Add latent restriction loss when recovering latent vector

Jeongik Cho[1]

Dept. of Computer Science and Engineering[1]

College of Engineering[1]

Konkuk University, Seoul, Korea[1]

jeongik.jo.01@gmail.com[1]

Abstract

When a pre-trained generative model is given, the process of finding the latent vector that produces the data closest to the input data is called the latent vector recover. The latent vector recover receives the difference between the generated data and the input data generated through the latent vector as reconstruction loss and performs gradient descent repeatedly on the latent vector to find the optimal latent vector.

In this paper, I propose a method to find a better latent vector by adding a latent restriction loss in addition to reconstruction loss during latent vector recovery. The latent restriction loss is a loss that makes the latent vector follow the distribution of the latent vector used when training the generative model during latent vector recovery. The distance between the "distribution of latent vector used in training the generative model" and "latent vector during latent vector recovery" becomes the latent restriction loss.

1. Latent vector recover

When a pre-trained generative model is given, the process of finding the latent vector that produces the data closest to the input data is called the latent vector recover. In general, the latent vector recover is performed through the process of repeatedly performing gradient descent on the latent vector by taking the difference between the input data and the generated data as a loss.

$function\ latent\ vector\ recover:$

$\quad for\ i = 0\ to\ n:$

$\quad\quad rcn\ loss \leftarrow diff(G(ltn), x)$

$\quad\quad ltn \leftarrow opt(rcn\ loss, ltn)$

$\quad return\ ltn$

ltn is a latent vector. G is a generative model. $G(ltn)$ is data generated by G receiving ltn. $diff$ is a function that outputs the difference between the two data. opt is a function that receives loss and variable and outputs the updated variable in the direction of minimizing loss.

Through the above process, the latent vector can be recovered.

2. Latent restriction loss

In this paper, I propose a method to find a better latent vector by adding a latent restriction loss to the loss during the latent vector recover process. The generative model is trained to receive the latent vector of a specific distribution and output the distribution of train data during training. However, in the process of latent vector recovery, when updating the latent vector through gradient descent, the latent vector may become very far from the distribution of the latent vector received during training. This means that even if $dif(G(ltn), x)$ is small, the latent vector ltn may not properly represent the input data x.

To prevent this, if the distance between the distribution of latent vectors used in training the generative model and the latent vectors in gradient descent during latent vector recovery is added to the loss, ltn that better represents the input data x can be found.

$function\ latent\ vector\ recover:$

$\quad for\ i = 0\ to\ n:$

$\quad\quad rcn\ loss \leftarrow diff(G(ltn), x)$

$\quad\quad lr\ loss \leftarrow dist(ltn, train\ ltn)$

$\quad\quad loss \leftarrow rcn\ loss + \alpha_{lr} lr\ loss$

$\quad\quad ltn \leftarrow opt(loss, ltn)$

$\quad return\ ltn$

$train\ ltn$ is the distribution of latent vectors used in G training. $dist$ is a function indicating the distance between two distributions. Each element of the vector input to $dist$ is treated as a sample. For example, the distance between a vector [1.5, 2.0, -0.5] and a vector [2.0, -0.5, 1.5] is 0. $lr\ loss$ is latent restriction loss. α_{lr} is the weight of $lr\ loss$.

3. Experiment

I tested the performance difference with latent restriction loss in Defense-GAN using Latent recovery. In the experiment, an MNIST handwriting dataset in which each pixel value was normalized from -1 to 1 was used. Classifier has an accuracy of 99.38%. GAN follows the structure of DC-GAN and receives a 256-dimensional latent vector following a gaussian distribution, and outputs MNIST handwriting data.

$n = 200, diff = mean\ squared\ error, dist = wasserstein\ distance, opt = Adam(learning\ rate = 0.001, beta1 = 0.9, beta2 = 0.999)$ was used for latent vector recovery, and 10 randomly initialized latent vectors per data were used. FGSM was used as an adversarial attack, and the noise magnitude was 0.7.

$$noised\ image = clip(input\ imag + noise\ magnitude \times FGSM\ noise, -1\ to\ 1)$$

Because the latent vector recovery took a long time, 1000 randomly selected data among 10000 MNIST test data were used for evaluation.

As a result of the experiment, the accuracy of the classifier was 1.4% when the Defense GAN was not used, 55.3% for the Defense GAN without a Latent restriction loss, and 64.1% for the Defense GAN with a Latent restriction loss weight of 1. This shows that latent restriction loss helps to find latent vectors that better represent the input data.

4. References

Pouya Samangouei, Maya Kabkab, Rama Chellappa

Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models

https://arxiv.org/abs/1805.06605

Yann LeCun, Corinna Cortes, Christopher J.C. Burges

THE MNIST DATABASE of handwritten digits

http://yann.lecun.com/exdb/mnist/

Alec Radford, Luke Metz, Soumith Chintala

Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

https://arxiv.org/abs/1511.06434

Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy

Explaining and Harnessing Adversarial Examples

https://arxiv.org/abs/1412.6572

Diederik P. Kingma, Jimmy Ba

Adam: A Method for Stochastic Optimization

https://arxiv.org/abs/1412.6980