Conditional Activation GAN: improved conditional GAN

JeongIk Cho

Konkuk University

Note: this paper is part of my other paper (http://vixra.org/abs/1909.0061?ref=10946100, Temporary name is ACL-GAN). Therefore, there is an overlap in the content. I will update both papers not to overlap the content after the supplement.

Abstract

Conditional GAN is a GAN that can generate data with the desired condition from the latent vector. In this study, propose a conditional activation GAN that can replace conditional GAN to reduce hyperparameter and improve training speed. Conditional activation loss is the sum of the losses of each GAN when creating a GAN for each condition, and since each GAN shares hidden layers, it does not increase the amount of computation much. Also, like conditional activation GAN is a combination of multiple GANs, proposes a method for evaluating conditional GAN: the average of each GAN's evaluation score. Also, purpose mixed batch training to apply batch normalization in discriminator.

1. Introduction

Conditional GAN [1] has been used by many GANs to generate data with desired conditions. In this study, propose conditional activation GAN, which replaces conditional GAN, to reduce hyperparameter of conditional GAN, and improve training speed. Loss of conditional activation GAN is the sum of losses of each GAN that each GAN trains only one attribute. Because every GAN shares all hidden layers, it is possible to consider all GANs as one single GAN. Unlike conditional GAN using two losses (adversarial loss, classification loss), conditional activation GAN uses only one loss (conditional activation loss), which means it does not need to find the ratio of adversarial loss and classification loss. Also, conditional activation loss always produces meaningful gradients, whereas generator classification loss using cross-entropy produces meaningless gradients at the beginning of training.

There are several ways to evaluate a single GAN, such as an inception score [2] or FID [3]. However, in the case of conditional GAN (or conditional activation GAN), it is difficult to evaluate because generated data distribution must follow not only real data distribution but also condition distribution. To evaluate conditional GAN (or conditional activation GAN), I purpose the average of each GAN's evaluation score. Conditional GAN (or conditional activation GAN) can be thought of as a collection of multiple GANs that each GAN trains only one condition. Simply averaging each GAN's evaluation score (such as inception score or fid or other evaluation methods to evaluate single GAN) can evaluate conditional GAN (or conditional activation GAN).

In conditional GAN (or conditional activation GAN), applying batch normalization to discriminator distorts condition distribution of input batch. If discriminator applied batch normalization, the distribution of generated data batch tries to follow the distribution of real data batch. Likewise, condition distribution of generated data batch tries to follow the condition distribution of real data batch. This means that some data in the generated data batch may ignore the input condition to follow the condition distribution of the real data batch. I suggest mixed batch training, which is composing batch always with the same ratio of real data and generated data, to keep condition distribution of batch same to apply batch normalization in the discriminator.

2. Conditional Activation GAN

2.1 Conditional Activation GAN

The loss of conditional GAN is as follows.

$$L_D = L_{adv}^{d} + \lambda_{cls} L_{cls}^{r}$$

$$L_G = L_{adv}^{g} + \lambda_{cls} L_{cls}^{g}$$

$$L_{cls}^{r} = E_{x,att \sim P_r(x,att)}\left[-\log\left(D_{cls}(att|x)\right)\right]$$

$$L_{cls}^{g} = E_{x',att' \sim P_g(x',att')}\left[-\log\left(D_{cls}(att'|x')\right)\right]$$

In $x, att \sim P_r(x, att)$, x is real data, and att is the binary vector that expresses the attributes of real data. In $x', att' \sim P_g(x', att')$, x' means generated data, and att' is the target binary vector to make x'.

In the conditional GAN, adversarial loss trains model well because there are well known adversarial losses such as LSGAN [4] or WGAN-GP [5] that can produce meaningful gradients even if real data distribution and generated data distribution are far from each other. However, classification loss of conditional GAN, which is using cross-entropy, is hard to produce meaningful gradients if real data distribution and generated data distribution are far from each other because cross-entropy measures only KL-divergence.

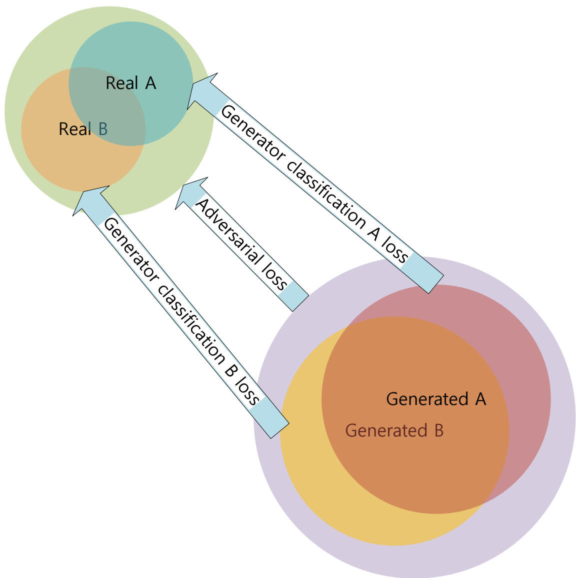


Fig1. Data distribution at the beginning of training using conditional GAN

In the early stage of learning, the generator classification loss L_{cls}^{g} does not produce meaningful gradients because the distance between Real A and Generated A, Real B and

Generated B are too far from each other. Only adversarial loss produces meaningful gradients.

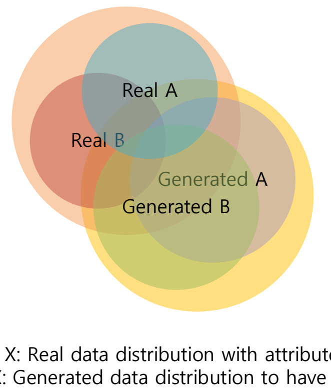


Fig2. After some training using conditional GAN

As learning progresses to some degree with adversarial loss, when the real data distribution and the generated data distribution become somewhat similar, the generator classification loss L_{cls}^{g} begins to produce a meaningful gradient because real A and generated A, real B and generated B become somewhat similar.

Also, conditional GAN has important hyperparameters: adversarial loss weight and classification loss weight. If adversarial loss weight is too bigger than the classification loss weight, the generated data would not have the target condition. If classification loss weight is too bigger than adversarial loss weight, the data does not look real.

To solve these problems of conditional GAN, I propose conditional activation loss, which is similar to having multiple GANs that each GAN trains each attribute.

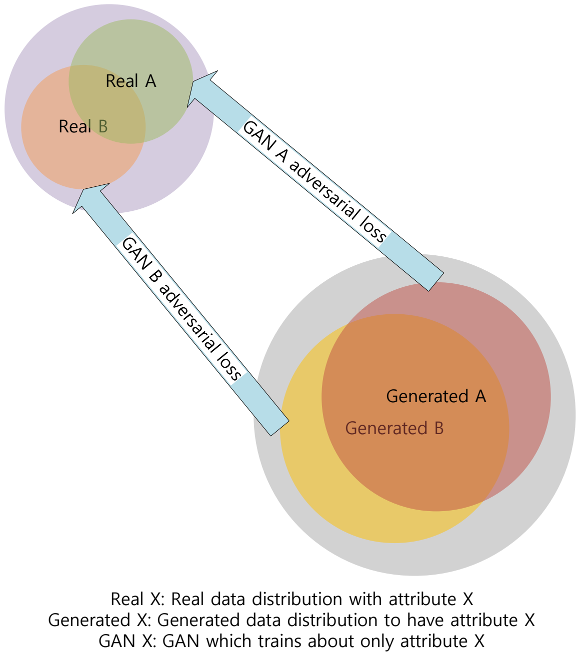


Fig3. Conditional activation loss

Conditional activation loss is the sum of each GAN's loss. Each GAN trains only one attribute.

$$L_{ca}^{D} = \sum_{c}^{att} L_{D_c}$$

$$L_{ca}^{G} = \sum_{c}^{att} L_{G_c}$$

$$L_{D_c} = E_{x,c\sim P_r(x,c)}[f_r^D(D_c,x)] + E_{x'\sim P_{G_c}(x',1)}[f_g^D(D_c,x')]$$

$$L_{G_c} = E_{x'\sim P_{G_c}(x',1)}[f^G(D_c,x')]$$

c means one specific attribute among several attributes. GAN c is the GAN that train about only attribute c.

G_c and D_c are generator and discriminator of GAN c. G_c receives a binary activation value with a latent vector. If G_c receives 1 as an activation value, G_c tries to trick D_c, and D_c tries to discriminate generated data as fake. If G_c receives 0 as activation value, G_c and D_c don't care about it (do not train). D_c only tires of discriminating real data, which has attribute c as real, and don't care about other real data.

In $x, c \sim P_r(x, c)$, x is real data which has attribute c. In $x' \sim P_{G_c}(x', 1)$, x' is generated data by G_c when it receives latent vector and 1 as activation value.

f_r^D is an adversarial loss of discriminator about real data. f_g^D is an adversarial loss of discriminator about generated data. f^G is an adversarial loss of generator.

The following formula is an example of LSGAN adversarial loss.

$$L_{D_c} = E_{x,c \sim P_r(x,c)}[(D_c(x) - 1)^2] + E_{x' \sim P_{G_c}(x',1)}[D_c(x')^2]$$

$$L_{G_c} = E_{x' \sim P_{G_c}(x',1)}[(D_c(x') - 1)^2]$$

Since each GAN shares all hidden layers, conditional activation loss can be changed as the following formula.

$$L_{ca}^D = E_{x,att \sim P_r(x,att)}[f_r^D(D, x) \cdot att]$$

$$+E_{x',att' \sim P_g(x',att')}\left[f_g^D(D, x') \cdot att'\right]$$

$$L_{ca}^G = E_{x',att' \sim P_g\left(x',att'\right)}\left[f^G\left(D, x'\right) \cdot att'\right]$$

In $x, att \sim P_r(x, att)$, x is real data, and att is the binary vector that expresses the attributes of real data. In $x', att' \sim P_g(x', att')$, x' means generated data, and att' is the target binary vector to make x'. '$\cdot$' is an inner product.

The following formula is an example of conditional activation loss with LSGAN adversarial loss.

$$L_{ca}^D = E_{x,att \sim P_r(x,att)}\left[(D(x) - 1)^2 \cdot att\right]$$

$$+E_{x',att' \sim P_g(x',att')}\left[\left(D(x')\right)^2 \cdot att'\right]$$

$$L_{ca}^G = E_{x',att' \sim P_g\left(x',att'\right)}\left[\left(D\left(x'\right) - 1\right)^2 \cdot att'\right]$$

Also, in conditional GAN, when the output of classifier A is 0, that means input data does not have attribute A. However, in conditional activation GAN, GAN A does not care about attribute not-A. Therefore, to train attribute not-A, new GAN which trains attribute not-A should be added.

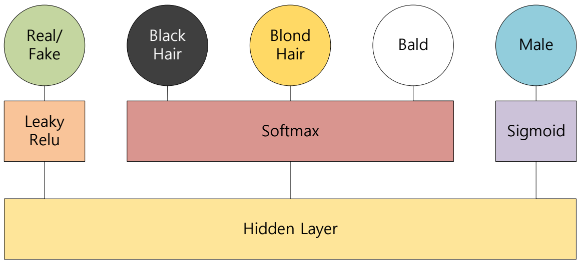


Fig4. conditional GAN discriminator output example

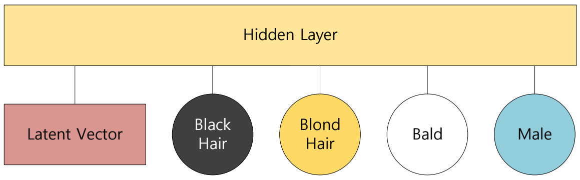


Fig5. conditional GAN generator input example

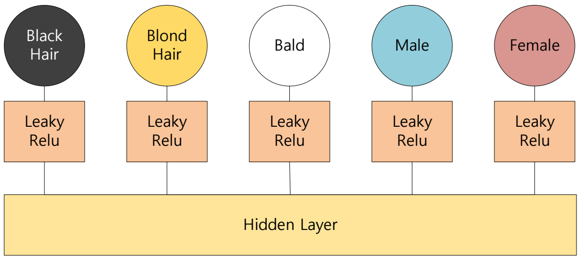


Fig6. conditional activation GAN discriminator output example

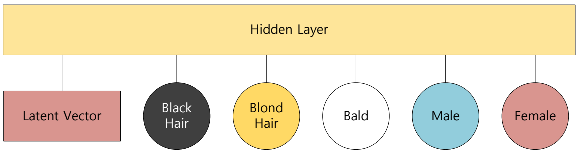


Fig7. conditional activation GAN generator input example

(Assume P(Black hair) + P(Blond hair) + P(Bald) = 1, P(Male) + P(Female) = 1)

Using conditional activation loss with adversarial loss of LSGAN or WGAN-GP or other GAN can generate meaningful gradients at the beginning of the training when real data distribution and generated data distribution are far from each other. Also, conditional activation loss can replace adversarial loss and classification loss, which can reduce one important hyperparameter of conditional GAN. Conditional activation GAN loss has only one hyperparameter: conditional activation loss weight, while conditional GAN loss has two hyperparameters: adversarial loss weight, classification loss weight.

2.2 Mixed batch training

In conditional GAN (or conditional activation GAN), applying batch normalization to discriminator distorts condition distribution of input batch. If discriminator applied batch normalization, the distribution of generated data batch tries to follow the distribution of real data batch. Likewise, condition distribution of generated data batch tries to follow the condition distribution of real data batch. This means that some data in the batch may ignore the input condition to follow the condition distribution of the real data batch. I suggest mixed batch training, which is composing batch always with the same ratio of real data and generated data, to keep condition distribution of batch same to apply batch normalization to the discriminator.

2.3 Average evaluation score

There are several ways to evaluate a single GAN, such as an inception score or FID. However, in the case of conditional GAN (or conditional activation GAN), it is difficult to evaluate because generated data distribution must follow not only real data distribution but also condition distribution. To evaluate conditional GAN (or conditional activation GAN), I purpose the average of each GAN's evaluation score. Conditional GAN (or conditional activation GAN) can be thought of as a collection of multiple GANs that each GAN trains only one condition. Simply averaging each GAN's evaluation score (such as inception score or fid or other evaluation methods to evaluate single GAN) can evaluate conditional GAN (or conditional activation GAN).

3. Material and methods

Used train dataset of MNIST handwriting number dataset [6]. Data size is 60000, resolution is 28x28 and channel size is 1.

Used tensorflow2.0. Model architecture used basic design of DCGAN [7]. However, used batch normalization on only discriminator, not generator.

Used conditional activation loss with LSGAN adversarial loss.

Used average of FID for evaluation. Used all test dataset for calculate FID. Generated data size is same as each test dataset size. Since the MNIST dataset has one channel and their resolution is too low to input the inception network, triple the resolution and channel (84x84x3).

4. Results and Conclusions

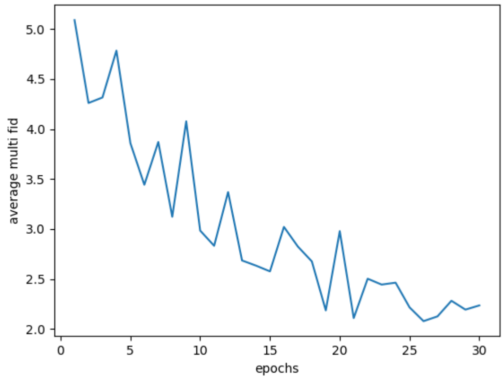
Each row has same condition. Each col has same latent vector.

Epoch 1

Epoch 10

Epoch 20

Epoch 100



This graph is average fid of each GAN(low is better).

5. Discussion and Future works

Experiments show that conditional activation GAN can replace conditional GAN. Since there is one less hyperparameter than conditional GAN loss, conditional activation loss can significantly reduce the time to search for optimal hyperparameters. Further experimentation is needed to compare the training speed of conditional activation GAN and conditional GAN.

6. Funding

This work was supported by "University Innovation Grant" from the Ministry of Education and National Research Foundation of Korea

7. References

[1] Mehdi Mirza, Simon Osindero

Conditional Generative Adversarial Nets

https://arxiv.org/abs/1411.1784

[2] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen

Improved Techniques for Training GANs

https://arxiv.org/abs/1606.03498

[3] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Sepp Hochreiter

GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium

https://arxiv.org/abs/1706.08500

[4] Xudong Mao, Qing Li, Haoran Xie, Raymond Y.K. Lau, Zhen Wang, Stephen Paul Smolley

Least Squares Generative Adversarial Networks

https://arxiv.org/abs/1611.04076

[5] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, Aaron Courville

Improved Training of Wasserstein GANs

https://arxiv.org/abs/1704.00028

[6] Yann LeCun, Corinna Cortes, Christopher J.C. Burges

THE MNIST DATABASE of handwritten digits

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

[7] Alec Radford, Luke Metz, Soumith Chintala

Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

https://arxiv.org/abs/1511.06434