A "2-in-1" deep neural model for generating and scoring galaxy images

Mélanie Bernhardt, Laura Manduchi and Mélanie Gaillochet

***Abstract*—Combining ideas taken from Deep Convolutional Generative Adversial Networks (DCGAN) and from Supervised Convolutional Networks for regression tasks, this paper presents a new deep neural architecture capable of simultaneously generating artificial galaxy images and scoring images in terms of similarity with real galaxy images. After detailing the architecture of our model, we present the results achieved in terms of quality of generated images as well as in terms of regression performance for the scoring problem.**

I. Introduction

GENERATIVE modeling has increasingly attracted attention in the machine learning community, as state-of-art deep neural networks require great amounts of data for their training. Indeed, as labeled data is sometimes too scarce for these models to perform well, it has become a worthwhile pursuing goal to learn how to generate artificial data that is good enough to be used later in a supervised setting. For this project, we designed a single "2-in-1" generative model that can be used for two different tasks related to cosmology images. More precisely, the first task consists in generating galaxy images that are highly similar to a reference set of true galaxy images. The second goal of this project is to use the same model in a regression context to predict similarity scores for a given set of images (in terms of similarity to the reference galaxy images set).

II. Related Work - a short review on GANs

Given the dual nature of the project - generating and grading images - the structure of our model was inspired by the Generative Adversial Network (GAN) architecture [1]. Given a set of images to reproduce, a GAN consists of two-networks:

- A generator that creates fake images that should look like the true input images.
- A discriminator that classifies images as true or fake.

These two-networks are trained simultaneously. Ideally, by the end of the training, the generator creates images that are so realistic that the discriminator is unable to make the difference between real input images and generated ones.

Practically however, GAN generated images can easily be noisy or not representative of the input images; hence an improvement of this architecture, called Deep Convolutional GANs (DCGAN), was proposed by Radford et al. in 2015 [2]. Their idea was to replace the GAN's max pooling or fully connected layers - used for downsampling and upsampling - by convolutional layers in the discriminator network and deconvolutional layers in the generator network. This resulted in a more stable architecture to train generative adversarial network and learn good representations of images.

III. Methods and models

A. Data

For this project, we were given 3 datasets:

- LABELED: 1200 images labeled as galaxy images or irrelevant images.
- SCORED: 9600 images scored between 0 and 8 in terms of their similarity to the reference galaxy images set (hold out reference set). Typically, a score of 2.61 meant that the image almost co-incided with the prototypical cosmology image, while a similarity score of 0.00244 meant that it was poorly representative of a cosmology image.
- QUERY: a set of images for which we had to predict a similarity score (regression task).

B. Architecture of the model

The goal of the project was to generate images with a high similarity score with respect to the reference set. Hence, we did not only want to reproduce the whole input dataset (containing both galaxy and irrelevant images), but we wanted to build a model that would generate only galaxy images, following the idea of conditional GANs [3]. In that setting, the main difficulty lied in preventing the generator from learning how to fool the discriminator, making the latter perform poorly in the end.

Given the nature of training the data at hand (scored images on one side and labeled images on the other) and the double task that our model had to solve, we decided not to use a classical DCGAN, but rather to build a new model that nonetheless contained both a discriminator and generator network.

Our final model consists of three components (see Fig. 1):

- the "score discriminator": a convolutional neural network (CNN) trained in a fully supervised manner on the scored dataset, using mean-absolute-deviation as a training loss (see section III-C).
- the "label discriminator": a CNN trained in a fully supervised manner on the labeled dataset, using cross entropy as a training loss (see section III-C).
- the "generator": a deconvolutional network (deCNN) which aimed to produce realistic images in terms of score and label predicted by both discriminators (see section III-D).

C. Discriminators

Both discriminators, trained in a fully supervised manner, are deep convolutional neural networks that receive as input

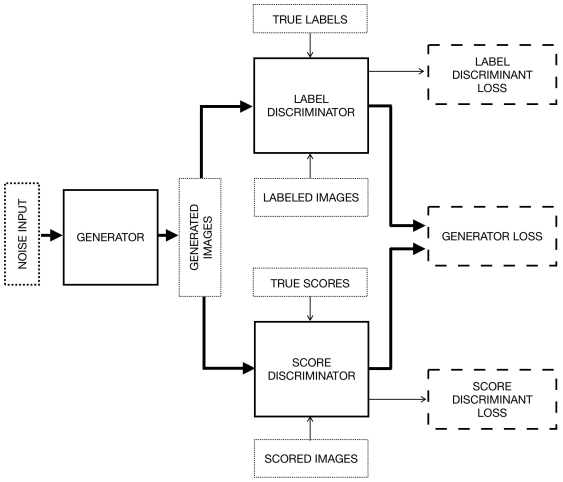


Fig. 1: Global architecture of our model. Thick arrows represent the data flow for the generator training whereas thin arrows represent the flow for the discriminators.

arrays of dimension [1000, 1000]. The regressor ("score discriminator") and classifier ("label discriminator") differ only in their final output layer. The detailed architecture is presented in Fig. 2.

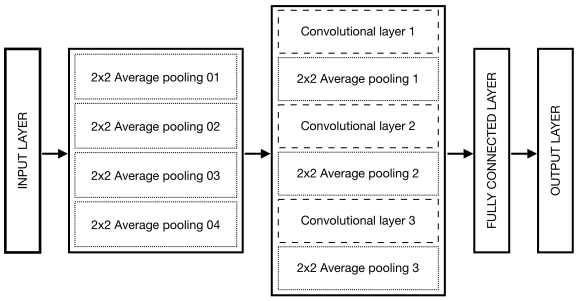


Fig. 2: Architecture of discriminators.

Since the input image dimension was too big to encode meaningful information, we first used four average pooling layers with a window and stride of size 2×2 to reduce the dimensions. We then obtained a vector of dimension [63, 63], to which a series of four convolutional layers with their respective pooling layers, were applied.
We used 32 filters for the first convolutional layer and 8 filters for the second and third, each filter having size 3×3. The fully connected layer contains 28 units and its activation function is ReLu. For the regressor, the final output layer is a dense layer with one unit (no activation because the output is the image score), and for the classifier, it is a two-units dense layer with a ReLu activation (the output units here are the logits used for the cross-entropy loss).

D. Generator

The general architecture of our model's generator, depicted in Fig. 3, follows the guidelines proposed by Radford et al. [2]. It projects a 100 dimensional input taken from a uniform distribution to a small spatial extent representation with a feature map of dimension [25, 25, 124], through a fully connected layer. A series of four transposed convolutions then converts this high level representation into a 1000×1000 grayscale image (pixel value between 0 and 255).

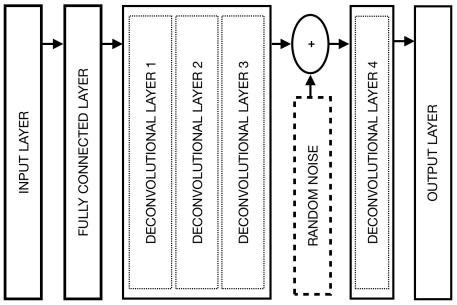


Fig. 3: Architecture of the generator.

We encountered many difficulties while attempting to tune the parameters of the cascade of transposed convolution, however, after a extensive model exploration, we identified a set of parameters that outputs realistic galaxy images.

We gradually decreased the number of filters in the transposed convolutions from 100 to 1, with intermediate layers having 50 and 10 filters. In practice, a lower number of filters has proved to output images poorly representative of the true images since recurrent patterns (like grey strides or patches) were always present. Hence adding more filters helped capture additional information which eventually eliminated the recurrent pattern and allowed us to generate more realistic galaxy images. While the strides size was constantly set to 2 for all layers, the kernel dimension, initially at [5, 5] was changed to [2, 2] for the last two layers. Between the third and the fourth layers, we added a noise signal taken from a uniform random distribution because it allowed us to increase the variance (and thus the diversity) of the output images. Indeed, without this added noise, the images were almost identical to each other. Finally the ReLU activation was used for all layers with the exception of the output layer which made use of the Tanh function.

Our model's loss function took into account both the output of the "label discriminator" and that of the "score discriminator". In the first case, we used the cross entropy loss between the real label y_i, valued at 0 or 1, and the softmax probability, p_i, produced by label discriminator. On the other hand, to account for the score discriminator, we used the difference between a target score of 3 and the predicted score s_j. The formula used was then:

$$-\sum_{i=1}^{M} y_i log(p_i) + \frac{\sum_{j=1}^{N} 3 - s_j}{N}$$

Our target score was set to 3, because images scored 2.6 were said to be nearly perfectly co-inciding with true galaxy images. Accounting for prediction errors in the score of generated images, we set the target score just a little higher

than this. On the other hand, we did not choose a too high target score as a regressor is stable only on its training domain and very few images of the training set had a score higher than 3.5, meaning that our discriminator might be unstable for higher scores.

E. Training procedure

The three components of our model - the two discriminators and the generator - were first trained simultaneously, each using a separate Adam Optimizer for each variable scope. However, because the generator trained faster than the discriminators, we faced severe overfitting by the time the loss of both discriminators had converged. As a result, the generator nearly always produced the same image. To solve this problem, we first trained the discriminators for 20.000 steps (tensorflow defines one step as one optimization step, since we optimize both discriminant this means that we actually trained each of them for 10.000 steps) before allowing the generator to train jointly with both discriminators for only 2.000 steps more.

The model was implemented with Python and Tensorflow[1]. We adapted the main steps of R. Varma's tutorial [4] describing the implementation of a GAN in Tensorflow to our model.

F. Using the regression discriminator as similarity score predictor

In order to be able to score the images of the QUERY dataset, we extracted the trained score discriminator of our model after training completion. Hence, our model can be used as well as for the generation task as for the regression task. Indeed, once we have restored the model, it is sufficient to input the query images to the network subgraph corresponding to the score discriminant to get the predicted similarity score.

IV. Results

For this project we were required to generate a diverse set of galaxy images with a high similarity score with respect to the images in the reference dataset held by the instructors of the course. We were also required to assign a similarity score to each image in the query dataset, this part of the project being judged via Kaggle[2] using mean absolute error (MAE) as evaluation metric.

A. Prediction of similarity scores

To assess the quality of the score predictions of our 2-in-1 model, we implemented two baseline models: a Ridge regression and a random forest regressor. The training was performed on 99% of the dataset, and the validation, on the remaining 1%. Both training and validation set were identical to those of used for the final 2-in-1 model. For both baseline models, a simple preprocessing with a 10 bins histogram-based feature extraction was used[3].

With a penalty parameter $\alpha = 1000$ (fine-tuned with Scikit-learn's Gridsearch cross validation function) Ridge regression gave a mean absolute error (MAE) of 0.752 on the validation set. For the RandomForest, 100 decision trees were used (after parameter tuning with GridSearch cross-validation) with default parameters for controlling the size of the trees. Predictions on the validation set gave a MAE of 0.312.

The following table summarizes the results obtained by our different models.

Model	Validation Set	"Public" Test Set	"Private" Test Set
2-in-1 model	0.55622	0.61219	0.63589
Random Forest - 100 trees	0.31236	0.31269	0.33127
Ridge Regression - $\alpha = 1000$	0.75246	0.74620	0.76589

TABLE I: Mean absolute error (MAE) obtained on the validation set (96 images) and on both Kaggle test sets.

From Table I we see that our model beats the Ridge Regression baseline but not the RandomForest Regression baseline. The RandomForest is a very strong baseline as it outperforms half of the submissions from the final Kaggle leaderboard. Indeed, it seems natural that a model fine-tuned and trained only for the purpose of answering the first task (scoring regression) would perform better than a model created to get good (albeit not excellent) results *simultaneously* for both tasks.

Given the public test set results (available before the end of the competition), we knew that our baseline method would have given us a better ranking on the scoring competition. However, we decided to use the results given by our 2-in-1 model for our final submission, even though this meant lowering our competition rank, because we believed that the challenging part of this project lied in creating a single model that could solve both tasks. The 2-in-1 model is thus a compromise in terms of performance for both tasks.

B. Generating space images

We used our generator to produce images similar to galaxy images. More precisely, we saved only the generated images whose predicted score was above 3.0 (as 2.61 was supposed to be an excellent score according to the project guidelines). After evaluating our discriminant on the validation and test sets, we chose a high score threshold in order to take the scoring error of the discriminator into account. With a threshold of 3, we expect that the real mean similarity score of the generated images lies between 2.0 and 2.5.

The SCORED training set contained roughly 20% of irrelevant images and 80% of galaxy images (as the distribution between 'good' and 'bad' images was assumed to be equivalent to the distribution observed in the LABELED dataset). The 20% percentile of the score distribution in the SCORED dataset was 0.74. Hence, we could infer that the

[1]The code can be found under *https://github.com/lauramanduchi/CIL2018*

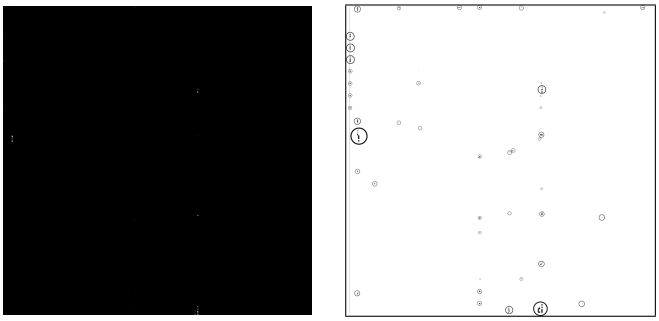
[2]The competition can be found under *https://www.kaggle.com/c/cil-cosmology-2018*.

[3]This feature preprocessing was presented during one exercise session of the Computational Intelligence Lab class.

'good' images of the SCORED set were the subset of images for which the score lied above this 0.74 threshold. Thus, we can get an estimation of the mean score of the 'good' galaxy images in the training set by calculating the mean score on this subset of training images. The mean score obtained was 2.02 and the standard deviation of scores was 0.89.

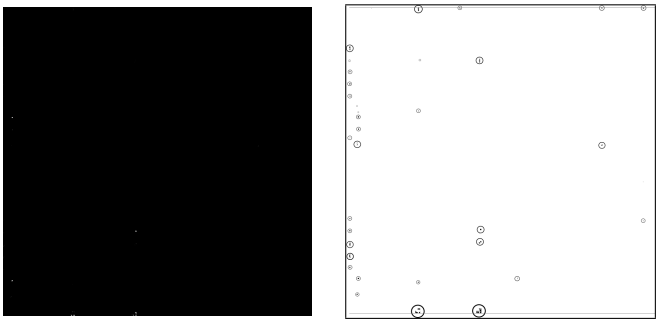
As our images will be evaluated in terms of mean similarity score with respect to the galaxy image set, we double-checked the scores of the generated images to submit, by predicting their similarity score using our strong RandomForest baseline. The mean score of the generated images predicted by this baseline was 2.30 and the scores standard deviation was 0.62. Thus, our generated set of images has a predicted mean score which is better than the mean score of the good training images even though there are bit less diverse in terms of scores (lower standard deviation).

Fig. 4 and Fig. 5 show two examples of generated images. We can observed small white dots similar to the stars observed on the training images. Moreover, we can see that their location changes from image to image.



(a) Raw image (b) Adjusted image

Fig. 4: Generated image with score = 3.003 predicted by the RandomForest regressor model. The adjustment of the image on the right was done with 'Auto levels' a colour adjustment filter. The coloured pixels were then circled for clarity.



(a) Raw image (b) Adjusted image

Fig. 5: Another generated image with score = 3.093 predicted by the RandomForest regressor model.

One of the main problems that we encountered in the generation of images was the dilemma between quality and diversity. We had to choose between creating one or two realistic images, or creating a large number of different image with lower realism. We opted for the latter because we were asked to provide a set of diverse images. To do so, we added a random noise layer in the generator.

V. Discussion

As we already discussed in section IV-A, our model is a "compromise" model. On one hand, one drawback of our model, as a compromise model, is that it does not give outstanding results for the discriminator part, only beating one baseline out of two. Further work on this model would involve trying to train our discriminators for a very long time (by tripling the number of epochs compared to our current final model) before starting to train the generator. We did not choose to follow this idea for the current project as computation time was taken into account for the grading. Finally, the triple nature of this model (label discriminator, score discriminator and generator networks) makes training and fine-tuning very subtle as changing the training or architecture parameters of one of the networks impacts the training behavior of the other two networks.

On the other hand, the force of our model lies in the fact that it is only necessary to train one single model to solve two different tasks simultaneously. This shows how the idea behind GANs can be reinterpreted to build a new class of "2-in-1 models". Moreover, to train our model, we were able to use a combination of two different training sets: one containing labeled images and one containing scored images, taking full advantage of the provided training data (instead of just using the scored training data). Finally, we generated images that were quite diverse and that had a very good predicted similarity score - the prediction being double checked after generation using our best baseline.

VI. Summary

In this project, we came up with a novel architecture to build a "2-in-1 model" capable of solving two different tasks at the same time. As a compromise model, the score prediction quality was acceptable and, even if not outstanding, it enabled generation of galaxy images with a very good predicted similarity score to the galaxy images reference. Finally, the main outcome of this project was to show that the idea behind GANs can be re-interpretated to build "2-in-1 models" capable of solving two different tasks. We believe that this idea can be re-used to solve other kind of classification and generation tasks simultaneously.

References

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair†, A. Courville, and Y. Bengio, "Generative adversarial nets," *arXiv:1406.2661 [stat.ML]*, 2014.

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

[3] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv:1411.1784 [cs.LG]*, 2014.

[4] Rohan Varma, "Generative Adversial Network tutorial," online, https://github.com/uclaacmai/Generative-Adversarial-Network-Tutorial/blob/master/Generative%20Adversarial%20Networks%20Tutorial.ipynb.