https://github.com/MrPacha/project-dl

INFO8010: Project report

Clément Laisney,[1] **Quentin Lalou**,[2] and **Hugo Marmet**[3]

[1] *Clément.Laisney@student.uliege.be (s214186)*
[2] *Quentin.Lalou@student.uliege.be (s214962)*
[3] *Hugo.Marmet@student.uliege.be (s212673)*

I. INTRODUCTION

Machine learning (ML) methods can find applications in a very large diversity of fields. Indeed, they can be found in medication for drug discovery, autonomous cars, image recognition, or image compression for instance. It is the very good capacity of those systems to recognize patterns and structures in data that makes them able to efficiently find behaviors and rules that they may follow or replicate to perform a task. If the said task is repetitive, they are even easier to adapt and apply. Such systems are thus used in a variety of domains, among which astrophysics. In astronomy and astrophysics, we are dealing with humongous amounts of objects (we would study the whole Universe if we could! And it is quite large) and, as a consequence, data. One problem in astrophysics is galaxy characterization. Indeed, galaxies can take several shapes (as described by the Hubble sequence) and this may give insights into their life cycle. This is why they are of interest, and their classification depending on their type is an interesting task to do. Since the number of galaxies is extremely high, it is a laborious and expensive task to do. The "GalaxyZoo" dataset is made of images of galaxies that have been labeled by Internet voluntary users, who classified them depending on their shape so that they can be characterized. The "GalaxyZoo2" dataset we use in this work is an extension of the GalaxyZoo one, with 304 122 galaxies as described in [1]. If here we won't be interested in the labeling of the said galaxies (since we don't look for a classification problem), we will still use them as they should be a nice, large dataset of images to use as examples for our generations. It is important to note that those images are in colors (3 channels), and have a size of 424 x 424 pixels (much bigger than the images of the MNIST dataset for instance, which have a size of 28 x 28 pixels) which may make their generation more difficult and computationally expensive. You can find all the code on this GitHub[2] repository.

II. RELATED WORK

The first related work we will present is the one that seems the most evident: "Deep generative models for galaxy image simulations" by F.Lanusse ([4]) from 2021.

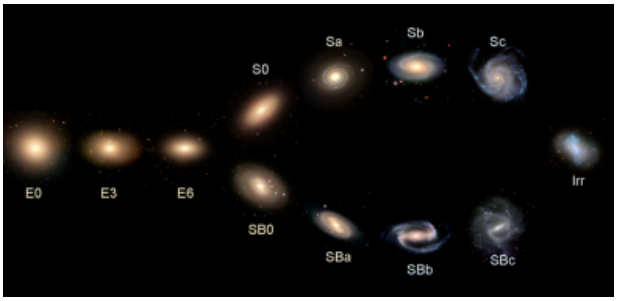

FIG. 1. The Hubble sequence. Picture taken from [3], in a non-related work

This article is directly linked to our own work since it uses a VAE model to generate galaxy images. The authors work on the COSMOS dataset of galaxy images and use a VAE with a normalizing flow. In fact, they begin by building a VAE composed of ResNet blocks and with a 16 dimensions latent space, a learning rate of 1.10^{-3}, a batch size of 64, and using an Adafactor optimizer. There are several interesting "tricks" they describe. First, the use of a "free bits" process allows for reducing the impact of the Kullback-Leiber (KL) divergence regularizing and making the model tend towards very smooth results (that are over-regularized). This *free bits* process prevents the penalizing process of the KL divergence from applying to a part of the information. To do so, the authors use a factor diminishing the influence of the KL divergence in the full loss term. Another interesting thing is that they use a supplementary network (another neural network) called a normalizing flow, which allows for density estimation in the latent space and learning the posterior of the data, for a certain VAE. In practice, this normalizing flow used is a masked auto-regressive flow (MAF) which, to be efficient, must be applied as several successive layers. Eight of them are used in this work, and the normalizing flow is trained itself with an ADAM optimizer. We may note another particularity from this work which is the use of a total variation term in the loss, penalizing irrelevant high-frequency artifacts the network may generate. All those tools and this work are useful for us, as they give us examples and ideas of tricks that we could ourselves use. Furthermore, the performances of the VAE appear to be better than the ones of parametric fits (the most used

method when not considering image generation), meaning that our goal of generating galaxy images to increase samples' size is relevant. Another work we studied is the one from [5] in which the authors use a Vector Quantized VAE (VQ-VAE) for image generation. Those images in question aren't galaxies, but the work is still of interest to us as the models used are VAEs as we intend to. The particularity of the VQ-VAE is that, contrary to a VAE in which the decoder maps from distributions in the latent space, the output of the encoder is quantized by its similarity to a set of vectors. Two supplementary (compared to the usual ELBO loss, which will be described later in this report) loss terms are brought so that this quantization implies a loss as well. One important aspect of this work is that several stages of quantized vectors (a *top* and a *bottom* latent code) have been used so that the global and the local information are separated. Figure 2 presents the architecture of this VQ-VAE model, with the *top* and *bottom* latent codes. It is this hierarchy in latent codes that allows for better representing high-resolution details (since a part of the model will be used for this purpose alone). For better performances, the authors also use PixelCNN, a neural network, for learning a prior over the latent codes (in the latent space thus) and allowing better and easier decoding from the latent space. They also improve this prior network by using self-attention layers. An important statement from this work (to keep in mind for our own) is that, very often, in image generation, a trade-off exists between the quality (the fidelity to the input sample, the ability of a generated image to "look real") and the diversity of the sample (we want to generate new various images, not to have almost always the same image). Another interesting thing we could take from this article is the performance estimation, made using the Inception Score (IS) and the Fréchet Inception Distance (FID) which are two metrics used for GANs. The IS takes into account the diversity of the generated images and the fact that they look like real samples. On its side, the FID will rather focus on the quality of the samples generated by comparing them with real images. Despite those metrics, we must note that the authors stress the high importance of visual inspection to determine the goodness of the results.

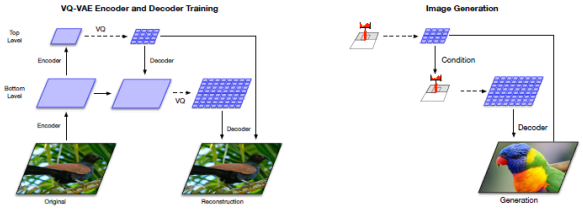

FIG. 2. Architecture of the VQ-VAE model from [5]. The figure is taken from this article.

The final article we will briefly describe in this related work section is [6]. This directly tackles the problem we chose to work on, using also the GalaxyZoo dataset (not the GalaxyZoo2 like us, but its predecessor). In this work, the authors try to find a way to assess the performances of the generative models. Indeed, a classifier model (for instance) can separate its dataset into a training set, a test set, and a validation set so that it can assess its performance. But, since our goal is to create totally new images, it is hard to determine their goodness as they can't really be compared to samples from our test set. [5] already introduces the IS and the FID, but [6] rather presents another metric specific to a galaxy dataset. They even state that "there are currently no metrics that are capable to assess in a qualitative way the resemblance of the target distribution by generated data". The first interesting manipulation from this article for our work is the pre-processing of the images, which are cropped from their 424x424 size rather towards a 207x207 size (keeping the central part), allowing for less voluminous objects (so a lower computational cost and a smaller distribution from which the VAE must learn) and for individual galaxies (as several of them may be present on the images). Also, they perform data augmentation by randomly rotating, flipping, and translating the images, as those modifications could be present in real galaxy images. A re-scaling of the images pixels' values from [0,255] to [-1,1] is also performed to increase the model's performance, but it appears that `PyTorch` automatically normalizes the input images to [0,1] when converting them into tensors (so we won't need to apply a supplementary normalization). This work is very strongly focused on the determination of a metric that would be efficient for image generation. This is why they chose to use the GalaxyZoo dataset, which is labeled and allows for estimating the diversity of generated images by classifying them and counting the distribution in the different classes. To do so, they also introduce a metric based on a k-means clustering algorithm that will estimate this diversity. This is a metric assessed as very efficient, but we won't use it here as we are rather interested in the image generation process than in the classification aspect of the galaxies. Still, we note that the article once again cites the IS and FID criterion as good for assessing the overall quality of the generated images and that other metrics like the Gini criterion can be used for assessing the physical sense of the results. Also, several models are compared here (two sophisticated GANs, a simple VAE, and a mode collapse model which always produces the same detailed image) and the simple VAE one appears worse than the other ones in terms of image quality. Also, it has difficulties in realizing a diverse dataset. This isn't very good news for us at first sight, but it is still good to know that our models may not be the most adapted ones. In fact, [6] states that the models that would allow the best performances are diffusion models, but they are out of the scope of our work and on another level. Indeed, such models make use of a hierarchy of VAEs and achieve state-of-the-art performances in image generation (e.g. like MidJourney of Dall-E).

III. METHODS

In this project's scope, we considered several possible methods to generate the desired images. Initially, we explored the use of a Generative Adversarial Network (GAN) which, by the competition of two networks, allows for the generation and the evaluation of images. A generative model produces outputs (galaxy images in this project) the most realistic possible, this level of realism being assessed by a discriminant network. The latter discerns the real inputs from the outputs the generative model produced. When the discriminant model can no longer differentiate them, it means that the generator is good enough. The problem with such GAN is that the training must be performed for both the generator and the discriminant network, which isn't easy. There are good results of such systems presented in the literature, but they imply very good and finely-tuned training as it isn't very stable because of the adversarial aspect of the model, and the difficulty to reconstruct complex distributions as indicated in [7]. Another possibility to generate the galaxy images is the use of a Variational Auto-Encoder (VAE). Composed of two different networks like the GAN, a VAE presents both an encoder and a decoder (unlike the GAN). The purpose of the former is to extract the main features of the input that the latter will use to reconstruct the said input by generating its own output. The particularity of the VAE (compared to a classical auto-encoder) is that the encoder doesn't generate a vector of features, but rather a distribution into a latent space, from which the decoder will generate. This distribution will allow for a generalization of the generation process and regularize, in a way, the output of the VAE. The main problem of a VAE will arise from backpropagation (necessary for training). Indeed, the latent space will pose the difficulty of being composed of learned parameters only, as they are the outputs of the encoder. This is where the "reparametrization trick" is useful. This trick will input a known distribution (usually a Gaussian distribution with a mean of 0 and a standard deviation of 1) and multiply the parameters of the latent space. This way, the backpropagation isn't to be made on an unknown parameter, since only this reparametrization parameter will be "unknown" (we know which kind of distribution it is) and isn't one on which we will need to back-propagate. this reparametrization trick thus allows for training the VAE and adapting its weights. This way, the VAE is made usable for us.
On a more general note, we recall that the metrics for performance evaluation described previously like the IS and the FID are defined for GANs and, more importantly, that they are estimated using a pre-trained model (the Inception v3 model), which has been trained on very large datasets like ImageNet and is thus unlikely to be adapted to galaxies (on which it hasn't been trained). Since we don't look for classifying the galaxies in the GalaxyZoo2 dataset either, we won't put much emphasis on the different galactic parameters that could be used as metrics of the goodness of our generative models. We will rather have a visual estimation and study the different losses we can extract from them.

A. GAN

Despite the considerations of [7], we decided to try to use a GAN. Nonetheless, our results with this method are quite disappointing.

Contrary to the VAE, the GAN doesn't use a well-defined loss function. To do so, 2 networks are trained simultaneously. The first one is called the discriminator and allows for recognizing if an image is considered a galaxy or not. Its purpose will be to train and generate a loss for the second network, the generator. With random numbers as inputs, it creates a new image with the pixel size required.

The first major issue was the pixel size. The GAN has a very big computational cost compared to the VAE. To do so, the images were resized from 3x424x424 to 3x28x28.

As expected, the training was a real pain. The main problem came from the discriminator. Indeed, to train it, we inject a series of images from the database, with a loss fixed to 0 + a certain noise. Also, images with pure noise were generated with a loss fixed to 1 - a certain noise. Despite a very long training and a discriminator that converge as shown in the figure 3, the never succeeded to produce any good image.

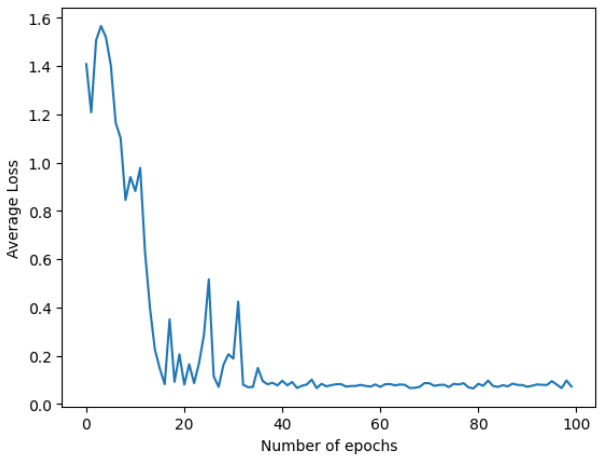

FIG. 3. Average loss of the discriminator over 100 epochs.

Here on the figure 4, we see sample produced by the GAN generator. Every images are similar, with a black background and some pixels lighter. It seems that the neural network understand the black background coming from the space, but is not able to produce any galaxy. We didn't see any evolution in the state of art even with

a bigger network and more complexity. Since the results are disappointing, we won't discuss any further this method.

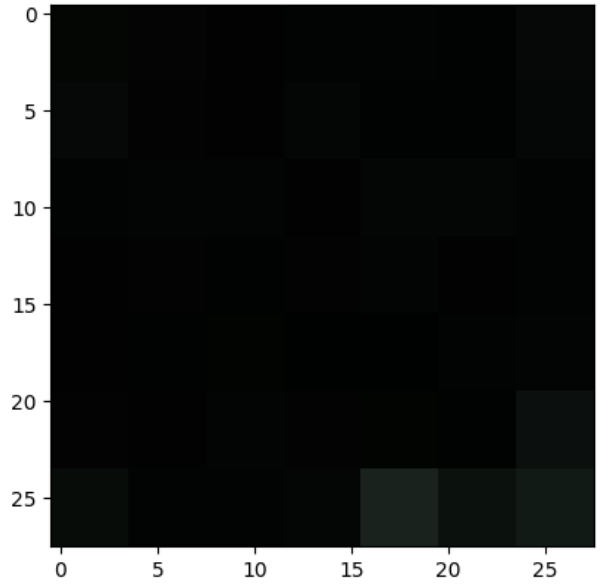

FIG. 4. Image produced by the GAN generator.

B. VAE

1. MLP

Our first tests on the VAE consisted in using a simple Multi-Layer Perceptron (MLP). Such a model is composed of fully connected layers only, denominated "Linear" in `PyTorch`. To use such a model, it is necessary to define how it works. The idea here is to feed the encoder the images, make them go through several linear layers with a Rectified Linear Unit (ReLU) activation function, infer the two parameters of the latent space's distribution (its mean and its standard deviation) by linear layers as well, and finally use other fully-connected layers (as many as used for the encoder) to decode the data after the reparametrization trick. As usual from a VAE, the loss is computed by the Evidence Lower Bound (ELBO) loss, composed of both a reconstruction loss (to compare the reconstructed images and the input ones) and of the KL divergence which works as a regularization loss (by trying to make the output distribution similar to the latent space one). Usual optimization and backward propagation methods allow for the learning process by training. To determine the performance of the MLP model, we tried to modify several of its parameters. First, we chose to use an MLP model composed of 2 layers for both the encoder and the decoder with a size of 40 hidden dimensions. The first layer gathers the input image, brings it to this 40 size, and goes through a ReLU activation function. The output is fed to another layer that outputs at a size of 10 for the latent space. This space has 2 dimensions (corresponding thus to 2 distributions the decoder network can pick from), and the mapping of the latent space feeds the decoder with vectors of size 10. This latter network itself uses 40 hidden dimensions, and output with the initial input shape. Table I presents the architecture of this first version of the MLP. To ease things, we will designate it by "MLP VAE 1". This simple design doesn't consider any fancy additions that could make its results better, and that explains its bad results visible in the left part of figure 6 (see section IV A for the results of the models). Based on this, it was evident that there was a need for improvements, so we tried "tricks" to improve our network. We began by modifying the loss function. Indeed, the loss we use is the ELBO loss, defined as such:

α	1.10^{-6}
Input sample	image size
Linear	dimension = input $\rightarrow$ 10
ReLU	
Linear	dimension = 10 $\Longrightarrow$ 5
Linear (mu)	dimension = 5 $\Longrightarrow$ 2
Linear (logvar)	dimension = 5 $\Longrightarrow$ 2
Linear (latent mapping)	dimension = 2 $\Longrightarrow$ 5
Linear	dimension = 5 $\Longrightarrow$ 10
ReLU	
Linear	dimension = 10 $\Longrightarrow$ input
Sigmoid	

TABLE I. Architecture of MLP VAE 2

$$ELBO = \mathbb{E}_{q_\nu(z|x)}[logp(x|z)] - KL(q_\nu(z|x)||p(z))$$

where we compare thus the KL divergence and a reconstruction loss (which is often one of the usual we know, like the cross entropy or the L2 loss). But it is known (we can cite [4]) that VAEs have a tendency to be too smooth as they accord too much importance to the minimization of the KL divergence, which will regularize the system and make its output as likely as possible to the original Gaussian distribution. This is why we induce a factor in the final loss so that it becomes:

$$ELBO = \mathbb{E}_{q_\nu(z|x)}[logp(x|z)] - \alpha * KL(q_\nu(z|x)||p(z))$$

This way, we can control the importance of the KL divergence regularization term, allowing the generated samples to be more diverse.

We then applied a dropout mechanism. Just to remind the reader, the dropout consists in applying, to each individual neuron, a probability not to be used by the network. This dropout allows for reducing the individuality of the neurons as they are all susceptible to being dropped during each iteration, and thus the system will learn without them (it avoids having one specific neuron as very important, for instance). Also, we made the model deeper, with a higher dimension latent space and

α	1.10^{-6}
Dropout	Probability = 0.2
Input sample	image size
Linear	dimension = input $\rightarrow$ 40
ReLU	
Linear	dimension = 40 $\Longrightarrow$ 20
Linear (mu)	dimension = 20 $\Longrightarrow$ 10
Linear (logvar)	dimension = 20 $\Longrightarrow$ 10
Linear (latent mapping)	dimension = 10 $\Longrightarrow$ 20
Linear	dimension = 20 $\Longrightarrow$ 40
ReLU	
Linear	dimension = 40 $\Longrightarrow$ input
Sigmoid	

TABLE II. Architecture of MLP VAE 2

more hidden dimensions. Table II presents the architecture of the model for this second test, which we will name "MLP VAE 2".

Keeping the architecture of "MLP VAE 2", we tried several values of the batch size. The effects are described in the section IV A, along with the other results. We also decided to test the effect of the dropout mechanism by trying different values of the dropout probability. The results, once again, are described in section IV A.

2. Convolution

Our first attempt with an MLP-based VAE gave us many hints for our next model, and more precisely about the `batch_size`, the value of α or even for `latent_space` dimensions. Since the beginning, we knew that the MLP VAE would not give any striking results, but it has the advantage of being a low computationally cost model, on which we can perform fast and multiple tests.

However, we knew since the beginning that a Convolutional model would give better results since convolutional layers are able to feature details. To be consistent, we still made the same tests as we did with the MLP model and we had the same behavior (the results are given in sect IV B).

In this work, we considered 2 architectures. The first one is designed with 3 successive convolutional layers. The second one is designed with 2 successive blocks of `convolutional`, `Maxpooling`, and `ReLU` layers as indicated in table III. Quickly, we focused on the second architecture because of the highly demanding computational resources as well as the long computation time for the first one. We will therefore discuss the results on the dedicated part. As we did in the MLP model, we added a dropout layer with a dropout probability of 0.2.

3. Data augmentation

To reinforce the network and have more diversity in the dataset, we used transformations on the data loader. As for algorithms used for image classification, we add random horizontal and vertical flips to the dataset. Doing so allows for more diverse samples. This data augmentation has been tested for both the MLP VAE and the convolutional VAE so that we can see the effect it has no matter the type of model. The transformations we perform must be thought about before application. Indeed, it would be irrelevant to apply transformations that would make the dataset totally different from its initial goal. We consider here the generation of galaxy images, by the use of a dataset made of galaxy images. If we study galaxies, we may augment the data but must keep the samples as galaxies. This means that only certain transformations make sense. For instance, it wouldn't seem judicious to randomly modify the colors of the images. Indeed, galaxies very often have a certain colorimetry (for instance, it is very likely to observe reddish or blueish images), and some colors wouldn't make sense. If an images dataset like ImageNet presents a very large variety of colors, it isn't the case for GalaxyZoo2 and it would be weird to randomly have some flashy green, pink, purple, or yellow pixels on an image that would otherwise very likely only present gloomy red, blue, or orange colors only. But, aside from the colors, other transformations can be applied and are useful. If we follow the data augmentation procedure described in [6], we can apply random rotations, translations, and flips (horizontal or vertical). This seems relevant, as we don't know what the respective orientations of a galaxy and of the imaging instrument would be in real conditions. This is why we apply those transformations to the data in order to augment its effective size. Also, to make the computation faster and easier for the model, we chose to follow further the process described in [6] and crop the images from a 424x424 size to a 207x207 size. The reason for that is the one invoked in the previously cited article: certain images present several galaxies at once, and this cropping should allow for keeping only the central one in most cases. Also, those 207x207 cropped images are resized (by a regular down-scaling) to 64x64 pixels.

4. Data reduction

Another possibility of regularization is the opposite of data augmentation: data reduction. If data augmentation tries to make the training sample the most diverse possible so that the final model learns from a very large distribution, we may also think of data reduction that could allow for regularizing by keeping the model from learning "too much". The idea is to restrain the network by keeping a part of the dataset from being viewed. Again, to regularize the models, we add a dropout at the entry network. Even though, looking at the subsequently obtained results doesn't seem to show any difference due to data reduction.

IV. RESULTS

A. MLP

The results of the different tests we performed on the MLP are presented in this section. Following the advice from [5], we put emphasis on showing the results as images to have a visual estimation of the model's performances. Still, we show the associated training and test losses, which should allow for a quantitative analysis of the results (while the images give the qualitative one), despite not using a rigorous and robust metric like the ones presented by [6]. Figure 6 presents the loss and images generated by our model designed "MLP VAE 1" previously. We can observe that the training loss quickly decreases until reaching a value very close to 0, after approximately 40 epochs. From our tests, this model converges very quickly (only 19 iterations) but generates (as we can see in figure 6) very similar images (a very small diversity in the generated sample) as the whole batch looks exactly identical. It is clear that this model isn't powerful enough and needs to be improved.

This is what we did with "MLP VAE 2", of which the results are presented in figure 7 for the training loss and one batch of generated images. It is very clear that the loss decreases much faster than before, in one epoch only. Also, we may note that the batch of generated images is more diverse than the one from figure 6, which is an improvement (the model outputs less smooth results). This may be due to the presence of dropout itself, which has the effect of regularizing. At first, we may think that regularization by dropout would make less diverse samples as the model as the learning of the network is made "harder". But, also, regularization avoids the model from learning one single image for the whole training set, and so to always reproduce the same one no matter the test set input sample. This is an analogy to the mode collapse that can appear in GANs, as described in [6].

Improving the results implied testing different parameters to observe their impacts on the model's efficiency. This is why we tried to modify the dropout probability first, and then the batch size. Also, we performed data augmentation as described in section III B 3, trying to make the model more performant by having better (more diverse) training. The results of the dropout tests are visible in figure 8 for the evolution of the training loss, and figure 9 for the visual confirmation. The former shows that the loss variation differs a bit depending on the dropout probability, but always converges very quickly. As expected, the convergence is the fastest for the highest value of the probability as, in this case, each neuron is more likely to be dropped and the regularization is more prominent. As a result, the model is expected to quickly find itself in a situation where it can't adapt its parameters anymore to match the data distribution because it doesn't have enough parameters to adapt. As visible in figure 9, the effect of the dropout probability is very small. If the highest probability gives more various images, this diversity is only gained by producing worse images. Considering the very small effect of the dropout, it does not appear relevant to show the test loss evolution (which stays almost constant and has a very similar value for all the values of p, except for p=0.2). This analysis proves that the dropout doesn't have much of an impact on the MLP VAE model, so we chose to keep the one that seemed to have a certain effect without making the results worse: a dropout probability p=0,2.

We also tried to modify the number of elements in the batch, to observe the influence. The corresponding training losses are visible in figure 10. It is visible that the batch size has an impact on the loss, but didn't appear to have a significant one on the test loss and on the generated images. Still, a few variations were visible, so we chose to fix the batch size to 32 images. Additionally, we tried to modify the size of the latent space of the VAE. This could have a significant influence since the dimensionality of this space defines the number of distributions from which the decoder can sample to reconstruct the input images. Theoretically, it should define the ability of the model to reproduce the different main features of the images. Figure 11 presents the evolution of the training loss along the epochs for different sizes of the latent space. It appears that, unfortunately, the impact wasn't as significant as expected. Generated samples were very similar, and the test loss was almost constant no matter the size of the latent space. Finally, we decided to apply a latent space with 12 dimensions, which seemed reasonable for good results. We note that, for visibility in this report, the generated images and the test losses aren't shown for the tests on the bath size and on the dimensions of the latent space.

The next step of our improvement of the MLP model was the data augmentation, of which we show the results after. Following the procedure described in section III B 3, we increased the effective size of our training set by performing various transformations on the original dataset. Figures 12, and 13, present the results of the testing (the test loss evolution along the epochs, into one epoch, respectively). We can compare figures 13 and figure 14, the latter presenting the evolution of the test loss as a function of the iterations, for the same model but without data augmentation. It is visible here that, in both cases, the test loss stays approximately constant. But, while it orbits around a value of 5.7 for the data-augmented model, it is closer to a value of 8.7 when not augmented. This less important loss shows that the data augmentation allowed for a better model (as it has better performances on the test model, for the same loss metric)[8].

B. Convolution

As the MLP model tests suggested, a `batch_size` of 32 or 64 ensures a better-looking result, especially with a smooth delimitation between the bulge and the arms of

galaxies. However, a 64 `batch_size` introduce noise and grain and seems to have difficulties with colors. In the following, we will use a `batch_size` of 32.

α	1.10^{-6}
Dropout	Probability = 0.2
Convolutional	dimension = 3 → 16 kernel size = 5
ReLU	
Max pooling	kernel size = 2
Convolutional	dimension = 16 → 32 kernel size = 5
Max pooling	kernel size = 2
ReLU	
Flatten	
Linear (mu)	dimension = 32*13*13 ⟹ 12
Linear (logvar)	dimension = 32*13*13 ⟹ 12
Linear (latent mapping)	dimension = 12 ⟹ 32*13*13
Unflatten	Dimensions = (-1,32,13,13)
ReLU	
Upsample	scale factor = 2
Transpose convolutional	dimension = 32 → 16 kernel size = 5
ReLU	
Upsample	scale factor = 2
Transpose convolutional	dimension = 16 → 3 kernel size = 5
Sigmoid	

TABLE III. Architecture of the CNN VAE

The next test focus on the `latent_space` dimensionality. With `latent_space`=4, we start to observe ellipticity, at higher dimensions the diversity of ellipticity, inclination, and other galactic parameters. This lets us suppose that high dimensionality will let us reach more details. But again from a computational point of view, we choose a `latent_dim`=12.
Again, we tested the effect of the parameter α in the ELBO loss. It's clear that the best results are given with $\alpha = 1e-6$. With $\alpha = 1e-7$, we get a noisier result, particularly on what we think are binary galaxies images. In the following, we choose $\alpha = 1e-6$
Next, we tested the impact of the reconstruction loss and we thus tested the ELBO loss with `Binary Cross Entropy`, `L1 loss`, and `Mean Square Error loss`. We noticed similar results from those 3 loss functions and it's hard to choose one but we keep the `Binary Cross Entropy` one.
Finally, we studied the impact of the dropout probability, and as the MLP model tests suggested; at high dropout probability, we have an apparition of grain which results in a degraded quality.
In general, the ELBO loss function converges quickly toward a value around 0.20 with a faster decrease depending on parameters, but this value is reached between 3 to 5 epochs.
We also tried to have a view of the latent space and the distribution of useful indices in the latent space. We limited ourselves to the first 2 dimensions of the latent space and scattered the points with a colormap depending on the value of the index. To do so, we used a specialized Python library to determine indexes such as `ellipticity`,`effective radius`, and `concentration` which are common indexes in astrophysics. It's hard to distinguish clusters corresponding to different types of galaxies but further studies with machine learning models on the latent space could help to discriminate clusters. A conditioning mechanism using text could also help manipulate and control the generative mechanisms. The best view of the latent space is with the concentration index as could be seen in fig 17

State of art

The first interesting feature is the fact that the neural network is able to remove the other galaxies on the background that we are not interested in. As shown on the first row of the figure 5, the central galaxy is conserved but the background becomes dark.
Moreover, about the background, we want that the network conserves and shows different features in the morphology of the galaxies. On the second row, we see a barred spiral galaxy with its satellite galaxy (a smaller galaxy orbiting around a major one). The two bodies are conserved, which means the neural network can differentiate background and features, but also their morphology. Which is why the central light dot is elongated. Nonetheless, the color of the satellite is not conserved. We estimate that, since the training set is in majority composed with red/yellow galaxies, the neural network struggle with non-centered blue targets. Centered target seem to be non-affected (see row 3).
On the other hand, we observe some wrong generations. On the fourth row, we observe a similar case as above, a galaxy and its satellite. But the generated image is only a bright dot elongated. When the secondary is too close to the centered galaxy, the neural network doesn't differentiate the 2 bodies and interpret it as a large target.
Some galaxies seem to be "smothered" by the architecture. The last row shows a spiral galaxy, but the generated image doesn't conserve the arms on the outside and instead create a large light dot. The result can then be confused with lenticular galaxies, which is clearly a issue considering the context of our work.

V. DISCUSSION

This project allowed us to discover the difficulties and issues of image generation. First, it appeared clearly that the GAN model wasn't a good idea for us to use. If such models are used and can efficiently generate good images, they use various tricks to ease the training of the discriminator and the generator at the same time, which proved very difficult. The GAN model appeared much more computationally expensive than the VAE

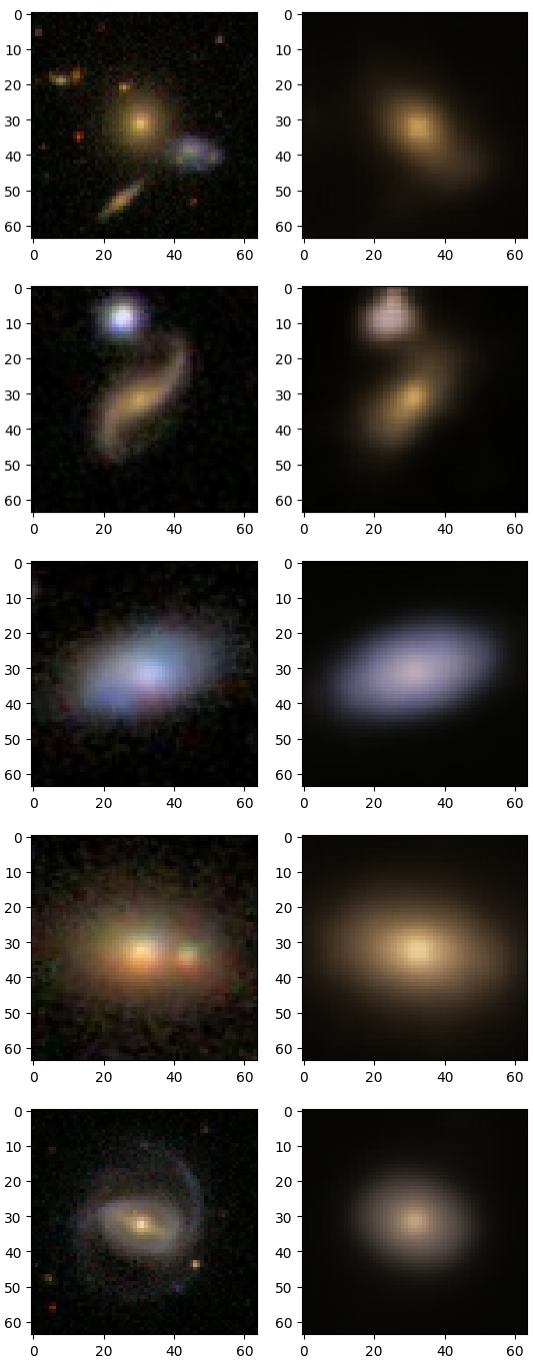

FIG. 5. Left : Entry of the encoder CNN. Right : output of the decoder CNN.

models we tried later. Furthermore, despite our efforts, the GAN appeared to be inefficient and did not produce any satisfying output. This confirms our considerations from the proposal and what is stated in several articles we read: GAN models are difficult to train and other, better, models exist for image generation. This is why we chose to use a VAE, which appears to be easier to use and train. Even a simple form of VAE composed only of MLP networks for the encoder and the decoder (without any further improvement) proved more efficient in image generation than the GAN did. This first version converged in very few iterations, which is a sign that the model easily found a sampling of the latent space giving the ideal weights, and not changing at all. This is very likely to imply a lack of diversity in the outputs and has been confirmed by the images generated (indeed, the whole batch looked almost identical as we can see in figure 6). An improvement of the results was made by introducing a dropout mechanism in the model, with which images were once again very similar but, still, presented some variations they previously didn't. Testing the limits of this still simple model, we observed that the batch size had an impact, and a visual inspection shows that a batch size of 32 may be the most adapted. Also, the dropout probability doesn't seem to have a major influence on the generated images, making it relevant to keep the same value of 0.2 for all tests.
On a more general note, it is clear that the use of CNN for the encoder and the decoder provides much higher performances than the one of MLP. Several analyses similar to the ones performed on the MLP were applied to the CNN VAEs.

In both cases, it appears that data augmentation (performed following the hints and justifications from [6] that also worked on a GalaxyZoo dataset) improved the model's performances. Indeed, the larger diversity implied by this data augmentation allowed the network for training from a larger dataset, learning more different features and possibilities that it could reproduce afterward when generating its images.

[1] Kyle W. Willett, Chris J. Lintott, Steven P. Bamford, Karen L. Masters, Brooke D. Simmons, Kevin R. V. Casteels, Edward M. Edmondson, Lucy F. Fortson, Sugata Kaviraj, William C. Keel, Thomas Melvin, Robert C. Nichol, M. Jordan Raddick, Kevin Schawinski, Robert J. Simpson, Ramin A. Skibba, Arfon M. Smith, and Daniel Thomas. Galaxy Zoo 2: detailed morphological classifications for 304 122 galaxies from the Sloan Digital Sky Survey. *Monthly Notices of the Royal Astronomical Society*, 435(4):2835–2860, November 2013.

[2] `https://github.com/MrPacha/project-dl`.

[3] Yin Cui, Yongzhou Xiang, Kun Rong, Rogerio Feris, and Liangliang Cao. A spatial-color layout feature for representing galaxy images. In *IEEE Winter Conference on*

Applications of Computer Vision, pages 213–219, Steamboat Springs, CO, USA, March 2014. IEEE.

[4] François Lanusse, Rachel Mandelbaum, Siamak Ravanbakhsh, Chun-Liang Li, Peter Freeman, and Barnabás Póczos. Deep generative models for galaxy image simulations. *Monthly Notices of the Royal Astronomical Society*, 504(4):5543–5555, May 2021.

[5] Ali Razavi. Generating Diverse High-Fidelity Images with VQ-VAE-2.

[6] S. Hackstein, V. Kinakh, C. Bailer, and M. Melchior. Evaluation metrics for galaxy image generators. *Astronomy and Computing*, 42:100685, January 2023.

[7] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models, April 2022. arXiv:2112.10752 [cs].

[8] Due to computation and server difficulties from Paperspace.com, we haven't been able to generate the batch of images as we usually do. We would have appreciated having this figure for a visual comparison of the generation goodness, but the comparison of the test loss in both cases already provides a good understanding of the effect of data augmentation.

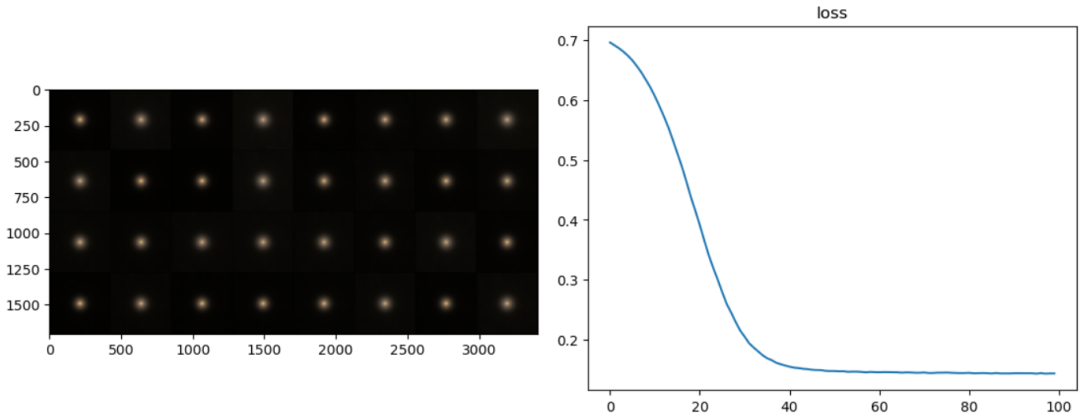

FIG. 6. Left: Batch (32 images) of generated images by the first MLP VAE. Right: Evolution of the training ELBO loss as a function of the epochs (trained on 100 epochs).

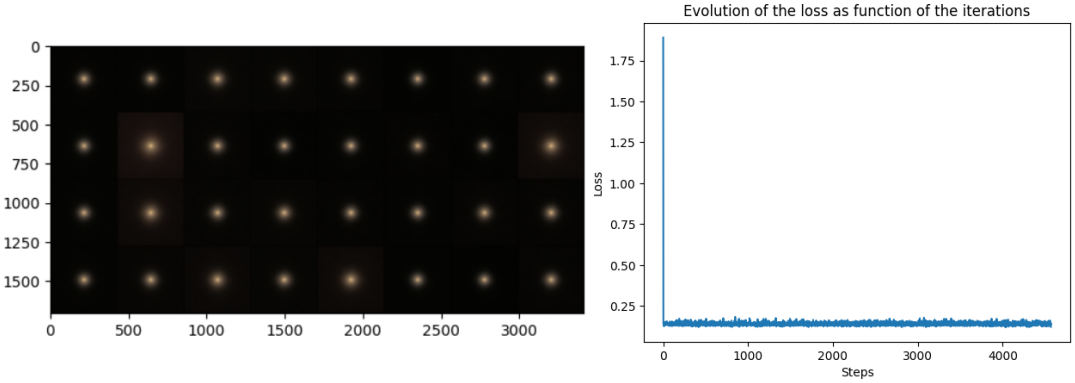

FIG. 7. Left: Batch (32 images) of generated images by the second MLP VAE. Right: Evolution of the training ELBO loss as a function of the epochs (trained on 100 epochs).

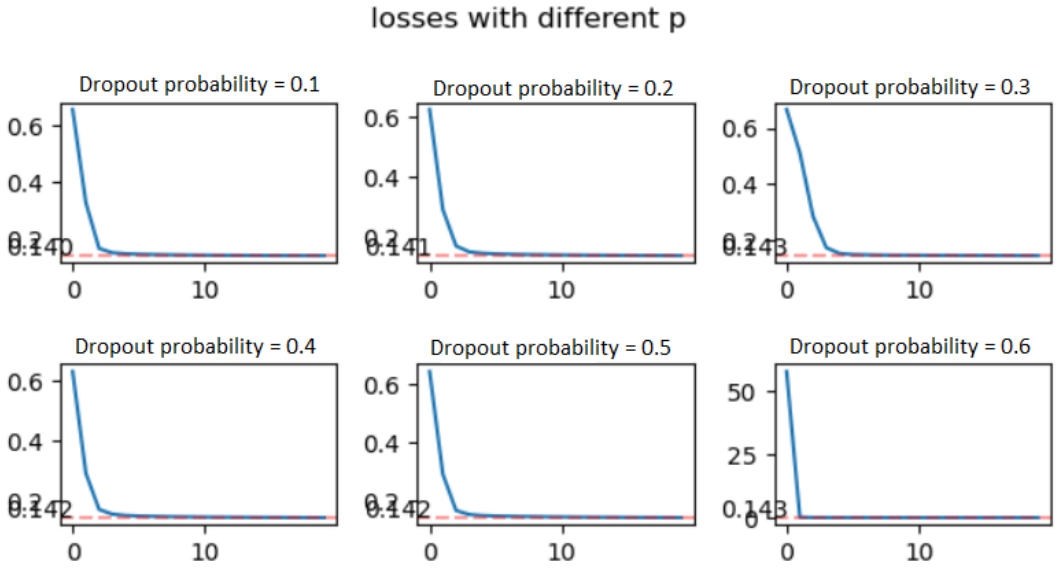

FIG. 8. Evolution of the training loss along the epochs, depending on the value of the dropout probability p.

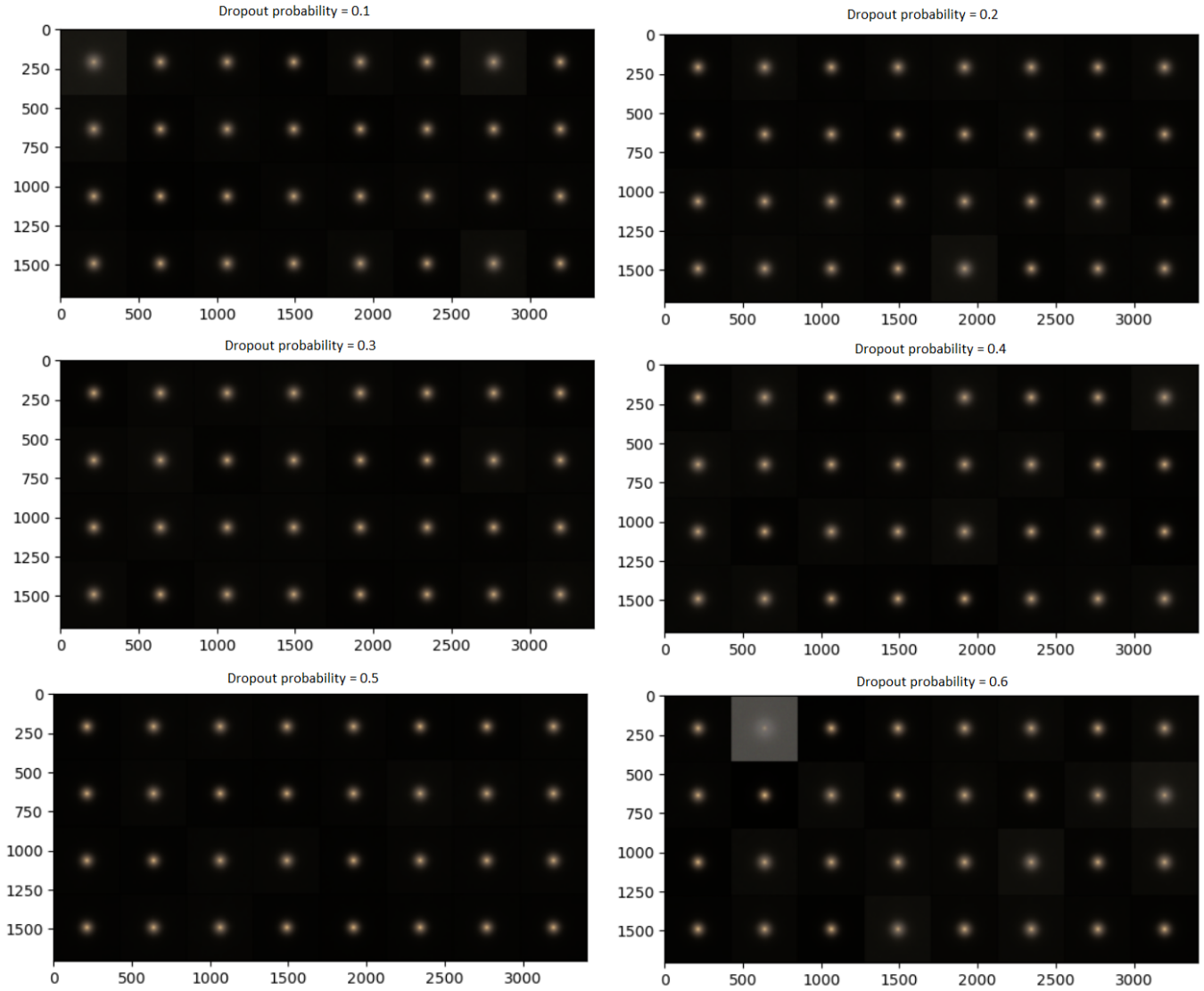

FIG. 9. Batch (32 images) of generated images for each value of the dropout probability that have been tested.

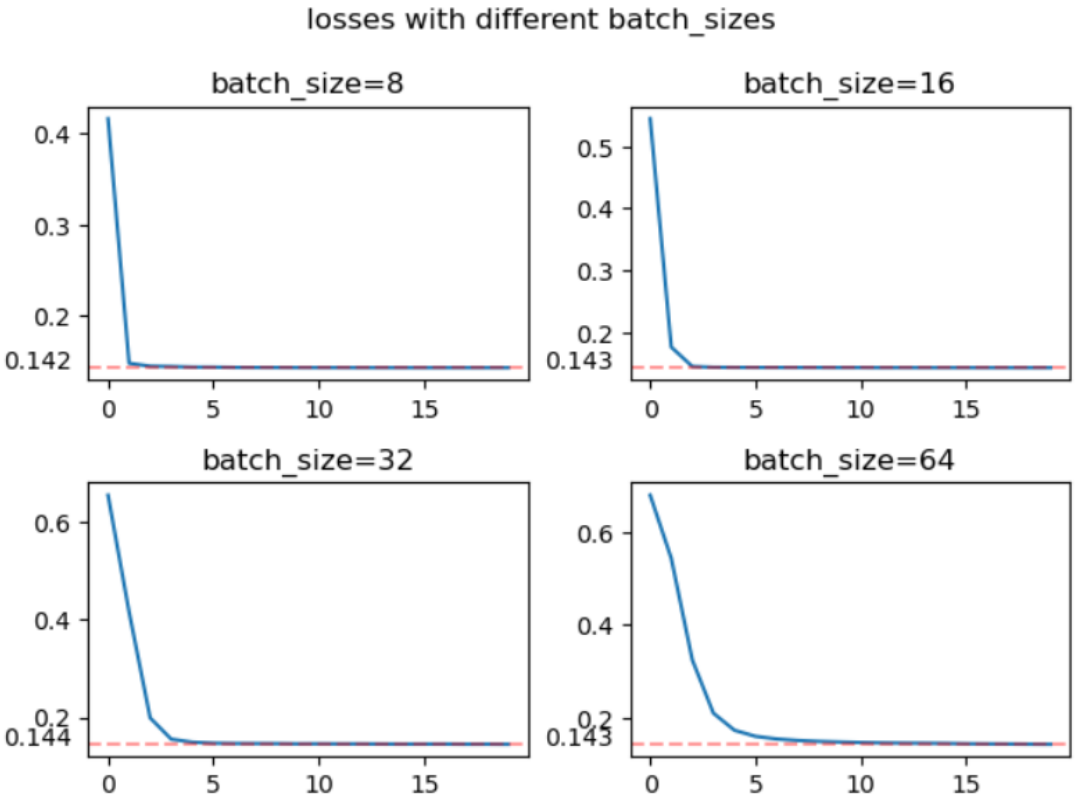

FIG. 10. Evolution of the training loss as a function of the epoch, for different values of the batch size.

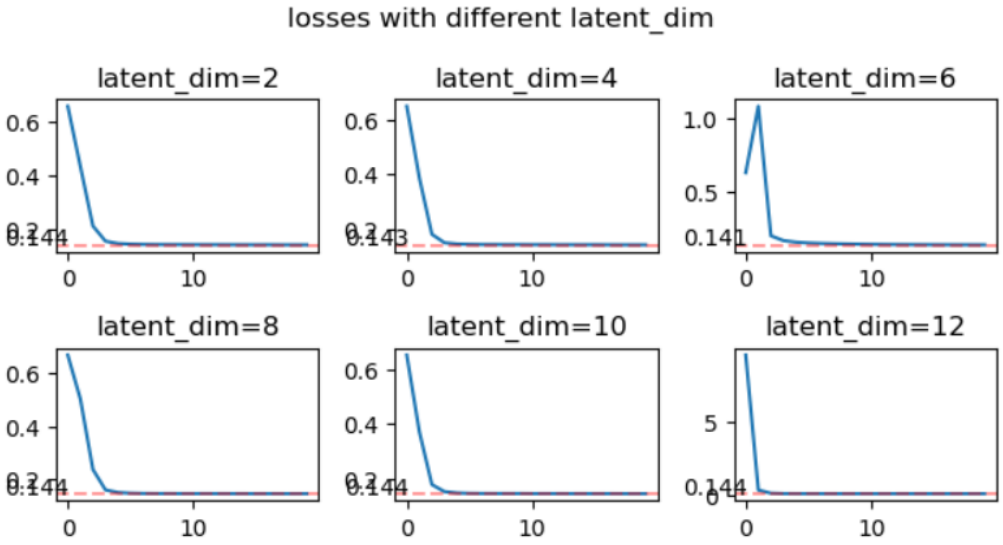

FIG. 11. Evolution of the training loss as a function of the epoch, for different values of the latent space size.

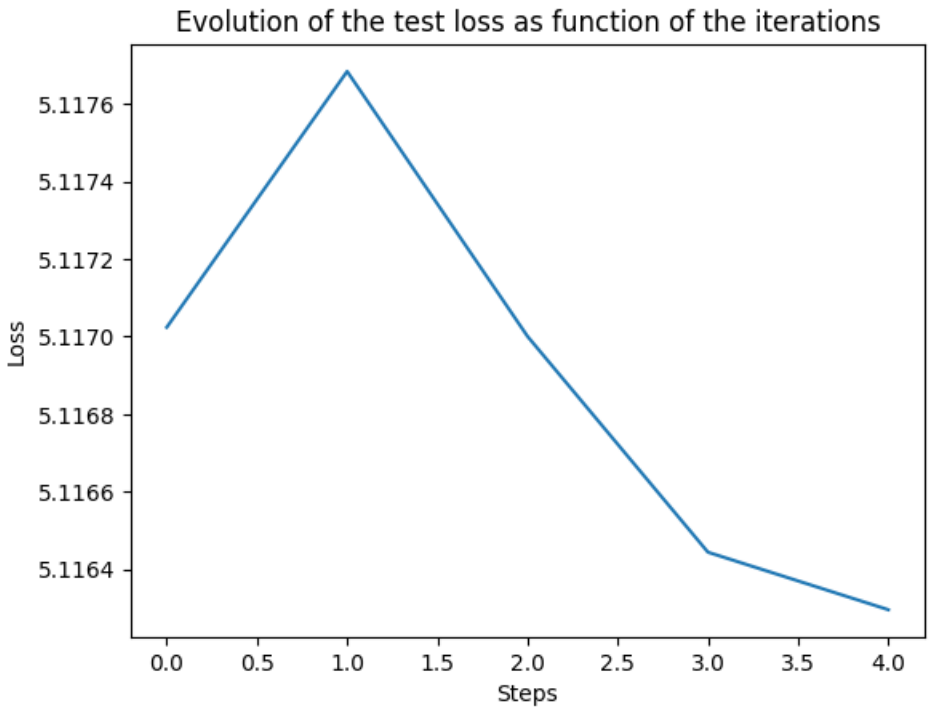

FIG. 12. Evolution of the test loss along the epochs, for the MLP VAE with an augmented dataset.

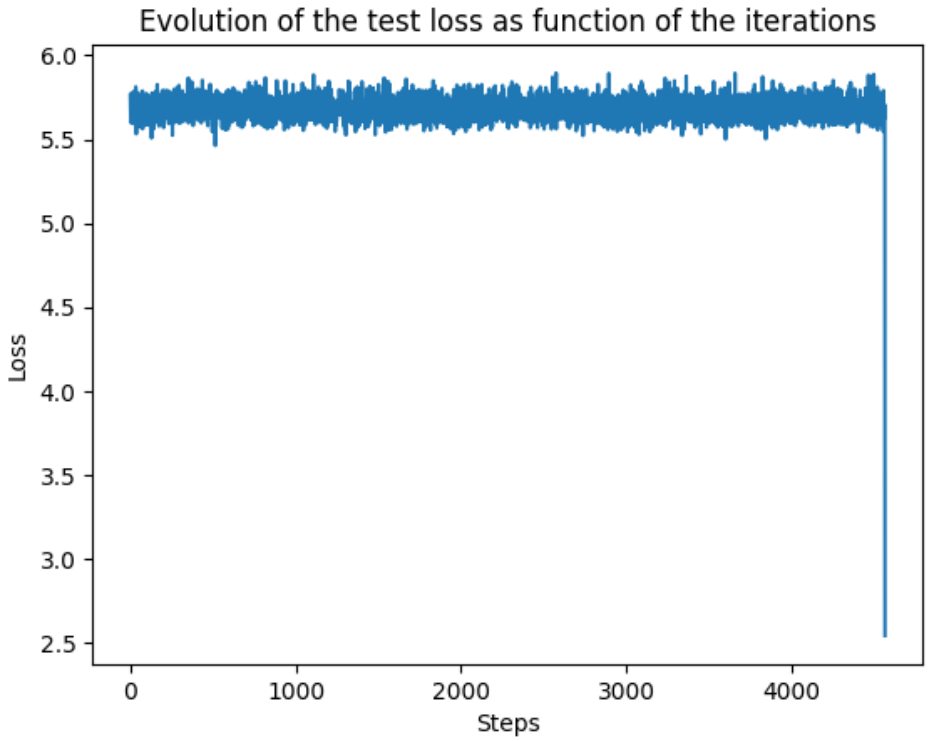

FIG. 13. Evolution of the test loss in one epoch, for the MLP VAE with an augmented dataset.

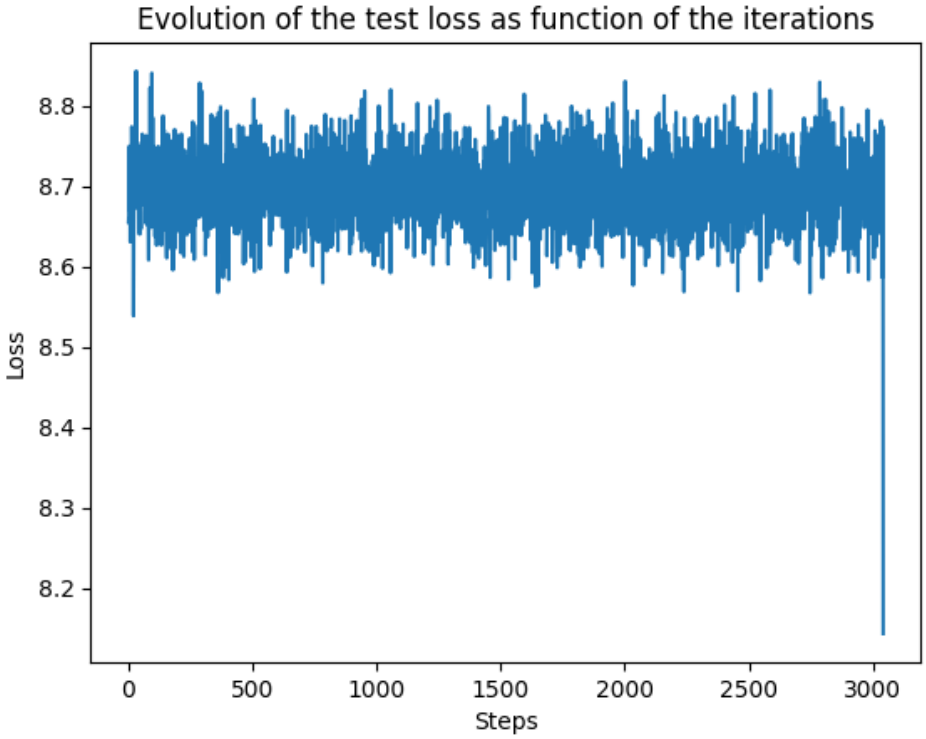

FIG. 14. Evolution of the test loss in one epoch, for the MLP VAE without data augmentation.

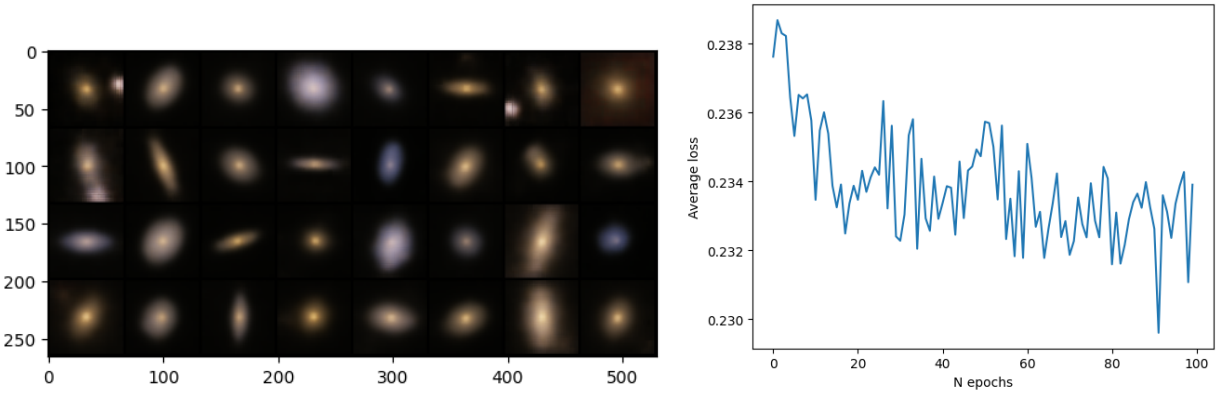

FIG. 15. Left: Batch (32 images) of generated images by the CNN VAE. Right: Evolution of the training ELBO loss as a function of the epochs (trained on 100 epochs).

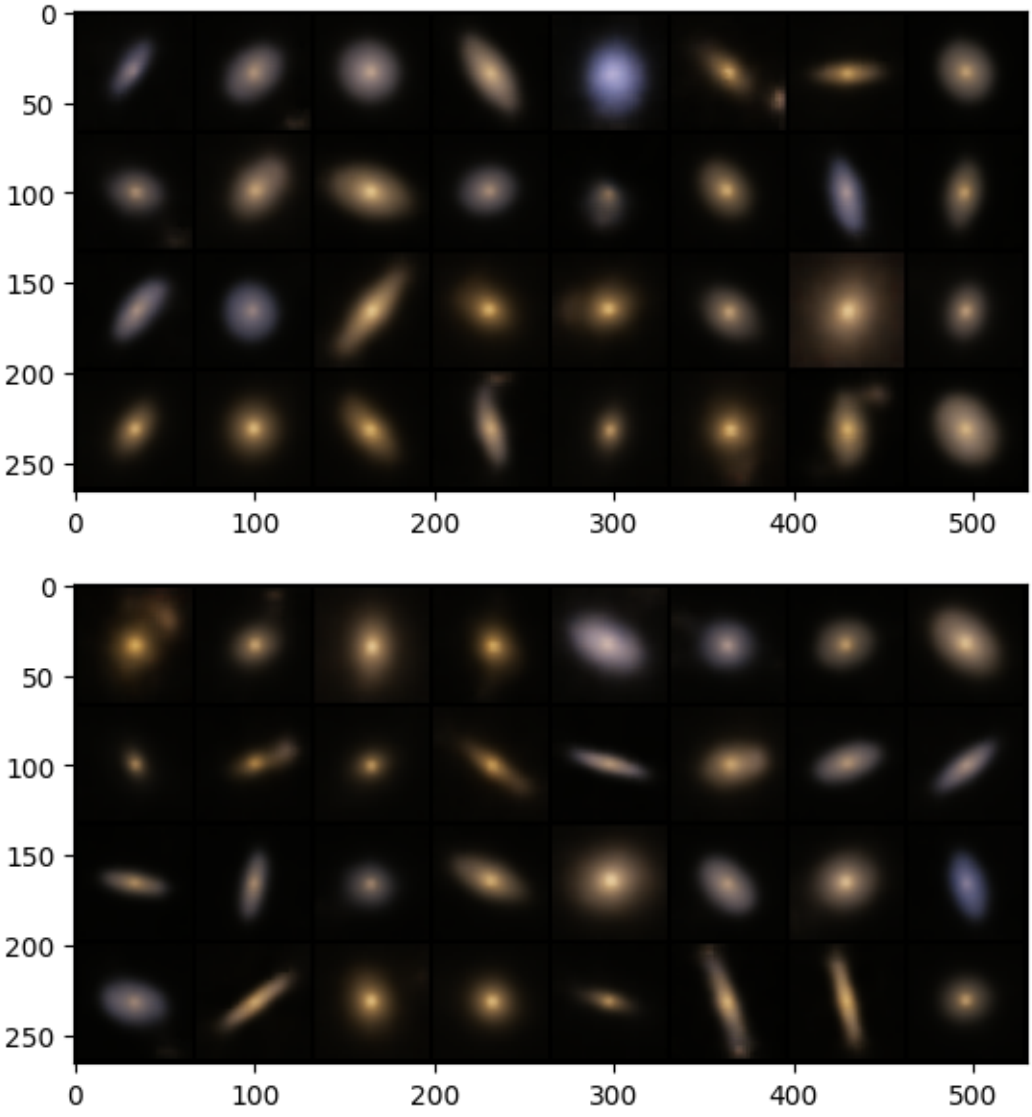

FIG. 16. Samples from the CNN (batch size = 32).

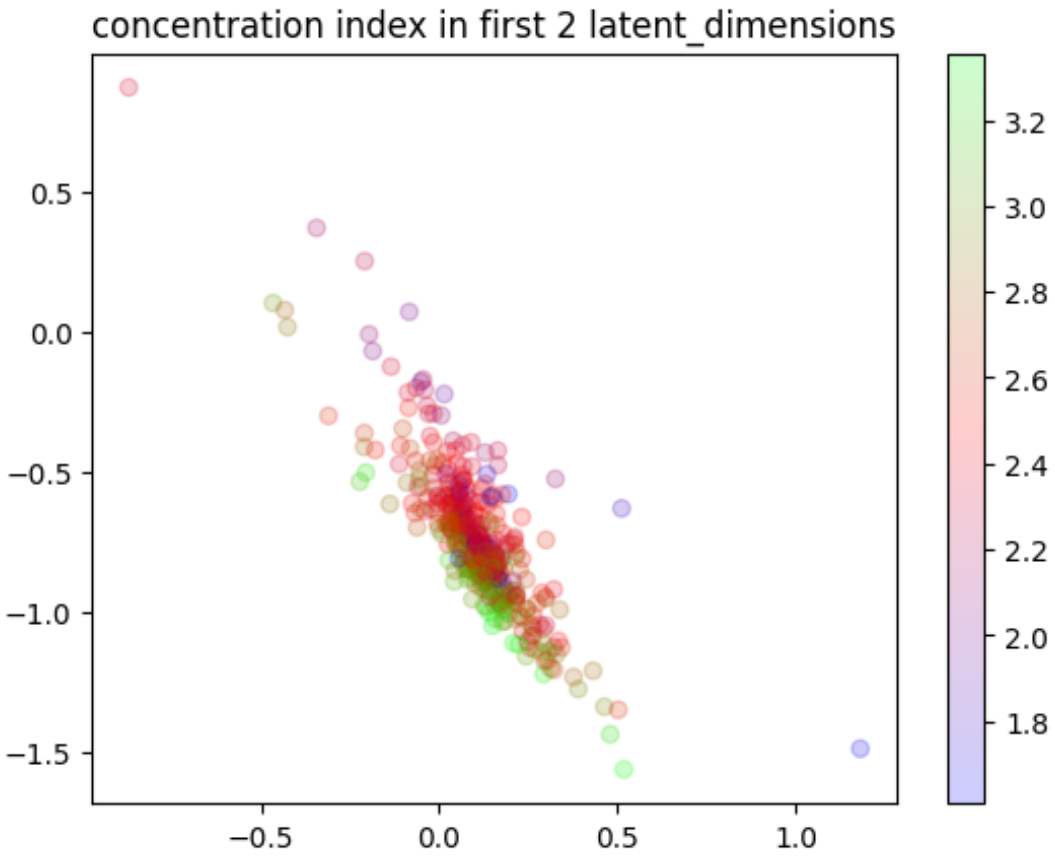

FIG. 17. View of the 2 first dimensions of the latent space generated with the CNN autoencoder. The index is the concentration index of statmorph [a]

[a] https://statmorph.readthedocs.io/en/latest/