Dynamic Latent Scale for GAN Inversion

Jeongik Cho[a] and Adam Krzyzak[b]
Department of Computer Science and Software Engineering, Concordia University, Montreal, Quebec, Canada

Keywords: Generative Adversarial Network, Feature Learning, Representation Learning, GAN Inversion.

Abstract: When the latent random variable of GAN is an i.i.d. random variable, the encoder trained with mean squared error loss to invert the generator does not converge because the generator loses the information of the latent random variable. In this paper, we introduce a dynamic latent scale GAN, a method for training a generator that does not lose the information of the latent random variable, and an encoder that inverts the generator. Dynamic latent scale GAN dynamically scales each element of the latent random variable during GAN training to adjust the entropy of the latent random variable. As training progresses, the entropy of the latent random variable decreases until the generator does not lose the information of the latent random variable, which enables the encoder trained with squared error loss to converge. The scale of the latent random variable is approximated by tracing the element-wise variance of the predicted latent random variable from previous training steps. Since the scale of latent random variable changes dynamically, the encoder should be trained with the generator during GAN training. The encoder can be integrated with the discriminator, and the loss for the encoder is added to the generator loss for fast training.

1 INTRODUCTION

The generator of generative adversarial networks (GAN) (Goodfellow et al., 2014) is trained to map the latent random variable to the data random variable. Generally, independent and identically distributed (i.i.d.) random variable following simple distribution such as normal or uniform distribution is used as a latent random variable.

Inverting generator is finding an inverse mapping of a generator of GAN. It can be used for feature learning (or representation learning) or various useful applications such as data manipulation.

There are learning-based methods, optimization-based methods, and hybrid methods for GAN inversion. Many methods and applications of GAN inversion are introduced in the GAN inversion survey paper (Xia et al., 2021).

Among the learning-based methods, ALI (Dumoulin et al., 2017), AFL (Donahue et al., 2017), and BigBiGAN (Donahue et al., 2019) used cGAN (Mirza et al., 2014) to train an encoder that inverts the generator. However, those methods are difficult to train model, and the performance is not good.

InfoGAN (Chen et al., 2016), ICGAN (Perarnau et al., 2016), and Controllable GAN (Zhuang et al., 2021) used mean squared error (MSE) loss to train the encoder to recover the latent random variable. Assuming that the encoder is a gaussian model, training the encoder with MSE loss is a maximum likelihood estimation of the encoder (minimize negative log-likelihood). In-Domain GAN Inversion (Zhu et al., 2020), and AGEN (Ulyanov et al., 2018) added reconstruction loss to MSE loss for better performance. StyleMapGAN (Kim et al., 2021), Collaborative Learning for Faster StyleGAN Embedding (Guan et al., 2020), and Encoding in Style (Richardson et al., 2021) proposed model (StyleGAN (Karras et al., 2019) and StyleGAN2 (Karras et al., 2020)) specific methods.

However, using MSE loss to train the encoder results in a convergence problem because the generator may lose the information of the latent random variable. In other words, it is impossible to train an encoder that inverts the generator trained with the latent random variable as is because the generator may ignore some information of the latent random variable.

[a] https://orcid.org/0000-0001-5396-2375
[b] https://orcid.org/0000-0003-0766-2659

Cho, J. and Krzyzak, A.
Dynamic Latent Scale for GAN Inversion.
DOI: 10.5220/0010816800003122
In *Proceedings of the 11th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2022)*, pages 221-228
ISBN: 978-989-758-549-4; ISSN: 2184-4313

In this paper, we introduce a dynamic latent scale GAN (DLSGAN), a learning-based method for training an encoder that inverts the generator of GAN. DLSGAN dynamically adjusts the scale of the latent random variable so that the generator does not lose the information of the latent random variable. This enables the encoder to converge when training the encoder with squared error loss (maximum likelihood estimation).

The scale of the latent random variable depends on the amount of information that the encoder can recover. It can be approximated from the element-wise variance of the predicted latent random variable from the encoder. DLSGAN traces the predicted latent codes of past training steps to approximate the element-wise variance of the predicted latent random variable.

In DLSGAN, since the scale of a latent random variable dynamically changes, the encoder should be trained with a generator during GAN training. Furthermore, the encoder can be integrated with the discriminator for efficient training. Also, training the encoder can be accelerated by adding an encoder loss to generator loss. It means that the encoder and generator are trained cooperatively to minimize the encoder loss. This is possible because the encoder is trained during the GAN training.

Full codes of our work are available at "https://github.com/jeongik-jo/DLSGAN".

2 PROBLEM STATEMENT

Assume that generator G maps the latent random variable Z to the data random variable X (i.e., $X = G(Z)$). Our goal is to train an encoder E that inverts the generator G (i.e., $Z = E\big(G(Z)\big)$).

When the latent random variable Z is a d_z-dimensional i.i.d. random variable, the encoder E can be considered as an integration of d_z encoders, where each encoder is trained to recover each element of a latent random variable Z (i.e., $Z_1 = E_1\big(G(Z)\big), Z_2 = E_2\big(G(Z)\big), \dots, Z_{d_z} = E_{d_z}\big(G(Z)\big)$). Assuming each encoder is a gaussian model, training each encoder with an MSE loss minimizes negative log-likelihood of each encoder.

However, the integrated encoder E cannot fully recover the latent random variable Z because there is no guarantee that the generator G uses all the information of the latent random variable Z. For example, when the latent random variable Z has too many dimensions, generator G can be trained to ignore some elements of the latent random variable Z. Or, generator G can be trained so that some elements of the latent random variable Z have relatively more information than others. In other words, different latent codes a and b sampled from the latent random variable Z can be mapped to the same or similar generated data points $G(a)$ and $G(b)$. Therefore, some encoders of the integrated encoder E cannot converge to predict some element of the latent random variable Z. It means that the generator loses the information of the latent random variable Z, and the encoder E cannot perfectly recover the latent random variable Z from the generated data random variable $G(Z)$.

3 DYNAMIC LATENT SCALE GAN

To prevent the generator G from losing information of the latent random variable Z, we introduce a DLSGAN that dynamically adjusts the scale of each element of the latent random variable Z.

Assume that the latent random variable Z is d_z-dimensional i.i.d. random variable with the variance σ^2. When the encoder E is trained long enough to predict the latent random variable Z from the generated data random variable $G(Z)$ with MSE loss, the variance of each element of the predicted latent random variable $Z' = E\big(G(Z)\big)$ represents information of the latent random variable Z that can be recovered from the generated data random variable $G(Z)$. If the variance of n-th predicted latent random variable Z'_n is zero, it means that the encoder E cannot recover any information of n-th latent random variable Z_n from the generated data random variable $G(Z)$. On the other hand, if the variance of the n-th predicted latent random variable Z'_n is σ^2, then the encoder E can recover all information of n-th latent random variable Z_n from the generated data random variable $G(Z)$. Therefore, if the element-wise variance of the predicted latent random variable Z' and the element-wise variance of the latent random variable Z are the same, it means that the generator G does not lose the information of the latent random variable Z, and the encoder can converge to predict the latent random variable Z from the generated data random variable $G(Z)$.

DLSGAN dynamically adjusts the scale of each element of latent random variable Z according to the variance of each element of the predicted latent random variable Z' so that the element-wise variance of the latent random variable Z and predicted latent random variable Z' are equal. Since the dynamic latent scale GAN requires both the encoder E and the generator G to be trained together during GAN training, it is efficient to integrate the encoder E into the discriminator D. For the same reason, the generator G and the encoder E can be trained cooperatively. That is, encoder loss L_{enc} can be added to generator loss L_g.

The following algorithm shows the process of obtaining the loss for training DLSGAN.

Algorithm 1: Obtaining loss for training DLSGAN.

```
function GetLoss(D,G,Z,X,v):
1    $z \leftarrow sample(Z)$
2    $x \leftarrow sample(X)$

3    $s \leftarrow \frac{\sqrt{d_z} v^{\circ 1/2}}{\|v^{\circ 1/2}\|_2}$
4    $a_g, z' \leftarrow D(G(z \circ s))$
5    $L_{enc} \leftarrow avg((z-z')^{\circ 2} \circ s^{\circ 2})$
6    $a_r, \_ \leftarrow D(x)$

7    $L_d \leftarrow f_d(a_r, a_g) + \lambda_{enc} L_{enc}$
8    $L_g \leftarrow f_g(a_g) + \lambda_{enc} L_{enc}$

9    $v \leftarrow update(v, z'^{\circ 2})$

10   return $L_d, L_g, v$
```

In Algorithm 1, D, G, Z, and X represent discriminator, generator, latent random variable, and data random variable, respectively. Since the encoder E is integrated with the discriminator D, the discriminator D outputs two values: 1-dimensional adversarial value and d_z-dimensional predicted latent code. v represents the element-wise variance of the predicted latent random variable Z'. It is ideal to approximate the predicted latent variance vector v for every training step, but for efficiency, the predicted latent variance vector v is approximated through predicted latent codes from the past training steps.

In lines 1 and 2, Z is a d_z-dimensional i.i.d. latent random variable, and X is a data random variable. In Algorithm 1, it is assumed that latent random variable Z follows a distribution with a mean of 0 and a variance of 1 for convenience. $sample$ is a function that samples a single sample from a random variable. z represents a latent code, which is sampled from the latent random variable Z. x represents a data point, which is sampled from the data random variable X.

In lines 3 and 4, s is the latent scale vector. $vec^{\circ 1/2}$ represents the element-wise square root of the example vector vec. $\|vec\|_2$ represents the L2 norm of example vector vec. "$\circ$" represents element-wise multiplication. $G(z \circ s)$ is the generated data point with scaled latent code $z \circ s$. In line 4, a_g represents the adversarial value of generated data, and z' represents the predicted latent code, respectively. When all elements of v are the same, i.e., when the variance of all elements of predicted latent random variable Z' are the same, the scaled latent random variable $Z \circ s$ has the largest differential entropy. On the other hand, when the variance of only one element of the predicted latent random variable is not 0, and the other elements are 0, the scaled latent random variable $Z \circ s$ has the least differential entropy. $\sqrt{d_z}$ is a constant multiplied to make the scaled latent random variable $Z \circ s$ equal to the latent random variable Z when the differential entropy of the scaled latent random variable $Z \circ s$ is the largest. The differential entropy of the scaled latent random variable $Z \circ s$ dynamically changes according to the variance of the predicted latent random variable Z' during GAN training. As GAN training progresses, the scaled latent random variable $Z \circ s$ converges to have an optimal entropy representing the real data random variable X through the generator G.

In line 5, $vec^{\circ 2}$ represents the element-wise square of the example vector vec. avg is a function that calculates the average of a vector. L_{enc} is encoder loss. The encoder loss L_{enc} is equal to the MSE loss between the scaled latent code $z \circ s$ and the scaled predicted latent code $z' \circ s$.

In line 6, a_r represents the adversarial value of a real data point x. "_" represents not using value. Since the latent code of the real data point x is unknown, the predicted latent code for the real data point x is not used in DLSGAN training.

In lines 7 and 8, f_d and f_g are adversarial loss functions for discriminator D and generator G, respectively. One can find many adversarial losses in GAN adversarial losses compare paper (Lucic et al., 2018). λ_{enc} is encoder loss weight. One can see encoder loss L_{enc} is added to both generator loss L_g and discriminator loss L_d. This means that the generator G and discriminator D are trained cooperatively to reduce the encoder loss L_{enc}. Training the encoder E during the GAN training enables to add encoder loss L_{enc} to generator loss L_g.

In line 9, $update$ function updates the predicted latent random variable variance v with the new predicted latent code z'. Since the mean of the predicted latent random variable Z' becomes automatically zero, $z'^{\circ 2}$ (element-wise square of predicted latent code z') can be considered as the sample variance of the predicted latent random variable Z'. A moving average or an exponential moving average can be used for the $update$ function.

Note that the latent code input to the generator should always be scaled by the scale vector s. Therefore, the generated data point is $G(z \circ s)$, and the recovered data point of x is $G(z_x \circ s)$, where z_x is the predicted latent code of the real data point x.

DLSGAN is still the maximum likelihood estimation of the encoder (minimize negative log-likelihood), but the generator G does not lose the information of the latent random variable Z, which allows the encoder E to converge when training the encoder E with squared error loss.

4 EXPERIMENT RESULTS AND DISCUSSION

4.1 Experiment Settings

We trained GAN to generate the CelebA dataset (Liu et al., 2015) resized to 128×128 resolution. As the model architecture, StyleGAN2 with a reduced filter size of convolution layers was used.

Batch operation (minibatch stddev layer) in the discriminator and noise in the generator are removed so that the encoder encodes one data point as one latent code. As an adversarial loss, NSGAN with R1 regularization (Mescheder et al., 2018) was used as StyleGAN2. The hyperparameters used for the experiments are as follows. Most hyperparameters are the same or similar as StyleGAN2.

$$\lambda_{enc} = 1$$
$$d_z = 512$$
$$optimizer = Adam \begin{pmatrix} learning\ rate = 0.001 \\ beta_1 = 0 \\ beta_2 = 0.99 \end{pmatrix}$$
$$learning\ rate\ decay\ per\ epoch = 2\%$$
$$\lambda_{r1} = 10$$
$$batch\ size = 16$$
$$epoch = 50$$

Note that optimizer for the mapper of generator has $\times\, 0.01$ learning rate as same as StyleGAN2. λ_{r1} is R1 regularization weight. The original paper introduced R1 regularization used $\gamma/2$ as regularization weight, so based on that definition, γ is 20 when λ_{r1} is 10.

We compared the performance of the model with and without dynamic latent scale. Without dynamic latent scale means MSE loss was used for encoder training. We also compared the effect of the encoder loss L_{enc} on generator loss L_g. For the $update$ function for dynamic latent scale, we used a moving average using the past $512 \times batch\ size$ samples.

The experiments were conducted for i.i.d. latent random variables with distributions $N(0,1^2)$ and $U(-\sqrt{3},\sqrt{3})$ so that the mean and the variance of the distribution are 0 and 1, respectively.

4.2 Experiment Results

The following figures present the performance results with and without dynamic latent scale and encoder loss L_{enc} on generator loss L_g or not.

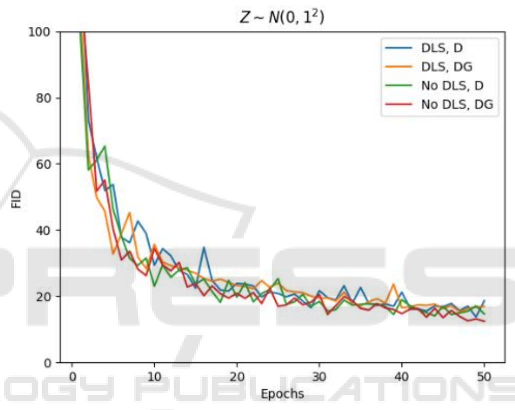

Figure 1: FID comparison when $Z = (Z_i)_{i=1}^{d_z} \overset{i.i.d.}{\sim} N(0,1^2)$.

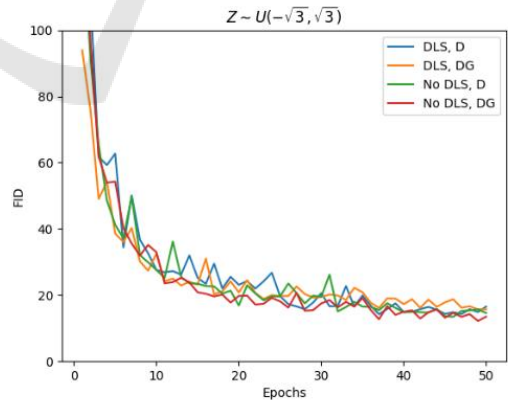

Figure 2: FID comparison when $Z = (Z_i)_{i=1}^{d_z} \overset{i.i.d.}{\sim} U(-\sqrt{3},\sqrt{3})$.

Figures 1 and 2 show the FID (Heusel et al., 2017) according to the training methods for each epoch. In figures 1 and 2, “DLS” and “No DLS” represent with dynamic latent scale and without dynamic latent scale, respectively. “D” represents weighted encoder loss $\lambda_{enc} L_{enc}$ was added to only discriminator loss L_d, and “DG” represents weighted encoder loss $\lambda_{enc} L_{enc}$ was added to both discriminator loss L_d and generator loss L_g. “No DLS, D” corresponds to previous learning-based methods that do not use a dynamic latent scale for GAN inversion (e.g., ICGAN, Controllable GAN).

One can see that there is little difference in the generative performance for each training method.

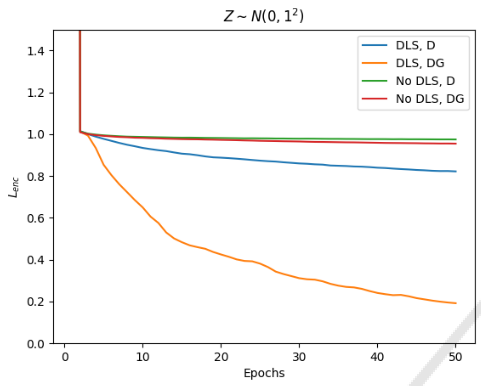

Figure 3: Average L_{enc} when $Z = (Z_i)_{i=1}^{d_z} \overset{i.i.d.}{\sim} N(0,1^2)$.

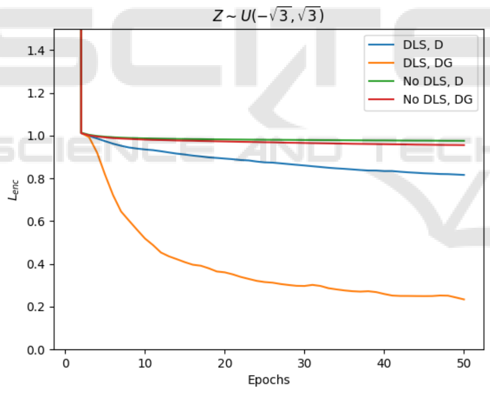

Figure 4: Average L_{enc} when $Z = (Z_i)_{i=1}^{d_z} \overset{i.i.d.}{\sim} U(-\sqrt{3}, \sqrt{3})$.

Figures 3 and 4 show the average encoder loss L_{enc} according to the training methods for each epoch. One can see that without dynamic latent scale, encoder loss L_{enc} hardly changes from 1. This shows that encoder E trained without dynamic latent scale fails to converge because generator G loses information of latent random variable Z. On the other hand, one can see that the encoder loss L_{enc} continuously decreases as training progresses with dynamic latent scale. This shows that the model converges with the dynamic latent scale. Also, one can see that the encoder loss L_{enc} is much lower when encoder loss L_{enc} is added to the generator loss L_g.

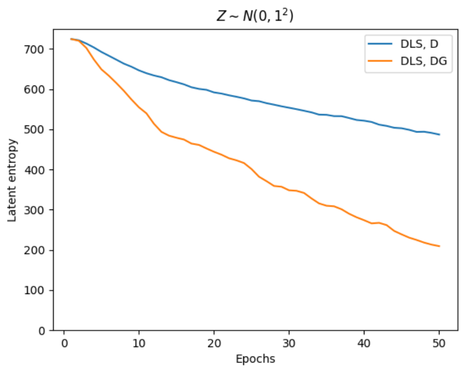

Figure 5: Differential latent entropy of DLSGAN when $Z = (Z_i)_{i=1}^{d_z} \overset{i.i.d.}{\sim} N(0,1^2)$.

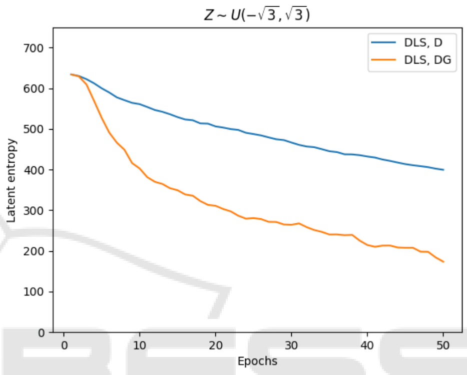

Figure 6: Differential latent entropy of DLSGAN when $Z = (Z_i)_{i=1}^{d_z} \overset{i.i.d.}{\sim} U(-\sqrt{3}, \sqrt{3})$.

Figures 5 and 6 show differential entropy of scaled latent random variable $Z \circ s$ with dynamic latent scale for each epoch. Like encoder loss L_{enc}, one can see that the differential entropy of the scaled latent random variable $Z \circ s$ decreases faster when encoder loss L_{enc} is added to the generator loss L_g. Note that differential entropy can be negative.

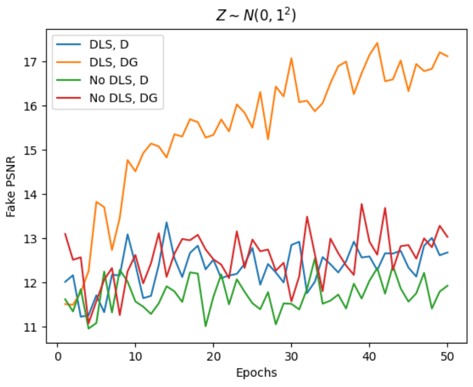

Figure 7: Average PSNR for generated images reconstruction when $Z = (Z_i)_{i=1}^{d_z} \overset{i.i.d.}{\sim} N(0,1^2)$.

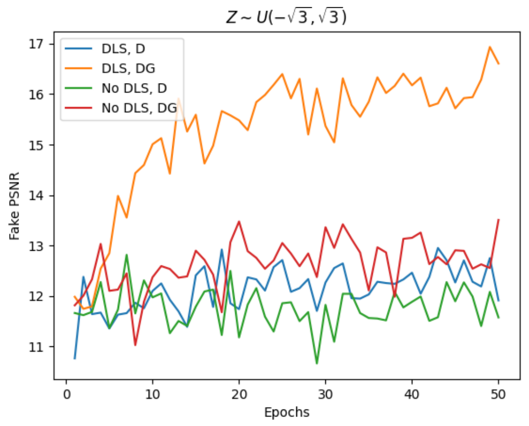

Figure 8: Average PSNR for generated images reconstruction when $Z = (Z_i)_{i=1}^{d_z} \overset{i.i.d.}{\sim} U(-\sqrt{3}, \sqrt{3})$.

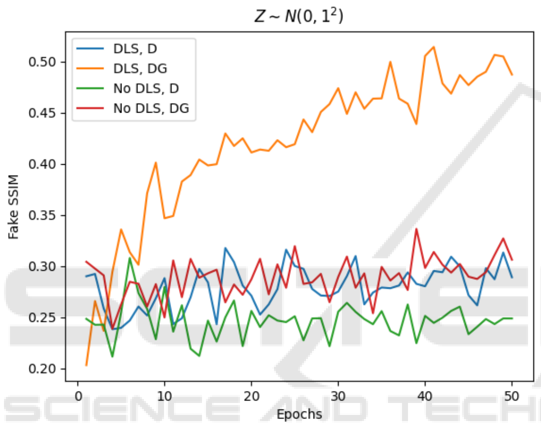

Figure 9: Average SSIM for generated images reconstruction when $Z = (Z_i)_{i=1}^{d_z} \overset{i.i.d.}{\sim} N(0, 1^2)$.

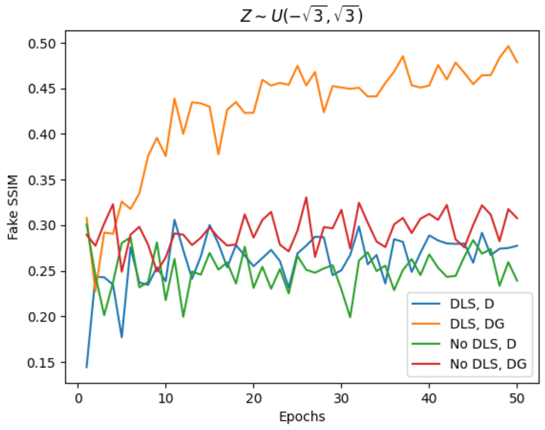

Figure 10: Average SSIM for generated images reconstruction when $Z = (Z_i)_{i=1}^{d_z} \overset{i.i.d.}{\sim} U(-\sqrt{3}, \sqrt{3})$.

Figures 7-10 show the average PSNR and SSIM for each epoch when reconstruction is performed on the generated images. The higher the PSNR and SSIM, the better the image reconstruction performance. The PSNR ranges from zero to infinity, and the SSIM ranges from zero to one. One can see that the performance of reconstruction on generated images is much better with DLS, DG. Also, both with dynamic latent scale and without dynamic latent scale performed better when the encoder loss L_{enc} is added to the generator loss L_g.

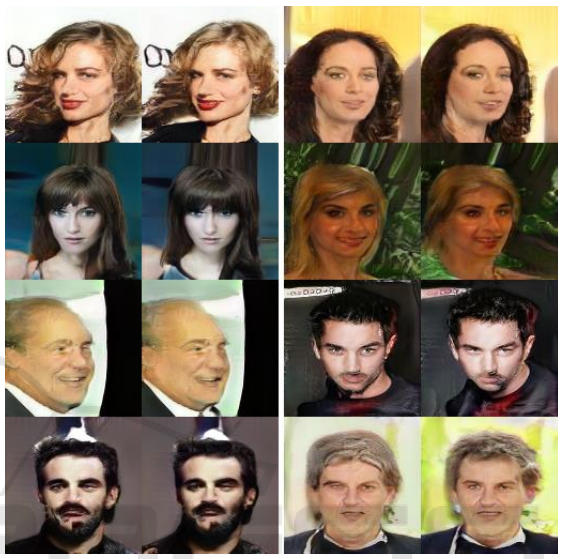

Figure 11: DLS, DG generated images reconstruction examples when $Z = (Z_i)_{i=1}^{d_z} \overset{i.i.d.}{\sim} N(0, 1^2)$. Left: generated image, right: reconstructed image in each image pair.

Figures 11 shows examples of generated images reconstruction, with dynamic latent scale and encoder loss L_{enc} on generator loss L_g when $Z = (Z_i)_{i=1}^{d_z} \overset{i.i.d.}{\sim} N(0, 1^2)$.

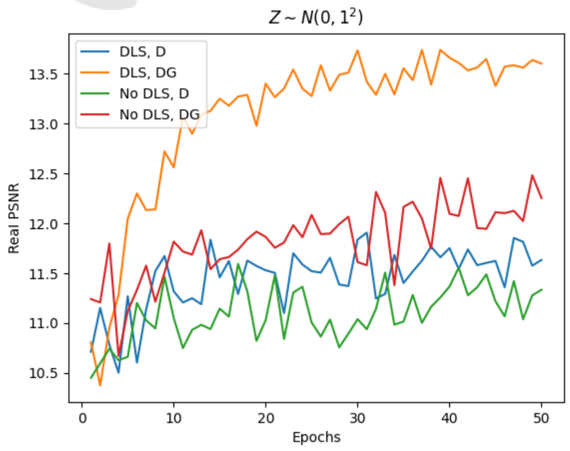

Figure 12: Average PSNR for test images reconstruction when $Z = (Z_i)_{i=1}^{d_z} \overset{i.i.d.}{\sim} N(0, 1^2)$.

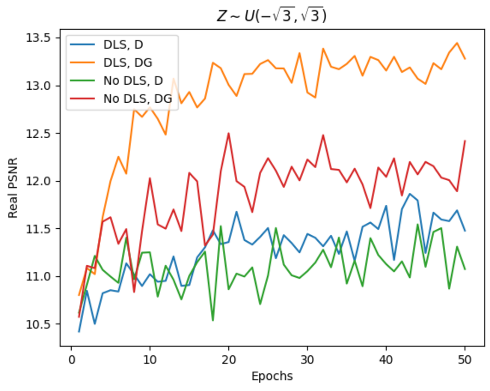

Figure 13: Average PSNR for test images reconstruction when $Z = (Z_i)_{i=1}^{d_z} \overset{i.i.d.}{\sim} U(-\sqrt{3}, \sqrt{3})$.

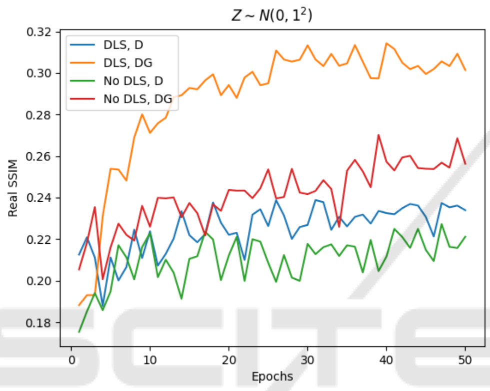

Figure 14: Average SSIM for test images reconstruction when $Z = (Z_i)_{i=1}^{d_z} \overset{i.i.d.}{\sim} N(0,1^2)$.

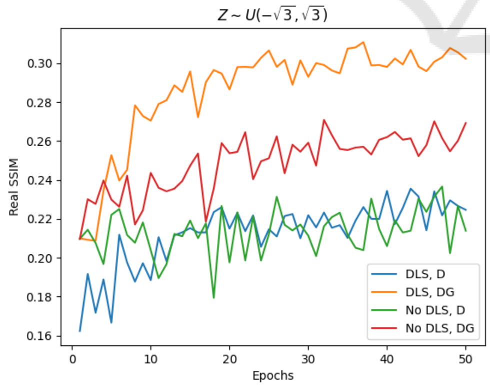

Figure 15: Average SSIM for test images reconstruction when $Z = (Z_i)_{i=1}^{d_z} \overset{i.i.d.}{\sim} U(-\sqrt{3}, \sqrt{3})$.

Figures 12-15 show the average PSNR and SSIM for each epoch when reconstruction is performed on the test images (real images). One can notice that reconstruction performance on test images is much better with DLS, DG.

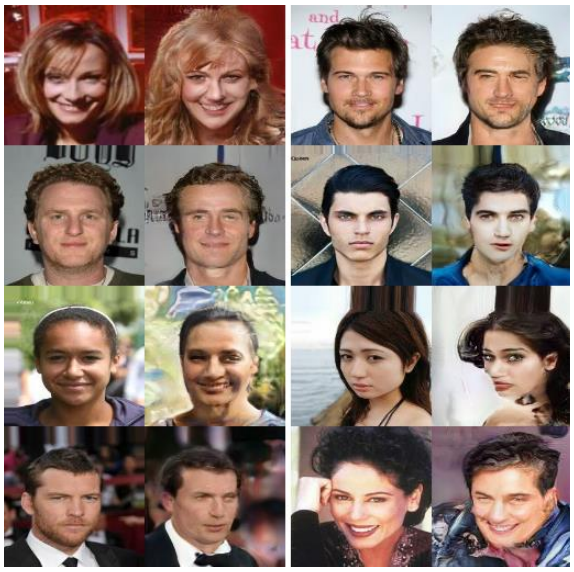

Figure 16: DLS, DG test images reconstruction examples when $Z = (Z_i)_{i=1}^{d_z} \overset{i.i.d.}{\sim} N(0,1^2)$. Left: test image, right: reconstructed image in each image pair.

Figures 16 shows examples of generated images reconstruction with dynamic latent scale and encoder loss L_{enc} on generator loss L_g when $Z = (Z_i)_{i=1}^{d_z} \overset{i.i.d.}{\sim} N(0,1^2)$.

4.3 Latent Interpolation of DLSGAN

The following figures show some additional results with DLS, DG, and $Z = (Z_i)_{i=1}^{d_z} \overset{i.i.d.}{\sim} N(0,1^2)$.

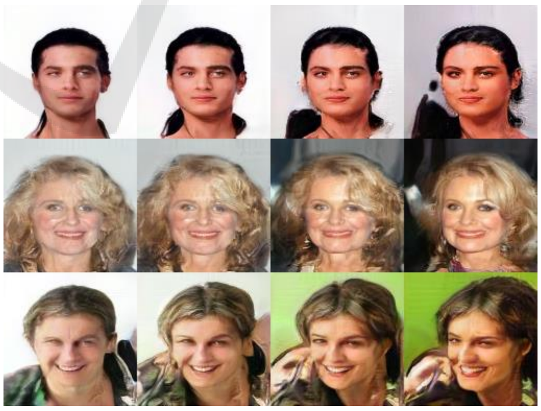

Figure 17: Latent interpolation on most important element.

Figures 17 shows interpolating one important element of the latent random variable Z from -2 to 2 with DLS, DG, and $Z = (Z_i)_{i=1}^{d_z} \overset{i.i.d.}{\sim} N(0,1^2)$. The larger the elements of the scale vector s, the more important (more informative) elements.

5 CONCLUSIONS

In this paper, we proposed a DLSGAN, a method for training a generator that does not lose the information of the latent random variable, and an encoder that inverts the generator. Dynamic latent scale GAN dynamically adjusts the scale of the i.i.d. latent random variable to have the optimal entropy to express the data random variable. This ensures that the generator does not lose the information of the latent random variable so that the encoder can converge to invert the generator with maximum likelihood estimation. The encoder of DLSGAN showed much better performance than without dynamic latent scale.

REFERENCES

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*.

Xia, W., Zhang, Y., Yang, Y., Xue, J. H., Zhou, B., and Yang, M. H. (2021). GAN Inversion: A Survey. *arXiv preprint arXiv:2101.05278*.

Dumoulin, V., Belghazi, I., Poole, B., Lamb, A., Arjovsky, M., Mastropietro, O., and Courville, A. (2017). Adversarially Learned Inference. In *International Conference on Learning Representations 2017*.

Donahue, J., Krähenbühl, P., and Darrell, T. (2017). Adversarial Feature Learning. In *International Conference on Learning Representations 2017*.

Donahue, J., and Simonyan, K. (2019). Large Scale Adversarial Representation Learning. In *International Conference on Learning Representations 2019*.

Mirza, M., Osindero, S. (2014). Conditional Generative Adversarial Nets. *arXiv preprint arXiv:1411.1784*.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 29*.

Perarnau, G., Weijer, J. V. D., Raducanu, B., and Álvarez, J. M. (2016). Invertible Conditional GANs for image editing. *arXiv preprint arXiv:1611.06355*.

Zhuang, P., Koyejo, O. O., and Schwing, A. (2021). Enjoy Your Editing: Controllable GANs for Image Editing via Latent Space Navigation. In *International Conference on Learning Representations 2021*.

Zhu, J., Shen, Y., Zhao, D., and Zhou, B. (2020). In-domain GAN Inversion for Real Image Editing. In *Proceedings of European Conference on Computer Vision*.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2018). It Takes (Only) Two: Adversarial Generator-Encoder Networks. *Proceedings of the AAAI Conference on Artificial Intelligence*.

Kim, H., Choi, Y., Kim, J., Yoo, S., and Uh, Y. (2021). Exploiting Spatial Dimensions of Latent in GAN for Real-Time Image Editing. *Conference on Computer Vision and Pattern Recognition*, pages 852-861.

Guan, S., Tai, Y., Ni, B., Zhu, F., Huang, F., Yang, X. (2020). Collaborative Learning for Faster StyleGAN Embedding. *arXiv preprint arXiv:2007.01758*.

Richardson, E., Alaluf, Y., Patashnik, O., Nitzan, Y., Azar, Y., Shapiro, S., and Cohen-Or, D. (2021). Encoding in Style: A StyleGAN Encoder for Image-to-Image Translation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2287-2296.

Karras T., Laine, S., and Aila, T. (2019). A Style-Based Generator Architecture for Generative Adversarial Networks. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4396-4405.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. (2020). Analyzing and Improving the Image Quality of StyleGAN. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8107-8116.

Lucic, M., Kurach, K., Michalski, M., Gelly, S., and Bousquet, O. (2018). Are GANs Created Equal? A Large-Scale Study. *Advances in Neural Information Processing Systems 31*.

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep Learning Face Attributes in the Wild. *Proceedings of International Conference on Computer Vision*, pages 3730-3738.

Mescheder, M., Nowozin, S., and Geiger, A. (2018). Which Training Methods for GANs do actually Converge? *Proceedings of the 35th International Conference on Machine Learning*, pages 3481-3490.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. *Proceedings of the 31st International Conference on Neural Information Processing System*.