Monthly Notices
of the
ROYAL ASTRONOMICAL SOCIETY

MNRAS **484,** 282–293 (2019)
Advance Access publication 2019 January 14

doi:10.1093/mnras/stz010

Deep learning from 21-cm tomography of the cosmic dawn and reionization

Nicolas Gillet [1]★ Andrei Mesinger,[1] Bradley Greig,[2,3] Adrian Liu[4,5]† and Graziano Ucci [1]

[1]*Scuola Normale Superiore, Piazza dei Cavalieri 7, I-56126 Pisa, Italy*
[2]*ARC Centre of Excellence for All-Sky Astrophysics in 3 Dimensions (ASTRO 3D), University of Melbourne, VIC 3010, Australia*
[3]*School of Physics, University of Melbourne, Parkville, VIC 3010, Australia*
[4]*Department of Astronomy and Radio Astronomy Laboratory, University of California Berkeley, Berkeley, CA 94720, USA*
[5]*Department of Physics and McGill Space Institute, McGill University, Montreal QC H3A 2T8, Canada*

Accepted 2018 December 20. Received 2018 November 27; in original form 2018 May 7

ABSTRACT

The 21-cm power spectrum (PS) has been shown to be a powerful discriminant of reionization and cosmic dawn astrophysical parameters. However, the 21-cm tomographic signal is highly non-Gaussian. Therefore there is additional information which is wasted if only the PS is used for parameter recovery. Here we showcase astrophysical parameter recovery directly from 21-cm images, using deep learning with convolutional neural networks (CNN). Using a data base of 2D images taken from 10 000 21-cm light-cones (each generated from different cosmological initial conditions), we show that a CNN is able to recover parameters describing the first galaxies: (i) $T_{\rm vir}$, their minimum host halo virial temperatures (or masses) capable of hosting efficient star formation; (ii) ζ , their typical ionizing efficiencies; (iii) $L_{\rm X}$/SFR , their typical soft-band X-ray luminosity to star formation rate; and (iv) E_0 , the minimum X-ray energy capable of escaping the galaxy into the IGM. For most of their allowed ranges, log $T_{\rm vir}$ and log $L_{\rm X}$/SFR are recovered with < 1 per cent uncertainty, while ζ and E_0 are recovered with $\sim$10 per cent uncertainty. Our results are roughly comparable to the accuracy obtained from Monte Carlo Markov Chain sampling of the PS with 21CMMC for the two mock observations analysed previously, although we caution that we do not yet include noise and foreground contaminants in this proof-of-concept study.

Key words: galaxies: high-redshift – intergalactic medium – cosmology: theory – dark ages, reionization, first stars – diffuse radiation – early Universe.

1 INTRODUCTION

The cosmic dawn (CD) of the first galaxies and subsequent reionization of the Universe remain among the most compelling yet elusive cosmological epochs. Little is currently known beyond approximately when the bulk of reionization occurred. The properties of the unseen first galaxies and intergalactic medium (IGM) structures thought to govern this cosmic milestone, remain unknown.

Fortunately, the field is set to undergo a Big Data revolution, driven by interferometric observations of the cosmic 21-cm signal. Corresponding to the spin-flip transition of neutral hydrogen, the 21-cm line is sensitive to the thermal and ionization state of the IGM, making it an ideal probe of CD and the epoch of reionization (EoR). Current interferometers, such as the Low Frequency Array (LOFAR; Haarlem et al. 2013; Yatawatta et al. 2013), the Murchison Wide Field Array (MWA; Tingay et al. 2013), and the Precision Array for Probing the Epoch of Reionization (PAPER; Parsons et al. 2010), are hoping for a statistical detection of the EoR. The upcoming Hydrogen Epoch of Reionization Array (HERA; Deboer et al. 2017) will go beyond that, capturing the fluctuations of the signal over a large range of scales and redshifts, allowing us to tightly constrain galaxy properties (e.g. Greig & Mesinger 2018). Eventually, the Square Kilometer Array (SKA; Mellema et al. 2013; Koopmans et al. 2015) will allow us to do high-signal-to-noise imaging of the EoR and CD, providing a 3D map of the first billion years of our Universe.

The timing and patterns of the signal encode the star formation histories, as well as the UV and X-ray properties of the first galaxies. The challenge is in interpreting the signal, in order to learn these properties. Early work showed general qualitative trends. For

★ E-mail: nicolas.gillet@sns.it
† Hubble fellow.

example, if the EoR is driven by rare, bright galaxies, the resulting 21-cm power would be larger than if it were driven by abundant, faint galaxies (e.g. Furlanetto, Zaldarriaga & Hernquist 2004; McQuinn et al. 2007; Iliev et al. 2012). Abundant absorbers in the IGM (so-called Lyman limit systems; LLSs) would suppress the large-scale power (e.g. McQuinn et al. 2007; Sobacchi & Mesinger 2015). Hard X-ray sources would heat the IGM more uniformly, compared to soft X-ray sources, thus decreasing the available contrast in 21-cm images of the CD (e.g. Pacucci et al. 2014; Fialkov et al. 2017).

These trends can now be quantified in detail, given the advent of efficient Monte Carlo Markov Chain (MCMC) samplers of 21-cm simulations, such as 21CMMC[1] (Greig & Mesinger 2015, 2017, 2018 hereafter referred to as GM18). For a given parametrization of astrophysics, 21CMMC computes the parameter constraints available from upcoming 21-cm observations. However, a choice must be made in which summary statistic is used in computing the likelihood (i.e. to quantify the similarity between a prediction based on a particular parameter set and the observation). A simple and popular choice of likelihood statistic is the 21-cm power spectrum (PS). Indeed the PS was shown to be a powerful discriminant of reionization and CD astrophysics (e.g. GM18).

However, the 21-cm signal is highly non-Gaussian, as various radiation fields, driven by biased sources, induce complicated correlations in the ionization and thermal state of the gas (e.g. Bharadwaj & Pandey 2005; Zahn et al. 2007; Barkana & Loeb 2008; Watkinson & Pritchard 2015; Shimabukuro & Semelin 2017; Majumdar et al. 2018). Therefore there is much additional information which is wasted if only the PS is used for parameter recovery (see for example fig. 1 in Mellema et al. 2015). Motivated by this, *here we showcase astrophysical parameter recovery directly from 21-cm images, using machine learning (ML).*

ML is powerful because it allows a model to adapt and learn complex relationships in data, without requiring the user to a priori specify functional forms. ML is becoming popular in various fields of astronomy (e.g. Kamdar, Turk & Brunner 2016a,b; Ucci et al. 2016, 2018; Schaefer et al. 2018; Gupta et al. 2018; Parks et al. 2018; Rodriguez et al. 2018). Recently, ML was also applied to the 21-cm signal, by creating an emulator to replace more expensive simulation codes (Kern et al. 2017; Schmit & Pritchard 2018) or performing astrophysical parameter recovery with the PS statistic (Shimabukuro & Semelin 2017).

In this study, we will use a Convolutional Neural Network (CNN) which is an ML technique designed to work on images. CNNs are widely used today, most famously for facial recognition and image classification. Applying a CNN directly on a 21-cm image allows the network to adaptively chose summary statistics when performing parameter inference, rather than a priori specifying the summary statistic (such as a PS). Thus, the CNN can implicitly take advantage of non-Gaussian information in the images.

This work is organized as follows. In Section 2 we present our cosmological 21-cm simulations and the resulting data base of images. Then in Section 3 we describe what is a CNN. Finally, in Section 4 we quantify the performance of the CNN in astrophysical parameter inference from 21-cm images. Unless stated otherwise, we quote all quantities in co-moving units and adopt the cosmological parameters: $(\Omega_\Lambda, \Omega_M, \Omega_b, n, \sigma_8, H_0) = (0.69, 0.31, 0.048, 0.97, 0.81, 68\,\mathrm{km\,s^{-1}\,Mpc^{-1}})$, consistent with recent results from the Planck mission (Ade et al. 2016).

[1]https://github.com/BradGreig/21CMMC

2 21-CM IMAGES

We start by briefly describing the simulations used to create the 21-cm images, before proceeding to discuss the CNN techniques. For more details about the simulation set-up, we refer the interested readers to GM18 and references therein.

2.1 Data base of simulated light-cones

We simulate our 21-cm light-cones (LC) using the public code, 21CMFAST[2] (Mesinger & Furlanetto 2007; Mesinger, Furlanetto & Cen 2011). In order to do a direct comparison to the parameter recovery using the PS with 21CMMC, we use the same version of the code and free parameters from GM18. Specifically, we vary four astrophysical parameters found to have the strongest impact on the 21-cm signal (and thus having the tightest parameter constraints):

(i) ζ : the UV ionizing efficiency of galaxies. This efficiency can be expressed as

$$\zeta = 30 \left(\frac{f_{\rm esc}}{0.1}\right) \left(\frac{f_*}{0.05}\right) \left(\frac{N_{\gamma/{\rm b}}}{4000}\right) \left(\frac{1.5}{1+n_{\rm rec}}\right) \quad (1)$$

where, $f_{\rm esc}$ is the fraction of ionizing photons escaping into the IGM, f_* is the fraction of galactic gas in stars, $N_{\gamma/{\rm b}}$ is the number of ionizing photons produced per baryon in stars, and $n_{\rm rec}$ is the typical number of times a hydrogen atom recombines. ζ primarily controls the timing of the EoR. As in GM18 we consider the range $\zeta \in [10\text{–}250]$.

(ii) $L_{\rm X}/{\rm SFR}$: the soft X-ray emissivity (below 2 keV)[3] per unit of star formation rate escaping the galaxy. X-rays are responsible for heating the neutral IGM during the CD, during the so-called Epoch of Heating (EoH). $L_{\rm X}/{\rm SFR}$ primarily impacts the timing of the EoH, with lower values delaying heating and thus increasing the strength of the absorption signal. As in GM18 we consider the range $\log_{10}(L_{\rm X<2\,keV}/{\rm SFR}) \in [38\text{–}42]$ erg s^{-1} $M_\odot^{-1}$ yr.

(iii) $T_{\rm vir}$: the minimum virial temperature of haloes capable of hosting star-forming galaxies. Star formation is suppressed for haloes below this threshold, due to feedback, and/or inefficient gas cooling. Since star formation governs all epochs of the 21-cm signal, $T_{\rm vir}$ affects the timing of the cosmic milestones, e.g. EoR, EoH, and Wouthuysen–Field (WF) coupling (Wouthuysen 1952; Field 1958), thus affecting the entire signal. Moreover, $T_{\rm vir}$ determines the bias of the typical galaxy population and the resulting radiation fields. As in GM18 we consider the range $\log_{10}(T_{\rm vir}/1{\rm K}) \in [4\text{–}6]$.

(iv) E_0 : the X-ray energy threshold for self-absorption by the galaxy. X-ray photons below this energy are absorbed by the interstellar medium (ISM) of the host galaxies. E_0 determines the hardness of the X-ray SED escaping the first galaxies. Since the absorption cross-section is a strong function of energy, E_0 governs how homogeneous and efficient is the X-ray heating. We consider the range, $E_0 \in [0.1\text{–}1.5]$ keV, equivalent to an average H I column density of $\log_{10}(N_{\rm H\,I}) \in [19.3\text{–}23.0]$ cm^{-2}.

Two additional parameters were studied in GM18: $R_{\rm MFP}$, the maximum ionizing photon horizon within ionized regions, and $\alpha_{\rm X}$, the X-ray spectral energy index. Since the authors found that these parameters have a comparably small impact on the 21-cm signal

[2]https://github.com/andreimesinger/21cmFAST

[3]Harder photons have mean free paths longer than the Hubble length at the redshifts of interest, and thus do not contribute to heating the IGM (e.g. McQuinn 2012; Das et al. 2017).

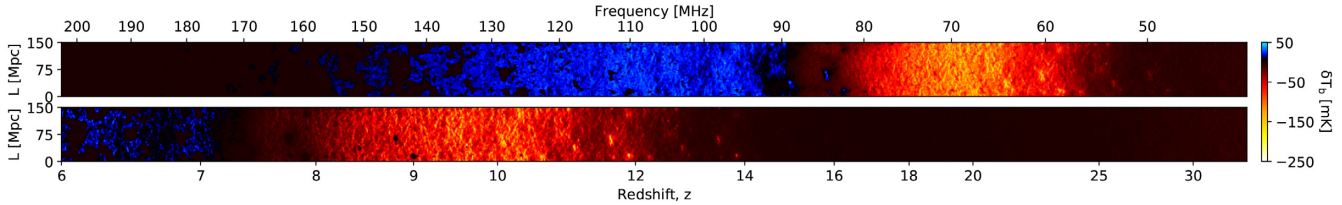

Figure 1. Example of the reduced, 2D 21-cm light-cone images used in the learning phase of the CNN. The top/bottom panel corresponds to the FAINT GALAXIES / BRIGHT GALAXIES mock observation in GM18 (see their fig. 1).

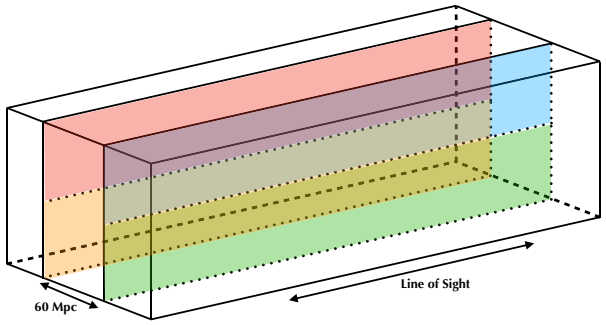

Figure 2. Scheme illustrating the slicing of LC. Two slices along the LOS are represented, separated by 60 Mpc. Each slice is again divided in two along the LOS, leading to four 'LC images' on the scheme: blue, green, red, and orange. In practice five slices are taken separated by 60 Mpc, producing 10 images per LC.

(cf. fig. 1 of Greig & Mesinger 2017), in this work we fix them to $R_{\rm MFP} = 15$ Mpc and $\alpha_{\rm X} = 1$.

We generate a data base of 10 000 21-cm LC by randomly sampling the four astrophysical parameters, uniformly over the ranges quoted above. We stress that *each LC is generated from an independent realization of the initial Gaussian random field*. This is very important for ML, as otherwise, the network can adapt to features at a specific position in the image, resulting in spuriously good results.

The transverse faces of the resulting LC[4] are 300 × 300 Mpc, while they extend from $z = 6$–30 in the parallel direction [line-of-sight (LOS) direction]. The resolution of the LC is 1.5 Mpc, corresponding to the dimensions of 200 × 200 × 2200.

2.2 Reducing the light-cone to 2D images

CNN require many iterations and tests in order to select the best architecture and tune the various free hyperparameters (see the next section for details). Performing such optimization with large data sets can be computationally expensive. For reference, our fiducial CNN described in the next section would take of order a month to train on the full data base of 3D LC, using 30 CPUs. Even perfunctory optimization of the CNN architecture would require tens of these training runs, which is not currently computationally practical.

For this reason, we instead train our CNN on 2D slices through the LC (see Fig. 1). The slicing is illustrated by the scheme in Fig. 2. Specifically, we take five slices along the LOS axis for each LC, separated by 60 Mpc, thus preserving the redshift evolution. Moreover, we split those slices in two, so that the training set images each correspond to half of the transverse volume (150 Mpc), which should preserve most of the large-scale structures (e.g. Iliev et al. 2014). The resulting 2D slices are 100 × 2200 pixels. The resulting 10 slices per LC represent a 40-fold compression from the full 3D light cuboid of 200 × 200 × 2200. Our fiducial CNN takes under 5 d to train on the resulting set of 2D images.

Training the CNN on this reduced data is a fairly conservative choice: keeping more data could result in better parameter recovery. However, for this proof of concept, which requires many iterations to tune hyperparameters, we opt for computational efficiency. In the future, we plan to extend the CNN to train on the full 3D LC, taking advantage of the speed-ups available with GPU-optimized ML toolkits.

[4]To be precise, they are actually light-*cuboids*, though we stick with *cone* as per convention.

3 CONVOLUTIONAL NEURAL NETWORK

Here we go into detail about what is a CNN, and how we use it to perform parameter inference. We start with some general discussion on ML, before focusing on our specific application of a CNN on 21-cm images.

3.1 Introduction to machine learning and neural networks

ML allows the modelling of physical processes without having to specify functional forms. It is composed of free parameters that will be adapted based on some data i.e. the relation between inputs and outputs will be learned without a priori parametrization. An example could be predicting the star formation rates of galaxies (output), based on their DM halo properties (input). This learning is an iterative process, where the model adjusts itself depending on the difference between its prediction and the 'truth'. The learning phase requires a training set for which the 'true' answer is known (e.g. the outputs of hydrodynamic galaxy formation simulations for the example above).

A neural network (NN) is a type of ML. The base block of an NN is a neuron. A neuron takes an input array x and performs a linear combination with an array of trainable weights w plus a bias factor b to compute the net-input s. In other words, a given neuron j produces the following output s^j from an input vector x_i:

$$s^j = \sum_{i=\rm input} w_i^j x_i + b^j . \quad (2)$$

Neurons are organized in layers. All neurons in a given layer see the same input vector x, but each neuron has its own independent weights vector w^j. The resulting s^j vector outputted by a given layer is then used to make the input for the successive layer (which could be comprised by a different number of neurons). Several layers of neurons then collectively form the NN.

The linear combinations of each layer can be used to reproduce arbitrary, non-linear functions, using the so-called activation

function $\phi(s)$. It is applied to the output of each neuron, in order to quantify if it is significant or not. In essence, the activation function serves as an 'on/off' switch, allowing the NN to ignore the outputs of irrelevant neurons. Typical activation functions are sigmoid, hyperbolic-tangent, or Rectified Linear Units (ReLU). The choice of the activation function is a free hyperparameter of the network. Here we choose the ReLU activation function (Nair & Hinton 2010): $\phi(\mathrm{s}) = \mathrm{s}$ if $s > 0$, otherwise 0. The neurons of the final layer which produces the desired output do not have activation functions.

NNs are very efficient when applied to relatively small input arrays. However, since the number of weights which need to be learned depends on the size of the input, learning can become very slow for large or high-dimensional data. More specific types of NNs have been developed to account for this.

3.2 General structure of CNNs

CNNs have become popular in recent years as they provide a computationally cheaper alternative to classical NNs. They are commonly used in image recognition (e.g. LeCun et al. 1999; Krizhevsky, Sutskever & Hinton 2012), and have recently started being used in cosmology, for example, to find or analyse strong gravitational lenses in images (e.g. Hezaveh, Levasseur & Marshall 2017; Schaefer et al. 2018)

The first part of the CNN is the feature extractor (cf. Fig. 3). Its purpose is to reduce the image, extracting patterns.[5] The feature extraction is composed of successive convolution and pooling layers (cf. Fig. 3).

A convolution layer takes an image as input and returns a series of concatenated convolutions analogous to a classical RGB (red–green–blue) image where the different convolutions represent the different colours. The next layer performs a series of convolutions on the output of the previous one. Those convolutions are in their turn concatenate in order to feed the next layer, etc.

Each convolution (also called a channel) is performed by an independent filter and is fed to an array of neurons (one per image pixel) each with their own activation function. The number of filters, N_f, and their size, S_k, are the two hyperparameters of the layer. Each 2D filter is composed of S_k^2 trainable weights: these weights are adapted to the learning data.

A convolutional network is built by chaining convolution layers, each one working on the output of the previous one, after passing through the activation function, exactly as for a classical NN. The size of filters in one layer is fixed, therefore, one convolution layer can probe only one typical scale. In order to access multiscale information, the images are downsampled by a pooling layer. A pooling layer shrinks the image by keeping the maximal value in a kernel size S_kMAX, which is another hyperparameter. The function used in the pooling (here maximum) is also a hyperparameter.

The sequence of convolutional and pooling layers defines the feature extraction part of the CNN. The resulting images are then flattened into a 1D array, by concatenating the rows from the final pooling layer. This 1D image then serves as an input into a 'classical' NN, used for parameter inference. This NN operates in so-called regression mode (i.e. used to predict continuous values of parameters, as opposed to categorizing). Each layer of this NN is fully connected, with the final layer consisting of the four neurons corresponding to our astrophysical parameters. These four neurons are the only ones without activation functions.

[5]This is a form of data compression, analogous to the standard 21-cm analysis of performing an Fast Fourrier transforma and taking the PS, with the important difference that one does not a priori specify the convolution kernel but instead allows the network to learn it.

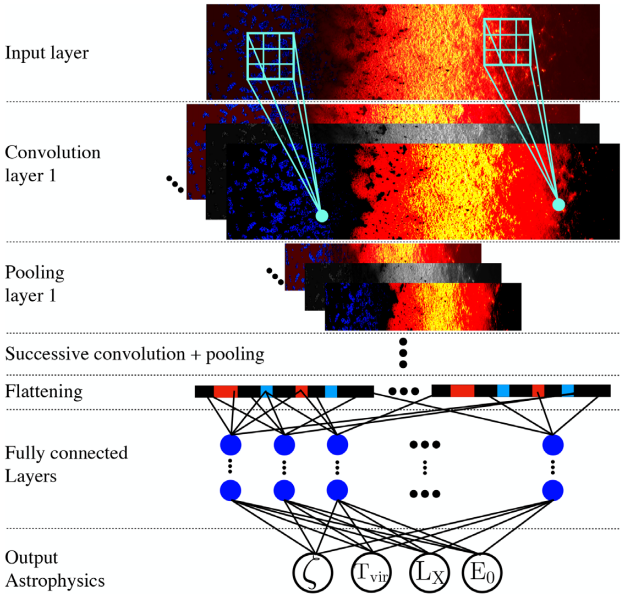

Figure 3. A schematic of a CNN. The first part of a CNN takes an image and performs a series of convolutions with square filters (here shown with a 3×3 matrix), with each convolution having its own adjustable weights. The results from these filters are fed to neurons (one per pixel; here represented by blue circles) which have their own activation function. This data is further reduced by pooling (downsampling) and then again performing convolution and pooling using as input the output of the preceding layer. The second part consists of a classical NN operating on a flattened (1D) image resulting from the convolution and pooling layers. At the flattening stage, the colours illustrate the propagated information from the EoH (red) and EoR (blue). The outputs of the network are astrophysical parameters, shown at the bottom. Note that the colours and values used here are purely illustrative; examples of actual inputs/outputs of our convolution layers can be found in the appendix. For clarity, only one convolution and pooling layer are shown, and the filters of the convolution layer are illustrated by a 3×3 matrix, instead of the 10×10 matrix used in our CNN.

3.3 Training a neural network

The training (or learning) of a network is the adaptation of all the weights as a function of how good the response of the network is for a given input. The algorithm we describe here is general for NNs (CNNs do not require specific algorithms for training).

Training an NN is an iterative process of guessing, computing the corresponding error, and updating the network until the error becomes arbitrarily small. The guessing part is called 'forward propagation': running the network on the training sample. Then the error can be computed between the predicted (y_pred) and the true values (y_true) of the parameters. This error is quantified by a loss function. Finally, the updating of the weights is done by backpropagating in the network the loss information (the loss information goes from the parameters end to the input layer). Each iteration over the whole training sample is called an epoch, and the process of training to update the weights requires several epochs. In our study, the network needs ∼90 epochs to converge (see Fig. 4 and associated discussion).

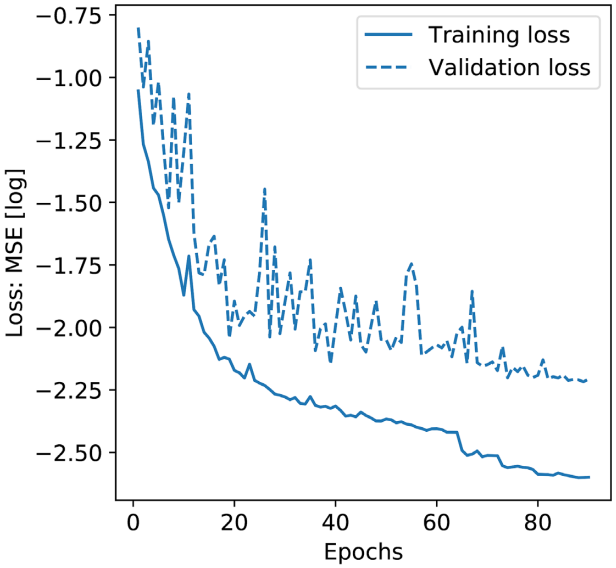

Figure 4. Loss of the CNN at each epoch during the training phase. The training loss (continuous line) is directly computed from the training sample. The validation loss (dashed line) is estimated at the end of each epoch with the validation sample (which is independent of the training and testing samples).

There are many loss functions to choose from in the literature. Here we use the mean square error defined by: $\mathrm{MSE} = \sum(y_{\mathrm{pred}} - y_{\mathrm{true}})^2/N_{\mathrm{y}}$ where N_{y} is the data sample size and the summation occurs on the whole sample. During training, the NN tries to minimize this loss by changing the weights. The weights are updated by gradient descent: computing the derivative of the loss with respect to the weights, and following the gradient towards the minimum.

The updating of the weights is controlled by the 'learning rate'. It controls how much the weights step from their current positions towards the minimum. If the learning rate is too small, the convergence will be slow. If it is too large, the learning might never converge, as the weights keep stepping over the minimum. Algorithms have been developed to optimize this process. Here we use the RMSprop optimizer (Hinton, Srivastava & Swersky 2012) in combination with a Batch Normalization layer (Ioffe & Szegedy 2015). Additionally, we use an adaptive learning rate i.e. when the training loss does not evolve for a defined number of epochs (called 'patience'), the learning rate is decreased by a certain factor. The patience and factor are additional hyperparameters.

The learning phase can be further optimized by splitting the training sample into groups, called 'mini-batches'. This allows the network to update its weights after only computing the loss from a mini-batch of training data (e.g. Bengio 2012; Masters & Luschi 2018). The size of this mini-batch, N_{batch} is another hyperparameter.

3.4 Tuning hyperparameters and avoiding overfitting

As shown in the previous sections, the building and training of a CNN include choosing a number of hyperparameters. For some of these, there can be physically motivated choices. For example, to reduce the required number of pooling layers, one can choose a convolutional filter size which is large enough to pick up the characteristic scales of the ionized and heated regions of the 21-cm signal.

However, most of the choices made in setting the CNN do not have obvious a priori motivations, and have to be 'tuned' (e.g. Bengio 2012). Tuning an NN is an iterative process, as for each new hyperparameter the network can be re-trained and its recovery ability re-evaluated. As a result, tuning can be extremely computationally expensive.

In addition to the computational cost, one must be careful to avoid 'overfitting'. In choosing hyperparameters which optimize performance on the training set, the network can become very specialized to the training data. If this overfitting happens, the network will not be able to generalize the learned features of data outside of the training set.

A common solution to avoid overfitting is to set aside some fraction of the training data, and not use it during the learning phase when evaluating the hyperparameters. This is called the 'validation set' and is independent of the 'testing set', which is set aside and only used at the end in order to quantify the final performance of the CNN (see Section 4). In our case, the validation set consists of 1000 randomly selected LC. The performance of the CNN on the validation set serves as a figure of merit for each chosen set of hyperparameters.

In addition to using a validation sample to avoid overfitting, we also turn-off a random selection of 20 per cent of the neurons during training. This technique is called 'dropout' (Srivastava et al. 2014), and it prevents the NN from relying too much on a fixed subset of information, preferring weights which include information from more neurons (so-called 'regularization').

We demonstrate that our CNN does not overfit the training set in Fig. 4. This figure presents the loss during learning, as a function of learning epoch, for our final CNN set-up discussed in the following section. The solid line corresponds to the 'training loss', i.e. computed from the training sample and used in the backpropagation step of the learning. The dashed line corresponds to the 'validation loss', computed from the validation sample, but not used in the learning. As the network adapts, the losses decrease and at some point start to asymptote. In our case, the validation loss asymptotes after ∼80 epochs. If the network were overfitting, the training loss would continue to decrease, while the validation loss would start to *increase*: the network would start 'overspecializing' on the training set. This does not happen for our CNN; instead, the two losses show the same general trends.

3.5 The final CNN set-up

The final set-up of the CNN used in this study is presented in Table 1. The CNN is composed of two layers of convolution plus pooling. The first convolution layer contains eight filters and the second contains 16, with both using 10 × 10 filters. Each pooling layer downsamples by a factor of 2. The fully connected part is composed of four layers with 64, 16, 8, and 4 neurons, respectively. Our dropout fraction is 20 per cent, applied on the layer that contains most of the weights: the one between the flattening layer and the first fully connected layer. Our training/validation/test sets consists

Table 1. Architecture and hyperparameters of the CNN used in this study.

(1) Order	(2) Layer/step name	(3) Data dimension	(4) Number of weights
1	Input	100×2200	.
2	2D Convolution	$8 \times 91 \times 2191$	808
3	Max Pooling	$8 \times 45 \times 1095$	0
4	2D Convolution	$16 \times 36 \times 1086$	12 816
5	Max Pooling	$16 \times 18 \times 543$	0
6	Flattening	156 384	0
7	Dropout	(20 per cent – only in learning phase)	0
8	Fully connected	64	10 008 640
9	Batch Normalization	.	.
10	Fully connected	16	1040
11	Fully connected	8	136
12	Fully connected Out	4	36
Total number of unknowns	.	10 023 604	
Size convolution filter	(10×10)		
Size MaxPool filter	(2×2)		
Activation function	ReLU		
Optimizer	RMSprop		
Loss	MSE		
Number of epochs	100		
Batch size	200		
Training set	8000×10		
Validation set	1000		
Testing set	1000		
Learning rate patience	5 epochs		
Learning rate factor	0.5		

of (8000×10)[6] / 1000 / 1000 LC slices. The CNN has been built using the PYTHON API for neural networks KERAS[7] (Chollet 2015).

4 RESULTS: PARAMETER INFERENCE WITH A CNN

Using the 1000 test LCs, we now quantify the performance of the CNN described in the previous section.

4.1 Coefficient of determination

We first evaluate the coefficient of determination for each of the four astrophysical parameters, R^2, defined by

$$R^2 = \frac{\sum(y_{\rm pred} - \overline{y}_{\rm true})^2}{\sum(y_{\rm true} - \overline{y}_{\rm true})^2} = 1 - \frac{\sum(y_{\rm pred} - y_{\rm true})^2}{\sum(y_{\rm true} - \overline{y}_{\rm true})^2}, \quad (3)$$

where $\overline{y}_{\rm true}$ is the average of the true parameter and the summation is performed over the entire test set. The R^2 range can be between 0 and 1, where 1 indicates a perfect inference of the parameters.

The resulting values of R^2 are presented in Table 2 column (2). The parameters $T_{\rm vir}$ and $L_{\rm X}$/SFR are almost perfectly inferred by the network with $R^2 = 0.99$. The parameter ζ results in a slightly lower score, $R^2 = 0.95$, which is still good. Finally, the last parameter E_0 has a comparably low score, $R^2 = 0.73$. We shall explore the reason for this in the following section.

Table 2. Coefficient of determination (R_2) for our four astrophysical parameters, using the testing sample. Columns (2) and (3) present results for two different training sets, using 10 and 1 slices per LC, respectively.

(1) Parameter	(2) R_2 score (10 images / LC)	(3) R_2 score (1 image / LC)
$T_{\rm vir}$	0.997	0.984
$L_{\rm X}$/SFR	0.987	0.981
ζ	0.955	0.851
E_0	0.728	0.531

We additionally present in Table 2 column (3) the scores resulting after learning is performed on a training set which is reduced in size by a factor of 10: consisting of only one slice per LC. As expected, a smaller training set worsens the recovery, especially for the parameters ζ and E_0 . This indicates that we could expect even better performance if we were to use the entire LC in the training (corresponding to 400 of these 2D images), instead of throwing away 39/40 of the data as we are doing in our fiducial CNN. We plan on investigating this further in follow-up work, taking advantage of GPU-accelerated ML packages.

4.2 Predicted versus true distributions

The R^2 score from the previous section gives an estimate of the CNN's performance averaged over the whole parameter range. Here we go into more detail by presenting the distributions of predicted versus true parameter values in Fig. 5. This figure is the main result

[6] As mentioned previously, the training set contains 8000 LC and we take 10 image slices per LC, resulting in 8000×10 images using in the training. For validation and testing we only take one slice per LC as these are only used to check the performance of the inference and the size of the sets is already sufficient to obtain good statistics.

[7] https://github.com/keras-team/keras

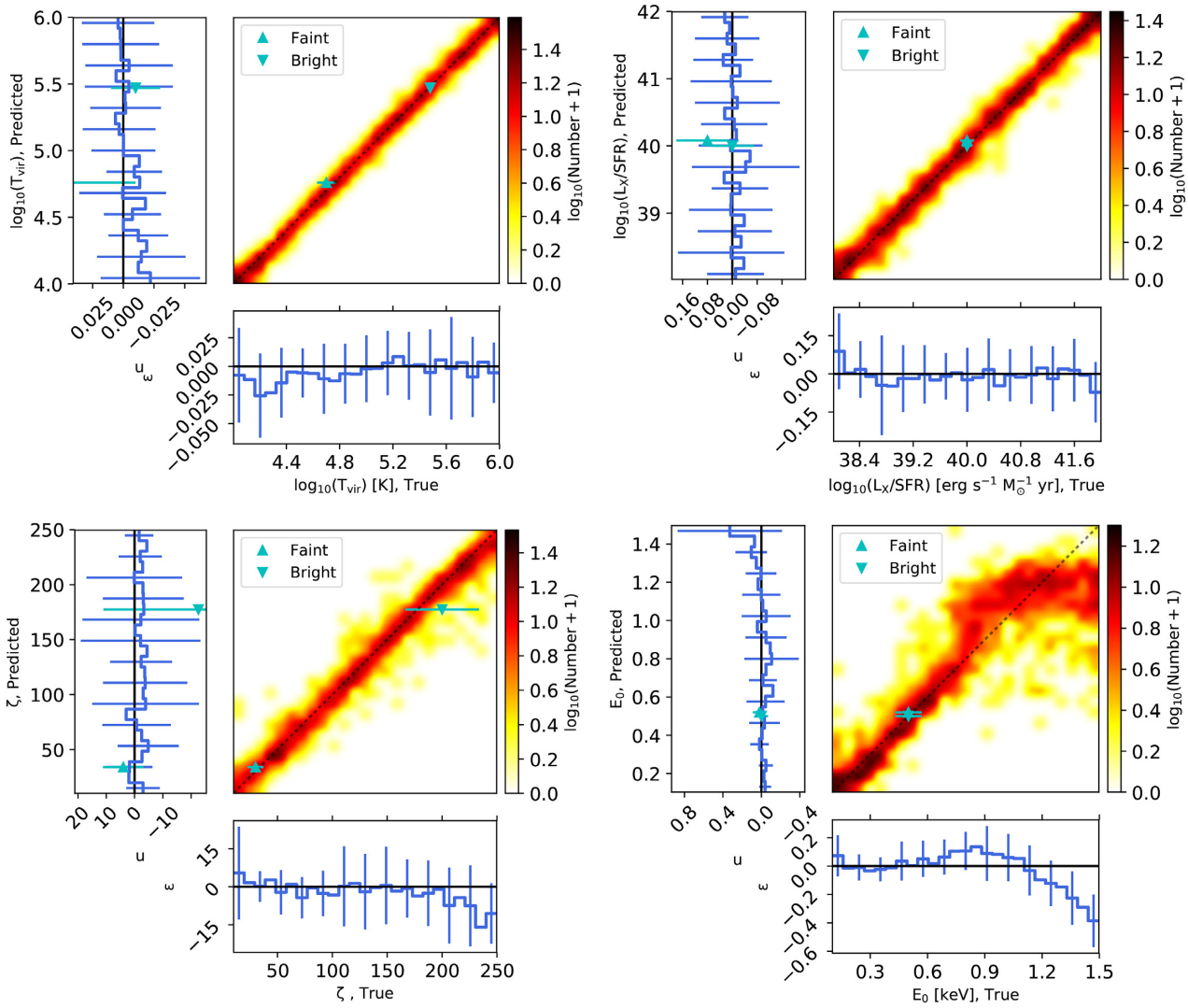

Figure 5. CNN parameter inference using the testing sample which contains 1000 LC. The central image in each panel contains the distributions of true versus predicted parameter values, $p(y_{\rm pred}, y_{\rm true})$, with the side colour bar indicating the log number count in each bin. The closer the points are to the diagonal, the better is the performance. The bottom and left histograms present the mean and the standard deviation (taking a Gaussian fit) of the residual, $y_{\rm pred} - y_{\rm true}$. The bottom histogram is the learning error, $p(y_{\rm pred} - y_{\rm true}|y_{\rm true})$, while the side histogram is the recovery uncertainty, $p(y_{\rm pred} - y_{\rm true}|y_{\rm pred})$. The two triangle points mark the position and the marginalized error for MCMC analysis of GM18 using 21CMMC.

of this work.

For each parameter, the central panel shows the density of points (see adjacent colour bar) in the corresponding 2D space. The bottom and left side histograms present the mean and standard deviation of the residual $r \equiv y_{\rm pred} - y_{\rm true}$, along the vertical and horizontal directions, respectively. Specifically, the bottom histogram shows the so-called network error, ε, i.e. the distribution of residuals as a function of the true values, $p(r|y_{\rm true})$. The left histogram shows the prediction uncertainty, u, i.e. the distribution of residuals as a function of the prediction, $p(r|y_{\rm pred})$. In practice, the left-hand panel is the most relevant for quantifying the performance, since we will eventually have a single 21-cm LC observation of our single Universe with the 'true' values unknown to us. Thus we are interested in what is the allowed range of true values, given the prediction we will have from the CNN.

We also denote with cyan points and error bars the recovered values and marginalized error from the MCMC analysis in GM18. Those authors performed recovery using two mock observations: 'Faint' and 'Bright' galaxies, with correspondingly different values of ζ and $T_{\rm vir}$. We caution that it is difficult to directly compare against these results since: (i) the marginalized uncertainty from GM18 does not directly translate to our recovery uncertainty; and (ii), unlike GM18, we do not include instrument noise. As such, quantitative comparison of the uncertainties only makes sense if they are not noise dominated, but are intrinsic to the theoretical model and parameter degeneracies. As we shall see below, the uncertainties in both studies follow similar qualitative trends. As this work is only a proof of concept, we defer more detailed comparisons, including various telescope noise and point spread function models for future work.

For the parameters $T_{\rm vir}$ and $L_{\rm X}$/SFR (top row), the distributions are tightly centred along the diagonal over the entire range. There are no obvious biases and the prediction uncertainties shown in the left-hand side panels are $\sigma_{\rm T_{vir}} = 0.028$ and $\sigma_{\rm L_X} = 0.11$. These uncertainties are roughly constant over the entire parameter range

and are comparable to those from GM18 for the two 'true' values used in that work. [8]

The recovery of ζ (bottom left-hand panel), is in general good though worse than the previous two parameters. Unlike for the previous two parameters, the recovery uncertainty is not constant over the whole parameter range. Consistent with the MCMC analysis of GM18, high values of ζ (their Bright Galaxies model) have larger uncertainties than low values (their Faint Galaxies model). An explanation of this could be the following. ζ controls the timing of reionization. Reionization driven by rare, bright galaxies must be rapid, requiring very high ionizing efficiencies, ζ , in order to match the constraints from *Planck* and high-z QSOs. Having increasingly rapid reionization compresses the corresponding 21-cm signal into an increasingly narrow redshift range, decreasing and saturating how much information is available for inference. As for the previous parameters, the size of the errors from the CNN is comparable to the MCMC analysis which uses the PS.

Finally, in the bottom right-hand panel, we quantify the CNN's performance in recovering E_0 . As expected from its comparably low coefficient of determination, $R^2 \sim 0.7$, the recovery is modest. But despite this low score, we can see that the parameter is relatively well learned on the lower half of the parameter range ($E_0 \in [0.1, 0.8]$ keV), while on the higher part it seems to saturate. Note, that beyond 0.8 keV we no longer trust our reported errors because of the saturation. The conclusion here is that above $\sim$0.8 keV the results are not trustworthy. This saturation is understandable. High values of E_0 imply a harder X-ray spectrum. Because the ionization cross-sections of hydrogen and helium are very strong functions of energy, $\sigma \propto E^{-3}$, harder X-rays interact increasingly less with the IGM and thus leave an increasingly smaller imprint in the LC. Indeed beyond energies of $\sim$1.5 keV, the mean free path of X-rays surpasses the Hubble length and so we should not be able to distinguish any model with $E_0 \geq 1.5$ keV. It would be interesting to perform a parameter recovery using 21CMMC with a mock observation having $E_0 \sim 1.2$ keV to confirm this trend.

5 DISCUSSION AND CONCLUSIONS

The advent of next-generation 21-cm interferometers, like HERA and SKA, will allow us to create 3D images of the first billion years of our Universe. We need sophisticated analysis tools in order to interpret these observations and understand the underlying astrophysics. Recent effort has been placed on astrophysical parameter recovery, using Bayesian approaches like MCMC. These require assuming a summary statistic when computing the likelihood, with the power spectrum being an obvious choice. Although the PS was shown to be a powerful discriminant of reionization and CD astrophysical parameters, the 21-cm tomographic signal is highly non-Gaussian (unlike the cosmic microwave background). Therefore there is additional information which is wasted if only the PS is used for parameter recovery.

In this proof-of-concept paper, we demonstrate that astrophysical parameters can be recovered directly from 21-cm images using deep learning with CNN. CNNs are able to adapt to the training data, without requiring the user to a priori specify a summary statistic. However this flexibility comes at a cost of physical intuition (see the Appendix for more details), and also makes having a fully Bayesian interpretation more difficult. This statement comes from the fact that the CNN in inference mode (regression) only returns the best guess (analogous to a maximum likelihood), and not the probability of the result. The inferred marginalized errors of an observation can be estimated from Fig. 5, effectively marginalizing over a sample of 'best guesses' but this is not a posterior (as for a classical MCMC). The posterior information is encoded in all neurons' weights, but quantifying this is non-intuitive. Levasseur, Hezaveh & Wechsler (2017) provide a framework to estimate the final error made on parameter inference, taking into account the noise from the data and the error of the network itself. It requires different kind of neural networks than the CNN used in this study, but it should be used for future networks that will be deployed on the observations.

We train our CNN using a data base of 10 000 LC, of which 2000 are reserved for validation and testing. We demonstrate that our tuned network does not overfit the training set. Using 10 2D image slices per LC for training, we show that the CNN is able to recover popular parameters describing the first galaxies: (i) $T_{\rm vir}$, the minimum host halo mass capable of hosting efficient star formation; (ii) ζ , their typical ionizing efficiencies; (iii) $L_{\rm X}$/SFR their typical soft-band X-ray luminosity to star formation rate; and (iv) E_0 , the minimum X-ray energy capable of escaping into the IGM (governed by their mean HI column densities).

For most of their allowed ranges, parameters $T_{\rm vir}$ and $L_{\rm X}$/SFR are recovered with < 1 per cent uncertainty, while ζ and E_0 are recovered with $\sim$10 per cent uncertainty. Our results are roughly comparable to the accuracy obtained from MCMC sampling of the PS with 21CMMC for the two mock observations analysed previously. Although we caution that we do not yet include noise in this proof-of-concept study.

Moreover, the computational cost of parameter inference using a CNN is distributed differently than using an MCMC code. Training the network requires (i) generating a large data base of simulated LCs, and (ii) tuning hyperparameters. In the case of this study, each of these steps took a couple of weeks of computation time on a dedicated server. However, once the data base is in place and the network tuned, performing parameter inference on an 'observation' is extremely fast, requiring only a few seconds per LC. In contrast, 21CMMC requires no tuning; however, parameter inference for each new 'observation' requires a comparable number ($\sim$10 000) of LC samples. This 'up front' computational cost makes NN useful when one wishes to study parameter inference for a large number of mock observations before we have an actual observation in-hand. Indeed, with our CNN test sample we demonstrated that the recovered accuracy of ζ and especially E_0 depend strongly on the 'true' parameters. In particular, we show that the recovery of E_0 starts to worsen at energies $\gtrsim 0.8$ keV, as such hard X-ray photons do not interact strongly with the IGM.

It is difficult to a priori estimate how much these idealized conditions affect the results of our parameter recovery. However, we note that we only used a small fraction of the current information content of the LCs, due to the high computational costs associated with training and optimizing the CNN. This reserve of data could offset the effective signal loss from noise and systematics. Indeed, in a paper which appeared subsequent to the completion of this work, La Plante & Ntampaka (2018) showed that an empirical parameter like the duration of reionization, can be recovered to 10 per cent precision in the presence of noise from 3D LC data. In

[8] One of the benefits of using ML for parameter inference is that we can immediately evaluate the network's performance over the entire parameter range (cf. the left-hand side panels), without having to perform a separate MCMC for each mock observation. Obviously, this is only relevant before we have an actual observation, which will correspond to a single Universe.

future work, we will investigate astrophysical parameters recovery in the presence of realistic noise and systematics.

ACKNOWLEDGEMENTS

We thank B. Semelin for its useful comments and discussions. This work was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 638809 – AIDA – PI: Mesinger). The results presented here reflect the authors' views; the ERC is not responsible for their use. Parts of this research were supported by the Australian Research Council Centre of Excellence for All Sky Astrophysics in 3 Dimensions (ASTRO 3D), through project number CE170100013. AL acknowledges support for this work by NASA through Hubble Fellowship grant #*HST*-HF2-51363.001-A awarded by the Space Telescope Science Institute, which is operated by the Association of Universities for Research in Astronomy, Inc., for NASA, under contract NAS5-26555. We acknowledge support from INAF under PRIN SKA/CTA FORECaST. We thank contributors to SCIPY,[9] MATPLOTLIB,[10] PYDOE,[11] and the PYTHON programming language.[12]

REFERENCES

Ade P. A. R. et al., 2016, A&A, 594, A13
Barkana R., Loeb A., 2008, MNRAS, 384, 1069
Bengio Y., 2012, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 7700 LECTU, 437, preprint (ArXiv:1206.5533)
Bharadwaj S., Pandey S. K., 2005, MNRAS, 358, 968
Chollet F., 2015, Keras
Das A., Mesinger A., Pallottini A., Ferrara A., Wise J. H., 2017, MNRAS, 469, 1166
DeBoer D. R. et al., 2017, PASP, 129, 45001
Fialkov A., Cohen A., Barkana R., Silk J., 2017, MNRAS, 464, 3498
Field G., 1958, Proc. IRE, 46, 240
Furlanetto S. R., Zaldarriaga M., Hernquist L., 2004, ApJ, 613, 16
Greig B., Mesinger A., 2015, MNRAS, 449, 4246
Greig B., Mesinger A., 2017, MNRAS, 472, 2651
Greig B., Mesinger A., 2018, MNRAS, 477, 3217
Gupta A., Matilla J. M. Z., Hsu D., Haiman Z., 2018, Phys. Rev. D, 97, 103515
Haarlem M. P. V., Wise M. W., Gunst A. W., Heald G., Mckean J. P., Hessels J. W. T., Bruyn A. G. D., 2013, A&A, 556, A2
Hezaveh Y. D., Levasseur L. P., Marshall P. J., 2017, Nature, 548, 555
Hinton G. E., Srivastava N., Swersky K., 2012, COURSERA: Neural Networks for Machine Learning, p. 31
Iliev I. T., Mellema G., Shapiro P. R., Pen U.-L., Mao Y., Koda J., Ahn K., 2012, MNRAS, 423, 2222
Iliev I. T., Mellema G., Ahn K., Shapiro P. R., Mao Y., Pen U.-L., 2014, MNRAS, 439, 725
Ioffe S., Szegedy C., 2015, preprint (arXiv:1502.03167)
Kamdar H. M., Turk M. J., Brunner R. J., 2016a, MNRAS, 455, 642
Kamdar H. M., Turk M. J., Brunner R. J., 2016b, MNRAS, 457, 1162
Kern N. S., Liu A., Parsons A. R., Mesinger A., Greig B., 2017, ApJ, 848, 23
Koopmans L. et al., 2015, Proceedings of Advancing Astrophysics with the Square Kilometre Array (AASKA14). p. 1
Krizhevsky A., Sutskever I., Hinton G. E., 2012, Advances In Neural Information Processing Systems. p. 1
La Plante P., Ntampaka M., 2018, preprint (arXiv:1810.08211)
LeCun Y., Haffner P., Bottou L., Bengio Y., 1999, Shape, Contour and Grouping in Computer Vision. Springer, Berlin, Heidelberg, p. 319
Levasseur L. P., Hezaveh Y. D., Wechsler R. H., 2017, ApJ, 850
Majumdar S., Pritchard J. R., Mondal R., Watkinson C. A., Bharadwaj S., Mellema G., 2018, MNRAS, 476, 4007
Masters D., Luschi C., 2018, preprint (arXiv:1804.07612)
McQuinn M., 2012, MNRAS, 426, 1349
McQuinn M., Lidz A., Zahn O., Dutta S., Hernquist L., Zaldarriaga M., 2007, MNRAS, 377, 1043
Mellema G. et al., 2013, Exp. Astron., 36, 235

[9] http://www.scipy.org
[10] http://www.matplotlib.sourceforge.net
[11] https://pythonhosted.org/pyDOE/
[12] http://www.python.org

Mellema G., Koopmans L., Shukla H., Datta K. K., Mesinger A., Majumdar S., Group o. b. o. t. C. S. W., 2015, Advancing Astrophysics with the Square Kilometre Array (AASKA14), preprint (ArXiv:1501.04203)
Mesinger A., Furlanetto S., 2007, ApJ, 669, 663
Mesinger A., Furlanetto S., Cen R., 2011, MNRAS, 411, 955
Nair V., Hinton G. E., 2010, Proceedings of the 27th International Conference on Machine Learning. p. 807
Pacucci F., Mesinger A., Mineo S., Ferrara A., 2014, MNRAS, 443, 678
Parks D., Prochaska J. X., Dong S., Cai Z., 2018, MNRAS, 476, 1151
Parsons A. R. et al., 2010, AJ, 139, 1468
Rodriguez A. C., Kacprzak T., Lucchi A., Amara A., Sgier R., Fluri J., Hofmann T., Réfrégier A., 2018, preprint (arXiv:1801.09070)
Schaefer C., Geiger M., Kuntzer T., Kneib J.-P., 2018, A&A, 611, A2
Schmit C. J., Pritchard J. R., 2018, MNRAS, 475, 1213
Shimabukuro H., Semelin B., 2017, MNRAS, 468, 3869
Sobacchi E., Mesinger A., 2015, MNRAS, 453, 1843
Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R., 2014, J. Mach. Learn. Res., 15, 1929
Tingay S. J. et al., 2013, PASA, 30, e007
Ucci G., Ferrara A., Gallerani S., Pallottini A., 2016, MNRAS, 465, 1144
Ucci G., Ferrara A., Pallottini A., Gallerani S., 2018, MNRAS, 477, 1484
Watkinson C. A., Pritchard J. R., 2015, MNRAS, 454, 1416
Wouthuysen S. A., 1952, AJ, 57, 31
Yatawatta S. et al., 2013, A&A, 550, A136
Zahn O., Lidz A., McQuinn M., Dutta S., Hernquist L., Zaldarriaga M., Furlanetto S. R., 2007, ApJ, 654, 12

APPENDIX A: WHAT THE CNN SEES

Oftentimes NNs are used as a 'black boxes' because the interpretation of the internal weights is difficult. In the case of CNNs, the weights of the convolution layers correspond to filters. In order to peek inside the black box somewhat, in Fig. A1 we show the eight filters of the first convolutional layer after the learning is completed. The weights are obviously not random anymore, and some shapes could be interpreted as extracting bubbles and web-like structures. Other filters are difficult to interpret with human eyes.

Another way to illustrate the CNN behaviour is to look at the output of the convolution layers, after passing through the filters from the previous figure. In Fig. A2 we present two examples of the output of the first convolution layer, chosen to highlight different astrophysics and the resulting different network behaviours. For both, the top panel corresponds to the inputted LC (the colour bar is the same as in Fig. 1). The eight following lines are the responses of the input by the eight filters as shown in Fig. A1 (in left-to-right, top-to-bottom order). The output contains the activation of the layer (the colour is arbitrary).

We can see that some of the channels (i.e. images produce by individual convolutions) seem to pick up reionization bubbles; others are more sensitive to heating structures, while others seem to also pick up density structures. In the bottom example we can see that three channels are completely black, i.e. they do not include any information. But those same channels in the top example contain information, picking up pre-reionization fluctuations. This illustrates the fact that information takes a different path in the network, depending on the input values.

The interpretation of the first convolutional layer is possible because it is directly linked to the input data. The interpretation of deeper layers become more challenging, as they involve increasing non-linearity, and include convolutions of convolved images. Once the information reaches the fully connected layers, it is completely obscure to human interpretation.

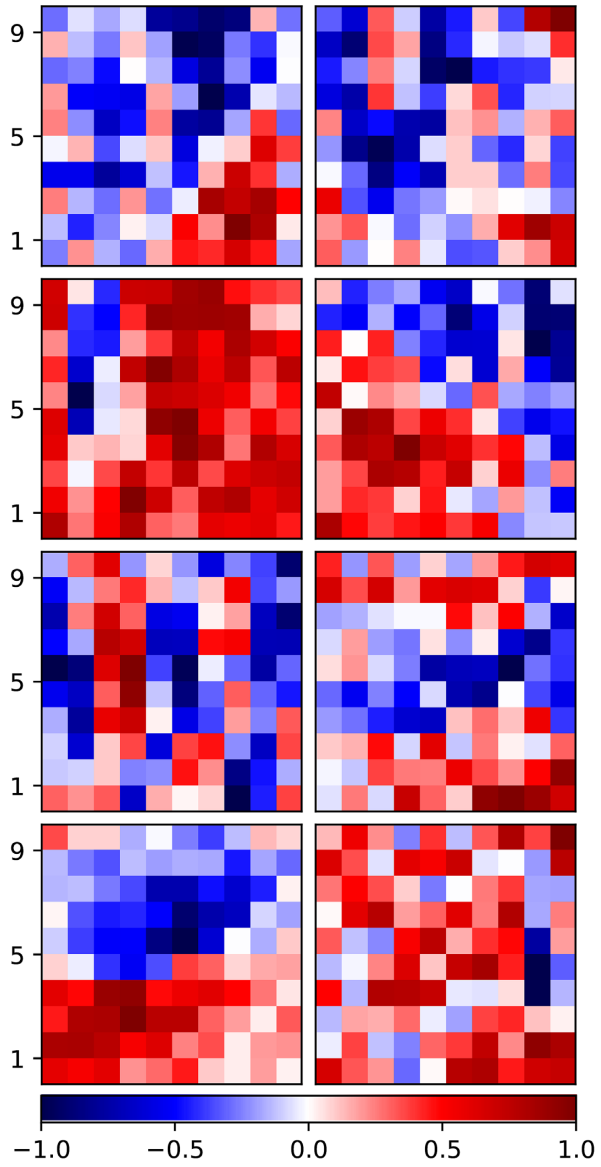

Figure A1. Filters of the first convolutional layer after learning is completed (the colour corresponds to a unitless weight). It is difficult to find obvious physical counterparts, though some could be interpreted as extractors of bubbles and/or connected web-like structures. Comparing to the convolution results shown in the following figure, the characteristic shapes and scales of these filters allow certain channels to specialize in particular cosmic epochs for some regions of parameter space.

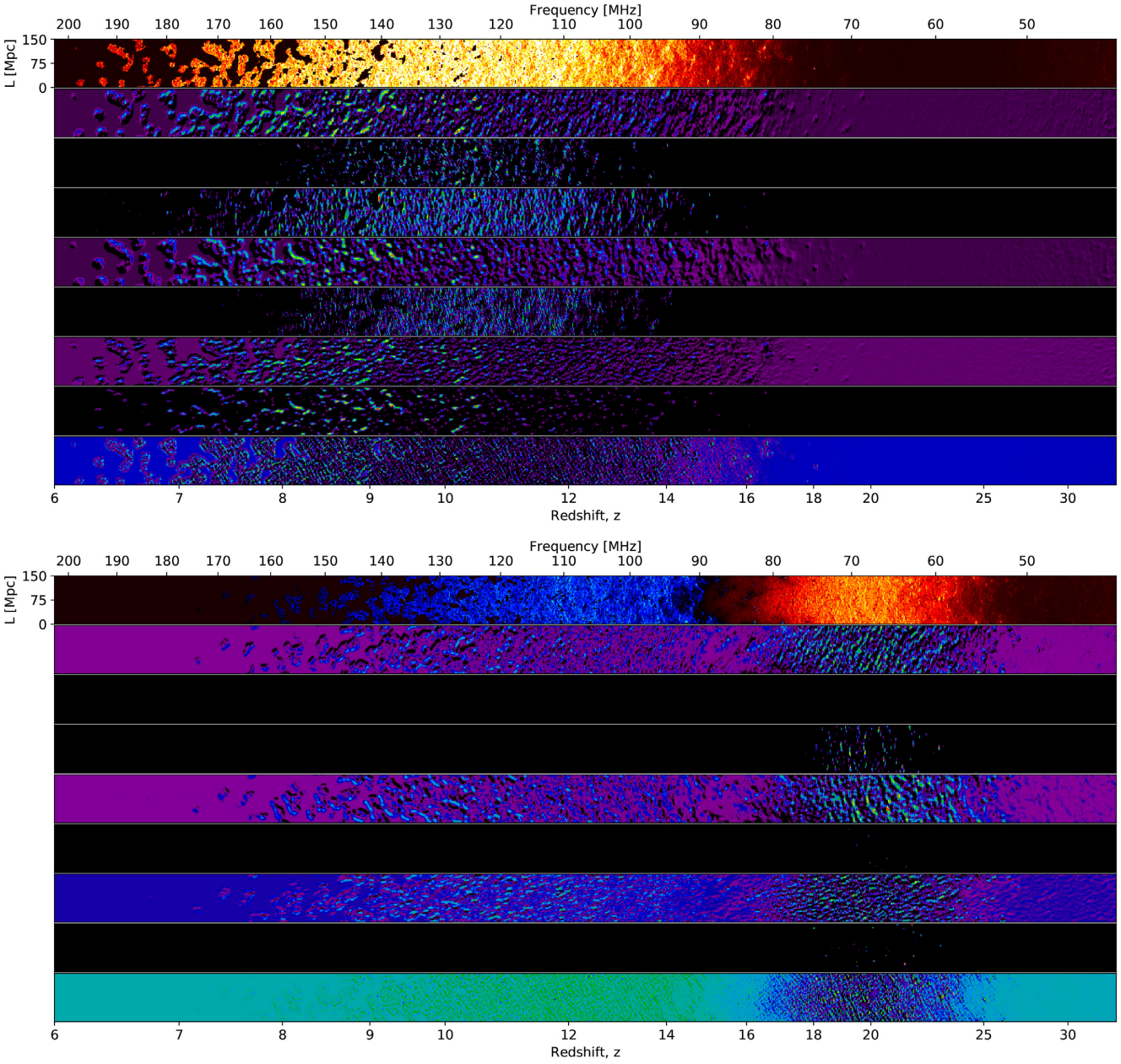

Figure A2. Response of the first convolutional layer to two different LC inputs, chosen to highlight different feature extraction, with [ζ , $T_{\rm vir}$, $L_{\rm X}$/SFR , E_0]=[106, 5.2, 38.7, 0.89] and [30, 4.15, 40.5, 0.65], respectively. The first line of each panel is the inputed LC slice (with the same colour bar as Fig. 1). The following eight lines are the response by each filters with the activation function (the first line of Fig. A1 correspond to the two first convolutions and so on).

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