Science* **116**, 13825–13832 (2019).
- [47] Charnock, T. *et al.* Neural physical engines for inferring the halo mass distribution function. *MNRAS* **494**, 50–61 (2020).
- [48] Kodi Ramanah, D., Charnock, T. & Lavaux, G. Painting halos from cosmic density fields of dark matter with physically motivated neural networks. *Phys. Rev. D* **100**, 043515 (2019).
- [49] Modi, C., Feng, Y. & Seljak, U. Cosmological reconstruction from galaxy light: neural network based light-matter connection. *J. Cosmology Astropart. Phys.* **2018**, 028 (2018).
- [50] Zhang, X. *et al.* From Dark Matter to Galaxies with Convolutional Networks. Preprint at arXiv:1902.05965 (2019).
- [51] Nair, V. & Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML’10, 807–814 (Omnipress, USA, 2010). URL <http://dl.acm.org/citation.cfm?id=3104322.3104425>.
- [52] Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. *Nature* **323**, 533–536 (1986).
- [53] Scardapane, S., Comminiello, D., Hussain, A. & Uncini, A. Group sparse regularization for deep neural networks. *Neurocomputing* **241**, 81–89 (2017).
- [54] Kullback, S. & Leibler, R. A. On information and sufficiency. *Ann. Math. Statist.* **22**, 79–86 (1951). URL <https://doi.org/10.1214/aoms/1177729694>.
- [55] Yip, J. H. *et al.* From Dark Matter to Galaxies with Convolutional Neural Networks. In *33rd Annual Conference on Neural Information Processing Systems* (2019).
- [56] Kingma, D. P. & Welling, M. Auto-encoding variational bayes. *CoRR* **abs/1312.6114** (2014).
- [57] Rezende, D. J., Mohamed, S. & Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. vol. 32 of *Proceedings of Machine Learning Research*, 1278–1286 (PMLR, Beijing, China, 2014). URL <http://proceedings.mlr.press/v32/rezende14.html>.
- [58] Springel, V. The cosmological simulation code GADGET-2. *MNRAS* **364**, 1105–1134 (2005).
- [59] Pontzen, A., Roškar, R., Stinson, G. & Woods, R. pynbody: N-Body/SPH analysis for python. *Astrophysics Source Code Library* (2013).
- [60] Dunkley, J. *et al.* Five-Year Wilkinson Microwave Anisotropy Probe Observations: Likelihoods and Parameters from the WMAP Data. *ApJS* **180**, 306–329 (2009).
- [61] Stopyra, S., Pontzen, A., Peiris, H., Roth, N. & Rey, M. GenetIC – a new initial conditions generator to support genetically modified zoom simulations. Preprint at arXiv:2006.01841 (2020).
- [62] Glorot, X. & Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. vol. 9 of *Proceedings of Machine Learning Research*, 249–256 (JMLR Workshop and Conference Proceedings, Chia Laguna Resort, Sardinia, Italy, 2010).
- [63] Nwankpa, C., Ijomah, W., Gachagan, A. & Marshall, S. Activation functions: Comparison of trends in practice and research for deep learning. Preprint at arXiv:1811.03378 (2018).
- [64] Denil, M., Shakibi, B., Dinh, L., Ranzato, M. & de Freitas, N. Predicting parameters in deep learning. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS’13, 2148–2156 (Curran Associates Inc., Red Hook, NY, USA, 2013).
- [65] J. Reddi, S., Kale, S. & Kumar, S. On the convergence of adam & beyond (2018).
- [66] Kingma, D. & Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations* (2014).

METHODS

Simulations

We generated the training data from 20 dark matter-only N -body simulations produced with P-GADGET-3 [8, 58], each consisting of a box of size $L = 50 \text{ Mpc } h^{-1}$ (comoving) and $N = 256^3$ simulation particles evolving from $z = 99$ to $z = 0$. We made use of pynbody [59] to analyse the information contained in the simulation snapshots. The simulations adopt a WMAP5 ΛCDM cosmological model; the cosmological parameters are given by $\Omega_\Lambda = 0.721$, $\Omega_m = 0.279$, $\Omega_b = 0.045$, $\sigma_8 = 0.817$, $h = 0.701$ and $n_s = 0.96$ [60]. Some of the simulations are part of a suite of existing simulations, which were performed at times where these cosmological parameters were up-to-date; the newer simulations were then run with the same set of old parameters for consistency. However, we do not expect our conclusions to change when updating the cosmological parameters to more recent constraints from observations [4], as demonstrated in previous work [18]. Each simulation is based on a different realization of a Gaussian random field drawn from the initial power spectrum of density fluctuations, generated using genetIC [61]. The simulation particles of the

validation and testing data were randomly drawn from additional, independent simulations to those used for training and their inputs/outputs were generated in the same way as for the training data.

Dark matter halos were identified at $z = 0$ using the SUBFIND halo finder [8], a friends-of-friends method with a linking length of 0.2, with the additional requirement that particles in a halo be gravitationally bound. We consider the entire set of bound particles that make up a halo and do not account for substructure within halos. The resolution and volume of the simulation limit the resulting range of halo masses: the lowest-mass halo has $M = 2.6 \times 10^{10} M_{\odot}$ and the highest $M = 4.1 \times 10^{14} M_{\odot}$. We restrict our analysis even further to the mass range $\log(M/M_{\odot}) \in [11, 13.4]$. This is because halos with mass $M \lesssim 10^{11} M_{\odot}$ contain less than ~ 100 particles and are therefore not well resolved in the simulation, whereas halos with mass $M \gtrsim 3 \times 10^{13} M_{\odot}$ are under-represented as a result of the small volume of our simulations.

Inputs and outputs of the deep learning models

The final snapshots of the simulations ($z = 0$) were used to label each dark matter particle with its ground truth variable, given by the logarithmic mass of the dark matter halo to which each dark matter particle belongs. We only consider particles that make up dark matter halos at $z = 0$ in this analysis. The ground truths were rescaled to the range $[-1, 1]$ before training; this step sets a similar scale and dynamic range for the inputs and outputs of the model, which facilitates the model's training.

The initial conditions of the simulations ($z = 99$) were used to generate the deep learning inputs associated with each particle. In the initial conditions, the density field is given by a random realization $\delta(\mathbf{x}, t_{\text{initial}})$ on a uniform 256^3 grid in the $(50 \text{ Mpc } h^{-1})^3$ simulation volume. We generate two different types of inputs, one for the *raw-density* training set model and the other for the *averaged-density* training set (see Results).

The input to the *raw-density* training set model associated with any given particle is given by $\delta(\mathbf{x}, t_{\text{initial}})$ in a cubic sub-region of the full simulation centred on the particle's initial position. This sub-volume has size $L = 15 \text{ Mpc } h^{-1}$ (comoving) and resolution $N = 75^3$. The size of the sub-box was chosen to be large enough to capture large-scale information that is relevant to the algorithm to learn the initial conditions-to-halo mass mapping. When training a GBT to learn about halo formation, given the density field smoothed on different mass scales, we found that the algorithm was able to learn relevant information from the smoothed density field up to a scale of $M_{\text{smoothing}} \sim 10^{14} M_{\odot}$ [19]. Therefore, we chose a sub-box length $L_{\text{box}} = 15 \text{ Mpc } h^{-1}$, which encloses a total mass of $M \sim 4 \times 10^{14} M_{\odot}$, which is more than the largest relevant mass scale for the GBT. The resolution of the sub-box was chosen such that the length of each voxel, l_{voxel} , is the same as the initial grid spacing in the simulation i.e., $l_{\text{voxel}} = 0.2 \text{ Mpc } h^{-1}$ (comoving). The training set inputs were rescaled to have 0 mean and standard deviation 1; the same rescaling was then applied to the validation and test sets.

The input to the *averaged-density* training set model is given by a modified version of the initial density field $\delta(\mathbf{x}, t_{\text{initial}})$ within the same sub-volume as that used for the *raw-density* training set model. Here, we removed the

anisotropic information from the initial density field as follows. Given 20 concentric shells around the centre of the box, evenly spaced in radius r within the range $r \in [2, 75]$ voxels, we computed the average density within each shell. The voxels of the sub-box were then assigned the averaged density within the shell to which each voxel belongs. Outside the largest shell in the box, the voxels were assigned a value of 0 which corresponds to the cosmic mean density in rescaled units (Fig. 3A).

The last set of inputs used in this work were generated for the test case where the CNN was trained to learn the mapping between the non-linear density field at present time ($z = 0$) and the mass of the resulting halos. In this case, the inputs are given by the non-linear density field $\delta(\mathbf{x}, t_{\text{final}})$ at $z = 0$ in a sub-volume centred at each particle's position. We revisited our choices of box size and resolution of the 3D sub-box, as the scales of interest at $z = 0$ naturally differ from those in the initial conditions. We fixed the resolution to that used for the $z = 99$ case, $N = 75^3$, and chose a box size of $L = 1.5 \text{ Mpc } h^{-1}$, which approximately corresponds to the virial radius of a halo with mass $M = 10^{14} M_{\odot}$. These choices resulted in a voxel length $l_{\text{voxel}} \sim 30 \text{ kpc } h^{-1}$, which is approximately equivalent to half the virial radius of a $M = 10^{10} M_{\odot}$ halo. Given that the box captures the virial radius of the largest and smallest halos probed by our simulations, we expect the input boxes to contain the information required by the algorithm to learn the density field-to-halos mapping.

CNN architecture

Our CNN model is composed of three main ingredients:

- *Convolutional layers*: these extract features from the input data by performing convolutions between the input and a number of kernels in every layer. Each kernel of a convolutional layer learns to detect a specific type of feature present in the input. We used three-dimensional kernels whose values, known as *weights w*, are first randomly initialized and subsequently updated during the training process of the CNN. Convolutional layers also carry a number of hyperparameters which control the way convolutions are performed and that must be set prior to training. These include the size of the kernels, which is typically set to be small in order to detect low-level local features across different regions of the input volume; the number of kernels, which dictates the number of individual features detected in a single convolutional layer; the *stride*, the number of pixels by which to slide the kernel across the input when performing the convolution; and the amount of *zero-padding*, whether to pad the input boxes with zeros around the borders so that the kernels can be centred on elements at the edge of the box. The end product of each convolution is a 3D *feature map*, indicating the strength and location of the feature detected by that kernel. A non-linear activation function is then applied to every feature map. By stacking multiple convolutional layers onto each other, low-level local features assembled in the first layers are then combined into high-level global features by subsequent layers.

- *Pooling layers*: these decrease the resolution of the feature maps by taking the average (average-pooling) or the maximum value (max-pooling) in small regions of

the feature maps. The size of the pooling region is the only hyperparameter in this layer, which does not contain trainable parameters. The effect of the pooling layer is to produce lower-resolution feature maps, which are less sensitive to small changes in the position of the feature in the input compared to the higher-resolution feature maps returned by the convolutional layer.

- *Fully-connected layers:* these combine the features assembled in the convolutional layers and return a single value representing the halo mass prediction variable. Each layer is made of a number of neurons, such that every neuron in one layer is connected to every neuron in adjacent layers. Each neuron follows $y = \sigma(\mathbf{w} * \mathbf{x} + b)$, where \mathbf{x} are the inputs, y is the output, σ is the non-linear activation function and \mathbf{w}, b are trainable parameters known as weights and biases.

Our deep learning architecture consists of six convolutional layers, all but the first one followed by max-pooling layers, and three fully-connected layers. The convolutions were performed with 32, 32, 64, 128, 128, 128 kernels for the six convolutional layers, respectively – all with a stride of 1 and zero-padding. The initial weights of the kernels in a layer were set following the Xavier initialization technique [62], which randomly draws values from a uniform distribution bounded between $\pm\sqrt{6}/(n_i + n_{i+1})$, where n_i and n_{i+1} are the number of incoming and outgoing network connections to that layer. The kernels have size $3 \times 3 \times 3$ in all convolutional layers, meaning that the first layer learns features on scales of $0.6 \text{ Mpc } h^{-1}$. As more convolutional layers are stacked on top of each other, the algorithm becomes sensitive to features at increasing scales. In this way, both local and global information are able to propagate through the network. We applied a non-linear activation function to every feature map given by a leaky rectified linear unit (LeakyReLU) [51]

$$f(x) = \begin{cases} x & \text{for } x \geq 0, \\ \beta \times x & \text{for } x < 0, \end{cases} \quad (3)$$

with $\beta = 0.03$. A LeakyReLU activation, with β of order 10^{-2} , is a common choice that has proved successful in many deep learning applications [63]. The feature maps are then fed to max-pooling layers, which reduce their dimensionality by taking the average over $2 \times 2 \times 2$ non-overlapping regions of the feature maps.

After the sixth convolutional layer and subsequent pooling layer, the output is flattened into a one-dimensional vector and fed to a series of 3 fully-connected layers, each made of 256 and 128 and 1 neuron, respectively. The non-linear activation function of the first two layers is the same ReLU activation (Eq. 3) as that used in the convolutional layers, whereas the last layer has a linear activation in order for the output to represent halo mass. The weights were initialized using the same Xavier initialization technique used for the kernel weights of the convolutional layers. Regularization in the convolutional and fully-connected layers was incorporated in the form of priors over the parameters of the model, as explained in the next subsection.

We chose the architecture that returned the best performance (i.e., the lowest loss score on the validation set af-

ter convergence) amongst many, but not all, alternative models with different choices of architecture-specific and layer-specific hyperparameters. We investigated the change in the validation loss in response to the following modifications: adding batch-normalization layers; introducing dropout; varying the amount of dropout; adding/removing convolutional layers and/or fully-connected layers; increasing/decreasing the number of kernels/neurons in each convolutional/fully-connected layer; changing the weight initializer; changing the convolutional kernel size. In all cases, we found that the final loss score either increased or showed no change compared to that of the architecture retained in this work. We leave further hyperparameter exploration, including changes to the optimizer, the addition of skip connections, and other variations in the architecture, to future work.

The loss function

A neural network can be viewed as a probabilistic model $p(y | \mathbf{x}, \mathbf{w})$, where given an input \mathbf{x} , a neural network assigns a probability to each possible output y , using the set of parameters \mathbf{w} . The parameters are learnt via maximum likelihood estimation (MLE): given a set of training examples $\mathcal{D} = \{x_i, d_i\}_{i=1}^N$, the optimal weights are those that minimize the negative log-likelihood, $\ln[p(\mathcal{D} | \mathbf{w})] = \sum_{i=1}^N \ln[p(d_i | x_i, \mathbf{w})]$, more generally called the loss function \mathcal{L} in the machine learning community. The issue is that deep neural networks are generally over-parametrized; it has in fact been demonstrated that there exists a major redundancy in the parameters used by a deep neural network [64]. This means that when minimizing the negative log-likelihood with a deep neural network model, one almost always encounters the problem of overfitting. The algorithm tends to fit the samples of the training data \mathcal{D} extremely well but fails to learn patterns that are generalizable to unseen data. To overcome this issue, one modifies the loss function of the neural network in such a way that prevents the algorithm from overfitting and improves its generalizability. This is known as regularization.

We introduce regularization by adopting priors over the weights. Following Bayes' theorem, the goal of the neural network then becomes to maximize the posterior distribution $p(\mathbf{w} | \mathcal{D}) = p(\mathcal{D} | \mathbf{w}) p(\mathbf{w})$, rather than the likelihood $p(\mathcal{D} | \mathbf{w})$. The loss function, \mathcal{L} , is then given by

$$\mathcal{L} = -\ln[p(\mathbf{w} | \mathcal{D})] = -\ln[p(\mathcal{D} | \mathbf{w})] - \ln[p(\mathbf{w})], \quad (4)$$

where the first is the likelihood term, or predictive term $\mathcal{L}_{\text{pred}}$, and the second is the prior term, or regularization term \mathcal{L}_{reg} , as in Eq. (1). If \mathbf{w} are given a Gaussian prior, this yields L2 regularization; if \mathbf{w} are given a Laplacian prior, then one obtains L1 regularization. The advantage of this form of regularization is that it can be incorporated in terms of priors on the weights. There exist many other regularization techniques, including for example dropout, but we choose to focus on those that have a direct Bayesian interpretation.

Technically, the parameters optimized during training include not just the weights, but also the biases. These consist of a constant value that is added to the product of inputs and weights for every kernel (neuron) in a convolutional (fully-connected) layer. These parameters add little flexibility to the model and are therefore typically not responsible for overfitting. Therefore, we choose not to consider setting priors on

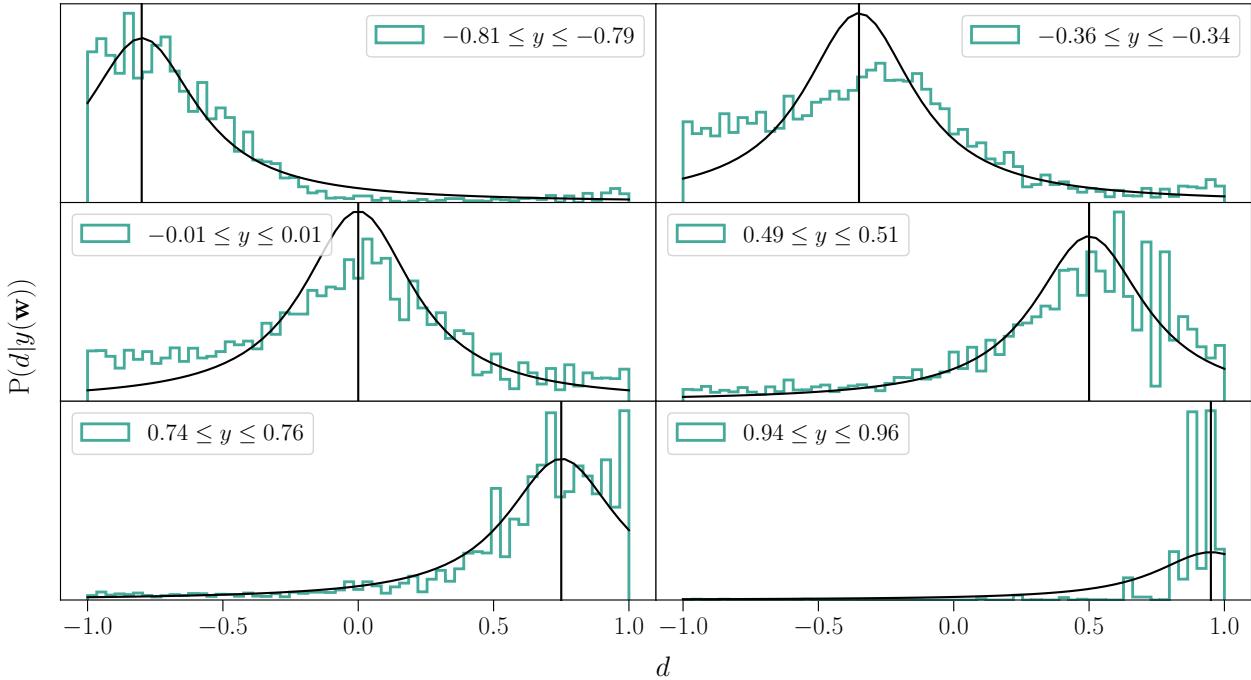


Figure 5 | Distribution of d for fixed slices in y , where d is the ground truth logarithmic halo mass variable rescaled to the range $[-1, 1]$ and y is the predicted value returned by the CNN in rescaled units. The black line is the likelihood function given the value of γ at the epoch where the validation loss reaches its minimum, $\gamma = 0.26$.

the biases as they do not require regularization.

The choice of the likelihood function

We denote $d = d(\mathbf{x})$ as the ground truth variable rescaled to $[-1, 1]$ and $y = y(\mathbf{x}, \mathbf{w})$ as the prediction returned by the CNN model with weights \mathbf{w} , for a given set of rescaled inputs \mathbf{x} . The likelihood function describes the distribution of ground truth values d for a given value of predicted output y , returned from the neural network model with weights \mathbf{w} . A typical choice in the field is that of a Gaussian or Laplacian likelihood, yielding the popular mean-squared-error or

mean-absolute-error losses. For our problem, we found that a Gaussian distribution is a poor description of the training data: the distributions of d for fixed values of y contain long tails, especially when y is close to the boundaries $y = -1$ and $y = 1$, that a Gaussian distribution fails to account for. Not accounting for these tails led to biased predictions, especially towards the boundaries. Instead, we chose a Cauchy distribution function for the likelihood, characterized by the scale parameter γ , which has broader tails than a Gaussian and a well-defined form for its conditional distribution function.

The negative log-likelihood then becomes

$$-\ln [p(d | y, S)] = \frac{1}{N} \sum_{i=1}^N \left[\ln(\gamma) + \ln \left[1 + \left(\frac{d_i - y_i}{\gamma} \right)^2 \right] + \ln \left[\arctan \left(\frac{d_{\max} - y_i}{\gamma} \right) - \arctan \left(\frac{d_{\min} - y_i}{\gamma} \right) \right] \right] \quad (5)$$

for a Cauchy likelihood function with scale parameter γ , under a top-hat selection function S over the ground truth variable, $p(S | d) = \Theta(d_{\max} - d) \Theta(d - d_{\min})$, where Θ is the Heaviside step function. The latter arises from the fact that, by construction, the rescaled ground truth is restricted to $d_{\min} \leq d \leq d_{\max}$, where $d_{\min} = -1$ and $d_{\max} = 1$. This selection function was needed in order to correctly model the loss at the boundaries. The first two terms in Eq. (5) arise from the Cauchy likelihood: the first is effectively a prior on γ which is insensitive to the predictions y , and the second measures the difference between predicted and ground truth values weighed by the scale parameter γ . The third term in Eq. (5) comes from accounting for the selection function S . The normalization factor $1/N$ is not part of the negative log-likelihood but

is typically introduced in the loss function so that the loss is insensitive to the size of the training set (or, of the batch size if performing batch gradient descent when training). The scale parameter γ determines the half-width at half-maximum of the Cauchy distribution. Since the optimal value of γ is not known a priori, we optimize that parameter during training using back-propagation, together with the rest of the weights and biases optimized by the network. The above likelihood term in the loss function satisfies our desiderata of having a heavy-tailed probability distribution function and accounting for the restricted range of ground truth d .

The expression in Eq. (5) is valid under the condition that $d, y \in [-1, 1]$. However, since the activation function in the last layer is given by the unbounded linear function $\sigma(z) = z$,

the predictions can technically take any value $y \in \mathbb{R}$. To solve this, we introduced a super-exponential function, denoted as $f(y)$, in the regime $|y| \geq 1$, to counter balance the Cauchy limits at the boundaries and sharply disfavour predictions outside the interval $[-1, 1]$. The function is continuously matched to the Cauchy distribution at the boundaries $y = \pm 1$. The likelihood term of the loss function $\mathcal{L}_{\text{pred}}$ is then given by a piecewise function conditional on y ;

$$\mathcal{L}_{\text{pred}} = \frac{1}{N} \sum_{i=1}^N [-\ln p(d_i | y_i, S) \Theta(|y| + 1) + f(y_i) \Theta(|y| - 1)]. \quad (6)$$

Fig. 5 compares the form of the likelihood, given the optimized value for γ returned by the model, compared to the empirical distribution of ground truth values at fixed slices in y , the predicted variable returned by the trained CNN, for the training set samples. The Cauchy likelihood provides a good fit to the empirical likelihood distribution of the model. However, we note that for small values of y , the fit could have been improved by adopting a two-tailed Cauchy likelihood function. Further flexibility could be provided by using a Student's t-distribution. We leave this to future work.

The choice of priors: regularization and model compression

We adopt weight priors that can simultaneously (i) improve the optimization during training by preventing overfitting and (ii) compress the neural network model into the least number of parameters without loss in performance. Regularization and model compression are very much related: these tasks can be achieved simultaneously by minimizing a properly defined cost function. We selected weight priors that penalize large values (for regularization) and induce sparsity (for model compression). To regularize the network, we adopted weight priors that promote smaller values, as these typically lead to more generalizable solutions. We chose Gaussian priors for the weights of the convolutional layers and Laplacian priors for the weights of the fully-connected layers, which penalize the sum of the squared values or the sum of the absolute values of the weights, respectively. The choice of Laplacian prior has the additional benefit that it induces sparsity on the weights by driving most weights to be zero; a Laplacian prior thus combines the idea of model compression and regularization. For model compression, our aim is to induce a more compact network with the smallest number of non-zero neurons in the fully-connected layers. To do this, we adopted the group Lasso formulation [53] which imposes group-level sparsity, meaning that all the variables in that group are either simultaneously set to 0, or none of them are. For the case of fully-connected layers, a group is equivalent to an entire neuron.

The log priors over the weights become

$$\begin{aligned} \ln p(w) = \alpha & \left[\sum_{l \in l_c} \sum_{p=1}^{P_l} \left(w_p^{(l)} \right)^2 + \sum_{l \in l_d} \sum_{q=1}^{Q_l} |w_q^{(l)}| \right. \\ & \left. + \sum_{l \in l_d} \sum_{i=1}^{N_{l-1}} \left[\sum_{j=1}^{N_l} w_{ij}^2 \right]^{1/2} \right], \end{aligned} \quad (7)$$

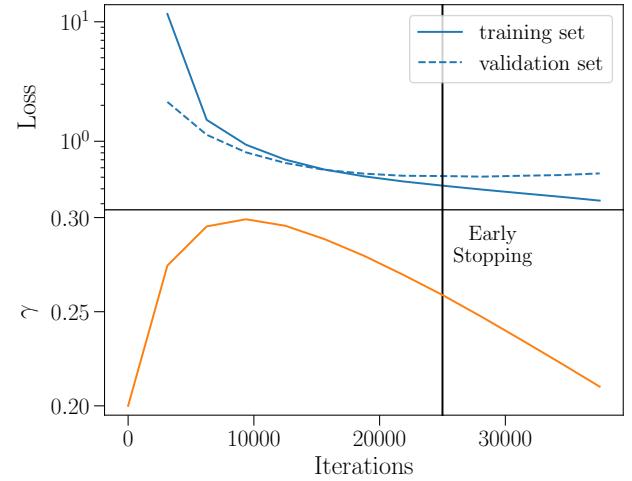


Figure 6 | Top panel: The loss function evaluated for the training set and the validation set at each batch iteration. Early stopping was employed to interrupt the training at the epoch where the validation loss reaches its minimum. The weights of the CNN at the early-stopping iteration were retained. **Bottom panel:** The half-width at half-maximum parameter of the Cauchy likelihood function, γ , as a function of iteration.

where the first term is a Gaussian prior over each of the P_l weights of each convolutional layer l_c , $w_p^{(l)}$, the second term is a Laplacian prior over each of the Q_l weights of each fully-connected layer l_d , $w_q^{(l)}$, and the third term is a group Lasso prior over the set of weights that determine the connections between a single neuron in the $(l-1)$ -th layer and all the neurons in the l -th layer. The idea of group-level sparsity can also be applied to convolutional layers, where a single group is given by the collection of weights from a single kernel of the layer. This can be thought of as a feature selection method, in that it removes entire kernels (and thus the feature represented by that kernel) within each convolutional layer. Given that our network is relatively small, we chose not to perform feature selection; we leave this for future work.

The prior term $\ln [p(w)]$ is added to the likelihood term in the loss function, as in Eq. (4). The regularization parameter α in Eq. (7) weighs the prior term relative to the likelihood term in the loss function. Its value sets the balance between an overly-complex model (which overfits and has a high variance) and an overly-simple model (which underfits and has a high bias). We optimized this parameter, in combination with the learning rate, using cross-validation.

Training and optimization

The algorithm was trained on 200,000 particles, randomly drawn from the ensemble of particles of 20 simulations based on different realizations of the initial conditions. We found no improvement in the performance of the algorithm as we added to the training set an additional 300,000 particles from another independent simulation, implying that our choices were sufficient to yield a training set representative of the mapping between initial conditions and halos. The training set was subdivided into batches, each made of 64 particles. Batches were fed to the network one at a time, and each time the CNN updates its parameters according to the samples in that batch.

Training was done using the AMSGrad optimizer [65], a variant of the widely-used Adam optimizer [66], with a learn-

ing rate of 0.00005. The learning rate was optimized via cross-validation, together with α , the parameter weighting the regularization term in the loss function. The number of trained parameters in the network is 2,108,258. Fig. 6 shows the loss function and the values of the parameter γ in the likelihood term of the loss as a function of the number of iterations. Early stopping was employed to interrupt the training at the epoch where the validation loss reaches its minimum value; the early-stopping iteration is shown as a vertical grey line in Fig. 6. The final weights of the CNN and the optimized value of γ are given by those characterizing the model at the end of the early-stopping iteration.

Validation and testing was performed on particles from an independent simulation based on a different realization of the initial density field to those used for training. Although the validation set does not directly enter the training process of the algorithm, it is indirectly used to test the response of the algorithm to changes in the architecture, and to determine the stopping point for training. Validating and testing on independent realizations ensures that the algorithm is not overfitting patterns specific to the simulations used for training. Instead, it ensures that the CNN is learning physical connections between the initial conditions and the final halos which are generalizable to any realization of the initial density field.