A deep learning model to emulate simulations of cosmic reionization

Jonathan Chardin, Grégoire Uhlrich, Laurence Voutquenne-Nazabadioko, Nicolas Deparis, Nicolas Gillet, Pierre Ocvirk, Joseph Lewis

Monthly Notices
of the
ROYAL ASTRONOMICAL SOCIETY

MNRAS **490,** 1055–1065 (2019)
Advance Access publication 2019 September 7

doi:10.1093/mnras/stz2605

A deep learning model to emulate simulations of cosmic reionization

Jonathan Chardin,[1]★ Grégoire Uhlrich,[1,2] Dominique Aubert,[1] Nicolas Deparis,[1] Nicolas Gillet,[3] Pierre Ocvirk[1] and Joseph Lewis[1]

[1]*Observatoire Astronomique de Strasbourg, Université de Strasbourg, CNRS UMR 7550, 11 rue de l' Université, F-67000 Strasbourg, France*
[2]*IPNL, Université de Lyon, Université Lyon 1, CNRS/IN2P3, 4 rue E. Fermi, F-69622 Villeurbanne cedex, France*
[3]*Scuola Normale Superiore, Piazza dei Cavalieri 7, I-56126 Pisa, Italy*

Accepted 2019 September 7. Received 2019 August 28; in original form 2019 May 16

ABSTRACT

We present a deep learning model trained to emulate the radiative transfer during the epoch of cosmological reionization. CRADLE (Cosmological Reionization And Deep LEarning) is an auto-encoder convolutional neural network that uses 2D maps of the star number density and the gas density field at $z = 6$ as inputs and that predicts 3D maps of the times of reionization $t_{\rm reion}$ as outputs. These predicted single fields are sufficient to describe the global reionization history of the intergalactic medium in a given simulation. We trained the model on a given simulation and tested the predictions on another simulation with the same parameters but with different initial conditions. The model is successful at predicting $t_{\rm reion}$ maps that are in good agreement with the test simulation. We used the power spectrum of the $t_{\rm reion}$ field as an indicator to validate our model. We show that the network predicts large scales almost perfectly but is somewhat less accurate at smaller scales. While the current model is already well suited to get average estimates about the reionization history, we expect it can be further improved with larger samples for the training, better data pre-processing and finer tuning of hyper-parameters. Emulators of this kind could be systematically used to rapidly obtain the evolving H II regions associated with hydro-only simulations and could be seen as precursors of fully emulated physics solvers for future generations of simulations.

Key words: methods: numerical – galaxies: evolution – intergalactic medium – cosmology: theory – diffuse radiation.

1 INTRODUCTION

The process of cosmic reionization is the period that sees the cosmic hydrogen content of the intergalactic medium (IGM) being progressively ionized by the first sources of ionizing radiation during the first billion years of cosmic history (Gnedin 2000; Barkana & Loeb 2001; Choudhury & Ferrara 2005; Dayal & Ferrara 2018 for a recent review). This process marks the last major transition for cosmic gas in the history of the Universe and is of prime importance to explain what happened to the next generation of galaxies and to understand the Universe we see today at $z = 0$.

Correctly modelling this phenomenon in order to interpret future observational results is one of the upcoming challenges in astrophysics. With the promise of new facilities dedicated to the study of this epoch with instruments like SKA (see Barry et al. 2016; Datta et al. 2016) or JWST (see Windhorst et al. 2006; Wang et al. 2019), the community wants to be ready to investigate the parameter space from the theoretical side. This can be done with a variety of models ranging from analytical (Chiu & Ostriker 2000; Furlanetto, Zaldarriaga & Hernquist 2004; Benson et al. 2006; Choudhury, Haehnelt & Regan 2009), seminumerical (Zahn et al. 2007; Alvarez et al. 2009; Thomas et al. 2009; Zahn et al. 2011), to full simulations (Gnedin & Abel 2001; Iliev et al. 2006; Ocvirk & Aubert 2011; Ocvirk et al. 2013; Rosdahl et al. 2013a; Chardin, Aubert & Ocvirk 2014; Gnedin 2014; Aubert, Deparis & Ocvirk 2015; Ocvirk et al. 2016; Aubert et al. 2018; Ocvirk et al. 2018a) incorporating an increasingly accurate description of the physics at play during the epoch of reionization.

Simulations of cosmic reionization are computationally expensive because of the necessary inclusion of radiative transfer physics: propagation at the speed of light and out-of-equilibrium thermochemistry induce short time-scales, leading to large amounts of calculations to cover the first billion years in the Universe history. Hardware acceleration, with e.g. GPUs, can reduce their cost (see e.g. Aubert & Teyssier 2010; Ocvirk et al. 2016) but requires dedicated devices, available only in limited numbers or in specific supercomputing facilities, even though their usage is becoming more widespread thanks to the rise of machine learning. Another way to accelerate such calculations is to use the so-called

★ E-mail: jonathan.chardin@astro.unistra.fr

reduced speed of light approximation (see e.g. Rosdahl et al. 2013b; Katz et al. 2017 for an extension of this technique to variable speed of light). With typical values of $\tilde{c} = [0.01 - 0.1] \times c$, computing times can be divided by factors ranging from 10 to 100. However, even with such performances, such simulations remain costly and can introduce spurious artefacts compared to simulations using the actual speed of light (Gnedin 2016; Ocvirk et al. 2018b; Deparis et al. 2019). Overall, even with such techniques, simulations of cosmic reionization are still challenging.

In this paper, we propose to use the recent advent of deep learning methods to reassess this issue. Machine learning algorithms are powerful in the sense that they can learn complex relationships in data, without requiring any prior functional form to describe a particular physical problem. Due to the increasingly large amounts of data encountered in astronomy, be it through observations or simulations, the use of machine learning techniques is becoming more and more widespread. For example, Kamdar, Turk & Brunner (2016a) and Kamdar, Turk & Brunner (2016b) used a large set of semi-analytical and full hydrodynamical data obtained from simulations to learn the underlying physics governing the galaxy formation process. Ucci et al. (2018, 2019) used supervised learning to infer physical properties of galaxies from their emission-line spectra. Among others, Schaefer et al. (2018) used convolutional neural network to build a strong gravitational lens detector while Parks et al. (2018) used the same kind of technique to characterize strong absorbers of neutral hydrogen (i.e. damped Ly α systems) in quasar spectra.

Ntampaka et al. (2019) recently reviewed what has been done in the field of cosmology with the advent of machine learning techniques. Among them, studies were undertaken to address the epoch of reionization in the context of deep learning. Shimabukuro & Semelin (2017) used such methodologies on synthetic **21 cm** power spectra to extract physical properties of the reionization process. Gillet et al. (2019) first proposed to use light cones of the 21cm surface brightness as input of convolutional neural network to predict cosmological parameters. In the same spirit, Hassan et al. (2019) used synthetic 21 cm light cones drawn from simulations to predict what the relative contribution to reionization between star-forming galaxies and AGNs is.

With this study, we aim to go beyond the aforementioned works, to predict physical fields relevant to reionization from other physical fields. Our aim is to use fields of gas density and star number counts as inputs of a neural network to predict maps of reionization times $t_{\rm reion}$. The reionization times maps encode the whole reionization history of a given simulation: having a neural network predictor would allow to assign locations of H II bubbles at all times in simulations without radiative transfer, which would in turn make possible the quick acquisition of, e.g. a mean reionization history associated with those simulations.

We propose to use actual radiative-hydrodynamics simulations of cosmic reionization to feed the learning process of such networks. To some extent, we aim at designing a tool similar to semi-analytical models, but rather than using an explicit model we propose to create an implicit model. Such a model would be provided by full-physics simulations and would constitute rather a 'seminumerical' model, orders of magnitude faster than the simulations it originates from. More generally, one can envision deep learning methods to emulate physics solvers (i.e. coupled differential equations solvers), using simulations' products as training models but with a much smaller execution times than for actual simulation codes: the radiative transfer case used here should merely be seen as an example of a much greater potential.

This paper is organized as follows. We first present the simulation of cosmic reionization used in this study. Secondly, we detail the architecture and the training performance of our neural network in Section 3 before giving the strengths of this model in Section 4. We finally discuss how our results could be improved and generalized in the near future in Section 5.

2 SIMULATIONS OF COSMIC REIONIZATION WITH EMMA

In this work, cosmological simulations of the reionization were produced using the EMMA simulation code (Aubert et al. 2015): the code tracks the collisionless dynamics of dark matter, the hydrodynamics of baryons, star formation and feedback and the radiative transfer using a moment-based method (see e.g. Aubert et al. 2018; Deparis et al. 2019). This code adheres to a Eulerian description, with fields described on grids, and enables adaptive mesh refinement techniques to increase the resolution in collapsing regions.

For the current study, we used an existing pair of large-scale, well-resolved simulations (with high enough resolution to follow the densest absorbers that are subject to self-shielding as shown in Chardin, Kulkarni & Haehnelt 2018). The two simulations share the same parameters, but with different displacement phases in the initial conditions. In both cases, the $(128\ {\rm Mpc\ h^{-1}})^3$ volume is sampled with 1024^3 cells at the coarsest level. Refinement is triggered when the number of dark matter particles exceeds 8, up to 6 refinement levels.

A Planck 2015 cosmology was used (Planck Collaboration XIII 2015) to generate the initial conditions, with a starting redshift of $z = 150$. Simulations were stopped at $z = 6$. The dark matter mass resolution is $2.1 \times 10^8\ {\rm M_\odot}$ and the stellar mass resolution is $6.1 \times 10^5\ {\rm M_\odot}$. Star particles produce ionizing radiation for 3 Myrs, with an emissivity provided by the Starburst99 model for a top-heavy initial mass function and a $Z = 0.001$ metallicity. Star formation proceeds according to standard recipes described in Rasera & Teyssier (2006), with an overdensity threshold equals to 20 to trigger the gas-to-stellar particle conversion with a 0.1 efficiency: such values allow the first stellar particles to appear at $z \sim 17$. Supernovae feedback follows the prescription used in Aubert et al. (2018): as they reach an age of 15 Myr, stellar particles dump 9.8×10^{11} J/stellar kg in the surrounding gas, one-third in the form of thermal energy, two-third in the form of kinetic energy.

Using these parameters, we obtain a cosmic star formation history consistent, but somewhat underestimated, with constraints by Bouwens et al. (2015). Even if these simulations are not fully consistent with observations, it is more than enough for the purpose of this paper, which is to demonstrate the idea that a machine learning algorithm can learn the physics of reionization and can be used to predict the ionization field of non-RT simulations. Hence, the technique exposed here could be generalized in the future on simulations more carefully calibrated with observations.

This pair of simulations was produced on the Occigen supercomputer (CINES, France) on a standard CPU architecture: EMMA GPU acceleration capabilities were not enabled and a reduced speed of light $\tilde{c} = 0.1c$ has been used to reduce the cost of radiative transfer. For the purpose of the current investigation, we did not use the simulation products at full resolution: outputs were degraded to a 256^3 resolution to fit within the capabilities of our hardware dedicated to neural network training.

These two simulations will be labelled, respectively, as TESTSIM and TRAINSIM. TRAINSIM is the simulation used for the actual training of our model, whereas TESTSIM is used to quantify

its predicting power. TESTSIM is never used during the training process and thus provides a way to test the model on a completely independent data set.

3 AUTO-ENCODER CONVOLUTIONAL NEURAL NETWORK

3.1 Outputs: $t_{\rm reion}$ fields

Our aim with this study is to predict the 3D $t_{\rm reion}$ field of a simulation, also known as 'reionization maps', built by marking cells with the cosmological time at which it crosses a given ionization fraction threshold. At the end of a simulation, it provides the full reionization history and this field can be used to reconstruct the H II regions' spatial distribution at all cosmic times (see Ocvirk et al. 2013; Aubert et al. 2018; Deparis et al. 2019). Predicting such a field with a neural network should be very useful for those who only have hydrosimulations at their disposal to get rough estimates of the mean reionization history of their simulations. In our case, $\rm t_{reion}$ maps are built on the fly by the EMMA simulation code. We choose an ionization fraction threshold $x_{\rm H\,II} \geq 0.5$ to consider cells of the simulation as ionized and to mark them with the corresponding cosmological time of reionization $t_{\rm reion}$. Varying this threshold from $x_{\rm H\,II} \geq 0.5$ to $\rm x_{H\,II} \geq 0.9$ is unlikely to affect the shape and size of H II regions (see Chardin, Aubert & Ocvirk 2012, appendix A.3) and thus is unlikely to affect the spatial distribution of $t_{\rm reion}$ maps. Therefore, varying this threshold should not impact what the neural network will learn in the next sections.

3.2 Inputs and data set pre-processing

In order to predict $t_{\rm reion}$ 3D maps, we use both the gas density (taken as the log of the baryon overdensity) and star particle number density, at $z = 6$. This choice is arbitrary and driven by simplicity: gas density tracks the distribution of photons absorbers, whereas the star number density tracks the distribution of emitters. Note that the star number density is only an incomplete view of the photon production history: no information about the age or the emissivity is provided here. Moreover, we could also imagine different kinds of ionizing sources at play in a single simulation. This would entail Lyman continuum photon production phases that differ in length (see e.g. Stanway, Eldridge & Becker 2016, who have shown that binary interactions increase the length of the ionization photon bright phases of sources). Evidently other choices would have been possible, possibly using more information, but as a proof of concept we will show that even this admittedly simple choice of inputs provides satisfying predictions at this stage.

As explained hereafter, we will use a convolutional neural network for our predictions, usually used for image processing in 2D. In theory, such networks can process 3D fields, such as an image with multiple channels, but then become quite memory consuming and less efficient, especially when the three dimensions are of commensurable sizes. Accounting for the limitations of the hardware currently available to us, we thus decided to make this first study using 2D CNN: gas and stellar number densities are provided to the CNN as 2D slices, and equivalently, predictions on $t_{\rm reion}$, are returned as 2D planes. Nevertheless, to capture some information along the direction normal to the plane, we Gaussian smooth the 3D gas and stellar fields along this direction: we take a smoothing length of $\sigma = 30$, corresponding to a size of 3.75 cMpc $\rm h^{-1}$ for the simulations studied here. We marginalized our search over different values for this smoothing scale, and the latter value gave us the best performance when training the neural network. For a 3D reconstruction, all successive slices of a $t_{\rm reion}$ cube are predicted and stacked. Of course, this creates discontinuities along the stacking direction: to mitigate this effect, we perform three separate 3D predictions using this procedure, stacking along the three different main directions and combine them to obtain our final 3D prediction of $t_{\rm reion}$. Further details can be found in Appendix B.

From our 256^3 3D fields taken from TRAINSIM, we construct a sample of 3000 maps of 128×128 cells for the stellar and gas densities, and $t_{\rm reion}$. This constitutes what we usually call the training set. The maps are picked randomly inside the whole 3D fields and the same location is taken for the three fields. In addition to the training set, we also build a test set composed of 500 additional 128×128 maps for the three fields, still from TRAINSIM. Such a test set is here to measure the accuracy of the trained model on unseen data during the training process. Therefore, we ensure that the maps taken to build this test set are different from the ones belonging to the training set.

Finally, we normalize the input in the neural network and we proceed as follows for both the stellar (S) and gas density (D) fields:

(i) We take the mean of our fields in the whole training set: $<S>$ and $<D>$

(ii) We subtract that value from all the values in the maps of the training set: $S = S - < S >$ and $D = D - < D >$

(iii) We calculate the standard deviation of those new fields: std(S) and std(D)

(iv) We divide all the values of S and D by this value: $S = S/\mathrm{std}(S)$ and $D = D/\mathrm{std}(D)$

Fig. 1 shows an example of data used to train the neural network. We show five different examples of both the transformed stellar and gas density fields that are the input of the network, and the corresponding $t_{\rm reion}$ field the network aims to predict. We can see at first glance the correlation between the three different fields in each example case. Therefore, we can expect that the network, with data that are transformed this way, should be able to infer the underlying correlation.

3.3 Convolutional neural network architecture

Fig. 2 shows a schematic view of the neural network used here (see Appendix A for full details). The neural network we build is a special case of convolutional neural networks. It is called an auto-encoder and has the unique property of generating a complete image as an output. An auto-encoder is a non-recurrent neural network propagating forward (i.e. we process from left to right in Fig. 2) with an input layer, an output layer and one or more hidden layers in-between. An auto-encoder is always divided in two parts: the encoder and the decoder. The encoder is a succession of convolutions of the input of a layer with filters of a particular size. The results are then downsampled and given as inputs to the next layer. The decoder is the symmetric counterpart of the encoder. Instead of convolution and downsampling, a layer is composed of a deconvolution plus an upsampling of the results.

The auto-encoder we build here is special because it has more than one input image even if we aim to predict a single output image. In practice, we just apply the same series of convolutions and downsamplings independently for both our input stellar and gas density fields (the two distinct red rectangles on the left in Fig. 2). After the forward passing of these maps through the layers of their dedicated branches of the encoder, the results are merged together (i.e. we take the average of the output maps of the gas density and the

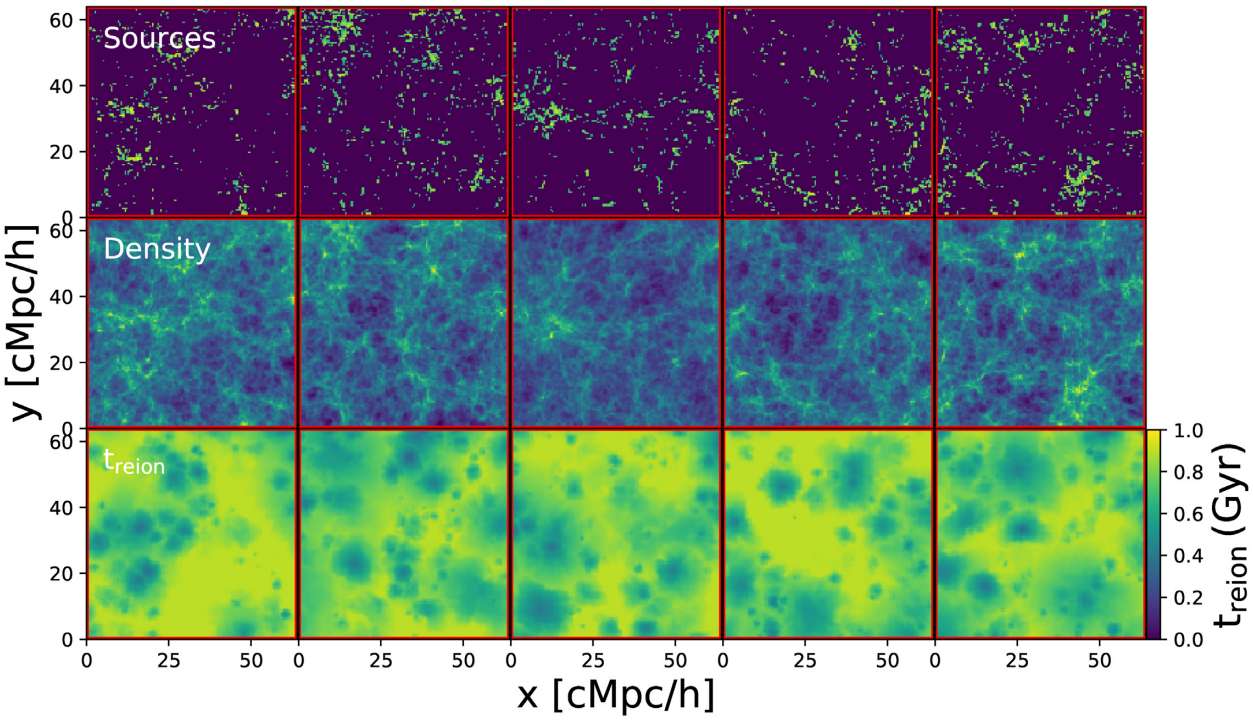

Figure 1. Example of fields used to train our neural network. The stellar particles number counts and gas density fields are used as inputs of the network and the t_{reion} field is what should be predicted. The stellar and the gas density fields are pre-processed as described in Section 3.2 while the t_{reion} field is not touched. Briefly, they are zero-centred as well as unit variance transformed. Moreover, the stellar and gas density fields are Gaussian smoothed in the transverse direction of the plane seen on the figure to keep 3D information in two dimensions. We can see at first glance that the three fields are correlated. In principle, the neural network should be able to infer this underlying correlation.

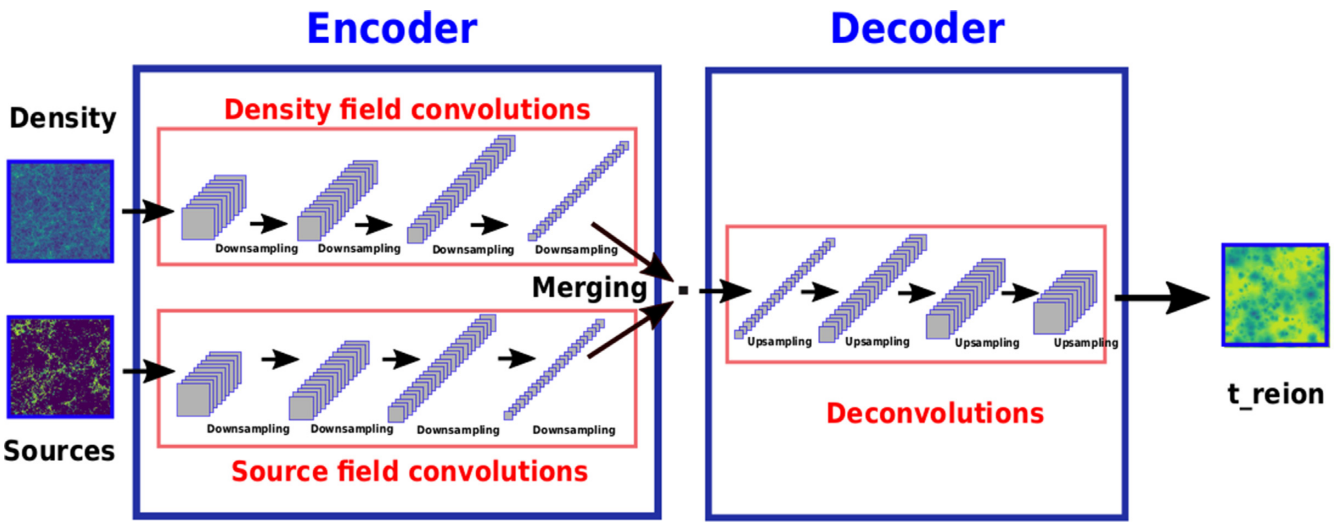

Figure 2. Architecture of the convolutional auto-encoder model used to predict maps of t_{reion} (see Appendix A for full details). The auto-encoder has two entries: the maps of the gas density field and of the star number density which are both Gaussian-smoothed (see Section 3.2 for detailed explanations). There are two distinct blocks of convolution filters applied on both fields independently represented by the two red rectangles on the left. These two branches of convolution represent the encoder. After those independent series of convolutions, the outputs of the two last layers in the network are averaged together before entering the process of deconvolution (represented here as the third red rectangle which constitutes the decoder). At the end, one last layer with a linear activation function is applied to produce a full 2D map of t_{reion} with continuous values.

stellar branches) to feed the decoder. This input successively goes through the same number of layers as in the encoder. This eventually leads to a final map (t_{reion}) of the same size as the starting inputs.

We employ the usual Adam algorithm for the optimizer and we choose the mean squared error between the predicted and true 2D maps for the loss function to optimize. The choice of mean squared error is dictated by the regression nature of the problem we are facing here (i.e. predicting continuous values of t_{reion} instead of discrete values) in contrast to classification problems.

3.4 Training the neural network

The auto-encoder described in the last section is built using the python API for neural networks KERAS[1] (Chollet 2015). Training the network is done on two Tesla K20 GPUs with the parallel training option of KERAS. Training on our set-up takes about 28 h with both GPUs when aiming to achieve best performance. In principle, we can train our network on a much larger number of GPUs and therefore the wall clock time needed to train the network can be greatly reduced.

As already mentioned, to train a neural network, we build both a training set and test set. The training set, is made of data that are used to minimize the loss function. On the other hand, the test set is made of data of the same nature as in the training set but are not used during the minimization process. They are only produced to control how a current version of the trained model performs on unseen data. Therefore, we use both the training and test set during training to monitor each training process.

To monitor our training performance we use two indicators. First, we use the mean squared error (MSE) between the predicted $t_{\rm reion}$ and the true values. In practice, we want the MSE to decrease during the learning process until it reaches a plateau, indicating that the maximum learning potential has been achieved. However, the MSE value is not meaningful taken in isolation, and does not tell us much about the quality of the predictions. The same MSE value can correspond to predictions of variable quality from one problem to another.

To measure the correctness of our prediction, we use a second indicator which is called the coefficient of determination R^2 calculated with the following formula (see Gillet et al. 2019):

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

In practice, a value close to 1 represents a 100 per cent match between our original data and the ones predicted by the neural network.

The upper panel of Fig. 3 shows the evolution of the MSE as a function of the number of training epochs, while the lower panel shows the evolution of R^2. We show these curves for our best model, after we found the best way to pre-process our data, and the best network architecture with the best hyperparameters. Trends are shown for the training set and test set, both from TRAINSIM. We clearly see the MSE decreasing quickly at the beginning of training for the first 250 epochs for both the training and test set. This means that the choice of parameters is well suited to the current problem and that the model learns efficiently. After about 250 epochs, the test set reaches a plateau while the training set keeps decreasing. We continue training up to the moment when the MSE curve reaches a plateau for the training set. The beginning of the training set plateau is generally considered as the moment when the best performances are achieved. We achieve this after 2500 epochs.

Focusing on R^2 in the bottom panel, we observe that the model reaches an accuracy of about $R^2 \sim 0.99$ on the training set when the MSE stabilizes. Meanwhile the test set reaches a value of about $R^2 \sim 0.9$, which means that our model generalizes well on unseen data. However, both the training and test set are built from the TRAINSIM simulation. Even if we make sure they are not the same maps, nothing guarantees that the model generalizes well on other, completely disconnected simulations. That is why, we ran the TESTSIM to test our model's performances on new data. All the results given in Section 4 will thus be given by applying the model to this TESTSIM simulation, unseen during the training.

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

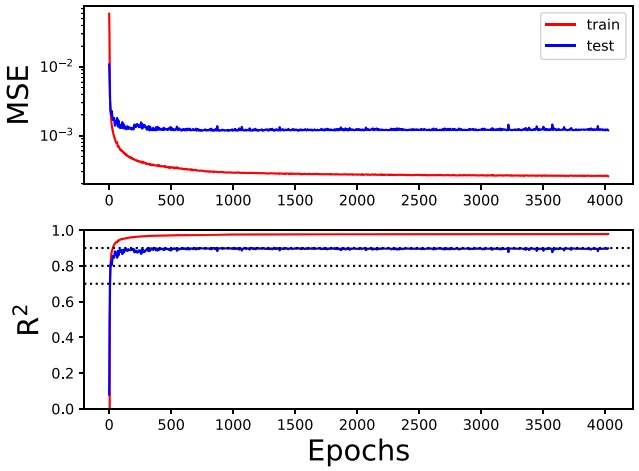

Figure 3. Training performance curves. The top panel shows the evolution of the mean squared error (MSE) between the predicted values and the real ones from the TRAINSIM simulation as a function of the number of epochs for both the training set and testing set. This is the actual value of the loss function used to train the model and what the gradient descent algorithm is trying to minimize. The bottom panel shows the evolution with the number of epochs of the coefficient of determination R^2 of equation (1). It allows to monitor how our model matches the original values and in particular how it performs on unseen data with the testing set drawn from the TRAINSIM simulation. The different horizontal dashed lines show values of $R^2 = 0.7$, 0.8, and 0.9.

4 RESULTS

4.1 Field reconstruction

To measure the performance of our model on unseen data, we first use our trained network to reconstruct a field of $t_{\rm reion}$ in the TESTSIM simulation. We use the gas density and the stellar fields of this simulation and we transform them the same way we transformed the input training data (see Section 3.2). We then predict multiple 2D slices to reconstruct the whole $t_{\rm reion}$ cube following the procedure described in Appendix B.

Fig. 4 shows central slices of the reconstructed cube of $t_{\rm reion}$ in the three different directions. The model is well adapted for predicting $t_{\rm reion}$ maps in the three directions. The colourmap in both the predictions and original data is set to be the same to enable direct comparison. Overall, the model predicts a range of continuous values of $t_{\rm reion}$ that are within the same range as the original simulation. Moreover, the global shape of the field seems to be well predicted by the network which means that the large-scale structures of the field seems to be well-learned by the model.

However, we report some differences at smaller scales. The network struggles to predict the exact same shape for the edges of $t_{\rm reion}$ bubbles. Moreover, the peaks of small $t_{\rm reion}$ values are too high compared to the original data, meaning that the first sources episodes of reionization are only partially recovered. Some small $t_{\rm reion}$ bubbles are missed during the reconstruction, or some spurious bubbles are created where they are not present in the original simulation.

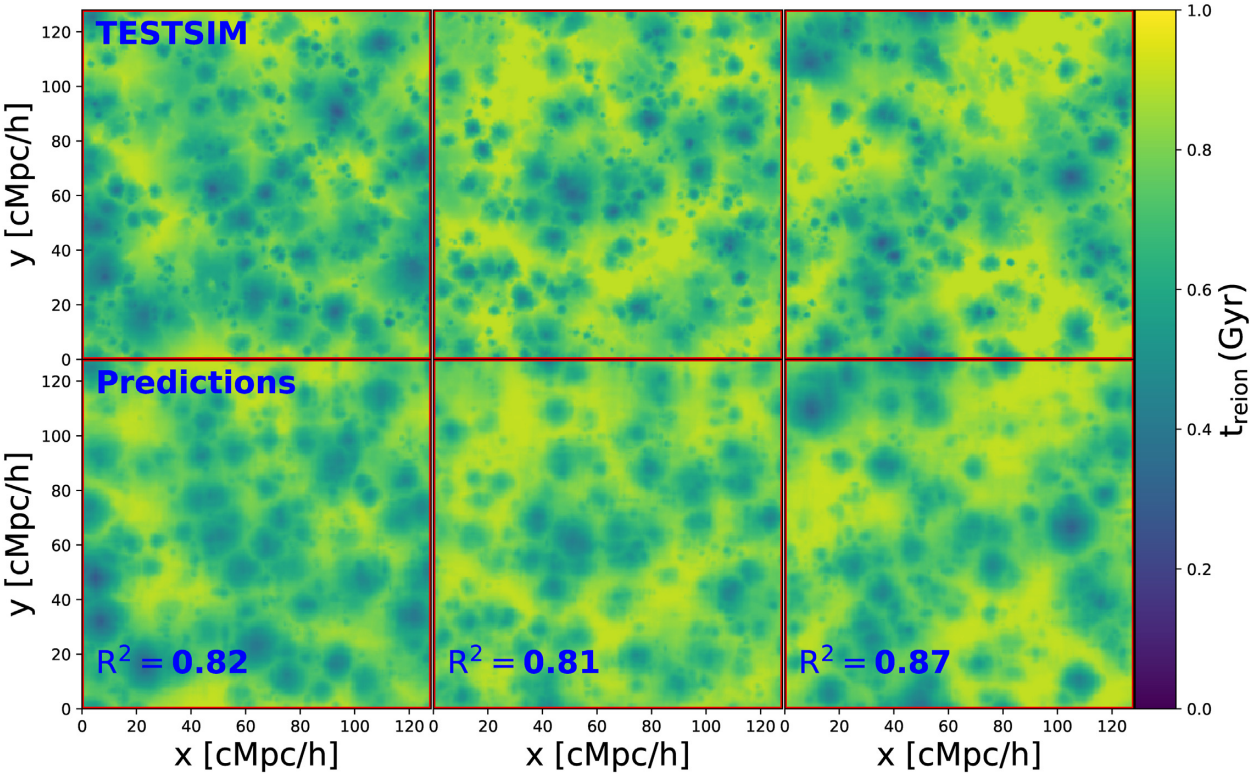

Figure 4. Example of slices of $t_{\rm reion}$ from the TESTSIM simulation and the corresponding predictions from the model. The upper panel shows three different slices from the TESTSIM simulation, while the lower panel shows the prediction of the same slice with our best model. Average values of $R^2 \sim 0.83$ are currently achieved with our best model. The predicted fields are reconstructed following the procedure described in Section 4.1.

All the aforementioned drawbacks of the model show its limited capacity to make robust predictions at small scales, which could be improved with a combination of a larger data set for the training and different kinds of inputs. For example, the star number density at $z = 6$, is actually degenerate and can be similar for different source production histories: it is therefore not surprising that our model struggles to perfectly reproduce the time evolution. In fact, given this limitation, the ability of the network to predict a reionization timeline similar to the actual one can even be seen as surprising and indicates that to some extent, the star production history is encoded in the gas density distribution and stellar number density. The inclusion of information about the source ages could surely improve the prediction on reionizations' evolution, especially at early times.

Finally, in the lower left corner of each slice of the reconstructed fields, we also show the value of the R^2 coefficient of prediction for the corresponding slices compared to the original slice of the TESTSIM simulation. We reach average values of $R^2 \sim 0.83$ over the three directions with our best model, without any fine-tuning of the model. Again, we expect to increase this accuracy with fine tuning of the hyperparameters of the model, a bigger sample for the training, and better pre-processing of the input data.

4.2 True versus predicted values

As a second test, we also construct the 2D histogram of the true versus predicted values of $t_{\rm reion}$. Fig. 5 shows the number count of cells lying in the true-predicted plane. The red line on the figure shows the one-to-one relation. To highlight the differences, the bottom and left histograms show the mean and the standard deviation of the residual: r = Predicted − True along the vertical and horizontal directions, respectively.

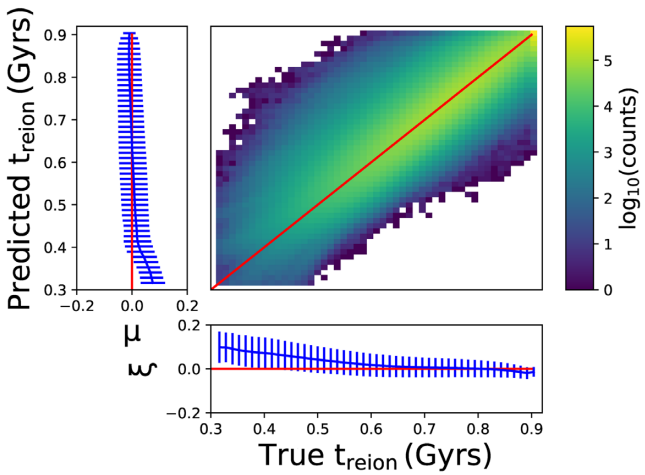

Figure 5. 2D histogram of the true versus predicted values of $t_{\rm reion}$ for our best neural network model once the training is finished. The histogram is constructed on the whole 3D reconstructed cube of the TESTSIM simulation as explained in Section 4.1. The red line shows the one-to-one relation while the colourmap encodes the number count of cells lying in the 2D space of true versus predicted. The bottom and left histograms show the mean and the standard deviation of the residual: r = Predicted − True in the vertical and horizontal directions. The bottom histogram is the learning error, $p(r|\text{True})$, while the side histogram is the recovery uncertainty, $p(r|\text{Predicted})$.

The left-hand side histogram which displays the distribution of residuals as a function of the prediction ($p(r|\text{Predicted})$) is an actual

measure of the uncertainties on predictions μ. Such a test is standard in deep learning model validation, and the closer the distribution to the one-to-one relation, the better the performance of the network (see Gillet et al. 2019). Overall, our model tends to predict values of $t_{\rm reion}$ close to the one-to-one relation which once again demonstrates the ability of our network to perform this particular task.

We note that the average of the residuals in this case is well centred on the zero residual value for $t_{\rm reion} \geq 0.4$ Gyrs. We report values of $\overline{\sigma} = 0.045$ Gyrs for the mean of the uncertainty on prediction along all this range of $t_{\rm reion}$ values. This value is fairly constant over all this range and it means that we only have a 4.5 per cent error on our predictions on average. We measure a minimum value of $\sigma = 0.010$ Gyrs and a maximum uncertainty of $\sigma = 0.061$ Gyrs indicating an error fluctuating between 1 and 6 per cent compared to real values. However, the model clearly overestimates the values of $t_{\rm reion} < 0.4$ Gyrs which means that we miss the first ionized regions in our predictions.

Finally, the bottom histogram, showing the distribution of residuals as a function of true values ($p(r|\text{True})$), represents the network error ξ. The average value of the residual is well-centred around zero in the range [0.5–0.9] Gyrs for values of $t_{\rm reion}$. We conclude that our network makes robust predictions in this range with a mean uncertainty of $\overline{\sigma} = 0.05$ along these values. This is not surprising since these values of $t_{\rm reion}$ correspond to the large scales that are well predicted in the maps of Fig. 4. However, we note that larger uncertainties are reported for $t_{\rm reion} < 0.5$ Gyrs. In this case, the mean residual is significantly above the zero value, which suggests a larger predicted $t_{\rm reion}$ in this range. This can be seen in Fig. 4, where these peaked locations in the maps have higher values in the predictions compared to the original data. Therefore, our model seems to struggle to predict the smaller scales in the maps, which correspond to the first locations in the simulation that were reionized by the first generation of ionizing sources.

4.3 Power spectra

As a third test, we also compute the 1D power spectrum $P_{\rm 1D}$(k) of the 3D field of $t_{\rm reion}(\vec{r})$ for both the original TESTSIM simulation and the reconstructed prediction of the network. The power spectrum is defined as the azimuthal average of the square of the module of the 3D Fourier transform $\delta(\vec{k})$ of the 3D $t_{\rm reion}(\vec{r})$ field. This is computed as follows:

$$\delta(\vec{k}) = \int t_{\rm reion}(\vec{r}) \mathrm{e}^{-2\pi i \vec{k}.\vec{r}} \mathrm{d}\vec{r} \tag{2}$$

$$P_{\rm 1D}(k) = < |\delta(\vec{k})|^2 >_{|\vec{k}|=k} = \frac{\sum_{|\vec{k}|=k} |\delta(\vec{k})|^2}{\sum_{|\vec{k}|=k} 1} \tag{3}$$

Fig. 6 shows these two power spectra as well as the ratio of the two.

Overall, we report a perfect match between true and predicted values at large scales. The two spectra are on top of each other up to scale $k = 0.1$ h cMpc^{-1} (i.e the ratio of both spectra is almost equal to 1). This is in agreement with what is observed in Figs 4 and 5, where large-scale structure in the maps (with values of $t_{\rm reion} > 0.5$ Gyrs) are well predicted by the network.

However, at scales $k > 0.1$ h cMpc^{-1} the network seems to underpredict the power compared to the real simulation. Again, this discrepancy corresponds to small scales that are missed in the maps of Fig. 4 with $t_{\rm reion} < 0.5$. This means that the network struggles to keep track of the first ionizing sources that appeared during the simulation.

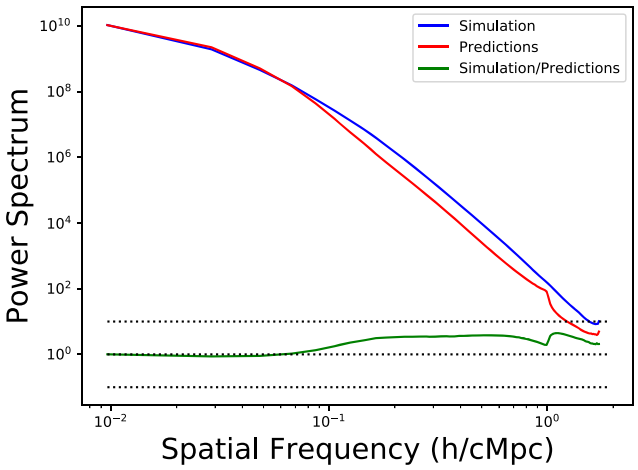

Figure 6. Power spectra of the $t_{\rm reion}$ fields. The blue and red lines show, respectively, the power spectrum of the original TESTSIM simulation and the one predicted by our best model once the training is finished. Power spectra are computed on the whole 3D cube. The green line shows the ratio of both power spectra.

4.4 Reionization history

The previous sections show results on the prediction of the field of $t_{\rm reion}$, which encapsulates the whole reionization history of a given simulation. Here, we propose to use this map to demonstrate the potential of our model to predict the evolution of average quantities during the process of cosmic reionization, and how it compares with real simulations. We aim at showing how this could be useful for those who only have hydro simulations, and want to get an emulation of the radiative transfer calculation without performing it.

First, in Fig. 7, we show the reconstructed evolution with cosmic time of the ionized regions' expansion. To synthesize this evolution, we construct light cones from $t_{\rm reion}$ slices. The upper panel shows the evolution of the light cone constructed from the central $t_{\rm reion}$ slice of the TESTSIM simulation, while the lower panel shows the same evolution predicted by the model. Such fields are constructed as follows:

(i) We first take a slice of the 3D $t_{\rm reion}$ field.
(ii) We consider a cosmic time of $t_{\rm H\,II}$ at which we want to create the H II regions' spatial distribution.
(iii) We keep all the $t_{\rm reion} < t_{\rm H\,II}$ cells.
(iv) We mark them as ionized with a value of one.
(v) We consider the other cells as neutral with a value of zero.
(vi) We repeat this for multiple values of $t_{\rm H\,II}$ and stack the results to construct the light cone.
(vii) We follow this procedure for both the $t_{\rm reion}$ field of the TESTSIM simulation and prediction of the network.

Overall the two light cones look rather similar between the prediction and the TESTSIM simulation. We observe better agreement at large cosmic times (i.e. when the reionization process is well advanced). The H II bubbles in the predicted field are at the right location with sizes comparable to the original ones. However, the edges of the bubbles are somewhat different: some of them are merged in a single bubble in the predicted field, whereas several disconnected bubbles are reported in the original data. We also

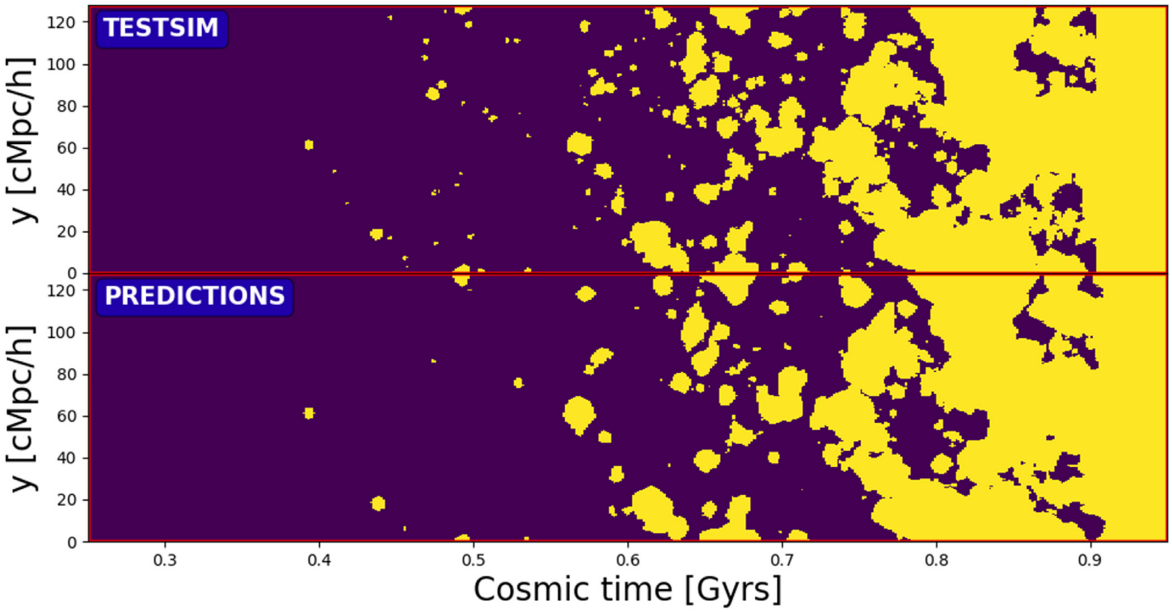

Figure 7. Example of light cone from the TESTSIM simulation and the corresponding predictions from the model. The upper panel shows the light cone of the central slice of the TESTSIM simulation, while the lower panel shows the prediction of the same slice with our best model.

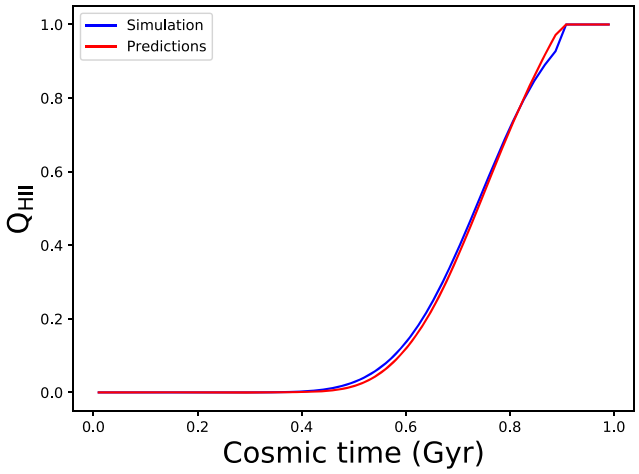

Figure 8. Evolution of the volume filling factor of H II regions $Q_{\mathrm{H\,II}}$ with cosmic times in both the original simulation (TESTSIM) and prediction of the neural network. $Q_{\mathrm{H\,II}}$ is calculated from the whole 3D field of t_{reion} in both cases by taking the cumulative sum of the histogram of the t_{reion} values.

observe some bubbles completely disappearing in the prediction compared to the TESTSIM simulation. The inaccuracy of the model at predicting the smallest scales is reflected here when predicting the H II regions' spatial distribution as a function of cosmic time.

At early times, for $t \sim 0.5$ Gyrs, some of the first and smallest bubbles are missing. This illustrates again the fact that the model struggles at predicting the first stages of the H II regions' expansion. This is due to the fact that the model is unable to predict the smallest t_{reion} values in Fig. 4, which reveals the limitations of the model to get accurate apparition times for the first generations of ionization sources early on in the simulation.

As a second test, we compute the evolution of the fraction of the volume that is ionized at a given cosmic time. In Fig. 8, we present the evolution of the volume-filling factor of H II regions $Q_{\mathrm{H\,II}}$ in both the TESTSIM simulation and model prediction. The evolution of this quantity is calculated by binning the cosmic time period and by getting the cumulative sum of ionized cells at a given cosmic time from the t_{reion} field. We report an almost perfect match of the two curves in Fig. 8 at all cosmic times. This demonstrates the ability of our model to predict a global reionization history during the whole simulation. We observe some minor differences at cosmic time $0.4 \leq t \leq 0.6$, where $Q_{\mathrm{H\,II}}$ of the TESTSIM is somewhat above the model prediction. Once again, this is due to the inability of the model to perform at predicting the smallest scale in the t_{reion} field. Overall, our deep learning model is already well designed to emulate a global reionization history, which can be useful for a wide range of studies.

5 DISCUSSION AND CONCLUSION

In this section, we present and discuss our global results with their successes and drawbacks. We finally conclude with the implications of such a study for the near future and what it may imply for the future of numerical simulations.

5.1 Successes

With this study, we have demonstrated that deep learning models can emulate the physics of the radiative transfer occurring during the reionization epoch of the Universe. We used an auto-encoder neural network, usually designed for data compression, to create a generative model that predicts the whole reionization history encapsulated in a single field: the map of reionization times t_{reion}. The built network takes the stellar number density as well as the gas density field from a simulation at the end of the reionization as inputs and produces the t_{reion} field as an output.

With our current optimization strategy, we achieve a determination coefficient $R^2 \sim 0.83$ in recovering the whole 3D field of t_{reion} on a test simulation that was never seen during the training of the network. The model was therefore successful at reproducing

a variety of scales in this field down to $k = 0.1$ h cMpc^{-1} which is already useful for a wide range of studies. Moreover, the model has shown its ability to generate H II bubble dynamics in good agreement with real simulations with full radiative transfer. It gives at the end a mean reionization history almost identical compared to real data.

For the time being, we have therefore proved that an auto-encoder neural network architecture for emulating radiative transfer during reionization is a promising solution. This is really encouraging since the prediction of a complete 256^3 $t_{\rm reion}$ cube is produced in a much faster way than a complete reionization simulation that includes radiative hydrodynamics. The prediction of the total 256^3 cube is done in about 3 min on a single CPU. The prediction could be even faster than this if we parallelize the reconstruction process of the whole cube, and optimize the current python API. In comparison, post-processing radiative transfer in the same way as in Chardin et al. (2015) for a simulation with the same resolution and the same number of cells would be done in about 1 h on a single last generation Pascal P100 GPU.

5.2 Caveats and road to improvements

The training of the current neural network is not perfect, but we expect a great scope of improvement. First, we have shown that the predicted $t_{\rm reion}$ field is inaccurate at small scales. This is related to the inability of the model to predict the smallest values of $t_{\rm reion}$, which correspond to the location where the first ionizing sources appeared during the TESTSIM simulation. This is not surprising as the current model is trained with the gas density and the stellar number density fields taken at the end of the simulation, at $z = 6$. Therefore, these data used as inputs of the model do not hold any temporal information about the ionizing source history through the whole simulation. However, the $t_{\rm reion}$ field we aim to predict is a summary of this integrated history. We could therefore expect to gain much accuracy in the prediction if we introduce such temporal information when training the network. For example, we could imagine taking a third field as an entry of the network, which could be the average cosmic time of apparition of the ionizing sources inside each cell throughout the whole simulation. In practice, adding such a third field should not prove troublesome when getting access to a larger number of GPUs for the training phase of the neural network.

We also expect improvement by increasing the quantity of data used to train the network. Indeed, the current performance was achieved with training on only a sample of 3000 independent images for the training set. It is well known that increasing the training sample increases the performance and usually training sample of the order of 80 000 images (see e.g Gillet et al. 2019) are used, which is much higher than what we currently have. Moreover, the training set was built from a single simulation. However, building the training sample from different simulations instead of a single one should improve the performance. Indeed, the current model can somewhat overfit in learning a biased representation of the density field for the particular initial conditions used for the simulation taken for the training.

We also expect some improvement with the properties of the network itself. First of all, the optimization of hyperparameters was only briefly investigated in the current training of the model. We can expect to get even better performance by focusing more on hyperparameter tuning. Other choices for the loss function to minimize could also be investigated such as a customized loss function tuned to perform this particular task. Moreover, we could also reconsider our network architecture. We could imagine adding layers and changing the number of filters in each layer. The size of the convolution filters in each layer could also be changed, and systematic studies for tuning these parameters could be investigated more carefully.

Furthermore, the predictions of our neural network are currently done in two dimensions, as we are limited by hardware considerations. Therefore, we had to smooth the gas density field and the stellar field in the transverse direction to the plane we are trying to predict. However, we could expect better performance of the network by using 3D convolutions instead of 2D as currently done, providing a direct prediction for 3D cubes and getting rid of the additional process of reconstructing the volume from 2D maps.

Finally, for the time being, the neural network has only been trained on a given simulation, with a particular box size and resolution as well as with a specific set of parameters for the input physics. Therefore, the network trained this way is only adequate when predicting $t_{\rm reion}$ maps on top of hydrosimulations with the same properties. We briefly tested the prediction of $t_{\rm reion}$ maps associated with simulations with different parameters with the current version of the network. Unsurprisingly, the model failed to perform satisfyingly on simulations with different parameters compared to the one used for training. However, We see clear avenues of improvement to pursue, and hope to train networks that could perform on a variety of hydrosimulations with a large range of parameters. For the time being, we have shown that the methodology is feasible, and we plan to work on more generic networks for forthcoming studies.

5.3 Conclusion

With this study, we lay the groundwork for developing emulators of reionization simulations. We have shown that deep learning methods should help to emulate realistic simulations very efficiently. Such techniques could considerably speed-up our way to predict the ionization field associated with a hydrodynamic simulation, when compared to full radiative transfer calculations. The model presented in this study still suffers from disparities with actual simulations, but we expect great possibilities of improvement. Of course, training such models requires a large sample of existing simulations, but many of these simulations have already been run by the community and constitute a data base that could be used for the systematic training of neural networks (see Wise & Abel 2011; Chardin et al. 2012; Gnedin 2014; Gnedin & Kaurov 2014; Chardin et al. 2015; Ocvirk et al. 2016; Ocvirk et al. 2018a; Rosdahl et al. 2018, among others). We plan to use large sets of existing simulations with different parameters and different sizes and resolutions, thus aiming at creating a data base of networks that could be used by the community to emulate the radiative transfer on a variety of hydrosimulations with different parameters. Finally, a long-term objective would be to end up with a neural network model that could emulate a complete simulation at once using only the initial conditions of the original simulation. Such an idealistic network could perhaps emulate all the ingredients of cosmic reionization simulation at once: the dynamic of dark matter (see e.g. Rodríguez et al. 2018), the hydrodynamic of the gas (see e.g. Zamudio-Fernandez et al. 2019) and the radiative transfer. More realistically, one could imagine replacing specific modules within existing simulation codes with trained neural networks. With the data base of simulations currently at our disposal, and the promise of more to come, a very exciting time for deep learning science applied to cosmology is upon us.

ACKNOWLEDGEMENTS

We thank contributors to SciPy, Matplotlib, pyDOE, and the PYTHON programming language. We thank the KERAS and Talos API for deep learning machinery and optimization in PYTHON. This work was granted access to the HPC resources of CINES (Centre Informatique National de l'Enseignement Supérieur) under the allocation 2019- A005041061 attributed by GENCI (Grand Equipement National de Calcul Intensif).

REFERENCES

Alvarez M. A., Busha M., Abel T., Wechsler R. H., 2009, ApJ, 703, L167
Aubert D. et al., 2018, ApJ, 856, L22
Aubert D., Teyssier R., 2010, ApJ, 724, 244
Aubert D., Deparis N., Ocvirk P., 2015, MNRAS, 454, 1012
Barkana R., Loeb A., 2001, Phys. Rep., 349, 125
Barry N., Hazelton B., Sullivan I., Morales M. F., Pober J. C., 2016, MNRAS, 461, 3135
Benson A. J., Sugiyama N., Nusser A., Lacey C. G., 2006, MNRAS, 369, 1055
Bouwens R. J. et al., 2015, ApJ, 803, 34
Chardin J., Aubert D., Ocvirk P., 2012, A&A, 548, A9
Chardin J., Aubert D., Ocvirk P., 2014, A&A, 568, A52
Chardin J., Haehnelt M. G., Aubert D., Puchwein E., 2015, MNRAS, 453, 2943
Chardin J., Kulkarni G., Haehnelt M. G., 2018, MNRAS, 478, 1065
Chiu W. A., Ostriker J. P., 2000, ApJ, 534, 507
Chollet F., 2015, Available at: https://github.com/keras-team/keras
Choudhury T. R., Ferrara A., 2005, MNRAS, 361, 577
Choudhury T. R., Haehnelt M. G., Regan J., 2009, MNRAS, 394, 960
Datta K. K., Ghara R., Majumdar S., Choudhury T. R., Bharadwaj S., Roy H., Datta A., 2016, J. Astrophys. Astron., 37, 27
Dayal P., Ferrara A., 2018, Phys. Rep., 780, 1
Deparis N., Aubert D., Ocvirk P., Chardin J., Lewis J., 2019, A&A, 622, A142
Furlanetto S. R., Zaldarriaga M., Hernquist L., 2004, ApJ, 613, 1
Gillet N., Mesinger A., Greig B., Liu A., Ucci G., 2019, MNRAS, 484, 282
Gnedin N. Y., 2000, ApJ, 535, 530
Gnedin N. Y., 2014, ApJ, 793, 29
Gnedin N. Y., 2016, ApJ, 833, 66
Gnedin N. Y., Abel T., 2001, New A, 6, 437
Gnedin N. Y., Kaurov A. A., 2014, ApJ, 793, 30
Hassan S., Liu A., Kohn S., La Plante P., 2019, MNRAS, 483, 2524
Iliev I. T., Mellema G., Pen U., Merz H., Shapiro P. R., Alvarez M. A., 2006, MNRAS, 369, 1625
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
Katz H., Kimm T., Sijacki D., Haehnelt M. G., 2017, MNRAS, 468, 4831
Labach A., Salehinejad H., 2019, preprint (arXiv:1904.13310)
Ntampaka M. et al., 2019, BAAS, 51, 14
Ocvirk P. et al., 2016, MNRAS, 463, 1462
Ocvirk P. et al., 2018a, preprint (arXiv:1811.11192)
Ocvirk P., Aubert D., 2011, MNRAS, 417, L93
Ocvirk P., Aubert D., Chardin J., Knebe A., Libeskind N., Gottlöber S., Yepes G., Hoffman Y., 2013, ApJ, 777, 51
Ocvirk P., Aubert D., Chardin J., Deparis N., Lewis J., 2019, A&A, 626, 77
Parks D., Prochaska J. X., Dong S., Cai Z., 2018, MNRAS, 476, 1151
Planck Collaboration XIII, 2015, A&A, 594, 63
Rasera Y., Teyssier R., 2006, A&A, 445, 1
Rodríguez A. C., Kacprzak T., Lucchi A., Amara A., Sgier R., Fluri J., Hofmann T., Réfrégier A., 2018, Comput. Astrophys. Cosmol., 5, 4
Rosdahl J. et al., 2018, MNRAS, 479, 994
Rosdahl J., Blaizot J., Aubert D., Stranex T., Teyssier R., 2013a, MNRAS, 436, 2188
Rosdahl J., Blaizot J., Aubert D., Stranex T., Teyssier R., 2013b, MNRAS, 436, 2188
Schaefer C., Geiger M., Kuntzer T., Kneib J.-P., 2018, A&A, 611, A2
Shimabukuro H., Semelin B., 2017, MNRAS, 468, 3869
Stanway E. R., Eldridge J. J., Becker G. D., 2016, MNRAS, 456, 485
Thomas R. M. et al., 2009, MNRAS, 393, 32
Ucci G. et al., 2019, MNRAS, 483, 1295
Ucci G., Ferrara A., Pallottini A., Gallerani S., 2018, MNRAS, 477, 1484
Wang L. et al., 2019, BAAS , 51, 399
Windhorst R. A., Cohen S. H., Jansen R. A., Conselice C., Yan H., 2006, New A Rev., 50, 113
Wise J. H., Abel T., 2011, MNRAS, 414, 3458
Xiao J. M., Shen C., Yang Y., 2016, Advances in Neural Information Processing Systems 29, 2802
Zahn O., Lidz A., McQuinn M., Dutta S., Hernquist L., Zaldarriaga M., Furlanetto S. R., 2007, ApJ, 654, 12
Zahn O., Mesinger A., McQuinn M., Trac H., Cen R., Hernquist L. E., 2011, MNRAS, 414, 727
Zamudio-Fernandez J., Okan A., Villaescusa-Navarro F., Bilaloglu S., Derin Cengiz A., He S., Perreault Levasseur L., Ho S., 2019, preprint (arXiv:1904.12846)

APPENDIX A: AUTO-ENCODER ARCHITECTURE DETAILS

Here, we detail the architecture of our auto-encoder neural network. Our neural network is composed of a series of four hidden layers for both the encoder and decoder. Table A1 gives the details on the number of filters and their size in all four hidden layers. To improve our model architecture, we also add what we call skip connections in the decoder part. Skip connections are here to add information when inputting maps into a layer of the network. In practice, it is a merging of multiple layer outputs to construct an input which is not only the output of the previous layer. In our case, we merge the outputs of every layer in the decoder with the corresponding outputs in the encoder in both the gas density and stellar branches. It allows us to combine information from the current decoded version and the one at the corresponding step in the encoded version. In practice, training with skip connections improves the results compared to training without (Xiao, Shen & Yang 2016).

To avoid overfitting during the training (i.e. the fact of achieving a good fit for our model on the training data, while it does not generalize well on new, unseen data), we also add batch normalization plus dropout regularization right after the convolution/deconvolution, respectively, in the encoder/decoder in each layer. Batch normalization is a transformation that maintains the mean output of a layer close to zero and its standard deviation close to one. Dropout regularization is performed right after batch normalization and is what prevents overfitting. Dropout regularization is activated through a value between zero and one that corresponds to a probability to shut down certain neurons (i.e. a given filter in a given layer). The fact of randomly shutting down neurons at every epoch is known to improve the accuracy of the model on unseen data (Labach & Salehinejad 2019).

We use the Talos[2] tool with KERAS to tune our hyperparameters. Talos allows to marginalize over the hyperparameter space and gives correlations between them. In our case, we only marginalize over the learning rate and the dropout regularization values. We delay further improvement on hyperparameters to upcoming studies as we want to highlight the proof of concept of predicting $t_{\rm reion}$ maps in this study.

Finally, we use the usual ReLU activation function in every layer except in the last one. We use the Linear activation function in

[2]https://github.com/autonomio/talos

Table A1. Details of the architecture the auto-encoder convolutional neural network used to predict maps of the cosmic reionization time $t_{\rm reion}$. Each row shows the properties of a given layer in the encoder or the decoder. The different columns show different properties of the corresponding layers. The layers of the encoder are applied both to the input composed of the gas density and stellar field (see Fig. 2 and Section 3.3 for explanations). The outputs of the encoder are averaged before entering the layers composing the decoder.

Network branch	Layer #/step name	Number of filters/data	Filter size/data dimension	Activation function
	Input	3000	128×128	–
	1	32	3×3	Relu
Encoder	2	64	3×3	Relu
	3	128	3×3	Relu
	4	128	3×3	Relu
	1	128	3×3	Relu
	2	128	3×3	Relu
Decoder	3	64	3×3	Relu
	4	32	3×3	Linear
	Output	3000	128×128	–

the last layer because we want to predict continuous values as an output of the network instead of discrete ones. This is different from common neural networks that aim to predict discrete values for classification problems.

APPENDIX B: $t_{\rm REION}$ CUBE RECONSTRUCTION

Here, we describe our procedure to reconstruct a complete 3D $t_{\rm reion}$ cube with our network. Our neural network predicts 2D slices of size 128×128 cells from maps of the gas density and the stellar density fields with the same size. However, we choose to only keep the central submap of size 64×64 cells from a complete 128×128 prediction. This procedure ensures we do not miss sources nearby that are just outside the slice we are trying to predict. Therefore, we need to make 16 predictions to reconstruct the whole 256×256 slice for the simulations studied here.

To reconstruct the whole 256^3 cube, we repeat the 2D reconstruction of a slice 256 times. Since the stellar and density fields used as inputs of the neural network are smoothed along the transverse direction to the plane we aim to predict, we reconstruct the cube by piling up reconstructed slices along this particular direction. However, in practice, it generates spurious grid artefacts when looking at slices taken along directions different from this direction. Therefore, instead of reconstructing the cube along only one particular direction, we reconstruct three different cubes along the three main directions. Then, we take, for each cell, the minimum and maximum values of $t_{\rm reion}$ from these three cubes. We then take the average of these two values to get our final 3D reconstruction. Such a procedure has the advantage of eliminating the grid artefacts during the reconstruction procedure.

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