Conditional Activation GAN: Improved Auxiliary Classifier GAN

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

Conditional GAN is a GAN that generates data with the desired condition from the latent vector. Among the variations of conditional GANs, currently, auxiliary classifier GAN is commonly used. 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 learning speed. Loss of conditional activation GAN is the sum of the loss of each GAN when GAN is created for each condition, and since each GAN shares all hidden layers, the GANs can be considered as a single GAN. Also, in order to apply batch normalization to the discriminator of conditional GANs, we propose mixed batch training, which is making a batch with the real data batch and the generated data batch.

Keywords

Auxiliary classifier GAN

Conditional GAN

1. Auxiliary classifier GAN

The loss of AC-GAN is as follows.

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

$$L_G = L_{adv}^g + L_{cls}^r + L_{cls}^g$$

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

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

$$L_{adv}^d = E_{x\sim P_r(x)}[-\log D(x)] + E_{x\sim P_g(x)}[-\log(1-D(x))]$$

$$L_{adv}^g = E_{x\sim P_g(x)}[\log(1-D(x))]$$

L_D is loss of discriminator and L_G is loss of generator. L_{adv}^d is adversarial loss of discriminator and L_{adv}^g is adversarial loss of generator.

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

D_{cls} is the auxiliary classifier of the discriminator. $-\log D_{cls}(cnd|x)$ is the cross-entropy loss between cnd and $D_{cls}(x)$.

L_{cls}^r in L_G is meaningless because the generator is not involved in the calculation of

L^{r}_{cls}.

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, uses cross-entropy adversarial loss, and shares all hidden layers.

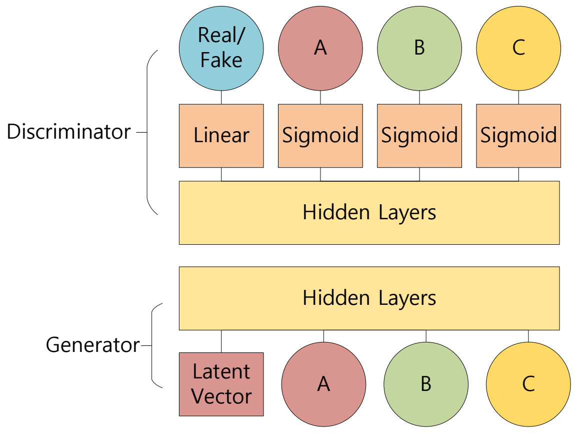


Fig1. AC-GAN that trains A, B, and C conditions

Suppose that AC-GAN training three independent conditions (A, B, C) only trains with adversarial loss, so that the real data distribution and the generated data distribution are the same. Node A of the discriminator is trained by $L^{r}_{cls}[A]$ in L_D to output real (1) when it receives real data with condition A and fake (0) with condition not-A. When the generator receives condition A (1) as its node A's input, attempts by $L^{g}_{cls}[A]$ in L_G to generate data with condition A, and trains the discriminator's node A output to be real (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 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 generate 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 real (1) 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 not-A that generates data with condition not-A.

AC-GAN uses cross-entropy loss as 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-GAN used in StarGAN [10] and AttGAN [11] is as follows.

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

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

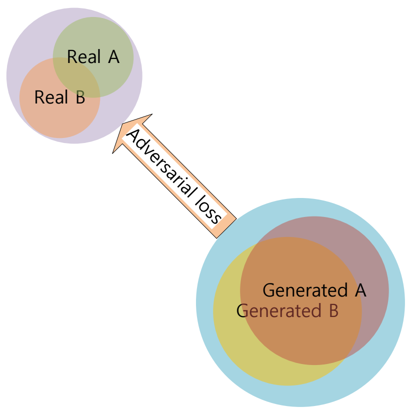
$$L^{r}_{cls} = E_{x,cnd \sim P_r(x,cnd)}[-\log D_{cls}(cnd|x)]$$

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

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 $x, cnd \sim P_r(x, cnd)$, x is real data, and cnd is the binary vector that expresses the conditions of real data. In $x', cnd' \sim P_g(x', cnd')$, x' is generated data and cnd' is the target binary vector to generate x'. λ_{cls} is classification loss weight.

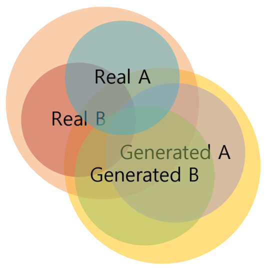
As explained above, modified AC-GAN can be considered as a group of GANs. However, each GAN training each condition can be trained as a GAN only if the real data distribution and the generated data distribution are the same.



Real X: Real data distribution with attribute X
Generated X: Generated data distribution to have attribute X

Fig2. Data distribution at the beginning of training using modified 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.1.



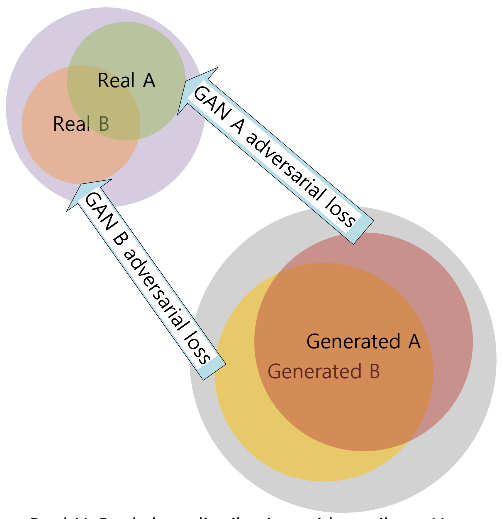
Real X: Real data distribution with attribute X
Generated X: Generated data distribution to have attribute X

Fig3. After some training using modified AC-GAN

As training with adversarial loss progresses, when the real data distribution and the generated data distributions become somewhat similar, classification loss begins to produce meaningful gradients, and training is performed to generate data with each condition.

2. Conditional activation GAN

Modified AC-GAN has the disadvantage of requiring one additional hyperparameter to adjust the ratio of adversarial loss and classification loss and not producing meaningful gradients early in learning. To solve these problems of modified AC-GAN, we propose conditional activation GAN (CA-GAN), which is similar to having multiple GANs that each GAN trains each 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. Each GAN trains only one condition.

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

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

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

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

cnd is conditions what CA-GAN wants to train. c is one specific condition in cnd. GAN c is the GAN that train about 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 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 condition 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 condition 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.

$$L_{ca}^{D} = E_{x,cnd\sim P_r(x,cnd)}[f_r^D(D,x)\cdot cnd] + E_{x',cnd'\sim P_g(x',cnd')}[f_g^D(D,x')\cdot cnd']$$

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

In $x,cnd\sim P_r(x,cnd)$, x is real data, and cnd is the binary vector that expresses the conditions of real data. In $x',cnd'\sim P_g(x',cnd')$, x' means generated data, and cnd' 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.

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

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

In AC-GAN, GAN A that trains condition A generates data with condition not-A as well as data with condition A. However, in CA-GAN, since GAN A training condition A does not care about condition not-A, a new GAN training condition not-A must be added to train condition not-A.

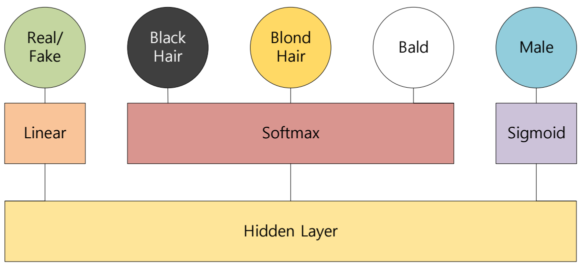


Fig5. AC-GAN discriminator output example

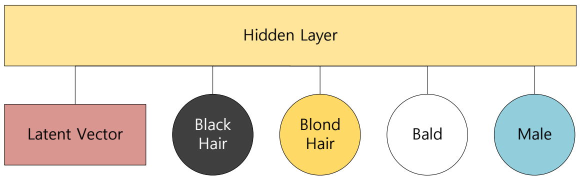


Fig6. AC-GAN generator input example

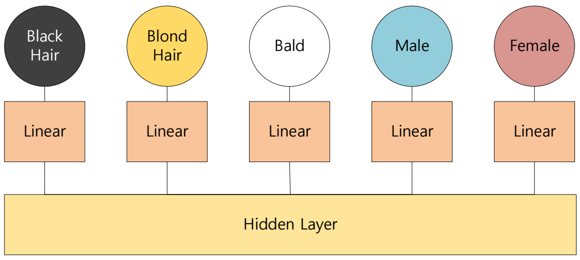


Fig7. conditional activation GAN discriminator output example

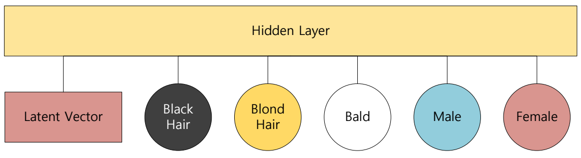


Fig8. conditional activation GAN generator input example

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

In CA-GAN, since each GAN can be trained 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.

3. Mixed batch training

In conditional GANs, training by applying batch normalization to the discriminator induces the generator to distort the input condition distribution. When batch normalization is applied to the discriminator and the input target condition distribution used for training and the real data condition distribution are different, the discriminator uses the batch condition distribution for real/fake discriminate, so 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 configure each batch always with the same ratio of real data and generated data so that each batch always has the same condition distribution. If input batch of discriminator always has the same condition distribution, the discriminator will not discriminate real/fake by condition distribution, and the generator will not ignore the input target condition distribution and will not attempt to follow the real data condition distribution.

4. Material and methods

Used train dataset of MNIST handwriting number dataset [15] for the train. The data size is 60000, a resolution is 28x28, and the channel size is 1. The model architecture used the basic design of DCGAN [16]. Used instance normalization [17] for normalization. Used LSGAN adversarial loss and Adam optimizer [18]. Trained for 50 epochs.

Used tensorflow2.0 for implement.

Used an average of FID [19] for each condition for evaluation. Used all test data for calculating FID. All experiments were performed three times and used the average of the results.

when evaluate, generated data size is the 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).

In all pictures, trained 50 epochs, each row has the same latent vector, and each col has the same condition.

5. Results and Conclusions

5.1 AC-GAN

Used learning rate 3e-6. When using a high learning rate, the convergence is so fast that it is difficult to compare, so used a low learning rate. Used instance normalization in discriminator.

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

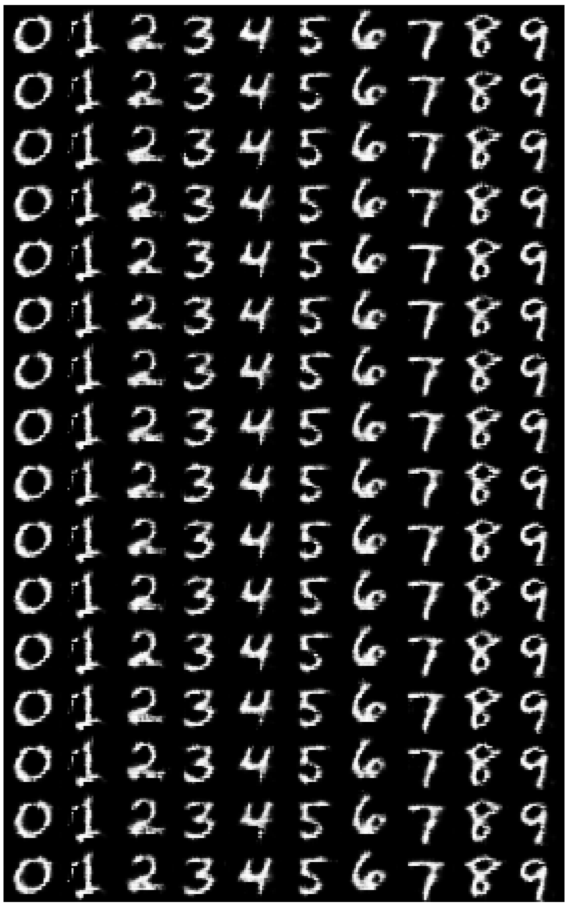


Fig.9 Modified AC-GAN without L_{cls}^{g} in discriminator loss and adversarial loss epoch 50 results

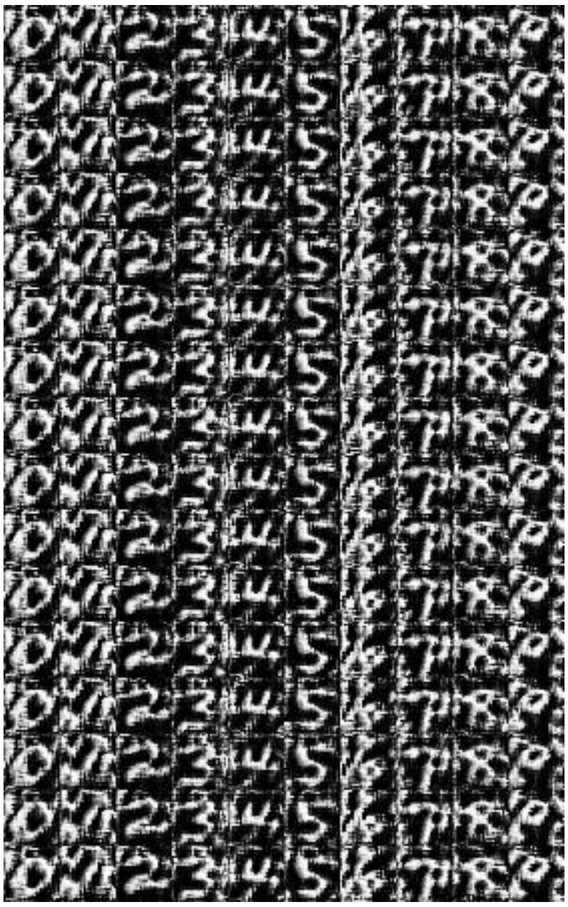


Fig.10 Modified AC-GAN with L^g_{cls} in discriminator loss and without adversarial loss

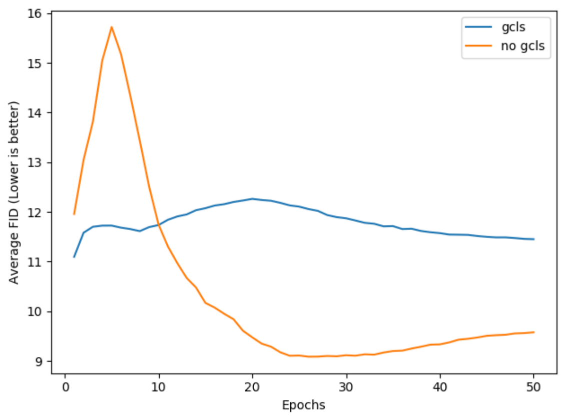


Fig.11 AC-GAN without adversarial loss performance comparsion

In Fig.9, although modified AC-GAN has no adversarial loss, it generates MNIST handwriting number data, although the quality is not good. This means that the modified AC-GAN can be considered as a group of multiple GANs. Also, when L^g_{cls} is in discriminator loss, the quality of the results are not good as it does not exist. This shows that L^g_{cls} in discriminator loss interferes with the training of each GAN.

Next, compared the performance of AC-GAN with or without L^g_{cls} in discriminator loss.

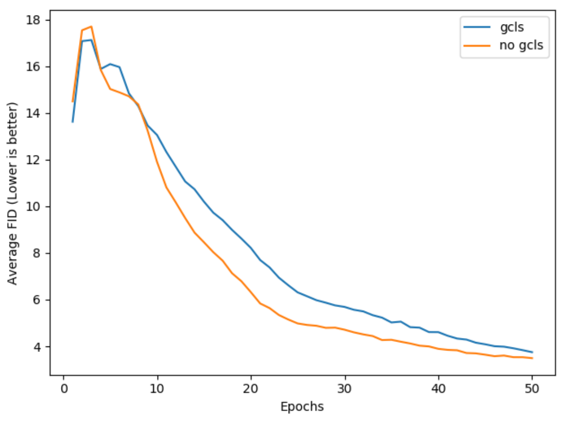


Fig.12 AC-GAN performance comparsion

Better performance without L^g_{cls}.

Next, compared the performance difference by ratio of adversarial loss weight and classification loss weight.

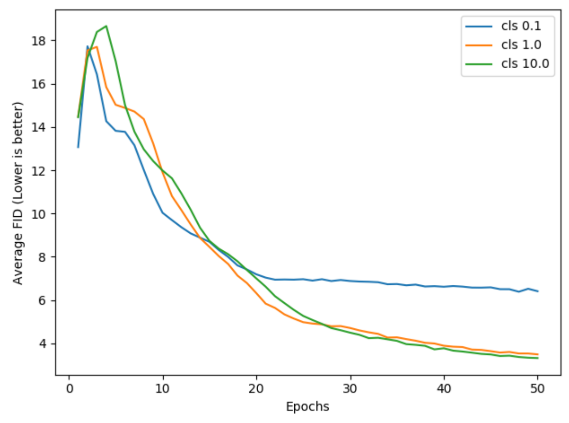


Fig.13 AC-GAN performance comparision

Fig.13 shows the performance difference by classification loss weight when the adversarial loss weight is 1.0. It can be seen that the speed of training and the quality of the results vary depending on the ratio of the adversarial loss weight and the classification loss weight

6.2 CA-GAN

Used learning rate 3e-6. Used instance normalization in discriminator.

First, compared AC-GAN and CA-GAN. AC-GAN has a 1:1 ratio of adversarial loss weight to classification loss weight.

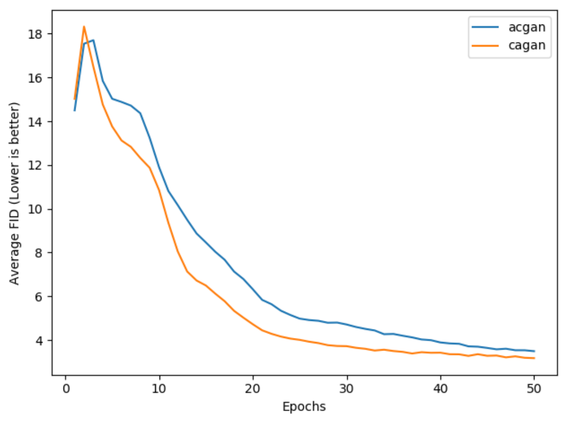


Fig.14 AC-GAN vs CA-GAN

The performance of CA-GAN is better than AC-GAN.

6.3 Mixed batch training

Used batch normalization in the discriminator.

In the original MNIST handwriting number training dataset, the ratio of each number is almost the same, but for the experiment, we used a dataset consisting of 5500 of number 0 and 500 of other numbers 1~9 each in MNIST handwriting number training dataset. That is, 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 with the previous experiments, the learning rate was increased to 18e-6, which is 6 times the previous learning rate.

First, in AC-GAN, compared the performance with and without mixed batch training.

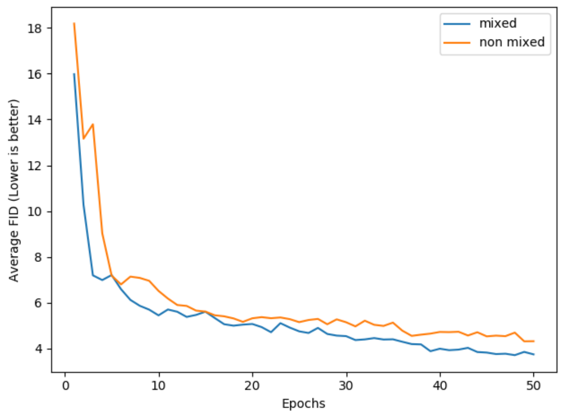


Fig.15 AC-GAN mixed batch training performance comparison

Similarly, compared the performance in CA-GAN.

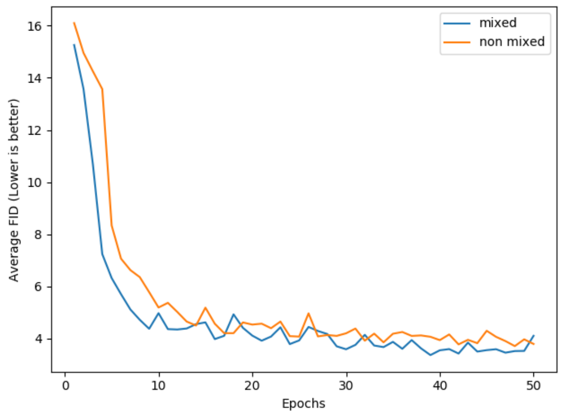


Fig.16 CA-GAN mixed batch training performance comparison

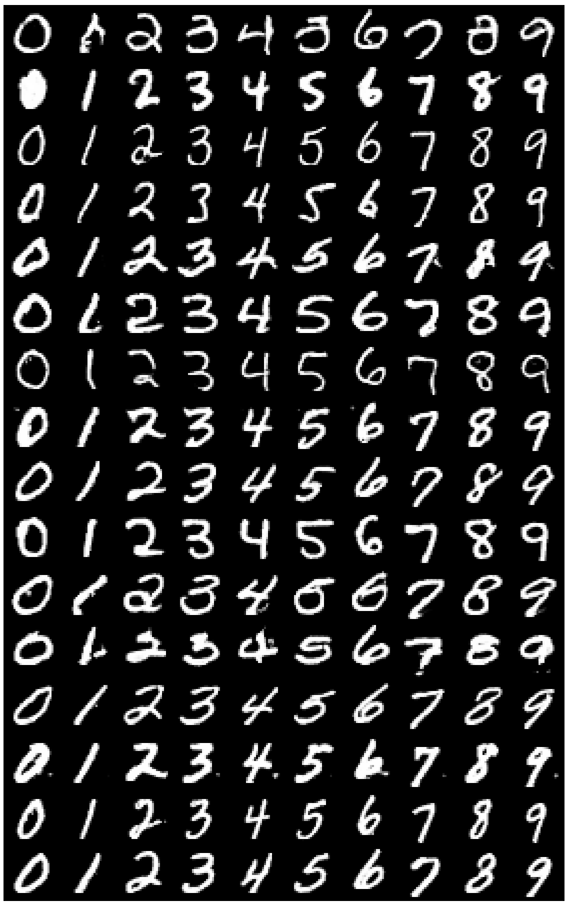
Both AC-GAN and CA-GAN show better performance when using mixed batch training.

6. Funding

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

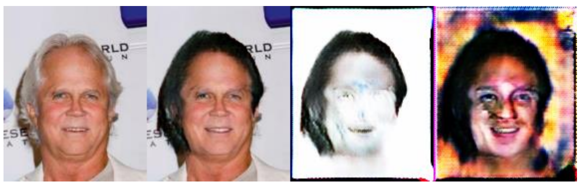
7. Appendix

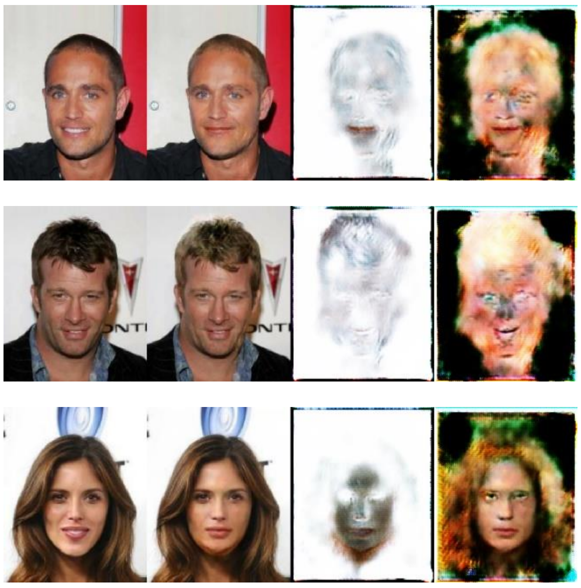
7.1 CA-GAN results



7.2 CASL-GAN

These are the results of CASL-GAN (Image-to-image translation GAN, http://vixra.org/abs/1909.0061?ref=10946100), which is using conditional activation GAN loss. All first pictures are original pictures, second pictures are generated pictures, third pictures are mask images, and fourth pictures are generated segment images.





8. References

[1] Mehdi Mirza, Simon Osindero

"Conditional Generative Adversarial Nets", arXiv preprint arXiv:1411.1784, 2014. https://arxiv.org/abs/1411.1784 (accessed 13 January 2020)

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

Generative Adversarial Nets

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

[3] Takuhiro Kaneko, Kaoru Hiramatsu, Kunio Kashino; 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] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, Pieter Abbeel

InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets

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

[5] Augustus Odena, Christopher Olah, Jonathon Shlens

Conditional Image Synthesis With Auxiliary Classifier GANs

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

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

[6] Lvmin Zhang, Yi Ji, Xin Lin

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

https://arxiv.org/abs/1706.03319

[7] Ayushman Dash, John Cristian Borges Gamboa, Sheraz Ahmed, Marcus Liwicki, Muhammad Zeshan Afzal

TAC-GAN - Text Conditioned Auxiliary Classifier Generative Adversarial Network

https://arxiv.org/abs/1703.06412

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

Fairness GAN

https://arxiv.org/abs/1805.09910

[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

https://arxiv.org/abs/1803.01229

[10] Zhenliang He, Wangmeng Zuo, Meina Kan, Shiguang Shan, Xilin Chen

AttGAN: Facial Attribute Editing by Only Changing What You Want

https://arxiv.org/abs/1711.10678

[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

Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, Jaegul Choo; 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

Xudong Mao, Qing Li, Haoran Xie, Raymond Y.K. Lau, Zhen Wang, Stephen Paul Smolley; The IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2794-2802

http://openaccess.thecvf.com/content_iccv_2017/html/Mao_Least_Squares_Generative_ICCV_2017_paper.html

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

Improved Training of Wasserstein GANs

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

https://arxiv.org/abs/1502.03167

[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

https://link.springer.com/chapter/10.1007/978-3-319-71589-6_9

[17] Dmitry Ulyanov, Andrea Vedaldi, Victor Lempitsky

Instance Normalization: The Missing Ingredient for Fast Stylization

https://arxiv.org/abs/1607.08022

[18] Diederik P. Kingma, Jimmy Ba

Adam: A Method for Stochastic Optimization

https://arxiv.org/abs/1412.6980

[19] 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

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