Summary Driven WGAN for LSS

Lorenzo Cavezza†

***Abstract*—The frontiers of Generative Models in image generation have been pushed in the last few years by architectures such as Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs). Such revolutions have made their way into scientific research and have particularly been of great interest in Large Scale Structure Cosmology (LSS), as these new approaches removed the need for the time-intensive and computationally costly simulations used for LSS generation. In this paper we build a Wasserstein DCGAN based architecture for such task proposing a new kind of summary driven loss to enhance the training of the generator in the first epochs. We also test the system's convergence with appropriate domain agnostic metrics such as the duality gap. The results achieved were not optimal but still really good as an excellent compatibility between real and fake summary statistics was reached. This paper can be easily scaled to patch based approaches in order to upsample the image size further.**

***Index Terms*—Deep Learning, Generative Adversarial Networks, Wasserstein GANs, Large Scale Structure Cosmology, N-body Simulations**

I. Introduction

The production of statistically independent simulations of the *Cosmic Web* has become of crucial importance for Large Scale Structure Cosmology (LSS) as its complicated features can tell a lot about the physical processes that originated it. Recently machine learning inference techniques have been developed to study such structures bringing the need for a very high number of LSS replicas in order to properly train them. Given there is just one Observable Universe the demand for generative models is clear.

In the past these simulations were created using state-of-the-art N-body techniques: numerical integration-based evolutions of initial conditions through cosmic time according to the laws of Gravity and Magnetohydrodynamics which rendered unique samples of the LSS structure. These computations were really demanding in cost and time as one single simulation can take up to several hours to complete. Faster approaches recently introduced [1] [2] leveraged newly developed generative deep learning models such as *Generative Adversarial Networks* (*GANs*) to shorten the computation time by several orders of magnitudes. These remarkable approximants of highly complex multidimensional probability distributions are perfect tools for the generation of the deeply non-Gaussian features of the LSS.

Unlike other generative architectures they don't model the distribution explicitly but rather sample from it through a zero-sum game between two players: a *generator* that produces fake samples and a *discriminator* that assesses the closeness of the generated samples to the real ones. The training of the two networks leverages game theory to define a loss function which mimics the adversarial behaviour between the two agents.

Despite the great results achieved Vanilla *GANs* come with several mathematical and empirical disadvantages which make their stability and convergence very delicate. New models such as *Wasserstein GANs* have tried and successfully solved many of these issues. In this paper we try to further improve on such models by introducing a *Summary Loss* term to initially guide the training and improve convergence, as well as using spectral normalization in the discriminator and residual convolutional blocks in the generator to enhance the model's structures capturing capabilities. We also test the GAN convergence with a Duality Gap test as proposed in [3] and evaluate the generated samples with ad hoc summary statistics, as well as test their compatibility with respect to the real samples

The paper's structure is the following: we briefly describe DCWGANs and their use in recent literature in Section II, in section III and IV we go over the proposed architecture and the data used respectively, while in section V we'll discuss in detail all the aspects regarding the proposed model and see its results in section VI. Concluding remarks are provided in Section VII.

II. Wasserstein GANs

Generative Adversarial Networks have been proposed [4] as an implicit method of generating the probability distribution of the real data $p_{data}(\mathbf{x})$.

To do so we sample from a noise input $p_z(z)$ and feed it to a differentiable function $G(z;\theta_g) : z \to \mathbf{x}_g$, which represents the Generator with parameters θ_g. The critic is defined as another differentiable function $D(x;\theta_d) : \mathbf{x} \to \mathbb{R}$ which outputs a scalar indicating the probability of the input coming from the $p_{data}(\mathbf{x})$ rather than p_g. One crucial aspect is that the Generator may never see the data that it aims at faking, but only the critic's verdicts on its results. The game involves maximising the probability of D assigning the correct label to true and fake data, while the generator tries to minimize it to fool the critic, in other words the objective function is:

$$\min_G \max_D \left\{ \mathbb{E}_{x\sim p_{data}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{z\sim p_z(z)} [\log (1 - D(G(z)))] \right\}$$

This comes with some disadvantages as G has vanishing gradients when $D(G(z)) \sim 0$. A trick to solve this is to define a new loss as:

$$\min_G \max_D \left\{ \mathbb{E}_{x\sim p_{data}(\mathbf{x})} [\log D(\mathbf{x})] - \mathbb{E}_{z\sim p_z(z)} [\log (D(G(z)))] \right\} \sim$$
$$\sim \min_G \max_D \left\{ \mathbb{E}_{x\sim p_{data}(\mathbf{x})} [D(\mathbf{x})] - \mathbb{E}_{z\sim p_z(z)} [D(G(z))] \right\}$$

†Department of Physics and Astronomy, University of Padova, email: lorenzo.cavezza@studenti.unipd.it

Training a GAN essentially results in minimizing the distance between the fake and real probability distributions. Its training's convergence and stability depends in particular on how weak the topology induced by such distance is, as weaker distances allow for better uniform convergence of distributions' sequences and better overall differentiability. Vanilla approaches generally approximate the Kullback-Leibler divergence or the Jensen-Shannon distance. A distance weaker and therefore better than them is the *Earth-Mover* (EM) distance or Wasserstein-1:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$$

where $\gamma(x, y)$ represents how much probability mass you have to move to get from $\mathbb{P}_r$ to $\mathbb{P}_g$ and the distance itself is the cost of the optimal transport plan. Now the Kantorovich-Rubenstein duality tells us that:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_g}[f(x)]$$

where the supremum is over all the 1-Lipschitz functions. Now such distance clearly resembles the discriminator's view of the GAN's loss. It is therefore sufficient to find a function f that maximises $W(\mathbb{P}_r, \mathbb{P}_g)$ in order to approximate such distance. It is clear that such function is the GAN's optimal discriminator which we find maximising the adversarial loss. One crucial condition on the discriminator is that the function it approximates must be 1-Lipschitz. We can ensure this in two main ways:

1) Weight Clipping (WC): clamping the weights ω that parametrize the approximation in a compact space
2) Gradient Penalty (GP): adding a penalty term on the norm of the gradient of D, namely enforcing it to be lower than 1

WC can lead to great instability in training as either vanishing or exploding gradients were observed if not properly tuned [5]. GP solves these issues by adding to the discriminator's loss [6]:

$$\lambda \mathbb{E}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}\left[(\|\nabla_{\hat{x}} D(\hat{x})\| - 1)^2\right]$$

where $\hat{x}$ are samples belonging to lines between pairs of samples from $\mathbb{P}_r$ and $\mathbb{P}_g$. This allows to constrain the norm in a reasonably selected region rather than on all the available space, which would be intractable and useless.

Among papers attempting our same task the first was [1]. Here they develop a WGAN-GP as well which is smaller in size but that obtains remarkable results. They manage to replicate the LSS filaments and structures and to produce statistically independent samples. However they observe problems at small scales, as we also will find.

Another more advanced and notable approach is in [2] where they improve upon the previous one by using a patch-based upsampling technique with different WGANs for each level as well as conditioning on neighbouring patches to preserve the long range structures. While both of these papers try to emulate boxes with a high number of voxels (up to 256^3) this paper will try to generate smaller snapshots (32^3) given the limited hardware at hand. The methods we propose can however be used with more advanced architecture producing higher voxels boxes.

III. Processing Pipeline

The main pipeline consists in alternating the training between the generator and discriminator following the adversarial loss. The discriminator needs to be optimal to perfectly train the generator, but training it until convergence for every generator update would be computationally intractable. A compromise is to train the discriminator for a finite number of epochs everytime we update the generator. The discriminator itself takes as input a generated or real box. Thanks to many 3D Convolutional layers it downsamples the input to a smaller cube and learns its features while doing it. Then the cube is flattened and goes through an MLP to a single final values approximating the W-1 distance. The generator is instead made of several 3D Transposed Convolutional layers which upsample the initial noise input into a generated cube. It also contains some residual 3D Convolutional layers for better capturing the structures and improve gradient flow.

Both Generator and Discriminator have normalization and non-linear layers between upsampling/downsampling actions. Now the training of the discriminator relies only on computing the adversarial GP upgraded loss, where we feed the discriminator with batches of real and fake samples to find the adversarial loss, with the fake data being produced by the past iteration's generator. After the discriminator trains for a number of epochs the generator does one single update using the summary loss upgraded generator loss, which uses the newly trained discriminator on generated data to compute the generator's loss in its more stable form as the described in the last section. Lastly before each epoch the duality gap is computed: we train the best discriminator and generator unilateraly and compute the difference between their adversarial losses. If the game has reached an equilibrium the duality gap will result close to 0 as better explained in section V. The pipeline described is summarized in image 1.

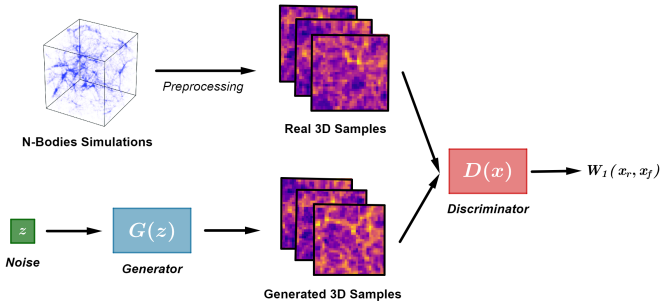

Fig. 1: WGAN Training Pipeline

IV. Dataset and Preprocessing

The raw data used consists of 512 N-bodies simulations from the Indra suite [7]. Each simulation is made of 1024^3 particles inside a box of length $1h^{-1}Gpc$ evolved using the code L-Gadget2 and WMAP7 cosmological parameters. The first step in preprocessing the simulations consisted in building 3D histograms of size (256, 256, 256) reflecting the value of the underlying matter overdensity in each voxel. We do this by simply binning the particles inside a 3D histogram.
Then in order to augment the dataset we exploited the homogeneity and isotropy of the universe to randomly shift and rotate the histograms so as to obtain new simulations. We obtained about 800 new boxes for a total of about 1300 snapshots. Then we selected random sections of smaller size (32, 32, 32) inside each simulation in order to build the final dataset to use (the boxes have size $125Mpc^3$)
As we will see in section VI the original distributions show a very skewed distribution towards 0, as we expect it to follow a log-normal distribution. Furthermore, the values expand well outside the range [0,1]. Generative models notoriously struggle with very sparse and skewed distributions so a further transformation was applied before training to fix this problem. Given a distribution x the forward transformation is defined as: $f(x, c, s) = f'(x + s, c) - f'(s, c)$ with:

$$f'(x, c) = \begin{cases} 3\log_e(x+1) & \text{if } x \leq c, \\ 3(\log_e(c+1) + \frac{x}{c} - 1) & \text{otherwise} \end{cases}$$

The values chosen for c and s where 11 and 0.002 respectively. This together with a normalization step defines a new distribution which is less skewed and defined in [0,1].
The train/validation split used was 992 snapshots for training divided in batches of size 16, 128 for the final validation and 128 for computing the Duality Gap in batches of 8, for a final split of roughly 80/10/10.

V. Learning Framework

We begin now to analyze the architecture proposed in detail so as to explain its features and motives.
As said before the **Discriminator** is made of several 3D Convolutional layers with a final MLP. The structure is detailed in table 1 and visually represented in image 2:

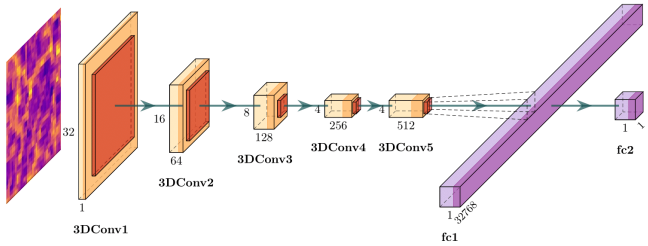

Fig. 2: Discriminator

In table 1 we can see the different discriminator's layers. The parameters used in the convolution layers are input channels, output channels, kernel size, stride, padding and dilation. The output shape formula is:

Operation	Parameters (i, o, k, s, p, d)	Output
Input box		(b, 32, 32, 32, 1)
Conv 3D	(1, 64, 4, 2, 1, 1)	(b, 16, 16, 16, 64)
Spec. Norm		(b, 16, 16, 16, 64)
LReLU	($\alpha = 0.2$)	(b, 16, 16, 16, 64)
Conv 3D	(64, 128, 4, 2, 1, 1)	(b, 8, 8, 8, 128)
Spec. Norm		(b, 8, 8, 8, 128)
LReLU	($\alpha = 0.2$)	(b, 8, 8, 8, 128)
Conv 3D	(128, 256, 4, 2, 1, 1)	(b, 4, 4, 4, 256)
Spec. Norm		(b, 4, 4, 4, 256)
LReLU	($\alpha = 0.2$)	(b, 4, 4, 4, 256)
Conv 3D	(256, 512, 3, 1, 2, 2)	(b, 4, 4, 4, 512)
Spec. Norm		(b, 4, 4, 4, 512)
LReLU	($\alpha = 0.2$)	(b, 4, 4, 4, 512)
Conv 3D	(512, 512, 3, 1, 1, 1)	(b, 4, 4, 4, 512)
Spec. Norm		(b, 4, 4, 4, 512)
LReLU	($\alpha = 0.2$)	(b, 4, 4, 4, 512)
Reshape		(b, 32768)
Dense	(32768, 1)	(b, 1)

TABLE 1: Discriminator Structure

$$O = \frac{I + 2P - D(K-1) - 1}{S} + 1$$

We preferred downsampling quickly to a size of (4, 4, 4) to better focus on large scale features. Each layer has spectral normalization [8], which enforces Lipschitz continuity by dividing the weight matrix by its maximum singluar value (the spectral norm). The non-linearity used is Leaky ReLU, a choice widely used in the literature for GANs.
Regarding the **Generator**'s architecture it is made of several 3D Transposed Convolutional layers which up-sample the initial noise input up to the generated box. Its structure is explained in table 2 and image 3:

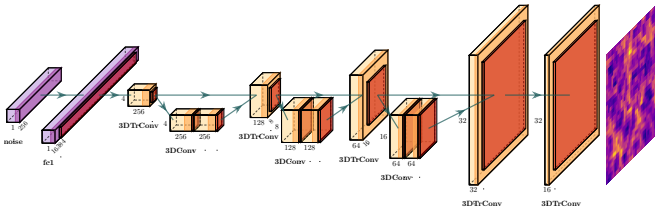

Fig. 3: Generator

The parameters used in the transposed convolutional layers are input channels, output channels, kernel size, stride and padding. The output shape formula is:

$$O = S(I-1) - 2P + K$$

Again Leaky ReLU was used together with Batch Normalization at each layer. A final Sigmoid layer constrains the output values between 0 and 1.

Operation	Parameters (i, o, k, s, p)	Output
$z \sim N(0,1)$		(b, 256)
Dense	(256,16384)	(b,16384)
Reshape		(b,4,4,4,256)
TrConv 3D	(256, 128, 4, 2, 1)	(b, 8, 8, 8, 128)
BatchNorm3D		(b, 8, 8, 8, 128)
LReLU	($\alpha = 0.2$)	(b, 8, 8, 8, 128)
ResBlock		(b, 8, 8, 8, 128)
TrConv 3D	(128, 64, 4, 2, 1)	(b, 16, 16, 16, 64)
BatchNorm3D		(b, 16, 16, 16, 64)
LReLU	($\alpha = 0.2$)	(b, 16, 16, 16, 64)
ResBlock		(b, 16, 16, 16, 64)
TrConv 3D	(64, 32, 4, 2, 1)	(b, 32, 32, 32, 32)
BatchNorm3D		(b, 32, 32, 32, 32)
LReLU	($\alpha = 0.2$)	(b, 32, 32, 32, 32)
ResBlock		(b, 32, 32, 32, 32)
TrConv 3D	(32, 16, 3, 1, 1)	(b, 32, 32, 32, 16)
BatchNorm3D		(b, 32, 32, 32, 16)
LReLU	($\alpha = 0.2$)	(b, 32, 32, 32, 16)
TrConv 3D	(16, 1, 3, 1, 1)	(b, 32, 32, 32, 1)
Sigmoid		(b, 32, 32, 32, 1)

TABLE 2: Generator Structure

After the first 3 transposed convolutional layers Residual Blocks are used with structure shown in table 3. They are composed of 3D Convolutional layers wich to do not alter the input shape but just help learning the structure's features better. Normal convolution was used because we didn't need to upsample the input box and because it is less computationally demanding than transposed convolutions.

Operation	Parameters (i, o, k, s, p)	Output
Input box		(b, l, l, l ch)
Conv 3D	(ch, ch, 3, 1, 1)	(b, l, l, l ch)
BatchNorm3D		(b, l, l, l ch)
LReLU	($\alpha = 0.2$)	(b, l, l, l ch)
Conv 3D	(ch, ch, 3, 1, 1)	(b, l, l, l ch)
BatchNorm3D		(b, l, l, l ch)
LReLU	($\alpha = 0.2$)	(b, l, l, l ch)
Sum	+ Input box	(b, l, l, l ch)

TABLE 3: Residual Block Structure

The training algorithm is shown in table 4. As we can see the discriminator is trained 5 times for every generator's update for the reasons explained in III. The optimizer used is ADAM with parameters shown in table 4 and a learning rate of 10^{-4}, a standard choice for WGANs which also resulted in the most stable training.

Gradient Penalty is implemented by adding a term to the discriminator's loss to penalize gradients of the discriminator with norm bigger than 1. Actually this enforces the norm to be around 1 from both ends. This is done to be less restrictive, as the norm is usually already below 1 in most of the space as explained in [6].

The generator's loss also has the **Summary Loss** term. Such element is composed in the following way. First we compute the summary statistics for each batch of real and fake data. The summaries used are:

- Mass Histogram: it is the histogram of the values inside each voxel. It represents how many voxels have a certain value. We expect it to be log-normally distributed
- Peak Histogram: it is the distribution of maxima in the density distribution. A peak is defined as a pixel greater than every pixel in its 2-pixels neighbourhood (124 pixels)
- Power Spectrum: it is the amplitude of the Fourier modes as a function of their frequency. It's related to the 2-point correlation function of the distributions

Then we compute the Wasserstein-1 distance defined in section II between the mean of each summary for real and fake data. Such distance is also weighted point-wise by the term:

$$e(\sigma_f, \mu_f, \sigma_r, \mu_r) = \left(\frac{\sigma_f}{\mu_f + \delta_f} + \frac{\sigma_r}{\mu_r + \delta_r} + \epsilon \right)^{-1}$$

with $\sigma_f$and σ_r the fake and real point-wise standard deviations of the summaries and μ_f and μ_r the fake and real point-wise means of the summaries (σ_f, σ_r and ϵ are used to avoid division by 0). Such term ensures that for a fixed mean W-1 distance the loss is greater if the relative error in that point is smaller, so as to be more stringent in low variance areas and vice versa.

The three distances obtained are then summed with weights shown in table 4, λ_{pk} for the power spectrum, λ_m for the mass histogram and λ_p for the peak histogram. Then the total summary loss is weighted in the loss by λ_{summ}.

The summary loss term was used in the first 170 epochs to help stabilize the initial training and was later removed in order to not interfere with more subtle changes in the adversarial game.

As a way to assess the GAN convergence we also compute the **Duality Gap** every 10 epochs, as explained in [3]. The gap is computed as $DG = L_{MAX} - L_{MIN}$ where L_{MAX} and L_{MIN} are computed as follows:

- L_{MAX}: we fix the generator and train the discriminator to "convergence" (we train it for 20 epochs on its dedicated dataset with a slightly higher learning rate than normal training). L_{MAX} will be the adversarial loss with the best newly trained discriminator and the old generator
- L_{MIN}: we fix the old discriminator and train the generator to "convergence" in the same way as before. L_{MIN} will be the adversarial loss with the best newly trained generator and the old discriminator

If the discriminator and generator computed at that particular main epoch can't unilaterally improve then the gap will be close to 0. After each gap computation cycle the old generator and discriminator are re-established to continue the main training.

Algorithm: $\lambda = 10$, $\lambda_{summ} = 0.3$, $\lambda_{pk} = 2$, $\lambda_m = 2$, $\lambda_p = 1$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$

While θ has not converged do
  For $i = 1, \ldots, n_{\text{critic}}$ do
    For $i = 1, \ldots, m$ do
      Sample real data $x \sim \mathbb{P}_r$, latent variable $z \sim p(z)$, $\epsilon \sim U[0, 1]$.
      $\hat{x} \leftarrow G_\theta(z)$
      $\hat{x} \leftarrow \epsilon x + (1 - \epsilon)\hat{x}$
      $L^{(i)} \leftarrow D_w(\hat{x}) - D_w(x) + \lambda \left(\|\nabla_{\hat{x}} D_w(\hat{x})\|_2 - 1\right)^2$
    End for
    Update critic: $w \leftarrow \text{Adam}\left(\nabla_w \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2\right)$
  End for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$
  $S(x, z) \leftarrow \lambda_{pk} W(P_k(G_\theta(z)), P_k(x)) + \lambda_m W(M_h(G_\theta(z)), M_h(x)) + \lambda_p W(P_h(G_\theta(z)), P_h(x))$
  Update generator: $\theta \leftarrow \text{Adam}\left(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_w(G_\theta(z)) + \lambda_{summ} S(x, z)), \theta, \alpha, \beta_1, \beta_2\right)$
End while

TABLE 4: WGAN with Gradient Penalty Algorithm

VI. Results

We now analyze the results obtained. We start by looking at the loss trends in image 4.

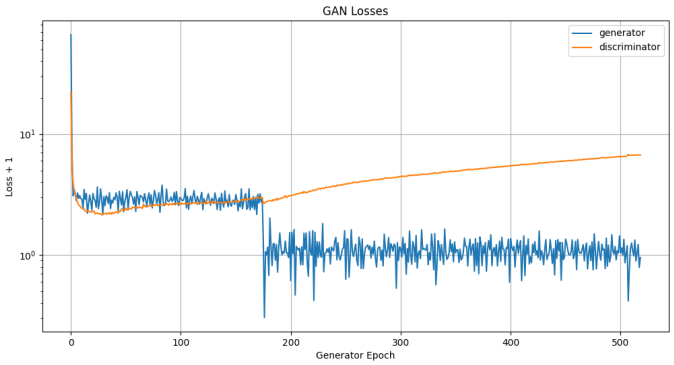

Fig. 4: Losses

As we can see the generator immediately goes down and then oscillates close to 0. After we remove the summary loss at about epoch 170 it decreases further and stabilizes. The discriminator immediately loses against the generator but then slowly increases, this means it is learning to tackle the generator. Unfortunately we cannot see a clear convergence of the discriminator due to too few epochs (about 500). However, we can see that the rising trend was starting to stabilize, maybe with more computational power and epochs we could have seen a convergence. Both the generator and discriminator are very stable.

To further assess the convergence of the GAN we can look at the Duality Gap in image 5.

We can see that the gap dropped near 0 and then started to oscillate. This could mean that a clear convergence was not reached yet as we already saw in the losses. More epochs are probably needed, but the increase in the oscillations could mean a problem in either the generator or discriminator structures. Looking at the losses we could say that the discriminator

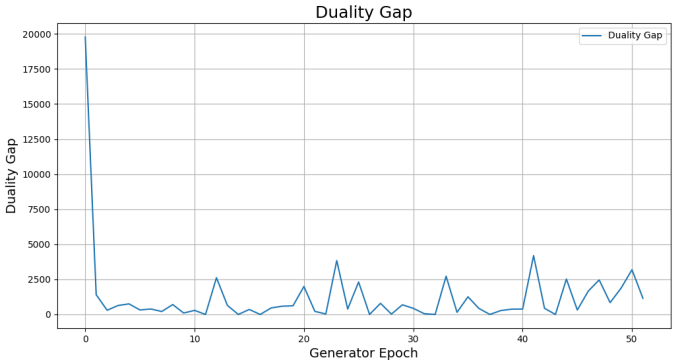

Fig. 5: Duality Gap

is a bit too weak for the generator.

Despite the not optimal results seen so far we can still look at the quality of the generated data by plotting 2D slice of real and generated samples as we've done in image 6. The similarity is remarkable, even though we can observe that in general the generated data appears to be slightly more bright and with more peaks. Furthermore the structures at low brightness are not exactly the same but still really similar. Again further training could have helped the generator learn the structures better.

We can better see this similarity in the total summaries plot as well in image 7. Here we used the testing dataset to plot the real and fake power spectrum, mass histogram and peak histogram, as well as the bispectrum. The latter measure the three point correlation inside the distribution and tells us about higher order non-gaussian statistics of the distribution. K1 and K2 are the length of two sides of the triangle used to compute it while on the x-axis θ is the angle of the triangle between the two sides in radians.

The power spectrum and bispectrum similarity is remarkable at most scales. This means that the model captures easily the non-gaussianity of the data, namely its filaments, sheets and voids. However, the mass histogram and peak histogram

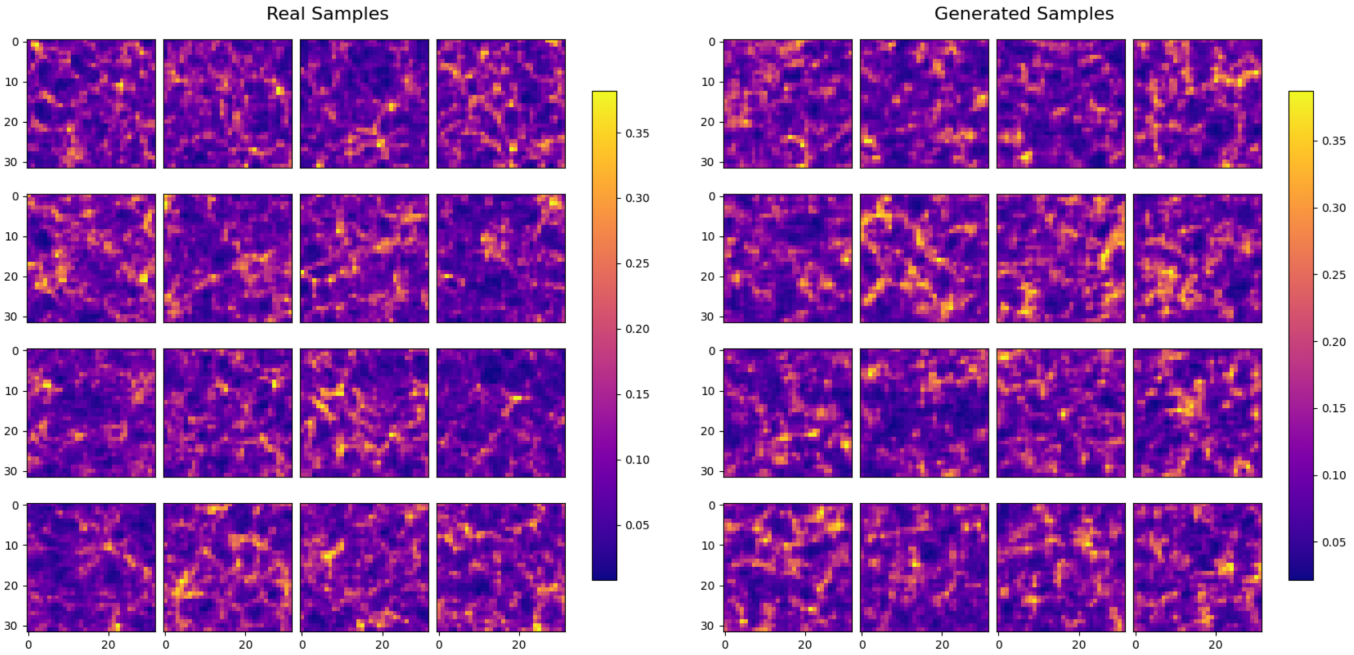

Fig. 6: 2D Slices of Real and Generated Samples

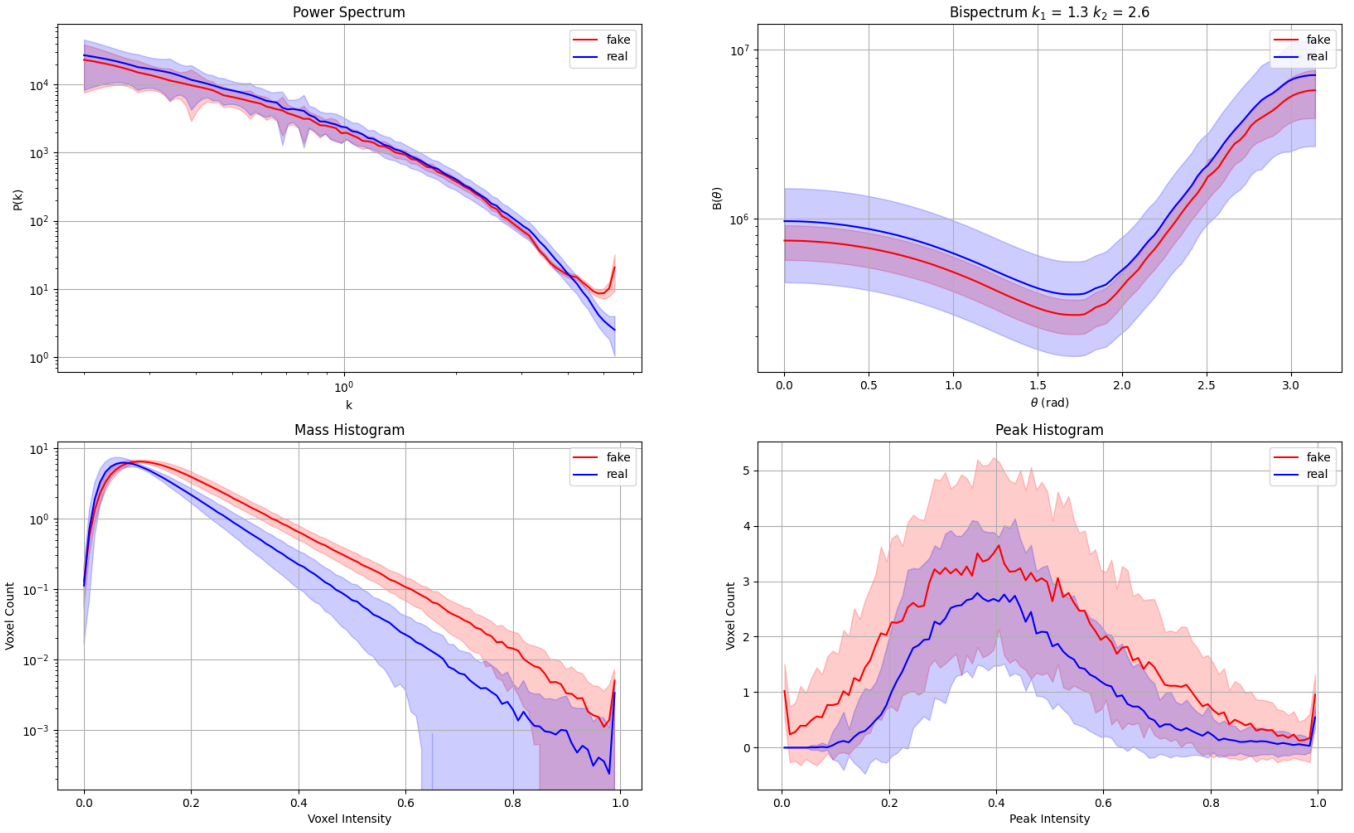

Fig. 7: Summaries

appear slightly shifted towards a brighter distribution. This can explain what we observed in the generated samples. It can also be seen in the power spectrum as the only area where there is not a clear similarity is for low scales (high k). It could be a symptom of a possible architecture's weakness at capturing the small scales well. This shift could also explain the slight underestimation of the power spectrum at large scales, which is however well inside the error bar. Unfortunately we weren't able to solve this problem but we can say that we are well satisfied by the results nonetheless. As a last quality measure we compute the compatibility between real and fake data:

$$c(p_f, p_r) = \frac{|\mu_f - \mu_r|}{\sqrt{\sigma_f^2 + \sigma_r^2}}$$

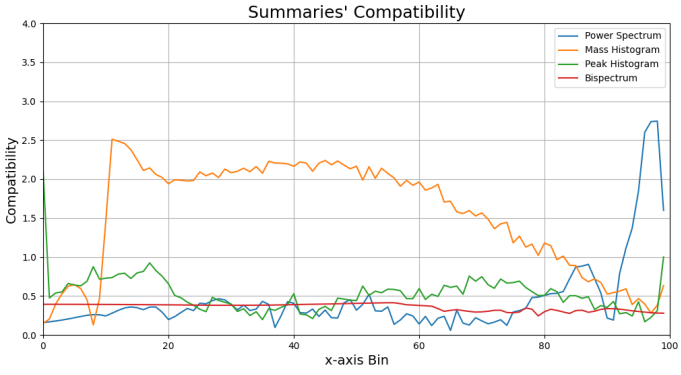

Fig. 8: Compatibility

The compatibility is really good for all bins of the summaries except for the power spectrum at low scales and the overall mass histogram where we can see a discrete compatibility.

VII. Concluding Remarks

We were able to build a WGAN-GP architecture to properly reproduce 3D histograms of the large scale structure of the universe starting from N-body simulations. As seen in section VI the overall quality of the results is good, even though a slight shift towards a brighter distribution was observed. This together with the lack of proper convergence of the discriminator could indicate that the discriminator's architecture was not optimal as it could not capture and transmit the structures perfectly to the generator, particularly at small scales.

The small scales are inherently very non-gaussian and the small size of our cubes (just about 125 Mpc) surely didn't help. Problems with this scale was also observed by [1]. The possible weakness of the discriminator could also imply that a deeper structure is needed.

However, this is a very good start for a possible patch-based approach as done in [2]. The summary loss appears to have helped the training in the first epochs and can therefore be used in general to enhance the training. The next steps would be to improve the discriminator in order to correct the shift observed and to try and build an approach similar to [2].

As for the project goes I have learned really well to program in pytorch as well as all the nuances of GAN's training. The architectures were really delicate and prone to mode collapse. Furthermore the generator initially struggled to learn the structures, hence its deep architecture. Even though the results were not perfect I can say that I'm satisfied given the difficulty of the task.

References

[1] A. Rodriguez, T. Kacprzak, A. Lucchi, A. Amara, R. Sgier, J. Fluri, T. Hofmann, and A. Refregie, "Fast Cosmic Web Simulations with Generative Adversarial Networks," in *Arxiv*, (ETH Zurich, Universitatstrasse 6, 8092 Zurich, Switzerland), Nov. 2018.

[2] N. Perraudin, A. Srivastava, A. Lucchi, T. Kacprzak, T. Hofmann, and A. Refregier, "Cosmological N-body simulations: a challenge for scalable generative models," in *Computational Astrophysics and Cosmology Springer Journal*, (ETH Zurich, Universitatstrasse 6, 8092 Zurich, Switzerland), Dec. 2019.

[3] P. Grnarova, K. Y. Levy, A. Lucchi, N. Perraudin, I. Goodfellow, T. Hofmann, and A. Krause, "A Domain Agnostic Measure for Monitoring and Evaluating GANs," in *Arxiv*, (ETH Zurich, Universitatstrasse 6, 8092 Zurich, Switzerland), July 2020.

[4] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems 27*, (Departement d informatique et de recherche operationnelle Universite de Montreal), June 2014.

[5] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," in *Arxiv*, (Courant Institute of Mathematical Sciences, Facebook AI Research), Dec. 2017.

[6] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved Training of Wasserstein GAN," in *Arxiv*, (Montreal Institute for Learning Algorithms, Courant Institute of Mathematical Sciences, CIFAR Fellow), Dec. 2017.

[7] B. Falck, J. Wang, A. Jenkins, G. Lemson, D. Medvedev, M. C. Neyrinck, and A. S. Szalay, "Indra: a Public Computationally Accessible Suite of Cosmological N-body Simulations," in *Monthly Notices of the Royal Astronomical Society*, (Department of Physics and Astronomy, Johns Hopkins University, 3400 N Charles St, Baltimore, MD 21218, USA), Aug. 2021.

[8] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral Normalization for Generative Adversarial Networks," in *6th International Conference on Learning Representations*, (Ritsumeikan University), Feb. 2018.