Dynamic Latent Scale GAN

Jeongik Cho

jeongik.jo.01@gmail.com

Abstract

Generators in generative adversarial networks map latent distributions into data distributions. GAN inversion is mapping data distribution to latent distribution by inverting the generator of GAN.

When training the encoder for generator inversion, simply using the mean squared error causes the encoder to not converge due to information loss on the latent distribution from the generator. In other words, it is impossible to invert the generator as it is due to the information loss on the latent distribution.

This paper introduces a dynamic latent scale GAN, a method for training a generator without information loss on latent distribution, and an encoder that inverts the generator. Dynamic latent scale GAN dynamically scales each element of the normal i.i.d. (independent and identically distributed) latent distribution during GAN training to adjust the entropy of the latent distribution so that information loss on the latent distribution does not occur in the generator. The amount of information that can be recovered from the generated data distribution can be obtained through the variance of the predicted latent distribution (encoder output distribution). By dynamically adjusting the scale of the latent distribution through the variance of each element of the predicted latent distribution, it is possible to train a generator that does not have information loss on latent distribution. This means that mutual information between the latent distribution and predicted latent distribution can be maximized, and the encoder can converge.

Since the latent distribution scale of the dynamic latent scale GAN changes dynamically, the encoder should be trained together during GAN training. The encoder can be integrated with the discriminator, and the loss for the encoder can be added to the generator loss because the encoder converges.

1. Introduction

It is very useful to learn the inverse transform of the generator of GAN. It can perform the role of feature learning or can be used for various useful applications such as data manipulation. Many useful applications and methods for GAN inversion are introduced in the GAN inversion survey paper [1].

2. Problem analyze

The generator maps the latent distribution to the data distribution. However, there is no guarantee that the generator uses all the information of the latent distribution. For example, when the latent distribution has too many dimensions, the generator can be trained to ignore some elements of latent distribution. Or, the generator can be trained so that some elements of latent distribution have relatively more information than others. In other words, different latent vectors can be mapped to the same or nearly similar data points by the generator. It means that information loss on the latent distribution occurs in the generator.

Due to information loss by the generator, the encoder cannot perfectly recover the latent distribution from the generated data distribution. Therefore, training the encoder with mean squared error to recover unrecoverable latent distribution [6], [7] causes a convergence problem. When the encoder encodes the generated data distribution into a d_z -dimensional predicted latent distribution, this encoder can be considered as an integration of d_z encoders, each encoder encoding each dimension. Since all encoders share all layers except the output layer, if some encoders are trained to recover elements of latent distribution that are impossible to recover, it also prevents other encoders from convergence. This degrades the performance of the entire encoder. Likewise, when an encoder is integrated into a discriminator, it also degrades the performance of the discriminator.

3. Dynamic latent scale GAN

To prevent information loss on the latent distribution that occurs in the generator, I introduce a dynamic latent scale GAN that dynamically adjusts the scale of each element of the latent distribution.

Assume that the latent distribution Z follows d_z-dimensional i.i.d. normal random variable $N(\mu, \sigma^2)^{d_z}$. Suppose d_z number of independent one-dimensional encoders $(E_1, E_2, \dots, E_{d_z})$ are sufficiently trained to invert generator G with squared error. In that case, the output of each encoder (i.e., $E_1(G(Z)), E_2(G(Z)), \dots, E_{d_z}(G(Z))$) follows an independent normal distribution but has different variances less or equal to σ^2 with mean μ because some of the encoders cannot fully recover the information from generated data distribution $G(Z)$.

At this time, the variance of each trained encoder represents information that can be recovered from the latent distribution. Therefore, it is possible to train a generator without information loss by training the generator and encoder together with a scaled latent vector during the GAN training. If no information loss on latent distribution occurs in the generator, each encoder can converge. Therefore, each encoder can be integrated into one encoder that shares all hidden layers. Furthermore, the encoder can also be integrated with the discriminator because it converges. Since the dynamic latent scale GAN requires both the encoder and the generator to be trained together, it is efficient to integrate

the encoder into the discriminator. Also, because the encoder converges, the generator and encoder can be trained cooperatively. That is, encoder loss (objective function) can be added to generator loss (objective function).

The following algorithm shows the process of obtaining the loss (objective function) for training dynamic latent scale GAN.

$function\ GetLoss(D, G, x, v)$:

1 $z \leftarrow sample(N(0,1^2)^{d_z})$

2 $s \leftarrow \frac{\sqrt{d_z v}}{\|\sqrt{v}\|_2}$

3 $a_g, z' \leftarrow D\big(G(z \odot s)\big)$

4 $L_{enc} \leftarrow avg((z - z')^2 \odot s^2)$

5 $a_r, \_ \leftarrow D(x)$

6 $L_d \leftarrow f_r^d(a_r) + f_g^d(a_g) + \lambda_{enc} L_{enc}$

7 $L_g \leftarrow f_g(a_g) + \lambda_{enc} L_{enc}$

8 $v \leftarrow update\big(v, z'^2\big)$

9 $return\ L_d, L_g, v$

Algorithm 1. Obtaining loss (objective function) for training dynamic latent scale GAN

In the above algorithm, D, G, and x represent discriminator, generator, and real data, respectively. Real data x is sampled from real data distribution X. Since encoder E is integrated with discriminator D, discriminator D outputs two values: 1-dimensional adversarial value and d_z-dimensional predicted latent vector. v represents the element-wise variance of the predicted latent distribution Z'. It is ideal for calculating v for each training step, but for efficiency, in practice, v is calculated through samples of predicted latent vector z' obtained from the past k train steps.

In line 1, $N(0,1^2)^{d_z}$ is a d_z dimensional i.i.d. random variable following normal distribution $N(0,1^2)$. For convenience, a mean of 0 and a standard deviation of 1 were assumed. $sample$ is a function that samples a single value from a random variable.

In line 2, s is the latent scale vector. $\sqrt{vec}$ represents the element-wise square root of the example vector vec. $\|vec\|_2$ represents L2 norm of example vector vec.

In line 3, $\odot$ represents element-wise multiplication. $G(z \odot s)$ is the generated data point. The discriminator D outputs the adversarial value a_g and the predicted latent vector z'. The generator G receives scaled latent vector $z \odot s$ as input. When all elements of v are the same, i.e., when the distribution of all encoder outputs has the same variance, the scaled latent distribution has the largest amount of information. On the other hand, when the variance of only one element of the encoder output is not 0, and the variances of the other elements are 0, the scaled latent vector has the least information. $\sqrt{d_z}$ is a constant multiplied to make the latent distribution follow $N(0,1^2)^{d_z}$ when the entropy

of the latent distribution is the largest. The entropy of the scaled latent distribution input to the generator dynamically changes according to the variance of each element of the encoder's output during GAN training. That is, during training, the scaled latent distribution $Z \odot s$ converges to have an optimal entropy representing the real data distribution X.

In line 4, vec^2 means 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 mean squared error between the scaled latent vector $z \odot s$ and the scaled predicted latent vector $z' \odot s$.

In line 5, a_r is the adversarial value of real data. "_" represents not using value.

In lines 6 and 7, f_r^d, f_g^d, and f_g are adversarial loss functions. There are several adversarial loss functions such as Original GAN [2], NSGAN [3], LSGAN [4], WGAN-gp [5], etc. 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 and discriminator are trained cooperatively to reduce the encoder loss L_{enc}.

In line 8, $update$ function update the variance of predicted latent distribution Z' with the new predicted latent vector z'. Since the mean of the predicted latent vector z' become automatically zero, z'^2 (element-wise square of predicted latent vector z') can be considered as the variance of predicted latent vector z'. An exponential moving average or a simple moving average can be used for the $update$ function.

Note that the latent vector input to the generator should always be scaled by the scale vector s. Therefore, the generated data point is $G(z \odot s)$, and the recovered data of x is $G(z_x \odot s)$, where z_x is from $\_, z_x \leftarrow D(x)$.

4. Experimental results and discussion

4.1 Experiment settings

I trained GAN to generate the celeb A dataset [8] resized to 128x128 resolution. As the model architecture, a partially reduced version of styleGAN2 [9] was used. Batch operations (minibatch stddev layer) of StyleGAN2 are removed so that the encoder encodes one data point as one latent vector. As an adversarial loss, NSGAN [3] with r1 loss was used as StyleGAN2. I used encoder loss weight $\lambda_{enc} = 1.0$.

I compared the performance with and without dynamic latent scale. Without dynamic latent scale means using the mean squared error as the encoder loss L_{enc}. For $update$ function for dynamic latent scale GAN, I used a simple moving average using past $512 \times batch\ size = 512 \times 32$ samples.

4.2 Experiment results

The following figures show the performance with and without dynamic latent scale.

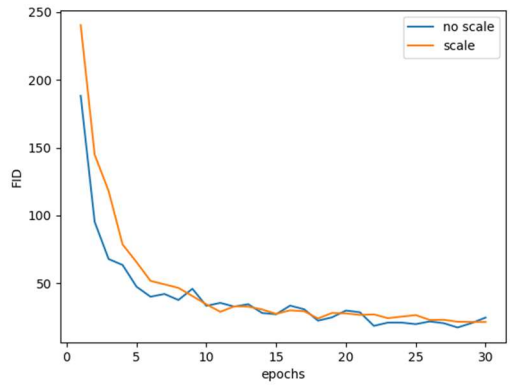


Figure 1. FID with and without dynamic latent scale

Figure 1 shows the FID [10] with and without a dynamic latent scale for each epoch. One can see that the dynamic latent scale has little effect on the generative performance of the model.

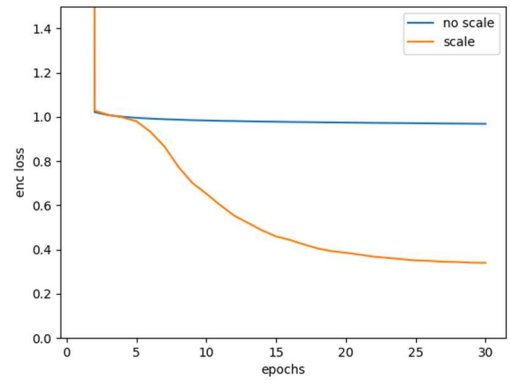


Figure 2. Average L_{enc} with and without dynamic latent scale

Figure 2 shows the average encoder loss L_{enc} with and without a dynamic latent scale for each epoch. One can see that without dynamic latent scale, encoder loss L_{enc} hardly changes from 1. This shows that the model without dynamic latent scale fails to converge due to the information loss on the latent distribution in the generator. 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 a dynamic latent scale.

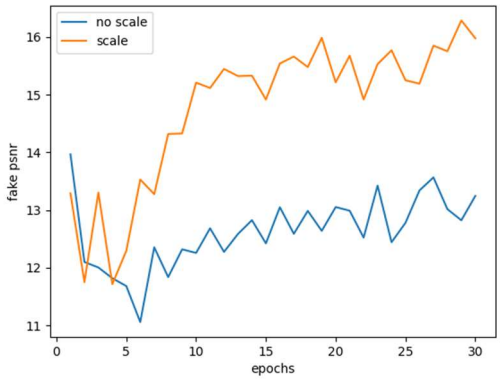


Figure 3. Average PSNR for generated images reconstruction with and without dynamic latent scale

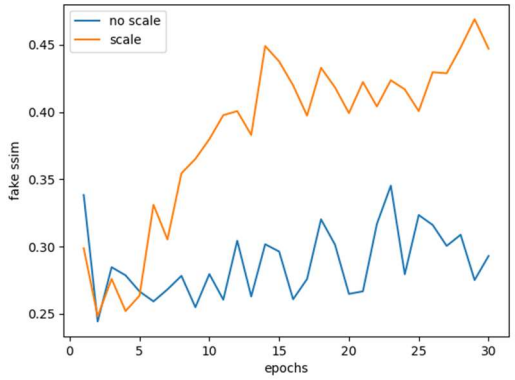


Figure 4. Average SSIM for generated images reconstruction with and without dynamic latent scale

Figures 3 and 4 show the average PSNR and SSIM for each epoch when reconstruction is performed on the generated images. One can see that the performance of reconstruction on generated images is much better with dynamic latent scale.

Figure 5. Generated images reconstruction without dynamic latent scale

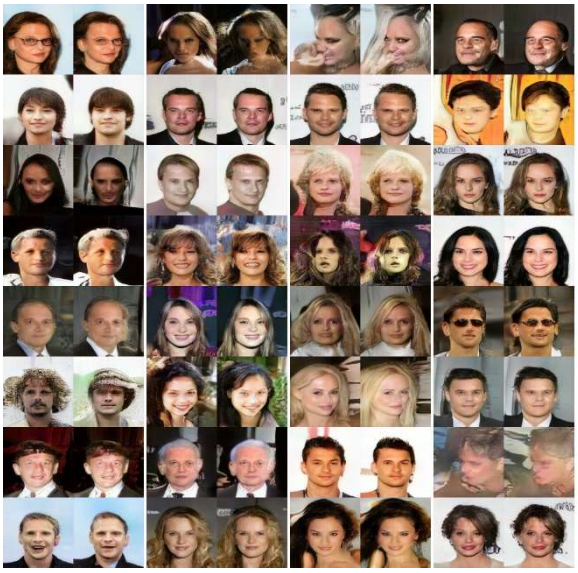


Figure 6. Generated images reconstruction with dynamic latent scale

Figures 5 and 6 show examples of generated images reconstruction with and without dynamic latent scale. Big size images of all figures for generated images are in the appendix.

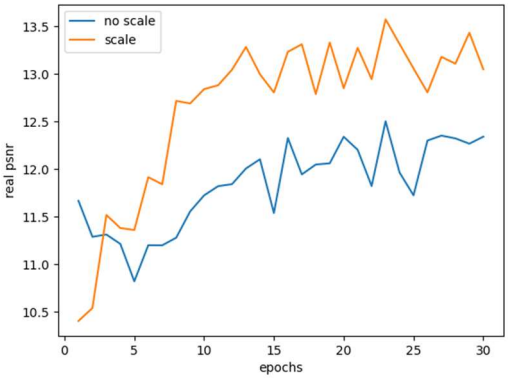


Figure 7. Average PSNR for test images reconstruction with and without dynamic latent scale

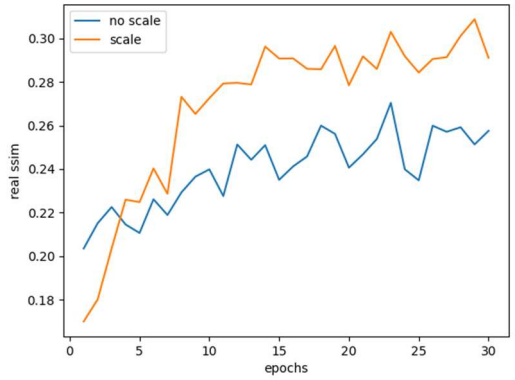


Figure 8. Average SSIM for test images reconstruction with and without dynamic latent scale

Figures 7 and 8 show the average PSNR and SSIM for each epoch when reconstruction is performed on the test images (real images). As reconstruction on the generated images, one can see that the performance of reconstruction on test images is much better with dynamic latent scale.

Figure 9. Test images reconstruction without dynamic latent scale

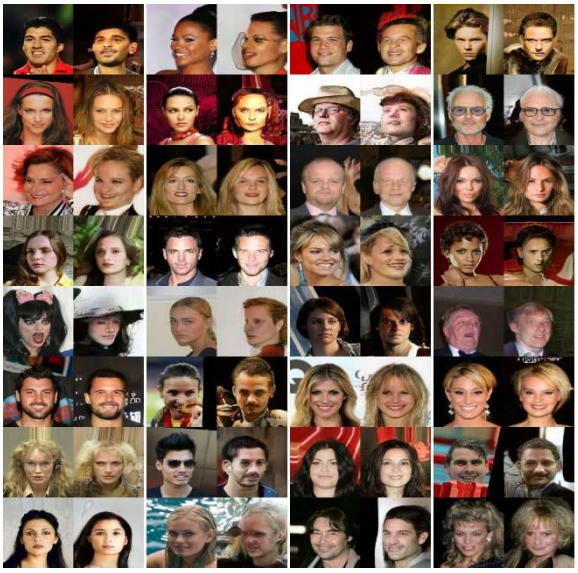


Figure 10. Test images reconstruction with dynamic latent scale

Figures 9 and 10 show examples of generated images reconstruction with and without dynamic latent scale.

4.3 additional results

The following figures show some additional results with dynamic latent scale GAN.

Figure 11. Stochastic latent element flip

Figure 11 shows the result of the stochastic flip (multiplying by -1) of each element of the latent vector of test images. The first column shows the input images (test images), and the second column shows the reconstructed image. The images on the right side based on the thick white line show the result of stochastic flipping of the latent vector. In each column, elements of the same index are flipped. The flip probability was 10%.

5. Conclusion

In this paper, I proposed a dynamic latent scale GAN, a method for training a generator without information loss on latent distribution, and an encoder that inverts the generator. Dynamic latent scale GAN dynamically adjusts the scale of the normal i.i.d. latent distribution to have the optimal entropy to express the data distribution. This ensures that there is no information loss on the latent distribution in the generator so that the encoder can converge. The encoder of dynamic latent scale GAN showed much better performance than simply using mean squared error to train the encoder.

6. References

[1] https://arxiv.org/abs/2101.05278

[2] https://arxiv.org/abs/1406.2661

[3] https://arxiv.org/abs/1801.04406v4

[4] https://arxiv.org/abs/1611.04076

[5] https://arxiv.org/abs/1704.00028

[6] https://arxiv.org/abs/2102.01187

[7] https://arxiv.org/abs/1611.06355

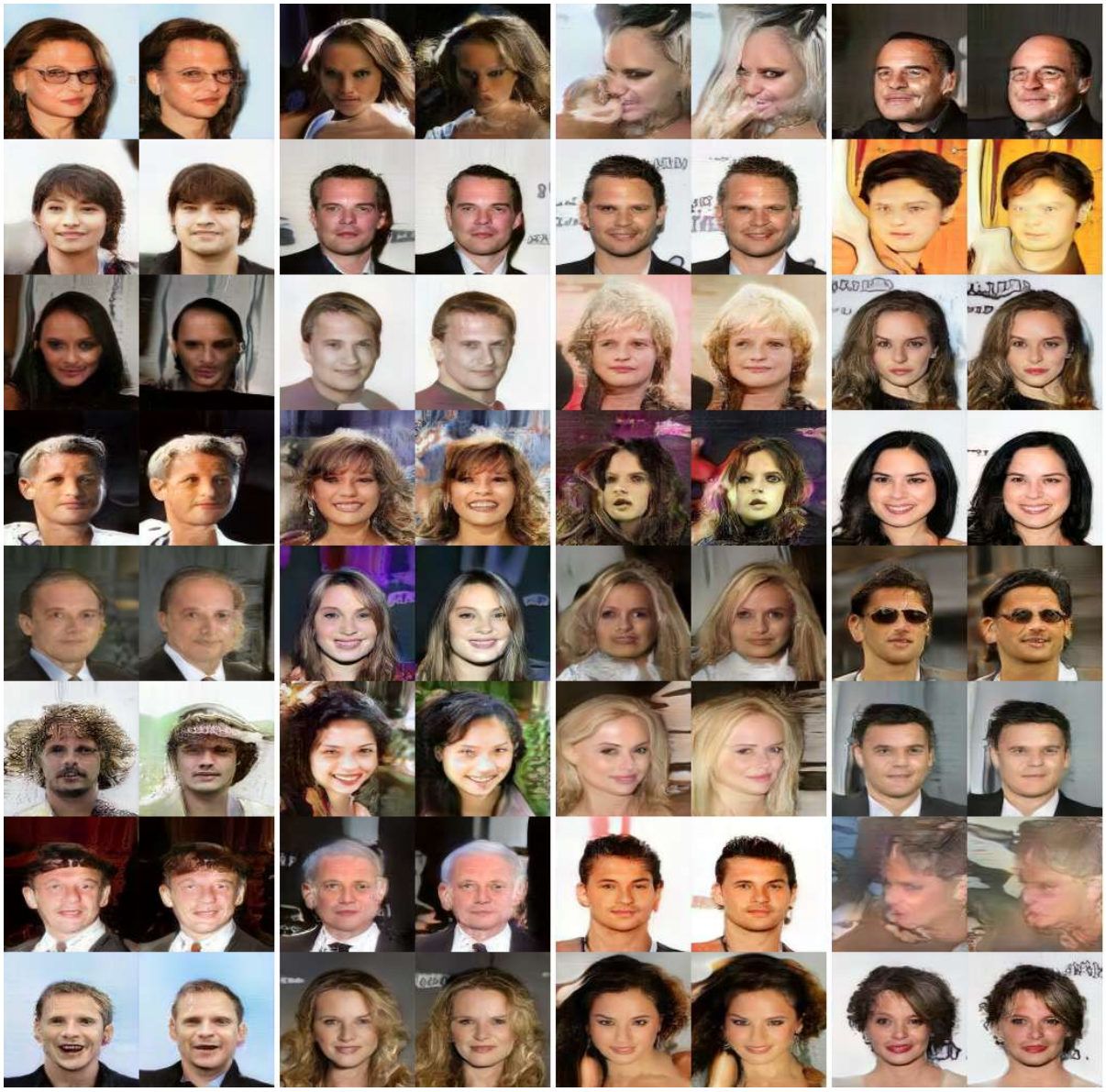
[8] https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

[9] https://arxiv.org/abs/1912.04958

[10] https://arxiv.org/abs/1706.08500

Appendix

Big size image of figure 5. Generated images reconstruction without dynamic latent scale



Big size image of figure 6. Generated images reconstruction with dynamic latent scale

Big size image of Figure 9. Test images reconstruction without dynamic latent scale

Big size image of Figure 10. Test images reconstruction with dynamic latent scale

Big size image of Figure 11. Stochastic latent element flip