Galaxy Image Generation: Computational Intelligence Laboratory Report

Mounir Amrani
ETH Zürich
mamrani@student.ethz.ch

Philippe Andreu
ETH Zürich
pandreu@student.ethz.ch

Gabriel Hayat
ETH Zürich
ghayat@student.ethz.ch

Émilien Pilloud
ETH Zürich
epilloud@student.ethz.ch

***Abstract*—We propose a GAN [1] model based on the DC-GAN model [2] to generate 64x64 images, followed by a Super Resolution network upsampling these images to 1000x1000. We aim to have a diverse generation set while preserving the sharpness seen in the training images.**

I. Introduction

With the recent increase in available cosmological data, researchers have naturally turned to techniques capable of handling the abundance of data for analysis and exploration of the digital cosmos [3] [4]. Scientists have looked to gain deeper understanding of various celestial bodies such as stars, galaxies [5] and more recently black holes.

Acquisition of cosmological data is quite involved and expensive. Therefore, interest has peaked in being able to generate such data at will, to be used for calibration data for e.g [6]. The quality and diversity of the generated data is a very important evaluation criterion for such images [7].

II. Models and Methods

We split our models into two distinct classes, with some of the models belonging simultaneously to both of the classes:

1) Generative Models for the Galaxy Image Generation task
2) Regression Models for the Score Regression task

For each of the model classes, we will present all the models we explored in an incremental fashion, building upon the previously detailed model at each step.

A. Data Preprocessing

1) Manual Features: Cosmological imagery classification has traditionally been a human performed task. We extracted manual features from the images to guide and facilitate the classification, including a normalized 20 bin histogram of the brightness values, the number of blobs detected by a Laplacian of Gaussian approach, a 9 bin histogram of shape indexes, the Shannon entropy of the image, the highest and lowest brightness values, the pixel variance and finally the approximation coefficients of the level in a multilevel 2D Discrete Wavelet Transform such that there are only 4 (i.e. a 2x2 array of approximation coefficients). This gives a total of 38 hand crafted features per image.

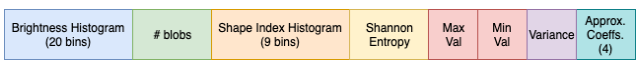


Figure 1: Manual Features extracted from the cosmology images

B. Generative Models

1) Patches Generator: This model generates an image of size 1000x1000 by tiling together 50x50 patches from galaxy images of the labeled dataset. This can be seen as splitting the 1000x1000 image to be generated into a grid of 50x50 cells then filling each cell by the corresponding cell (i,e at the same position) from a random galaxy image of the labeled dataset. A simple smoothing is then applied at the borders of these cells.

2) Stacked Super Resolution Model (SRM): Inspired by [8], we designed a model capable of upsampling 64x64 images to 1024x1024. Figure 2 summarizes the model's architecture. This model is composed of 4 stacks, each of which doubles the resolution of its input. A preprocessing step is applied as can be seen in the upper branch of Figure 2. This produces a set of 5 ground-truth sizes: 64x64, 128x128, 256x256, 512x512 and 1024x1024. The 1st stack's input has size 64x64.

We apply intermediate supervision on the output of each stack by applying the pixel-wise L2 loss as can be seen in Figure 2. Without this intermediate supervision, earlier layers have a harder time improving as the gradient information for early layers gets lost in deeper layers.

We refer to Figure 3 for the architecture of a single stack and a single residual block. In the residual module, the number of channels (i,e 64) is preserved and it is composed of 6 residual blocks. The upsampling type we apply is Nearest Neighbor upsampling. All convolutions have a padding of 1, and all inputs to the stacks are normalized to [0,1].

In order to speed the training up and give earlier stacks a head start to produce outputs with meaningful information that can be used by the next stack, we train our model incrementally. We first train stack 1, followed by stacks 1 and 2, after which we train stacks 1, 2 and 3, and finally training all stacks simultaneously. Each training procedure is done for 1/4 of the total the number of epochs. The training procedure i minimizes the sum of L2 losses applied to stacks $1...i$ for $i = 1, ..., 4$.

Training is performed with the Adam optimizer for 100 epochs on the labeled dataset with a batch size 4, a learning rate of $2 * 10^{-4}$, $\beta_1 = 0.5$ and $\beta_2 = 0.999$.

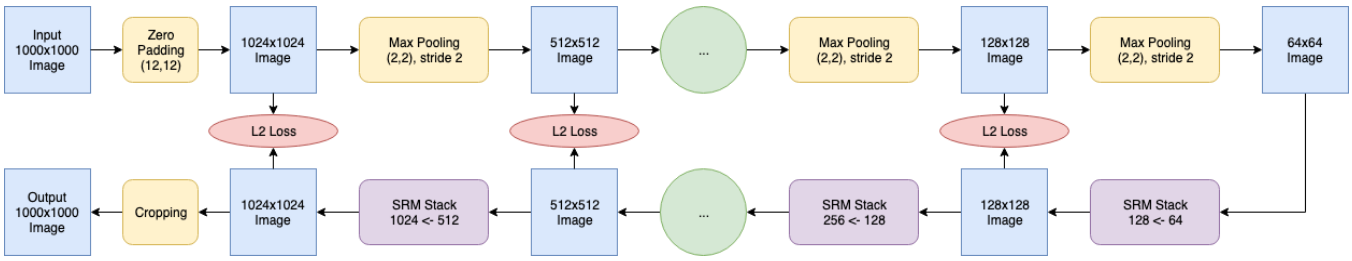


Figure 2: Diagram of the Stacked SRM

3) DCGAN: Our DCGAN model follows the model in [2] closely. The discriminator preprocesses the real images as in the SRM model, i.e. as in the upper branch in Figure 2 with the difference that we pad the value -1 since these images are normalized to [-1, 1]. The image of size 64x64 is then input to the DCGAN's discriminator. No preprocessing is applied to the generator's images. The input goes through 4 convolution blocks with output channels 128, 256, 512 and 1024. Each convolution block is a sequence of a convolution, a batch norm and leakyReLU layer with alpha=0.3. Each convolution layer has filter size 4 and stride 2. The output of the 4th convolution block is flattened and a dense layer with 2 units is applied. Optionally, we add a minibatch discrimination layer after the flattening but before applying the dense layer with 2 units.

The generator first applies a dense layer of 4*4*1024 units to the noise vector of dimension 1000 followed by a ReLU layer. We reshape the ReLU's output to get a 4x4 image of 1024 channels. 3 deconvolution blocks follow, with each deconv block doubling the image size and halving the number of channels. Each deconv block is a succession of a deconv layer, a batch norm layer and a ReLU layer. A last deconv layer is applied on the output of the third deconv block to bring the number of channels down to 1, which is then activated using tanh activation. All deconv layers have filter size 2 and stride 2.

Training is performed with Adam for 500 epochs on the labeled dataset with batch size 16, learning rate $2 * 10^{-4}$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for both the generator and discriminator.

4) Full Resolution GAN: This approach looks to directly generate 1000x1000 images using a standard GAN. As the architecture is fairly similar to DCGAN, we refer the reader to the attached code for more details.

5) MCGAN: In order to promote more diversity in the generated images, the Manual feature Conditioned GAN (MCGAN) generates images at a 64x64 resolution, conditioned on the number of blobs detected in the train image, which approximates the number of stars present. The 64x64 images are then upsampled using the SRM as in the DCGAN model. This architecture is similar to DCGAN, except for the manual feature conditioning. The number of blobs is a scalar which is input both to the discriminator and the generator. The generator receives the manual feature concatenated to the noise vector. The discriminator receives it as a second input, feeds it to a dense layer of 3 units with ReLU activation, followed by a batch normalization and a second non-activated dense layer of 1 unit. The discriminator's extracted features and the transformed manual feature are concatenated, and then fed to a final dense layer of two units which produces the logits.

Training is performed for 500 epochs on the labeled dataset with batch size 16, learning rate $2 * 10^{-4}$, $\beta_1 = 0.5$ and $\beta_2 = 0.999$ for both the generator and discriminator.

C. Scoring Models

1) Manual Feature Regressors: These models score the images based on the manual features we extract from them. We tested three different regressors for this task: XGBoost, Random Forest and Ridge regressors.

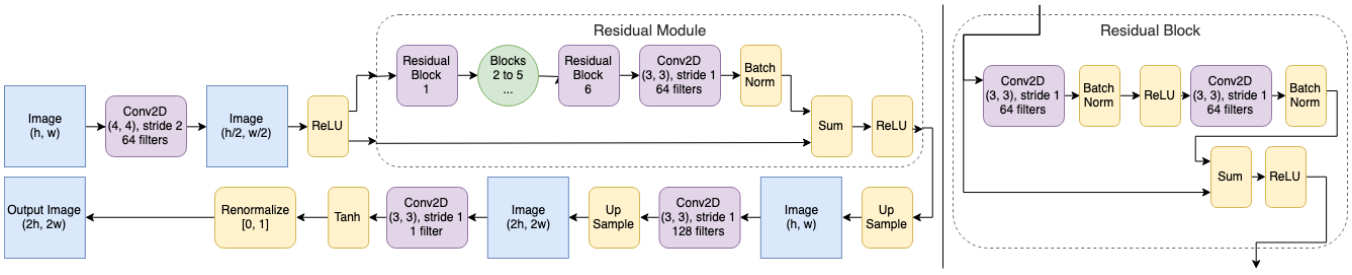


Figure 3: Left: Diagram of a single SRM Stack - Right: Diagram of a single Residual Block

2) DCGAN Scorer: We load the discriminator of the trained DCGAN and fetch the flattened features of size 4*4*1024. A scoring head, which is a simple feed-forward network with 4 dense layers with units 1024, 512, 256 and 1, is applied on these features in order to learn to predict the score. Dense layers 1, 2 and 3 are ReLU activated.

We feed the images from the scored dataset to the discriminator which is only used to get the features of the images and is not trained. Only the scoring head is trained using Adam with learning rate 10^{-3}, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for 100 epochs with batch size 16.

D. Extensions

We present here extensions that were added to some of our models.

1) Minibatch Discrimination (BD): We implemented a minibatch discrimination layer to encourage the generator to produce a diverse set of samples per batch.

2) Label Smoothing (LS): In order to avoid mode collapse, we added fixed label smoothing to some models to discourage overconfidence on the discriminator's part.

3) Data Augmentation (ROT): For some of our models, we experimented with randomly augmenting each batch by rotations of $k * 90°$ for $k \in \{0, ..., 3\}$.

III. Results

There are two distinct goals for this project: generating high-quality and diverse cosmological images, and using part of the generative model to learn a similarity function. To assess the quality of our scorer, we rely on the MAE metric.

1) Regression task: We consider the mean absolute error of our scoring models in table I.

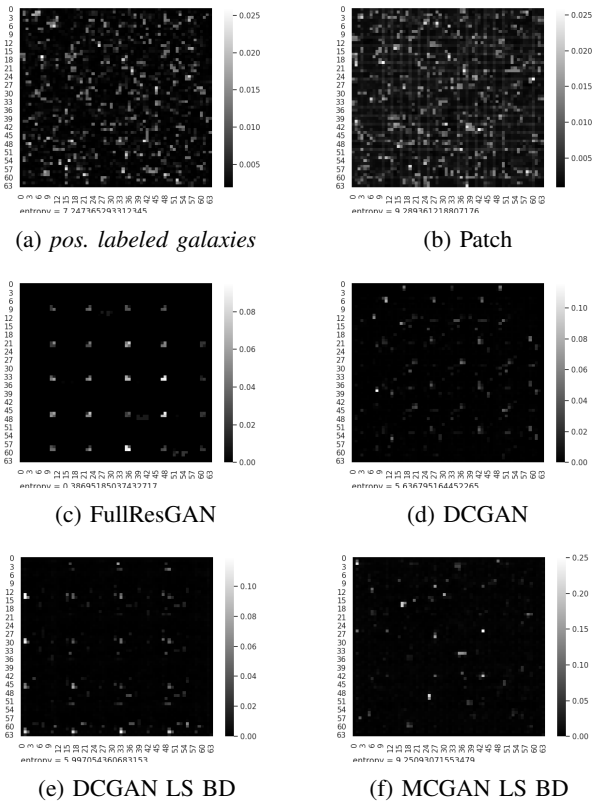
	public score	private score
Sample mean score	0.87506	0.83903
DCGAN scorer head	0.62572	0.67047
Random Forest on manual feats	0.19297	0.21237
XGBoost regressor on manual feats	0.19074	0.20897
Ridge regressor on manual feats	0.65078	0.63115

Table I: Score Regression Task

2) Predicted Scores from the BOOST baseline: Our most successful model for the regression task is the XGBoost baseline. We use it in table II to evaluate the quality of each of our models' generated samples.

	μ	σ	min	max
pos. labeled galaxies	1.94	0.9	0.26	4.51
PATCHES	1.37	1.07	0.01	3.36
FullResGAN LS BD	0.96	0.5	0.0	2.42
DCGAN	1.58	0.32	0.9	2.76
DCGAN LS BD	**1.82**	0.48	0.26	3.16
MCGAN LS BD	1.73	0.38	0.38	3.26
MCGAN LS BD ROT	1.78	**0.33**	**0.93**	**3.33**

Table II: XGBoost Estimated Scores



(a) *pos. labeled galaxies* (b) Patch

(c) FullResGAN (d) DCGAN

(e) DCGAN LS BD (f) MCGAN LS BD

Figure 4: Heatmaps of mean generated images

3) KNN-distance: The metric consists in finding the k nearest neighbors in the Minkowski distance sense, and computing the mean distance to those k neighbors across all generated samples. This captures how close our samples are to their neighbors. In the event of a total mode-collapse, the expected KNN-distance is hence 0.

		$k = 1$	$k = 3$	$k = 5$
positively labeled galaxies	μ	2.25	2.25	2.23
	σ	0.41	0.45	0.43
	min value	1.56	1.55	1.5
PATCHES	μ	1.93	1.96	1.99
	σ	0.6	0.59	0.58
	min value	0.5	0.57	0.68
FullresGAN LS BD	μ	0.81	1.05	1.20
	σ	0.75	0.73	0.74
	min value	0.0	0.0	0.0
DCGAN	μ	1.31	1.38	1.55
	σ	0.49	0.51	0.52
	min value	0.31	0.41	0.45
DCGAN LS BD	μ	1.48	1.6	1.65
	σ	0.48	0.48	0.49
	min value	0.31	0.45	0.42
MCGAN LS BD	μ	1.35	1.47	1.55
	σ	0.54	0.53	0.52
	min value	0.26	0.38	0.52

Table III: KNN-Distance results

4) Heatmaps: For a generated set of images, we summarize them by taking the pixel-wise mean across the set. Figure 4 displays the mean activation of each pixel and compares it to the positively labeled distribution.

5) Mean and variance in the number of detected blobs: The number of blobs is one of our manual features which varied the most across the different sets of generated images. The distinct number of centers gives an intuitive estimation of the spacial placement diversity of the stars.

	#distinct centers	μ	σ
pos. labeled galaxies	394	4.15	1.7
PATCHES	401	4.33	2.16
FullResGAN LS BD	54	2.29	1.36
DCGAN	167	3.31	2.14
DCGAN LS BD	123	3.12	1.6
MCGAN LS BD	119	4.92	2.54
MCGAN LS BD ROT	441	5.4	2.64

Table IV: Statistics on Detected Blobs
$\sigma = 1.5$

The statistics from tables II III IV and figure 4 were all computed from sets of $n = 100$ generated images from the various methods with resolution 64x64. Images available in higher resolution have been downsampled via max pooling to 64x64. LS refers to label smoothing, BD to minibatch discrimination and ROT to augmented images via rotation.

IV. Discussion

A. Regression Task

On the test data set, fitting a scoring head onto the discriminator of the DCGAN produced a private score of 0.67. This is significantly better than the sample scoring consisting of the mean training score. This leads us to believe that the discriminant learns a meaningful representation of what a galaxy is during its adversarial training.
However, generating well chosen manual features combined with a powerful machine learning algorithm such as XGBoost is shown to be more efficient. One potential reason to this is that we had to generate very specific cosmological images, which allows for a careful expert feature selection. This approach would be intractable for a regression problem with a large number of classes.

B. Generation Task

Let's start by looking at the statistics of the positively labeled image set. We see that its mean predicted score is $\mu = 1.94$. Therefore, we can consider images that get a predicted score of 2.0 or more by our XGBoost scorer as a high scoring image. From the conducted experiments, we notice that generating such a high scoring image was not difficult; both baselines were able to achieve it. However, consistency and diversity are harder constraints to enforce.

The PATCHES method produces samples that are hard to differentiate from real galaxies by the naked eye. However, its mean predicted score of 1.37 is much lower than that of our more evolved methods. This difference can originate from the border artifacts which are noticeable on its heatmap in Figure 4 or from the difference in the number and shape distributions of the stars in table IV. However, in terms of KNN-distance and number of blobs, the PATCHES baseline has results that are closer to the labeled galaxies distribution than the results produced by our GANs. This is not very surprising as it copies patches directly from the labeled galaxies and therefore has a high probability of being statistically similar in these measurements.
Our second baseline, the FullResGAN, was able to generate plausible star placements on the 1000x1000 image plane. It struggled however to replicate artifacts or have continuous brightness patterns on the stars. It also performed quite poorly in terms of mean score in comparison to the upsampled GANs ($\mu_{predscore} = 0.96$).
The two baselines produced some bad samples as the minimum predicted scores suggests in table II.

The grid patterns on the heatmaps in figure 4 and the low number of distinct blob centers in table IV on the GANs' part suggests that they have difficulties to be as diverse as the training set. To increase this diversity, we tried introducing mini-batch discrimination and label smoothing. This seems to have helped, as we can see comparing the DCGAN to the DCGAN LS BD model: the mean distance between each sample and its neighbors increased from 1.31 to 1.48 in table III and the mean predicted score increased from 1.58 to 1.82 in table II. However, we also notice that the grid pattern is still present in the DCGAN LS BD's heatmap in figure 4 and the distinct blob number decreases in table IV.
Conditioning on a manual feature can help promote diversity, since the output of the generator is now driven by a variable we control. In our case, we did not notice any significant increase to diversity compared to the DCGAN with minibatch discrimination. It is however able to break the grid pattern of the stars that we noticed previously. Overall, it has slightly worse performances compared to the DCGAN, except for the minimal predicted score which is much higher in table II. This leads us to believe it might be more consistent.

C. Submitted Images

In the light of these experiments, we choose to submit a set of images generated by the DCGAN model and filtered by the DCGAN scoring head with a threshold $\tau = 3.0$.

V. Summary

We have discussed the challenges in the cosmology image generation and regression tasks, and presented various architectures suited for both tasks. We presented our results, discussed our metrics to evaluate our models and finally justified the choice of our final submission model.

REFERENCES

[1] I. 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*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680. [Online]. Available: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

[2] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[3] J. Kremer, K. Stensbo-Smidt, F. Gieseke, K. Pedersen, and C. Igel, "Big universe, big data: Machine learning and image analysis for astronomy," *IEEE Intelligent Systems*, vol. 32, pp. 16–22, 03 2017.

[4] N. M. Ball and R. J. Brunner, "Data mining and machine learning in astronomy," *International Journal of Modern Physics D*, vol. 19, 06 2009.

[5] E. J. Kim and R. J. Brunner, "Stargalaxy classification using deep convolutional neural networks," *Monthly Notices of the Royal Astronomical Society*, vol. 464, no. 4, pp. 4463–4475, 10 2016. [Online]. Available: https://doi.org/10.1093/mnras/stw2672

[6] S. Ravanbakhsh, F. Lanusse, R. Mandelbaum, J. Schneider, and B. Poczos, "Enabling Dark Energy Science with Deep Generative Models of Galaxy Images," 2016.

[7] K. Shmelkov, C. Schmid, and K. Alahari, "How good is my gan?"," in *Proceedings of European Conference on Computer Vision*, 2018.

[8] L. Fussell and B. Moews, "Forging new worlds: high-resolution synthetic galaxies with chained generative adversarial networks," *Monthly Notices of the Royal Astronomical Society*, vol. 485, no. 3, pp. 3203–3214, 2019.