# Conditional Activation GAN: Improved Auxiliary Classifier GAN

JeongIk Cho[1], Kyoungro Yoon [2] (Corresponding Author)

Dept. of Computer Science and Engineering[1], Dept. of Smart ICT Convergence Engineering[2]

College of Engineering[1], KU Institute of Technology[2]

Konkuk University, Seoul, Korea[1], Konkuk University, Seoul, Korea[2]

jeongik.jo.01@gmail.com[1], yoonk@konkuk.ac.kr[2]

## Abstract

*Conditional GAN is a GAN that generates data with the desired condition from the latent vector. The auxiliary classifier GAN is the most used among the variations of conditional GANs. In this study, we explain the problem of auxiliary classifier GAN and propose conditional activation GAN that can replace auxiliary classifier GAN to reduce the number of hyperparameters and improve training speed. The loss function of conditional activation GAN is defined as the sum of the loss of each GAN created for each condition. Since each GAN shares all hidden layers, the GANs can be considered as a single GAN and it does not increase the amount of computation much. Also, in order to apply batch normalization to the discriminator of conditional GANs, we propose a mixed batch training, in which each batch for discriminator is always configured to have the same ratio of real data and generated data so that each batch always has the same condition distribution.*

## 1. Introduction

Conditional GAN [1] is a GAN [2] that can generate data with the desired condition from the latent vector. Among the variations of conditional GANs [3, 4], the most commonly used conditional GAN is the Auxiliary Classifier GAN (AC-GAN) [5] used in [6, 7, 8, 9, 10, 11]. Some papers used a variation of AC-GAN [10, 11] without giving any details on the rationalization of the variations made. In this study, we explain the reasons for the modification of AC-GAN and the disadvantages of AC-GAN.

In AC-GAN, when real data distribution and generated data distribution is the same, auxiliary classifier of the discriminator and the generator can be considered as a group of GANs, each of which trains each condition and cross-entropy adversarial loss by sharing all hidden layers. Considering the AC-GAN as a set of GANs, the generated data classification loss of the AC-GAN discriminator loss interferes with the training of each GAN and hence is removed in the modified AC-GAN.

Since each GAN can be trained as a GAN only when the real data distribution and the generated data distribution are the same, there is a problem that individual GAN may not be trained at the beginning of the AC-GAN training.

Also, to use the advanced adversarial loss as used in papers such as LSGAN [12] or WGAN-GP [13] in AC-GAN, a hyperparameter that is adjusting the ratio of adversarial loss and classification loss should be decided.

We propose a conditional activation GAN (CA-GAN) that can replace AC-GAN to reduce the number of hyperparameters and improve training speed to overcome the upper mentioned problems of AC-GAN. Loss of CA-GAN is the sum of the losses of each GAN when each GAN is created for each condition. Since each GAN shares all hidden layers, the CA-GAN composed on a conceptual aggregation of individual GAN can be considered as a single GAN.

Unlike AC-GAN's use of two losses (adversarial loss, classification loss), CA-GAN uses only one loss (conditional activation loss), so there is no need to find the proper ratio of adversarial loss and classification loss.

Also, while AC-GAN starts to train each condition when the real data distribution is the same to the generated data distribution, CA-GAN always trains each condition simultaneously, which means that CA-GAN always produces meaningful gradients, even in the early training stage.

In conditional GANs, training by applying batch normalization [14] to the discriminator induces the generator to distort the input condition distribution.

When batch normalization is applied to the discriminator, and the real data and the generated data condition distribution are different, the discriminator may use the batch condition distribution for real/fake discrimination and the generated data condition distribution follows the real data condition distribution, not the input target condition distribution.

To prevent the generator from ignoring the input target condition distribution, we suggest mixed batch training. Mixed batch training is to always configure each batch for discriminator with the same ratio of real data and generated data so that each batch always has the same condition distribution.

## 2. Analysis of Auxiliary classifier GAN

The loss of AC-GAN is defined as follows [5]:

$$L_d = L_{adv}^d + L_{cls}^r + L_{cls}^g \quad (1)$$

$$L_g = L_{adv}^g + L_{cls}^r + L_{cls}^g \quad (2)$$

$$L_{cls}^r = E_{x,cnd \sim P_r(x,cnd)}[-\log D_{cls}(cnd|x)] \quad (3)$$

$$L_{cls}^g = E_{x',cnd' \sim P_g(x',cnd')}[-\log D_{cls}(cnd'|x')] \quad (4)$$

$$L_{adv}^d = E_{x \sim P_r(x)}[-\log D_{adv}(x)] + E_{x \sim P_g(x)}\left[-\log\left(1 - D_{adv}(x)\right)\right] \quad (5)$$

$$L_{adv}^g = E_{x \sim P_g(x)}\left[\log\left(1 - D_{adv}(x)\right)\right] \quad (6)$$

In (1) and (2), $L_d$ is the loss of the discriminator and $L_g$ is the loss of the generator. $L_{adv}^{d}$ is the adversarial loss of the discriminator and $L_{adv}^{g}$ is the adversarial loss of the generator. In (5), $D_{adv}$ is the probability distribution function of the data in the adversarial module. $D_{adv}(x)$ is the probability distribution of $x$, which is given as the input of the adversarial module. $E$ is the expectation of the given variable. Symbol "~" means "is distributed as". For example, $E_{x\sim P_z(x)}[f(x)]$ is an expectation value of $f(x)$ when $x$ follows the distribution of $P_z(x)$.

In $x, cnd \sim P_r(x, cnd)$ of (3), $x$ is the real data, and $cnd$ is the binary vector that expresses the conditions of real data. In $x', cnd' \sim P_g(x', cnd')$ of (4), $x'$ is the generated data and $cnd'$ is the target binary vector to generate $x'$. $D_{cls}(x)$ is the probability distribution of data $x$ within auxiliary classifier of the discriminator. $-\log D_{cls}(cnd|x)$ is the cross-entropy loss between $cnd$ and $D_{cls}(x)$. Minimizing $-\log D_{cls}(cnd|x)$ means that $D_{cls}$ is trained to estimate the conditions of $x$ ($cnd$) well.

Note that $L_{cls}^{r}$ in $L_g$ does not play any role because the generator does not affect the calculation of $L_{cls}^{r}$.

In AC-GAN, when real data distribution and generated data distribution is the same, auxiliary classifier of the discriminator and the generator can be considered as a group of GANs that each GAN trains each condition using cross-entropy adversarial loss, and shares all hidden layers as shown in Fig. 1.

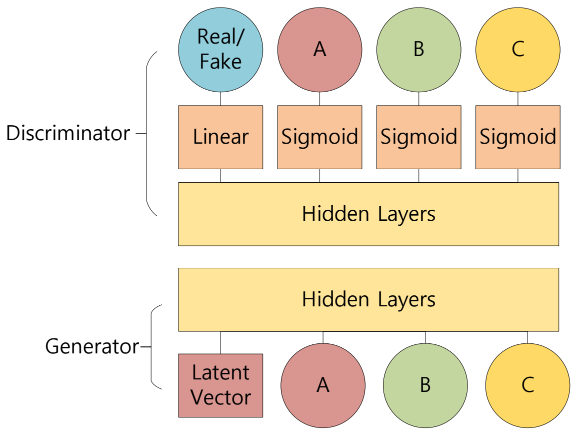


Fig1. AC-GAN that trains A, B, and C conditions

Suppose that AC-GAN training three independent conditions (A, B, C) trains only with adversarial loss, and the real data distribution and the generated data distribution are the same.

Node A of the discriminator is trained by $L_{cls}^{r}[A]$ in $L_d$ to output 1 to represent real when it receives real data with condition A, and 0 to represent fake with condition not-A.

When the generator receives 1 as its node A's input, it attempts to generate data by $L_{cls}^{g}[A]$ in $L_g$ with condition A, and trains the discriminator's node A output to be 1.

If the generator attempts to generate data with condition A but fails, the generated data distribution will be close to the real data distribution with condition not-A since it is assumed that the real data distribution and the generated data distribution are the same.

Thus, the hidden layers of the discriminator and node A, the hidden layers of the generator and the latent vector input, and node A can be thought of as a single GAN A that generates data with condition A trained by $L^r_{cls}[A]$ in $L_d$ and $L^g_{cls}[A]$ in $L_g$. However, $L^g_{cls}[A]$ in $L_d$ trains node A of the discriminator to be 1 representing real when the discriminator receives generated data. Therefore, $L^g_{cls}[A]$ in $L_d$ interferes with the training of GAN A.

Also, when the generator receives 0 as its node A's input, it can be thought of as a GAN that generates data with condition not-A.

AC-GAN uses cross-entropy loss as an adversarial loss. However, in order to use advanced adversarial loss such as LSGAN or WGAN-GP, a hyperparameter is needed to adjust the ratio of adversarial loss and classification loss.

To solve these problems, the loss of the modified AC-GANs used in StarGAN [10] or AttGAN [11] is modified as follows:

$$L_d = L^d_{adv} + \lambda_{cls} L^r_{cls} \quad (7)$$

$$L_g = L^g_{adv} + \lambda_{cls} L^g_{cls} \quad (8)$$

$$L^r_{cls} = E_{x,cnd \sim P_r(x,cnd)}[-\log D_{cls}(cnd|x)] \quad (9)$$

$$L^g_{cls} = E_{x',cnd' \sim P_g(x',cnd')}[-\log D_{cls}(cnd'|x')] \quad (10)$$

In (7) and (8), $L_d$ is loss of discriminator and $L_g$ is loss of generator. $L^d_{adv}$ is adversarial loss of discriminator and $L^g_{adv}$ is adversarial loss of generator. In $x, cnd \sim P_r(x, cnd)$ of (9), $x$ is real data, and $cnd$ is the binary vector that expresses the conditions of real data. In $x', cnd' \sim P_g(x', cnd')$ of (10), $x'$ is generated data and $cnd'$ is the target binary vector to generate $x'$. $\lambda_{cls}$ is classification loss weight.

As explained above, modified AC-GAN also can be considered as a group of GANs. However, each GAN can only be trained as a GAN for each condition only if the real data distribution and the generated data distribution for the corresponding condition are the same.

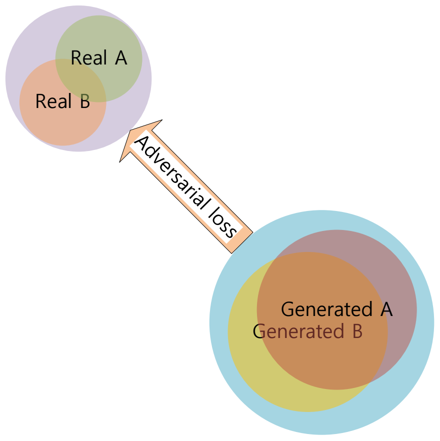

Real A
Real B
Adversarial loss
Generated A
Generated B
Real X: Real data distribution with condition X
Generated X: Generated data distribution to have condition X

Fig2. Data distribution at the beginning of training using AC-GAN

In other words, if the real data distribution differs from the generated data distribution at the beginning of the training, the training does not proceed with classification loss, but only with adversarial loss, as shown in Fig.2.

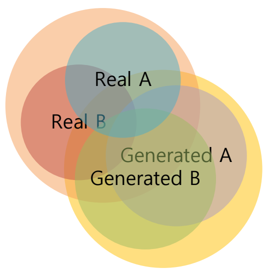

Real A
Real B
Generated A
Generated B
Real X: Real data distribution with condition X
Generated X: Generated data distribution to have condition X

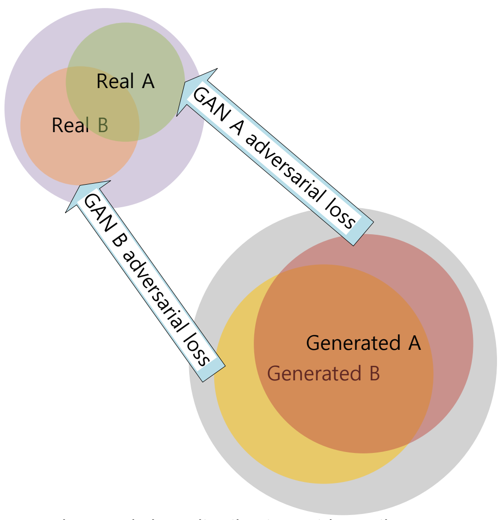
Fig3. After some training using AC-GAN

By training with adversarial loss, the real data distribution and the generated data distribution gets closer. As these distributions get closer to each other, the classification loss gradually acts as the cross-entropy adversarial loss of each GAN, and produces meaningful gradients and training is performed to generate data with each condition.

AC-GAN has the disadvantage of requiring one additional hyperparameter to adjust the ratio of adversarial loss and classification loss in both discriminator and generator and not producing meaningful gradients early stage of training.

## 3. Conditional activation GAN

To solve these problems of AC-GAN, we propose conditional activation GAN (CA-GAN), which is similar to having multiple GANs each of which is defined to train corresponding condition.



Real X: Real data distribution with attribute X
Generated X: Generated data distribution to have attribute X
GAN X: GAN which trains about only attribute X

Fig4. Conditional activation GAN

Loss of conditional activation GAN is the sum of each GAN's loss where Each GAN trains only one condition as defined in the following equation.

$$L_d = \sum\nolimits_{for\ each\ c\ in\ S_{cnd}} L_{d_c} \quad (11)$$

$$L_g = \sum\nolimits_{for\ each\ c\ in\ S_{cnd}} L_{g_c} \quad (12)$$

$$L_{d_c} = E_{x,c\sim P_r(x,c)}\left[f_r^d\left(D_c(x)\right)\right]$$

$$+E_{x'\sim P_{g_c}(x',1)}\left[f_g^d\left(D_c(x')\right)\right] \quad (13)$$

$$L_{G_c} = E_{x'\sim P_{g_c}(x',1)}\left[f^g\left(D_c(x')\right)\right] \quad (14)$$

In (11) and (12), $L_d$ and $L_g$ represent the discriminator and the generator losses of conditional activation GAN, respectively. $S_{cnd}$ represents the set of conditions that the given CA-GAN is intended to be trained for. $c$ is one specific condition in $S_{cnd}$. GAN $c$ is an individual GAN that trains for only condition $c$.

$g_c$ 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 from $g_c$ as fake. If $g_c$ receives 0 as the activation value, both $g_c$ and $d_c$ do not care about what has been generated. $d_c$ only cares about discriminating real data, which has condition $c$, and does not care about other real data including real data with condition not-$c$.

In $x,c\sim P_r(x,c)$ of (13), $x$ is the real data which has condition $c$. In $x'\sim P_{g_c}(x',1)$, $x'$ is generated data by $g_c$ when it receives latent vector with 1 as activation value.

$f_r^d$ is a function that calculates the adversarial loss of the discriminator about real data. $f_g^d$ is a function that calculates the adversarial loss of the discriminator about generated data. In (14), $f^g$ is a function that calculates the adversarial loss of the generator.

The following equation is an example of the adversarial loss of GAN $c$ that uses adversarial loss given in LSGAN [12].

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

$$+E_{x' \sim P_{g_c}(x',1)}[D_c(x')^2] \qquad (15)$$

$$L_{g_c} = E_{x' \sim P_{g_c}(x',1)}[(D_c(x') - 1)^2] \qquad (16)$$

In CA-GAN, since each GAN shares all hidden layers, conditional activation loss can be changed as the following equation.

$$L_d = E_{x,cnd \sim P_r(x,cnd)}[f_r^d(D(x)) \cdot cnd]$$

$$+E_{x',cnd' \sim P_g(x',cnd')}[f_g^d(D(x')) \cdot cnd'] \qquad (17)$$

$$L_g = E_{x',cnd' \sim P_g(x',cnd')}\left[f^g\left(D(x')\right) \cdot cnd'\right] \qquad (18)$$

In $x, cnd \sim P_r(x, cnd)$ of (17), $x$ is real data, and $cnd$ is the binary vector that expresses the conditions of real data. In $x', cnd' \sim P_g(x', cnd')$ of (18), $x'$ means generated data, and $cnd'$ is the target binary vector to make $x'$. "$\cdot$" is an inner product.

The following equation is the loss of CA-GAN when it is using the adversarial loss of LSGAN.

$$L_d = E_{x,cnd \sim P_r(x,cnd)}[(D(x) - 1)^2 \cdot cnd]$$

$$+E_{x',cnd' \sim P_g(x',cnd')}\left[\left(D(x')\right)^2 \cdot cnd'\right] \qquad (19)$$

$$L_g = E_{x',cnd' \sim P_g(x',cnd')}\left[\left(D(x') - 1\right)^2 \cdot cnd'\right] \qquad (20)$$

In AC-GAN, GAN A that trains condition A also generates data with condition not-A as well as data with condition A.

However, in CA-GAN, since GAN A, training with condition A, does not care about condition not-A, a new GAN training condition not-A must be added to train condition not-A.

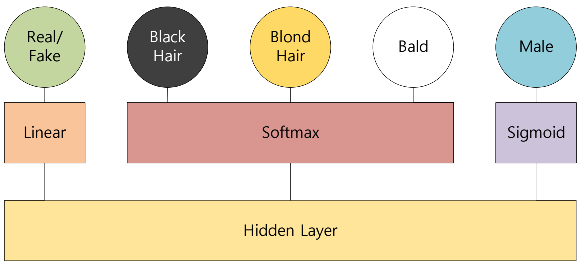


Fig5. AC-GAN discriminator output example

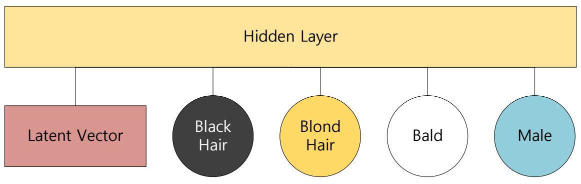


Fig6. AC-GAN generator input example

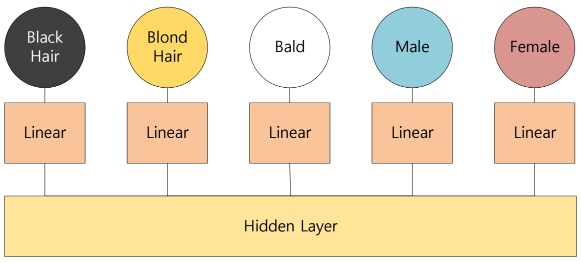


Fig7. conditional activation GAN discriminator output example

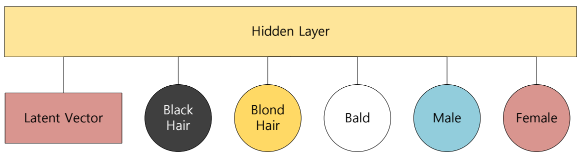


Fig8. conditional activation GAN generator

input example

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

In CA-GAN, since each GAN can be trained through advanced adversarial loss that generates meaningful gradients even if the real data distribution and the generated data distribution are different, meaningful gradients are generated even at the beginning of the training.

Also, unlike AC-GAN's use of two losses (adversarial loss, classification loss), CA-GAN uses only one loss (conditional activation loss), so there is no need to find the proper ratio of adversarial loss and classification loss. This means that it takes less time to search for an important hyperparameter: the ratio of adversarial loss and classification loss.

## 4. Mixed batch training

In conditional GANs, training by applying batch normalization to the discriminator may induce the generator to distort the input condition distribution.

When batch normalization is applied to the discriminator and the target condition distribution used for training is different from the real data condition distribution, the discriminator may use the batch condition distribution for real/fake discrimination, which leads generated data condition distribution to follow real data condition distribution. To prevent the generator from ignoring the input target condition distribution, we suggest mixed batch training.

Mixed batch training is configuring each batch for discriminator always to have the same ratio of real data and generated data so that each batch always has the same condition distribution. Since each training batch is always configured to keep the same condition distribution, the discriminator will not discriminate real/fake by condition distribution, and the generator will not attempt to follow the real data condition distribution.

## 5. Material and methods

In this experiment, we used the training dataset of the MNIST handwriting number dataset [15] for the training. The dataset has 60000 images with an image resolution of 28 pixels x 28 pixels, and the channel size is 1. The basic design of DCGAN [16] without batch normalization is used for the model architecture. Adversarial loss of LSGAN was used both for AC-GAN and CA-GAN. Adam optimizer [17] is used and the network is trained for 50 epochs.

The batch size of all experiments was 32 and the latent vector dimension was 128.

For the implementation, tensorflow2.0 is used [18].

For the evaluation of the proposed network, an average of Fréchet Inception Distance (FID) [19] over all conditions is used.

All the experiments were conducted three times each and the average of each result was used.

The size of the generated data set is the same as the size of each test dataset in evaluation. Since the MNIST dataset has one channel and their resolution is too low for the inception network, the resolution and channel are tripled for the evaluation $(84 \times 84 \times 3)$.

## 6. Experimental Results and Discussion

### 6.1 AC-GAN

The convergence rate of AC-GAN is so fast that it is difficult to compare the performance when the learning rate is high, a low learning rate of $3 \times 10^{-6}$ is used.

The network is trained along 50 epochs, each row has the same latent vector, and each column has the same condition.

First, to prove that AC-GAN is composed of multiple GANs and $L_{cls}^{g}$ of discriminator loss interferes with training, when there is no adversarial loss, the performance of the modified AC-GAN with $L_{cls}^{g}$ is compared with one without $L_{cls}^{g}$ in discriminator loss, in Fig. 9, Fig. 10, and Fig. 11.

As shown in Fig.9, even without adversarial loss, AC-GAN generates MNIST handwriting number data, although the quality is not very good. As $L_{cls}^{g}$ and $L_{cls}^{r}$ can be considered as the summation of adversarial losses of individual GAN, this shows that the AC-GAN consists of multiple GANs. When $L_{cls}^{g}$ is in discriminator loss, the quality of the results gets even worse than the one without it, as shown in Fig. 10. This shows that $L_{cls}^{g}$ in discriminator loss interferes with the training of each GAN. Performance comparison of those cases shown in Fig. 9 and Fig. 10 is shown in Fig. 11 in terms of FID.

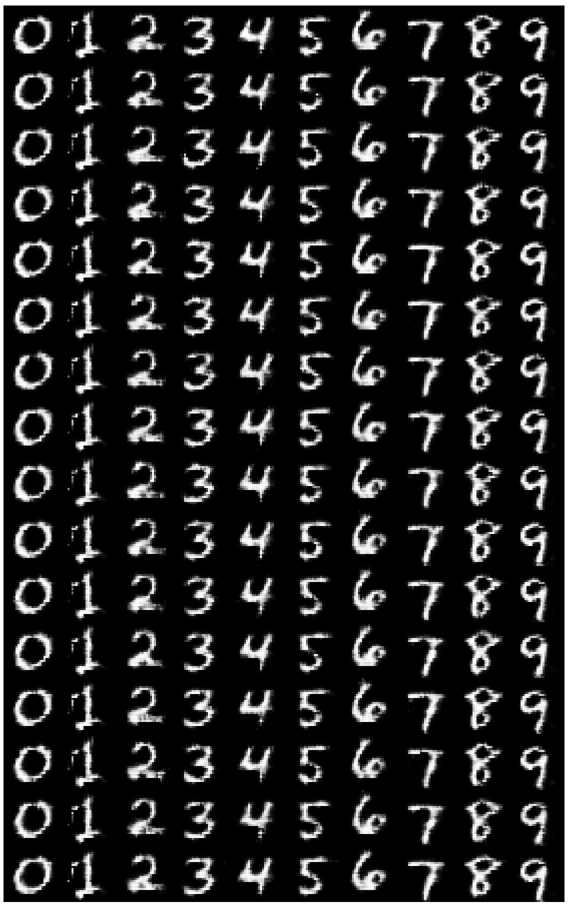


Fig.9 Results of modified AC-GAN with $L_d = L_{cls}^{r}$ and $L_g = L_{cls}^{g}$ after 50 epochs

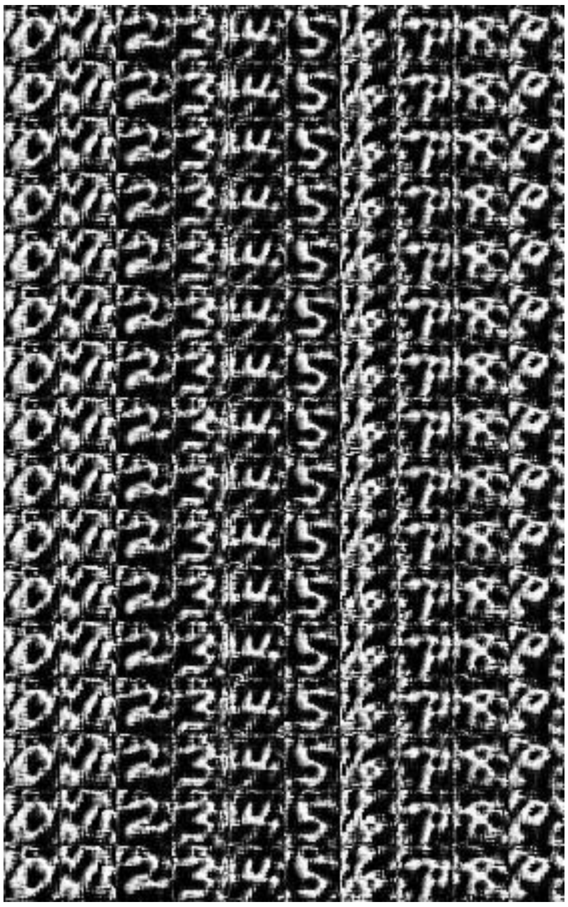


Fig.10 Results of modified AC-GAN with $L_d = L_{cls}^r + L_{cls}^g$ and $L_g = L_{cls}^g$ after 50 epochs

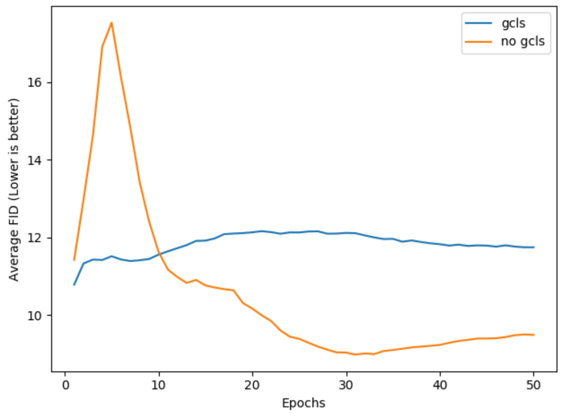


Fig.11 Effect of $L_{cls}^g$ in AC-GAN without adversarial loss

We also compared the performance of modified AC-GAN with or without $L_{cls}^g$ in discriminator loss when the adversarial loss exists.

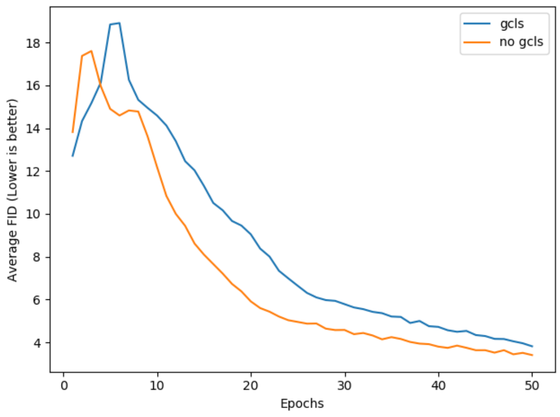


Fig.12 Effect of $L_{cls}^g$ in modified AC-GAN performance with adversarial loss ($L_{adv}^d$)

In Fig. 12, the blue graph shows the average FID of modified AC-GAN with $L_d = L_{adv}^d + \lambda_{cls}(L_{cls}^r + L_{cls}^g), \lambda_{cls} = 1$ and the orange graph shows the average FID of modified AC-GAN with $L_d = L_{adv}^d + \lambda_{cls} L_{cls}^r, \lambda_{cls} = 1$. As the graph shows, the performance of the network without $L_{cls}^g$ is better.

The next experiment is to compare the performance when the adversarial loss weight and classification loss weight are different in modified AC-GAN.

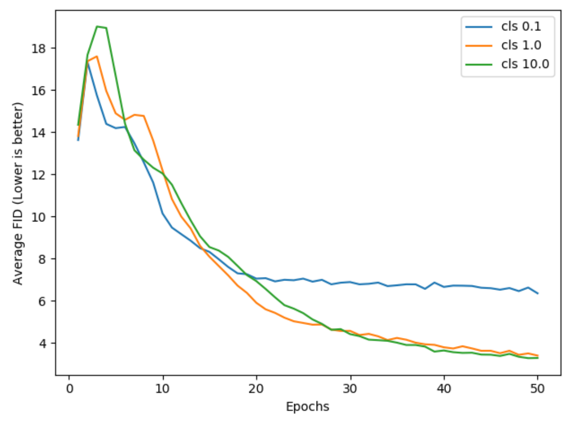


Fig.13 modified AC-GAN performance comparison with different weight of adversarial loss and classification loss

Fig.13 shows the FID when the classification loss weight $\lambda_{cls}$ varies from 0.1 to 10, with the adversarial loss weight fixed to 1.0. The changes in training speed and the quality of the results as the ratio of the adversarial loss weight and the classification loss weight changes can be easily seen through this graph.

6.2 CA-GAN

For the comparison of proposed CA-GAN with modified AC-GAN, we also used the learning rate of $3 \times 10^{-6}$ which is the same as the case of AC-GAN. The ratio of the adversarial loss weight and the classification loss weight of 1.0 is used for AC-GAN.

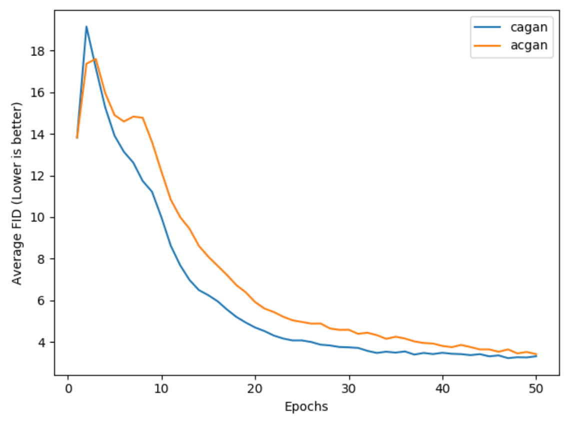


Fig.14 Performance comparison of modified AC-GAN vs CA-GAN

As shown in Fig. 14, FID of CA-GAN is lower than that of modified AC-GAN, meaning that the generated images by the CA-GAN are more realistic than the images generated by the modified AC-GAN. The performance of CA-GAN is better than modified AC-GAN.

6.3 Mixed batch training

In the original MNIST handwriting number training dataset, the number of images for each number is almost the same. For the experiment, we intentionally used a dataset consisting of 5500 of number 0 and 500 of other numbers 1~9 each from the MNIST handwriting number training dataset, to create an unbalanced dataset. The number 0 in the dataset occupies 55% of the total 10000 data, and the remaining numbers 1~9 accounts for 5% each. Since the number of data per epoch has been reduced by 1/6 compared to the experiments presented in the previous sections, the learning rate was increased to $18 \times 10^{-6}$, which is 6 times bigger than the previous learning rate.

We applied batch normalization in the discriminator in this experiment.

The effectiveness of mixed batch training in modified AC-GAN is shown in Fig. 15, and in CA-GAN is shown in Fig. 16.

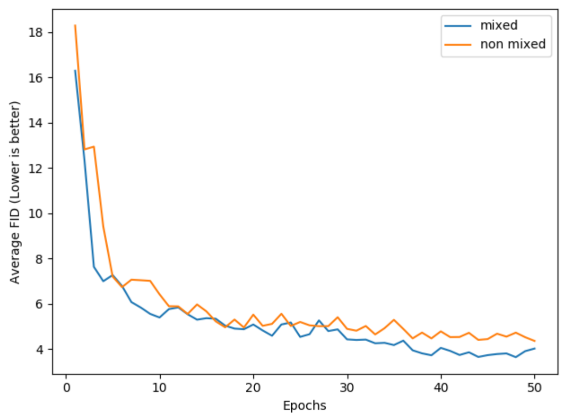


Fig.15 Mixed batch training performance comparison for modified AC-GAN

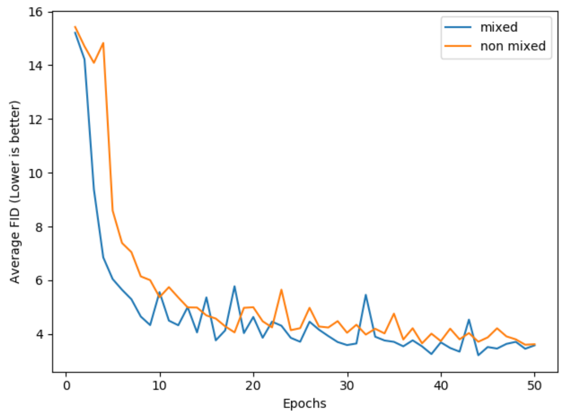


Fig.16 Mixed batch training performance comparison for CA-GAN

These figures clearly show that the mixed batch training provides better performance in general.

## 7. Conclusion

In this paper, we tried to interpret AC-GAN as a set of GANs and explained why generated data classification loss of discriminator loss in AC-GAN interferes with training and confirmed this theory through the experiments.

Based on this interpretation, we proposed a novel approach of GAN, called Conditional Activation GAN(CA-GAN). CA-GAN can be interpreted as an integration of GANs in which each individual GAN trains only one condition. Unlike modified AC-GAN, CA-GAN generates a meaningful gradient even at the beginning of the training, so that the training speed is fast, as shown in the experiments.

CA-GAN is expected to be used as a replacement for modified AC-GAN in many GAN applications because it has fewer hyperparameters and trains faster than modified AC-GAN, while it is compatible with AC-GAN.

We also predicted that the discriminator with batch normalization might use batch condition distribution to discriminate real/fake, which would cause performance degradation, in conditional GAN.

To prevent this degradation, we proposed mixed batch training. The mixed batch training is configuring each batch for discriminator with the same ratio of real data and generated data so that each batch always has the same condition distribution. Through experiments, the performance improvement of conditional GANs: modified AC-GAN and CA-GAN, due to mixed batch training is confirmed.

Mixed batch training is expected to help train conditional GANs using batch normalization for discriminators.

In conclusion, CA-GAN, which we propose in this paper, provides better performance than AC-GAN in terms of training speed and hyperparameter search. The mixed batch training also improves conditional GAN performance by inducing healthy competition between generator and discriminator.

## 8. Funding

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

## 9. References

[1] Mehdi Mirza, Simon Osindero

"Conditional Generative Adversarial Nets", arXiv preprint arXiv:1411.1784, 2014.

https://arxiv.org/abs/1411.1784 (accessed 16 February 2020)

[2] Goodfellow, Ian and Pouget-Abadie, Jean and Mirza, Mehdi and Xu, Bing and Warde-Farley, David and Ozair, Sherjil and Courville, Aaron and Bengio, Yoshua

Generative Adversarial Nets

Advances in Neural Information Processing Systems 27 (NIPS), 2014, pp. 2672-2680

https://papers.nips.cc/paper/5423-generative-adversarial-nets

[3] Takuhiro Kaneko, Kaoru Hiramatsu, Kunio Kashino

Generative Attribute Controller With Conditional Filtered Generative Adversarial Networks

The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6089-6098

http://openaccess.thecvf.com/content_cvpr_2017/html/Kaneko_Generative_Attribute_Controller_CVPR_2017_paper.html

[4] Chen, Xi and Duan, Yan and Houthooft, Rein and Schulman, John and Sutskever, Ilya and Abbeel, Pieter

InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets

Advances in Neural Information Processing Systems 29 (NIPS), 2016, pp. 2172-2180

http://papers.nips.cc/paper/6399-infogan-interpretable-representation

[5] Augustus Odena, Christopher Olah, Christopher Olah, Jonathon B Shlens, Jonathon Shlens

Conditional image synthesis with auxiliary classifier GANs

ICML'17: Proceedings of the 34th International Conference on Machine Learning – Volume 70, 2017, pp. 2642-2651

https://dl.acm.org/doi/10.5555/3305890.3305954

[6] L. Zhang, Y. Ji, X. Lin and C. Liu

Style Transfer for Anime Sketches with Enhanced Residual U-net and Auxiliary Classifier GAN

2017 4th IAPR Asian Conference on Pattern Recognition (ACPR), Nanjing, 2017, pp. 506-511.

https://ieeexplore.ieee.org/abstract/document/8575875

[7] X. Xia, R. Togneri, F. Sohel and D. Huang

Auxiliary Classifier Generative Adversarial Network With Soft Labels in Imbalanced Acoustic Event Detection

IEEE Transactions on Multimedia, vol. 21, no. 6, pp. 1359-1371, June 2019.

https://ieeexplore.ieee.org/document/8523637

[8] Prasanna Sattigeri, Samuel C. Hoffman, Vijil Chenthamarakshan, Kush R. Varshney

Gated-GAN: Adversarial Gated Networks for Multi-Collection Style Transfer

IEEE Transactions on Image Processing, vol. 28, no. 2, pp. 546-560, Feb. 2019.

https://ieeexplore.ieee.org/abstract/document/8463508

[9] Maayan Frid-Adar, Idit Diamant, Eyal Klang, Michal Amitai, Jacob Goldberger, Hayit Greenspan

GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification

Neurocomputing, Volume 321, 2018, Pages 321-331, ISSN 0925-2312,

https://www.sciencedirect.com/science/article/abs/pii/S0925231218310749

[10] Z. He, W. Zuo, M. Kan, S. Shan and X. Chen

AttGAN: Facial Attribute Editing by Only Changing What You Want

IEEE Transactions on Image Processing, vol. 28, no. 11, pp. 5464-5478, Nov. 2019.

https://ieeexplore.ieee.org/document/8718508

[11] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, Jaegul Choo

StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation

The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 8789-

8797

http://openaccess.thecvf.com/content_cvpr_2018/html/Choi_StarGAN_Unified_Generative_CVPR_2018_paper.html

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

Least Squares Generative Adversarial Networks

The IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2794-2802

http://openaccess.thecvf.com/content_cvpr_2018/html/Choi_StarGAN_Unified_Generative_CVPR_2018_paper.html

[13] Gulrajani, Ishaan and Ahmed, Faruk and Arjovsky, Martin and Dumoulin, Vincent and Courville, Aaron C

Improved Training of Wasserstein GANs

Advances in Neural Information Processing Systems 30 (NIPS), 2017, pp. 5767-5777

http://papers.nips.cc/paper/7159-improved-training-of-wasserstein-gans

[14] Sergey Ioffe, Christian Szegedy

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

Proceedings of the 32nd International Conference on Machine Learning, PMLR 37:448-456, 2015.

http://proceedings.mlr.press/v37/ioffe15.html

[dataset][15] Yann LeCun, Corinna Cortes, Christopher J.C. Burges

THE MNIST DATABASE of handwritten digits

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

[16] Alec Radford, Luke Metz, Soumith Chintala

Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

arXiv preprint arXiv:1511.06434v2 [cs.LG], 2015

https://arxiv.org/abs/1511.06434 (accessed 16 February 2020)

[17] Diederik P. Kingma, Jimmy Ba

Adam: A Method for Stochastic Optimization

arXiv preprint arXiv:1412.6980v9 [cs.LG], 2014

https://arxiv.org/abs/1412.6980 (accessed 16 February 2020)

[18] tensorflow 2.0

http://www.tensorflow.org (accessed 16 February 2020)

[19] Heusel, Martin and Ramsauer, Hubert and Unterthiner, Thomas and Nessler, Bernhard and

Hochreiter, Sepp

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

Advances in Neural Information Processing Systems 30 (NIPS), 2017, pp. 6626-6637

http://papers.nips.cc/paper/7240-gans-trained-by-a-two-t