Research in Astronomy and Astrophysics, 24:115020 (18pp), 2024 November https://doi.org/10.1088/1674-4527/ad8b0e

Using Cartesian Slice Plots of a Cosmological Simulation as Input of a Convolutional Neural Network*

Guillermo Arreaga-García
Departamento de Investigación en Física, Universidad de Sonora., Apdo. Postal 14740, C.P. 83000, Hermosillo, Sonora, Mexico; guillermo.arreaga@unison.mx
Received 2024 June 8; revised 2024 September 20; accepted 2024 October 23; published 2024 November 26

Abstract

Using a uniform partitioning of cubic cells, we cover the total volume of a ΛCDM cosmological simulation based on particles. We define a visualization cell as a spatial extension of the cubic cell, so that we collect all simulation particles contained in this visualization cell to create a series of Cartesian plots in which the overdensity of matter is clearly visible. We then use these plots as input to a convolutional neural network (CNN) based on the Keras library and TensorFlow for image classification. To assign a class to each plot, we approximate the Hessian of the gravitational potential in the center of the cubic cells. Each selected cubic cell is then assigned a label of 1, 2 or 3, depending on the number of positive eigenvalues obtained for the Householder reduction of the Hessian matrix. We apply the CNN to several models, including two models with different visualization volumes, one with a cell size of type L (large) and the other with a cell type S (small). A third model combines the plots of the previous L and S cell types. So far, we have mainly considered a slice parallel to the *XY* plane to make the plots. The last model is considered based on visualizations of cells that also include slices parallel to the *ZX* and *ZY* planes. We find that the accuracy in classification plots is acceptable, and the ability of the models to predict the class works well. These results allow us to demonstrate the aim of this paper, namely that the usual Cartesian plots contain enough information to identify the observed structures of the cosmic web.

Key words: methods: numerical – cosmology: theory – (cosmology:) large-scale structure of universe

1. Introduction

With the advent of artificial intelligence and machine learning, a convolutional neural network (CNN) can recognize complex patterns in plots. For example, the simple MNIST-CNN described by Chollet (2018) recognizes a handwritten number from a data set of thousands of images. Among the many CNNs available, see Szegedy & Liu (2015), Krizhevsky et al. (2017) and He et al. (2015), we consider in particular the CNNs described by Bagnato (2023): one CNN is trained to recognize nine classes of sports from a data set of thousands of sport images and the other CNN is trained to classify images of dogs and cats from a data set of thousands of dog and cat images.

In this work, we train the image classification CNNs described by Bagnato (2023) with a data set of rendered two-dimensional (2D) plots commonly presented in particle-based numerical simulations to show the simulation results. These plots are mainly 2D Cartesian colored isodensity plots for a slice of particles from the simulation volume. Particularly in the early simulations, the comparison of the results of particle-based simulations with observations was usually done using plots of a Cartesian slice through the distribution of simulation particles, usually in two dimensions (plots in three dimensions were also produced, but less frequently as they became increasingly difficult to produce). It should be noted that an essentially subjective visual inspection of these 2D plots was used to perform morphology comparisons of matter structure between different simulation models; see for example, Weinberg & Cole (1992).

In supervised machine learning, the data set of plots must be supplemented by a data set of classes, i.e., each plot must have a class, which is simply a label. One could choose an attribute of interest to determine the class of a plot. In the case of the CNN described by Bagnato (2023), there are nine classes for the first CNN and only two classes for the second CNN, namely dog or cat, so each image must be labeled as dog or cat. After the training process, these CNNs can successfully distinguish the type of sport or whether the given image corresponds to a cat or a dog.

To give a physical content to the data set of the plots of this paper, we consider Hahn et al. (2007), who performed *N*-body particle-based cosmological simulations and considered only dark matter (DM) halos with at least 300 simulation particles in the snapshot at a redshift $z =$ Hahn et al. (2007) count the number of positive eigenvalues of the Hessian matrix, which is defined as $T_{ij} = \frac{\partial^2 \Phi}{\partial x_i \, \partial x_j}$, where Φ is the gravitational potential.

* Departamento de Investigación en Física, Universidad de Sonora.

The number of positive eigenvalues can be 0, 1, 2 or 3 and corresponds to a void, a sheet, a filament and a cluster respectively, which represent the most common distribution of matter in the cosmological simulation.

To follow the method described by Hahn et al. (2007), we define a uniform partition of cubic cells covering the total volume of a particle-based ΛCDM cosmological simulation, and determine the eigenvalues of the Hessian of the gravitational potential. We then count the number of positive eigenvalues in each cubic cell and use this number as the label for the plots constructed for that cell.

If one can successfully train the CNN with this kind of data set, what would be the prediction that the CNN could make? In this case, it would be possible to recognize and distinguish between filaments, walls and clusters. The aim of this paper is to show that the usual Cartesian plots contain enough information to identify the observed structures of the cosmic web. To be sure that this is the case, we need to make calculations. Let us first put the work of Hahn et al. (2007) in context to illustrate the importance of this problem.

The investigation of the spatial distribution of matter in the universe on scales of a few Mpc dates back to the 1930s; see Shapley & Ames (1932). Later, using galaxy catalogs, it was discovered that on these scales there were filaments, voids and sheets in which the galaxies were housed; see Geller & Huchra (1991). With the advent of numerical simulations, it became clear that the large-scale structure of the universe consists of a network of DM that is interconnected in complex ways and was aptly named the cosmic web; see Bond et al. (1996).

The quantitative analysis of the large-scale distribution of matter has a long and interesting history. For example, Shandarin & Zeldovich (1983) tried to find patterns in the distribution of galaxies, and defined a function $B(r)$ to represent the average number of galaxies within a sphere of radius r. They obtained a percolation radius that depended on the distribution of galaxies in the sample cube. From this result, they were able to deduce that in the universe the galaxies are distributed according to a network structure. Shortly afterwards, Barrow et al. (1895) proposed a graphical algorithm, the so-called minimal spanning tree (MST) to determine intrinsic patterns in point data sets. This algorithm was applied to two and three-dimensional (3D) data sets to compare the distributions of real galaxy catalogs with random distributions.

Gott et al. (1986) constructed density maps to investigate the relationships between the high and low-density regions of a smoothed distribution of galaxies from a galaxy catalog. Finally, they presented a model that assumes a sponge-like structure of the universe. At that time, the distribution of galaxies was also modeled as a homogeneous fractal, and several attempts were made to calculate its Hausdorff dimension, see Martinez et al. (1990). Babul & Starkman (1992) introduced the so-called structure functions to measure certain geometric properties, such as sphericity, prolateness and oblateness. With the help of these structure functions, they began to quantify the morphology of the matter distribution through simulations.

We should also mention some recent papers devoted to the study of the cosmic web. Libeskind et al. (2018) presents a very comprehensive overview of the various methods that have been developed to classify and identify the different matter elements of the cosmic web, such as: (i) graph and percolation techniques (see Shandarin & Zeldovich 1983 and Alpaslan et al. 2014) using a method known as the adapted MST; (ii) stochastic methods (see Tempel et al. 2014; Leclercq et al. 2015 and Tempel et al. 2016) based on the Bisous method; (iii) geometric, Hessian-based methods (see Aragon-Calvo et al. 2007 and Aragon-Calvo et al. 2007; Hahn et al. 2007 presented a multi-scale morphology filter to recognize three different structural configurations of the cosmic web, namely blobs, walls and filaments); (iv) scale-space multi-scale Hessian-based methods (see Cautun et al. 2013 with the NEXUS+ method); (v) topological methods (see Colombi et al. 2000 and Aragon-Calvo et al. 2010 with the Spineweb method and Sousbi et al. 2011 with the DisPerSE method); and (vi) phase-space methods (see Shandarin 2011 and Ramachandra & Shandarin 2015 with the multi-stream web analysis method, and Falck et al. 2012 and Falck et al. 2014 with the ORIGAMI method).

The β-skeleton method has been used successfully in various areas of pattern recognition. In particular, this method can be used to reconstruct images from partial data or from data considering only the contours. It is based on an algorithm that makes it possible to associate a graphical structure with a set of points in a discrete data set. The data can be in three or two spatial dimensions. This graphical structure determines the connectivity of the points in the data set and depends on the value of a parameter called β. Fang et al. (2019) applied this method to the large-scale structure of the universe and succeeded in recognizing the filaments of the cosmic web; see also Garcia-Alvarado et al. (2020).

Hoffman et al. (2012) and Forero-Romero et al. (2009) introduced a threshold eigenvalue, so that the counting of positive eigenvalues proposed by Hahn et al. (2007) is extended to counting the number of eigenvalues above a certain threshold eigenvalue to determine the class of the given halo. Aragon-Calvo (2019) presented a cosmic web classification using CNNs with a U-net architecture, previously used in the classification of 2D medical images and 3D data cubes of medical interest. The density function is smoothed from the discrete distribution of the simulation particles to obtain a continuous density field based on the second-order local variations of the density field encoded in the Hessian matrix. A very important step is the determination of the threshold eigenvalue. Aragon-Calvo (2019) proposed an automated algorithm, but the best results are obtained by visual inspection of the algorithm results.

Recently, Inoue et al. (2022) utilized a CNN fed with plots of the distribution of galaxies and particles from the IllustrisTNG

simulations. They used the velocity gradient tensor to obtain the eigenvalues above a threshold to yield the classification scheme. They also considered a set of four labels: voids, sheets (also called walls), filaments and knots (also called clusters). Their models can classify simulated galaxies with an accuracy (macro-averaged f1-score) of 64%. To obtain the cosmic structure classification, they used a coverage of 256^3 cells and considered only DM particles.

Considering the papers of Aragon-Calvo (2019) and Inoue et al. (2022), the calculations proposed in this paper are not new. And after mentioning some of the many existing methods for classifying the elements of the cosmic web, we emphasize that our method does not provide any element that improves these methods. The novelty of this work is that we show that the plots commonly used to see the results of a cosmological simulation can also be used to classify the elements of the cosmic web network using a CNN.

The structure of this paper is as follows. In Section 2.1 we describe the cosmological model. In Section 2.2 we provide some details about the simulation. In Section 2.3 we describe the coverage of the cubic cells and the method used to approximate the Hessian in each selected cubic cell. In Appendix A we complement Section 2.3. In Section 2.3.1 we present a consistency test of the calculation described above. In Section 2.3.2 we present a characterization of the matter content of the cells. In Section 2.4 we describe the development of the sets of training and validation plots for two models with different visualization volumes. In Section 2.5 we describe the CNN and in Appendix B we present an alternative CNN. In Section 3 we show the results, which include some reports on the training and validation analysis (in Section 3.1) and a report on the prediction ability using a confusion matrix analysis (in Section 3.2) for each CNN model. Finally, in Sections 4 and 5, we discuss the relevance of our results with respect to the results of previous papers and make some concluding remarks.

2. The Physical System and Computational Considerations

2.1. The Simulation

We consider a cubic box with a side length given by $L = 75$ Mpc. The initial content of matter is characterized by $\Omega_m = 0.2726$ and the content of dark energy is $\Omega_\Lambda = 0.7274$, so that $\Omega_m + \Omega_\Lambda = 1.0$, corresponding to a flat model of the universe, expanding with a Hubble parameter $H_0 = 100\,h$ km s^{-1} Mpc^{-1}. h is given by $h = 0.704$. The baryon mass is characterized by $\Omega_b = 0.0456$. These values have been chosen from Planck Collaboration (2013, 2016).

The density perturbations of this cosmological model have been generated with the publicly available code N-GenIC[1], so that an initial power spectrum $P(k)$ can be constructed by moving the simulation particles according to the linear spectrum defined by Eisenstein Hu (1999). The power spectrum normalization was fixed at a value of $\sigma_8 = 0.9$.

It should be noted that for each DM particle there is a gas particle. Thus, the number of DM particles $N_{\rm DM}$ and the number of gas (G) particles $N_{\rm G}$ are both equal to 11,239,424. Therefore, the particles have masses given by $m_{\rm DM} = 2.3 \times 10^9 M_\odot$ and $m_{\rm G} = 4.7 \times 10^8 M_\odot$, respectively. The average mass density is given by $\rho_0 = 5.12 \times 10^{-30}$ g cm^{-3} and the initial and final redshifts are fixed at $z = 127$ and $z = 0$, respectively.

2.2. The Evolution Code

The simulations used in this paper are evolved with the particle-based code Gadget2; see Springel (2005). Gadget2 implements the Monaghan–Balsara form of the artificial viscosity (Balsara 1995), so that the strength of the viscosity is regulated by setting the parameter $\alpha_\nu = 0.75$ and the parameter $\beta_\nu = \frac{1}{2} \times \alpha_\nu$. The Courant factor has been fixed at 0.1.

The time evolution of the simulation up to $z = 0$ required a little more than 80 hr; it ran on 60 processors in an Intel Xeon E5-2680 v3 cluster at 2.5 GHz at Laboratorio Nacional de Supercomputo (LNS) of the Benemerita Universidad Autonoma de Puebla (BUAP). The computational method, described in Section 2.3, the results of which are described in Section 3, was applied only to the last output, that is, the snapshot at redshift $z = 0$.

2.3. The Cubic Cells and the Hessian

With the aim of investigating the grouping properties of protogalaxy clusters, Arreaga-García (2021) used partitions with two sizes, with 64^3 and 128^3 cubic elements, to search for matter structures that had already gravitationally collapsed. We chose the first partition, so that the total number of identical cubic cells into which the entire simulation volume is divided is therefore $64^3 \equiv 262{,}144$. It should be noted that the side length of each cubic cell is $2^*\Delta_H = 1.17$ Mpc where $\Delta_H = 0.585$ Mpc.

A very small cell size would allow us to detect only the core of a protogalaxy, and almost all cores would be isolated, while a very large cell size would allow us to detect many structures within the cell, almost all of which would be connected, but it would be difficult to label the cell. Remember that we are trying to label each cubic cell according to the type of matter it contains. With this objective in mind, the size chosen for the cubic cell can be physically justified as follows.

In the percolation technique, this idea of the connectivity of protogalaxies can be quantified by the percolation radius (Zeldovich et al. 1982). In general, the connectivity of galaxies decreases with increasing percolation radius, a similar behavior

[1] https://www.h-its.org/2014/11/05/ngenic-code/

to the two-point correlation function, where the test radius r_0 is about 5 Mpc.

Let us consider the distribution of galaxies around the Virgo cluster. In the percolation technique, a suitable radius for modeling the Virgo cluster using a cylinder would be 2.2 Mpc, for that for Perseus 2.9 Mpc, and that for Coma 2.5 Mpc. Therefore, a radius of 2 Mpc can be assumed to cover this type of distribution of galaxies with a cylinder model. This model allows us to consider a chain of galaxies that are part of a single matter structure. In this case, we can in principle assign a label to a grid cell containing this type of connected matter structure.

To this physical justification for the size of the cubic cell we can also add a technical justification. The resolution of the simulation described in Section 2.1 is sufficient to see the matter structures considered above, i.e., protogalaxy clusters. Furthermore, the plots generated for each selected cubic cell must have enough features to be useful as input for the neural network. For this reason, the cell size must not be too small for matter structures to be recognized, but at the same time, the cell size must not be too large to obtain a label. Therefore, a test cell size in the range 1–7 Mpc can be used, and plots are then created to assess whether they are useful. Due to this indeterminacy of the cell size, we use two cell sizes in this study.

We continue with the description of the following steps. We determine the number of simulation particles (including DM and gas particles) that are in each cubic cell. If the actual cell has more particles than a certain threshold number N_t, then we approximate the Hessian for that cell as follows. We need to calculate $T_{ij} = \frac{\partial^2 \Phi}{\partial x\, \partial y}$, where Φ is the gravitational potential. We use the particle approximation $\Phi(\boldsymbol{r}) = \Sigma \Phi(\boldsymbol{r}_i)\, W(\boldsymbol{r} - \boldsymbol{r}_i)$, where W is a window function. The second derivatives are approximated by $f''(x_0) = \frac{f(x_0+h)+f(x_0-h)-2f(x_0)}{h^2} + \frac{h^2 f^4(\zeta)}{12}$ for $\zeta \in (x_0 - h, x_0 + h)$, where h is a small increment, here assumed to be $h = \Delta_H/100$.

We then use our own code to calculate all components of the symmetric tensor T_{ij} in this particle approximation. The subroutine tred2.c is used to reduce the matrix T_{ij} to a diagonal form, and the subroutine tqli.c is applied to find the eigenvectors and eigenvalues of the matrix T_{ij}. Both subroutines were taken from Flannery et al. (1998).

In this approach, the particles located in the neighboring cells are not taken into account when calculating the Hessian for a given cell. Therefore, we do not quantify the effects of including neighboring cells in the Hessian diagonalization. This calculation was performed in this way to simplify the process. However, we have performed a comparison of the results obtained when calculating the Hessian with two differently sized cells as can be seen in Appendix A.

This code runs on all cubic cells, so we get the center of the selected cells, the number of particles contained in each selected cell and the number of positive eigenvalues of the Hessian. We mentioned in Section 1 that we only considered the cells whose number of simulation particles is above a certain threshold number. We need to note the importance of the threshold number of particles as a free parameter. For example, using $N_t = 15$ the total number of cells was 77,367, with the following distribution: there were 59,544 cells with three positive eigenvalues; there were 7462 cells with two positive eigenvalues and there were 10,361 cells with only one positive eigenvalue.

If you increase this parameter to $N_t = 150$ particles, these numbers change to 14,515, 145 and 133 cells, for three, two and one positive eigenvalue, respectively. In both cases, the number of class 3 cells is much larger than the number of class 1 and 2 cells. In this paper, we have set the threshold for the number of particles to 150.

By setting a minimum number of particles per cell, we can ensure that the plot of each cell has sufficient visual content for the analysis to be meaningful. This minimum limit for the number of particles is equivalent to setting a minimum limit for the density of the cells. For example, when a plot of a simulation box is created, only the particles whose density is greater than a certain limit are selected to facilitate the visualization of the dense matter of a simulation.

Finally, we order these data sets in terms of the number of simulation particles contained in the cell, from the highest to the lowest value to generate training plots with the best possible resolution, that is, with more simulation particles, as we will explain in Section 2.4. The cells with only a few particles generated plots with low resolution and were therefore discarded.

2.3.1. Consistency Tests

As a consistency test for this code, the halo finder Rockstar described by Behroozi et al. (2013) was used to locate the halos of matter determined by constructing maximum isodensity curves. We placed cells in the halo centers located by Rockstar and then executed the code described in Section 2.3. We found that in this test, the output of the code indicates that all cells are of class 3, i.e., they have three positive eigenvalues. So, if you use the overdensity centers obtained by the halo finder code Rockstar, our code will not find any class 1 or 2 cells, as expected. In Section 4, we will compare our results with those obtained with the code described by Forero-Romero et al. (2009).

In addition, in Figure 1 we show the distribution of some simulation particles extracted from a section of the simulation box as well as the cells with matter structures classified as filaments, walls or clusters. It should be emphasized that this figure only shows simulation particles. All particles that are located in a grid cell and have been assigned a class label are collected in a file, and colored according to their class, as shown at the top right of each plot in Figure 1.

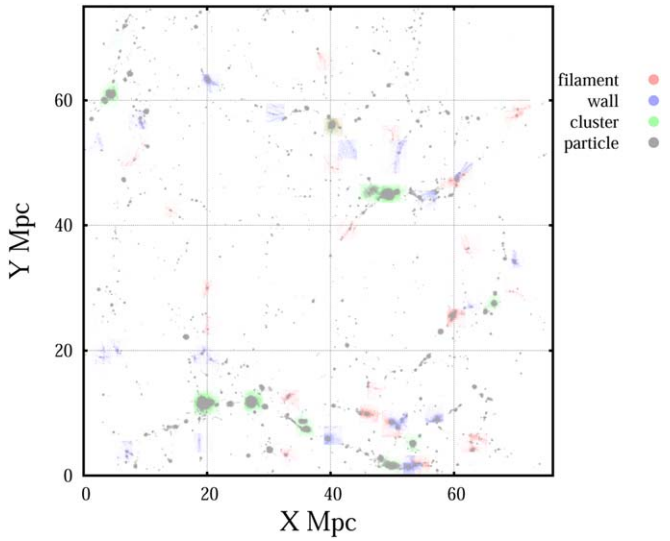

Figure 1. This figure shows a comparison between the spatial distribution of the three classes of matter structure and the simulation particles located in a slice. The plot is a 2D projection of a 3D slice of the simulation box, with a width of 9 Mpc in the coordinate perpendicular to the plane indicated on the axes of each panel. The height of the middle plane is located at 37.5 Mpc in the perpendicular coordinate.

These particles are superimposed on the simulation particles so that we can search for matches. It can be seen that there is a good match for clusters (class 3). As expected, the identification of filaments and walls by the human eye is not easy. Therefore, we are investigating whether a CNN can perform this classification task across the entire simulation volume.

2.3.2. The Distribution of Cells in Terms of the Mass

To conclude Section 2.3, we present the distribution of cubic cells in terms of the mass they contain. As we mentioned above, we have considered the gas and DM components when calculating the Hessian. Now we consider these components separately to investigate their distribution in the structure classes. First, in the left panel of Figure 2 we calculate the fraction of cells with a mass greater than or equal to the given mass indicated on the horizontal axis of the panel, regardless of the class of the cell. There are three curves in this panel: one is labeled "c," which means that all the matter contained in the cubic cell has been taken into account. For the curve labeled "cg," only the gas contained in the cell has been considered and analogously, for the curve labeled "cdm," only the DM contained in the cell has been taken into account. As expected, we see that the DM dominates the matter content of the cells.

In the right panel of Figure 2 we show the distribution of the cells and the classes of the cells. The curves are labeled g1 (dm1), which refer to gas (DM) in the cells of class 1 (with only one positive eigenvalue), and g2 (dm2) and g3 (dm3), which analogously refer to gas and DM in cells with two and three positive eigenvalues, respectively. In this panel we see that there are two groups of curves, one group for the gas component and the other group for the DM component.

It should be noted that there is a threshold for the number of particles that must be reached for a cell to be selected; see Section 2.3. For this reason, most curves in each group coincide at small masses and are separated at large masses. It can be seen that the structures in class 3 (the clusters) have the largest masses in both groups, i.e., for gas and DM components, as reported by Ganeshaiah Veena et al. (2018). However, in the upper-left corner of each group of curves, it can also be seen that the smallest masses are also detected for class 3 cells. The curves for class 2 and class 3 cells agree in each group for small masses, but are slightly separated for large masses.

2.4. Training Plots and the CNN Models

In this section we consider the visualization cells as follows. We take the coordinates of each cell center x_c, $y_c z_c$ and determine all simulation particles whose coordinates are within the interval $x \in [x_c - \Delta_L, x_c + \Delta_L]$ where Δ_L is a width, that we will define below and analogous relations for the y and z coordinates are not shown. All these simulation particles are considered and plots are created for this visualization volume, which has a side length of $2 * \Delta_{L_L} = 3.0$, where $\Delta_{L_L} = 1.5$ Mpc. This visualization cell is called an L cell.

In Section 2.3 we mentioned that to approximate the Hessian, we considered all particles in the cubic cell H with a side length of $2 * \Delta_H = 1.17$ Mpc, so that the visualization volume is larger than the volume of the Hessian. This must be so because we expect to capture extended objects such as filaments and walls, which can be better sampled by a wider grid.

We also consider a smaller visualization volume where the side length is $2 * \Delta_{L_S} = 1.5$ and $\Delta_{L_S} = 0.75$ Mpc. This visualization cell is called an S cell. To distinguish these models, we use the labels "L" and "S" respectively, as shown in Table 1. In Section 3 we present the results of these runs with different sizes for the visualization volume.

To create a plot, we change the origin of the coordinates to the center of the cell so that the plots are normalized with respect to the side length Δ_L. In Figure 3 we show examples of class 1 (in the left panel) and class 2 (in the right panel) matter structures and compare the size of the three cell types. To summarize, we use a cubic cell of type H to compute the number of positive eigenvalues of the Hessian, which is shown as the smallest case in Figure 3. Visualizations of cells of type S and type L are also shown.

As already mentioned in Section 1, due to the 3D nature of the cosmic web, it is almost impossible to distinguish the different matter structures using only a 2D Cartesian projection. For example, it would not be possible to distinguish between a

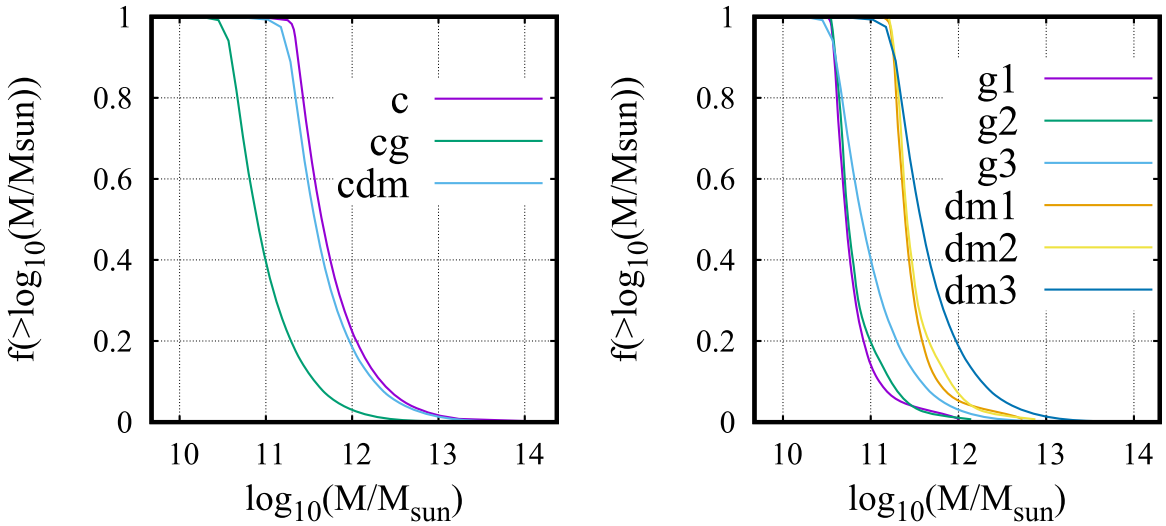

Figure 2. Distribution of cubic cells vs. the mass that they contain. We show on the vertical axis the fraction of cells with a mass greater than (or equal) to the given mass on the horizontal axis. We take into account both components, the gas and DM, irrespective of the number of eigenvalues of each cell (left panel). We also consider separately the gas and DM in the cells according to the number of eigenvalues of each cell (right panel). The classes here are referred to as 1, 2 and 3. The curves in the right panel are labeled as g1 (dm1), which refers to gas (DM) in the cells of class 1 (with only one positive eigenvalue); and g2 (dm2) and g3 (dm3), which analogously refer to gas and DM in cells with two and three positive eigenvalues, respectively.

Table 1
The Number of Images and Some Results of the CNN Models

Row		Model L	Model S	Model LS	Model LT
1	Class 1	240	360	610	600
2	Class 2	240	360	662	600
3	Class 3	217	347	544	300
4	training plots	557	853	1452	1200
5	validation plots	140	214	364	300
6	prediction plots	90	90	180	180
7	total plots	697	1067	1816	1500
8	total parameters	55948771	69973475	69973475	69973475
9	memory	214 MB	266 MB	266 MB	266
10	correct labels	100	175	256	201
11	incorrect labels	40	39	108	99
12	test accuracy	0.71	0.81	0.70	0.67
13	test loss	0.49	0.48	0.58	0.63
14	true positives	0.23, 0.16, 0.3	0.26, 0.1, 0.33	0.11, 0.11, 0.16	0.28, 0.1, 0.18
15	Δ_L	1.5 Mpc	0.85 Mpc	1.5 Mpc and 0.85 Mpc	1.5 Mpc

filament and the projection of a sheet. For this reason, we have created an additional model in which the set of plots fed to the neural network consist of three projections onto the Cartesian axes; i.e., for each cubic cell we create plots of the *XY*, *ZX* and *ZY* projections. For this model we use the label "LT," as can be seen in Table 1.

In Figure 4 we show examples of the training plots. In the left column of Figure 4 we show the *XY* (top row), *ZX* (second row) and *ZY* (bottom row) projections of a matter structure that could be identified as a filament (with only one positive eigenvalue). In the middle column, we show (from top to bottom) the *XY*, *ZX* and *ZY* projections of an object that could be identified as a wall (with two positive eigenvalues). In the right column, we show the projections of an object that could be identified as a cluster (with three positive eigenvalues). We emphasize that the columns in Figure 4 are not connected, but represent different matter structures.

The plots were created with the same Python code, based on the plot library Matplotlib. The basic features are the same for all plots, for example, they have a resolution of 100 × 100 dots per inch (dpi). The plots for model L have a width of 578 px and a height of 434 px, while the plots for model S have a

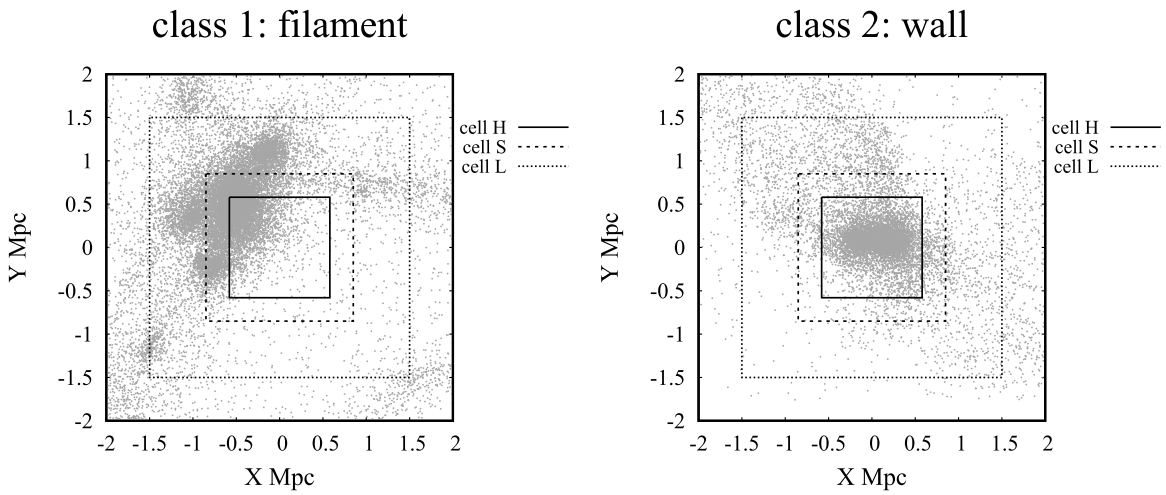

Figure 3. We show two examples of matter structures, in the left panel of class 1 (filament) and the right panel of class 2 (wall). We do not show an example of class 3. The different numbers of simulation particles that are captured by the two different visualization cells (types L and S) can be compared with the size of the calculation cell (type H).

width of 640 px and a height of 480 px. For both model L and model S, the original number of plots was the same, as we mentioned in Section 2.3.

However, for each plot, we created several additional plots by rotating the original plot, with the rotation angle randomly determined with respect to the coordinate center of each cubic cell. In this way, we increased the number of plots available for the training and validation sets. It should be noted that the number of rotated plots per original plot is different for each model. For this reason, the number of plots is different, as can be seen in Table 1.

As we mentioned in Section 2.3, the cell catalog consists of 14,515 cells in class 3, 145 cells in class 2 and finally 133 cells in class 1. We can create a plot and rotations of this plot from each cell catalog. It must be emphasized that we will not use all the cells in class 3, in order to avoid an imbalance in the number of samples with respect to the classes. For this reason, we have only considered a subset of class 3 so that the number of plot samples is similar for all classes; see Table 1 and Section 3. In Figure 5 we show a schematic diagram of the generation process of the data set.

To take advantage of Table 1, we have included some numbers that we will clarify in Section 2.5 and in Section 3, where we will present the CNN used to classify the plots and the results.[2]

2.5. The CNN

The CNN design of this paper was motivated by many successful models, including, among others, the simple MNIST convnet (Chollet 2018); in particular, the CNN for classifying sports described by Bagnato (2023) and the CNN described by Bagnato (2023) for classifying dogs and cats. In Table 2 we describe the CNN model considered in this paper.

In Figure 6 we show a schematic representation of the CNN with the main layers and their effects on the size of the plot (in pixels). In general, a CNN has two main stages: one for feature extraction and the other for classification. In the extraction stage, the input plot is passed to a convolutional layer to create a feature map of the plot, the size of which is then reduced by a pooling process. This step can be applied several times to the same input plot.

In the classification phase, the feature matrix is converted into a vector and the values of the parameters available in the CNN are determined so that the results match the class of the plot. To compare the performance and quality of the results of the CNN described in Section 2.5, we consider an alternative CNN in Appendix B. However, other models can also be tested to explore different CNN options.

It should be noted that in both CNNs, the densest layers are those that offer the largest number of adjustable parameters. In the alternative CNN shown in Appendix B, only the dense layer generates parameters. The other layers have other goals, such as creating a character map, or reducing the number of pixels, but the dense layers must establish the connection between all the neurons of the previous layer and the next layer, which is why they must have a parameter for each connection.

3. Results

We apply the CNN described in Section 2.5 to several sets of plots. The CNN is trained during different epochs for all models to determine the best values of the free parameters to assign each

[2] Rows 10–13 show the results generated during the code execution that used the validation data. Line 14 shows the results generated with the prediction data

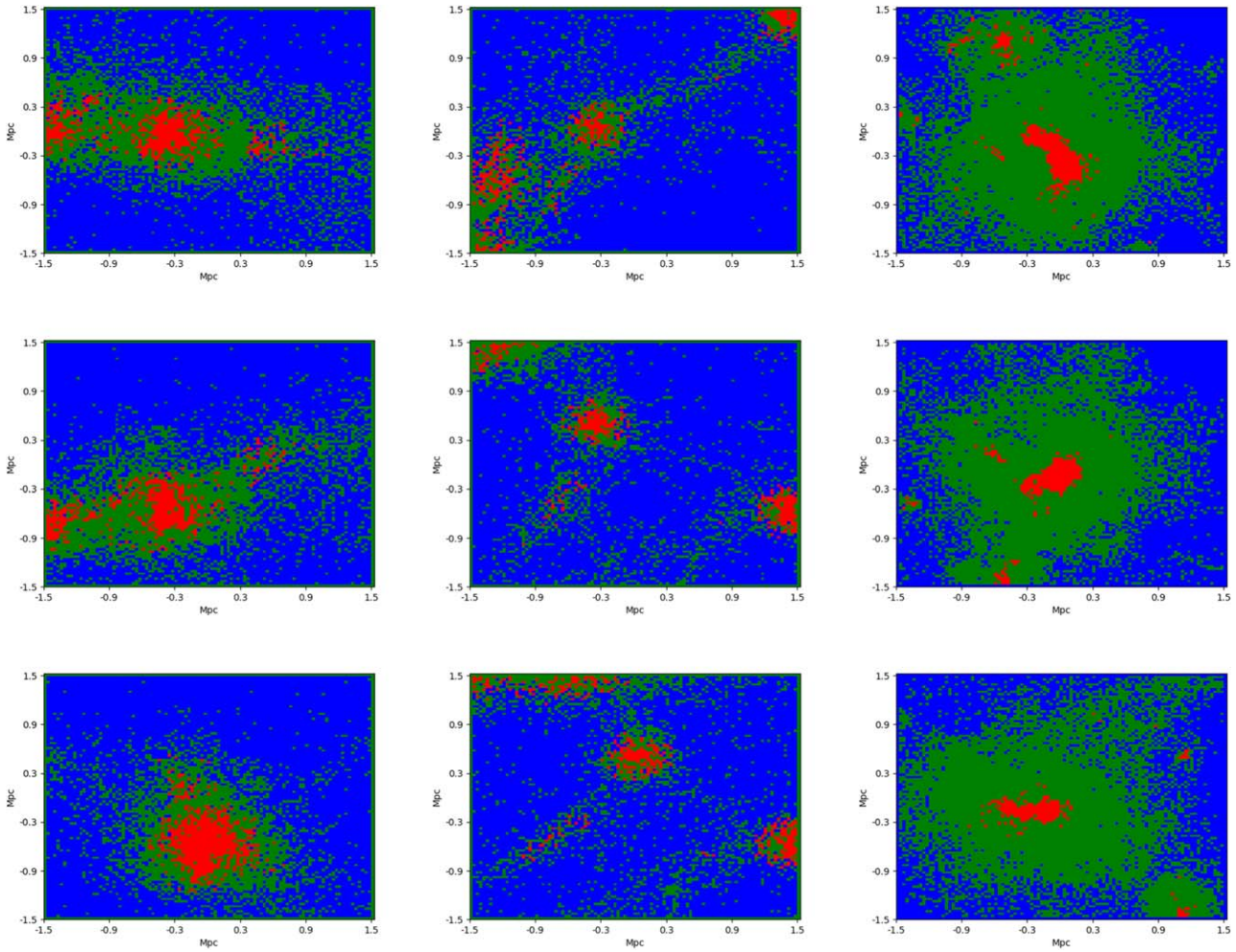

Figure 4. Examples of isodensity plots that were generated for cells with different numbers of positive eigenvalues. The vertical and horizontal axes of these plots indicate the length in Mpc. For model L, with a width of 578 px and a height of 434 px, the axes vary from −1.5 to 1.5, and we show the projections on the *XY* axes (top line), the *ZX* axes (second line from top to bottom), and the *ZY* axes (third line from top to bottom). For model S, with a width of 640 px and a height of 480 px, the *XY* axes vary from −0.85 to 0.85 (not shown in this figure). Plots are shown with only one positive eigenvalue (left column), two positive eigenvalues (middle column) and three positive eigenvalues (right column).

plot to the target class. The accuracy and loss metrics are two of the best known metrics for determining the performance of a CNN. Accuracy is expressed as a percentage, i.e., it indicates the proportion of plots whose classes were correctly predicted by the CNN. The loss metric, also known as the cost or error function, measures the failure in predicting the correct class for a plot with respect to the target class and is expressed as a real number.

3.1. Report on Training and Validation

In Figure 7 we show the accuracy and loss metrics in relation to the epochs. To determine the accuracy and loss metrics, the code uses the validation plots. The curves of all models show the expected behavior: the accuracy and validation curves for the training plots increase over the epochs. The same observation can be made for the loss curves, i.e., the loss curves for the training and validation plots decrease as the number of epochs increases.

To better examine the performance of the CNN, we need to go beyond accuracy and loss metrics. The f1-score is another metric that can be used in addition to the two simpler performance metrics mentioned above, namely the accuracy and loss. To explain the f1-score metric, we first introduce the concepts of precision and recall. Precision is defined as the ratio between the number of correct positive predictions and the sum of the correct positive predictions and the number of false positive predictions, i.e., precision measures how many of the CNN's positive predictions were correct. Recall is defined as the ratio between the number of true positives and the sum of the number of true positives and the number of false negatives, i.e., recall measures how many positive predictions the CNN found out of all positives.

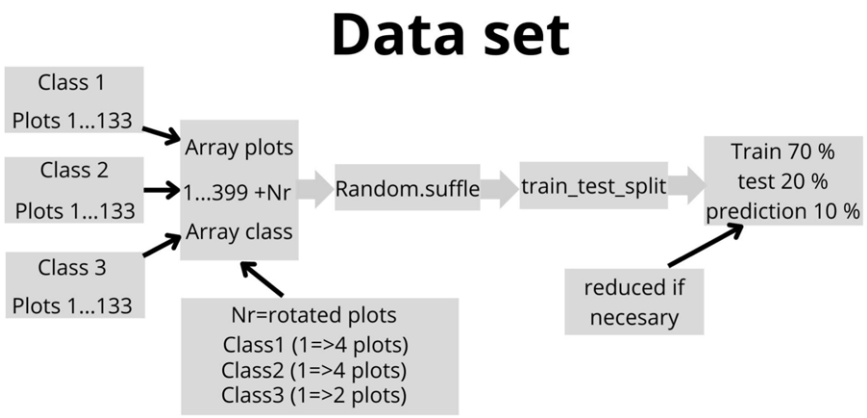

Figure 5. Schematic diagram of the generation process of the data set.

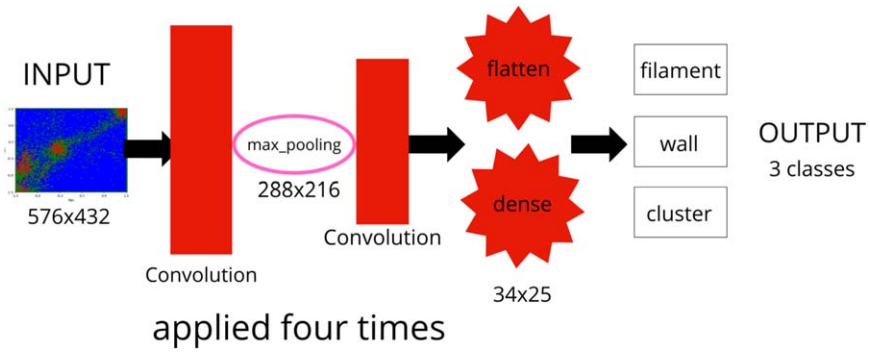

Figure 6. Schematic diagram of the CNN.

Table 2
Layers of the CNN, Model Sequential-5

Layer (Type)	Output Shape	Number of Parameters
$conv2d_{14}$ (Conv2D)	(None, 576, 432, 32)	1184
$max\text{-}pooling2d_{14}$ (MaxPooling2D)	(None, 288, 216, 32)	0
$conv2d_{15}$ (Conv2D)	(None, 286, 214, 64)	18496
$max\text{-}pooling2d_{15}$ (MaxPooling2D)	(None, 143, 107, 64)	0
$conv2d_{16}$ (Conv2D)	(None, 141, 105, 128)	73856
$max\text{-}pooling2d_{16}$ (MaxPooling2D)	(None, 70, 52, 128)	0
$conv2d_{17}$ (Conv2D)	(None, 68, 50, 128)	147584
$max\text{-}pooling2d_{17}$ (MaxPooling2D)	(None, 34, 25, 128)	0
$flatten_5$ (Flatten)	(None, 108800)	0
$dense_{10}$ (Dense)	(None, 512)	55706112
$dense_{11}$ (Dense)	(None, 3)	1539

The f1-score combines the metric precision and recall into a single metric. The f1-score is defined as the ratio between 2 times the precision times the recall and the sum of precision and recall, i.e., the f1-score is an average of precision and recall; see Tables 3–6. The macro-average values for precision and recall are calculated as the average of the individual precision and recall values of the individual classes.

For model L, we can see that for classes 1 and 2, which identify filaments and walls, the CNN recognizes these classes with an f1-scores of only 0.53 and 0.54, respectively; see

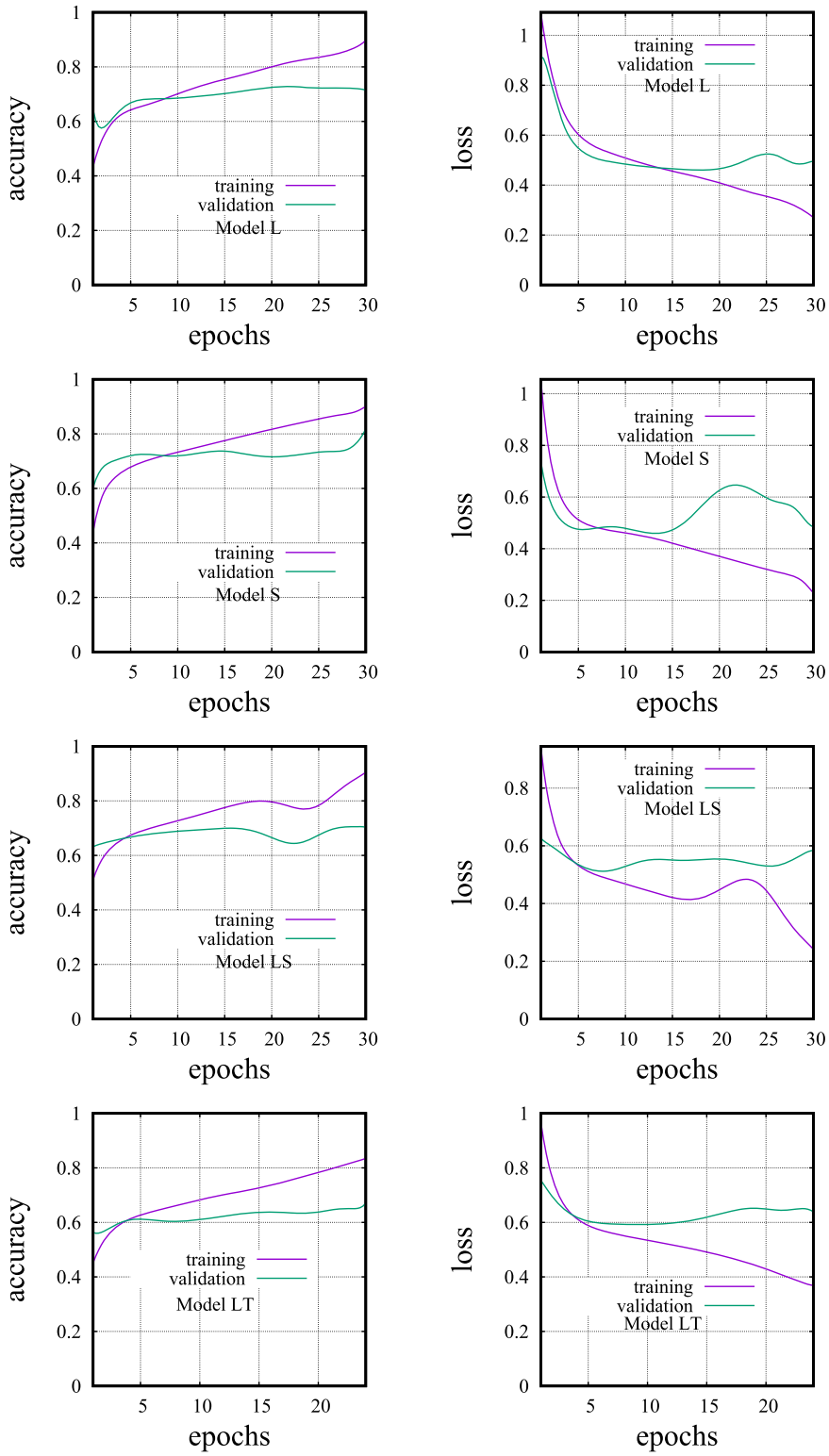

Figure 7. Accuracy and loss history in terms of the epochs, for model L (top line); model S (second line from top), model LS (third line from top) and model LT (bottom line).

Table 3
Report for Model L

⋯	Precision	Recall	f1-score	Support
Class 1	0.56	0.50	0.53	44
Class 2	0.52	0.56	0.54	41
Class 3	0.96	1.0	0.98	55
accuracy	⋯	⋯	0.71	140
macro avg	0.68	0.69	0.68	140

Table 4
Report for Model S

⋯	Precision	Recall	f1-score	Support
Class 1	0.73	0.52	0.61	58
Class 2	0.71	0.86	0.78	79
Class 3	1.0	1.0	1.0	77
accuracy	⋯	⋯	0.82	214
macro avg	0.81	0.79	0.79	214

Table 5
Report for Model LS

⋯	Precision	Recall	f1-score	Support
Class 1	0.61	0.33	0.42	120
Class 2	0.55	0.78	0.65	123
Class 3	0.95	1.0	0.98	121
accuracy	⋯	⋯	0.70	364
macro avg	0.71	0.70	0.68	364

Table 6
Report for Model LT

⋯	Precision	Recall	f1-score	Support
Class 1	0.61	0.71	0.65	130
Class 2	0.59	0.46	0.52	114
Class 3	0.97	1.0	0.98	56
accuracy	⋯	⋯	0.67	300
macro avg	0.72	0.72	0.72	300

Table 3. For the S model, these values are 0.61 and 0.78 respectively; see Table 4. For the LS model, these values are 0.42 and 0.65 respectively; see Table 5. For the LT model, these values are 0.65 and 0.52 respectively, see Table 6. Based on the f1- metric, model S and model LT deliver the best results. The worst results are achieved by the LS model.

If we look at the results of all models, we can summarize them as follows: the class 3 objects that identify clusters are recognized very well. This is the expected behavior: recognizing filaments and walls is much more difficult than recognizing clusters. It seems that the smallest cell size provides a better performance in the detection of filaments and walls.

3.2. Report on Prediction

Next, we use plots (which are not included in the training or validation sets) to see if the CNN can correctly predict the classes of these unseen plots. The results are shown in Figure 8 in the form of a confusion matrix and in Tables 7–10. Let us now present the results.

For model L, the CNN returned 63 correct labels and 27 incorrect labels, which mean that the proportion of true positives for classes 1, 2 and 3 is 0.23, 0.16 and 0.3, respectively. From these results, it can be concluded that the CNN has a global accuracy of 69% (the sum of all these proportions). For model S, the CNN provided 63 correct labels and 27 incorrect labels. The proportions of true positives are 0.26, 0.1 and 0.33, for classes 1, 2 and 3 respectively. The sum of all these proportions is 69%. These values are also shown in Table 1.

Model LS was an attempt to combine model L and model S, i.e., all the plots of individual models were combined in model LS. However, the size of the array was unmanageable due to the high memory requirements. For this reason, we only use the first half of the plots of each model; see Table 1. As we mentioned in the caption of Figure 4, the sizes of the plots in model L and model S are different. In order to obtain plots of uniform size for model LS, i.e., plots with the same width and height, we applied a procedure to scale the plots of model L so that they have the same size as the plots of model S. Nevertheless, the length scale of the coordinate axes is still different. We believe that this difference does not change the results of the neural network.

Using the LS model, we tried to test whether the CNN is sensitive to the different length scales of the plots. However, it seems that the results of this model LS change significantly compared to model L and model S; see Table 5. The confusion matrix of model LS has more unseen plots than that of model L or model S, as can be seen in the bottom panel of Figure 8. For model LS, the total number of unseen plots was also 90, of which 35 were correctly labeled and 55 were incorrectly labeled. The proportions of true positives are 0.1 for class 1, 0.1 for class 2 and 0.16 for class 3. The sum of all these values is 0.36. This value is much smaller than the values obtained for model L (0.69) and model S (0.69).

For the LT model, the proportions of true positives are 0.28, 0.1 and 0.18, respectively. The sum of these proportions is 0.56. Based on the CNN's ability to make predictions, the L and S models therefore deliver the best results. The LS model delivers the worst results.

From the f1-scores for the prediction, we can see that the LS model gives the worst results of 0.4, 0.25 and 0.5 for class 1, 2 and 3 matter structures respectively. The L and S models provide better results of 0.63. The LT model provides results at an intermediate level compared to the other models. The L and

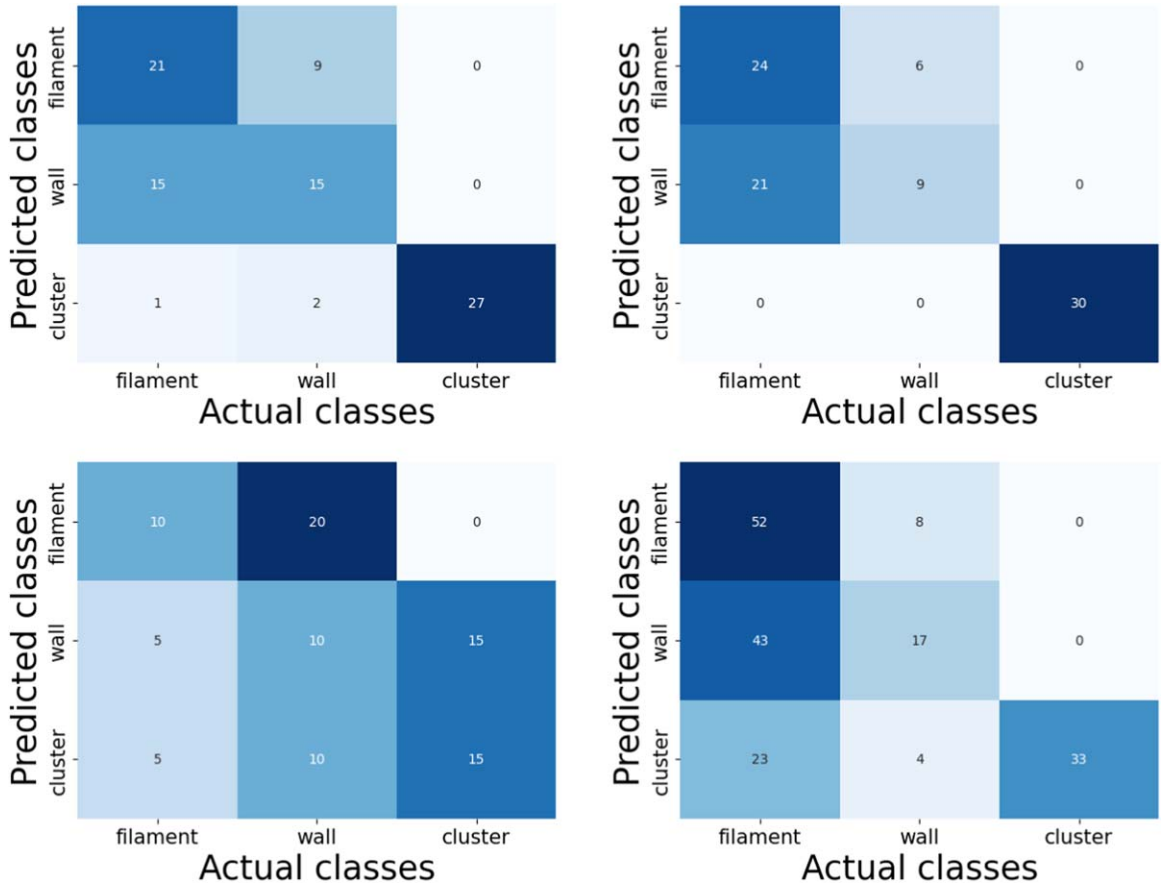

Figure 8. Confusion matrix for unseen plots. Model L (top left); model S (top right); model LS (bottom left) and model LT (bottom right).

Table 7
Report on Prediction for Model L

⋯	Precision	Recall	f1-score	Support
class 1	0.7	0.57	0.63	37
class 2	0.5	0.58	0.54	26
class 3	0.9	1.0	0.94	27
accuracy	0.7	0.7	0.7	0.7
macro avg	0.7	0.71	0.70	90

Table 8
Report on Prediction for Model S

⋯	Precision	Recall	f1-score	Support
class 1	0.8	0.53	0.64	45
class 2	0.3	0.6	0.4	15
class 3	1.0	1.0	1.0	30
accuracy	0.7	0.7	0.7	0.7
macro avg	0.7	0.71	0.68	90

S models also provide high identification performance for class 3 matter structures, as we have seen so far for all models. It should be emphasized that the LS and LT models have lower predictability for class 3 matter structures based on this metric.

4. Discussion: Some Concerns

The aim of this paper was to use a typical CNN to classify a set of Cartesian slice plots, each of which was constructed using the simulation particles located in (and around) a set of cubic cells (of type L and S) that are part of a uniform partition of the simulation volume (cell of type H). The first problem is that the assigned labels may not be entirely correct, but they are assumed to be the true target in CNN, which means that we examined the results under this assumption. However, other codes might assign different labels to the same set of cells, as we explain below.

After we completed this work, we found the code TVWEB of Forero-Romero et al. (2009). This code determines labels for all grid elements of a uniform partition, regardless of the number of simulation particles contained in each grid element. Then, through a smoothing process with a Gaussian kernel, this method obtains the density field, which is then transformed into Fourier space to yield the diagonalization of the Hessian matrix.

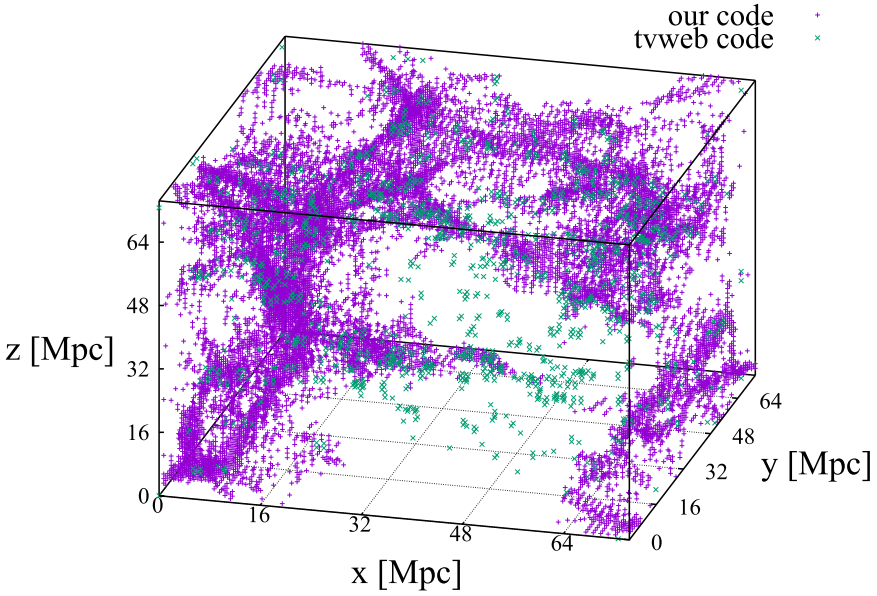

Figure 9. A comparison between the spatial distributions of the cells in class 3 according to our code and the TVWEB code.

Table 9
Report on Prediction for Model LS

···	Precision	Recall	f1-score	Support
class 1	0.33	0.5	0.4	20
class 2	0.33	0.25	0.29	40
class 3	0.5	0.5	0.5	30
accuracy	0.38	0.38	0.38	0.38
macro avg	0.38	0.41	0.39	90

Table 10
Report on Prediction for Model LT

···	Precision	Recall	f1-score	Support
class 1	0.86	0.44	0.58	118
class 2	0.28	0.58	0.38	29
class 3	0.55	1.0	0.70	33
accuracy	0.56	0.56	0.56	0.56
macro avg	0.56	0.67	0.55	180

The output of the TVWEB code provides different numbers than the ones we obtained, namely 113,209 cells in class 1, 121,109 cells in class 2 and 1416 cells in class 3. TVWEB recognizes many cells in class 1 and class 2, while the number of cells in class 3 is much smaller. As we have shown in Section 2.3, the number of cells labeled as class 3 with our code was 14,515 (with three positive eigenvalues), which is much larger than the number of cells labeled as class 2 and class 1 (145 and 133, respectively) and the number of cells of class 3 found with TVWEB. At this point, it must be mentioned that the number of cells with non-positive eigenvalues (including all cells discarded from the calculation because the number of simulation particles was below the threshold) found by our code was 247,351. TVWEB found 26,410 cells with non-positive eigenvalues. The total number of cells counted for both codes is 262,144, which takes into account all the cells in the partition.

To compare the assigned labels between the TVWEB code and our code, we look at a set of 15 plots (including samples of each class) and found nine matches and six discrepancies. To further compare these methods, we created a 3D plot of all cells in class 3 that were recognized by both codes, as shown in Figure 9. We emphasize that we only plot the centers of the cells, not the simulation particles. Despite the difference in the numbers shown above, it can be seen that the spatial distributions of class 3 cells from the two codes are very similar. We have observed similar results with other classes of matter structures.

In addition, differences in class assignment have also been noted in other works; see for example Forero-Romero et al. (2009), who made visual corrections to the class assignments. Another way to make adjustments to the class assignment was to implement a threshold eigenvalue; see Hoffman et al. (2012) and Forero-Romero et al. (2009). In this work, we did not use a threshold eigenvalue in order not to reduce the number of available plots, as we explain below.

According to the results shown in Section 3.1, the CNN was successfully trained and tested, as the test accuracy (in row 12 of Table 1) of the CNN is 0.72 (for model L 0.7, for model S 0.81, for model LS 0.70 and for model LT 0.67). The ability of the CNN to make correct predictions for the class of an unseen

set of plots was found to be 69% for models L and S; 38% for model LS and 56% for model LT. We calculate the sum of the true positives, as can be seen in row 14 of Table 1. Thus, combining the plots with the two cell sizes in a single set of training plots results in the CNN making more errors in identifying matter structures. Based on the macro-averaged f1-score, we obtained 0.68, 0.69 and 0.68 for model L for classes 1, 2 and 3, respectively. For model S, the scores were 0.81, 0.79 and 0.79 respectively. For model LS, the scores were 0.71, 0.70 and 0.68 respectively. For model LT the scores were 0.72, 0.72 and 0.72 respectively.

It must be emphasized that this level of performance is similar to that of Inoue et al. (2022) achieved a performance of 64% (macro-averaged f1-score) with a CNN for the classification of cosmic structures based on galaxies from simulations. Aragon-Calvo (2019) reported an accuracy of just over 0.9 for both filaments and walls. The precision values for filaments and walls were 0.7 and 0.78, respectively. Similarly, the recall values for filaments and walls were 0.78 and 0.77, respectively. In this paper, precision values of just over 0.56 (model L), 0.73 (model S), 0.61 (model LS) and 0.61 (model LT) were obtained for filaments. For walls, the accuracies were 0.52 (model L), 0.71 (model S), 0.55 (model LS) and 0.59 (model LT). For class 3 structures (clusters) we achieved precision and recall values of over 0.95 for all models.

A second problem is the small number of plots considered in this study. It was found that the power of CNNs increases on a logarithmic scale with the size of the data set; see Sun et al. (2017). Typically, the data sets for CNN models contain tens of thousands of training plots. To partially mitigate this situation, we increase the number of plots by rotating the original plots around the origin of the coordinates of each cubic cell.

For the LT model, we create plots of the three projections in the *XY*, *ZX* and *ZY* planes. Let us consider the usefulness of these plots for the CNN calculations. As far as we know, there is no way to tell the CNN that the three projections refer to the same 3D matter structure, and we believe that the CNN interprets these projections as different examples of labeled structures. For the CNN, plots obtained by rotating an original plot multiple times and those obtained by projection are equivalent. For the human eye, there is an obvious benefit, as the projections of a 3D structure in the three coordinate planes allow us to get a better idea of the overall image, while this is not the case for the CNN.

Therefore, in this paper we managed to run the models with only a few hundred plots. However, when we combined model L and model S, the number of plots was unmanageable for the system: "ArrayMemoryError: Unable to allocate 11.2 GiB for an array with shape (3006, 578, 434, 4)." For this reason, we have reduced the number of plots from the LS model that are used as input for the CNN.

There are also limitations on the number of epochs required to train the CNN, as the training process makes intensive use of the RAM, which is quickly exhausted. For example, the LT model could be trained for up to 25 epochs (with 1500 plots). In the literature, it is common for models to be trained for 60 or more epochs. Although the number of plots varies considerably between models, the results are at the same level and indeed similar.

A third problem is that some matter structures with excessive density may lie partially outside the cubic cells and the visualization volume. For this reason, some plots have cuts in the overdensity structure. This could be avoided if the overdensity structure is located first and then a cubic cell is placed at the corresponding location. In this case, the cover mesh would no longer be uniform and other techniques would have to be used.

A fourth point in this paper is to decide on a particular type of CNN with a particular set of parameters. We have pointed out in Section 2.5 that this CNN model comes from Bagnato (2023), who used it to classify 25,000 images of dogs and cats (12,500 of each class) with different sizes. With the parameters given in Table 2, they achieved an accuracy of 0.74 in recognizing images of dogs and cats. We employed an alternative CNN obtained from Bagnato (2023) who used it to classify sports in 70,000 images showing nine different types of sports. The images had a size of 21 $\times$ 28 and three color channels. Bagnato (2023) achieved an accuracy of 0.84 in recognizing sports images. Using the same parameters as in Bagnato (2023), we achieved a similar level of performance as the network described in Table 2; see Appendix B.

We applied these CNNs to classify images of matter structures (with excessive density) with the same parameters that the authors considered. The images they used are very different from our images. Our images also have three color channels and they have a width of 578 px and a height of 434 px for model L (640 px by 480 px for model S). We did not investigate any changes to these parameters. With the current parameter values, an accuracy of about 0.7 was achieved.

It must be mentioned that the original plots have a bar with a color scale associated with the density values. The blue color stands for low-density regions. The green color represents medium-density regions. The red color stands for high-density regions. The bar has been was removed.

5. Concluding Remarks

The formation of matter halos occurs through the process of gravitational collapse, see Arreaga-García (2007, 2016, 2017). Therefore, the plots of halos must contain information about how this formation process took place. A very homogeneous collapse leads to compact, spherical halos, while a very inhomogeneous collapse leads to elongated halos. Between these collapse extremes there is a whole range of possible halo configurations. Until a few years ago, such plots could only be analyzed by visual inspection.

In this paper, we have looked at the types of plots that are commonly used in particle-based numerical simulations to visualize the results. These are mainly 2D Cartesian plots, in particular colored isodensity plots for a slice of particles from the simulation volume. Then, we proposed the goal of this paper, which is to create hundreds of plots for a subset of cells of the simulation volume to characterize the type of matter structure that each cell contains. Due to the 3D nature of the cosmic web, it is almost impossible for the human eye to distinguish the other classes of matter using only a 2D Cartesian projection.

Motivated by the success of neural networks in discriminating between images of dogs and cats, the plots mentioned above were used in this paper as input for a typical CNN. In Section 2.5 we defined several CNN models with layers that have been successfully used in other domains to classify images. In Appendix B, we look at an alternative CNN. For the snapshot considered here at a zero redshift, the axes of the plots vary from -1.5 to 1.5 Mpc for model L (that is, for the visualization of cell type L) and -0.85 to 0.85 Mpc for model S (visualization of cell type S). For the LS model, we combine the plots of the two previous models. All these models use plots with the projection of the *XY* plane. The LT model relies on Cartesian plots of the *ZX* and *ZY* planes in addition to the projection of the *XY* plane.

In Section 3.1 we trained the CNN system for as many epochs as possible, using as many plots as possible. Test accuracy values of 60%–70% were achieved for all models, as can be seen in Table 1. This is a performance level comparable to other models in the literature. The predictive ability of this CNN was tested using a confusion matrix, and we found a performance of about 0.69. In general, we found that the recognition of class 1 and class 2 structures (filaments and walls) was more problematic and that class 3 structures (clusters, also called knots) were more easily recognized. This is to be expected, since the first two types of structure are spatially extended, while the latter are spatially compacted. More specific conclusions can be drawn as follows:

1. It appears that the smallest cell size (type S) provides a better performance in terms of filament and wall identification.
2. The LS model, which combines plots of different sizes (S and L type cells), gives the worst results.
3. Although the number of plots varies considerably between the models, the results are at the same level and even similar.
4. All models were able to classify class 3 matter structures, which we refer to as clusters. However, when using the f1-score metric, the LS and LT models showed a lower level of predictability for class 3 matter structures, as can be seen in Section 3.2.
5. The two CNNs considered in this paper provide generally similar results. However, the CNN in Appendix B provides better results in terms of its ability to predict the class of unseen plots, as shown in row 14 of Table B1. Due to the smaller number of layers in the alternative CNN, the execution time could be reduced to about half of the execution time of the CNN described in Section 2.5.

We would like to emphasize that we have neither presented this work as an improvement of the methods for detecting cosmic web structures, summarized in Section 1, nor to present a new result of physical interest. Our aim was to show that the plots commonly used to illustrate the results of a cosmological simulation contain valuable information about the cosmic structure that can be processed using CNN technology.

He et al. (2019) proposed a model to use the enormous potential of neural networks to distinguish between different cosmological models. This model is based on linking the Zeldovich initial conditions with an approximation to represent the structure evolution in the universe, which is less costly than using *N*-body simulations. The input data of the neural network consist of 10,000 cubes with the Zeldovich function applied to the particles of the simulation. The output data are the matter structure generated by the FastPM tool. The error function, to be minimized by the free parameters of the neural network, is the difference between the Zeldovich displacement functions for the particles of the simulation. Here it is worth investigating the possibility of using a slicing technique and generating plots of the simulation cubes, as we have done in this article, and seeing if a label can be assigned to each plot to distinguish between different cosmological models. See also the paper of Hong et al. (2020).

Acknowledgments

The author gratefully acknowledges the computer resources, technical expertize, and support provided by the Laboratorio Nacional de Supercómputo del Sureste de México through grant No. O-2016/047. We also appreciate the collaboration with the physics students José Antonio Sanabria Vázquez and Carlos Oswaldo Ochoa Bojórquez.

Appendix A
Comparison of the Distribution of Classes by Calculating the Hessian Value Using Two Cells of Different Sizes

In Figure 3 we compare the sizes of the cells considered in this study, i.e., the H-cell, which is used to calculate the Hessian, and the L-cell, which is used to visualize the matter structure. In this section, we investigate whether there is a significant difference in the set of class labels when we use the L-cell instead of the H-cell to diagonalize the Hessian. The mesh for the simulation box has the same center for both cell types, although for the L-cell mesh, many more particles need to be taken into account to compute the Hessian for each cell center.

Table A1
Comparison Between Cells to Calculate the Hessian

0	Matter Structure	Cell H (150)	Cell L (150)	Cell L (300)
1	nClass 1	133	817	636
2	nClass 2	145	998	805
3	nClass 3	14515	93482	75571
4	ncells	14793	95297	77012
5	fClass 1	0.008991	0.008573	0.008258
6	fClass 2	0.009802	0.010473	0.010453
7	fClass 3	0.981207	0.980954	0.981289
8	aveC1	338	1436	1781
9	aveC2	373	2107	2559
10	aveC3	903	2525	3071
11	execution time	20 Hr	60	40

The use of more particles in the Hessian calculation has two consequences: first, the effects of matter outside the H-cell are taken into account, which mean that the limitation mentioned in Section 2.3 when calculating the Hessian with the H-cell is reduced; and second, the computation time needed to traverse the simulation box increases significantly.

In Table A1, we present the results of this comparison. In column 1, in rows 1 to 3, nClass *i* denotes the number of cells labeled as class *i*. In the fourth row, *ncells* stands for the total number of selected cells in the partition. In lines 5 to 7, *fclass i* is the proportion of cells of class *i* to the total number of selected cells in the simulation box. In lines 8 to 10, *aveC i* is the average number of particles per cell of class *i*. In line 11, the execution time is the time required for a serial code to run through all cells of the mesh. In columns 3 and 4 of Table A1, the value in brackets (next to the cell name) indicates the threshold number of particles, i.e., the minimum number of particles that a cell must have in order to be selected.

The most important result of this comparison is that the proportions of cells for each class are very similar, as can be seen in rows 5, 6 and 7 of columns 3, 4 and 5 of Table A1. Consequently, we expect that the distribution of labels will be practically the same whether the H-cell or the L-cell is used to diagonalize the Hessian. It is therefore expected that the results of the CNN will be very similar to those already described in Section 3 of this study.

However, in the case of the L-cell, the number of cells per class increases, which means that more plots could be generated for each class, as shown in rows 1, 2 and 3 of columns 4 and 5 of Table A1. It is generally known that the larger the sample of training plots, the better the results of the CNN. Exploring this possibility will be of interest in future work.

Appendix B
Alternative CNN

We have mentioned in Section 4 that we employed an alternative CNN originating from Bagnato (2023), who used it to classify sports in 70,000 images with nine different sports. The input plots of this alternative CNN were the same that we used in this paper as input for the CNN defined in Section 2.5, but the number of plots is slightly different due to technical details, which do not change the results significantly. In Figure B1 we illustrate a schematic representation of the architecture of the alternative CNN, which is similar to the one shown in Figure 6.

In fact, the results of these two CNNs are very similar, as can be seen in Table B2. It should be noted that rows 10–13 of Table B2 display the results generated during the code execution where the validation data were used. Line 14 shows the results generated with the prediction data.

In Figure B1, we display a schematic representation of the architecture of the alternative CNN, for comparison with that shown in Figure 6.

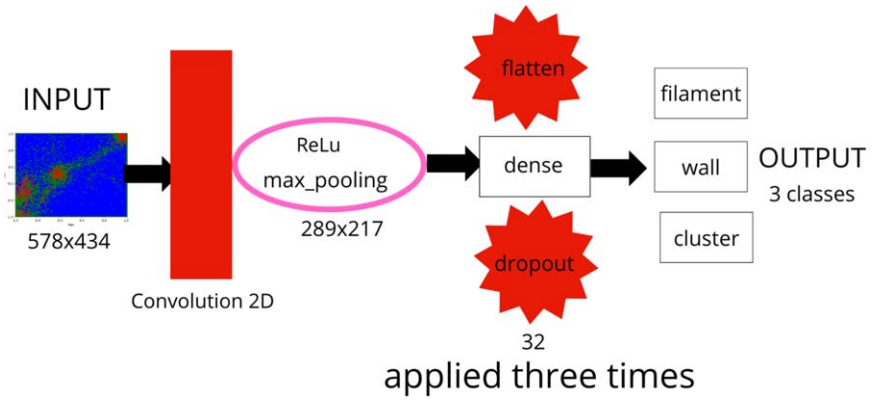

Figure B1. Schematic diagram of an alternative CNN. The most important difference between the CNNs is that, in the original CNN shown in Figure 6, the convolution is applied three times and the output process only once; in the alternative CNN of this section, the convolution is applied only once and the output process three times.

Table B1
Alternative CNN

Layer (Type)	Output Shape	Number of Parameters
conv2d 1	(None, 578, 434, 32)	1184
leaky relu 2	(None, 578, 434, 32)	0
max pooling 2d 1	(None, 289, 217, 32)	0
dropout 2	(None, 289, 217, 32)	0
flatten 1	(None, 2006816)	0
dense 2	(None, 32)	64218144
leaky relu 3	(None, 32)	0
dropout 3	(None, 32)	0
dense 3	(None, 3)	99

Table B2
The Number of Images Used and Some Results

Row		Model L	Model S	Model LS	Model LT
1	Class 1	462	762	610	600
2	Class 2	466	834	662	600
3	Class 3	433	801	624	300
4	training plots	1088	1917	1516	1200
5	validation plots	273	480	304	300
6	prediction plots	90	90	180	180
7	total plots	1451	2397	1896	1500
8	Total parameters	64219427	78644515	64219427	78644515
9	Memory	244.98 MB	300 MB	244.98 MB	300 MB
10	correct labels	186	323	267	193
11	incorrect labels	87	157	113	107
12	test accuracy	0.68	0.67	0.70	0.64
13	test loss	0.76	0.67	0.59	0.58
14	true positives	0.22, 0.18, 0.31	0.26, 0.13, 0.33	0.21, 0.13, 0.30	0.11, 0.21, 0.13
15	Δ_L	1.5 Mpc	0.85 Mpc	1.5 and 0.85 Mpc	1.5 Mpc

References

Alpaslan, M., Robotham, A S.G., Driver, S., et al. 2014, MNRAS, 438, 177
Aragon-Calvo, M. A. 2019, MNRAS, 484, 5771
Aragon-Calvo, M. A., Jones, B. J. T., Weygaert van der, R., & Hulst van der, J. M. 2007, A&A, 474, 315
Aragon-Calvo, M. A., Platen, E., van de Weygaert, R., & Szalay, A. S. 2010, ApJ, 723, 364
Arreaga-García, G. 2007, ApJ, 666, 290
Arreaga-García, G. 2016, RMxAA, 52, 155
Arreaga-García, G. 2017, Ap&SS, 362, 47
Arreaga-García, G. 2021, RAA, 21, 198
Babul, A., & Starkman, G. D. 1992, ApJ, 401, 28
Bagnato, J. I. 2023, Blog: Aprende Machine Learning, available at: www.aprendemachinelearning.com
Balsara, D. 1995, JCoPh, 121, 357
Barrow, J. D., Bhavsar, S. P., & Sonoda, D. H. 1895, MNRAS, 216, 17
Behroozi, P., Wechsler, R.H., & Wu, H.-Y. 2013, ApJ, 762, 20
Bond, J. R., Kofman, L., & Pogosyan, D. 1996, Natur, 380, 603
Cautun, M., van de Weygaert, R., & Jones, B. J. T. 2013, MNRAS, 429, 1286
Chollet, F. 2018, Deep Learning with Python (New York: Manning Publications Co.)
Colombi, S., Pogosyan, D., & Souradeep, T. 2000, PhRvL, 85, 5515
Eisenstein, D. J., & Hu, W. 1999, ApJ, 511, 5
Falck, B., Koyama, K., Zhao, G.-b., & Li, B. 2014, JCAP, 7, 058
Falck, B. L., Neyrinck, M. C., & Szalay, A. S. 2012, ApJ, 754, 126
Fang, F., Forero-Romero, J., Rossi, G., Li, X. D., & Feng, L. L. 2019, MNRAS, 485, 5276
Flannery, B. P., Press, W. H., Teukolsky, S., & Vetterling, W. T. 1998, Numerical Recipes in C: The Art of Scientific Computing (Cambridge: Cambridge Univ. Press)
Forero-Romero, J. E., Hoffman, Y., Gottloeber, S., Klypin, A., & Yepes, G. 2009, MNRAS, 396, 1815
Ganeshaiah Veena, P., Cautun, M., van de Weygaert, R., et al. 2018, MNRAS, 481, 414
Garcia-Alvarado, M. V., Li, X.-D., & Forero-Romero, J. E. 2020, MNRAS, 498, L145
Geller, M. J., & Huchra, J. P. 1991, in Physical Cosmology, ed. A. Blanchard (Gif-sur-Yvette Cedex: Editions Frontieres)
Gott, J. R., III, Melott, A. L., & Dickinson, M. 1986, ApJ, 306, 341
Hahn, O., Porciani, C., Carollo, C. M., & Dekel, A. 2007, MNRAS, 375, 489
He, K., Zhang, X., Ren, S., & Sun, J. 2015, in IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)
He, S., Li, Y., Feng, Y., et al. 2019, PNAS, 116, 13825
Hoffman, Y., Metuki, O., Yepes, G., et al. 2012, MNRAS, 425, 2049
Hong, S., Jeong, D., Hwang, H. S., et al. 2020, MNRAS, 493, 5972
Inoue, S., Si, X., Okamoto, T., & Nishigaki, M. 2022, MNRAS, 515, 4065
Krizhevsky, A., Sutskever, I., & Hinton, G. E. 2017, ACM, 60, 84
Leclercq, F., Jasche, J., & Wandelt, B. 2015, JCAP, 6, 015
Libeskind, N. I., van de Weygaert, R., Cautun, M., et al. 2018, MNRAS, 473, 1195
Martinez, V. J., Jones, B. J. T., & Dominguez-Tenreiro, R. 1990, ApJ, 357, 50
Planck Collaboration, Ade, P.A.R, Aghanim, N., et al. 2013, A&A, 571, 66
Planck Collaboration, Ade, P.A.R, Aghanim, N., et al. 2016, A&A, 594, 38
Ramachandra, N. S., & Shandarin, S. F. 2015, MNRAS, 452, 1643
Shandarin, S. F. 2011, JCAP, 5, 15
Shandarin, S. F., & Zeldovich, Y. B. 1983, CompAp, 10, 33
Shapley, H., & Ames, A. 1932, AnHar, 88, 43
Sousbi, T., Pichon, C., & Kawahara, H. 2011, MNRAS, 414, 384
Springel, V. 2005, MNRAS, 364, 1105
Sun, C., Shrivastava, A., Singh, S., & Gupta, A. 2017, in 2017 IEEE International Conf. on Computer Vision (ICCV) (Piscataway, NJ: IEEE), 843
Szegedy, C., & Liu, W. 2015, in 2015 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (Piscataway, NJ: IEEE),
Tempel, E., Stoica, R. S., Kipper, R., & Saar, E. 2016, A&C, 16, 17
Tempel, E., Stoica, R. S., Martinez, V. J., et al. 2014, MNRAS, 438, 3465
Weinberg, D. H., & Cole, S. 1992, MNRAS, 259, 652
Zeldovich, I. B., Einasto, J., & Shandarin, S. F. 1982, Natur, 300, 407