


Deep Generative Neural Networks for Galaxy Image Simulations

E.Danilov

EPFL (École polytechnique fédérale de Lausanne)
e-mail: egor.danilov@epfl.ch

October 06, 2020

ABSTRACT

The trend on all-sky surveys will provide the scientific community with thousands of objects, though it makes scientists look for ways of carrying out and validating the analysis of the images obtained with these wide-field surveys. One of the problems is that the typical representation of a galaxy relies on a simple parametric fit profile, therefore does not completely describe it. The other problem is that the use of Markov Chain Monte Carlo sampling for this parametric fitting takes a lot of time. Luckily, machine learning provides a solution to both problems. In this work we present a *Deep Neural Network* called *Variational autoencoder* that can encode a galaxy image as a set of 64 parameters and decode the image back from this representation with minor structural losses, therefore solving the problem of usually parametric fits simplicity. Furthermore, the VAE can encode and decode back thousands of images in seconds, making the parametric fitting much faster than MCMC sampling. We used the HST COSMOS dataset to train the VAE. The VAE itself was based on convolutional layers. Experiments with loss function lead to a conclusion that undamped Kullback-Leibler loss over regularizes the latent space resulting in random VAE output. Constrained Kullback-Leibler loss (Flow-VAE) enables the model to have determined output, however, Flow-VAE has poor reconstruction quality along with latent space inconsistent with the prior. Multiplicative decrease of Kullback-Leibler loss strength by $\beta = 10^{-2}$ (Beta-VAE) leads to good reconstruction quality and plausible latent space. A Framework for convenient VAE architecture experiments was developed and published on Github. 

Key words. Image processing, Machine learning, Variational autoencoder, Galaxies, Image simulation, HST, COSMOS

1. Data collection

The dataset was constructed from the COSMOS catalog images taken by Hubble Space Telescope (COSMOS). In this work, we chose F814W<23.5 sample that refers to 56000 real galaxy images. Along with the images themselves, the dataset contains a lot of useful information like Sersic fit parameters, angle of galaxies' tilt with respect to image axes, and background noise variance. Galaxies' images and parameters were extracted using a specialized python library *Galsim*.

1.1. Images size

In the task of VAE training on galaxy images, the ordinary image size to work with is 64x64. However, when working with *Galsim* one should know that the interface of drawing the images in *Galsim* is not very user-friendly and leaves some questions. Namely, explicitly setting the desired number of pixels nx, ny in the method *drawImage* leads to cropping the image instead of rescaling it. An example can be seen in Fig. 1. I have encountered an article, the author of which didn't account for this fact resulting in incorrect images, hence controversial results. Therefore it is crucial to obtain the images for the dataset in correct manner, however, it is also not so simple. Instead of setting argument *scale* in a natural form *desired_size/original_size* one should use the formula 1.

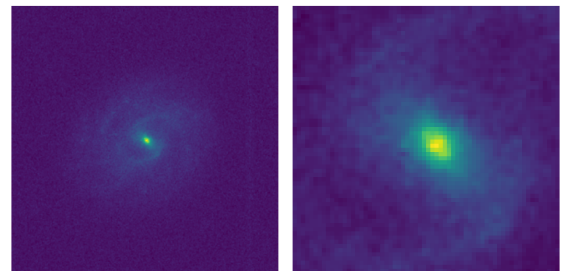


Fig. 1. Comparison of *Galsim drawImage* methods results. Both images are 64x64. The image on the left was created by adjusting the argument *scale* according to Eq. 1. This image contains 99.5 % of the flux as it is meant to do. The image on the right was created using arguments nx, ny or *bounds* and appears to be cropped, hence irrelevant.

Where 0.03 is actually *pixel_scale* - coefficient of conversion to arcsec.¹

¹ The formula was obtained experimentally and still do not work in 2 of 56000 cases. When working with *Galsim*, it is advised to use all kinds of sanity checks.

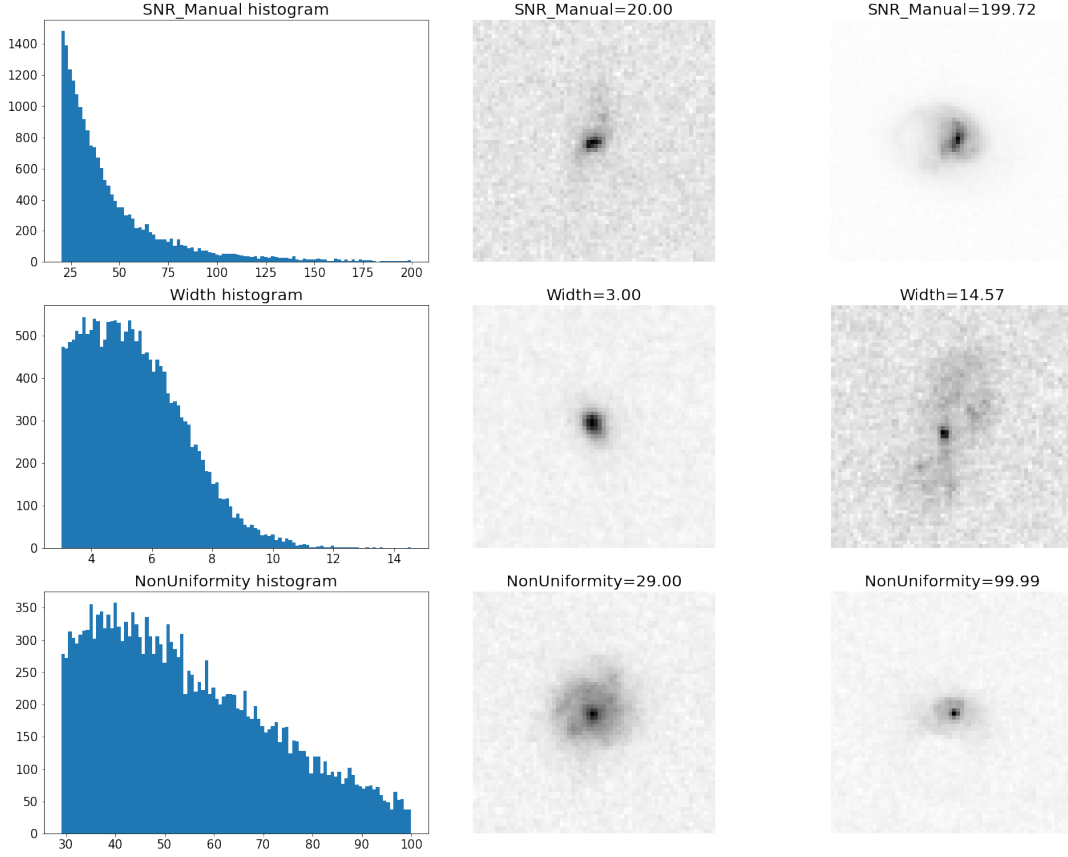


Fig. 2. Representation of metrics (eqs. 2) in filtered dataset. For a given metric, on the left - histogram, on the middle and right - galaxies referring to the lowest and the highest values of the metric correspondingly. SNR and Width are not constrained on the top, however, constraints increase the visibility of histograms and almost do not affect the appearance of characteristic images.

$$Scale = 0.03 \times \frac{original_size}{desired_size - 2} \quad (1)$$

1.2. Noise extraction

SNR of images in the dataset appears to be a very important feature for VAE, hence we extract the noise of the image ourselves, in order to be sure that it is correct. Given the 64x64 image, the noise was extracted as a minimum value of the standard deviation of border 8x8 regions. Such a procedure is reasoned by the fact that some spiral edge-on galaxies are placed diagonally on the image, hence their outer regions can be present in 8x8 border areas. Taking the minimum standard deviation ensures that we extract specifically the noise of the background.

2. Data filtering

We want our neural network to extract the data about the structural properties of the galaxies, however, images differ not only in these features. For example, datasets contain very bright and small galaxies with indiscernible structure along with galaxies that are indistinguishable from the noise. The effect of these confounding factors would be excluded if we had a dataset with the same SNR and galaxy sizes for all the images. However, it is not the case. One way of excluding the effects of the confounding factors is

to use only pairs of images that match both confounding factors. However, for this particular case, exact matching in SNR and galaxy width seems too strict and less useful than just restricting these two features with relevant limits that would result in images with distinguishable structure and not much noise. Though the quality metric is quite abstract and the limits have to be set using visual estimations.

2.1. Quality metrics

The metrics used are presented in eqs 2 and Fig. 2.

$$\begin{aligned} SNR &= I_{1/4} / \sigma_{noise} \\ Width &= \max(\sigma_{gaussian}) \\ NonUniformity &= \max(I) / \text{mean}(I) \end{aligned} \quad (2)$$

2.2. Signal-to-noise ratio

SNR is the most important factor in the filtering since $1/SNR$ represents the uncertainty of information about the galaxy that we obtain. The stochasticity of VAE, which is given by variance of latent space represents necessarily the same uncertainty, at the same time providing the continuity of latent space. Noise-governed images with low SNR would increase VAE's variance resulting in completely random VAE's output. SNR metric represents Signal-to-noise ratio proposed by Chianese (Chianese et al. 2020) based on

Conv2d(1,64,4,2,1)
LeakyReLU(0.2)
Conv2d(64,128,4,2,1)
BatchNormalization
LeakyReLU(0.2)
Conv2d(128,256,4,2,1)
BatchNormalization
LeakyReLU(0.2)
Conv2d(256,512,4,2,1)
BatchNormalization
LeakyReLU(0.2)
Conv2d(512,4096,4,2,0)
LeakyReLU(0.2)
Flatten
$\mu = \text{Dense}(4096,64)$
$\log_ \sigma = \text{Dense}(4096,64)$
Reparametrization
Encoded

Table 1. Architecture of the **encoder**. The notations used in convolution layers are number of input channels, number of output channels, kernel size, stride, zero paddings. LeakyReLU argument is the slope for negative values, Dense layers arguments are the number of input channels and output channels. All convolutional layers are unbiased and initialized with He Normal initializer.

a maximum of downscaled to 16x16 image and the noise manually extracted from 8x8 border regions.

2.3. Width

The other problem is that small dot-like galaxy do not have immediate structure, therefore they are not useful in our studies and may bias the training of VAE. Filtering galaxies by *Width* we ensure that the maximum size of the galaxy is bigger than some threshold, therefore it is not dot-like.

To obtain *Width* the images were rotated such that the galaxies longest size lied horizontally. After that images were fitted with 2D Gaussians and the standard deviation of the horizontal Gaussian were taken as *Width*. By choice of rotation, it is the maximum standard deviation that one can obtain from the 2D Gaussian fitting of the image.

2.4. NonUniformity

The NonUniformity metric compensates for the shortcomings of other metrics. For example one may have image with high but uniform background, namely $\max(I) \gg \sigma_{noise}$ along with $\max(I) \sim \text{mean}(I)$. The result would be a uniform image with necessarily high SNR and Width, though not relevant for our studies. Otherwise, a wide source with all the flux near the highest border of the detector's dynamical range would be represented as a wide dot, also without

Conv2dTranspose(64,512,4,2,0)
BatchNormalization
LeakyReLU(0.2)
Conv2dTranspose(512,256,4,2,1)
BatchNormalization
LeakyReLU(0.2)
Conv2dTranspose(256,128,4,2,1)
BatchNormalization
LeakyReLU(0.2)
Conv2dTranspose(128,64,4,2,1)
BatchNormalization
LeakyReLU(0.2)
Conv2dTranspose(64,1,4,2,1)
Sigmoid
Decoded

Table 2. Architecture of the **decoder**. The notations of layers are the same as for the encoder Tab. 1. All the convolutional layers are unbiased. All convolutional layers except for the last one are initialized with He Normal initializer. The last convolutional layer is initialized with Xavier (Glorot) Normal initializer.

any structure. In the first case, NonUniformity would be too low, and in the second - too high. The sense of the metric itself is to ensure that specifically the flux of the source, not the background, covers most of the detector's dynamical range.

3. Architecture

3.1. Introduction to autoencoders

The very sense of the Autoencoder is to compress or *encode* high dimensional data to some low dimensional representation *latent space* and *decode* the initial data from this representation. During the training, the autoencoder tries to learn the function $x = \text{decode}(\text{encode}(x))$ with the highest possible precision. A traditional autoencoder encodes data to a latent vector and decodes from it. *Variational autoencoder* encodes data to vector of means μ and vector of variances σ . Then parametrizes these two vectors to one by sampling values from normal distribution set by vectors μ and σ . This approach leads to not discrete, but continuous latent space, which is useful for image generation. If one wants to see more deep explanations he may see (Tensorflow VAE),(Tensorflow CVAE) or (Chianese et al. 2020). As a base of VAE's architecture, we used one from (Chianese et al. 2020), that represents convolutional VAE, with a decoder mirroring the encoder. The architecture used in this article is presented in Tab. 1 and Tab. 2. The architecture of our VAE is different from one presented in (Chianese et al. 2020) and all the choices of its layers are discussed in this section.

3.2. Variational type

Variational autoencoders are better than traditional autoencoders in tasks of generating images since they work with continuous latent space, hence enable one to generate relevant images continuously transforming from one to another with the change of latent parameters.

3.3. Convolutional layers

Let's consider the human's visual perception. The human brain distinct object by their integral properties, namely, in order to understand the object's natural human eye, focuses either on the entire object or on its insight meaningful parts. Building analogy with computer vision we can interpret integral estimation of properties as convolution and splitting objects image into parts as a choice of kernel size. In these terms, strides provide an economy of processing power, since we are not likely to extract two very different integral properties from regions that differ in a very small area.

3.4. LeakyReLU layers

As usual, ReLU activation layers are meant to add non-linearity to the model, in these terms LeakyReLU is a less strict function since it doesn't cut off half of the data letting it propagate to further computations, but with decreased impact.

3.5. Batch normalization layers

In (Ioffe & Szegedy 2015) authors showed that BatchNormalization increase performance of Deep Neural Network learning by boosting learning up to 14 times, enabling one to be less careful about initialization along with regularizing effect, eliminating the need for Dropout. Given 36 million parameters and hundreds of epoch on 15 thousand images dataset, our neural network's training time on Google Colab GPU is approximately 8 hours with BatchNormalization. This learning time is quite convenient for carrying out experiments with VAE architecture. A refusal to use BatchNormalization leads to a learning time of approximately 4-5 days, which makes experiments hardly feasible.

Although BatchNormalization may aggravate VAE's stochasticity, however, no significant effect of such kind has been observed during the experiments.

3.6. Reparametrization

As encoder results in μ and \log_σ the common for VAEs reparametrization trick is realized as $z = \mu + N(0, 1) \times \exp(\log_\sigma/2)$, with $N(0, 1)$ being a number randomly sampled from unity normal distribution.

3.7. Loss function

As usual to VAEs when training the model we want to not only increase the reconstruction quality but also to keep latent space distribution consistent with a unit Gaussian. The requirement of reconstruction quality hence results in *binary crossentropy*, whereas holding to the prior distribution is maintained by *Kullback-Leibler divergence*.

3.8. Activation layer

Consideration of activation function is based on properties of two objects: result image and loss function. The outline of the facts is as follows: original images are normalized to $[0,1]$, binary cross-entropy contains logarithms, therefore requires non-negative input. The natural solution is to activate the last layer with a function that has $[0,1]$ output. Given the fact that NonUniformity metric eq. 2 ensures that galaxy light should cover most of the detector's dynamical range, we can state that low pixel values should refer to either sky or external regions of the galaxy, whereas high pixel values would refer to the center of the galaxy. The main structural properties of a galaxy are likely to manifest themselves between the center and external regions, therefore in middle values of $[0,1]$ pixel value range, that necessarily refer to the linear region of the sigmoid function. In these terms, sigmoid seems to be the perfect option, with one disadvantage that it saturates pixel values that are on $[0,1]$ border in terms of brightness, though not distorting regions representing galaxy's structure.

3.9. Initialization

Initializing weights as constant as advised in (Chianese et al. 2020) leads to an immediate explosion of gradients, hence divergence of all the weights on the very first epochs. Therefore we used a different, more subtle approach, namely using He Normal initializer (He et al. 2015) for all the convolutional layers passing their results to LeakyReLU layers, so all the convolutional layers except for the last one. The last layer, that passes results to the sigmoid activation function is initialized by Xavier Normal initializer (Glorot & Bengio 2010). The difference between these two initializations is quite straightforward. Usual Xavier initializer preserves variance. ReLU turns half of Z-values (negative ones) to zero, effectively halving the variance. LeakyReLU at the same time does not neglect all negative values but decreases their effect nonetheless. In these terms, ReLU effectively lowers the variance by 2, so to compensate that we just double initialized variance, resulting in He initializer from Xavier one.

4. Training

4.1. Training dataset

The data obtained in the section *Data filtering* was normalized to values $[0,1]$ by subtracting $\min(I)$, and then dividing the image by its $\max(I')$. The dataset was reshaped to tensorial form, namely tensor $(N, 64, 64, 1)$ and splitted into train, test and validation datasets with $N = 15520, 512, 1248$ correspondingly. Such numbers were chosen as multiples of $batch_size = 32$.

4.2. Training algorithm

The model was trained using Adam optimizer (Kingma & Ba 2015) with moments $(\beta_1, \beta_2) = (0.5, 0.999)$ (Chianese et al. 2020) with start learning rate 10^{-6} . The loss function used is combination of *binary crossentropy* and *Kullback-Leibler divergence*, namely eq. 5. There y_{true}



Fig. 3. Impact on the image of the only deterministic latent variable (the first on Fig. 5) of original VAE with loss eq. 5

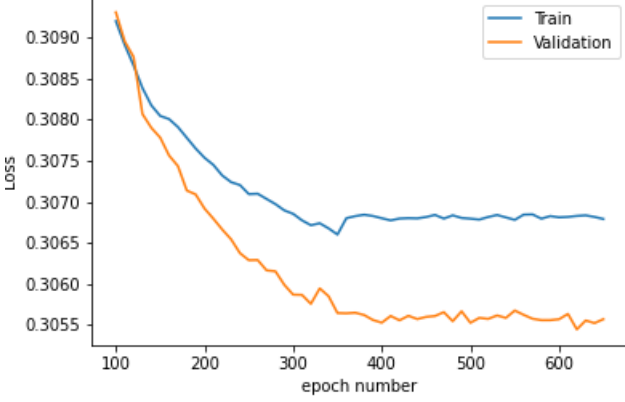


Fig. 4. Learning curve of the VAE, trained with loss eq. 5

- initial galaxy images, y_{pred} - VAE reconstructed galaxy images, μ and \log_{σ} - VAE's latent space variables.

$$BCE_loss = \sum_{i,j}^{64} y_{true}^{i,j} \times \log(y_{pred}^{i,j}) + (1 - y_{true}^{i,j}) \times \log(1 - y_{pred}^{i,j}) \quad (3)$$

$$KL_loss = \frac{1}{2} \sum_i^{64} 1 + \log_{\sigma_i} - \mu_i^2 - \exp(\log_{\sigma_i}) \quad (4)$$

$$Loss = \frac{1}{64 * 64} [BCE_loss - KL_loss] \quad (5)$$

4.3. Training

The training was performed on a dataset of 15520 images with validation on 1248 images. Start learning rate was $lr = 10^{-6}$, though it was set to automatically decreased by the factor of 10 if the metric stops improving for 25 epochs. Consideration of learning curve Fig. 4 leads us to the fact that the model falls to the local minimum of the loss function eq. 5 on ≈ 300 epoch.

4.4. Results

The consideration of reconstructed images leads to the conclusion that the reconstructed results are highly dependent on the stochasticity of sampling, namely reparametrization $z = \mu + N(0, 1) \times \sigma$. In the concept of VAE as a galaxy's

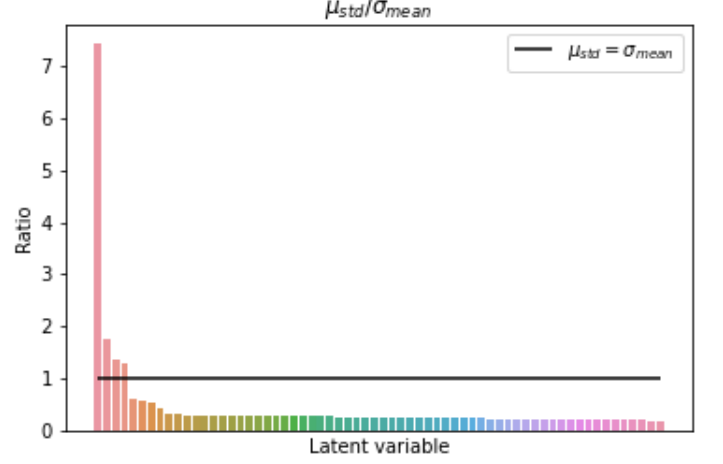


Fig. 5. Signal-to-Noise ratio for each of 64 latent z . SNR is comprised of μ 's standard deviation and latent σ 's mean. The sense is that $z = \mu + N(0, 1) \times \sigma$, therefore completely deterministic latent z refers to $\mu_{std} \gg \sigma_{mean}$, whereas $\mu_{std} \leq \sigma_{mean}$ leads to random output.

characteristic, we consider it's latent μ or, more specifically, deviation of its value from the mean value on the entire dataset $\mu - \mu_{mean}$. The latent σ is uncertainty or noise, corresponding to this characteristic. To work out SNR for each of 64 latent z we should consider overall dataset characteristics of a given latent z . Therefore as a signal we consider an overall dataset deviation of μ , namely latent μ 's standard deviation μ_{std} . As noise we consider mean value of latent σ , namely σ_{mean} . In the result, SNR_z^i for $i = 1 \dots 64$ is presented in eq. 6 and Fig. 5. Working with mean and standard deviations of latent variables μ and σ is relevant since they are indeed distributed as Gaussians (i.e. Fig. 13). Moreover, such an SNR metric implicitly shows over-regularization. When latent space is under-regularized KL_loss is very high, which is caused by a very low value of σ , which necessarily leads to very high values of SNR. As we expect to have Latent variable SNR approximately the same as images SNR the plausible values are $\sim 20 - 200$.

$$SNR_z^i = \frac{\mu_{std}^i}{\sigma_{mean}^i} \quad (6)$$

Given the Fig. 5 we see that $SNR \gg 1$ for only one latent variable all the others are governed with random. For a better understanding, we can construct an average galaxy by decoding a vector with $mean(\mu)$ (the middle one on the Fig. 3). The average galaxy is sensibly a gaussian. After that we construct galaxy images from this average latent vector but varying deterministic variable $z_j^0 = \mu_{mean}^0 + j \times \mu_{std}^0$ with $j = -3, -2, \dots, 2, 3$ to cover all the range of variable's values. The result is presented in the Fig. 3. We get the

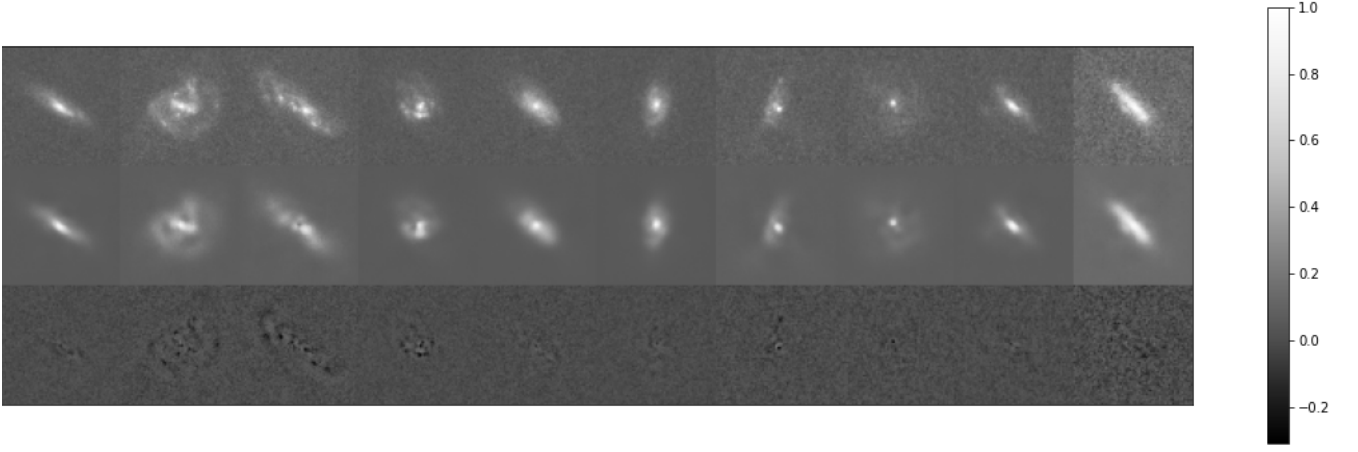


Fig. 6. Reconstruction quality Beta-VAE with Loss function eq. 7 and $\beta = 10^{-2}$. Reconstruction of test dataset images is presented. From top to bottom: initial image I_i , reconstructed image I_d , residual $I_r = I_d - I_i$



Fig. 7. Impact on the image of a latent variable of Beta-VAE eq. 7. The variable considered is the third one on the Fig. 9. One can see that like in the Fig. 3, the variable refers to the size of the galaxy, though now the relation is not so distinct. All the 3 first variables on the Fig. 9 correlate with the size.

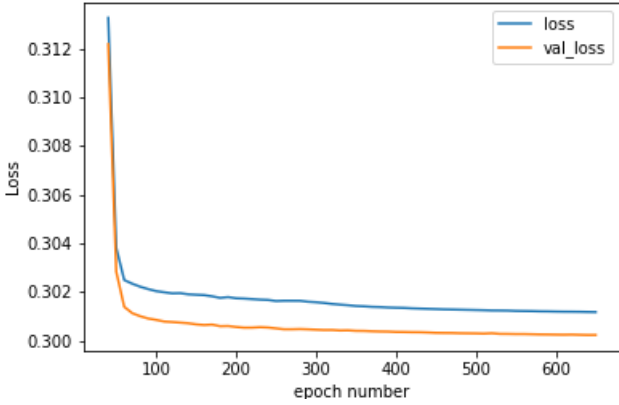


Fig. 8. Learning curve of the Beta-VAE with Loss function eq. 7 and $\beta = 10^{-2}$

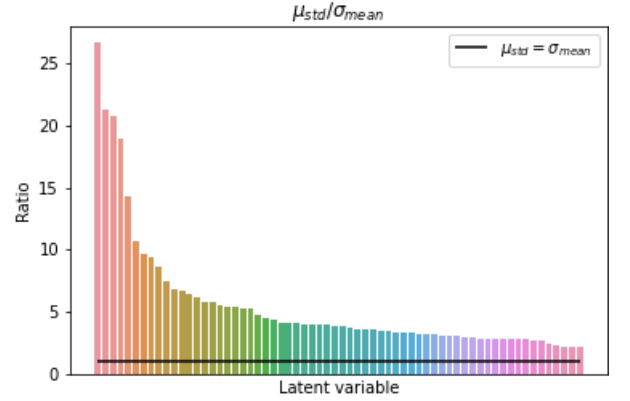


Fig. 9. SNR of latent space variables eq. 6 corresponding to Beta-VAE eq. 7

most sensible outcome. The model understood that it has a bright dot in the center and learned to credibly extract the dot's size.

However, this result is not what we were looking for and not the best result we can get. This stochasticity is a known issue for VAE (Lanusse et al. 2020). The reason is that in the eq. 5, the data fidelity term, namely BCE_loss , is too weak in comparison with KL_loss , which brings the latent space to the proper statistical distribution. In this work, we consider two solutions to this problem.

5. Beta VAE

5.1. Training

The first way to balance BCE_loss and KL_loss terms in the eq. 5 is to multiplicatively decrease the KL_loss impact. VAE with the loss function given by the eq. 7 is called Beta_VAE. β was chosen to be 10^{-2} as it was used in (Arcelin et al. 2020). The model was trained using a constant learning rate of 10^{-7} . Consideration of the learning curve Fig. 8 leads us to the conclusion that only 150 epochs we needed to extract most of the information from the dataset.

$$Beta_Loss = \frac{1}{64 * 64} [BCE_loss - \beta \times KL_loss] \quad (7)$$



Fig. 10. Reconstruction quality Flow-VAE with Loss function eq. 8. First image is the one to be reconstructed, then the reconstructed images with $\lambda = 4, 8, 16, 32, 40, 48, 56, 64$ correspondingly. Models with λ up to 32 are under-regularized, models with $\lambda = 56, 64$. Models with $\lambda = 40, 48$ couldn't learn anything useful.

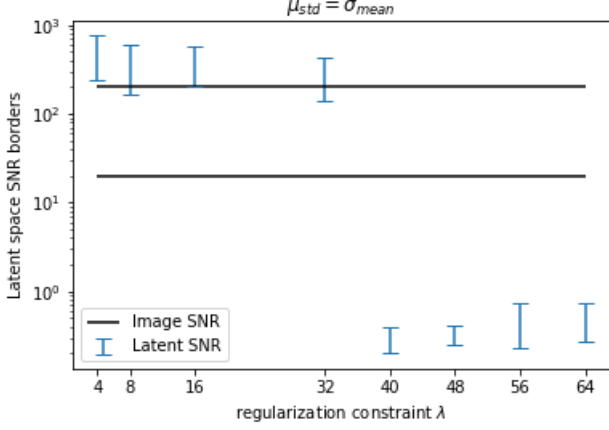


Fig. 11. Dependence of latent SNR values (eq. 6) on regularization constraint λ (eq. 8). The lines on 20 and 200 correspond to effective borders of the dataset image SNR (eq. 2). Correctly train model is expected to have SNR values between these lines.

5.2. Reconstruction quality

Regardless of such a small number of epochs, reconstruction quality is quite good Fig. 6. By residuals on this figure, we see that the VAE has difficulties mostly with the reconstruction of bright small objects. It was expected since convolutional layers of the VAE necessarily lead to smoothing the image, hence decreasing the quality of the small objects, whereas saturating sigmoid activation makes it hard to reconstruct the bright ones.

5.3. Latent space structure

The structure of latent space is still represented by Gaussians Fig. 14 as it is meant to be. In this time SNR of latent space variables eq. 6 refers to deterministic results $\mu_{std} \gg \sigma_{mean}$, as it is represented on the Fig. 9. Though, there is not one variable that controls the size of the galaxy. At least three first variables on the Fig. 9 correlate with the size. The impact of the third one is represented in the Fig. 7. The impact of latent variables on the image is very complex and not understandable in terms of simple characteristics like size, morphology type, or ellipticity.

5.4. Reconstruction features

It is sensible that changing the rotation angle of the galaxy doesn't change its structure, and fortunately, by training on only 15 thousand images, the Beta-VAE managed to learn rotational invariance without explicitly supplementing the dataset with rotated galaxies.

Moreover, due to its convolutional nature, the model also denoises images in the process of reconstruction.

Both effects can be observed in Fig. 12.

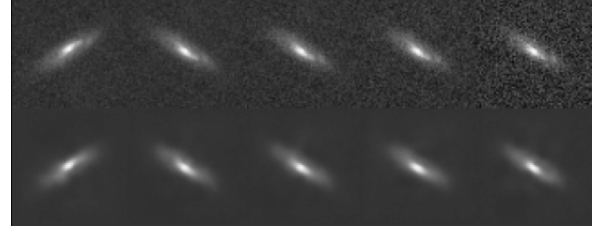


Fig. 12. Rotational invariance and denoising effects of the dataset. All the images are generated from one galaxy. In the top row first image is a rotated version of the original galaxy, represented on the second image. On the other images, a Gaussian noise was put such that SNR is 60,40,20.

6. Flow VAE

6.1. Training

The second way to balance BCE_loss and KL_loss terms in the eq. 5 is to put an explicit constraint on KL_loss term. Here one should know that eq. 4 has only negative values. Therefore a constraint is given in a form of eq. 8, so that we do not let KL_loss to over-regularize the system.

$$Flow_Loss = \frac{1}{64 * 64} [BCE_loss - \max(-\lambda, KL_loss)] \quad (8)$$

To find optimal parameters we tried to train the model with several λ . Starting from proposed in (Lanusse et al. 2020) $\lambda = 4$ we considered powers of 2, and then, additional values 40,48,56 were considered. As it was stated in consideration of latent space SNR metric eq. 6 the plausible values correspond to the SNR of the images in the dataset. Therefore we consider models with $SNR > 200$ as under-regularized, and models with $SNR < 20$ - over-regularized. The starting learning rate was chosen to be 10^{-6} which was automatically decreased by the factor of 10 if the metric stops improving for 30 epochs. The learning curve of each model saturates by 120 epoch, so all the results considered correspond to it. One can see dependence of the latent SNR value span on regularization constraint λ on the Fig. 11.

On the Fig. 11 one can see that the plausible values should lie somewhere within the region $\lambda = (32, 56)$. However, for examined values $\lambda = 40$ and 48 , validation loss starts diverging from 30 epoch. Moreover, these two models learned pretty much nothing as they had SNR values even less than in the case of $\lambda = 56$. As the result, finding a plausible regularization constraint appears to be a difficult task, though not very useful. Even for very strict regularization constraints reconstruction quality (Fig. 10) is much worse than the reconstruction quality of the Beta-VAE (Fig. 6 Image 2), furthermore, the latent space, in this case, does

not correspond to the prior distribution at all. As the result, Flow loss eq. 8 does not give any advantages for the studied VAE architecture, therefore is considered useless.

7. Results

The architecture of the VAE considered Tab. 1, Tab. 2, with latent space dimension $latent_{dim} = 64$ enables one to credibly reconstruct general structure of galaxies. However, the model faces difficulties with reconstructing small objects and bright objects, due to the use of convolutional layers and sigmoid activated output correspondingly.

The use of VAE trained with loss eq. 5 leads to stochastic model output due to over-regularization of the latent space by the Kullback-Leibler divergence term in the loss function. The two solutions inspected were either multiplicative decrease of KL_loss term effect, that results in the Beta-VAE, or setting a constant constraint on KL_loss value, that results in the Flow-VAE.

The Beta-VAE with $\beta = 10^{-2}$ appears to be the most useful of the models considered. It has a high quality of image reconstruction along with high latent variables SNR eq. 6 and useful properties like rotational invariance and denoising.

The Flow-VAE's parameters are more challenging to adjust. For the Beta-VAE the regularization fix can be found easily by inspecting values around $\beta = \sigma^2$ (Higgins et al. 2017), whereas plausible λ values span is not so obviously related to the latent space parameters that we are eager to obtain. Moreover, even though regularization constraint should improve the quality of the picture, it gives worse reconstruction results and latent space distribution than Beta-VAE. Therefore, it is advised to use Beta-VAE architecture, due to good reconstruction quality and latent distribution only slightly deviating from the prior.

8. Discussion

I think that setting a constraint on latent space to make it meaningful is a bit reckless. For example, I want μ^0 to represent half-light radius, so I change the KL loss term eq. 4 to make μ^0 converge to a given value of the galaxy. It may look like a good solution on the first look, however, it restricts the information that can be stored in the latent space. So if we use two correlating features, the crossing information would be stored twice, so some important information may not be represented in the latent space. Ideally, VAE should work like PCA or ICA methods to keep most of the galaxy data in the latent space.

It brought me to the idea that one can not aggravate learning possibilities of the VAE working with pictures, but instead build two additional Machine Learning models that would transform latent space to representable features and representable features to the latent space correspondingly. For example, one set Hubble-Vaucouler's type, sizes, and angles, then the model supplements this data with the other random generated plausible features, transforms it to latent space representation, and get the corresponding image using the VAE's decoder.

References

- [COSMOS] https://zenodo.org/record/3242143.YBZy_XYzY5k
- [F814W] https://www.stsci.edu/itt/APT_help/WFC3/appendixA42.html
- [Chianese et al. 2020] Chianese et al. 2020. Differentiable Strong Lensing: Uniting Gravity and Neural Nets through Differentiable Probabilistic Programming <https://arxiv.org/abs/1910.06157>
- [Tensorflow VAE] <https://www.tensorflow.org/tutorials/generative/autoencoder>
- [Tensorflow CVAE] <https://www.tensorflow.org/tutorials/generative/cvae>
- [He et al. 2015] He et al. 2015, Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, <https://arxiv.org/abs/1502.01852>
- [Glorot & Bengio 2010] Glorot & Bengio, AISTATS 2010, Understanding the difficulty of training deep feedforward neural networks <http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf>
- [Ioffe & Szegedy 2015] Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. <https://arxiv.org/abs/1502.03167>
- [Kingma & Ba 2015] Adam: A Method for Stochastic Optimization <https://arxiv.org/abs/1412.6980>
- [Lanusse et al. 2020] Deep Generative Models for Galaxy Image Simulations <https://arxiv.org/abs/2008.03833>
- [Arcelin et al. 2020] Deblending galaxies with Variational Autoencoders: a joint multi-band, multi-instrument approach <https://arxiv.org/abs/2005.12039>
- [Higgins et al. 2017] Higgins I., Matthey L., Pal A., Burgess C., Glorot X., Botvinick M., Mohamed S., Lerchner A., 2017, in ICLR.

9. Appendix

9.1. The latent space

The latent space represents a cut of the most important information of our galaxy. Since the image is normalized to $[0,1]$ the value $1/SNR$ represents necessarily uncertainty of the galaxy's image, hence it is sensible to expect that uncertainties of galaxy parameters would also be approximately $1/SNR$. Since we sample them from normal distribution, namely $\mu + N(0,1) \times \sigma$, the uncertainty is essentially the σ . Therefore the reasonable outcome would be σ converging to $1/SNR$ values, however, the SNR is not the same for all the images, but σ credibly should not converge to a value higher than one, referring to the lowest SNR. It will mark the case that the model treats as uncertainty not the noise of the image, but the structural properties of the galaxy itself. Though too low sigma is also irrelevant since it results in clusterization in latent space. The first problem of clusterization is its irrelevance in image generation, namely, it doesn't allow to generate relevant images from latent variables lying out of the cluster. The second problem is that the clusterization, strictly speaking, is not completely relevant in the physical sense. The most obvious clusterization option is based on Hubble-Vaucouleurs galaxy morphological classification. However, even though the galaxy classes are quite distinct, naturally, they only represent some transition points from the continuous transformation of one galaxy to another.

9.2. Original, Beta-, Flow- VAE's latent space

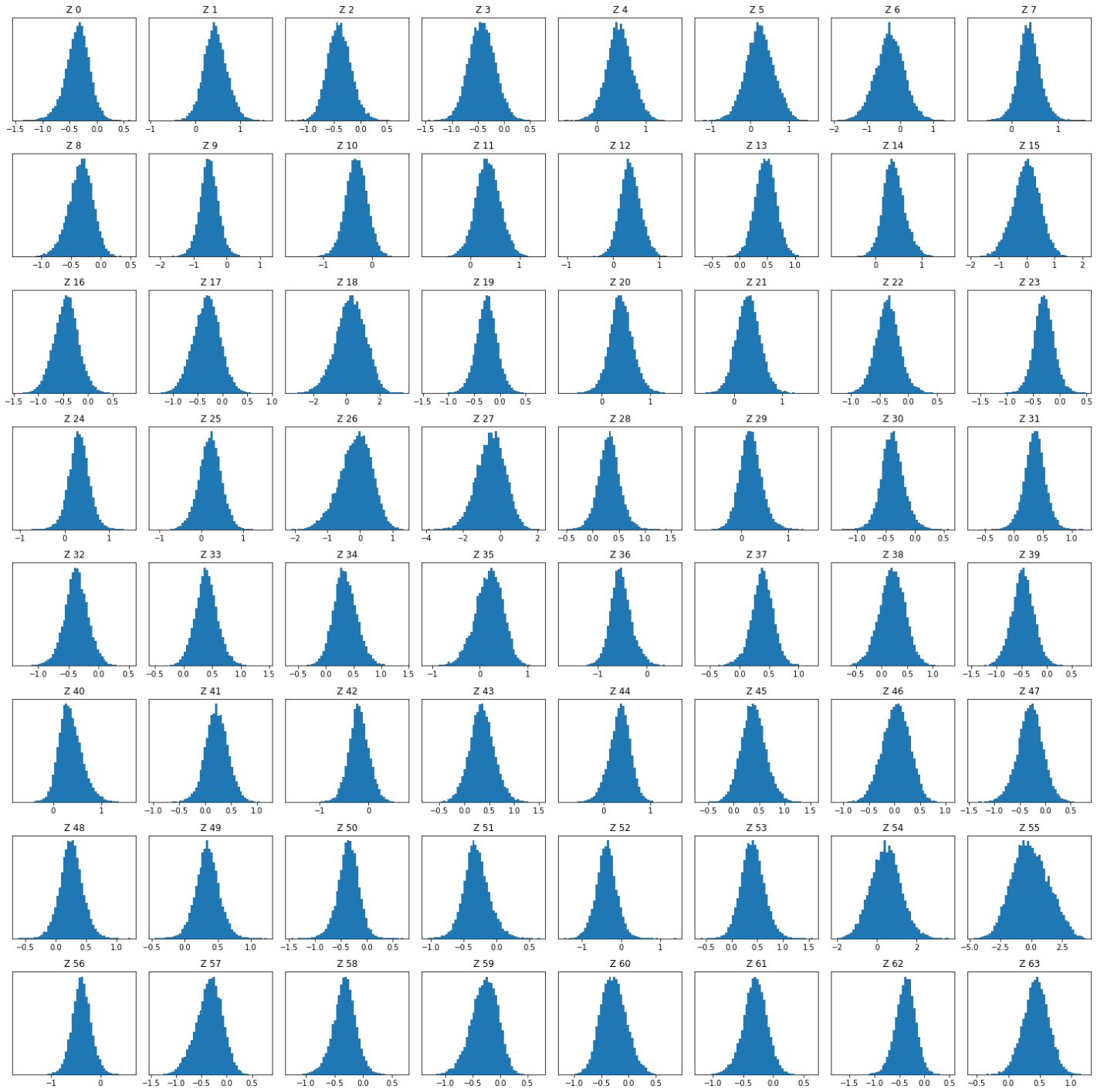


Fig. 13. Distribution of latent μ for original VAE, trained with loss given by function eq. 5. One can see that each of 64 distributions is a gaussian with variance ≈ 1

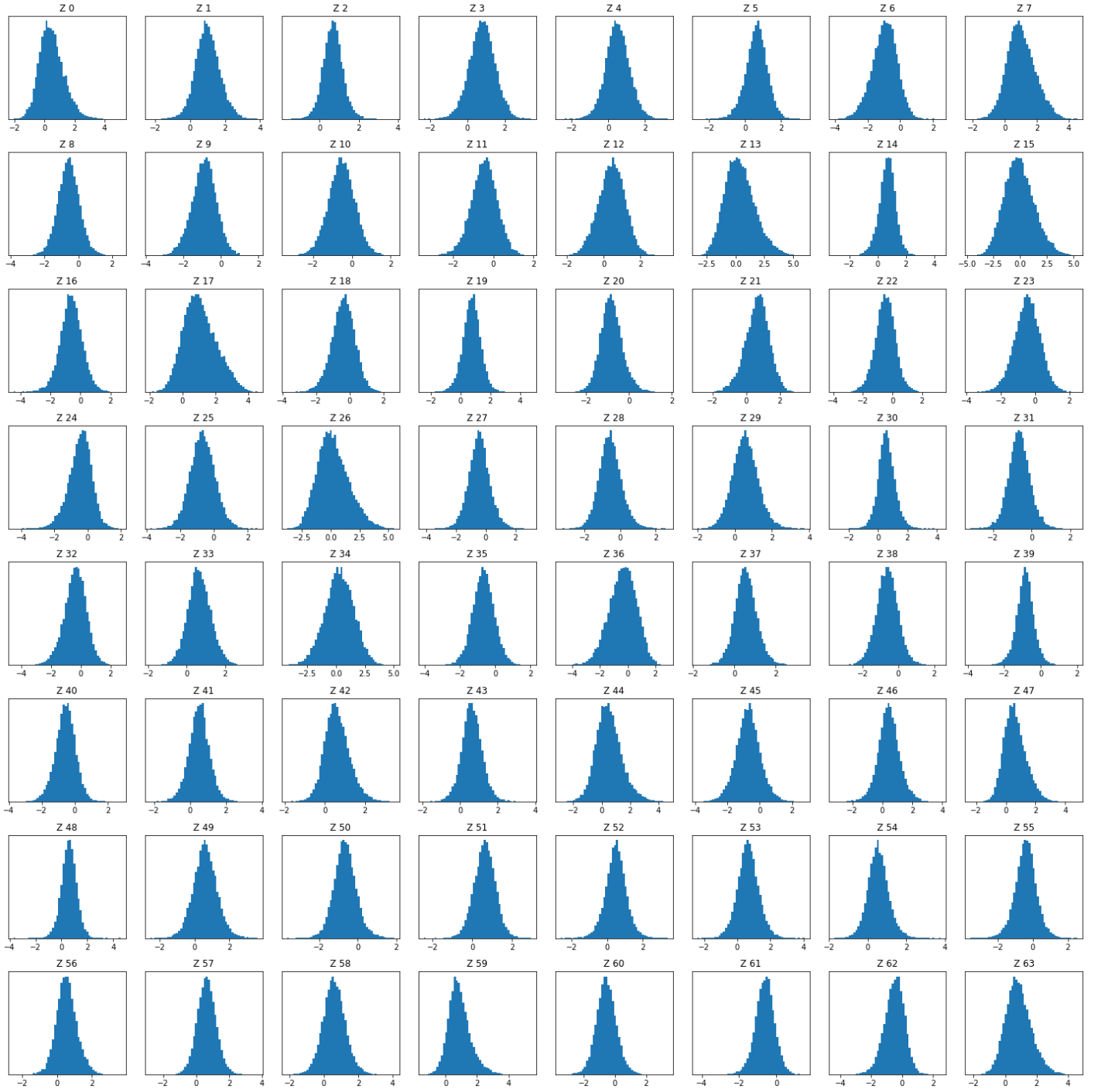


Fig. 14. Distribution of latent μ for Beta-VAE, trained with loss given by function eq. 7. One can see that each of 64 distributions is a gaussian with variance ≈ 1

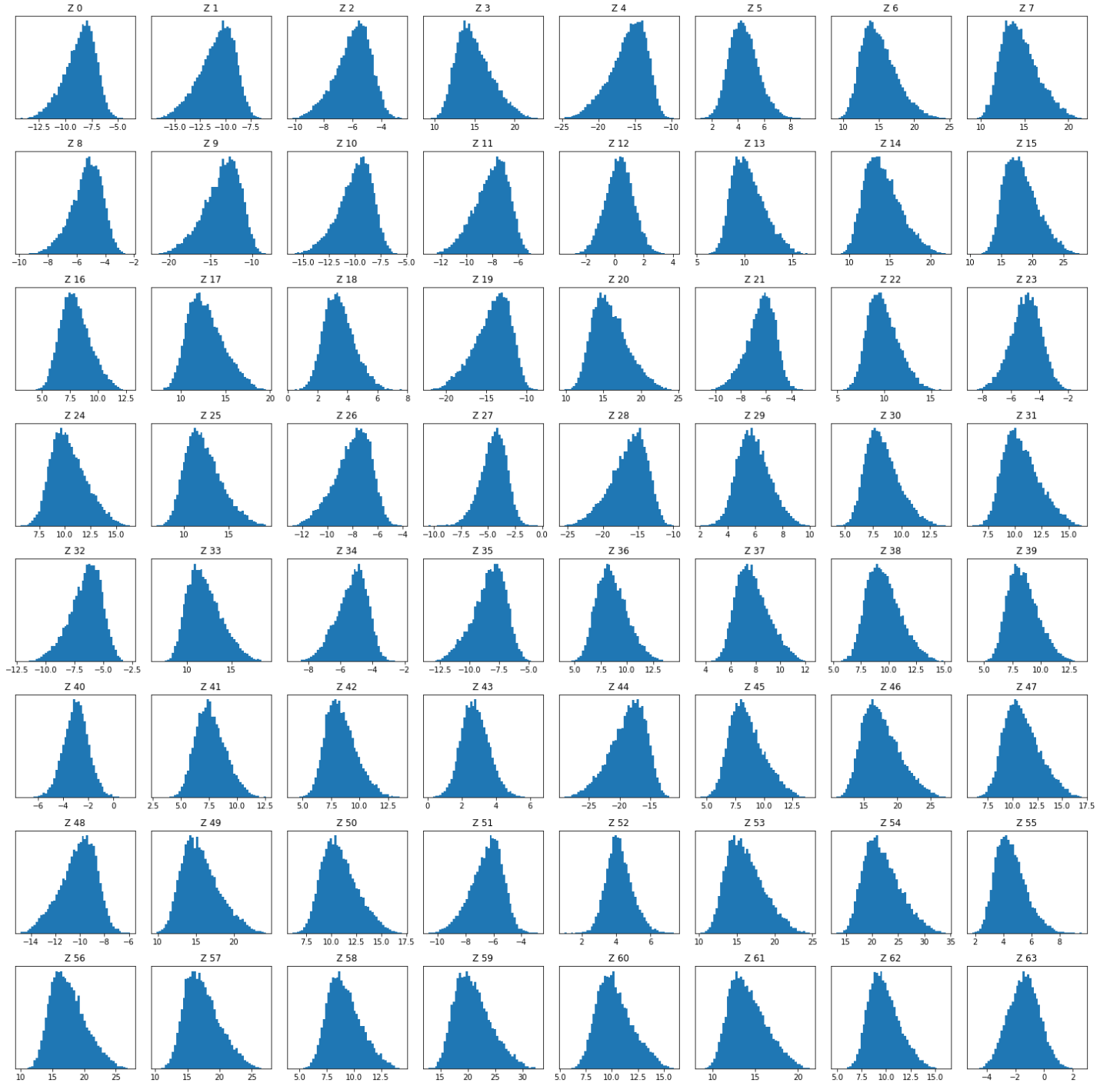


Fig. 15. Distribution of latent μ for Flow-VAE, trained with loss given by function eq. 8 and $\lambda = 4$. One can see that most of 64 distributions are tilted.