# A deep learning approach to halo merger tree construction

Sandra Robles,[1,2,3]★ Jonathan S. Gómez,[1,4]† Adín Ramírez Rivera,[5]‡ Nelson D. Padilla[6] and Diego Dujovne[7]

[1]*Departamento de Física Teórica, Universidad Autónoma de Madrid, E-28049 Cantoblanco, Madrid, Spain.*
[2]*Theoretical Particle Physics and Cosmology Group, Department of Physics, King's College London, Strand, London, WC2R 2LS, UK*
[3]*ARC Centre of Excellence for Dark Matter Particle Physics, School of Physics, The University of Melbourne, Victoria 3010, Australia.*
[4]*Instituto de Astrofísica, Pontificia Universidad Católica de Chile, Av. Vicuña Mackenna 4860, Santiago, Chile.*
[5]*Department of Informatics, University of Oslo, Gaustadalléen 23 B, N-0373, Oslo, Norway.*
[6]*Instituto de Astronomía Teórica y Experimental, UNC-CONICET, Córdoba, X5000BGR, Argentina*
[7]*Escuela de Informática y Telecomunicaciones, Universidad Diego Portales, Av. Ejército 441, Santiago, Chile.*

Accepted June, 2022.

**ABSTRACT**

A key ingredient for semi-analytic models (SAMs) of galaxy formation is the mass assembly history of haloes, encoded in a tree structure. The most commonly used method to construct halo merger histories is based on the outcomes of high-resolution, computationally intensive N-body simulations. We show that machine learning (ML) techniques, in particular Generative Adversarial Networks (GANs), are a promising new tool to tackle this problem with a modest computational cost and retaining the best features of merger trees from simulations. We train our GAN model with a limited sample of merger trees from the EAGLE simulation suite, constructed using two halo finders–tree builder algorithms: SUBFIND–D-TREES and ROCKSTAR–ConsistentTrees. Our GAN model successfully learns to generate well-constructed merger tree structures with high temporal resolution, and to reproduce the statistical features of the sample of merger trees used for training, when considering up to three variables in the training process. These inputs, whose representations are also learned by our GAN model, are mass of the halo progenitors and the final descendant, progenitor type (main halo or satellite) and distance of a progenitor to that in the main branch. The inclusion of the latter two inputs greatly improves the final learned representation of the halo mass growth history, especially for SUBFIND-like ML trees. When comparing equally sized samples of ML merger trees with those of the EAGLE simulation, we find better agreement for SUBFIND-like ML trees. Finally, our GAN-based framework can be utilised to construct merger histories of low and intermediate mass haloes, the most abundant in cosmological simulations.

**Key words:** methods: numerical – galaxies: haloes – dark matter.

## 1 INTRODUCTION

The dark energy dominated dark matter model, Λ cold dark matter (ΛCDM), provides a successful theoretical framework for understanding and simulating galaxies, which are believed to inhabit dark matter (DM) haloes. Galaxy formation and evolution are complex non-linear problems, theoretically and numerically. Gas-dynamical and radiative processes, such as the formation of stars and black holes as well as their respective feedback, have to be taken into account for simulated galaxies to resemble reality. Over the past decades, two different strategies have been developed to tackle this problem: hydrodynamical simulations (Carlberg et al. 1990; Katz et al. 1992) and semi-analytic models (SAMs) (Cole 1991; White & Frenk 1991; Croton et al. 2016).

Hydrodynamical simulations directly address a wide range of dynamical scales and solve numerically the combined non-linear N-body and hydrodynamic equations describing the formation of galaxies. They provide many advantages, among them, self-consistent evolution of DM and baryonic components, high resolution of the latter component, and, furthermore, they simultaneously model galaxies and the intergalactic medium. Even if high resolution simulations are massively parallelised and run on supercomputers, they are extremely computational resource intensive. Alternatives to circumvent this limitation are semi-analytic modelling and generating DM halo catalogues with less orders in Lagrangian perturbation theory (Munari et al. 2017). These models are established tools for connecting the predicted hierarchical growth of DM haloes, the halo merger trees, to the observed properties of the galaxy population (see e.g. Cole et al. 2000; Somerville et al. 2008; Guo et al. 2011). It is worth noting that even though both hydrodynamical simulations and semi-analytic models rely on subgrid physical models to account for processes that cannot be directly simulated, they are more approximate in the latter models.

In the standard model of cosmology, galaxies form in DM haloes collapsed from tiny overdensities. Large DM haloes are formed by the collapse and merger of smaller structures or progenitors. Thus, galaxy formation and evolution are driven by the halo merger history. If the progenitors contain galaxies, halo mergers eventually give rise

★ E-mail: sandra.robles@kcl.ac.uk (SR)
† E-mail: jsgomez1@uc.cl (JG)
‡ E-mail: adinr@uio.no (ARR)

to galaxy mergers. The merging and hierarchical formation history of DM haloes can be obtained in cosmological simulations by tracing all halo progenitors and storing them in tree structures, commonly referred to as "halo merger trees."

There is another method to produce halo merger trees, which is based on the extended Press-Schechter formalism (Bond et al. 1991) and Monte Carlo simulations (Kauffmann & White 1993; Kauffmann et al. 1993; Cole et al. 1994). This is a simple but relatively effective framework for the description of the mass history of particles in a hierarchical Universe. Its main advantage is a rapid merger tree construction in large volumes with high mass resolution (Lacey & Cole 1993; Somerville & Primack 1999; Cole et al. 2000; Somerville et al. 2008; Benson & Bower 2010; Ricciardelli & Franceschini 2010). However, only one tree can be built at a time and often this framework yields results that are in disagreement with simulations (Jiang & van den Bosch 2014).

Cosmological N-body simulations are a powerful and well established tool for studying theories of cosmic structure formation and for making predictions that can be compared directly to observations. Despite being computationally intensive, high-resolution DM only (N-body) simulations yield a more realistic evolutionary history of the haloes and are capable of producing thousands of merger trees at once (Roukema et al. 1997; Kauffmann et al. 1999; Okamoto & Nagashima 2001; Hatton et al. 2003; De Lucia et al. 2004; Croton et al. 2006; Bower et al. 2006; Guo et al. 2011). This method, however, is not exempt of subtleties. The most relevant being the dependence on the clustering algorithms employed to find haloes and to build trees (Knebe et al. 2011; Avila et al. 2014; Gómez et al. 2022). Another important caveat is the mass resolution limit, poorly resolved haloes or dense environments can be problematic for identifying substructures for some halo finders (or substructure finders; Muldrew et al. 2011; Onions et al. 2013; Elahi et al. 2013). Contrary to haloes with a small number of particles, the assembly history of massive haloes can be traced back to progenitors whose masses are a small fraction of the mass of the final descendant.

Despite the difficulties of studying the evolution of galaxies, merger trees of DM haloes play an important role in modern galaxy formation theory. They are the backbone of SAMs. SAMs populate dark matter haloes in cosmological simulations by using analytical approximations to self-consistently model the evolution of galaxies throughout cosmic time. Due to the flexibility of SAMs to explore physical phenomena, they are best suited to compare theoretical predictions with galaxy surveys (Lagos et al. 2018). However, the necessary condition for a SAM to produce galaxy formation and merger histories is to have a complete sample of well-constructed and realistic merger trees.

In recent years, deep learning algorithms, a subset of machine learning (ML) techniques that compose functions from parameterized operations that enforce few inductive biases over the parameterization, have increasingly been used in astrophysics because they can process large sets of data and extract features from them by observing patterns in the data. In particular, convolutional neural networks (CNNs), a deep learning model designed to learn spatial hierarchies of features, have been employed, among others, to measure the dynamical mass of galaxy clusters (Ho et al. 2019), map N-body and hydrodynamical simulations (Wadekar et al. 2021) and thereby exploring the halo-galaxy connection, to morphologically classify galaxies (Dieleman et al. 2015; Kim & Brunner 2017; Barchi et al. 2020; Cavanagh et al. 2021), and segment and classify the large scale structure of the Universe (Aragon-Calvo 2019).

With the aim of providing a new framework for halo merger tree construction, taking advantage of the best features of large volume simulations, but with a modest computational expense, we extend the deep convolutional neural network model designed and tested by Robles et al. (2019). This model is based on a Generative Adversarial Network (GAN) (Goodfellow et al. 2014), which we train with merger trees from the largest dark matter only simulation in the Evolution and Assembly of GaLaxies and their Environments (EAGLE) simulation suite (Schaye et al. 2015; Crain et al. 2015). GANs are an unsupervised deep learning technique, characterised by training a pair of neural networks in competition with each other in a sort of minimax game. Finding new applications for GANs is currently an active area of research. They have been employed mainly for computer vision applications such as image generation, editing and classification. Recently, they have been proved to be useful to solve computational and data intensive tasks in physics. We propose here a new application in astrophysics that can help SAMs of galaxy formation to more reliably simulate large upcoming galaxy surveys and allow us to more rapidly compare theory with observations.

**Table 1.** Parameters and technical specifications of the N-body simulations used in this work.

| Parameter | Physical meaning | Value |
|---|---|---|
| $\Omega_m$ | Present fractional matter density | 0.307 |
| $\Omega_\Lambda$ | Present fractional vacuum energy density | 0.693 |
| $h$ | $H_0/(100\ \mathrm{km\ s^{-1}\ Mpc^{-1}})$ | 0.6777 |
| $n_s$ | Primordial power spectral index | 0.9611 |
| $\sigma_8$ | rms linear density fluctuation | 0.8288 |

| Simulation | $L_{\mathrm{box}}$ (cMpc) | $N_{\mathrm{p}}$ | $m_{\mathrm{dmp}}(h^{-1}\mathrm{M}_\odot)$ |
|---|---|---|---|
| EAGLE100 | 100 | $1504^3$ | $6.57 \times 10^6$ |

We perform a series of experiments including up to three input variables in the training process. This choice of variables that describe halo merger trees is motivated by SAMs (Cole et al. 2000; Benson 2012; Croton et al. 2016; Cora et al. 2018). Namely these quantities are mass growth, distance between merging progenitors and the progenitor type: main or satellite halo. These experiments were designed to test the capabilities of our GAN model and the relevance of each input to the well reconstruction of the mass growth history of haloes. To validate the correct construction of the ML generated trees, we compare statistically significant samples of machine learning generated with "real" trees, where "real" stands for merger trees extracted from the EAGLE simulation and ML generated trees are the outputs of our GAN model, and find very good agreement.

This paper is organised as follows. In Section 2, we briefly give details of the EAGLE simulation, halo finder and tree builder algorithms utilised to extract merger trees from simulations. In Section 3, we outline the GAN model employed to generate merger trees and the statistical measures used to assess the quality of these trees. Details of the GAN architecture can be found in Appendix A and additional examples of ML generated trees in Appendix B. Our results are presented in Section 4. Concluding remarks are given in Section 5.

## 2 MERGER TREES FROM SIMULATIONS

In order to generate halo merger trees of dark matter haloes using neural networks, a training dataset of "real" merger trees is required, which we select from the EAGLE simulation suite. This suite consists of several hydrodynamical simulations (Schaye et al. 2015), but also contains dark matter only (DMO) versions of the reference simulations. The DMO simulations were obtained using the same

initial conditions and the same resolutions as the reference models, DMO simulations and their hydrodinamical counterparts were later matched (Schaller et al. 2015). We use the largest DMO simulation, which has a cubic periodic volume of 100 co-moving Mpc side (hereafter referred to as E100) and with a dark matter particle (DMP) mass $m_{\rm dmp} = 6.57 \times 10^6 h^{-1} {\rm M}_\odot$. The simulation adopts the Planck Collaboration et al. (2014) cosmological parameters, listed in Table 1. E100 has a high temporal resolution of 201 snapshots, numbered from 0 to 200, distributed between $z = 20$ and $z = 0$, respectively. This simulation supplies a large enough amount of merger trees for training purposes and its high temporal resolution helps us to better distinguish pre-merger phases.

To be able to compare different learning processes of our neural network architecture, we shall use two distinct halo merger tree databases, obtained from the aforementioned simulation by applying two clustering algorithms to construct merger trees in the E100 simulation. This process is performed in two steps:

(i) First, the halo finder identifies all the haloes in each snapshot using the dark matter particles of the simulation.
(ii) Next, the tree builder constructs links between haloes across different snapshots.

This is the general method to build halo merger trees, called by convention, halo finder–tree builder.

We employ halo merger trees identified with two combinations of halo finder–tree builder listed in Table 2. As a first step, both halo finders make use of the Friends-of-Friends standard algorithm (hereinafter FoF). Then, in subsequent steps they perform a more refined search of haloes and their substructures, using more sophisticated techniques. For our study, we use the SUBFIND (Springel et al. 2001) halo finder that identify 3D overdensities (i.e., they consider only the position of the particles) and ROCKSTAR (Behroozi et al. 2013a) halo finder that identifies 6D overdensities (basically phase-space, i.e., particle positions and velocities).

It is worth mentioning that there exist many other halo finder and tree builder codes in the literature. Both algorithms are at least equally important to produce well constructed merger trees (Avila et al. 2014). Yet they are both subject to issues which have been studied in detail. Knebe et al. (2011, 2013) compared different halo finders, and investigated the sources of errors and discrepancies among them when identifying haloes and their substructure. Several methods to build merger trees were compared using the same halo catalogue by Srisawat et al. (2013). They concluded that a reliable tree builder should be able to trace particle transfers in order to match haloes between adjacent snapshots, and skip at least one snapshot to correct for missing haloes. They also found that different tree builders even using the same halo catalogue yield distinct halo growth histories. This could alter galaxy properties when utilising them in SAMs (Lee et al. 2014). The effect of mass and temporal resolution on merger tree construction has also been investigated (Wang et al. 2016). An important recommendation from this study is that when merger trees are built using more than 100 snapshots, which is precisely our case, the tree builder algorithm should at least be able to deal with issues in the halo catalogue. In the following subsections, we briefly provide details of each of the two halo/finder tree builders and the terminology adopted in this work.

**Table 2.** Halo merger tree builders considered in this work. The first column gives the halo finder algorithm and the second column the corresponding tree builder.

| Halo finder | Tree builder |
|---|---|
| SUBFIND (Springel et al. 2001) | D-TREES (Jiang et al. 2014; Qu et al. 2017) |
| ROCKSTAR (Behroozi et al. 2013a) | ConsistentTrees (Behroozi et al. 2013b) |

### 2.1 SUBFIND – D-TREES

SUBFIND (Springel et al. 2001) is a self-bound particle substructure finder which first identifies particle groups using the Friends-of-Friends algorithm with linking length $b = 0.2$ times the mean inter-particle separation. In order to identify the gravitationally bound haloes from each group, a local density is estimated for each particle with an adaptive kernel interpolation that uses a prescribed number of smoothing neighbours. Starting from isolated density peaks, particles are added in sequence of decreasing density. Whenever a saddle point in the global density field is reached, such that it connects two disjoint overdense regions, the smaller candidate is treated as a separate satellite halo. All substructure candidates are subjected to an iterative unbinding procedure with a tree-based calculation of the potential.

The D-TREES algorithm (Jiang et al. 2014) is a tree builder created to be used together with SUBFIND, that finds merger histories taking into account the uncertainty in the definition of a halo and possible loss of particles. D-TREES first considers two consecutive snapshots in the simulation. Each halo is identified as the progenitor of whichever halo at the next snapshot contains the largest fraction of its particles. This process is repeated for all pairs of consecutive snapshots. It is then straightforward to trace the merger history of each halo that exists at the final output time. Then, the algorithm checks that the most bound particle of a halo remains a member of the descendant halo, and also requires that the majority of the constituent particles of a halo are present in its descendant at the next output time. If this is not satisfied, D-TREES chooses the most bound particle from those that are in a halo at the later output time that contains the largest number of the progenitor particles. Specifically, the merger trees used in this work have been obtained using the adaptation of the D-TREES algorithm to the EAGLE simulation (Qu et al. 2017).

### 2.2 ROCKSTAR – ConsistentTrees

ROCKSTAR, Robust Overdensity Calculation using K-Space Topologically Adaptive Refinement, (Behroozi et al. 2013a), is a phase-space halo finder[1] designed to maximise halo consistency across timesteps. The algorithm first selects particle groups with a 3D Friends-of-Friends variant with a very large linking length ($b = 0.28$). For each FoF group, particle positions and velocities are divided (normalised) by the group position and velocity dispersion, giving a natural phase-space metric. Then, for each main group, ROCKSTAR builds a hierarchy of FoF subgroups. Thus, the metric ensures an adaptive selection of overdensities at each successive level of the FoF hierarchy. This process is repeated for each subgroup; i.e. renormalisation, a new linking-length, and a new level of substructure are calculated for subgroups. When this is completed, ROCKSTAR converts FoF subgroups into seed haloes beginning at the deepest level of the hierarchy. If a particular group has multiple subgroups, then particles are assigned to subgroup seed haloes based

[1] https://bitbucket.org/gfcstanford/rockstar

on their phase-space proximity. This process is repeated at all levels of the hierarchy until all particles in the base FoF group have been assigned to haloes.

The ConsistentTrees algorithm[2] (Behroozi et al. 2013b), part of the ROCKSTAR package, first matches haloes between snapshots by identifying descendant haloes as those that have the maximum number of particles from a given progenitor. It then attempts to clean up this initial guess, either by correcting erroneous links or by adding missing haloes. To this end, ConsistentTrees simulates the gravitational motion of the set of haloes according to their known positions, velocities and mass profiles as returned by the halo finder (ROCKSTAR). Then, for haloes in any given simulation snapshot, expected positions and velocities at an earlier snapshot can be inferred. To correct for missing haloes, ConsistentTrees utilises halo trajectories from gravitationally evolving positions and velocities of haloes across timesteps. Thus, by using kinematic information from surrounding snapshots, it can correct for missing or extraneous haloes. This is required when ROCKSTAR can no longer find a satellite because it is too close to the host's centre, but the satellite is not fully merged so that a correction is needed to account for the missing satellite. In this way, ConsistentTrees modifies the original ROCKSTAR halo catalogue by adding missing haloes and their corresponding links.

### 2.3 Terminology

In this subsection, we briefly define the terminology used in the following sections. A halo is a gravitationally bound structure as returned by the halo finder. Haloes can contain self-bound substructures or subhaloes. In that case, the halo that contains the particle with the lowest value of the gravitational potential is the main halo, and the remaining haloes are satellites.

A merger tree, see Figs. 3 and B1, is a graph of chronologically ordered set of haloes, the outputs of the tree builder. This tree represents the mass growth of a halo (the final descendant) over time (snapshots). The final descendant is located at the top node of the tree ($z = 0$). The remaining haloes in a merger tree are dubbed progenitors. A distinctive feature of a merger tree is the so-called main branch,[3] the largest branch of the tree that contains the most massive progenitor, which is expected to be the main halo. Other branches emerge from the main branch connecting progenitors, which can be main or satellite haloes, backwards in time. When two or more branches fuse to become one, a merger has occurred.

## 3 HALO MERGER TREE GENERATION

In this section, we outline the halo merger tree generation process implemented here, which uses a deep-learning-based generative model. We also define statistical measures that help us to validate the appropriate construction of the ML generated trees.

### 3.1 Halo Merger Tree Representation

We selected halo merger trees for main haloes at $z = 0$ from the aforementioned EAGLE simulation, identified with SUBFIND–D-Trees and ROCKSTAR–ConsistentTrees. The selection criterion is based on the number of trees available per number of branches. We select merger trees with at least 6 branches, since we are interested in reproducing relatively complex tree structures. The maximum number of branches considered during training, as we shall see in Section 4, will depend on the halo finder–tree builder algorithm. In Fig. 1, top panel, we show a histogram of the initially selected merger trees binned by the number of branches. A large number of examples is required for our neural network model to properly learn to represent

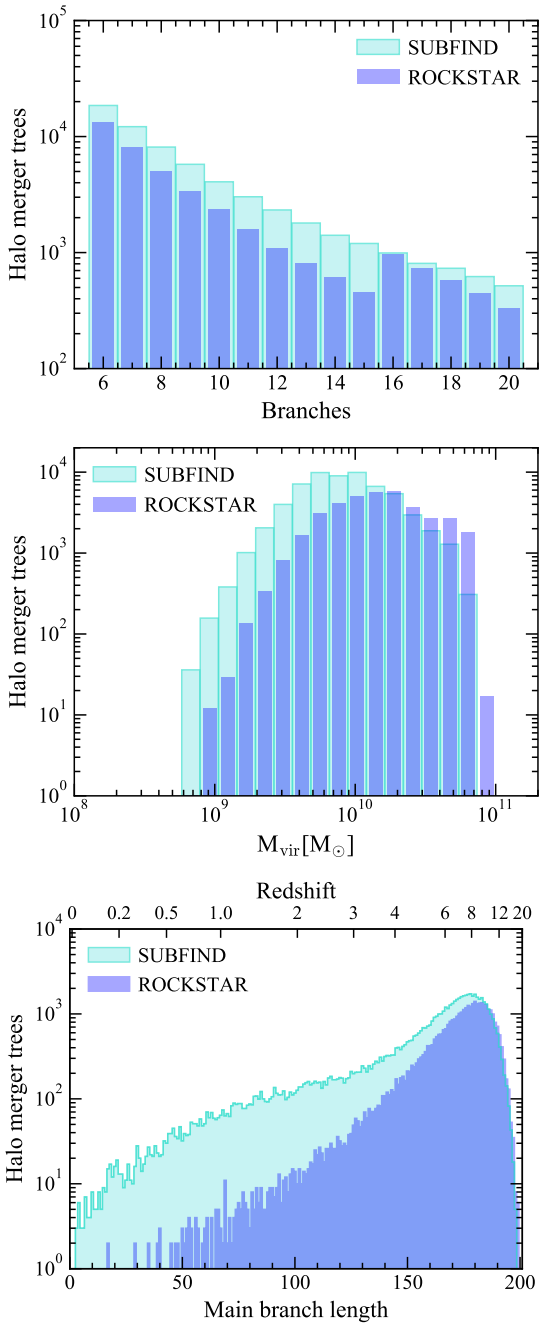


**Figure 1.** Histograms of the merger trees in the training dataset. These are trees for main haloes at $z = 0$, identified using SUBFIND–D-TREES (light blue) and ROCKSTAR–ConsistentTrees (blue), with number of branches in the $6 \leq n_{\rm br} \leq 20$ range (top panel). This sample corresponds to main haloes with virial masses in the $10^9 M_\odot \lesssim M_{\rm vir} \lesssim 10^{11} M_\odot$ range at $z = 0$ (middle panel). The bottom panel shows the histogram of the length of the main branch measured in number of snapshots. (The redshift axis is also shown in the top axis for reference.)

[2] https://bitbucket.org/pbehroozi/consistent-trees
[3] Also called trunk.

these tree structures. This initial selection results in merger trees of main haloes with virial masses in the $10^9 M_\odot \lesssim M_{vir} \lesssim 10^{11} M_\odot$ range at $z = 0$ (see middle panel of Fig. 1). More massive haloes have more intricate merger histories and less examples of them are available in simulations. Note that for less massive haloes we also have fewer merger trees, this is due to the fact that they can have a smaller number of branches and we are selecting merger trees with no less than 6 branches. To show how far in time, the merger histories of the haloes in the middle panel of Fig. 1 can be traced back, we plot the length of the main branch of their corresponding merger trees in the bottom panel, measured in number of snapshots, from $z = 0$ till the endpoint of the main branch at high redshift. We immediately notice that there are few merger trees with short main branches. These, in general, correspond to the low mass haloes in our sample, as expected from the bottom-up theory of structure formation. Most of the trees in the trraining dataset tend to have long branches, with their distributions peaking slightly above $z \simeq 8$, for both SUBFIND and ROCKSTAR trees. We also note that ROCKSTAR merger trees tend to feature longer main branches, with $\sim 90\%$ of trees having main branches that span more than 150 snapshots back in time, i.e. up to $z \gtrsim 4.5$; while for SUBFIND merger trees this fraction reduces to $73\%$.

For a consistent description of the merger trees and motivated by SAMs, we consider at most three quantities to be learned by our neural network, which are the most basic input variables required by SAMs (Cole et al. 2000; Benson 2012; Croton et al. 2016; Cora et al. 2018). Namely, these are the mass of the progenitors, distance of each progenitor to the corresponding halo in the main branch, and progenitor type, a discrete variable indicating whether the progenitor is a main or a satellite halo. Each variable was stored in matrix format, one matrix per variable, where columns represent branches and rows snapshots. Matrix elements were filled following the structure of a merger tree, where the first column is the main branch, i.e., the largest branch and progenitors in every branch and snapshot are denoted by non-zero matrix elements.

We obtained separate databases for both halo finder–tree builder algorithms, SUBFIND–D-TREES and ROCKSTAR–ConsistentTrees, hereafter referred as SUBFIND and ROCKSTAR, respectively. As mentioned above, we restricted these databases to merger trees with number of branches $n_{br} \geq 6$, see Fig. 1. These are among the most abundant merger trees for haloes in the above mentioned mass range.

### 3.2 Neural Network Model

Generative Adversarial Networks are a framework designed to learn generative models of complex data distributions. GANs consist of two neural networks a generator and a discriminator in competition with each other. Their ultimate goal is that the generator learns a distribution that matches the real data, while fooling the discriminator that is trained to distinguish real from generated data samples.

The GAN architecture adopted here is based on the layout previously designed and tested by Robles et al. (2019) to generate halo merger trees with a fixed number of branches, namely $n_{br} = 6$, and a temporal resolution of 29 snapshots. In this model, the generator comprises an encoder-decoder architecture (Bengio et al. 2013) that learns to reproduce the matrix representation of merger trees. Both, the discriminator and the encoder are implemented with convolutional neural networks (CNNs) (Krizhevsky et al. 2012), while the decoder is made of deconvolutional layers. Each CNN layer features either row- or column-wise filters, which intend to reproduce operations within a branch (column-wise filters) and among merging branches (row-wise filters). Finally, reconstruction losses for each considered input (typically the mass of the progenitors and the final descendant) are added to the classic GAN loss function. These reconstruction losses drive the learning process and improve considerably the quality of the generated trees. All the loss functions are computed with cross-entropy measures. Both the discriminator and the generator are trained with Adam, a stochastic gradient-based optimisation algorithm (Kingma & Ba 2014), with batches of "real" merger trees in matrix format.

In this paper, we generalise the GAN architecture introduced by Robles et al. (2019) to produce merger trees with higher time resolution which results in some of the CNN layers featuring filters with larger kernel sizes; the precise size depends on the halo finder–tree builder algorithm employed to construct the merger trees in the training database and the maximum number of branches considered. We also remove the restriction on the number of branches, namely we train our model with merger trees with higher number of branches, $n_{br} \geq 6$, fixed and arbitrary in a given range. These modifications also alter other parameters of the GAN model such as the size of the input of the decoder and the batch size for training. The exact details of the GAN architecture are given in Appendix A. It is worth remarking that the maximum number of branches that our GAN model can learn to generate is limited by the amount of merger trees in the training dataset and memory resources. We noticed that around 1000 merger trees per given $n_{br}$ are required for the GAN total loss to converge. This number of merger trees is easy to obtain from any DM-Only simulation nowadays, depending on the mass of the final descendant and the volume of the simulation. As can be seen from Fig. 1, this imposes a more stringent restriction on the maximum number of branches of a tree that can be reconstructed using merger trees of the E100 simulation identified with ROCKSTAR.

This refurbished GAN model is trained with a dataset of "real" merger trees from the E100 simulation, identified with either SUBFIND or ROCKSTAR, where each considered variable (up to three: mass of the progenitors and the final descendant, distance to the main branch, and progenitor type) corresponds to an input channel. As in any unsupervised learning technique, only examples of the inputs are given to the GAN, which learns to reproduce them. Thus, when adding more properties of the progenitors of a halo in a merger tree structure, the GAN model must also learn to generate this information. This is reinforced by the addition of a reconstruction loss per additional input. As a result, the GAN yields as many outputs as input channels per generated merger tree. Note that in principle, it is possible to add as many extra inputs as desired, subject to memory resources. The complete generation process is summarised in Fig. 2. The training process is performed in batches of randomly shuffled merger trees, several epochs after the total GAN loss converges, we obtain as many batches of generated ("fake') trees as convenient for the analyses in Section 4. Examples of these ML generated trees are shown in Figs. 3 and B1.[4]

### 3.3 Evaluation of Generated Merger Trees

Our neural network learns to generate well-constructed merger trees, i.e., progenitors with no drastic variation in their mass and no sudden jumps in physical location. To quantitatively evaluate the quality of our GAN generated trees, as in Robles et al. (2019), we construct probability distributions of the training dataset and equally sized samples of the reconstructed trees for a given total number of branches,

[4] For visualisation purposes, we show merger trees generated with $n_{snap} = 101$, time resolution that as we shall see yields the best results.

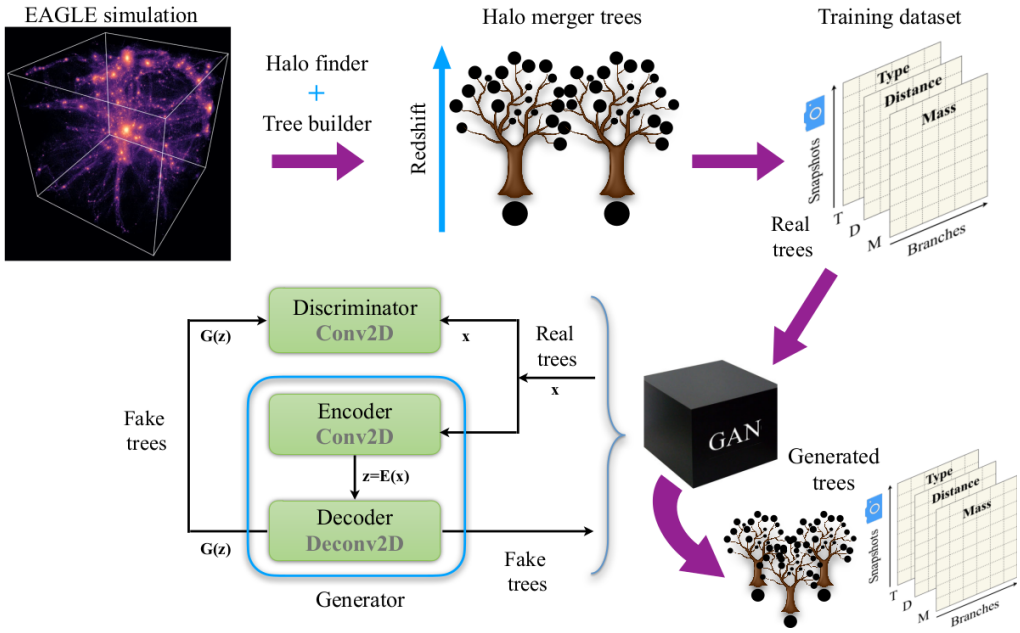


**Figure 2.** Halo merger tree generation process (adapted from Robles et al. 2019). Halo merger trees from the DM only EAGLE simulation of 100 cMpc constructed with either SUBFIND–D-TREES or ROCKSTAR–ConsistentTrees are stored in matrix format (one matrix per input variable), forming two separate datasets. Batches of these "real" merger trees are the inputs of our GAN model comprised of a discriminator and a generator that consists of an encoder and a decoder, all these neural networks are made of convolutional layers (Conv2D, Deconv2D). The outputs of the GAN are batches of "fake" (generated) trees in matrix format, as many matrices as input channels are obtained.

a given time resolution (i.e. number of snapshots: $n_{\rm snap}$ = 201 or $n_{\rm snap}$ = 101) and for each considered variable. We later compare these distributions using the Kolmogorov-Smirnov (KS) test.

It can be argued that the most important information a merger tree provides is the mass assembly history of a halo, i.e. the mass of their progenitors. The mass growth is the main input in galaxy formation models (e.g. Springel et al. 2001). Masses along a branch of a tree are expected to monotonically increase as time elapses, especially for main haloes. Nevertheless, as we shall see this behaviour, assumed in SAMs (Lacey & Silk 1991; White & Frenk 1991; Cole et al. 1994, 2000; Gonzalez-Perez et al. 2014; Lacey et al. 2016; Lagos et al. 2018), in fact depends on other properties of the progenitors, such as being a main or a satellite halo, and physical quantities, e.g. the distance between merging progenitors. Therefore, we do not enforce during the training process a progenitor mass increasing with time in a branch. To evaluate the fair reproduction of the mass growth in a sample of ML generated trees, we compare statistical distributions of the mass gain and loss of progenitors for merger trees with fixed number of branches. We show an example of these distributions in Fig. 4, where "real" denotes the cumulative probability obtained using merger trees from the training dataset. For a description of the combination of variables considered in the ML distributions see Table 3. From Fig. 4, we immediately note that there is an improvement in the learning process when providing the GAN model with more information of the merging events at each timestep, i.e., by including relevant variables or inputs (see shaded regions).

Another input of our GAN model is the physical distance between merging haloes, specifically we consider the distance between the centre of mass of progenitors in branches other than the main branch and that in the main branch. It should be noted that most semi-analytic models of galaxy formation do not require this variable to evolve galaxies (e.g. Somerville et al. 2008). Being able to predict this quantity would allow further physics to be included in such models; as in the case of GALFORM (Cole et al. 2000), a SAM that takes into account these distances (together with other physical parameters) to redefine the progenitor type (main or satellite halo; a variable that we have also considered) of a infalling halo (Helly et al. 2003; Jiang et al. 2014). In principle, the distance between progenitors should decrease with time as the merging event approaches, but there is no rule of thumb for the precise step in time this should occur. Hence, we construct normalised probability distributions of this distance at the snapshot before the fusion takes place, for real and ML merger trees with the same number of branches (see, e.g., Fig. 5. Note that the precise snapshot at which a merger occurs varies from 0 (dark blue) up to the last snapshot, 200 (dark red), as shown in Fig. 5, where we have normalised the distributions by the maximum peak of all curves, to facilitate comparison. In this figure, we observe that as expected, as time elapses merging haloes are found at a closer distance from each other one snapshot before the fusion. Hence, the peak of the distributions from $n_{\rm snap}$ = 0 to 200 progressively shifts to the left. We can also observe that most mergers occur at $n_{\rm snap}$ = 60 ($z \simeq 3.8$) for ROCKSTAR merger trees with 8 branches. With the above mentioned distributions, properly normalised, we construct cumulative distribution functions (CDFs), one distribution per snapshot (curves in Fig. 5), for the real and ML merger tree samples. These CDFs are suitable to compare using the KS test.

The third variable that we have considered is the progenitor type, i.e., the condition of being a main or a satellite halo. This variable is fundamental in semi-analytic models of galaxy formation, such as GALFORM (Cole et al. 2000), SAGE (Croton et al. 2016), SAG (Cora et al. 2018) and GALACTICUS (Benson 2012), among others; as it strongly influences the physical processes that take place in a galaxy, such as cooling (see e.g. Cole et al. 2000), affecting the cold gas component and the star formation rate (see e.g. Gómez et al.

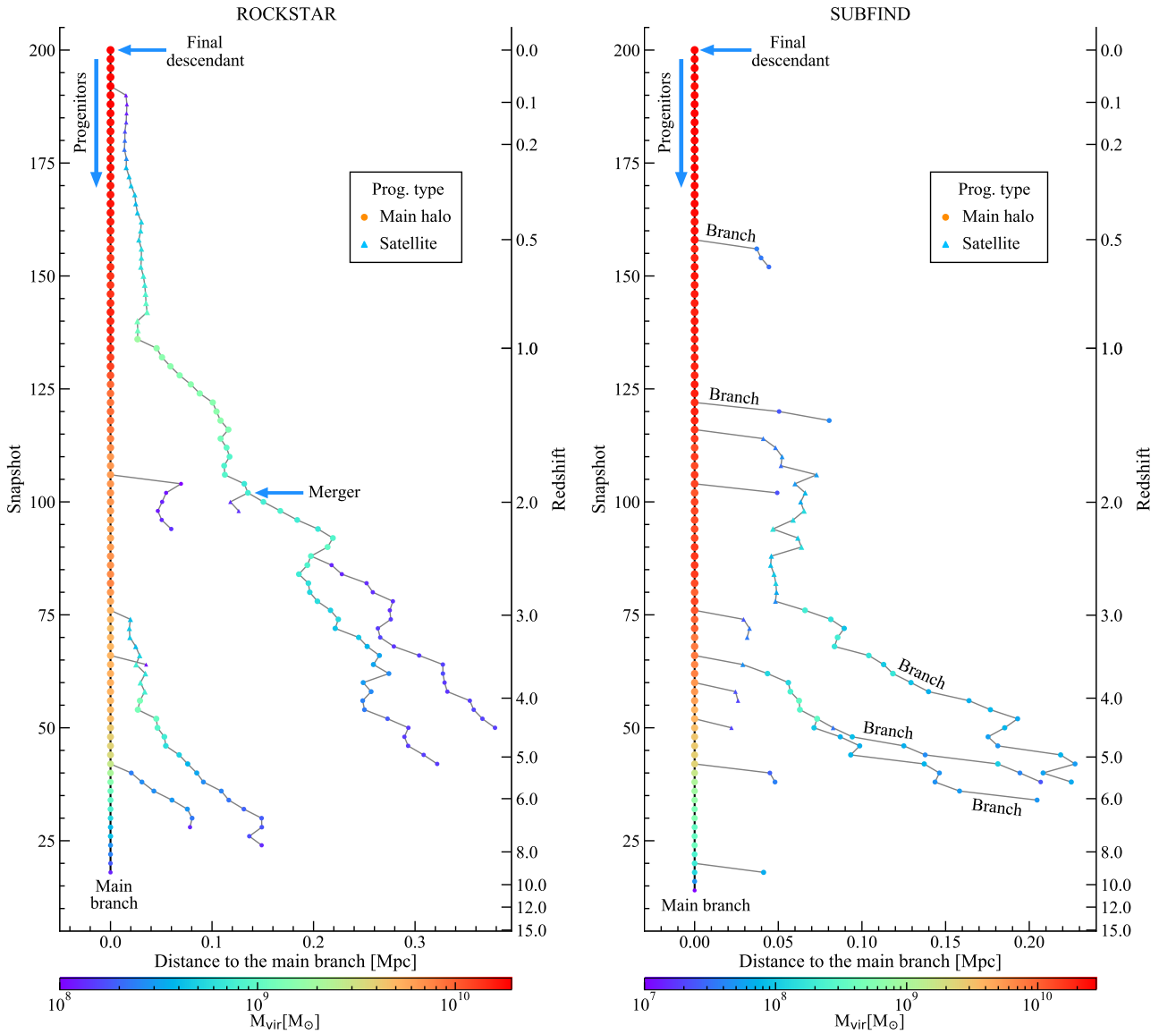


**Figure 3.** Examples of ROCKSTAR-like (left, 8 branches) and SUBFIND-like (right, 12 branches) machine learning generated merger trees in the plane snapshot (redshift) vs. distance to the main branch. Progenitors that are main haloes are denoted by circles and satellites by triangles, the colour map represents the virial mass of the halo progenitors.

2022). Halo progenitors in the main branch, with the sole exception of early snapshots, are expected to be main haloes. Conversely, progenitors in other branches can become satellites as a consequence of gravitational infall as they approach the other merging halo. The precise step in time when this occurs depends on the specific tree and also on the masses and distance of the merging progenitors (Diemand et al. 2006; Muldrew et al. 2011; Han et al. 2012; Onions et al. 2012; Elahi et al. 2011; Onions et al. 2013). Satellite haloes approach the main haloes they are going to merge with and their mass is allowed to decrease in time as the main halo grows in mass. An example of this behaviour can be found in Fig. 3. Therefore, we can evaluate the importance of including the progenitor type when reproducing its corresponding mass, by analysing the above mentioned mass gain and loss of the progenitors, but splitting each real and ML merger tree sample in two sub-samples according to the progenitor type, as illustrated in Fig. 6. Then, we proceed to compare real with ML distributions for main and satellite haloes. For that particular example, we can see that the mass gain (and loss) of the main haloes is much better reproduced by the GAN model than that of the satellites. This is probably due to the fact that most of the progenitors in a merger tree are main haloes. In this sense, we can perform an additional test and construct distributions of the number of snapshots a progenitor spends as a satellite, see, e.g., Fig. 7. This is an explicit test of the fair reconstruction of the progenitor type input. When comparing this figure with Fig. 5 of Robles et al. (2019), we note that when adding more time resolution, it becomes more difficult for the GAN model to predict in which time-step a progenitor becomes a satellite.

## 4 RESULTS

In this section, we apply the statistical measures outlined in the previous section and compare ML with real distributions using the

**Table 3.** Series of trainings performed using either SUBFIND or ROCKSTAR halo merger trees, $n_{\rm snap}$ denotes the number of snapshots (time resolution) and $n_{\rm var}$ the number of input variables, where 1 var. refers to the mass of the progenitors and the final descendant, hereafter mass, 2 vars. corresponds to trainings with either mass and distance to the main branch or mass and type of progenitor, 3 vars. refers to all the aforementioned inputs.

| Training | $n_{\rm var}$ | $n_{\rm snap}$ | Number of branches $n_{\rm br}$ | |
|---|---|---|---|---|
| | | | SUBFIND | ROCKSTAR |
| single $n_{\rm br}$ | 1 var. | 101, 201 | 6–15 | 6–13 |
| | 2–3 vars. | 101 | 6–15 | 6–13 |
| multiple $n_{\rm br}$ | 1 var. | 101, 201 | 6–10, 6–12, 6–15 | 6–10, 6–12, 6–13 |
| | 2 vars. | 101 | 6–10, 6–12, 6–15 | 6–10, 6–12, 6–13 |
| | 3 vars. | 101 | 6–10, 6–15, 6–16, 6–19 | 6–10, 6–12, 6–13 |

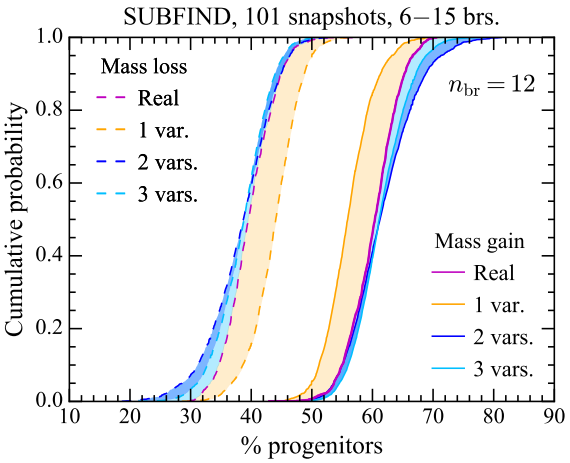


**Figure 4.** Cumulative distribution function (CDF) of the mass gain (solid) and loss (dashed) for all the progenitors in the SUBFIND merger tree dataset with 12 branches, denoted "real" (magenta). For comparison, we also show the CDFs for samples of trees generated with 1 (orange), 2 (blue), and 3 (light blue) variables, time resolution of 101 snapshots, and the GAN model trained with a dataset composed of trees with $6 \leq n_{\rm br} \leq 15$. The shaded regions depict the difference between the "real" and ML CDFs.

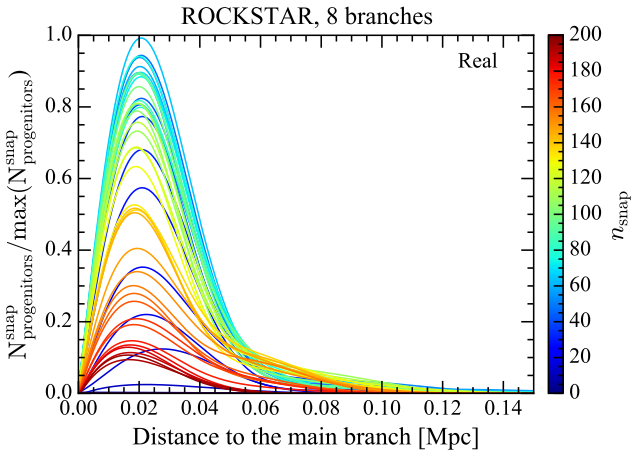


**Figure 5.** Distribution of the distance between merging progenitors, when one of them is located in the main branch, for merger trees with 8 branches identified by ROCKSTAR (denoted "real"). The distance is computed at the snapshot before the merging event takes place (colourmap). The number of progenitors at a given step in time $N^{\rm snap}_{\rm progenitors}$ is normalised by the maximum peak of all distributions.

Kolmogorov-Smirnov test. First of all, in Table 3 we summarise the series of trainings we have performed to study the relevance of each of the variables considered here to the well construction of the mass growth history of haloes, and other aspects related to the performance of our GAN model. These aspects include the quality of the merger trees generated with individual trainings, i.e., with datasets comprised of merger trees with a fixed number of branches denoted single $n_{\rm br}$ and trainings with $n_{\rm br}$ varying in a range (multiple $n_{\rm br}$). Recall that the number of branches is a fundamental parameter of the architecture of our GAN model, see Table A1; so that for our series of experiments, we select sub-samples of the initial training dataset in Fig 1 based on $n_{\rm br}$ and not on the final descendant mass, to maximise the amount of examples of different tree structures per given $n_{\rm br}$ and hence to ensure the convergence of the training process. In this sense, multiple $n_{\rm br}$ trainings are a more realistic scenario than single $n_{\rm br}$ since a set of merger trees with different numbers of branches are generated at once for final descendants with masses than span a wider range, as in cosmological simulations.

### 4.1 Mass assembly history

First, we focus on the fair reproduction of the mass assembly history of haloes and train our GAN model considering the progenitor mass as the sole input. Initially, we constructed training datasets with the maximum achievable resolution in time, i.e., $n_{\rm snap} = 201$. In Fig. 8, top panels we show the results of the KS test from the comparison of the mass gain and loss cumulative distributions (for an example of these CDFs, see Fig. 4) as dot-dashed lines. Recall that we construct CDFs for "real" and ML merger trees with a given $n_{\rm br}$, regardless if the training was performed with a single or a multiple $n_{\rm br}$ dataset. We carried out four series of trainings for each halo finder–tree builder algorithm (denoted SUBFIND and ROCKSTAR), three of them using merger trees with number of branches in the 6–10 (light blue), 6–12 (orange), 6–13 (brown) and 6–15 (magenta) ranges. Note that the maximum number of branches we have considered varies with the algorithm, $n_{\rm br} = 13$ for ROCKSTAR and $n_{\rm br} = 15$ for SUBFIND. This is due to the size of the training dataset, namely ROCKSTAR finds only $\sim$ 800 trees with 13 branches, while for SUBFIND there are $\sim$ 1800 trees with this number of branches in the training dataset, see Fig. 1. As mentioned in the previous section, $\sim$ 1000 trees for a given $n_{\rm br}$ are required to obtain well-constructed merger trees. The grey lines denoted "single $n_{\rm br}$" represent individual trainings, i.e., every point corresponds to a training with fixed $n_{\rm br}$. E.g., for SUBFIND the grey lines represent 10 independent trainings, while each magenta line was obtained with a single training. We immediately notice that the learned representation of SUBFIND merger trees always gives

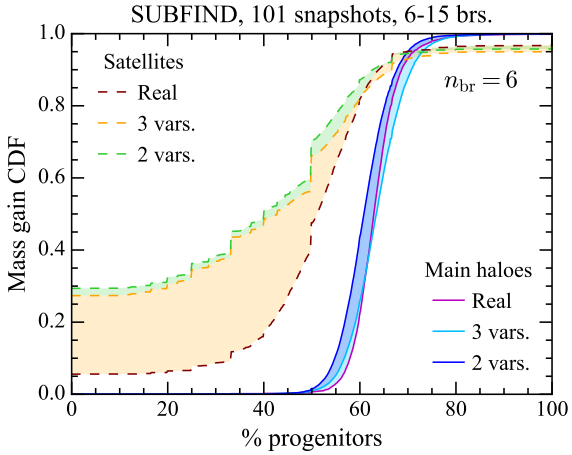


**Figure 6.** CDF of the mass gain for all the progenitors that are main haloes (solid) and satellites (dashed) in the SUBFIND merger tree dataset with 6 branches, denoted "real" (magenta). For comparison, we also show the CDFs for samples of trees generated with 2 (mass and progenitor type) and 3 variables. ML trees generated with a time resolution of 101 snapshots and a training dataset composed of trees with number of branches in the range $6 \leq n_{br} \leq 15$. The shaded regions highlight the difference between the "real" and ML CDFs.

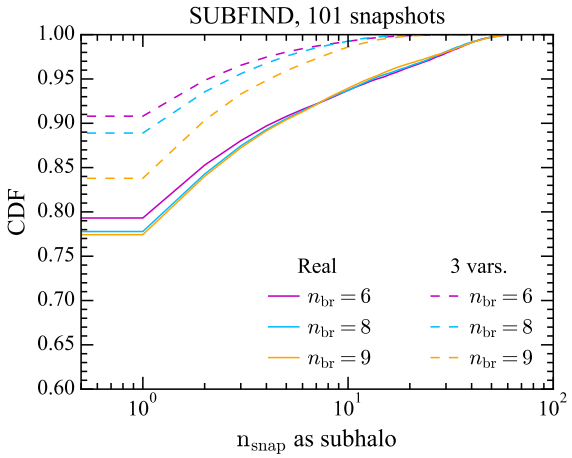


**Figure 7.** CDF of the number of snapshots that a progenitor is a satellite halo for the real and ML trees with $n_{br} = 6, 8, 9$, generated with 3 variables (individual trainings with a time resolution of $n_{snap} = 101$ were performed for every $n_{br}$ in this figure).

better results than that of ROCKSTAR. Note however that there are more merger trees identified with SUBFIND in the training database than ROCKSTAR examples for our GAN to learn to represent (see Fig. 1), and the morphology of SUBFIND and ROCKSTAR merger trees is not necessarily similar for haloes of the same mass.

As expected, reducing the temporal resolution to $n_{snap} = 101$, distributed between $z = 20$ and $z = 0$, allows us to improve the quality of the generated trees. Note that in general better values of the KS test are obtained with datasets that include trees with number of branches within a range rather than individual trainings with fixed $n_{br}$, because of the larger size of the dataset. This is more evident for trainings with $n_{snap} = 101$, compare dashed grey lines with solid lines. In both cases $n_{snap} = 201$ and $n_{snap} = 101$, the learned representation of the mass of the progenitors that better fits the real mass gain and loss distributions is that obtained with the $6 \leq n_{br} \leq 12$ dataset for both SUBFIND and ROCKSTAR ML trees. Of course, increasing the maximum number of branches in the training dataset enlarges its size and in principle improves the learning process. Although, the GAN model would have to learn to reproduce a more complex tree structure with less available examples, since in general there are less trees in the E100 simulation with larger number of branches (see Fig. 1). Consequently, there is a maximum $n_{br}$, above which the addition of merger trees to the training dataset does not improve the reproduction of the progenitor mass. Therefore, with this new tool we would be able to predict with very good precision the most abundant merger trees, i.e. those of low and intermediate mass haloes at $z = 0$ ($10^8 M_\odot < M_{vir} < 5 \times 10^{10} M_\odot$), at least for the E100 simulation. Later on, the properties of the corresponding galaxies with $5 \times 10^5 M_\odot < M_{galaxy} < 10^8 M_\odot$ could be obtained (see e.g. Gómez et al. 2022), using a SAM along with these merger trees.

Training our GAN model with different combinations of variables, as well as with a single variable, is important to understand which are the most influential variables to predict the mass growth history of haloes. In the second row of Fig. 8, we show similar trainings to those in the top panels, but performed with 2 variables: progenitor mass and distance to the main branch, for $n_{snap} = 101$. The test KS is performed only for the mass gain and loss cumulative distributions, i.e., we have evaluated if the addition of a second input, in this case the distance, helps to more accurately reproduce the progenitor mass distribution. In the third row, we introduce the progenitor type instead of the distance to the main branch as second input. By comparing the second and third row with the top panels, we see that the introduction of a second variable improves SUBFIND results for the progenitor mass, but not those of ROCKSTAR, which in fact are worse than those obtained with the progenitor mass as the only input. As mentioned in the previous section, the addition of another input implies that the GAN model needs to learn to reproduce that input too, which affects the convergence of the training process. Regarding the SUBFIND learned representation, the best results for the KS test of the mass distribution are those where the progenitor type is the additional input, this is particularly true for trees with $6 \leq n_{br} \leq 12$ (orange lines). For $6 \leq n_{br} \leq 15$ (magenta line) the trend of the KS test is to increase with the number of branches when the type is the second variable and to decrease when the distance is considered instead. Note here that the progenitor type, unlike other inputs, is a discrete variable in our GAN model, so that in principle it should be easier to learn than the distance. Once again, in both cases ROCKSTAR and SUBFIND and for both distance and type, better results are obtained when a set of merger trees with number of branches in a given range is used as training data (solid lines) instead of individual trainings with fixed $n_{br}$ (dashed grey lines).

In the bottom panels of Fig. 8, we consider the three inputs at the same time in the training process. As expected from the results with two inputs, ROCKSTAR learned representation of the progenitor mass is not improved by providing the GAN model with additional information about the progenitor type and distance to the progenitor in the main branch, with the sole exception of the results for $6 \leq n_{br} \leq 13$ (brown lines). For SUBFIND, we can see that the GAN model with three inputs, trained with halo merger trees with $6 \leq n_{br} \leq 16$ (yellow line) yields the best progenitor mass distributions, in particular for ML trees with number of branches in the range $6 \leq n_{br} \leq 11$, since there are more examples of these trees in the training dataset. These results are even better than those obtained with the progenitor mass as the only input and with mass and distance. Contrary to these cases (see panels above) the KS test tendency is to decrease with $n_{br}$ in the aforementioned range, which in general holds for all the trainings

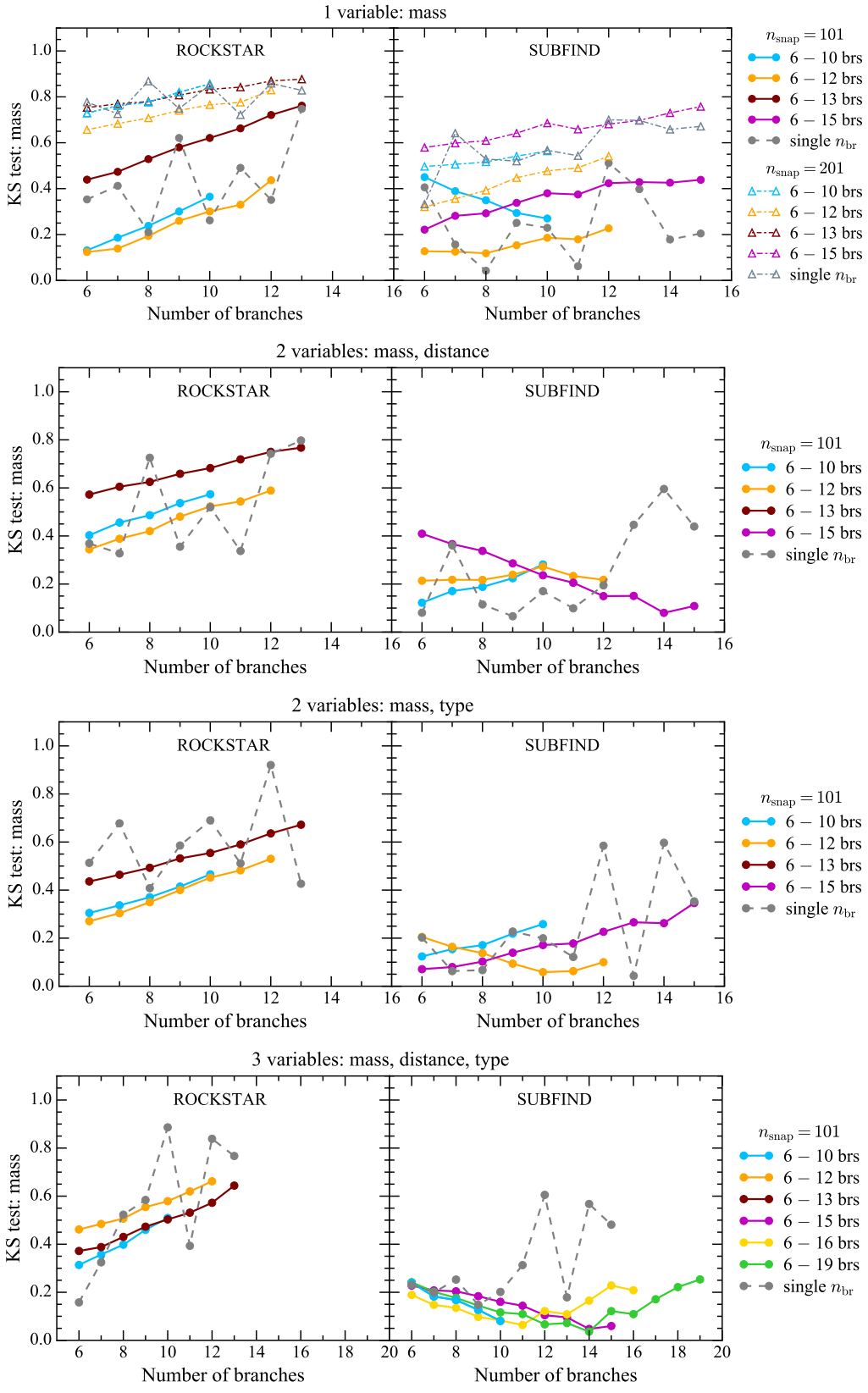


**Figure 8.** Kolmogorov-Smirnov (KS) test for the mass of the progenitors and the final descendant for merger trees generated with 1 (top), 2 (second row: mass and distance to the main branch, third row: mass and progenitor type) and 3 (bottom) input variables, GAN model trained with either ROCKSTAR (left) or SUBFIND (right) merger trees. Grey lines correspond to series of individual trainings (single $n_{\rm br}$, one training per $n_{\rm br}$) and colourful lines to single trainings performed with trees whose number of branches vary in a range (multiple $n_{\rm br}$, a single training per line).

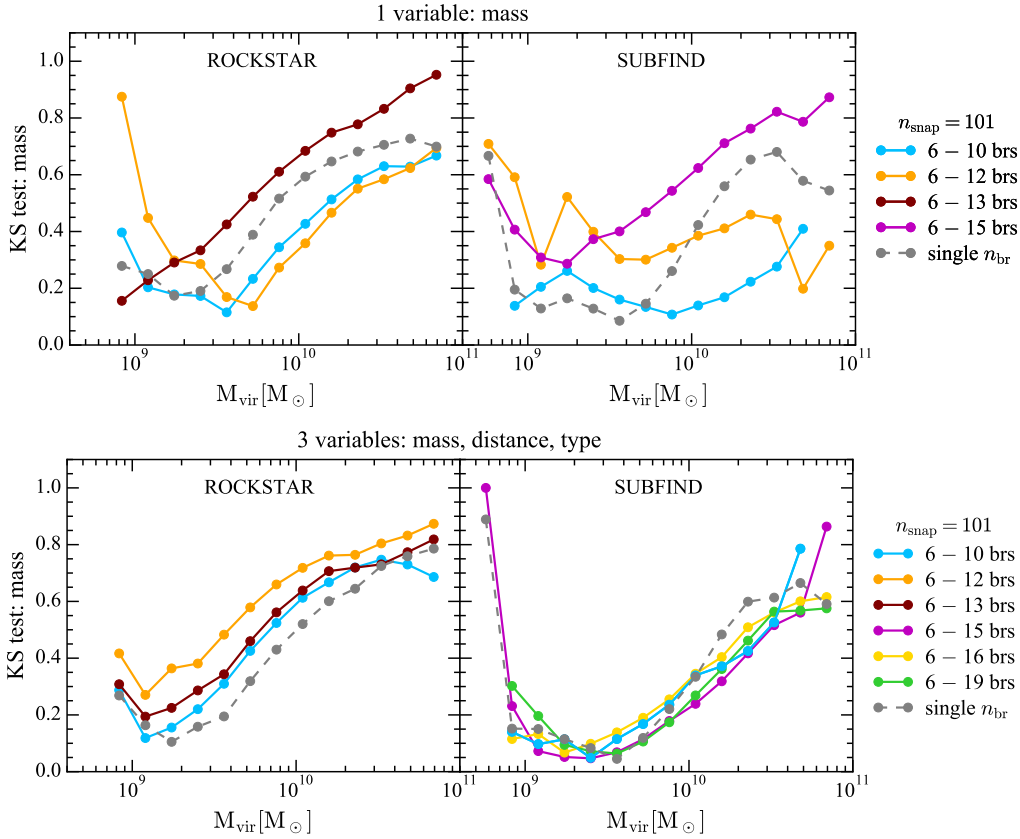


**Figure 9.** Kolmogorov-Smirnov (KS) test for the mass of the progenitors and the final descendant for merger trees generated with 1 (top), and 3 (bottom) input variables, GAN model trained with either ROCKSTAR (left) or SUBFIND (right) merger trees. CDFs were constructed for bins of the mass of the final descendant. Grey lines correspond to series of individual trainings (single $n_{br}$, one training per $n_{br}$) and colourful lines to single trainings performed with trees whose number of branches vary in a range (multiple $n_{br}$, a single training per line).

with multiple $n_{br}$. The second best results are those obtained with $6 \leq n_{br} \leq 19$ (green line), for which the KS test increases for $n_{br} \geq 15$. Note that to achieve such a number of branches, we have tuned the size of the column- and row-wise filters, the input of the decoder and the batch size (see Table A1). This is the maximum number of branches for which the GAN total loss function converges. Note however that the restriction on $n_{br}^{max} = 19$ is imposed not only by the amount of trees in the training dataset, but mainly by memory constraints. For ROCKSTAR, on the other hand, $n_{br} = 13$ is the maximum number of branches that in this case is due to the amount of trees available for training, as previously mentioned.

Finally, in Fig. 9 we show a global measure of the goodness of the reproduction of the mass gain-loss distribution of the progenitors across branches by ML merger trees of haloes of a given virial mass, regardless of the specific number of branches. To that end, we construct CDFs of the above mentioned quantity as in Fig. 4, but instead of comparing samples of merger trees with a given $n_{br}$, we select "real" and ML trees by the mass of the final descendant. In Fig. 9, we depict the value of the test KS at the centre of each $M_{vir}$ bin. From this figure, we immediately notice that as in Fig. 8, the best results are obtained for trainings carried out with 3 variables (bottom right panel) for SUBFIND ML trees, and that for main haloes with masses in the $10^9 M_\odot \lesssim M_{vir} \lesssim 10^{11} M_\odot$, the $6 \leq n_{br} \leq 15$ (magenta line) multiple $n_{br}$ training yields the best fit to the real trees. Note that for the first and last $M_{vir}$ bins, the value of the KS test increases; this is due to the fact there are less merger trees of main haloes with masses in those ranges in the training dataset (see middle panel of Fig. 1). For ROCKSTAR ML trees, on the other hand, we observe an improvement on the KS test for merger trees generated with 3 variables (bottom left panel) with respect to those trained with the mass as the sole input (top left panel), for single $n_{br}$ training. This was not evident in our results in Fig. 8. Moreover, when the GAN model receives 3 input variables the best fit to the "real" mass gain-loss distribution is obtained with series of individual trainings per number of branches in the training dataset from $n_{br} = 6$ to 13 (dashed grey line). Although, better results can be obtained with only one input variable for $M_{vir} \gtrsim 3 \times 10^9 M_\odot$; see orange and light blue lines in the top left panel of Fig. 9.

### 4.2 Progenitor type

Next, we analyse the importance of discriminating main from satellite haloes when comparing the progenitor mass gain and loss distributions. As we have seen, this is particularly relevant for SUBFIND. In Fig. 10, we show the values of the Kolmogorov-Smirnov test for main (left panels) and satellite (middle panels) haloes for series of trainings with 3 variables. For both ROCKSTAR and SUBFIND, the KS test for the satellite mass distribution remains approximately flat, especially for $6 \leq n_{br} \leq 19$ (green line). This is true even for individual trainings (grey lines). Conversely, for main haloes the KS test increases with $n_{br}$. It is worth noting that most of the progenitors in a merger tree are expected to be main haloes. Hence a better

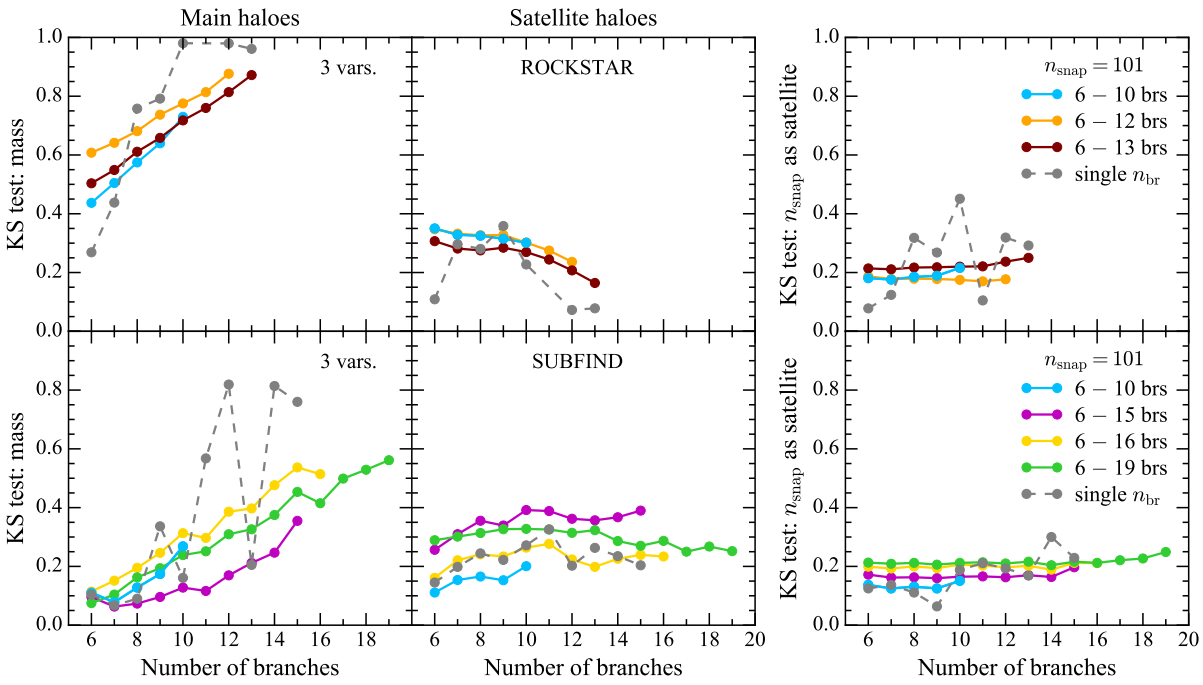


**Figure 10.** Kolmogorov-Smirnov (KS) test for the mass of the progenitors per type: main haloes (left) and satellites (middle) and for the number of snapshots a progenitor spend as a satellite (right), for merger trees generated with 3 variables, trainings performed with either ROCKSTAR (top) or SUBFIND (bottom) merger trees. Grey lines represent series of individual trainings (single $n_{\rm br}$, one training per $n_{\rm br}$) and colourful lines single trainings performed with trees whose number of branches vary in a range (multiple $n_{\rm br}$, a single training per line).

reconstruction of the mass gain and loss distributions of this fraction of the progenitors will improve the KS test of the full sample, shown in Fig. 8. This is particularly evident in the case of the trainings with 2 variables (mass and type), not shown in Fig. 10, and in the case of ROCKSTAR ML merger trees; compare orange, light blue and brown lines in the first and second rows of Fig. 8 with their respective counterparts in the third row of the same figure.

In addition, we observe that with 3 inputs, the reconstruction of the progenitor mass for main haloes identified by ROCKSTAR is improved for the $6 \leq n_{\rm br} \leq 13$ training (brown line) with respect to that for $6 \leq n_{\rm br} \leq 12$ (orange line) that yielded the best results with 1 and 2 variables when considering the full progenitor sample (see Fig. 8, left panels in the first and third rows). For satellite haloes, the $6 \leq n_{\rm br} \leq 13$ training also minimises the KS test for both 2 and 3 variables. For SUBFIND, on the other hand, the best fit for main haloes is obtained with the $6 \leq n_{\rm br} \leq 15$ dataset (magenta line), while for satellites the corresponding best fit is achieved with $6 \leq n_{\rm br} \leq 10$ (light blue). Therefore, comparing the main halo result with the bottom right panel of Fig. 8 where better fits are obtained with the $6 \leq n_{\rm br} \leq 16$ dataset (yellow line), we can infer that the inclusion of the distance between merging progenitors plays a greater role in SUBFIND learned representation of the progenitor mass, when the GAN is trained using datasets that contain 3 variables, than in that of ROCKSTAR.

Adding input channels to our GAN model implies they are also outputs of the neural network, as such they contribute to the GAN total loss and affect the convergence of the training process. Consequently, their fair reproduction validated against the training dataset should also be assessed. In the right panels of Fig. 10, we evaluate the fair reproduction of the progenitor type with 3 variables, using CDFs of the number of snapshots a progenitor is a satellite, see e.g., Fig. 7. It is worth noting that when considering only mass and progenitor type as inputs, the best results are obtained, in general, when performing individual trainings with merger tree samples with a fixed number of branches, for both ROCKSTAR and SUBFIND. Adding the distance to the main branch as an extra input, improves the KS test for trainings with multiple $n_{\rm br}$ and the opposite occurs for single $n_{\rm br}$ trainings in most cases. For SUBFIND, the number of snapshots that a progenitor is a satellite is better reproduced by the training performed with the $6 \leq n_{\rm br} \leq 10$ dataset (light blue line), followed by the results for $6 \leq n_{\rm br} \leq 15$ (magenta line). The KS test of the latter is very similar to the best ROCKSTAR result, obtained using merger trees with $6 \leq n_{\rm br} \leq 12$ (orange line).

Despite our results in Fig. 10 favour the SUBFIND learned representation, it is worth remarking that progenitors identified by SUBFIND suffer from main halo-satellite switching issues (Behroozi et al. 2015; Poole et al. 2017). As a merging process evolves and the progenitors involved approach each other, for instance a main halo-satellite pair (or more progenitors) identified as such at a given snapshot, at the next step in time they can be misidentified. This is the main halo is identified as a satellite, while the satellite becomes the main halo. This can occur several times as the progenitors move closer. Since we do not correct for this issue, it is transferred to the learned representation.

### 4.3 Distance to the main branch

Finally, we evaluate the fair reproduction of the distance of a progenitor to that in the main branch. In this case, we compare CDFs of this distance at the snapshot before the merger takes place. This snapshot varies from 0 to the last snapshot, so that there is a probability distribution per snapshot, see e.g. Fig. 5. Each distribution is normalised before constructing the corresponding CDF. As a result, we obtain as many values of the KS test as snapshots in the matrix representa-

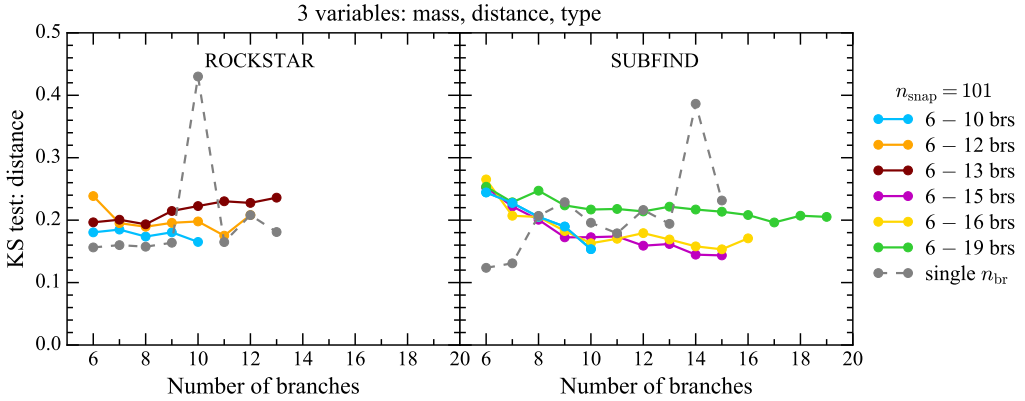


**Figure 11.** Snapshot averaged Kolmogorov-Smirnov (KS) test for the distance of the progenitors to that in the main branch at the snapshot before the fusion occurs, for merger trees generated with 3 variables, trainings performed with either ROCKSTAR (left) or SUBFIND (right) merger trees.

tion of the merger trees. In Fig. 11, we show these values averaged over this number of snapshots for series of trainings with 3 inputs. The best fits when comparing averaged values are found for trainings with multiple $n_{\rm br}$, specifically $6 \leq n_{\rm br} \leq 10$ for ROCKSTAR (light blue line) and $6 \leq n_{\rm br} \leq 15$ for SUBFIND (magenta line), which unlike the case of the mass of the progenitors are very similar to the results obtained with single trainings with a fixed number of branches. Note however that when including the standard deviation of the KS test in Fig. 11, there is in general good agreement among all trainings, especially for the 3 variables case. When considering only mass and distance as input variables, there is a more noticeable difference in the value of the KS test among different trainings. In addition, we find that the distance to the main branch is better reproduced when the GAN model is trained with merger trees constructed using ROCKSTAR. The difference being more striking for trainings with 2 variables.

### 4.4 Length of the main branch

As we have shown in section 4.1, trainings with 101 snapshots render a more accurate picture of the mass growth of haloes, when comparing ML merger trees with those in the training dataset. However, this shortens the length of the main branch, which, as we shall see, is not necessarily a shortcoming. In Fig. 12, we show a histogram of the main branch length for ML trees generated in multiple $n_{\rm br}$ trainings, with $n_{\rm snap} = 101$ and three input variables. We only show ML trees generated during two trainings, specifically those carried out with the $6 \leq n_{\rm br} \leq 12$ dataset for ROCKSTAR (top panel) and the $6 \leq n_{\rm br} \leq 15$ dataset for SUBFIND (bottom panel), since all multiple $n_{\rm br}$ trainings with 3 variables yield similar results. These are the samples of ML trees used to construct probability distribution functions for the KS tests in Figs. 8–11. We also show the corresponding histograms for the samples of merger trees from the training dataset that were used to construct the CDFs of the 'real' trees. We can see that despite having reduced the number of snapshots, our GAN model is still able to produce merger trees that trace progenitors back in time (high in redshift) as further as the trees in the training dataset (compare the peaks of the real with the ML trees, redshift axis). The peak of both distributions is still close to $z \simeq 8$. This is due to the fact that, when reducing the temporal resolution of the merger trees in the training dataset from $n_{\rm snap} = 201$ to 101, we skipped intermediate snapshots along the complete halo growth history.

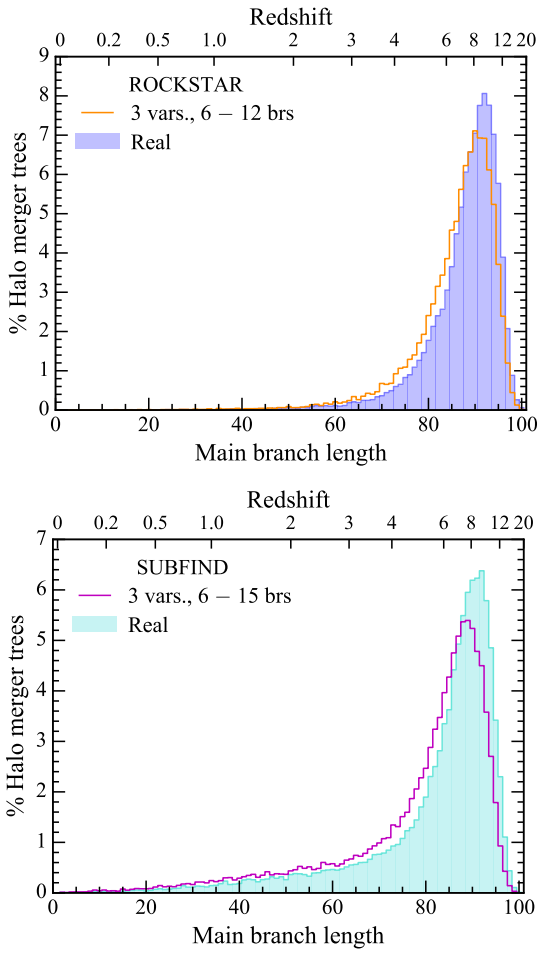


**Figure 12.** Histogram of the length of the main branch measured in number of snapshots for machine learning (ML) merger trees, generated with 3 variables, a temporal resolution of $n_{\rm snap} = 101$ and using the ROCKSTAR (top) training dataset that contains trees with 6 − 12 branches (orange), and the dataset of SUBFIND (bottom) trees with 6 − 15 branches (magenta). For comparison, we also show the corresponding histograms for samples of similar size of 'real' trees with the same number of branches. The redshift axis is also shown for reference.

## 5 CONCLUSIONS

Halo merger trees encode the mass assembly history of haloes. A complete set of merger trees is a necessary ingredient for semi-analytic models (SAMs) of galaxy formation and evolution. These models are best suited to compare theory with observations, especially with data from forthcoming surveys, since they are computationally inexpensive when compared to cosmological hydrodynamical simulations. Currently, the most common method to construct merger trees is based on computationally intensive N-body simulations. In this paper, we have shown that Generative Adversarial Networks (GANs), a deep learning framework, can be used to successfully learn and build halo merger tree structures from cosmological N-body simulations. Their only limitations being memory resources and a sufficiently large training dataset. The former can be circumvented by reducing the temporal resolution, while the latter depends on the the mass of the final descendant and the number of branches of the tree, features that ultimately depend on the volume of the dark matter only (DMO) simulation and the halo finder–tree builder algorithm. The main advantage of the machine learning (ML) generated trees is that they are produced with a modest computational expense in a short computation time, while preserving the best features of merger trees from cosmological N-body simulations. This opens up the possibility to use ML trees, along with SAMs, to simulate large samples of galaxies comparable to those to be obtained by upcoming surveys, in a comparatively much shorter computational time than that required by cosmological hydrodynamical simulations.

We trained our GAN model with merger trees from the EAGLE simulation suite, constructed using two halo finder–tree builder algorithms: SUBFIND–D-TREES and ROCKSTAR–ConsistentTrees. We conducted a series of experiments designed to test the capabilities of our neural network model and to study the importance of including other variables in the training process aside from the mass of the progenitors and the final descendant. These additional input variables are the progenitor type, i.e. the condition of being a main halo or a satellite, and the distance of a merging progenitor to that in the main branch. Remarkably, the same GAN architecture with few changes in its parameters can be used to learn to generate merger trees obtained with both algorithms and with and without keeping fixed the number of branches.

To evaluate the fair reproduction of the main properties of halo merger trees, we have compared probability distributions of equally large samples of ML generated trees with those from the EAGLE DMO simulation with co-moving cubic box length of 100 Mpc, "real" trees, using the Kolmogorov-Smirnov test. The examined features include mass gain and loss of progenitors along a branch, number of snapshots a progenitor is a satellite, and distance between two merging progenitors at the snapshot before the merger event, and the interplay of the above. Note that more properties of the merging progenitors can also be learned just by adding the corresponding quantities as additional input channels and tuning the parameters of our GAN model.

In broad terms, our GAN model learns to generate halo merger trees identified by both pairs of halo finder–tree builder algorithms. When evaluating the quality of both learned representations, we find that our GAN model is more successful in generating SUBFIND-like merger tree structures than in accurately reproducing the statistical features of the bulk of "real" ROCKSTAR merger trees. This is clearly influenced by the fact that there are more SUBFIND merger trees available for training per given number of branches in our training database. We also find that including the progenitor type as an input of the training process helps our GAN model to predict with more precision the mass gain and loss of the progenitors that are main haloes, especially for SUBFIND-like ML merger trees. Note that this is particularly relevant for ML trees intended to be used with SAMs of galaxy formation, like GALFORM, SAGE, SAG and GALACTICUS, as both mass growth and progenitor type are arguably the two most important inputs for galaxy evolution and to derive galactic properties such as the hot and cold gas mass fractions and the star formation rate.

Adding the distance of a progenitor to that in the main branch also improves SUBFIND learned representation of the halo mass assembly history. It is worth noting than when introducing an extra input, our GAN model must also learn to reproduce this new variable. In this regard, when evaluating the well reproduction of the statistical distributions of the distance of a progenitor to that in the main branch, we find that ROCKSTAR learned representation follows more accurately the distributions of the "real" sample. Although, this distance is not an essential ingredient of SAMs, some of them like GALFORM take these distances into account to redefine the progenitor type of an infalling halo; hence, being able to predict this variable is crucial for these SAMs.

Finally, our GAN-based halo merger tree generation framework can be used to construct merger histories of haloes of low and intermediate mass, for which there is, in general, a large number of samples available in cosmological simulations. Massive haloes, on the other hand, have a more intricate assembly history and appear less frequently. As mentioned above, ML merger trees can be used in SAMs to model galaxy formation and evolution, keeping in mind the lack of rare populations corresponding to massive haloes, which can be alleviated by training our GAN model using merger trees from simulations of larger volume than that of the EAGLE simulation suite. According to the specific requirements of a given SAM, more input variables such as position, peculiar velocity and velocity dispersion of the progenitors, among others, can be easily added and learned by our GAN model.

## ACKNOWLEDGEMENTS

SR was partially supported by MINECO/FEDER (Spain) under grant PGC2018-094975-C2, by the UK STFC grant ST/T000759/1, and by the Australian Research Council through the ARC Centre of Excellence for Dark Matter Particle Physics, CE200100008. SR also acknowledges funding from the European Union's Horizon 2020 Research and Innovation Programme under the Marie Sklodowska-Curie grant agreement No. 734374 (LACEGAL-RISE) for a secondment at the Pontificia Universidad Católica de Chile. JG acknowledges support from CONICYT project Basal AFB-170002, funding from the CONICYT PFCHA/DOCTORADO BECAS CHILE/2019 21191147, and the Predoctoral contract "Formación de Personal Investigador" from the Universidad Autónoma de Madrid (FPI-UAM, 2021). ARR acknowledges support from the Brazilian National Council for Scientific and Technological Development (CNPq) under grant No. 307425/2017-7. ARR was also with University of Campinas and University of Reykjavik while working on this research. This research was undertaken using the LIEF HPC-GPGPU Facility hosted at the University of Melbourne. This Facility was established with the assistance of LIEF Grant LE170100200. This work also used computer facilities at the Universidad Autónoma de Madrid and the Geryon computer at the Center for Astro-Engineering UC, part of the BASAL PFB-06, which received additional funding from QUIMAL 130008 and Fondequip AIC-57 for upgrades. The authors would also like to thank the CYTED AgIoT Project (520rt0011), CORFO

CoTH2O Consortium, and Proyecto Asociativo UDP "Plataformas Digitales como modelo organizacional", for their support.

## DATA AVAILABILITY

The data underlying this article will be shared on reasonable request to the corresponding author. Merger trees from the EAGLE simulation suite obtained with SUBFIND have been publicly released (McAlpine et al. 2016).

## REFERENCES


Aragon-Calvo M. A., 2019, MNRAS, 484, 5771
Avila S., et al., 2014, MNRAS, 441, 3488
Barchi P. H., et al., 2020, Astronomy and Computing, 30, 100334
Behroozi P. S., Wechsler R. H., Wu H.-Y., 2013a, ApJ, 762, 109
Behroozi P. S., Wechsler R. H., Wu H.-Y., Busha M. T., Klypin A. A., Primack J. R., 2013b, ApJ, 763, 18
Behroozi P., et al., 2015, MNRAS, 454, 3020
Bengio Y., Courville A., Vincent P., 2013, IEEE Trans. Pattern Anal. Mach. Intell., 35, 1798
Benson A. J., 2012, New Astron., 17, 175
Benson A. J., Bower R., 2010, MNRAS, 405, 1573
Bond J. R., Cole S., Efstathiou G., Kaiser N., 1991, ApJ, 379, 440
Bower R. G., Benson A. J., Malbon R., Helly J. C., Frenk C. S., Baugh C. M., Cole S., Lacey C. G., 2006, MNRAS, 370, 645
Carlberg R. G., Couchman H. M. P., Thomas P. A., 1990, ApJ, 352, L29
Cavanagh M. K., Bekki K., Groves B. A., 2021, MNRAS, 506, 659
Cole S., 1991, ApJ, 367, 45
Cole S., Aragon-Salamanca A., Frenk C. S., Navarro J. F., Zepf S. E., 1994, MNRAS, 271, 781
Cole S., Lacey C. G., Baugh C. M., Frenk C. S., 2000, MNRAS, 319, 168
Cora S. A., et al., 2018, MNRAS, 479, 2
Crain R. A., et al., 2015, MNRAS, 450, 1937
Croton D. J., et al., 2006, MNRAS, 365, 11
Croton D. J., et al., 2016, ApJS, 222, 22
De Lucia G., Kauffmann G., White S. D. M., 2004, MNRAS, 349, 1101
Dieleman S., Willett K. W., Dambre J., 2015, MNRAS, 450, 1441
Diemand J., Kuhlen M., Madau P., 2006, ApJ, 649, 1
Elahi P. J., Thacker R. J., Widrow L. M., 2011, MNRAS, 418, 320
Elahi P. J., et al., 2013, MNRAS, 433, 1537
Gómez J. S., Padilla N. D., Helly J. C., Lacey C. G., Baugh C. M., Lagos C. D. P., 2022, MNRAS, 510, 5500
Gonzalez-Perez V., Lacey C. G., Baugh C. M., Lagos C. D. P., Helly J., Campbell D. J. R., Mitchell P. D., 2014, MNRAS, 439, 264
Goodfellow I. J., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y., 2014, arXiv e-prints, p. arXiv:1406.2661
Guo Q., et al., 2011, MNRAS, 413, 101
Han J., Jing Y. P., Wang H., Wang W., 2012, MNRAS, 427, 2437
Hatton S., Devriendt J. E. G., Ninin S., Bouchet F. R., Guiderdoni B., Vibert D., 2003, MNRAS, 343, 75
Helly J. C., Cole S., Frenk C. S., Baugh C. M., Benson A., Lacey C., 2003, MNRAS, 338, 903
Ho M., Rau M. M., Ntampaka M., Farahi A., Trac H., Póczos B., 2019, ApJ, 887, 25
Jiang F., van den Bosch F. C., 2014, MNRAS, 440, 193
Jiang L., Helly J. C., Cole S., Frenk C. S., 2014, MNRAS, 440, 2115
Katz N., Hernquist L., Weinberg D. H., 1992, ApJ, 399, L109
Kauffmann G., White S. D. M., 1993, MNRAS, 261, 921
Kauffmann G., White S. D. M., Guiderdoni B., 1993, MNRAS, 264, 201
Kauffmann G., Colberg J. M., Diaferio A., White S. D. M., 1999, MNRAS, 303, 188
Kim E. J., Brunner R. J., 2017, MNRAS, 464, 4463
Kingma D. P., Ba J., 2014, arXiv e-prints, p. arXiv:1412.6980
Knebe A., et al., 2011, MNRAS, 415, 2293
Knebe A., et al., 2013, MNRAS, 435, 1618
Krizhevsky A., Sutskever I., Hinton G. E., 2012, in Adv. Neural Inf. Process. Sys. (NeurIPS). pp 1097–1105
Lacey C., Cole S., 1993, MNRAS, 262, 627
Lacey C., Silk J., 1991, ApJ, 381, 14
Lacey C. G., et al., 2016, MNRAS, 462, 3854
Lagos C. d. P., Tobar R. J., Robotham A. S. G., Obreschkow D., Mitchell P. D., Power C., Elahi P. J., 2018, MNRAS, 481, 3573
Lee J., et al., 2014, MNRAS, 445, 4197
McAlpine S., et al., 2016, Astronomy and Computing, 15, 72
Muldrew S. I., Pearce F. R., Power C., 2011, MNRAS, 410, 2617
Munari E., Monaco P., Sefusatti E., Castorina E., Mohammad F. G., Anselmi S., Borgani S., 2017, MNRAS, 465, 4658
Okamoto T., Nagashima M., 2001, ApJ, 547, 109
Onions J., et al., 2012, MNRAS, 423, 1200
Onions J., et al., 2013, MNRAS, 429, 2739
Planck Collaboration et al., 2014, A&A, 571, A16
Poole G. B., Mutch S. J., Croton D. J., Wyithe S., 2017, MNRAS, 472, 3659
Qu Y., et al., 2017, MNRAS, 464, 1659
Ricciardelli E., Franceschini A., 2010, A&A, 518, A14
Robles S., Gómez J. S., Ramírez Rivera A., González J. A., Padilla N. D., Dujovne D., 2019, arXiv e-prints, p. arXiv:1906.09382
Roukema B. F., Quinn P. J., Peterson B. A., Rocca-Volmerange B., 1997, MNRAS, 292, 835
Schaller M., et al., 2015, MNRAS, 451, 1247
Schaye J., et al., 2015, MNRAS, 446, 521
Somerville R. S., Primack J. R., 1999, MNRAS, 310, 1087
Somerville R. S., Hopkins P. F., Cox T. J., Robertson B. E., Hernquist L., 2008, MNRAS, 391, 481
Springel V., White S. D. M., Tormen G., Kauffmann G., 2001, MNRAS, 328, 726
Srisawat C., et al., 2013, MNRAS, 436, 150
Wadekar D., Villaescusa-Navarro F., Ho S., Perreault-Levasseur L., 2021, ApJ, 916, 42
Wang Y., et al., 2016, MNRAS, 459, 1554
White S. D. M., Frenk C. S., 1991, ApJ, 379, 52


## APPENDIX A: GAN ARCHITECTURE

As mentioned in section 3.2, the GAN architecture is based on the model introduced by Robles et al. (2019), whose layout is implemented using a combination of CNNs layers with either column- or row-like filters. We keep the same activation function between layers, namely the Exponential Linear Unit (ELU), and calculate the losses (classic GAN loss and reconstruction losses per input channel) with cross entropy with logits and sigmoid activation. These choices produced the best results in Robles et al. (2019). The parameters of the discriminator, encoder and decoder are given in Table A1. Note that we have indicated a range for the kernel (k) size of the filters. The specific size depends on the particular training dataset, namely the halo finder–tree builder algorithm used to construct them, ROCKSTAR or SUBFIND, the maximum number of branches and time resolution considered. We have found that for our GAN model to successfully learn to construct SUBFIND-like merger trees, a larger kernel size for all filters is always required. Both, the discriminator and generator, are trained with batches of merger trees, using the Adam optimiser. The optimal batch size varies in the range of 100–200 samples and depends mainly on the maximum number of branches in the training dataset, subject to memory resources. Finally, the input size of the decoder (output of the encoder) also changes according to the number of columns in the matrix representation.

**Table A1.** Layout and parameters of the discriminator, encoder and decoder networks, where $n_{\rm snap}$ denotes the maximum number of snapshots (temporal resolution), $n_{\rm br}$ the maximum number of branches and $n_{\rm var}$ the number of variables, k is the kernel structure and s the number of strides.

| Discriminator | | |
|---|---|---|
| Layer | Parameters | Output shape |
| Input | | ($n_{\rm snap}$,$n_{\rm br}$,$n_{\rm var}$) |
| Conv2D | k:(1,5–9) s:1 | ($n_{\rm snap}$,$n_{\rm br}$,16) |
| ELU | | ($n_{\rm snap}$,$n_{\rm br}$,16) |
| Conv2D | k:(1,5–9) s:1 | ($n_{\rm snap}$,$n_{\rm br}$,32) |
| ELU | | ($n_{\rm snap}$,$n_{\rm br}$,32) |
| Conv2D | k:(3–5,1) s:1 | ($n_{\rm snap}$,$n_{\rm br}$,64) |
| ELU | | ($n_{\rm snap}$,$n_{\rm br}$,64) |
| Conv2D | k:(3–5,1) s:1 | ($n_{\rm snap}$,$n_{\rm br}$,128) |
| ELU | | ($n_{\rm snap}$,$n_{\rm br}$,128) |
| Conv2D | k:(3–5,1) s:1 | ($n_{\rm snap}$,$n_{\rm br}$,256) |
| ELU | | ($n_{\rm snap}$,$n_{\rm br}$,256) |
| Flatten | | ($n_{\rm snap} \times n_{\rm br} \times 256$) |
| FC | | (1) |
| Encoder | | |
| Layer | Parameters | Output shape |
| Input | | ($n_{\rm snap}$,$n_{\rm br}$,$n_{\rm var}$) |
| Conv2D | k:(1,5–9) s:1 | ($n_{\rm snap}$,$n_{\rm br}$,16) |
| ELU | | ($n_{\rm snap}$,$n_{\rm br}$,16) |
| Conv2D | k:(1,5–9) s:1 | ($n_{\rm snap}$,$n_{\rm br}$,32) |
| ELU | | ($n_{\rm snap}$,$n_{\rm br}$,32) |
| Conv2D | k:(3–5,1) s:1 | ($n_{\rm snap}$,$n_{\rm br}$,64) |
| ELU | | ($n_{\rm snap}$,$n_{\rm br}$,64) |
| Conv2D | k:(3–5,1) s:1 | ($n_{\rm snap}$,$n_{\rm br}$,128) |
| ELU | | ($n_{\rm snap}$,$n_{\rm br}$,128) |
| Conv2D | k:(3–5,1) s:1 | ($n_{\rm snap}$,$n_{\rm br}$,256) |
| ELU | | ($n_{\rm snap}$,$n_{\rm br}$,256) |
| Flatten | | ($n_{\rm snap} \times n_{\rm br} \times 256$) |
| FC | | (100–300) |
| Decoder | | |
| Layer | Parameters | Output shape |
| Input | | (100–300) |
| FC | | ($n_{\rm snap} \times n_{\rm br} \times 256$) |
| ELU | | ($n_{\rm snap} \times n_{\rm br} \times 256$) |
| Deconv2D | k:(5–9,1) s:1 | ($n_{\rm snap}$,$n_{\rm br}$,128) |
| ELU | | ($n_{\rm snap}$,$n_{\rm br}$,128) |
| Deconv2D | k:(5–9,1) s:1 | ($n_{\rm snap}$,$n_{\rm br}$,64) |
| ELU | | ($n_{\rm snap}$,$n_{\rm br}$,64) |
| Deconv2D | k:(1,3–5) s:1 | ($n_{\rm snap}$,$n_{\rm br}$,32) |
| ELU | | ($n_{\rm snap}$,$n_{\rm br}$,32) |
| Deconv2D | k:(1,3–5) s:1 | ($n_{\rm snap}$,$n_{\rm br}$,16) |
| ELU | | ($n_{\rm snap}$,$n_{\rm br}$,16) |
| Deconv2D | k:(1,3–5) s:1 | ($n_{\rm snap}$,$n_{\rm br}$,$n_{\rm var}$) |
| Sigmoid | | ($n_{\rm snap}$,$n_{\rm br}$,$n_{\rm var}$) |

## APPENDIX B: EXAMPLES OF GENERATED MERGER TREES

In Fig. B1, we show more examples of generated merger trees in the classic tree representation, snapshot vs. branch. These halo merger trees were obtained after training our GAN model with three variables in the same multiple $n_{\rm br}$ training as those in Fig. 3, namely $6-10\,n_{\rm br}$ (ROCKSTAR, light blue lines in the left-bottom panels of Figs. 8 – 11) and $6-16\,n_{\rm br}$ (SUBFIND, yellow lines in the right-bottom panels of Figs. 8 – 11) and time resolution $n_{\rm snap} = 101$. Due to the more complex structure of these trees, we do not show the corresponding plots in the plane snapshot vs. distance to the main branch. Note that in the ROCKSTAR-like merger tree (left), there are two branches which have subbranches, in contrast to that of SUBFIND-like tree (right). In the particular halo mass range we have focused due to the selection criterion for the training dataset, i.e. $10^9 {\rm M}_\odot \lesssim {\rm M}_{\rm vir} \lesssim 10^{11} {\rm M}_\odot$, we note that on average ROCKSTAR merger trees tend to have more subbranches than those identified with SUBFIND as well as longer branches (excluding the main branch).

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

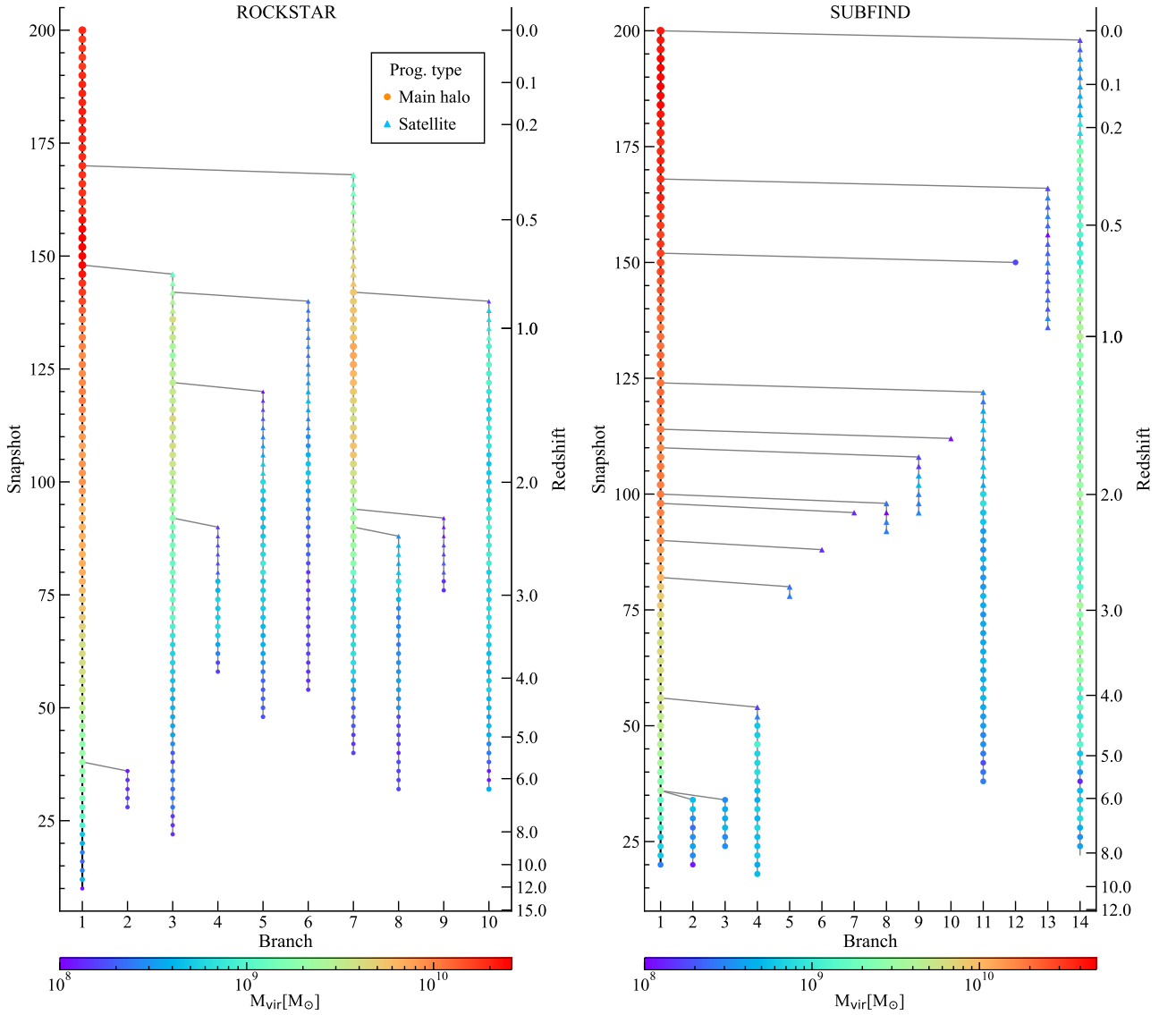


**Figure B1.** Additional examples of ROCKSTAR-like (left, 10 branches) and SUBFIND-like (right, 14 branches) ML generated merger trees, showing the tree structure (snapshot vs. branch). Progenitors that are main haloes are denoted by circles and satellites by triangles, the colour map represents the virial mass of the halo progenitors.