**Critic to paper 2 and 3**

The original halo merger tree generation papers, [2] and [3] propose a generative adversarial network for generating halo merger trees. However, there are some issues with their method, documentation, argumentation and reproduction of their experiments. Analysis and proof of the statements in this section are in the 5 - error with papers notebook.

**Reproduction**

First of all, there are no code available for their proposed method and model, which makes it hard to reproduce their results. Reproducing the results is even harder when there is little documentation and no specifications regarding important details of the model. There are lacking documentation and specifics regarding data processing like normalization methods and data analysis. For example, there is no mention of how big the training data set is, just that it must be “sufficient enough”, and that around 1000 merger trees is needed for a given number of branches for decent generations of merger trees with the same number of branches. Furthermore, there are no discussion about the possible issues with the different variables, like for example the extremely low distance values. One aspect of the data that I found strange is the subhalo variable, which is a categorical variable represented as floats. It would be natural to one-hot encode this variable, but in the papers, they keep the original subhalo variable with 0.0, 0.5 and 1.0 as the categorical values without any remarks about the variable. Is there a reason that keeping the values as floats is better? Maybe there is some “closeness” which is better kept with the floats, which seems implausible since a satellite halo is closer to a main halo than the absence of a halo. However, there is lacking documentation and argumentations for the decisions regarding data processing. And while I am on the topic of the subhalo variable, how did they manage to get all subhalo prediction to be exactly 0.0, 0.5 or 1.0 with a regular DCGAN with a sigmoid activation function? It seems unfeasible, so if they manipulated the values of the variable in some way, why isn’t that documented?

**Activation function choice**

Speaking of the sigmoid function, it seems like a weird choice in such a generative problem where separating zero values and nonzero values is a crucial part of the generative structure. I find it particularly strange since the sigmoid function technically only approaches zero asymptotically. I had an issue with this when looking at the data and variables, trying to reconstruct the data using an autoencoder. My reconstructions wouldn’t have any zeros, only values extremely close to zero. Therefore, the sigmoid function seems like an odd choice for the last layer of the generator. I would like to understand why they picked it, or what they did to work around it (get exactly 0.0 instead of values extremely close to 0.0) if they did anything. In fact, paper 2 states that they had issues where there were no progenitor present, but the mass value was a low number close to zero, but not zero. I believe that is an issue.

In fact, when analyzing the generated merger trees values from paper 3, using the sigmoid function as the last layer in the decoder (which is stated in appendix A), raises some serious discontinuity questions regarding the solution from the paper. Given the last layer in the decoder is the sigmoid functions, generated values will be between 0 – 1. With that in mind, it is alarming that the generated mass values from paper 3 are ranging from 0 – 10.289. Without any preprocessing or processing of the generated data, this should not be possible. Therefore, there must have been some processing of the data that is not mentioned.

A table with text and numbers

Description automatically generated**Model architecture – shape mismatch**

Continuing on reproductivity of the paper and the results, the model architecture in appendix A also seems to have some flaws. Managing to keep the two first input channels consistent while using the convolutions they add seems impossible. I am referring to the width and height of the image, namely the 29 and 6 values. Convolution layers are supposed to down sample the image unless you carefully pick its parameters, since adding a convolution will affect the shape of the input. The first thing you learn about convolution layers is the dimensionality change formula, how padding, dilation, kernel size and stride affects the input shape. In this example, 29 is the rows (time) of the image and 6 is the columns (branches), adding a convolution layer with kernel size (1, 3) would change the shape to like this formula:

Figure 1. encoder from paper 2

**A group of black text

Description automatically generated**

Figure 2. pytorch convolution dimension formula

Which means 29 🡪 29 + 2 \* 0 – 1 \* (1 – 1) – 1 + 1 = 29, but

6 🡪 6 + 2 \* 0 – 1 \* (3 – 1) - 1 – 1 = 4. This would continue for all convolution layers, which means the model architecture they present is inconsistent. The output shapes are completely off, which affects the output of the flatten function, which affects how they reshape the image in the generator. This inconsistency on top of the lack of documentation, argumentation and specifics makes it super hard to recreate their model and results. Proof of this is in the 5 - error with papers notebook.

**Loss function**

Furthermore, the same is true for loss function, why are they using cross entropy reconstruction loss instead of usual MSE reconstruction loss? And what loss are they using in the GAN architecture? I would assume the original loss from Goodfellow et al.? I believe it should be stated, even with an explicit loss formula.

**Evaluation – KS test**

The evaluation metrics used for the generated trees are somewhat a mystery. Why use the Kolmogorov-Smirnov (KS) test? There is no reasoning for the choice. The KS test is used for one-dimensional probability distributions, is it the best metric for evaluation of the generated samples? I believe an argument should be presented of why the KS test is so central in the evaluation of the generated trees.

More information and details regarding the KS test should be presented. There are multiple aspects of the KS test that is crucial to how to interpret the results that is not mentioned by any of the papers. I am referring to KS section, which explains these aspects of the KS test and their importance. The papers only mention and use the KS statistic, but there are no remarks regarding how sample size affects the KS statistic, or how this could be a weakness of the KS test. There are no comments about the shortcomings of the KS statistic and the KS test. And most importantly, there are no mention about the p-value, null hypothesis or how to evaluate if the two distributions are separable or could come from the same distributions. When none of these crucial features of the KS test is mentioned, is KS test a valid evaluation metric for this specific task? Are the good results justified to a reasonable degree if important part of the evaluation metric is left out?

One last thing I wish was mentioned in the papers is the difficulties of training a GAN. I don’t mean they should focus on the topic, since the papers focus on astronomical and cosmological topics, but if you are generating merger trees using a GAN, which is notoriously hard to train, the training process or aspects of it should be mentioned. Loss scaling, number of epochs trained, latent space testing, hyperparameters like learning rate, discriminator/generator training ratio, or how the training progressed. Nothing regarding these training aspects was mentioned, which again makes it hard to reproduce.

**Discontinuities in the generated merger trees of paper 3**

Lastly, I had access to the generated merger trees from paper 3 and have analyzed the generated trees. The results from my analysis, which is in the 5 - error with papers notebook, shows a lot of discontinuities in the result from the paper that is not mentioned at all. First of all, both the mass and distance variables predict extremely low nonzero values that are not supposed to be generated (“off-values”). In fact, for the distance variable a total of 9.0898% of all generated values are “off-values”, which is a shockingly high number. That means over 9% of the generated distance values illustrate the presence of progenitors, when they are not actually supposed to be there. However, for the distance variable, most of these “off-values” is in the main branch. Remember, the distance is a measure from the main branch, so distance should be 0 in the main branch. But when that is said, there are 0.8920% of all distance generated values who are “off-values” and not in the main branch. But for mass, the story is completely different, a total of 13.8759% of all mass generated values are “off-values”, which is shockingly high. Predicting that many progenitors that are not there is an issue. Combining the distance and mass “off-vales” have multiple results that are alarming:

* in some cases, both mass and distance have “off-values” in the same pixel spots, which means both channels technically say it’s a progenitor present, when it is not.
* mass predict a progenitor present in the beginning of the main branch, so it adds one timestep to the main branch which is bad.
* most of the "off-values" are location-wise close to the “good” progenitors, however there are multiple issues:
  + the "off-values" extend branches with more than 9 values
  + the "off-values" creates discontinuities in branches, that is gaps between progenitors in a branch
  + the "off-values" have created new branches that are not supposed to exist. with "off-values" in both the mass and distance variable

Adding this up creates some issued, and I find it alarming that this is not mentioned more or delt with in some (documented) way.

Moving on to the subhalo variable, the result of the analyzing is shocking. A total of 2.4721% of all subhalo generated values are NOT exactly 0.0, 0.5 or 1.0. These “off-values” ranges from 0.01 to 0.99, and represents all values in between, rounded to two decimals, except 0.13, 0.15, 0.24, 0.26, 0.27, 0.28, 0.29, 0.24 and 0.68. That means all values between 0.0 and 1.0 rounded up to two decimals except 9 values are present, that is 89 unique values (!!!) and almost 2.5% of all generated subhalo values. How would they deal with subhalo values of 0.25 or 0.75? or other values? I am really surprised that this is not mentioned and delt with and raises some foundation for questioning the results of the paper.

My continuity criteria in DATA continuity section, which refers to the structure of the merger trees which is inspired from paper 1, 2, 3 states that there should be no gaps between branches, no gaps within a branch, only one last descendant in the main branch in the last row, distance variable should be 0.0 in the main branch and all three variables should be zero and non-zero in the same pixel spots (except distance in main branch). Analysis regarding continuity of generated merger trees done in notebook 08 - consistency evaluation and analysis, show that 0% of the generated merger trees form paper 3 fulfill all structure continuity criteria. This result is shocking, and even more alarming that this is not mentioned.

**Overall**

The main issues with the two papers are reproducibility of the experiments and results and lack of documentation and argumentation of steps and decisions for several specifics regarding data processing and the model.

Even though some documentation regarding the process of their proposed solution is lacking, and that the generated merger trees from paper 3 are not 100% consistent in terms of structure, their result and provided constructed halo merger trees look very promising. They contain some discontinuities as described above, but overall, they contain the important aspects of halo merger trees, the generated merger trees only have minor inconsistencies, especially if the discontinuities described above are delt with. Ignoring the discontinuities mentioned, there are no gaps between progenitor within a branch, and there are no empty branches between branches containing progenitors, which is desired features. Additionally, they manage to generate complex merger trees with many branches of varying length, which is excellent.