DRAFT VERSION DECEMBER 14, 2017
Typeset using LaTeX **twocolumn** style in AASTeX61

GENERATING DARK MATTER HALO CATALOGS FOR COSMOLOGICAL ZOOM-IN SIMULATIONS

ETHAN O. NADLER (05818244)[1, *] AND DAVID THOMAS (05551845)[1, 2, †]

[1] *Kavli Institute for Particle Astrophysics and Cosmology and Department of Physics, Stanford University, Stanford, CA 94305, USA*
[2] *Stanford Institute for Computational and Mathematical Engineering, Stanford, CA 94305, USA*

ABSTRACT

We study dark matter halo populations in hydrodynamic and dark matter only (DMO) zoom-in simulations of Milky Way-mass systems using discriminative and generative machine learning algorithms. We show that random forest classification cannot reliably distinguish halos in hydrodynamic and DMO simulations; thus, we argue that generative algorithms trained on DMO simulations can produce halo populations that are consistent with hydrodynamic results. We develop a Gaussian mixture model (GMM) and a generative adversarial network (GAN) for this task by training on halos from a suite of DMO simulations. Our GMM and GAN produce halo feature distributions that are in good agreement with DMO data and with two hydrodynamic simulations. We conclude that GANs are a promising technique for efficiently generating realistic halo populations.

Keywords: dark matter – galaxies: abundances – galaxies: halos – methods: numerical

1. MOTIVATION

High resolution “zoom-in" simulations of Milky Way-size systems are an important tool in cosmological studies of the Milky Way’s satellite galaxy population (e.g., see Wetzel et al. 2016). In particular, these simulations resolve the evolution and properties of dark matter halos, which are related to the observable properties of satellite galaxies. However, most extant simulations only include dark matter, while relatively few include hydrodynamic physics such as star formation and supernova feedback due to the significant computational costs associated with simulating these processes. Thus, there is a significant incentive to develop algorithms that can efficiently produce halo catalogs that are consistent with hydrodynamic results.

In this project, we study whether generative machine learning algorithms trained on dark matter only (DMO) simulations can fulfill this role. We begin by showing that random forest classification cannot reliably distinguish halos in hydrodynamic and DMO simulations. This result implies that the distributions of halo features in these types of simulations are very similar, which we quantify with Kolmogorov-Smirnov tests. We then develop a Gaussian Mixture model (GMM) and a Generative Adversarial Network (GAN) to learn the distributions of halo features in a suite of DMO zoom-in simulations, and we study the halo populations generated by these algorithms. We show that the halo feature distributions generated by our GMM and GAN agree fairly well with the true distributions, and we argue that GANs can be used to generate entire halo populations that are consistent with snapshots from cosmological simulations.

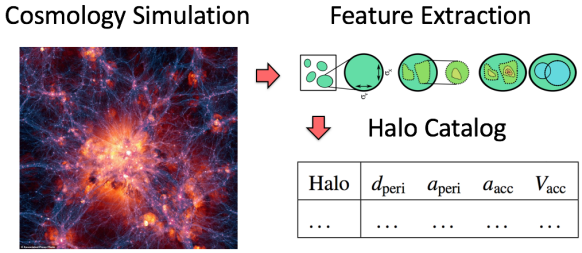

Figure 1. Data extraction process. For each cosmology simulation, we extract relevant features for every halo that exists at a particular temporal snapshot and we package these halos into a data vector.

Machine learning applications in cosmology are relatively new; for example, Schawinski et al. (2017) explored using generative models to recover features in imaging data and Mustafa et al. (2017) used GANs to create weak lensing convergence maps. In addition, one of the authors of this work (EN) recently examined halo disruption in zoom-in simulations using ML classification techniques (Nadler et al. 2017).

2. DATA

We study halos from hydrodynamic and DMO simulations of two Milky Way-mass systems, referred to as `m12i` and `m12f`, which were simulated as part of the Feedback in Realistic Environments (FIRE) project (Wetzel et al. 2016), and also from the suite of 45 DMO zoom-in simulations of Milky

* enadler@stanford.edu
† dthomas5@stanford.edu

Training Data and Features	# Halos (Hydro, DMO)	Test Set Accuracy	Hydro Test Set Accuracy	OOB Score	Mixup?
m12i, m12f ($\mathcal{X}_4$)	556, 1120	74%	42%	73%	No
m12i, m12f, DMO5 ($\mathcal{X}_4$)	556, 3222	90%	34%	89%	No
m12i, m12f, DMO5 ($\mathcal{X}_4$)	3222, 3222	81%	67%	79%	Yes

Table 1. Performance metrics for random forest classifiers trained on halos from the m12i and m12f hydrodynamic and DMO and from five independent DMO simulations ("DMO5;" Mao et al. (2015)). The first column lists the dataset, the second column lists the associated number of hydro and DMO training examples, and the third and fourth columns respectively list the percentage of *all* halos in the test set and the percentage of *hydro* halos in the test set that are classified correctly. The fifth column lists the out-of-bag classification score, and the sixth column indicates whether the mixup technique is applied to the hydro training data.

Way-mass systems presented in Mao et al. (2015). These simulations include a list of all halos at each temporal snapshot, a list of properties (e.g., position, mass, size) for each halo, and the relationships between different halos (e.g., most massive progenitor, most massive descendant, and so on). We now describe our method for processing the snapshots of these simulations in order to extract relevant halo features; a visual representation of our data extraction process is shown in Figure 1.

We extract the following features for each halo from the final temporal snapshot of each simulation referenced above:

- $d_{\rm peri}$: the minimum distance from the center of the system, calculated over a halo's entire orbital history;
- $a_{\rm peri}$: the cosmological scale factor (a proxy for the time) at which $d_{\rm peri}$ occurs;
- $a_{\rm acc}$: the scale factor at which a halo enters the system's virial radius, which is roughly its outer surface;
- $V_{\rm acc}$: internal circular velocity at accretion, which corresponds to a halo's mass.

We only consider halos with peak maximum circular velocity $V_{\rm peak} > 10$ km s^{-1} to ensure that we study well-resolved objects[1]. The features defined at accretion encode information about halos before they have been significantly altered by dynamical effects; for example, halos typically lose large amounts of mass after accretion, which decreases their internal circular velocity. Meanwhile, the pericentric features capture the most important orbital information for each halo. We refer to the set of four features defined above as $\mathcal{X}_4$.

3. HYDRODYNAMIC VS. DMO CLASSIFICATION

In this section, we use random forest (RF) classification to determine whether halos from hydrodynamic and DMO simulations are distinguishable; we refer to these as "hydro" and "DMO" halos, respectively. Since we eventually aim to generate halo catalogs that are consistent with hydrodynamic simulations by training on DMO halos, it would be *encouraging* if halos from these different types of simulations are difficult to distinguish.

A random forest is a collection of decision trees, each of which is trained on a random sample of the training data using a random subset of the input features at each split in the learning process. For our training objective, we minimize the Gini impurity of the RF prediction, which is defined as

$$I_G(p) = 1 - \sum_{i=1}^{2} p_i^2, \tag{1}$$

where p_i is the fraction of halos in the training set labeled with class i, which is either "hydro" or "DMO" for our application. Thus, the input to our RF is a vector of halos with features $\mathcal{X}_4$ and the output is a vector of "hydro" and "DMO" labels. Note that the RF prediction for a given halo is the majority vote of the tuned decision trees.

We use the RF algorithm in Scikit-Learn (Pedregosa et al. 2011) to classify halos. To train our classifier, we use the GridSearchCV function to search the space of random forest hyperparameters, including the number of decision trees and the maximum number of features used by each tree, and we select the ones that yield the highest out-of-bag (OOB) score[2] averaged over ten folds of the training data. First, we train our classifier on the hydro and DMO halos from the m12i and m12f simulations using a 75%/25% training-test split. We list our results in Table 1; in particular, we report the number of hydro and DMO halos in the training set, the percentage halos in the test set that are identified correctly by our RF classifier, the corresponding OOB score, and the percentage of *hydro* halos in the test set that are classified correctly. The optimal OOB score for our four-property classifier is 73%, but its accuracy for hydro halos in the test set is only 42%, which indicates that the RF cannot differentiate between hydro and DMO halos.

It is possible that the RF model described above does not learn robustly because of the relatively small size of the m12i and m12f training set. Thus, we also experiment with including five hosts from the Mao et al. (2015) DMO zoom-in suite. However, since the resulting training set is heavily skewed towards DMO halos, this training set might bias our model. Indeed, we find that the RF overfits the training set

[1] This is essentially equivalent to a mass threshold, i.e., we only study halos that host more than a critical number of simulation particles.

[2] The OOB score indicates the percentage of halos from the *training* set that the RF classifies correctly when each tree does not vote on halos in its own training set.

and performs poorly on hydro halos in the test set when it is trained on `m12i`, `m12f`, and five additional DMO simulations (see Table 1). We therefore employ the *mixup* technique (Zhang et al. 2017) to create artificial hydrodynamic training data by interpolating between `m12i` and `m12f` training examples. In particular, we create artificial hydro samples $(\tilde{x}, \tilde{y})$ by computing

$$\tilde{x} = \lambda x_i + (1-\lambda) x_j, \quad (2)$$

$$\tilde{y} = \Theta(\lambda y_i + (1-\lambda) y_j - 0.5), \quad (3)$$

where $\lambda \sim \text{unif}(0,1)^4$, $x_i,\ x_j \in \mathcal{X}_4$ are feature vectors for distinct training examples, and $\Theta(x)$ is the Heaviside step function[3]. The third row of Table 1 shows that applying mixup improves the hydro accuracy to 67%, which suggests that RF classification still performs poorly after accounting for the skewed nature and limited size of our training set. Thus, we conclude that hydrodynamic and DMO halo feature distributions are inherently similar. We confirm this by performing Kolmogorov-Smirnov (KS) tests for the `m12i` and `m12f` hydro and DMO halo feature distributions using the `ks_2samp` function in Scipy (Jones et al. 2001), which returns the KS statistic and the two-tailed p-value for two distributions. We find two-tailed p-values that are ≤ 0.2 for all of the features in $\mathcal{X}_4$ (indeed, the p-value is ~ 0.05 for $V_{\rm acc}$), which implies that hydro and DMO halo features are consistent with being drawn from the same distribution. Thus, hydrodynamic and DMO simulations cannot be differentiated on a halo-by-halo basis, which implies that generative models trained on DMO simulations should be able to produce halo catalogs that are consistent with hydrodynamic results.

4. GENERATING HALO CATALOGS

Having shown that hydrodynamic and DMO halo feature distributions are statistically similar, we develop two generative models in order to produce realistic halo catalogs.

4.1. *Gaussian Mixture Model*

We use a Gaussian mixture model (GMM) for our baseline generative algorithm. A GMM is a probabilistic model which assumes that the input data are generated from a mixture of a finite number of Gaussian distributions. The mixture probabilities, means, and covariances are then determined by maximizing the likelihood that the data are drawn from the GMM model. We use the expectation-maximization algorithm to iteratively increase this likelihood by repeatedly applying of the standard expectation (E) and maximization (M) steps.

We train our GMM on 2×10^4 halos from the suite of 45 DMO simulations described above (Mao et al. 2015) using the feature set $\mathcal{X}_4$ and a 75%/25% training-test split. We explore several models with different numbers of mixture components; increasing the number of components generally leads to a better fit to the true distributions, which we quantify using KS tests for each feature dimension. We choose 10 mixture components, which results in both a reasonably complex model and feasible computation times. To sample a halo from the trained GMM, we draw its mixture component using the learned mixture probabilities and we draw its features from the corresponding multivariate Gaussian distribution. We summarize our GMM results in Section 4.3.

4.2. *Generative Adversarial Network*

We also develop a generative adversarial network (GAN; Goodfellow et al. 2014) to produce realistic halo catalogs. We use `TensorFlow` (Abadi et al. 2015) to construct the generator and discriminator, which we denote by G and D; we implement these as fully connected neural networks with Leaky ReLU activation functions and batch normalization before each layer. We use a linear activation function for the final layer of G, which produces a 4-dimensional output corresponding to the feature set $\mathcal{X}_4$, and we use a sigmoid function for the final layer of D, which produces a number $D(x) \in (0,1)$. We find that using batch normalization, which processes each feature distribution $\mathcal{X}_{4,i}$ such that it has zero mean and unit variance, is critical for training the GAN; without this technique, the model is difficult to train and the loss functions do not converge.

Algorithm 1 summarizes our GAN training method. We train using minibatch gradient descent with the following optimization objective:

$$\min_G \max_D (\mathbb{E}_{x\sim P_{\rm data}}[\log D(x)] + \mathbb{E}_{z\sim P_{\rm latent}}[\log(1-D(G(z)))]). \quad (4)$$

We calculate expectation values by drawing samples from the true data distribution for x and from a "latent" uniform random distribution for z, i.e., $z \sim \text{unif}(-1,1)^4$. We first pretrain the generator for 3.5×10^3 iterations by minimizing the mean and variance of the generated distributions with respect to the batch normalized data using the ADAM optimization

Algorithm 1: Minibatch stochastic gradient descent algorithm for our GAN.

for *number of training iterations* **do**

- Sample minibatch of m samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from latent space Z according to $p_G(z) = \text{unif}(-1,1)^4$;
- Sample minibatch of m samples $\{x^{(1)}, \ldots, x^{(m)}\}$ from data-generating distribution $p_{\rm data}(x)$;
- Update the discriminator by stochastic gradient descent:

$$\theta_D = \theta_D - \nabla_{\theta_D} \frac{\alpha_D}{m} \sum_{i=1}^{m} \left[-\log D(x^{(i)}) - \log(1 - D(G(z^{(i)}))) \right];$$

- Update the generator by stochastic gradient descent:

$$\theta_G = \theta_G - \nabla_{\theta_G} \frac{\alpha_G}{m} \sum_{i=1}^{m} \log(1 - D(G(z^{(i)}))).$$

end

[3] Note that this is a slight variant of the mixup technique presented in Zhang et al. (2017), since we force the $\tilde{y}$ labels to remain discrete.

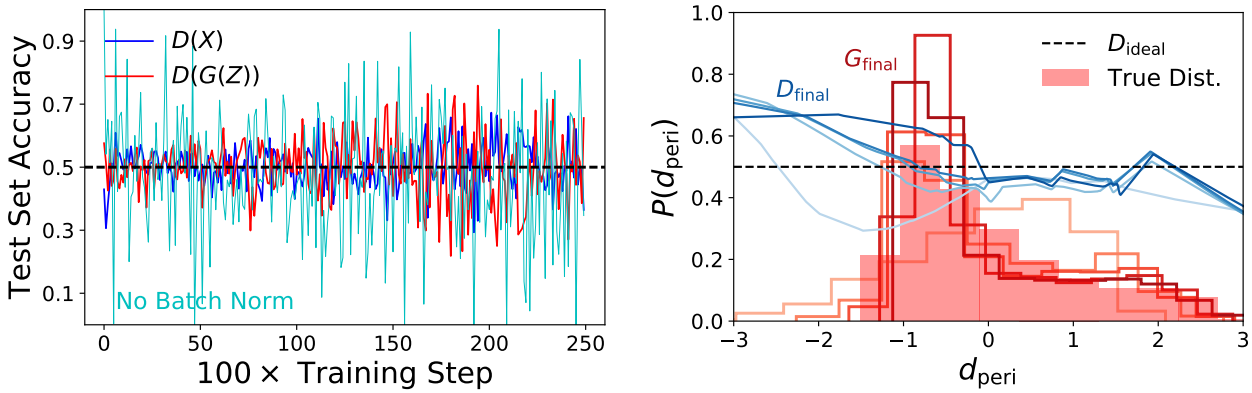

Figure 2. *Left:* Discriminator accuracy on the test set (blue) and on generated values (red) for our GAN. The cyan line shows the test set accuracy for the same model *without* batch normalization. *Right:* Values of the discriminator D (blue) and histograms of the generator G (red) for the halo feature $d_{\rm peri} \in \mathcal{X}_4$. The solid red histogram is the true $d_{\rm peri}$ distribution, and the dotted black line shows the ideal discriminator. The color gradients illustrate G and D over the course of the training, with darker colors corresponding to later training iterations.

algorithm (Kingma & Ba 2014)[4]. We then run ADAM for 2.5×10^4 training steps with learning rate α_G (α_D) for G (D) using the optimization objective in Equation 4. In practice, we *minimize* the negative of the expression for D in Equation 4, which improves training stability. As in Section 4.1, our dataset consists of 2×10^4 halos from the Mao et al. (2015) suite; we use a 75%/25% training/test set split and a minibatch size of 500 halos, which corresponds to $\sim 5\%$ of our training set size. Our results are not sensitive to the number of training steps or the minibatch size.

Our GAN contains the following hyperparameters:

- $n_{\rm layer,G} = n_{\rm layer,D}$: number of fully connected hidden layers in G and D, which we set equal for simplicity;
- $N_G = \beta\, N_D$: number of hidden layers in each layer of G and D, where $\beta > 1$;
- $\alpha_D = \gamma\, \alpha_G$: learning rates for G and D, where $\gamma > 1$.

We introduce β and γ since we find that the generated distributions are unstable unless i) G is *more complex* than D and ii) the discriminator learning rate α_D is *larger* than the generator learning rate α_G. If these requirements are not satisfied, we find that training often leads to "mode collapse," in which the generator singles out a small region of each feature space and produces values that are almost entirely confined to this region. This occurs when the discriminator dominates, i.e., when it classifies all generated examples outside of the mode collapse region as fake. We find that increasing the complexity of G relative to D by setting $N_G > N_D$ and training G more quickly than D by setting $\alpha_G < \alpha_D$ mitigates mode collapse.

To select the optimal values for these hyperparameters, we train models corresponding to all combinations of the hyperparameters shown in Figure 3. We fix $N_D = 16$ and $\alpha_G = 5 \times 10^{-4}$ to expedite the grid search. We select the

Layers	$[2, 3, 4]$
N_G/N_D	$[1, 2, 5]$
α_D/α_G	$[2, 5, 10]$

Figure 3. GAN hyperparameter grid search. We scan all combinations of these hyperparameters for the set that yields the largest minimum two-tailed p-value versus the training distributions.

hyperparameters that yield generated distributions with the *largest minimum two-tailed p-value* versus the training distributions, where the minimum is computed over all of the feature dimensions every 100 training steps[5]. In particular, we select the hyperparameters Θ that satisfy

$$\max_{\Theta} \min_{i,j} (P(G_j(Z)_i, \mathcal{X}_{4,i})), \tag{5}$$

where $P(X, Y)$ indicates the two-tailed p-value of two distributions X and Y and G_j refers to the generator evaluated at the j-th training epoch (or the $100 \times j$-th training step). We also compute the hyperparameters that yield the largest *maximum* two-tailed p-value, but we find that the resulting model performs poorly in the feature dimensions that do not correspond to this maximal value. Although we search for hyperparameters by evaluating the GAN every 100 training steps, we run the full 25,000 training steps after selecting the optimal model to avoid overfitting. Our grid search yields 3-layer networks with $N_G/N_D = 5$ (i.e., $N_G = 80$, $N_D = 16$) and $\alpha_D/\alpha_G = 2$ (i.e., $\alpha_D = 10^{-3}$, $\alpha_G = 5 \times 10^{-4}$).

We plot the discriminator accuracy for our GAN in the left panel of Figure 2; we also show the result for the same hyperparameters without batch normalization to highlight the

[4] We find that implementing pre-training prevents mode collapse due to "unlucky" fluctuations early in the training process. The number of pre-training steps does not significantly affect our results.

[5] Note that we pre-train the generator and use a fixed random seed, so the initial p-values are the same for every model we consider.

Model	$\|\|\mu_{model} - \mu_{hydro}\|\|_2$	$\|\|C_{model} - C_{hydro}\|\|_2$	AD
MG	0.4268	0.8268	2.4922
GMM	0.3984	0.8686	2.3993
GAN	0.4000	1.5419	2.4504

Table 2. Comparison metrics for the halo feature distributions produced by our generative models versus those from the `m12i` and `m12f` hydrodynamic simulations. The first column lists the model, and the second, third, and fourth column list the mean, covariance, and AD metrics defined in Section 4.3.

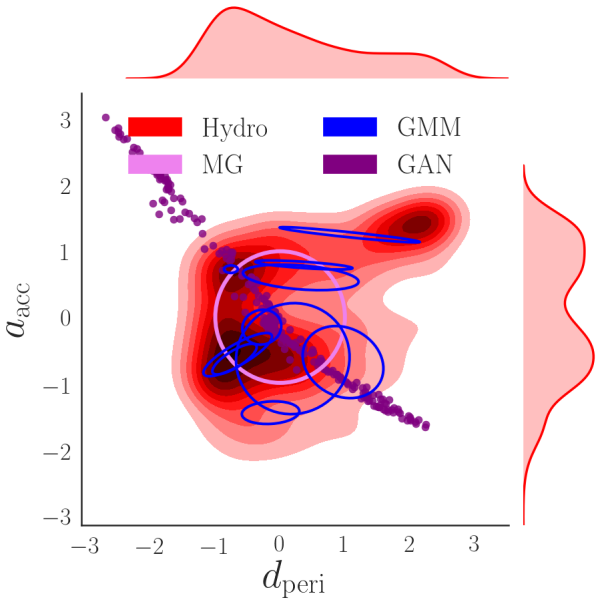

Figure 4. Joint and marginal a_{acc} and d_{peri} distributions for different generative models. The red contour shows the true joint distribution, the pink ellipse shows the 2σ confidence region for our MG model, the blue ellipses show the 2σ confidence regions for our GMM, and the purple points show 300 samples generated by our GAN.

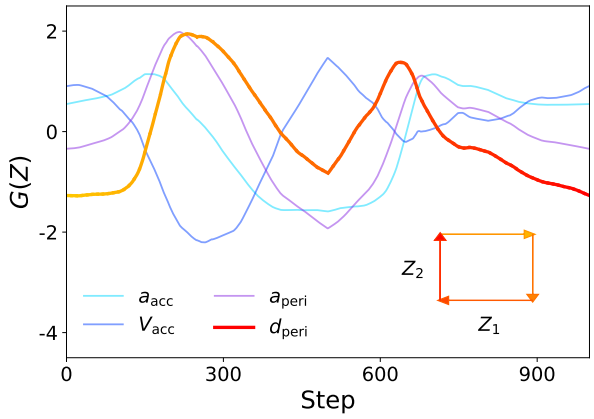

Figure 5. Values of the $\mathcal{X}_4$ halo features generated by our GAN for a uniform path through the four-dimensional latent space.

improvement due to this technique. Although D's test set accuracy is centered around the optimal value of 0.5, we find that it is difficult to reduce the relatively large noise illustrated in Figure 2. In the right panel of Figure 2, we plot the discriminator values and generated distributions for d_{peri} over the course of training. The generated distributions evolve to match the true d_{peri} distribution, but the $G(Z)$ distributions do not match the data as closely for the other halo features.

4.3. *Generative Model Comparison*

We evaluate our GMM and GAN models by comparing their generated halo feature distributions to the `m12i` and `m12f` hydrodynamic data. We compare the means and covariance matrices using the Euclidean norm, and we use the Average Distance (AD) metric to capture information beyond these moment-based estimators. In particular, the AD metric for two probability distributions G_1 and G_2 is defined as

$$AD(G_1, G_2) = \frac{1}{|X||Y|} \sum_{x \sim G_1} \sum_{y \sim G_2} ||x - y||_2, \tag{6}$$

where $|X|$ and $|Y|$ are the number of samples drawn from G_1 and G_2, respectively.

We report the mean, covariance, and AD metrics for our GMM and GAN models in Table 2, and we also show the results for a simple multivariate Gaussian (MG) model fit to the same DMO data as a baseline comparison. The mean and AD metrics for the three models are comparable, but our GAN does not capture the covariance of the hydrodynamic feature distributions. Our GAN seems to produce feature distributions that are too correlated, and we hope to address this issue in future work.

5. DISCUSSION AND FUTURE WORK

We have demonstrated that generative machine learning algorithms have the potential to produce realistic halo catalogs for cosmological simulations of dark matter halos, and our GAN results suggest several interesting points for discussion and future work. For example, even with the help of batch normalization, we find that our GAN behaves somewhat chaotically, in the sense that modest changes to the hyperparameters do not result in predictable changes to the generated halo distributions. It is worth exploring how to overcome this issue, and specifically how to ensure that the complexity and learning rates of G and D remain balanced for arbitrary training simulations in order to avoid mode collapse. GANs also provide valuable latent space information. In particular, while points from the latent space are randomly drawn during training, it is possible to strategically select latent space points to evaluate the trained model. We discern well-separated clusters when evaluating the generator along simple paths in latent space (e.g., see Figure 5), which suggests that it might be possible to partition this space into regions corresponding to different simulations. This would allow us to interpolate between training simulations in order to efficiently produce halo catalogs that map between different systems and initial conditions. We look forward to exploring this possibility in the future.

APPENDIX

A. CONTRIBUTIONS

The authors (EN and DT) collaborated on all of the work described above.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., et al. 2015, TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, software available from tensorflow.org

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., et al. 2014, ArXiv e-prints, arXiv:1406.2661 [stat.ML]

Hunter, J. D. 2007, Computing in Science Engineering, 9, 90

Jones, E., Oliphant, T., Peterson, P., et al. 2001, SciPy: Open source scientific tools for Python, [Online; scipy.org]

Kingma, D. P., & Ba, J. 2014, ArXiv e-prints, arXiv:1412.6980 [cs.LG]

Mao, Y.-Y., Williamson, M., & Wechsler, R. H. 2015, ApJ, 810, 21

Mustafa, M., Bard, D., Bhimji, W., Al-Rfou, R., & Lukić, Z. 2017, ArXiv e-prints, arXiv:1706.02390 [astro-ph.IM]

Nadler, E. O., Mao, Y.-Y., Wechsler, R. H., Garrison-Kimmel, S., & Wetzel, A. 2017, ArXiv e-prints, arXiv:1712.04467

Pedregosa, F., Varoquaux, G., Gramfort, A., et al. 2011, Journal of Machine Learning Research

Pérez, F., & Granger, B. E. 2007, Computing in Science Engineering, 9, 21

Schawinski, K., Zhang, C., Zhang, H., Fowler, L., & Santhanam, G. K. 2017, MNRAS, 467, L110

van der Walt, S., Colbert, S. C., & Varoquaux, G. 2011, Computing in Science Engineering, 13, 22

Wetzel, A. R., Hopkins, P. F., Kim, J.-h., et al. 2016, ApJL, 827, L23

Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. 2017, ArXiv e-prints, arXiv:1710.09412 [cs.LG]