Super-resolution Simulation of the Fuzzy Dark Matter Cosmological Model

Meris Sipp
Research Advisor: Rupert Croft Graduate Student Mentor: Patrick Lachance
McWilliams Center for Cosmology

Introduction

An alternative cosmological model to the cold dark matter (LCDM) model, the fuzzy dark matter (FDM) model proposes that dark matter is extremely light, on the order of 10^{-21}eV. Due to this extremely small mass, dark matter has de Broglie wavelengths that are macroscopic in size at light years in scale rather than the nanometers of other particles. At large-scales, FDM maintains the structure predictions of a cold dark matter cosmological model, but it predicts different small-scale structure including galaxy formation. [1]

As the FDM model is fairly new, little has been done to create highly detailed, super resolution (SR) particle distributions of the model to study its small-scale structure. Limitations in computing power inhibit this greatly as even with the aid of supercomputers a balance must be met between resolution and volume causing sacrifices in both areas. Through use of Machine Learning these limitations can be overcome.

Prior research in our research group [2] has shown that deep neural networks can be used with great success on the LCDM simulations to generate super resolution (SR) particle distributions from low resolution (LR) ones. We use the Deep Learning technique of Generative Adversarial Networks (GANs), These GANs learn from HR particle distributions to generate SR distributions from simulated LR ones of the FDM cosmological model. These SR distributions generated from the GAN of an FDM simulation will allow us to determine how well the network is at producing them and decide if the SR outputs can be used in the observation of small-scale galaxy formation and in determining the viability of this model for dark matter.

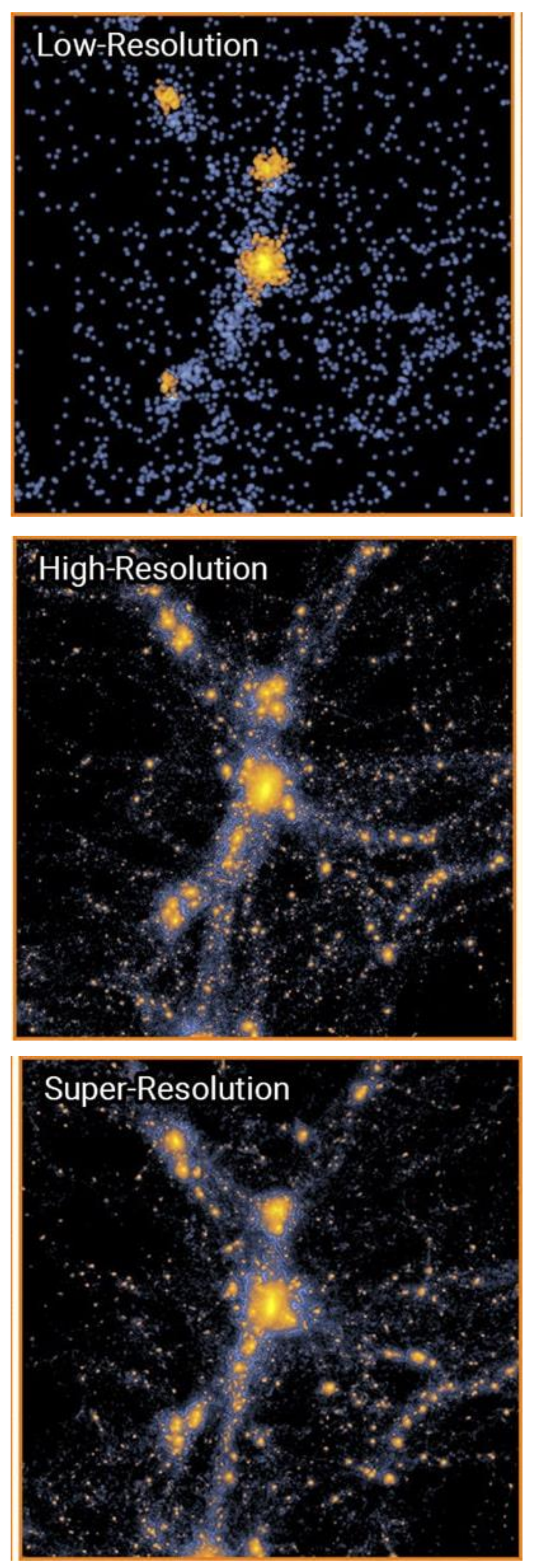

FIG 1 LR, HR, and SR 2D projections of LCDM density fields generated by Li et al. [2]

Model

GANs [2] are a deep learning technique that consists of two separate neural networks, named the generative and the discriminative network. The generative network generates candidate distributions, and the discriminative network compares them with the training set. In doing this, the GAN learns to generate new data with characteristics matching the training set.

In each epoch of training, the generator adds noise during convolution to add irregularity to the different resolutions that was previously absent. This noise becomes high-frequency features as training progresses creating a SR particle distributions.

Methods

We focus on training our GAN to generate FDM SR particle distributions from LR distributions. Additionally, this research was used as an opportunity to test the performance of the new GPU node, TWIG, on the Vera cluster at the Pittsburgh Supercomputing Center.

The LR and HR particle distributions used were simulated on Frontera by way of the FDM-only N-body simulation using MP-Gadget. We currently only have one complete simulation of distributions.

At this time, the GAN has only been trained once as a baseline and the initial analysis of this training are presented here. We hope to train it more times with varying parameters to determine what set of parameters are best for the FDM model.

To analyze the training, we use the Pytorch output of the training epochs to generate SR distributions from LR. We then look at both slices of the dataset and its power spectrum. Slices are analyzed to determine whether if particle clumping is occurring and the power spectrum is a commonly used summary statistic for the matter density field and can be used as a metric to evaluate the SR distributions fidelity, as it is expected to be the same as that of the LR distribution.

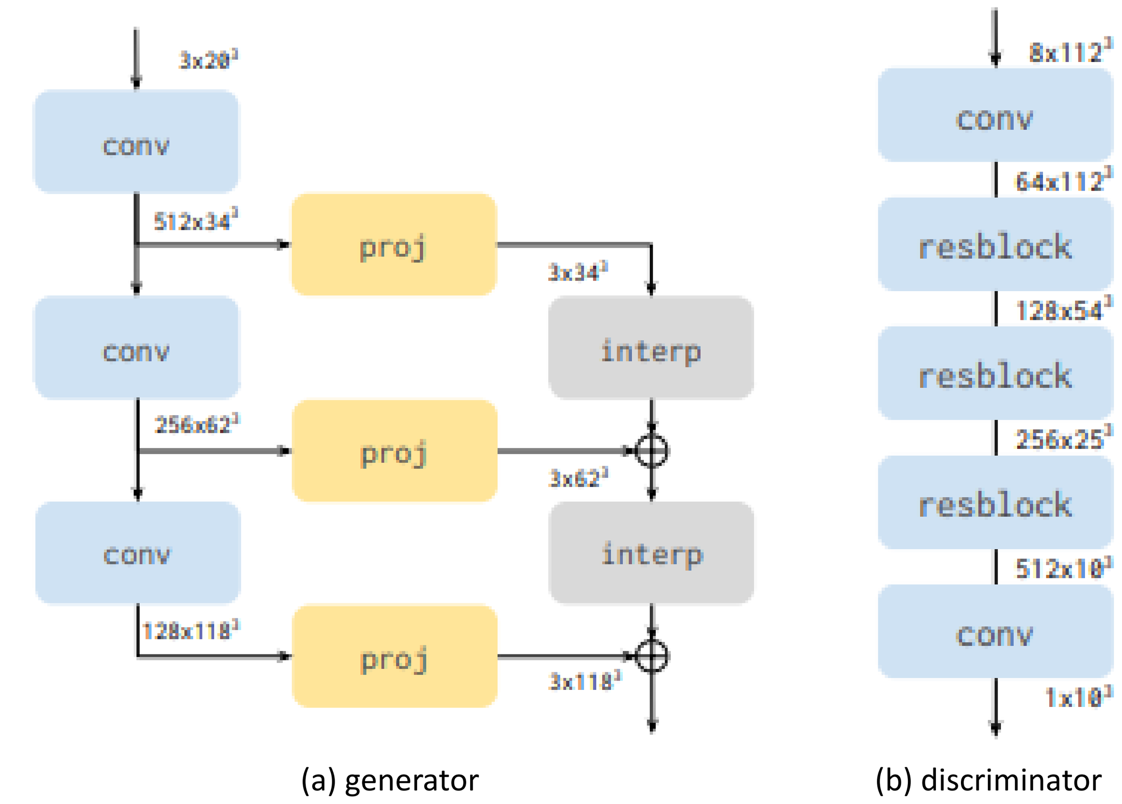

FIG 2 The structure of the two networks involved in a GAN [2]. Colored plates represent operations and arrows connect the operations from input to output.

Results

As of now, only epoch 909 has been process and was the latest epoch of the training due to file storage limits. Additionally, analysis has been done on an example SR distribution to get a baseline for expected distributions and power spectrums.

FIG 3 shows two slices from the example distribution and show the expected clumping of particles. The final slice is from epoch 909 and indicates clumping along the x and y axis, indicative of some issue with either the training or analysis methods.

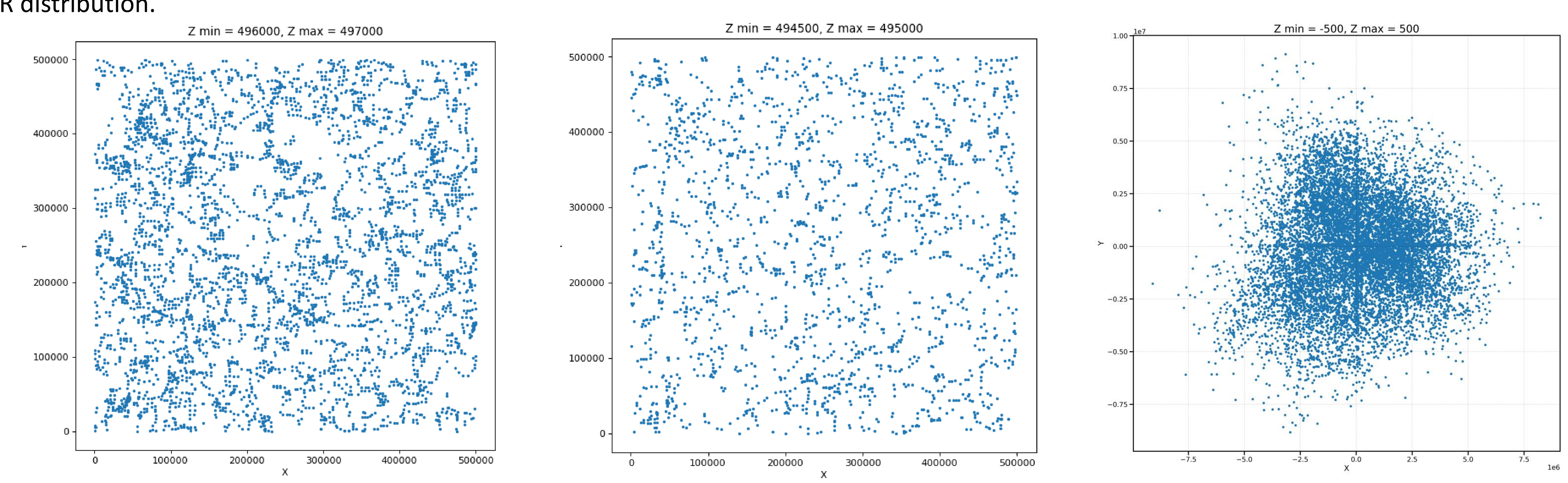

FIG 3 Slices of SR particle distribution simulations. The left and middle come from an example simulation, while the right is from epoch 909 and shows the clumping issue that occurred.

Results cont.

FIG 4 displays the power spectrum from the example SR particle distribution while FIG 5 displays that for epoch 909. The graph is plotted as the amplitude of density fluctuations versus the wavenumber. In FIG 4, there is a turndown on small scales is on the order of the de Broglie wavelength. FIG 5 does not show this behavior further indicating an issue with some step of the training and analysis.

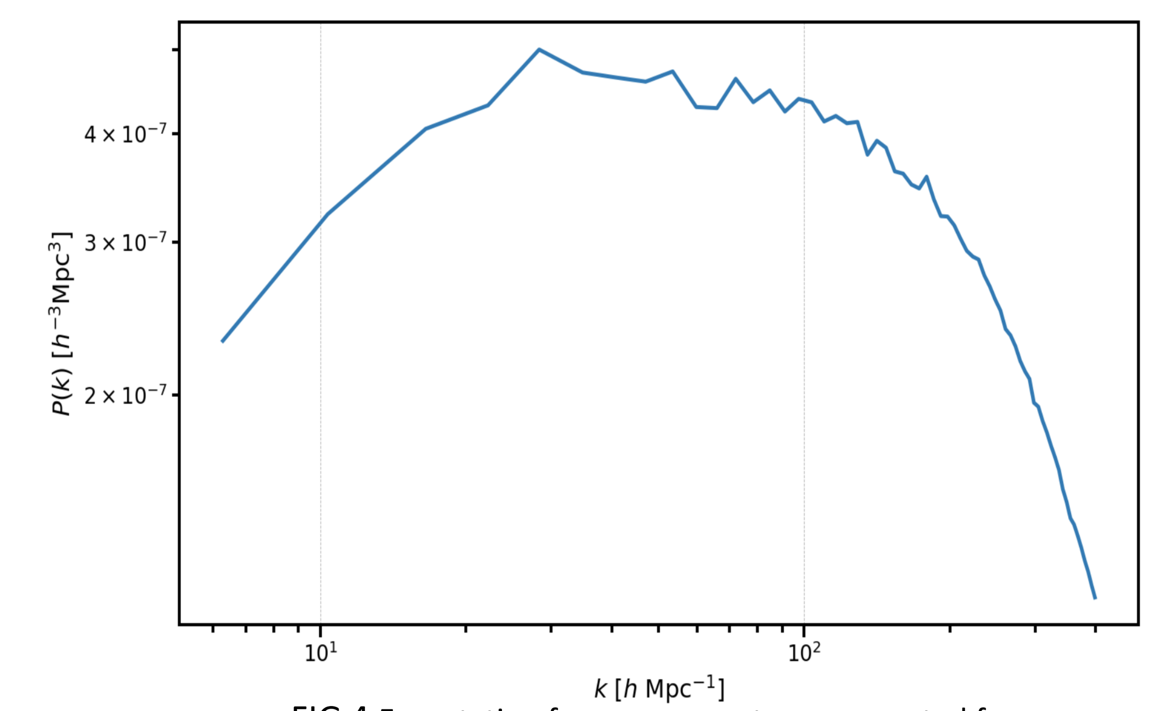

FIG 4 Expectation for power spectrum generated from the example SR distribution

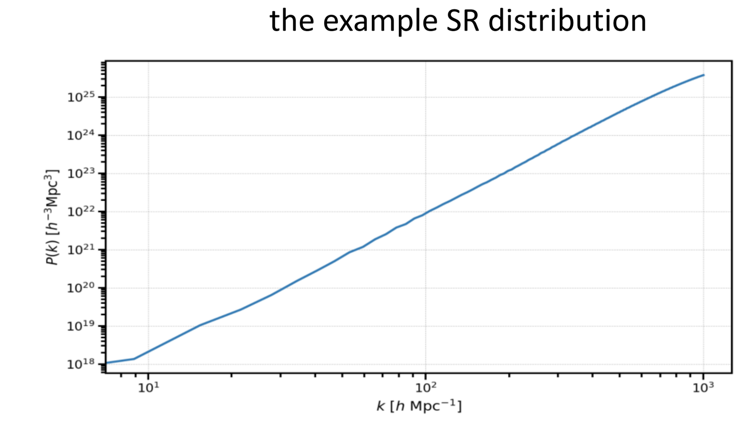

FIG 5 Power Spectrum for Epoch 909

Conclusion

From analysis of epoch 909 of the training, it appears that the neural network cannot process LR particle distributions into SR ones. This is misleading as only one epoch has been analyzed, the last one, so it is possible that by this point overtraining may have occurred leading to more inaccurate SR outputs. This research project is ongoing, and we plan to do more debugging, to analyze more epochs of the original training and to perform more training runs with differing input parameters to find the optimal set of parameters.

Much of this summer was spent determining how to properly set up scripts and submit slurm jobs for TWIG as the node was added to the Vera cluster in early July and has little documentation.

References

[1] Jones, D., Palatnick, S., Chen, R., Beane, A., et al. 2021, ApJ, 913, 1
[2] Li, Y., Ni, Y., Croft, R., Di Matteo, T., et al. 2021, PNAS, 118, 19
Additional thanks to Yueying Ni for help with debugging.