UNIVERSITY OF CALIFORNIA
SANTA CRUZ

COSMOLOGICAL DATA GENERATION WITH GENERATIVE ADVERSARIAL NETWORKS

A thesis submitted in partial satisfaction of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTATIONAL MEDIA

by

Kapil Gupta

June 2021

The Thesis of Kapil Gupta
is approved:

Professor Angus G. Forbes, Chair

Professor Adam Smith

Quentin Williams
Interim Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

Abstract

Cosmological Data Generation with Generative Adversarial Networks

by

Kapil Gupta

Cosmological data is comprised of dark matter and ordinary matter forming halos, filaments, sheets and voids under the influence of gravity over large periods of time, termed as the "Cosmic Web." Direct observations and experiments over the cosmic web are not possible, making computational models of the underlying processes extremely important in studying evolution of the universe. N-body simulations, which are the ubiquitously used model, are computationally expensive, having to evolve the position of millions of particles of matter over cosmic time while considering effects of various physical interactions. Thus, N-body simulations are a major bottleneck in such cosmological experiments. Generative Adversarial Networks (GANs), which have been successfully used to generate synthetic data from sparse inputs, are proposed as a possible alternative to such simulations in this thesis. We perform the training of GANs in three separate experiments, first training over normalized 2D slices, then over 2D slices of high-dynamic range data, and finally training over 3D voxels of data. We use a 1024^3 block of data of size 180 Mpc generated using Bolshoi-Planck simulation as the input to train all the models. All the trained models take much less amount of time to generate synthesized data with high visual quality. We also discuss several metrics that can be used to compare summary statistics of synthetic data with the input.

To Mom and Dad,

for their advice, patience and faith.

Acknowledgments

I would like to thank Dr. Oskar Elek for all his guidance and advice in working on this project. I would also like to thank my advisor, Professor Angus Forbes for his advice and support. I am grateful to have had the opportunity to work with him at the Creative Codings Lab. I am also thankful to Dr. Joseph Burchett for answering all those questions as well as providing useful feedback. I am also thankful for my lab mates Manu Mathew Thomas, Montana Fowler, Jeffrey Weekly and Henry Zhou, I learnt a lot from everyone of you.

I am also really thankful to my parents as well as my friends for their support and encouragement.

Part I

Background

Chapter 1

Introduction

The Cosmic Web is an intricate multiscale interconnected network composed of all galaxies in the universe and massive thread-like formations that connect these galaxies together [29]. The cosmic web is comprised of long filaments of hydrogen and other gases that act as warm-hot intergalactic medium (WHIM) [23]. The largest known structure in the observed universe is the Hercules–Corona Borealis Great Wall, which is about 10 billion light years in length [16]. With no direct observations or experiments possible, researchers mostly depend upon N-body simulations in order to study the properties of the universe. The N-body simulations consider dark matter, ordinary matter and dark energy to model inter-galactic structures over cosmic time [4]. They use millions of particles with varying mass in a large volume of space with specific initial parameters such as gravitational constants, Planck's constant, redshift value among others [44]. With 95% of the mass of the universe believed to consist of unknown particles collectively called dark matter, which has no significant interactions other than gravity, gravitational interactions are necessary and sufficient to understand galaxy formation and evolution. Gravitational N-body simulations are thus an extremely vital tool in gaining a better understanding of the large-scale structure of universe.

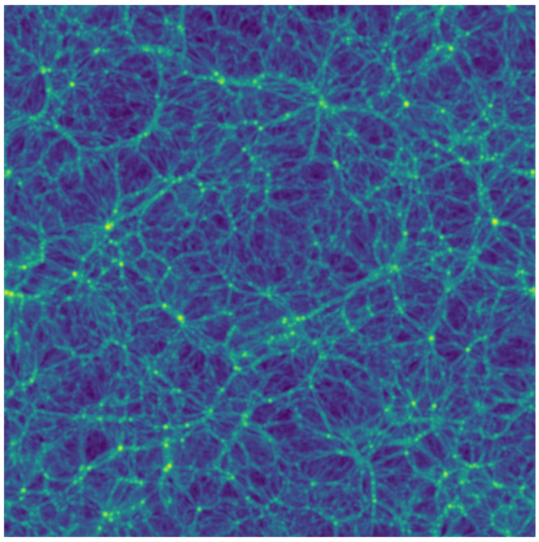


Figure 1.1: A single slice of size 1024x1024 from the simulated dataset, generated using Bolshoi-Planck simulation algorithm. Here, the bright points are the clusters of galaxies termed as halos and the thin strand-like structures connecting halos are the galaxy filaments

EAGLE [40] and Bolshoi-Planck [39] are widely accepted method for generating cosmological simulations, which include using a Monte Carlo Scheme and an equation that guides the statistical behavior of particles in given thermodynamic system (such as Boltzmann transport equation). These methods are computationally expensive, having to follow the N number of particles with specific properties over cosmic time while they are affected by various physical processes within the system. Generative Adversarial Networks (GAN) are one of the alternative methods that can be used to generate synthetic data that shows virtually the same properties as data generated from such simulations, but with a much easier generation process.

Generative Adversarial Networks (GANs) are a type of deep generative model that can be fit to a data distribution [14, 47]. GANs consist of two different neural networks that act as adversaries in a zero-sum game setup, where the resolution is obtained using a well-defined loss function. One of the networks, termed as the Generator, is trained over the dataset such that it can actively generate samples that are supposed to resemble data from the input distribution. The other network, termed as the Discriminator, is trained to be able to accurately judge and classify the input data between the original data and the adversarial sample. Both Generator and Discriminator are trained together in a step-wise manner. The input train set is first used by the Generator to train itself to generate unique samples. Then the discriminator is trained by providing a mix of input set and adversarial samples created by the generator. The loss function for generator is defined according to the ability of the generator to fool the discriminator. Similarly, Discriminator's loss function is defined according to its accuracy in predicting the correct class of input data.

The remaining chapters in Part I (Background) provide a deeper look into previous investigations about the Cosmic Web, N-body simulations and GANs as a method of generating synthetic data. Part II (Experiments) describes the experiments performed in detail, explaining data processing and model design, the process of model training, and providing visual evidence to demonstrate the possible usage of GAN for generating cosmic data. Part III (Discussion and Analysis) describes the metrics used to compare the quality of generated data in terms of summary statistics generally used in cosmology, and discusses methods of analyzing how GANs generate new data after the training process.

Chapter 2

Previous Work

2.1 Generative Adversarial Networks

Generative Models are widely used in various fields such as image enhancement [7, 10, 8], high quality image generation [5, 13, 27], style transfer (image translation) [21, 19, 49], image inpainting [48, 18], adversarial data generation [25, 17] among many others. Such models are able to create novel and representative samples from high dimensional input data distributions. Commonly used generative models include Variational Autoencoders [22], Recurrent Neural Networks [41], Autoregressive Language models such as Generative Pre-trained Transformer (GPT) [36, 43] and Generative Adversarial Networks [14].

2D Data Generation is performed using two-part neural network architecture composed of Generator and Discriminator models.

$$J^{(D)} = -\frac{1}{2}E_{x\ p_{data}} logD(x) - \frac{1}{2}E_{x\ p_{data}} log(1 - D(G(x))) \tag{2.1}$$

In Equation 2.1, J^D is defined as cost function for the discriminator model and in equation 2.2, J^G refers to the cost function for the generator model.

$$J^{(G)} = -J^{(D)} \tag{2.2}$$

2.2 Cosmological Data Generation

Cosmic web shows self-similarity in its large-scale structures [12, 11] and hence, several procedural methods have been used to generate cosmological data such as using physarum polycephalum slime mold model [6], geometrical models using voronoi tesselations [15, 34], delaunay tessellation field estimator (DTFE) [42] among others.

Deep Learning based methods of data generation have also become popular due to the ease of computation [28]. Mustafa et al. [30] used GANs to generate projected matter distribution and were able to achieve high accuracy in terms of non-gaussian summary statistics such as mass histograms and power spectra. Rodriguez et al. [37] used a generative model to create 2D slices for the N-body simulation data. Perraudin et al. [33] considered this problem in terms of upsampling, first generating a low resolution slice, and then upsampling it to the same resolution as input data. Kodi Ramanah et al. [24] worked on using GAN to create super-resolution maps of low resolution data. Inpainting missing data by using nearest neigbour and generative models was considered by Puglisi et al. [35].

Part II

Methods

Chapter 3

Data Processing

The project uses data generated using Bolshoi Planck simulation algorithm. The data is in the form of a 3D numpy array of shape (1024x1024x1024). It was generated using 94795 points as initialization to the algorithm(which were listed along with their mass and density values, and their positions were tracked to create a halos' catalog), with about 10 million agents running in parallel on a supercomputer to generate it. The simulation grid had a resolution of 1200^3 voxels, which was processed and reduced to 1024^3 grid. The data contains 1024^3 particles' intensity values which are directly correlated to the mass and density values as provided in the halos catalog. Given data has high dynamic range, with 64-bit floating point value and intensity value range being [0.017891133176195093, 448.9362993707059].

The intensity distribution in the input data is as shown in Figure 3.2. Although the range of intensity can vary from 0.01 to 448.9, most of the points have intensity in the range of (0.0, 2.0]

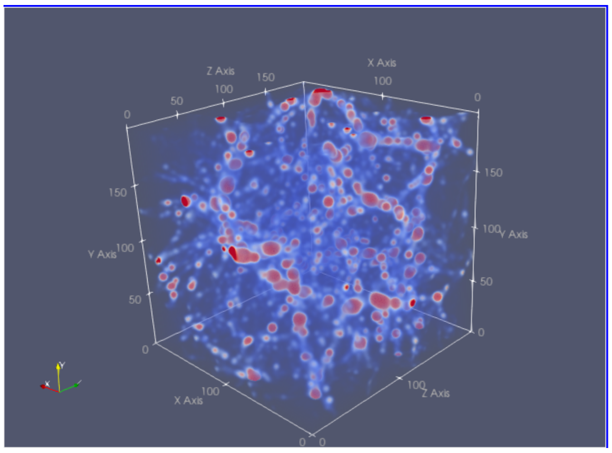


Figure 3.1: 3D cube from the input data generated using Volume Visualization using Paraview

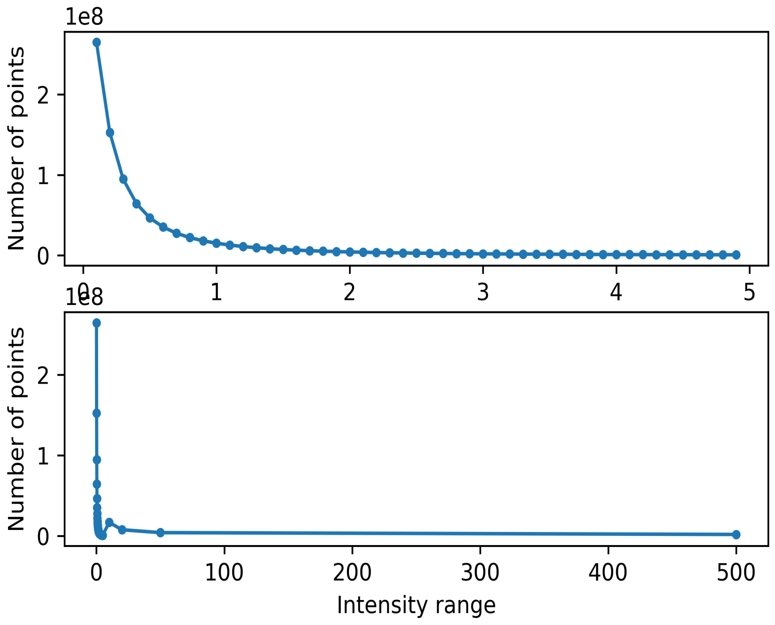


Figure 3.2: Number of points in the input data with intensity value close to given values. Most of the points are in the lower spectrum of intensity, showing that the intensity distribution is approximately logarithmic in nature.

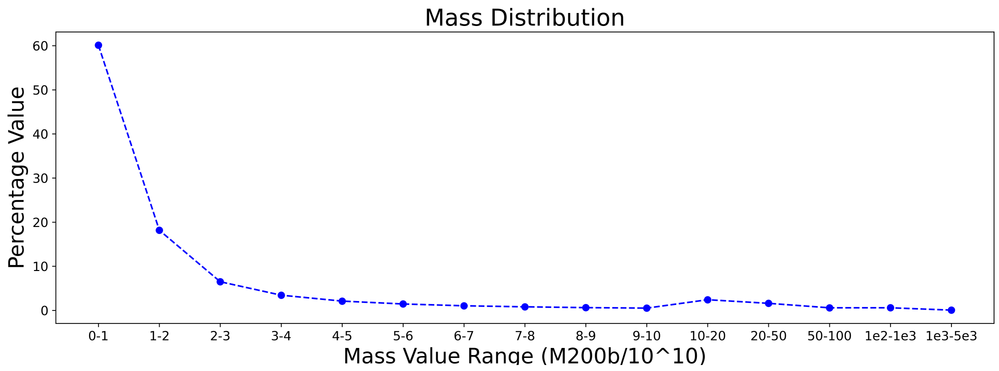


Figure 3.3: Percentage value of points that fall within a mass value range. As shown here, about 80% of the points fall in the [0, 2] range, while less than 10% points have a mass value that falls in range [5e0, 5e3].

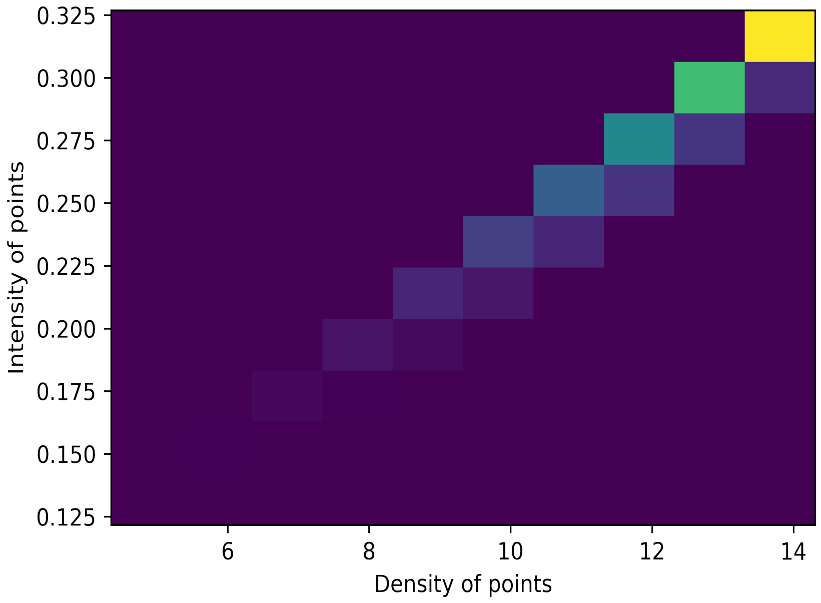


Figure 3.4: 2D Histogram of density distribution of points (from the halo catalog) and intensity distribution (from the 3d data cube). As can be observed from the histogram, there is a linear correlation between density and intensity of points in given data.

As can be observed from the histogram in fig 3.4, there is a direct (linear) correlation between intensity and density values of given data. The histogram was plotted by first sorting the density values (as taken from the halo catalog), sorting the intensity values and then finding the corresponding intensity values of pixels from the 3D input cube. The reason for sorting being that the linear correlation is only observed in top 90% of the data, while the data with mass in range [5e0, 5e3] not showing such correlation. The correlation between density and intensity values in given data is used in this project, as it is assumed that a model that can predict large-scale structures well (and hence, intensity values of pixels) is able to correctly predict the density values too (density values represent the physical interactions of halos).

Chapter 4

Generative Adversarial Networks

Model training is separated into three parts according to how the data was preprocessed. For the first experiment, the data was normalized into the RGB range of [0, 255] in order to avoid the HDR property of data. For the second experiment, the model was trained over 2D slices with no normalization performed. For the third experiment, model was trained over voxels constant dimension less than the original data cube.

4.1 2D Image GAN

The idea for this experiment was to understand how the HDR property affects the GAN and provide a baseline for further experiments. In order to do that, the input data cube was first divided into 2D slices of size (1024x1024) and then the intensity values were normalized into the RGB range [0, 255]. The input for the generator is a halo map, an image that contains a constant value of 100 for pixels where there are no corresponding halos in the data cube slice, and normalized intensity value in range [100, 255] where the halo are present.

The generator model used in this experiment is Resnet-6 model, which consists of 6 resnet layers with alternate downsampling and upsampling layers. Each of the resnet layers employ skip connections. The discriminator model is the Pixel Discriminator, which provides a 0/1 value according to fake/real classification of each pixel.

4.2 2D Tensor GAN

In this method, the 2D slices from the data cube were not normalized before being fed into the model to be trained. The Generator network used here is the Unet-256 model, which is a fully-convolutional model with upsampling and pooling layers. Although Unet-256 was initially proposed for segmentation tasks, it has been successfully used in image generation tasks as well as the usage of upsampling layers in the later layers allows for increased resolution of the output. The Discriminator is the Pixel discriminator, same as the one used in the previous experiment. The reason for using a different generator from the previous experiment is partly empirical, as better visual output was observed from unet-256 model. Another reason for using a different model is that unet-256 requires less computational resources as well as training time in comparison to resnet-6 and resnet-9 models, both of which were tried before moving to unet-256 model.

Although the data was not pre-processed, the precision of data was reduced from its original 64-bit floating point to 16-bit floating point value. There was no apparent difference in the output that could have been caused by reduced precision.

4.3 3D Tensor GAN

For the 3D training, instead of getting slices from the 3D cube, voxels of size (256x256x256) were sampled. A stride of 128 was chosen to allow the network to be able to understand and learn the interconnected nature of data. The Generator model used here is the 3D Unet-256 model, which has the same architecture as the 2D Unet-256 model except with 3D convolutions. The Discriminator used is pixel discriminator.

Data precision was dropped from 64-bit floating point to 16 -bit floating point value for this model in order to fit the model into available GPUs. Since the model was still too large to fit into a single GPU, the discriminator and generator models had to be trained on separate gpus. This lead to longer training time, as the discriminator model had to wait for the generator model to finish training and vice-versa.

Chapter 5

Evaluation Metrics

In order to understand and use any cosmological model, the two most important astronomical observations are the cosmic microwave background and large scale structure of the universe. In this work, the focus is on comparing the large scale structure of generated data with input data from physics based simulations.

5.1 Mass Histogram

Mass Histogram is the average (normalized) histogram of the pixel value in the image. As shown in the chapter on data, the pixel intensity value is correlated to the matter density. Hence, the mass histogram is able to give a good estimate of the matter density distribution and is a good metric of comparison. A matching histogram signifies that the algorithm was able to generate correct intensity values, and hence the correct matter distribution in the output data.

5.2 Peak Histogram

Peak Histogram represents the distribution of maxima in the density distribution. Peaks are defined as pixels with Signal-to-Noise ratio higher than their 8 neighbors using mass / density values from the halo catalog. The Peak S/N distribution is measured and the variation of the peak measurements are plotted in a histogram. Peak Histogram is able to capture the non-Gaussian features present in the data.

5.3 Two-Point Correlation

The Two-Point Correlation function is used to measure the galaxy density distribution in the given data [9] by tracing the amplitude of galaxy clustering as a function of a specific variable (here, density). The correlation function calculates the value of excess probability of pair of galaxies being separated by this distance (excess over and above the probability that would arise if the galaxies were simply scattered independently and with uniform probability) [1].

$\xi(r)$ in equation 5.1 refers to the correlation function, which measures the excess probability dP of finding a galaxy in a volume element dV above the expected poisson distribution at a distance r from another galaxy, where n is the mean number density of the galaxy sample in question.

$$dP = n[1 + \xi(r)]dV \tag{5.1}$$

In order to calculate the two-point correlation function $\xi(r)$, first a catalog with same dimensions with randomly distributed points is constructed. The ratio of pairs of galaxies observed in the data relative to pairs of points in the random catalog is then used to estimate $\xi(r)$. The most commonly used estimator is by Landy and Szalay [26] represented by equation 5.2 where DD and DR are counts of pairs of galaxies (in bins of separation) in the data catalog and between the data and random catalogs, n_D and n_R are the mean number densities of galaxies in the data and random catalogs and RR is the count of pairs of galaxies as a function of separation in the random catalog.

$$\xi = \frac{1}{RR}[DD * (\frac{n_R}{n_D})^2 - 2 * DR * (\frac{n_R}{n_D}) + RR] \tag{5.2}$$

5.4 Power Spectrum

Power Spectrum is the Fourier transform of Two-Point Correlation function. It defines the density contrast in the data, that is, the difference between local density and mean density.

Part III

Results and Discussion

Chapter 6

Results

6.1 2D Image GAN

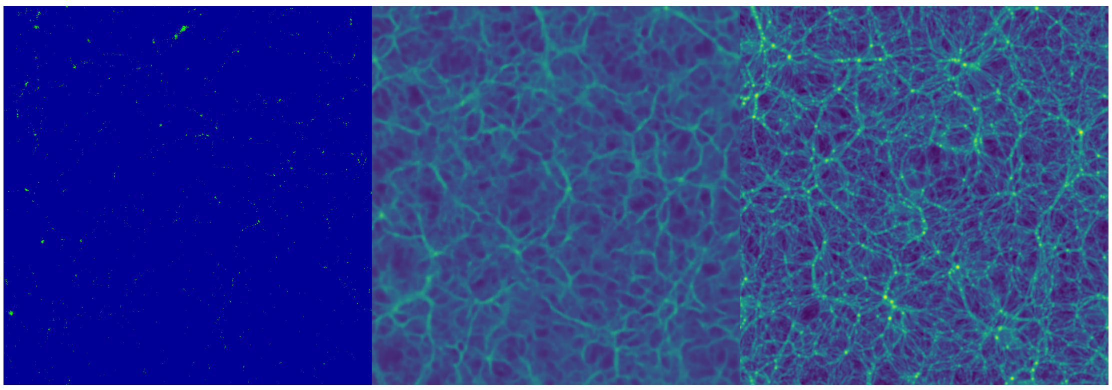


Figure 6.1: Output from the Image based model at the 69th epoch from the validation set. Here, the leftmost image is the input halo map, the rightmost is the original image and the image in the middle is the output. Although structurally similar, the output has a different intensity values from the real image, because the GAN is not able to reproduce the range of intensity distribution of the input data.

In this experiment, the model was able to create a visually high quality sample with similar large filament structures seen in the output as present in the real slice, as shown in Fig 6.1. The model was not able to match the local, small scale structures of the input data, which might be because of normalizing the input data into [0,255] range. The major inconsistency shown by the outputs tends to be in the areas where the filaments are present without any halos at the beginning or end of given filament. Such results are incorrect as filaments cannot exist if not connected by halos. The reasoning for such results can be that since the filaments are 3D structures, a 2D slice is incapable of containing all the information about them. The experiment was repeated with the slices having a small depth value (by concatenating and then averaging multiple slices), but that didn't remove this problem.

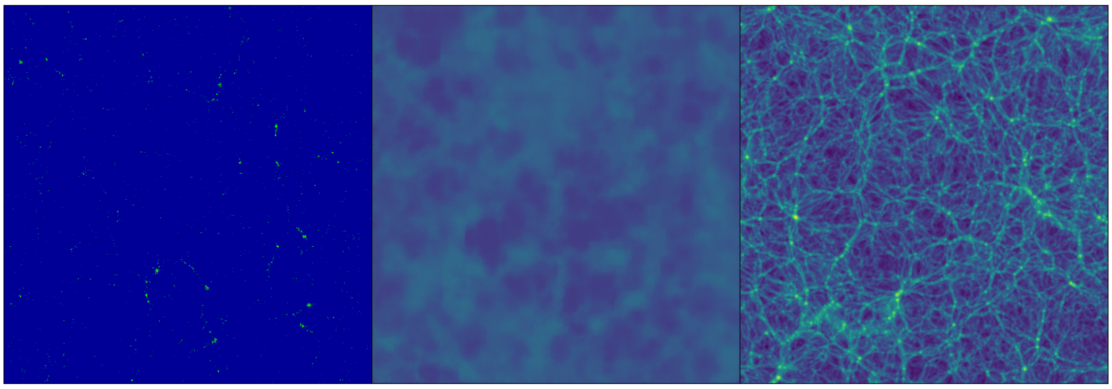


Figure 6.2: Output from the Image based model at the tenth epoch from the validation set. Here, the leftmost image is the input halo map, the rightmost is the original image and the image in the middle is the output. This image is present as a reference to the intermediate steps in training and how the ouput looked like at those epochs.

Fig 6.2 shows the output of the same model at the fourth epoch. The output shows that the model is able to recreate the most evident structures at the fourth epoch in training.

6.1.1 Loss Functions

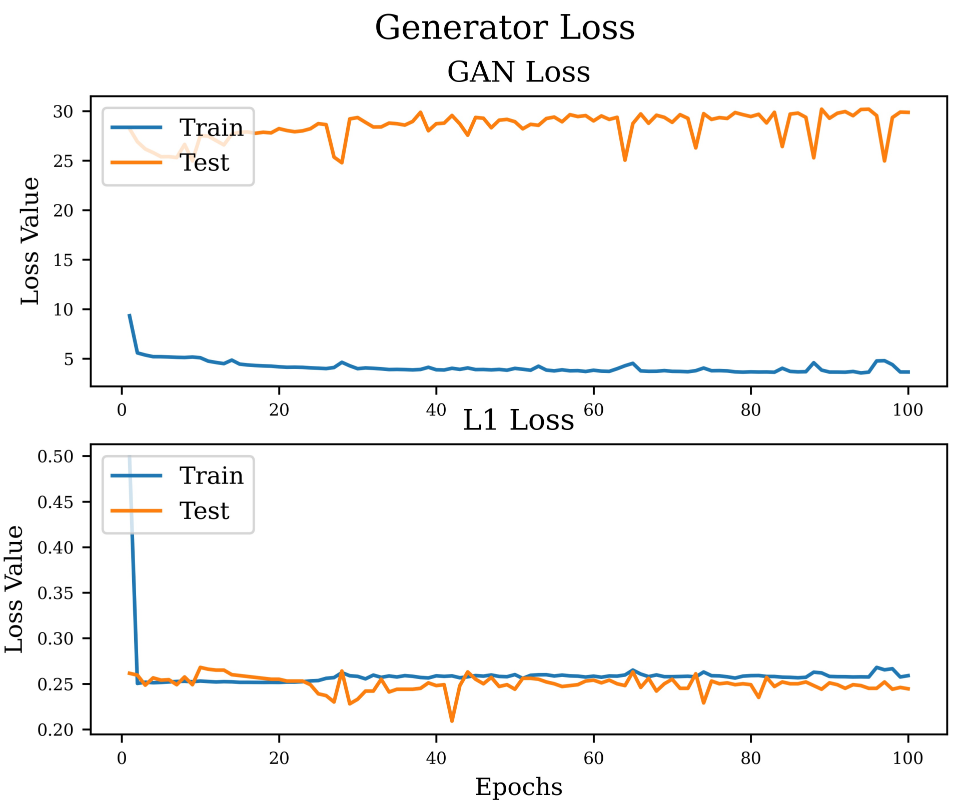


Figure 6.3: Evolution of generator's GAN and L1 loss functions from Train(blue) and Test(Orange) sets as a function of Epochs.

The training in Fig 6.3 shows a stable loss for the generator model at the train set, with a slighly varying loss shown in the test set. Both sets show the same behavior over a few epochs, thus showing that the training is stable. Very similar behaviour can be observed in the discriminator loss in Fig 6.4.

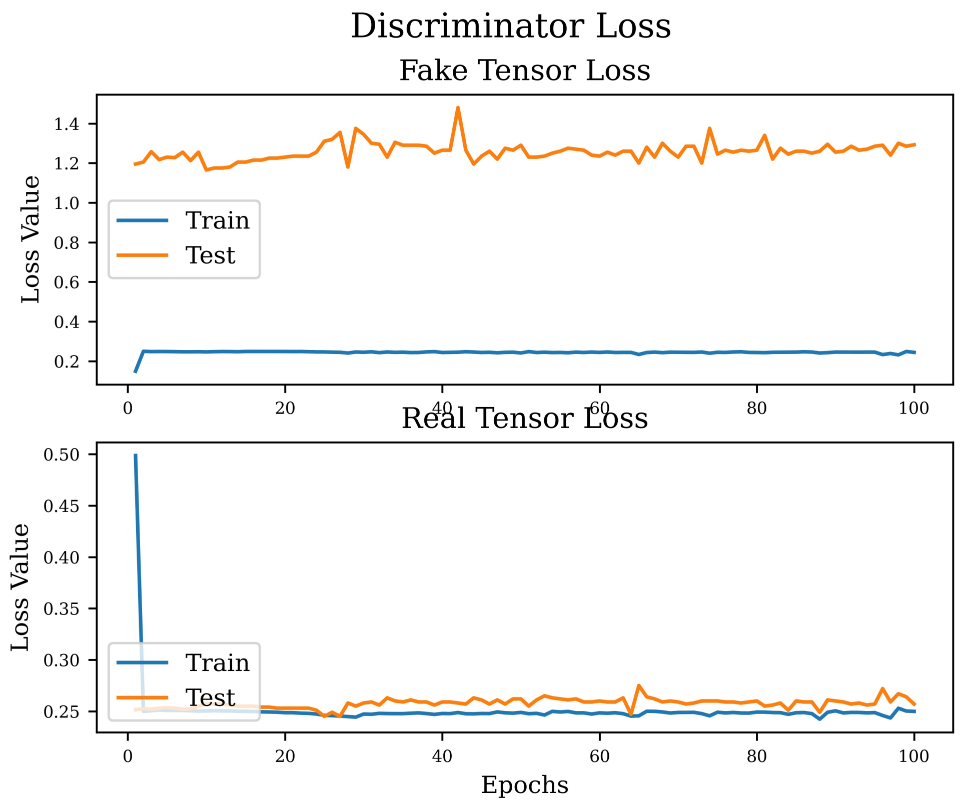


Figure 6.4: Evolution of loss value from the discriminator while classifying over the real and synthesized input from train (blue) and test (orange) sets as a function of epochs.

6.1.2 Comparison Metrics

6.1.2.1 Mass Histogram

The Mass histogram in Fig 6.5 shows that the majority(75%) of the particles have similar pixel values and hence intensity values, while the particles that have different intensity values vary by about 30% at average. This shows that although majority of particles in synthetic data have a matching distribution as the real data, some particles which do not match vary by a statistically significant value.

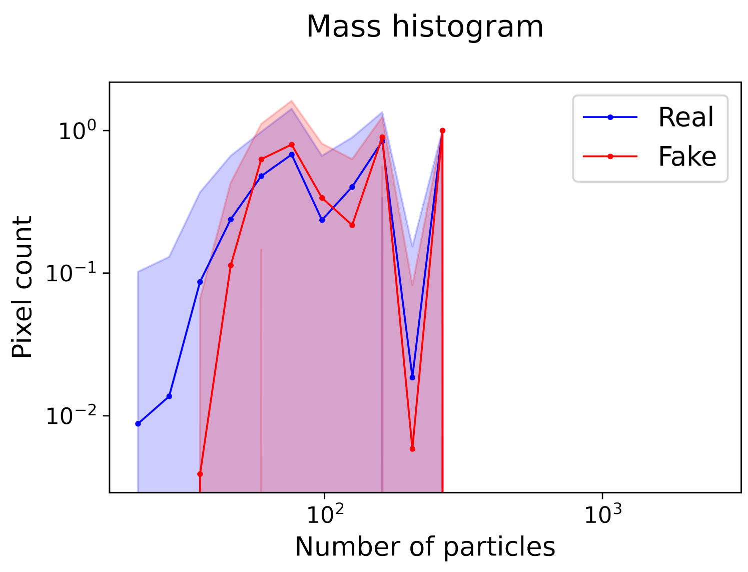


Figure 6.5: Mass histogram

6.1.2.2 Peak Histogram

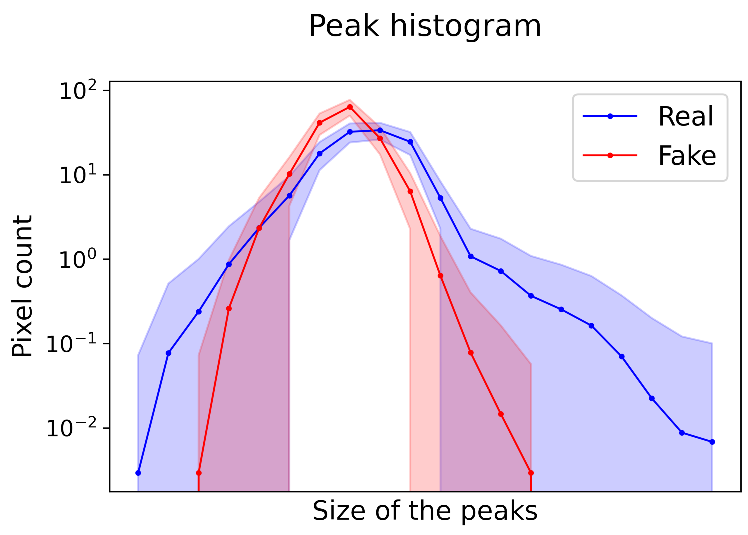


Figure 6.6: Peak histogram of real and synthesized samples from the 2D image model.

The Peak histogram in Fig 6.6 shows that majority of peak intensity pixels from the original data do not match the synthetic data. This finding is also supported by the resulting output image, where the high intensity filament structures do not show the same intensity distribution in synthetic data. This result is correlated to the colors in fig 6.1 and not the large scale structures.

6.1.2.3 Power Spectra

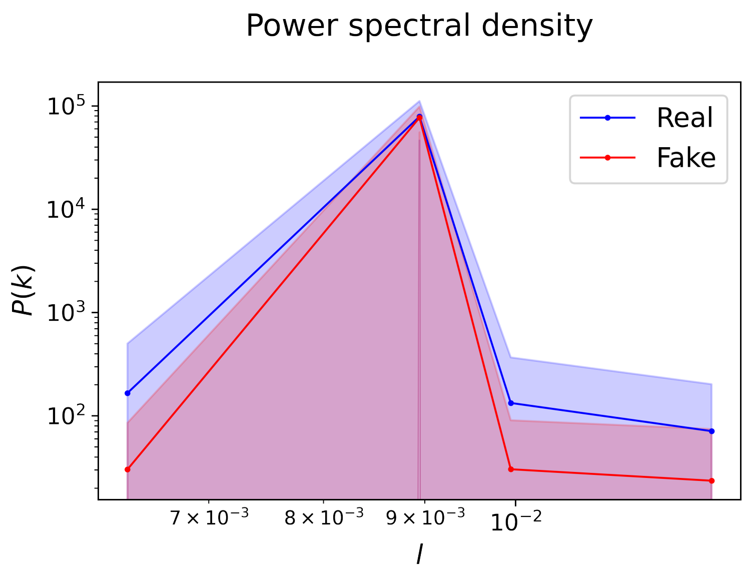


Figure 6.7: Power spectrum of real and synthesized samples from the 2D image model.

As can be observed in the fig 6.7, the peak power spectral density of synthesized data matches the real sample while for the rest of the points, there is an increasing difference between the two samples' values.

6.2 2D Tensor GAN

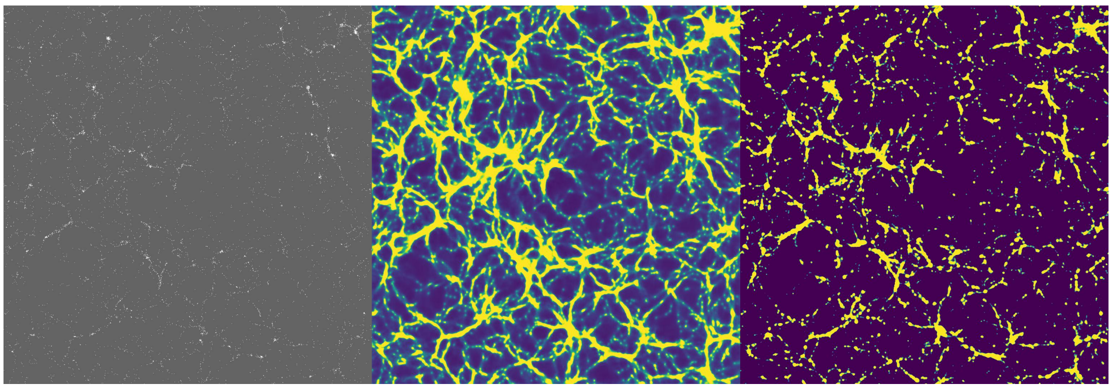


Figure 6.8: Output from the Tensor based model at the 39th epoch from the validation set. Here, the leftmost image is the input halo map, the rightmost is the original image and the image in the middle is the output from the generator. The model is able to generate correct shapes of the structures, but the intensity distribution is different from the original image which can be observed in the different color values of filaments in the two images.

The 2D HDR GAN model was able to understand the major filament structures present in the data and accurately recreate such structures when working with variable number of halos. The GAN model does have a clamping effect on the High Dynamic Range property of the data, cutting off the extremely high and extremely low values in the data to such effect that the output is closer to the normalized input rather than the true values. Moreover, the intensity values were not consistently generated by the model. When calculating the difference image to observe the difference between real and synthesized 2D tensors as shown in fig 6.9, we observed that no multiples of the output lead to a subtracted value close to 0. This shows that the synthesized sample's intensity values are not matched to the input, but since the difference image doesn't have a gradient, it shows that the offset is a consistent, linear value.

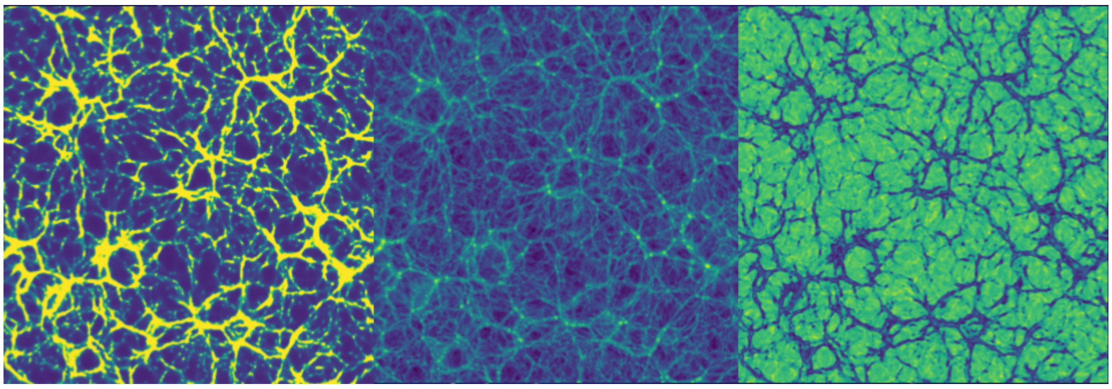


Figure 6.9: Leftmost image is the synthesized sample from the 2D HDR GAN, the middle image is the real image and the rightmost image is the difference between the first two images plotted using the same colormap. As can be seen here, all filament structures have a consistent offset in intensity values.

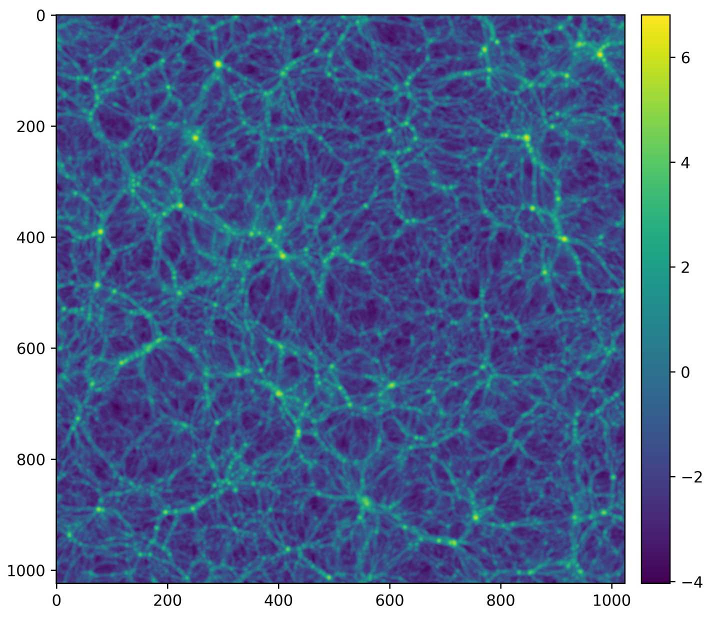
When concatenating the output generated using different axes-aligned halo maps (XY vs YZ vs XZ maps), the outputs were seen to be consistent and having maintained the filament structures. Interestingly, even though 2D slices with no depth were considered as data for training, some 3D structure was seen on concatenating contiguous slice outputs and generating a 3D output. The data's inherent 3D structure and the self-symmetric fractal like property might be the reason for such output. Running model on non-contiguous halo maps might indicate if the model is biased towards trying to generate 2D vectors that form some structure or not.

6.2.1 Loss Functions

Although both the generator and discriminator remain stable upto the 39^th epoch, mode collapse was observed after 40^{th} epoch.

As shown in fig 6.11, the discriminator loss for synthesized tensor is stable for the train set and shows the same average pattern over multiple epochs for the test set. The same can be said for the generator model's L1 and GAN loss in fig 6.12. At 40th epoch, discriminator's loss for both real and synthesized data shoots down while generator's loss shoots up, indicating a mode collapse at these epochs.

In order to avoid mode collapse, we tried to use an autoencoder to create a low dimensional encoding that could then be passed to the generator. The results were very similar to the ones without autoencoder and no apparent benefit to using the encoding was observed.



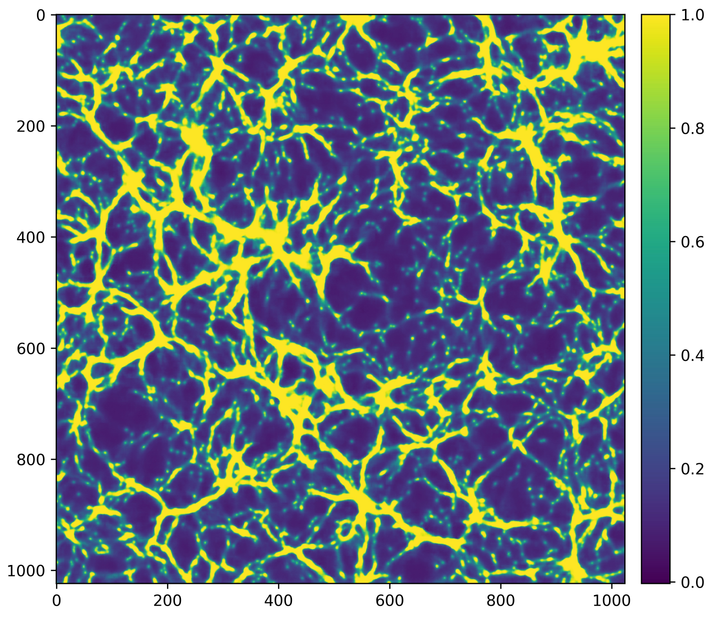


Figure 6.10: Real and Generated samples. As shown here, although the structure is well reconstructed, the generated sample's intensity distribution is not the same as the real input slice.

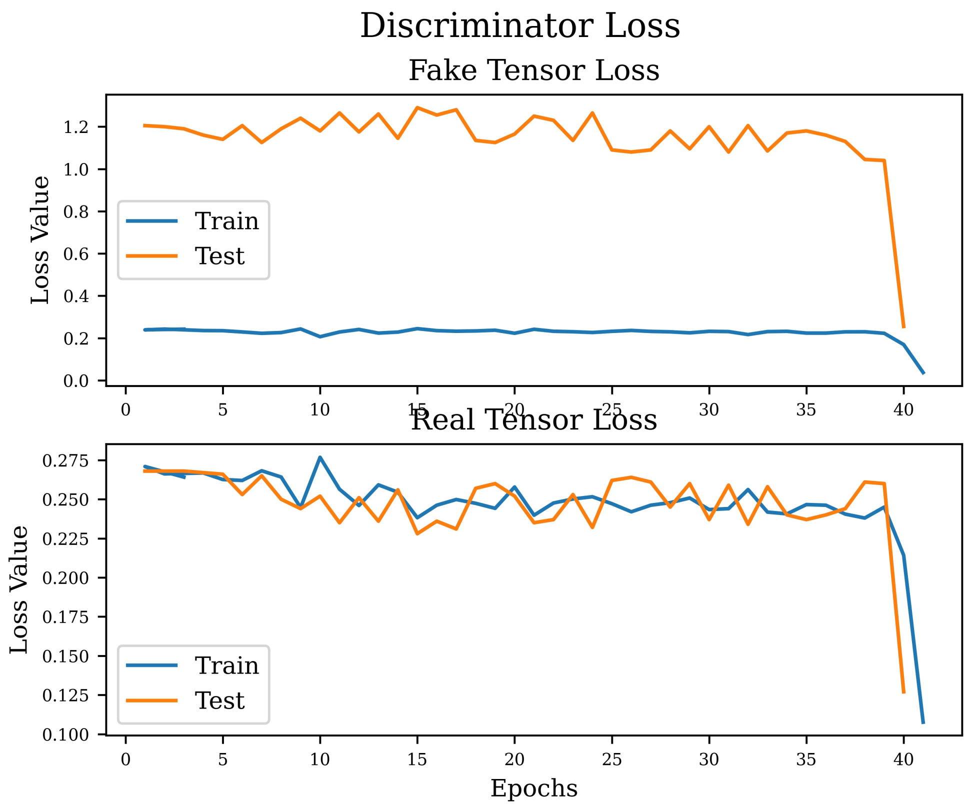


Figure 6.11: Evolution of loss value from the discriminator while classifying over the real and synthesized sample from train (blue) and test (orange) sets as a function of epochs.

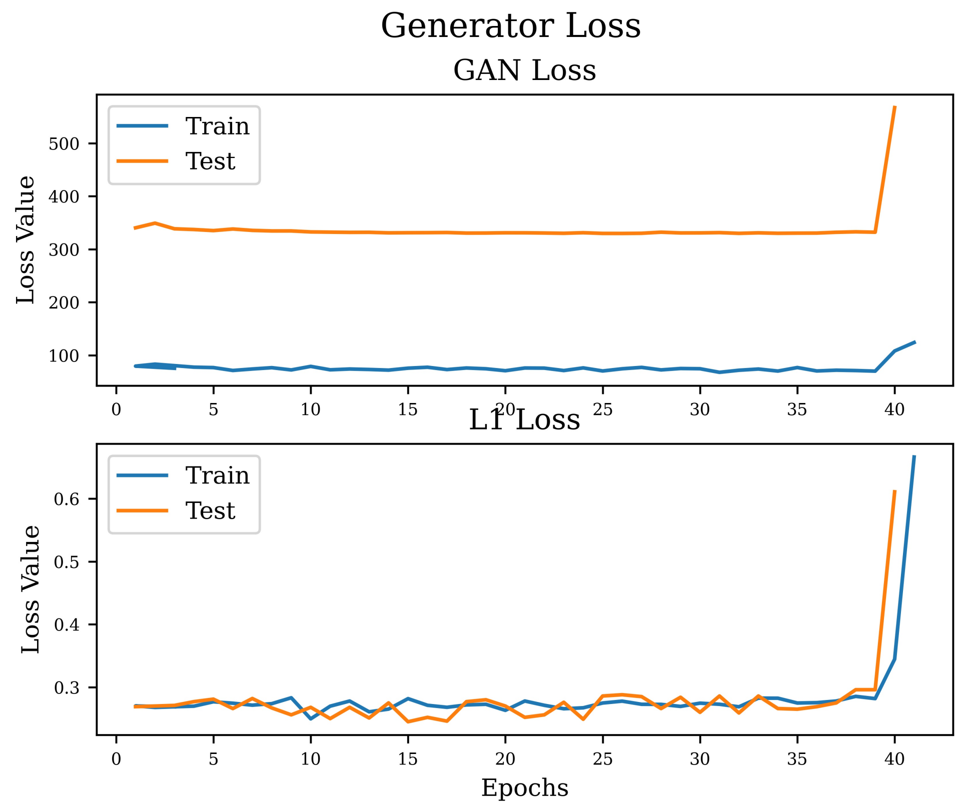


Figure 6.12: Evolution of generator's GAN and L1 loss functions from Train(blue) and Test(Orange) sets as a function of Epochs.

6.2.2 Comparison Metrics

6.2.2.1 Mass Histogram

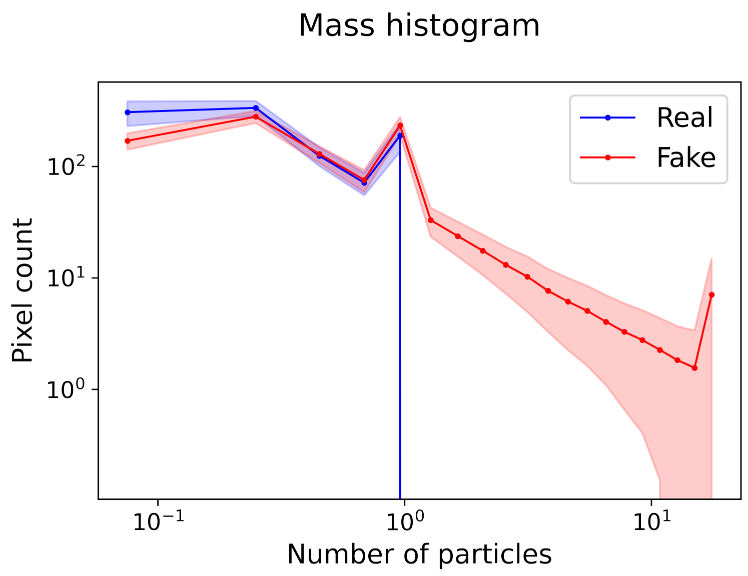


Figure 6.13: Mass histogram of real and synthesized samples' pixel counts with same mass values mapped over the number of particles.

As shown in fig 6.13, mass histogram of real and synthesized samples matches upto a specific number of particles. The rest of the particles show a stable piecewise linear plot in synthesized sample, while non-existent values in the real sample. This is observed because the synthesized sample's filament structures have an offset in their intensity values, and since mass histogram compares the two values by first creating a threshold, the real sample's particles are not able to cross the high threshold. For the non filament particles, the histogram shows that they are closely matched.

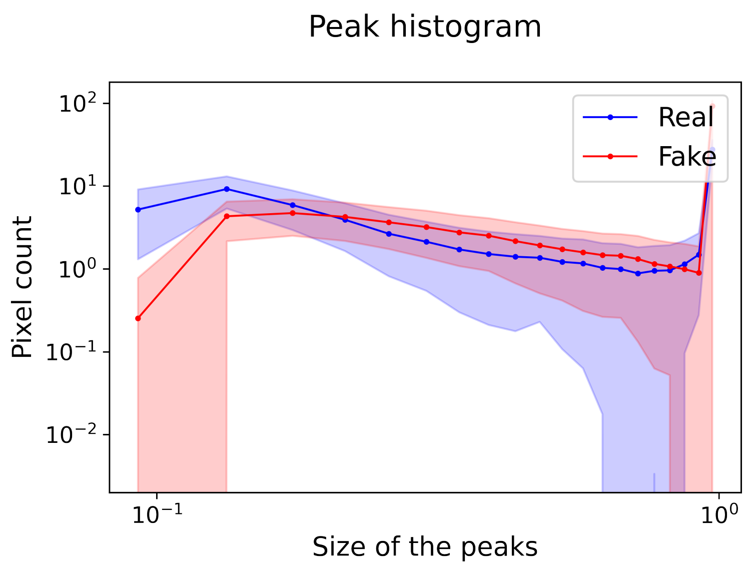


Figure 6.14: Peak histogram of the real and synthesized samples from 2D Tensor model.

6.2.2.2 Peak Histogram

Peak histogram of real and synthesized samples in fig 6.14 shows that the intensity values although not exactly matched, tend to be at a very small offset from each other. This shows that for the peak intensity values, the synthesized sample is able to recreate the intensity (and hence density) distribution of the input.

6.2.2.3 Power Spectra

As shown in fig 6.15, shape of power spectral density (psd) of synthesized sample matches the shape of psd of real sample.

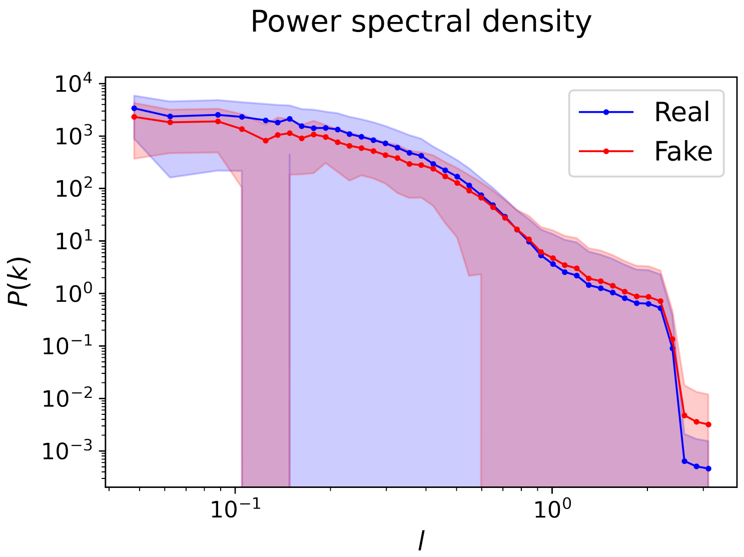


Figure 6.15: Power spectra of real and synthesized samples from 2D tensor model.

6.3 3D Tensor GAN

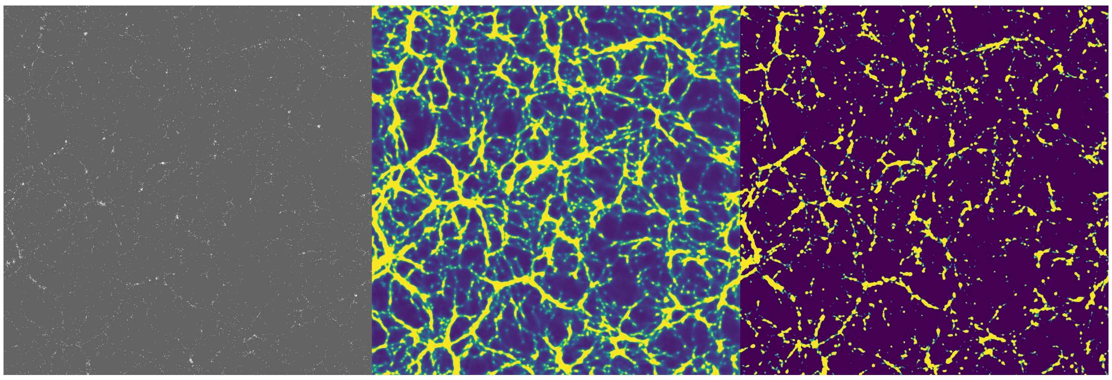


Figure 6.16: Output from the 3D tensor based model at the 30^th epoch from the validation set. Here, the image at the top is the input slice and the image at the bottom is the synthesized output. The colorbar at the right of both images represents the correlation of gradient to the intensity values of pixels. The intensity values have been normalized into [0,1] range. As can be observed from the figure, the synthetic sample has a higher intensity distribution compared to the original image.

6.3.1 Loss Functions

Although the loss function for both discriminator and generator was more unstable than the 2D models, no mode collapse was observed. The plot for discriminator in fig 6.17 for both real and synthesized samples shows a downward trend, while the generator's loss in fig 6.18 shows an upward trend. This shows that discriminator had an easier time of figuring out real from synthesized samples over multiple epochs. Although not the ideal loss function, the output synthesized were in high visual quality, so it can be assumed that the initial low values of loss must be because of the initial values of synthetic sample being close the mean of the intensity distribution in real sample.

Similarly, the generator's GAN loss as shown in fig 6.18 shows an upward trend, showing that the generator had a harder time creating similar outputs to real sample over the epochs.

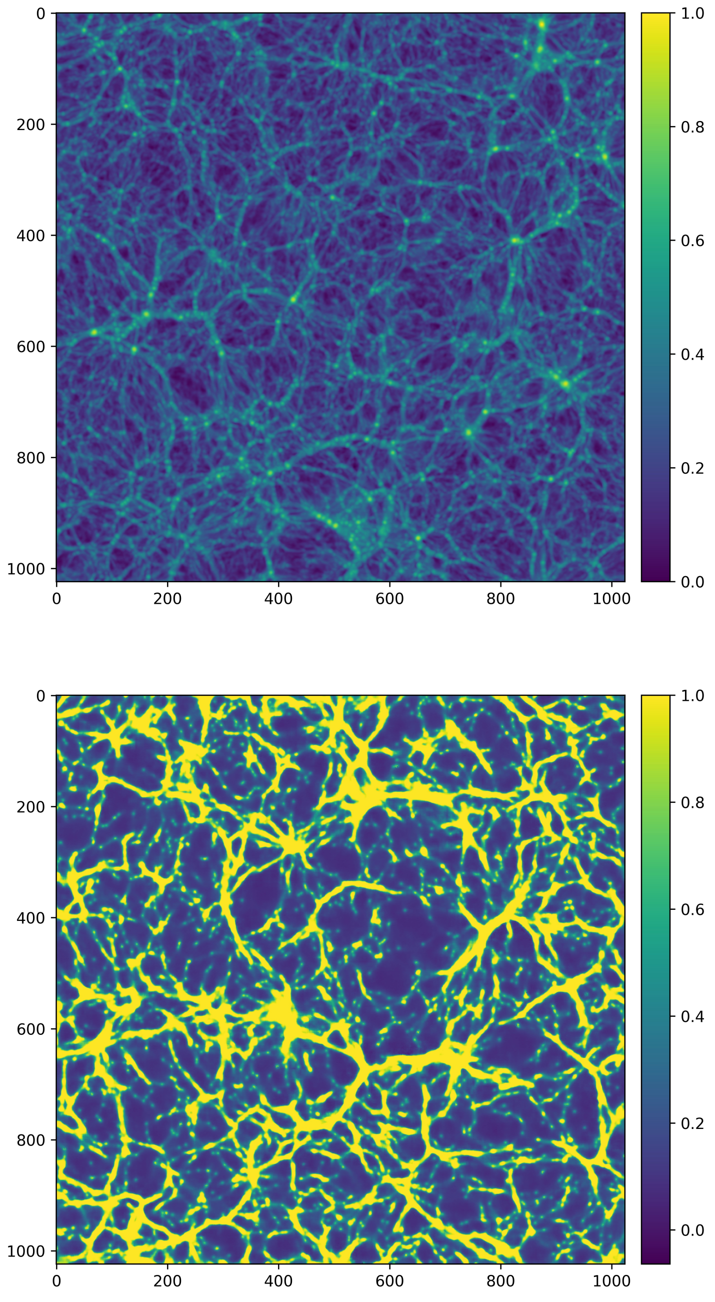


Figure 6.17: Output from the 3D tensor based model at the 30^th epoch from the validation set. Here, the image at the top is the input slice and the image at the bottom is the synthesized output. The colorbar at the right of both images represents the correlation of gradient to the intensity values of pixels. The intensity values have been normalized into [0,1] range. As can be observed from the figure, the synthetic sample has a higher intensity distribution compared to the original image.

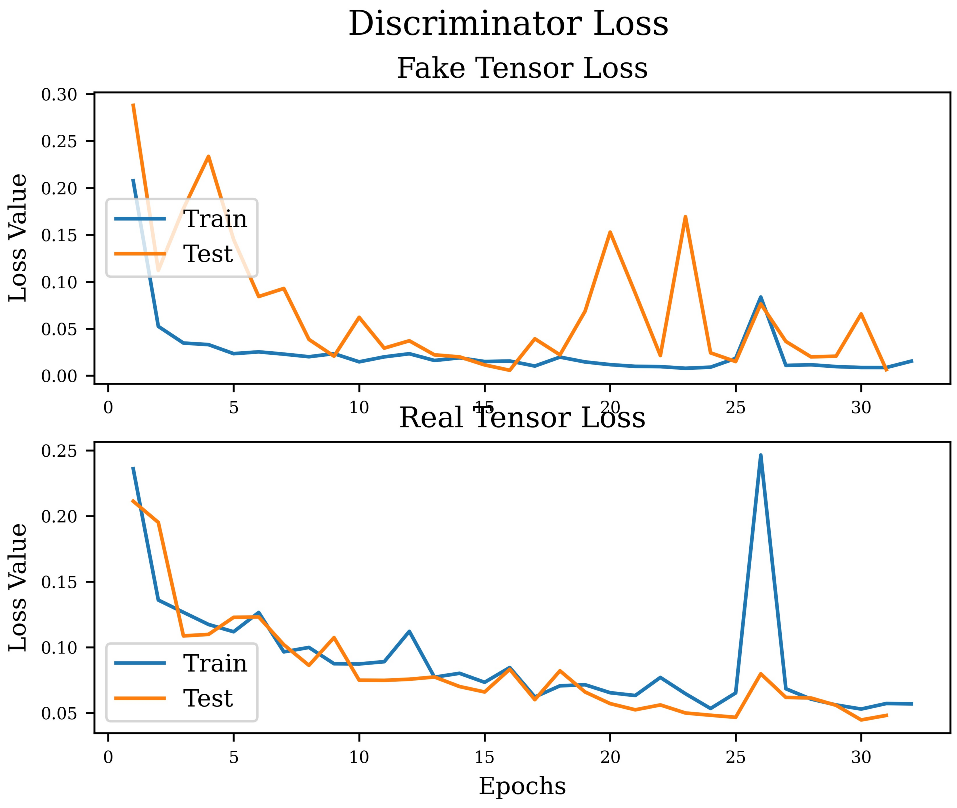


Figure 6.18: Evolution of loss value from the discriminator while classifying over the real and synthesized sample from train (blue) and test (orange) sets as a function of epochs.

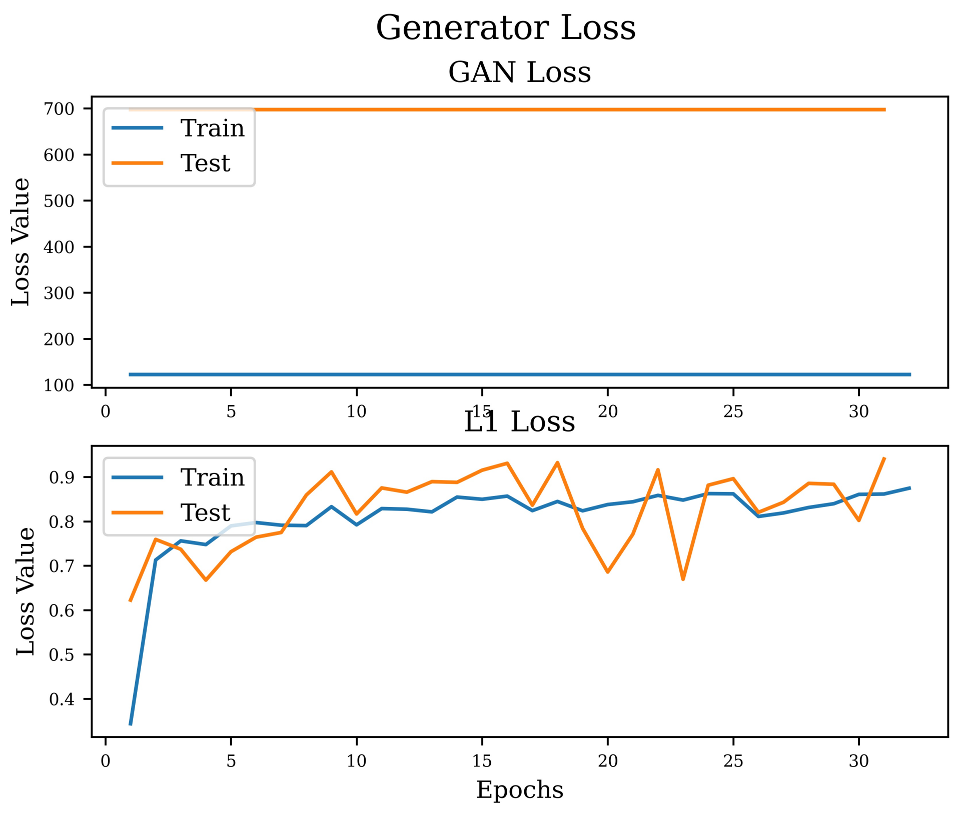


Figure 6.19: Evolution of generator's GAN and L1 loss functions from Train(blue) and Test(Orange) sets as a function of Epochs.

6.3.2 Comparison Metrics

6.3.2.1 Mass Histogram

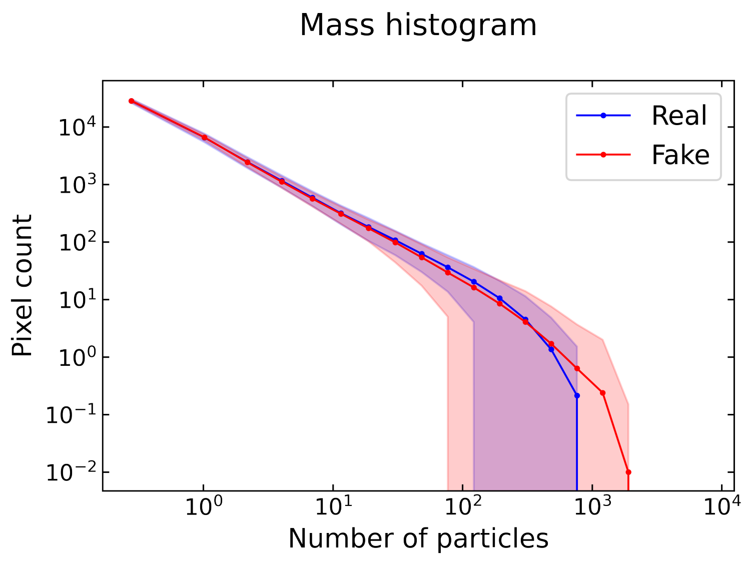


Figure 6.20: Mass Histogram of real and synthetic samples from 3D tensor model.

6.3.2.2 Peak Histogram

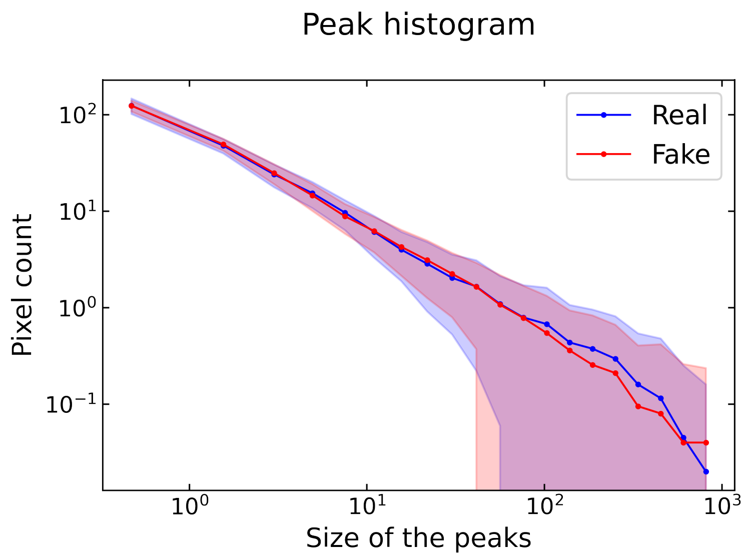


Figure 6.21: Peak histogram of real and synthesized samples from the 3D tensor model.

6.3.2.3 Power Spectra

As shown in fig 6.22, shape of power spectral density (psd) of synthesized sample matches the shape of psd of real sample with a much lower difference than as seen in psd plot of the 2D image model and measurably lower value from 2D tensor model.

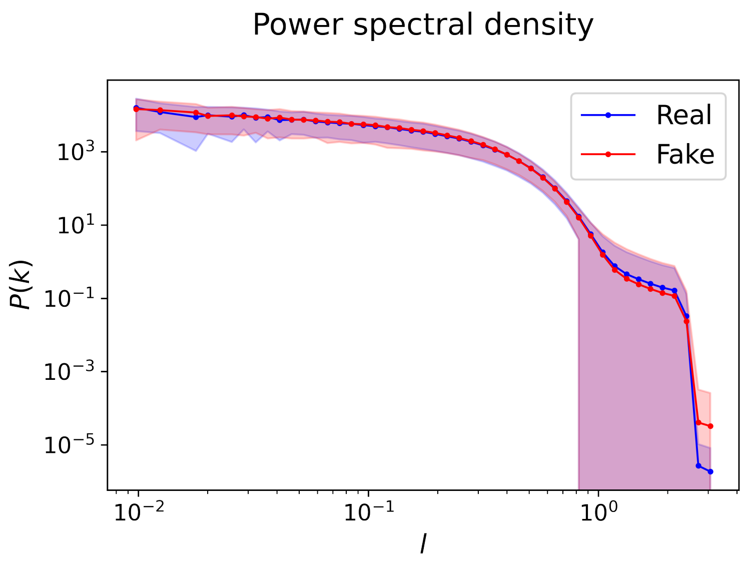


Figure 6.22: Power Spectra of real and synthesized samples from the 3D tensor model.

Chapter 7

Analysis

7.1 Dissecting GANs on same input

In this experiment, we used the same input sample over all the three models, and using NetDissect [3] observed the activations in all three models at the relevant (hidden) layers. NetDissect is a program by CSAIL Vision Lab which can be used to interpret visual representations of the hidden layers of trained models.

As can be observed in fig 7.1 and fig 7.2, there are some overlaps in the activation maps of 2d image and 2d tensor models. The 2D tensor model shows a much more complex activation map, and considers much more area when synthesizing the output in comparison of the 2D image model. This observation is consistent with better match in both large-scale structures and intensity distribution synthesized by the 2D tensor model.

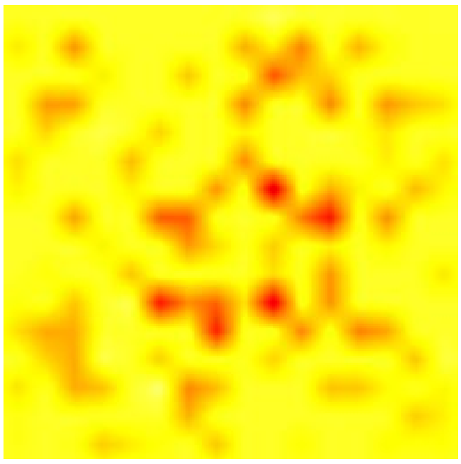


Figure 7.1: Activation Map for 2D Image model

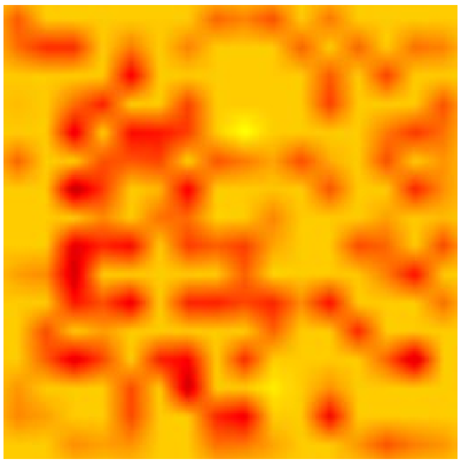


Figure 7.2: Activation Map for 2D Tensor model

7.2 Variations in Halo Count

In this experiment, we changed the number of halos in the input halo map. The idea was to observe how many minimum halos are needed for the gan model to be able to generate a high quality synthesized output. This experiment is also useful in determining how well the model understands the structure formations and what the model looks for when generating the filament structures.

The results of this experiment showed that the models were able to synthesize large-scale structures in the output when the halo count was varied by 20% (increase or decrease) from the initial value at which the model was trained (5000 halos per slice). Beyond that, there were major changes in the structure formed, with lowering the halo count leading to loss of the structures, while increasing the halo count leading to more complex structures.

Chapter 8

Conclusion

This thesis aimed to find a method for generating data that can represent large scale structures of universe, colloquially termed the cosmic web with lower computational requirements than n-body simulations (which are the most widely used algorithms to synthesize such data). Based on visual and quantitative analysis, we have shown that generative adversarial networks are a valid approach to generate such data. GANs are able to generate the cosmological data with evident large-scale structures, while having low variation from the original intensity distribution of the real (simulated) data. Thus, GANs can be used to generate both 2D slices as well as 3D voxels of cosmological data in a fraction of the time taken by computational algorithms with comparable quality.

There are quite a few obvious extensions to this project. Currently the model can generate new data with same parameters as the data it has been trained on. Given models could be trained such that along with the halo map, they take in as input the initial parameters which would guide the model while synthesizing the data. It is also important to have a model with more explainability so that it is easier to use as well as to change according to the requirements. So, another possible extension could be to create models with inbuilt properties such as mapping the gradients to output data in order to map changes in the output to the gradient values, or adding attributes in the model.

Bibliography

[1] Correlation function — Wikipedia, the free encyclopedia, Jun 2020.

[2] Peter Anninos. Computational Cosmology: from the Early Universe to the Large Scale Structure. *Living Reviews in Relativity*, 1(1):9, December 1998.

[3] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Computer Vision and Pattern Recognition*, 2017.

[4] Edmund Bertschinger and James M. Gelb. Cosmological n-body simulations. *Computers in Physics*, 5(2):164–179, 1991.

[5] Siddharth Bhatia, Arjit Jain, and Bryan Hooi. Exgan: Adversarial generation of extreme samples, Mar 2021.

[6] Joseph N. Burchett, Oskar Elek, Nicolas Tejos, J. Xavier Prochaska, Todd M. Tripp, Rongmon Bordoloi, and Angus G. Forbes. Revealing the Dark Threads of the Cosmic Web. *The Astrophysical Journal*, 891(2):L35, March 2020.

[7] Yu-Sheng Chen, Yu-Ching Wang, Man-Hsin Kao, and Yung-Yu Chuang. Deep

photo enhancer: Unpaired learning for image enhancement from photographs with gans. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR 2018)*, pages 6306–6314, June 2018.

[8] Yu-Sheng Chen, Yu-Ching Wang, Man-Hsin Kao, and Yung-Yu Chuang. Deep Photo Enhancer: Unpaired Learning for Image Enhancement From Photographs With GANs. pages 6306–6314, 2018.

[9] Alison L. Coil. Large scale structure of the universe - https://ned.ipac.caltech.edu/level5/march12/coil/coil2.html.

[10] Yubin Deng, Chen Change Loy, and Xiaoou Tang. Aesthetic-Driven Image Enhancement by Adversarial Learning. In *Proceedings of the 26th ACM international conference on Multimedia*, MM '18, pages 870–878, Seoul, Republic of Korea, October 2018. Association for Computing Machinery.

[11] J. Einasto, G. Hütsi, T. Kuutma, and M. Einasto. Correlation function: biasing and fractal properties of the cosmic web. *Astronomy & Astrophysics*, 640:A47, August 2020. arXiv: 2002.02813.

[12] Jose Gaite and Geza Kovacs. The fractal geometry of the cosmic web and its formation. *Advances in Astronomy*, December 2019.

[13] Santiago Gonzalez, Mohak Kant, and Risto Miikkulainen. Evolving GAN Formulations for Higher Quality Image Synthesis. *Neural and Evolutionary Computing*, February 2021. arXiv: 2102.08578.

[14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. June 2014.

[15] Johan Hidding, Rien van de Weygaert, Gert Vegter, Bernard J. T. Jones, and Monique Teillaud. The Sticky Geometry of the Cosmic Web. *Symposium on Computational Geometry*, May 2012. arXiv: 1205.1669.

[16] Istvan Horvath, Jon Hakkila, and Zsolt Bagoly. Possible structure in the grb sky distribution at redshift two. *Astronomy Astrophysics*, Jan 2014.

[17] Weiwei Hu and Ying Tan. Generating Adversarial Malware Examples for Black-Box Attacks Based on GAN. *arXiv:1702.05983 [cs]*, February 2017. arXiv: 1702.05983.

[18] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics*, 36(4):1–14, July 2017.

[19] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-To-Image Translation With Conditional Adversarial Networks. pages 1125–1134, 2017.

[20] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks, 2017.

[21] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. *Computer Vision and Pattern Recognition*, March 2016. arXiv: 1603.08155.

[22] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. *International Conference on Learning Representations*, May 2014. arXiv: 1312.6114.

[23] J. S. Klar and J. P. Mücket. Filaments and sheets of the warm-hot intergalactic medium. *Monthly Notices of the Royal Astronomical Society*, Mar 2012.

[24] Doogesh Kodi Ramanah, Tom Charnock, Francisco Villaescusa-Navarro, and Benjamin D Wandelt. Super-resolution emulator of cosmological simulations using deep physical models. *Monthly Notices of the Royal Astronomical Society*, 495(4):4227–4236, July 2020.

[25] Jernej Kos, Ian Fischer, and Dawn Song. Adversarial Examples for Generative Models. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 36–42, May 2018.

[26] Stephen D. Landy and Alexander S. Szalay. Bias and Variance of Angular Correlation Functions. , 412:64, July 1993.

[27] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. pages 4681–4690, 2017.

[28] Yin Li, Yueying Ni, Rupert A. C. Croft, Tiziana Di Matteo, Simeon Bird, and Yu Feng. AI-assisted superresolution cosmological simulations. *Proceedings of the National Academy of Sciences*, 118(19), May 2021.

[29] Noam I Libeskind, Rien van de Weygaert, Marius Cautun, Bridget Falck, Elmo Tempel, Tom Abel, Mehmet Alpaslan, Miguel A. Aragoon-Calvo, Jaime E. Forero-Romero, Roberto Gonzalez, and et al. Tracing the cosmic web. *Monthly Notices of the Royal Astronomical Society*, 473:1195–1217, May 2017.

[30] Mustafa Mustafa, Deborah Bard, Wahid Bhimji, Zarija Lukić, Rami Al-Rfou, and Jan M. Kratochvil. CosmoGAN: creating high-fidelity weak lensing convergence maps using Generative Adversarial Networks. *Computational Astrophysics and Cosmology*, 6(1):1, December 2019. arXiv: 1706.02390.

[31] Perraudin Nathanaël, Srivastava Ankit, Tomasz Kacprzak, Aurelien Lucchi, Thomas Hofmann, and Alexandre Réfrégier. Cosmological n-body simulations: a challenge for scalable generative models. 2019.

[32] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[33] Nathanaël Perraudin, Ankit Srivastava, Aurelien Lucchi, Tomasz Kacprzak, Thomas Hofmann, and Alexandre Réfrégier. Cosmological N-body simulations:

a challenge for scalable generative models. *Computational Astrophysics and Cosmology*, December 2019. arXiv: 1908.05519.

[34] Tom Peterka, Juliana Kwan, Adrian Pope, Hal Finkel, Katrin Heitmann, Salman Habib, Jingyuan Wang, and George Zagaris. Meshing the Universe: Integrating Analysis in Cosmological Simulations. In *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*, pages 186–195, November 2012.

[35] Giuseppe Puglisi and Xiran Bai. Inpainting Galactic Foreground Intensity and Polarization maps using Convolutional Neural Network. *The Astrophysical Journal*, 905(2):143, December 2020. arXiv: 2003.13691.

[36] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training.

[37] Andres C. Rodriguez, Tomasz Kacprzak, Aurelien Lucchi, Adam Amara, Raphael Sgier, Janis Fluri, Thomas Hofmann, and Alexandre Réfrégier. Fast cosmic web simulations with generative adversarial networks. *Computational Astrophysics and Cosmology*, 5(1):4, December 2018. arXiv: 1801.09070.

[38] Andres C. Rodriguez, Tomasz Kacprzak, Aurelien Lucchi, Adam Amara, Raphael Sgier, Janis Fluri, Thomas Hofmann, and Alexandre Réfrégier. Fast cosmic web simulations with generative adversarial networks. *Computational Astrophysics and Cosmology*, 5(1):4, December 2018. arXiv: 1801.09070.

[39] Aldo Rodriguez-Puebla, Peter Behroozi, Joel Primack, Anatoly Klypin, Christoph

Lee, and Doug Hellinger. Halo and subhalo demographics with planck cosmological parameters: Bolshoi-planck and multidark-planck simulations, 2016.

[40] Joop Schaye, Robert A. Crain, Richard G. Bower, Michelle Furlong, Matthieu Schaller, Tom Theuns, Claudio Dalla Vecchia, Carlos S. Frenk, I. G. McCarthy, John C. Helly, and et al. The eagle project: Simulating the evolution and assembly of galaxies and their environments, 2015.

[41] Alex Sherstinsky. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network. *Physica D: Nonlinear Phenomena*, 404:132306, March 2020. arXiv: 1808.03314.

[42] Rien van de Weygaert, Miguel A. Aragon-Calvo, Bernard J. T. Jones, and Erwin Platen. Geometry and Morphology of the Cosmic Web: Analyzing Spatial Patterns in the Universe. *Sixth International Symposium on Voronoi Diagrams*, 2009. arXiv: 0912.3448.

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *31st International Conference on Neural Information Processing Systems*, December 2017. arXiv: 1706.03762.

[44] Mark Vogelsberger, Federico Marinacci, Paul Torrey, and Ewald Puchwein. Cosmological simulations of galaxy formation. *Nature Reviews Physics*, 2(1):42–66, 2020.

[45] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-Resolution Image Synthesis and Semantic Manipulation With Conditional GANs. pages 8798–8807, 2018.

[46] Huikai Wu, Shuai Zheng, Junge Zhang, and Kaiqi Huang. GP-GAN: Towards Realistic High-Resolution Image Blending. *Computer Vision and Pattern Recognition*, August 2019. arXiv: 1703.07195.

[47] Chika Yinka-Banjo and Ogban-Asuquo Ugot. A review of generative adversarial networks and its application in cybersecurity. *Artificial Intelligence Review*, 53(3):1721–1736, 2019.

[48] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S. Huang. Generative Image Inpainting With Contextual Attention. pages 5505–5514, 2018.

[49] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks. pages 2223–2232, 2017.

Appendix A

Detailed Architecture

Sequential Upsampling Layer (SUL) in fig A.1 and Sequential Downsampling Layer(SDL) in fig A.2 are used by the UNet-256 model in fig A.3. All code for this project was written in pytorch library [32].

In all SUL layers, the negative slope value for leaky relu layer is 0.2, while the inplace parameter is set to True. The kernel size for convolutional layer is (4 x 4), while the stride size is (2 x 2) and padding is (1 x 1).

In all SDL layers, the inplace parameter for relus is set to True. The kernel size for transposed convolutional layer is (4 x 4), while the stride size is (2 x 2) and padding is (1 x 1). The probability of trainable parameter being made 0 (p) is 0.5.

The same architecture is used in the 3D GAN as well, while the 2D convolutions and 2D transposed convolutions are replaced by 3D convolutions and 3D transposed convolutions respectively.

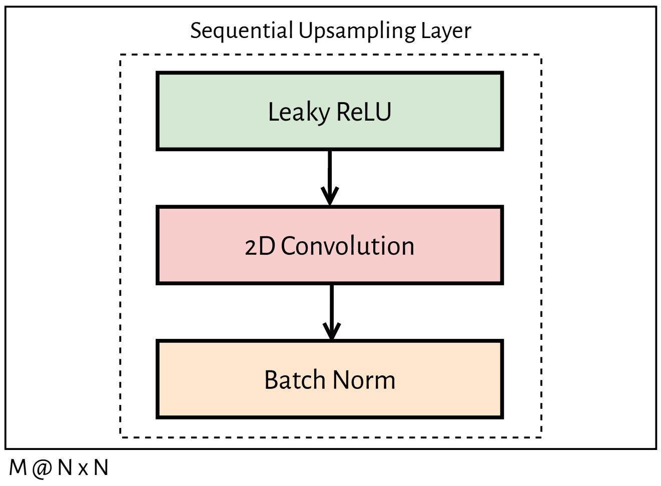


Figure A.1: Sequential Upsampling Layer (SUL) as used in unet-256 for 2d experiments consists of three layers; leaky relu, 2d convolution and batch norm. The bottom left tag mentions number of layers (M) and the shape of layers (NxN).

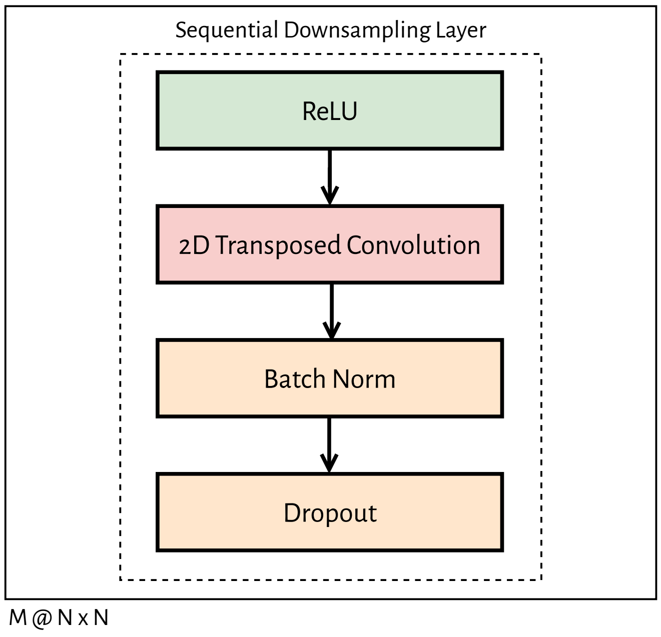


Figure A.2: Sequential Downsampling Layer (SDL) as used in unet-256 for 2d experiments consists of relu, 2d transposed convolutions, batch norm and a dropout layer.

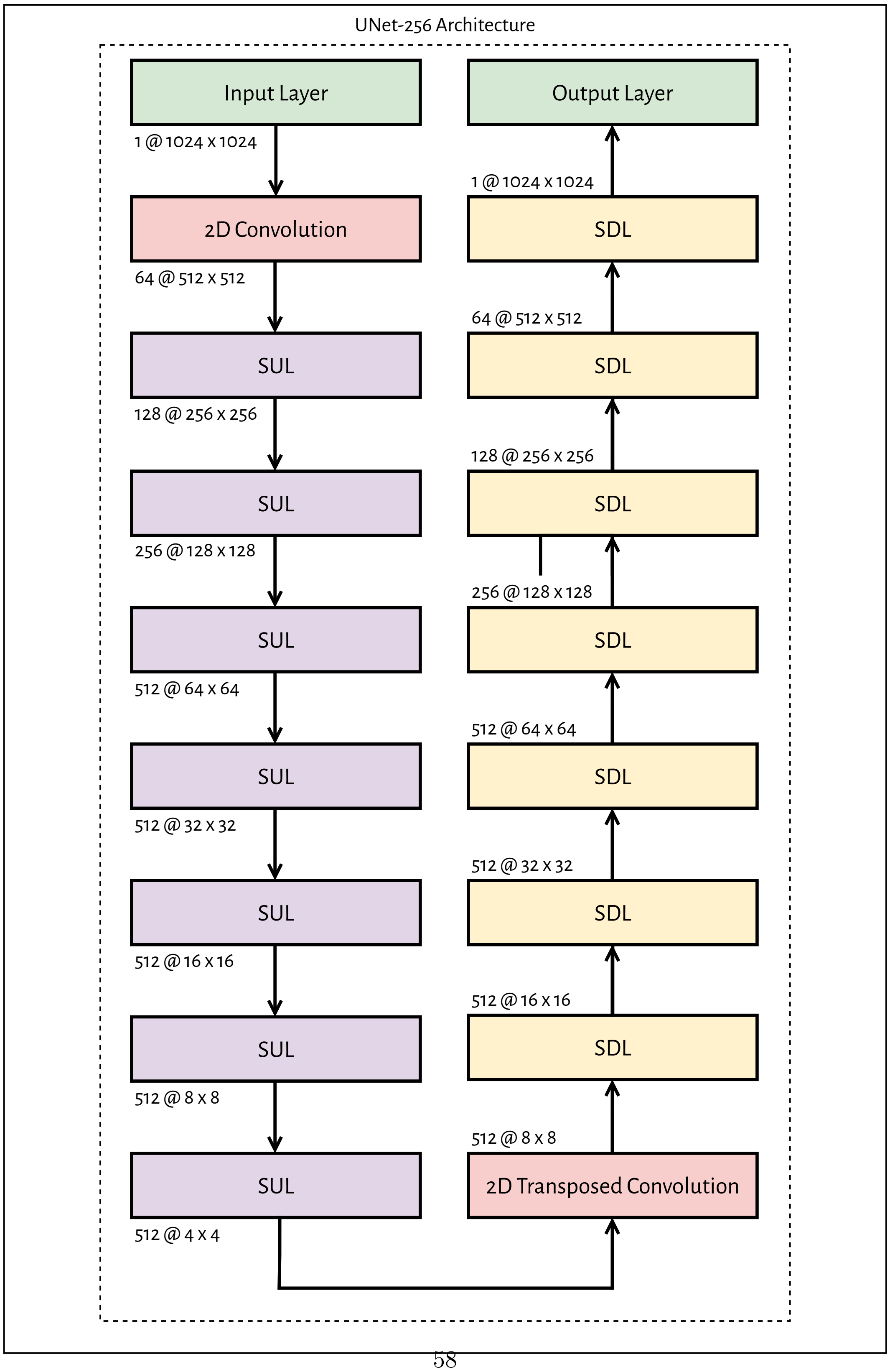


Figure A.3: UNet-256 Architecture as used in 2D experiments. The 3D experiment used the same model, except for using 3D convolutions and transposed convolutions.