Training self-supervised class-conditional GAN with virtual labels

Jeongik Cho
jeongik.jo.01@gmail.com

Abstract

Class-conditional GAN is a conditional GAN that can generate class-conditional distribution. Among class-conditional GANs, InfoGAN with categorical latent distribution can generate class-conditional data through a self-supervised (unsupervised) method without a labeled dataset. Instead, InfoGAN requires optimal categorical latent distribution to train the model.

In this paper, we propose a novel GAN that allows the model to perform self-supervised class-conditional data generation and clustering without knowing the optimal categorical latent distribution. The proposed method uses three different losses. The first loss is the cross-entropy classification loss to predict the label of the fake data. The classifier is trained with the classification loss. The second loss is the CAGAN loss for class-conditional data generation. The virtual label of the real data predicted by the classifier is used for CAGAN loss. The generator and discriminator are trained with CAGAN loss. The third loss is the classifier gradient penalty loss. The classifier gradient penalty loss regularizes the slope of the classifier's decision boundary so that the decision boundary converges to a better local optimum.

Additionally, the proposed method updates the categorical latent distribution with the output distribution of the classifier on the real data. As training progresses, the entropy of the categorical latent distribution gradually decreases by the classifier gradient penalty loss and converges to the appropriate value. The converged categorical latent distribution becomes appropriate to represent the discrete part of the data distribution.

The proposed method does not require labeled data, optimal categorical latent distribution, and a good metric to calculate the distance between data.

1 Introduction

Class-conditional GAN is a conditional GAN [2] that can generate class-conditional distribution with a labeled dataset. In general, the generator of class-conditional GAN takes a continuous latent distribution and a discrete categorical distribution as inputs and generates class-conditional distribution. ACGAN [3] and CAGAN [4] are examples of class-conditional GANs. However, these class-conditional GANs can only be trained given labels, which is the conditional categorical distribution of the dataset. Therefore, these methods cannot be utilized with unlabeled datasets.

Unlike ACGAN or CAGAN, InfoGAN [5] with categorical latent distribution can generate class-conditional data distribution even if the data is not labeled. However, InfoGAN requires optimal categorical latent distribution. It includes the number of categories and the probability for each category. For example, to perform class-conditional data generation with InfoGAN on the MNIST handwritten digits dataset [9] without labels, InfoGAN needs to know the number of categories (10 categories) and the probability of each category (0.1 for each category) for categorical latent distribution.

In this paper, we introduce a Virtual Conditional Activation GAN (VCAGAN) that is capable of generating class-conditional data without being given labels and optimal categorical latent distribution.

A VCAGAN consists of a discriminator, classifier, and class-conditional generator. The discriminator has d_c-dimensional output like CAGAN (d_c represents the dimensionality of the categorical latent distribution). The classifier can share all hidden layers with the discriminator. The class-conditional generator takes d_z-dimensional continuous latent distribution and d_c-dimensional categorical latent distribution as inputs to generate class-conditional data.

VCAGAN uses three different losses. The first loss is the cross-entropy classification loss to predict the label of the fake data. The classifier is trained with the classification loss. The second loss is the CAGAN loss for class-conditional data generation. The label of the real data for CAGAN loss is predicted from the classifier. The generator and discriminator are trained with CAGAN loss. The third loss is the classifier gradient penalty loss. The classifier gradient penalty loss regularizes the slope of the classifier's decision boundary so that the decision boundary converges to a better local optimum.

In addition, VCAGAN updates the categorical latent distribution with the classifier output distribution of real data. This makes the categorical latent distribution of the fake data similar to that of the real data.

There are several differences between InfoGAN and VCAGAN. The first difference is that the VCAGAN uses CAGAN loss as the adversarial loss and does not use classification loss for the generator. The generator of InfoGAN is trained to minimize both adversarial loss and classification loss like ACGAN. This lowers the generator performance of InfoGAN because adversarial loss and classification loss in the generator conflict with each other. On the other hand, since VCAGAN's generator is only trained with CAGAN loss (only adversarial losses), there is no conflict between adversarial loss and classification loss in the generator.

Second, VCAGAN gradually changes the categorical latent distribution during training. VCAGAN updates the categorical latent distribution to follow the classifier output distribution for real data. This allows VCAGAN to approximate the optimal categorical latent distribution without knowing it, unlike InfoGAN.

Third, VCAGAN can adjust the sensitivity of the clusters through classifier gradient penalty loss. Without the classifier gradient penalty loss, the gradient of the VCAGAN classifier's decision boundary can become very large, and the decision boundary will fall into a local optimum in a narrow region. The larger the classifier gradient penalty loss weight of VCAGAN, the decision boundary of the classifier falls into the local optimum of a larger region, and the entropy of the categorical latent distribution decreases. In other words, VCAGAN can adjust the sensitivity of the clusters through the weight (multiplier) of the classifier gradient penalty loss.

2 Class-conditional data generation

Typically, when training a GAN, everything is assumed to be continuous. This means that the data distribution and latent distribution are assumed to be continuous, and the generator and discriminator of GAN are assumed to be continuous functions (deep learning models must be differentiable continuous functions to be trained).

However, the data distribution is not necessarily continuous. When data distribution includes a discrete part and latent distribution is continuous, a sufficiently complex deep generative model can approximate the discrete part of the data distribution. However, approximating the discrete part of the data distribution is still not easy for most deep generative models, which is a continuous function.

The left part of Fig. 1 shows a data distribution example consisting of three Gaussian clusters. There is no perfect discrete part in this data distribution (i.e., data distribution is still continuous), but one can see that it is easier to represent this data distribution with a discrete (categorical) latent distribution.

The right part of Fig. 1 shows generated data with GAN trained only with a continuous latent distribution. One can see that the model generates lines connecting each cluster. This is because the latent distribution is continuous, and the generator is a continuous function, making it difficult to represent the discrete part of the data distribution. As training progresses, the probability density of the line connecting the clusters decreases, but it requires a long training period, and it is hard to say that the continuous latent distribution correctly represents the real data distribution.

For datasets with discrete parts, such as the example in Fig. 1, using a discrete latent distribution is more appropriate for model training and data representation. Class-conditional generative models, such as ACGAN [3] or CAGAN [4], take both continuous latent distribution and discrete categorical latent distribution as inputs and generate class-conditional data distribution. It makes them appropriate for representing datasets with discrete parts. However, ACGAN and CAGAN cannot be trained if there is no label for data.

InfoGAN [5] can perform class-conditional data generation and inversion (clustering) by maximizing mutual information of generator input categorical latent distribution and classifier output distribution, even if the data is not labeled. Following equations show losses for InfoGAN with the categorical latent

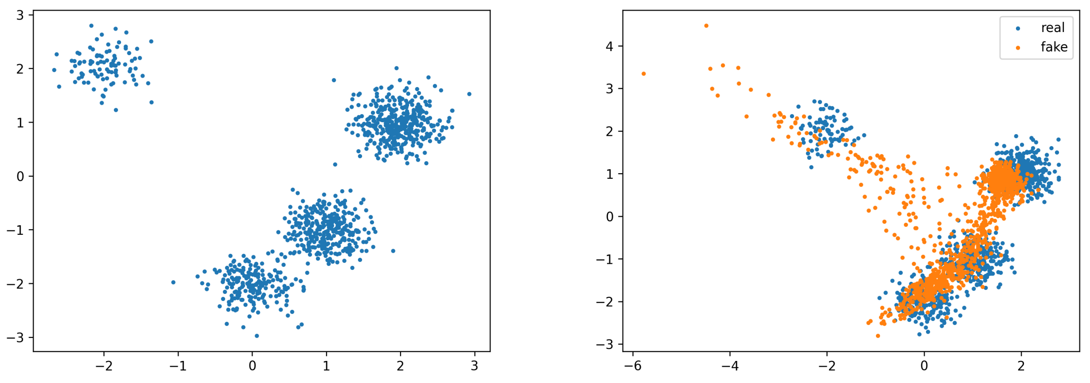

Figure 1: Left part: Two-dimensional dataset consisting of four Gaussian clusters. The centers and probabilities of each cluster are $(-2.0, 2.0), (2.0, 1.0), (0.0, -2.0), (1.0, -1.0)$ and $[0.1, 0.4, 0.2, 0.3]$, respectively. The standard deviation for all clusters is 0.3. Right part: Samples generated by GAN trained only with a continuous latent distribution.

distribution.

$$L_{cls} = \mathbb{E}_{z,c}\left[cross\ entropy(c, Q(G(z,c)))\right] \tag{1}$$

$$L_d = L^d_{adv} + \lambda_{cls} L_{cls} \tag{2}$$

$$L_g = L^g_{adv} + \lambda_{cls} L_{cls} \tag{3}$$

Eq. 2 and Eq. 3 show the discriminator loss and generator loss of InfoGAN, respectively. In Eqs. 2 and 3, L^d_{adv} and L^g_{adv} represent adversarial losses [6] for GAN training. L_{cls} and λ_{cls} represent classification loss and classification loss weight, respectively. In Eq. 1, L_{cls} is cross entropy between categorical latent code c and predicted label of generated data $Q(G(z,c))$. Q and G represent the classifier and generator, respectively. z represents continuous latent code sampled from the continuous latent distribution Z. In general, a classifier Q shares all its layers with a discriminator D for efficiency.

From the above equations, one can see that a discriminator (more precisely, a classifier integrated with a discriminator) and a generator are trained to minimize classification loss. InfoGAN has shown that, given an appropriate categorical latent distribution C, it can perform class-conditional data generation and clustering (inversion) even when the data is unlabeled.

However, InfoGAN still needs insight into the categorical latent distribution C. Without knowing the appropriate categorical latent distribution C, InfoGAN still cannot perform class-conditional data generation and clustering.

Additionally, InfoGAN's generator is trained with both adversarial loss and classification loss. It means that adversarial loss and classification loss conflict with each other in the generator. This conflict reduces the generative performance of InfoGAN.

In this paper, we introduce VCAGAN, which performs class-conditional data generation and clustering under more general conditions than InfoGAN. We assume the following general conditions:

1. All data is unknown to which cluster it belongs (i.e., there are no labels for all data).
2. Optimal categorical latent distribution is unknown.
3. Metric to measure the distance between the data is unknown.

Under these conditions, ACGAN and CAGAN cannot be used due to condition 1. InfoGAN cannot be used due to condition 2, and recent methods utilizing the K-means algorithm cannot be used due to condition 3. On the other hand, VCAGAN can still perform class-conditional data generation and clustering (inversion) even under these conditions.

3 Virtual Conditional Activation GAN

VCAGAN uses three different losses. Following equations show losses for training VCAGAN.

$$L^d_{adv} = \mathbb{E}_{z,c,x}\left[f_d(D(x) \cdot argmax\ onehot(Q(x)), D(G(z,c)) \cdot c)\right] \tag{4}$$

$$L^g_{adv} = \mathbb{E}_{z,c}\left[f_g(D(G(z,c)) \cdot c)\right] \tag{5}$$

$$L_{creg} = \mathbb{E}_x\left[\|\nabla(Q(x) \cdot argmax\ onehot(Q(x)))\|^2_2\right] \tag{6}$$

$$L_d = L^d_{adv} + \lambda_{cls}L_{cls} + \lambda_{creg}L_{creg} \tag{7}$$

$$L_g = L^g_{adv} \tag{8}$$

Eqs. 4 and 5 show CAGAN adversarial losses for VCAGAN. f_d and f_g represent adversarial loss functions for discriminator and generator, respectively. Operation "$\cdot$" represents the inner product. Since the true label c of the fake data $G(z,c)$ is known, the adversarial loss for fake data in VCAGAN is the same as CAGAN loss. However, the label of the real data x is unknown. Thus, in VCAGAN, $argmax\,onehot(Q(x))$ is used as the label of the real data x. The $argmax\,onehot$ function replaces the maximum value of the vector with 1 and all other values with 0 (e.g., $argmax\ onehot([0.2, 0.5, 0.3]) = [0.0, 1.0, 0.0]$).

Eq. 6 shows classifier gradient penalty loss for VCAGAN. As with adversarial loss, $argmaxonehot(Q(x))$ is used as the label of real data x. With CAGAN adversarial loss and classification loss, the classifier is trained so that the decision boundary falls on the local optimum that minimizes $P(X)$. However, the classifier may only increase the slope of the decision boundary to minimize classification loss. In such a case, the decision boundary of the classifier will converge to a local optimum in a very narrow region. To avoid this and ensure that the classifier's decision boundary falls on the local optimum of a larger region that minimizes $P(X)$, VCAGAN uses a classifier gradient penalty loss L_{creg}.

In Eq. 7, λ_{cls} and λ_{creg} represent weighting values for each loss. Classification loss L_{cls} is the same as Eq. 1. In Eq. 8, one can see that there is no classification loss L_{cls} in generator loss L_g.

Additionally, VCAGAN updates the probability of the categorical latent distribution during training to approximate $\mathbb{E}_x[Q(x)]$ (i.e., $P(C) \approx \mathbb{E}_x[Q(x)]$).

Algorithm 1 Training step of VCAGAN

```
Require: $X, Z, C, D^*, G$
 1: $x \leftarrow sample(X)$
 2: $z \leftarrow sample(Z)$
 3: $c_f \leftarrow sample(C)$

 4: $x' \leftarrow G(z, c_f)$
 5: $a_f, c'_f \leftarrow D^*(x')$
 6: $a_r, c'_r \leftarrow D^*(x)$
 7: $c_r \leftarrow argmax\ onehot(c'_r)$

 8: $L_{cls} \leftarrow cross\ entropy(c_f, c'_f)$
 9: $L_{creg} \leftarrow \|gradient(c_r \cdot c'_r, x)\|^2_2$

10: $L_d \leftarrow f_d(a_r \cdot c_r, a_f \cdot c_f) + \lambda_{cls}L_{cls} + \lambda_{creg}L_{creg}$
11: $L_g \leftarrow f_g(a_f \cdot c_f)$

12: $C \leftarrow update(C, c'_r)$

13: $return\ L_d, L_g, C$
```

Algo. 1 shows the training steps of VCAGAN.

The training step of VCAGAN requires X (data random variable), Z (continuous latent random variable), C (categorical latent random variable), D^* (discriminator integrated with classifier), and G (generator).

In lines 1-3, the *sample* function represents the sampling function from a random variable. x (real data point), z (continuous latent code), and c_f (fake categorical latent code) are sampled from X, Z, and C, respectively.

In line 4, G generates fake data x' with z and c_f. In line 5, D^* takes a fake data point x' as input and outputs two values a_f (fake adversarial value) and c'_f (fake categorical latent code prediction). Since the discriminator and classifier are integrated, the integrated discriminator D^* outputs a d_c-dimensional adversarial vector and d_c-dimensional categorical latent code prediction. a_f and c'_f represent fake adversarial vector and predicted label of c_f, respectively. Similarly, in line 6, D^* takes a real data point x as input and outputs adversarial vector a_r and real data categorical latent code prediction c'_r. In line 7, real categorical latent code c_r is calculated from c'_r.

In line 8, L_{cls} represents classification loss. *crossentropy* is a function that calculates cross-entropy loss. In line 9, $gradient(y, x)$ function calculates slope dy/dx.

In lines 10 and 11, L_d and L_g represent discriminator loss and generator loss, respectively. f_d and f_g represent adversarial functions for GAN.

In line 12, C is updated with predicted real categorical latent code c'_r. The *update* function can be a simple moving average, an exponential moving average, or others.

After line 13, D^* and G are updated with losses L_d and L_g, respectively.

4 Experiments

We trained the models to generate two-dimensional Gaussian clusters distribution and the MNIST dataset [9]. In Gaussian clusters experiments, we compare the performance of Vanilla GAN [1], InfoGAN [5], and our proposed VCAGAN. In MNIST experiments, we compared the clustering of VCAGANs according to classifier gradient penalty loss weight λ_{creg}.

The following hyperparameters were used for experiments.

$$\begin{gathered} Z \sim N(0, I_{d_z}) \\ \lambda_{r1} = 1.0 \\ optimizer = Adam \begin{pmatrix} learning\ rate = 0.001 \\ \beta_1 = 0.0 \\ \beta_2 = 0.99 \end{pmatrix} \\ batch\ size = 32 \\ train\ step\ per\ epoch = 1000 \end{gathered}$$

λ_{r1} represents R1 regularization [7] loss weight. Classification loss weight $\lambda_{cls} = 1.0$ was used for InfoGAN and VCAGAN. We used exponential moving average with $decay\ rate = 0.999$ as *update* function for VCAGAN. Equalized learning rate [8] was used for all weights.

4.1 Gaussian clusters experiments

In this experiment, we used the dataset consisting of four 2-dimensioanl Gaussian clusters as a training dataset. Left part of Fig. 1 shows data distribution for the experiments. One can see that there are four Gaussian clusters with different probabilities in data distribution. A generator and discriminator consisting of four fully connected layers with 512 units were used for training. We used $d_z = 16$, $d_c = 8$, and $epoch = 50$ for model training.

Fig. 2 shows samples generated with vanilla GAN (trained only with adversarial loss). The vanilla GAN is trained to generate data given a continuous latent distribution and a categorical latent distribution. One can see that the generator of vanilla GAN did not use categorical latent distribution, and training was exclusively performed only with a continuous latent distribution. Therefore, the generator output was also continuous, which caused a line generation between each cluster. For example, one can see a distinct line between the cluster $(1, -1)$ and cluster $(2, 1)$. One can also see some samples between the cluster $(-2, 2)$ and the other clusters.

Fig. 3 shows samples generated with InfoGAN. Unlike the Vanilla GAN, one can see that the model generates class-conditional distribution with the categorical latent distribution. However, the class-conditional distribution was not correct, even with the optimal categorical latent distribution. First, in the left part of Fig. 3 (4-dimensional optimal categorical latent distribution), one can see that

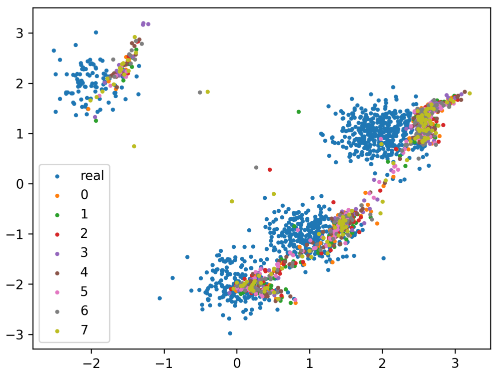

Figure 2: Vanilla GAN (trained only with adversarial loss) is trained with both continuous latent distribution and categorical latent distribution. Each category has the same probability.

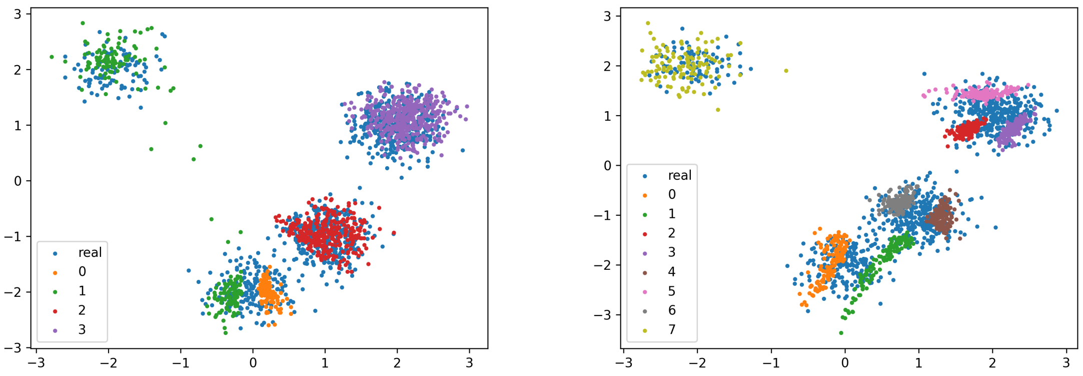

Figure 3: InfoGAN trained with the categorical latent distribution. Left: 4-dimensional optimal categorical latent distribution ($P(C) = [0.1, 0.2, 0.3, 0.4]$). Right: 8-dimensional categorical latent distribution ($P(C) = [0.125, 0.125, ..., 0.125]$).

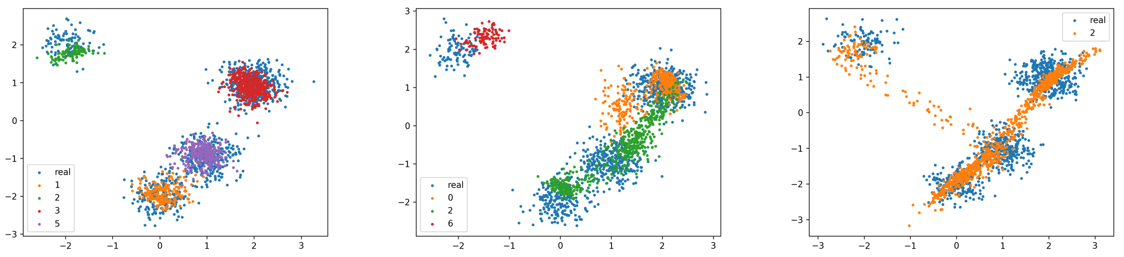

Figure 4: VCAGAN trained with different λ_{creg}. Categories with a probability of less than 1% were ignored. Left: $\lambda_{creg} = 0.1$. $P(C) = [0.1970, 0.0990, 0.4035, 0.3005]$. Middle: $\lambda_{creg} = 1.0$. $P(C) = [0.3985, 0.5024, 0.0991]$. Right: $\lambda_{creg} = 10.0$. $P(C) = [0.9999]$.

category 1 was separated with cluster $(-2, 2)$ and cluster $(0, -2)$. Also, since one category has been split into two clusters, one can see line generation between those clusters.

Furthermore, one can see that no data was generated between categories 0 and 1 in cluster $(0, -2)$. This is because InfoGAN's generator does not generate data near the decision boundary of the classifier to minimize classification loss. This shows that classification loss and adversarial loss are in conflict in the generator of InfoGAN.

In the right part of Fig. 3, one can see that each cluster has multiple categories in InfoGAN. Since there are multiple categories in a cluster, one can see that, as before, no data is generated near the decision boundary of the classifier.

Fig. 4 shows samples generated by VCAGAN trained with different λ_{creg}. In the left part of Fig. 4, one can see that VCAGAN creates four Gaussian clusters almost perfectly. There are no lines between clusters like in Vanilla GAN or InfoGAN, and the probability of each cluster is almost the same as in the original dataset. Additionally, categories 3 and 5, despite their proximity to each other, do not have a clear decision boundary like InfoGAN. This is because VCAGAN's generator is only trained with adversarial loss, not classification loss.

In the middle and right part of Fig. 4, one can see that the number of categories decreases as λ_{creg} increases. This is because as the classifier gradient penalty loss weight gets larger, the decision boundary of the classifier falls into the local optimum (minimize $P(X)$) over a wide region. This results in multiple clusters falling into the same category.

4.2 MNIST experiments

In this section, we trained VCAGAN to generate the MNIST handwritten digits dataset. The generator and discriminator are simply composed of CNNs. $d_z = 128$, $d_c = 16$, $epoch = 200$ were used for the experiments.

Figs. 5, 6, and 7 show the difference in clustering of VCAGAN according to λ_{creg}.

First, in Fig. 5, because $\lambda_{creg} = 1$ was too low, the classifier decision boundary converged on a local optimum in a narrow region. Thus, similar clusters were split into two or more categories. One can see that number 8 (category 0), number 6 (category 3), and number 3 (category 6) were correctly clustered into one category. For other numbers, they were divided into multiple categories (number 1 into categories $4, 7, 8$, number 4 into categories $1, 9$, number 2 into categories $2, 11$, etc).

However, as λ_{creg} increases, the classifier decision boundary converges to the local optimum over a wider region. In Fig. 6, $\lambda_{creg} = 5$ was used for training. One can see that number 2 (category 1), number 0 (category 2), number 6 (category 3), number 7 (category 4), number 4 (category 8), number 9 (category 9) were correctly clustered into one category. However, the number 1 was still split into categories 0 and 7.

In Fig. 7, $\lambda_{creg} = 10$ was used for training. Therefore, more clusters have been consolidated into the same category. One can see that number 1 (category 0), number 6 (category 1), number 0 (category 2), number 2 (category 5) were clustered into one category. Numbers $3, 5, 8$ were clustered into category 3, and numbers $4, 7, 9$ were clustered into category 4. This means that the perceptual distances between the numbers $3, 5, 8$ and the numbers $4, 7, 9$ are closer than the other numbers.

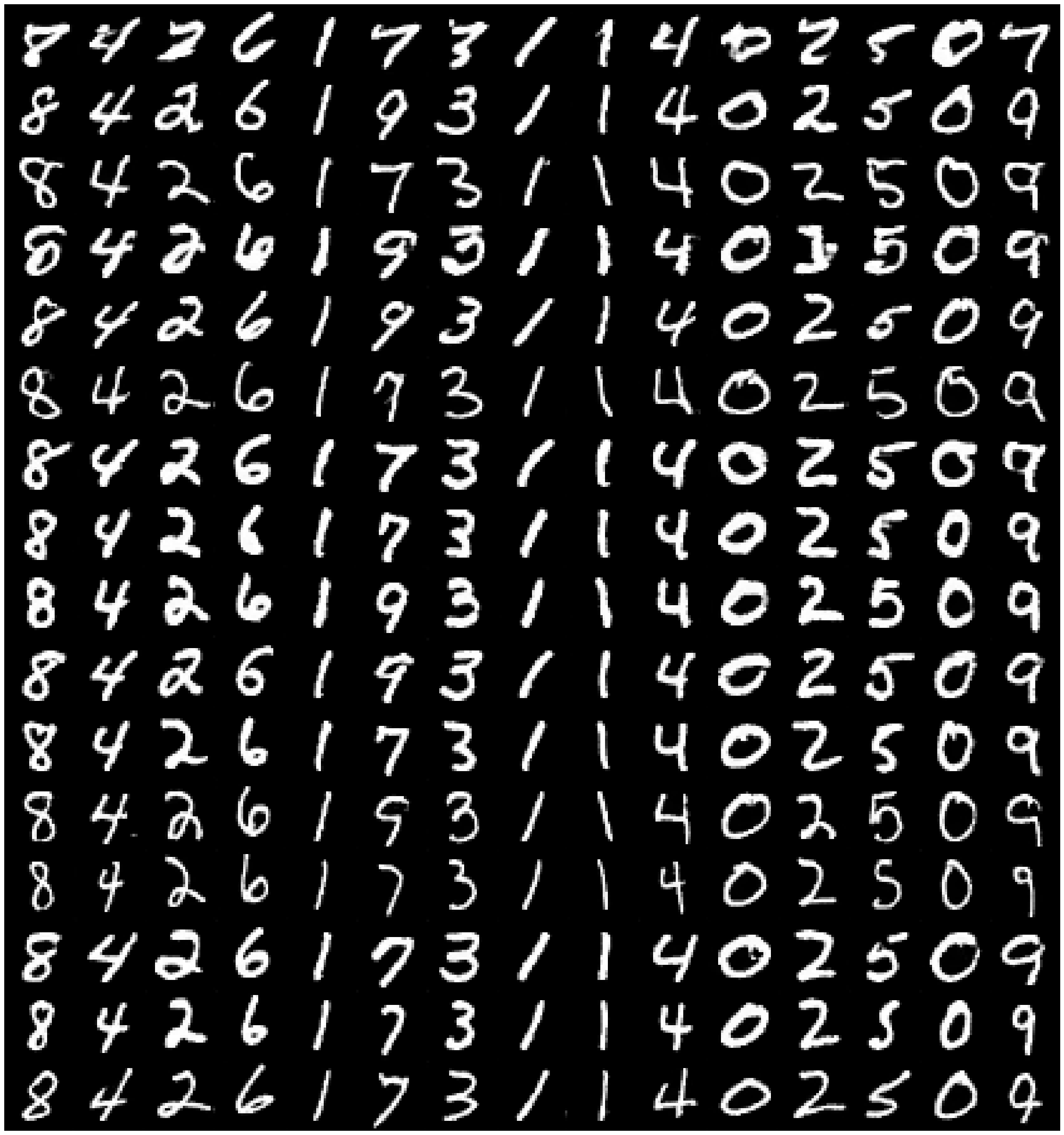

Figure 5: MNIST generated data with $\lambda_{creg} = 1.0$. Each row has the same continuous latent code, and each column has the same categorical latent code. Out of d_c categories, those with a probability less than 1% were excluded. The probabilities for each category are $[0.0968, 0.0382, 0.0660, 0.0983, 0.0170, 0.1378, 0.1015, 0.0492, 0.0441, 0.0583, 0.0178, 0.0357, 0.0898, 0.0806, 0.0670]$. The entropy of categorical latent distribution is 2.5874.

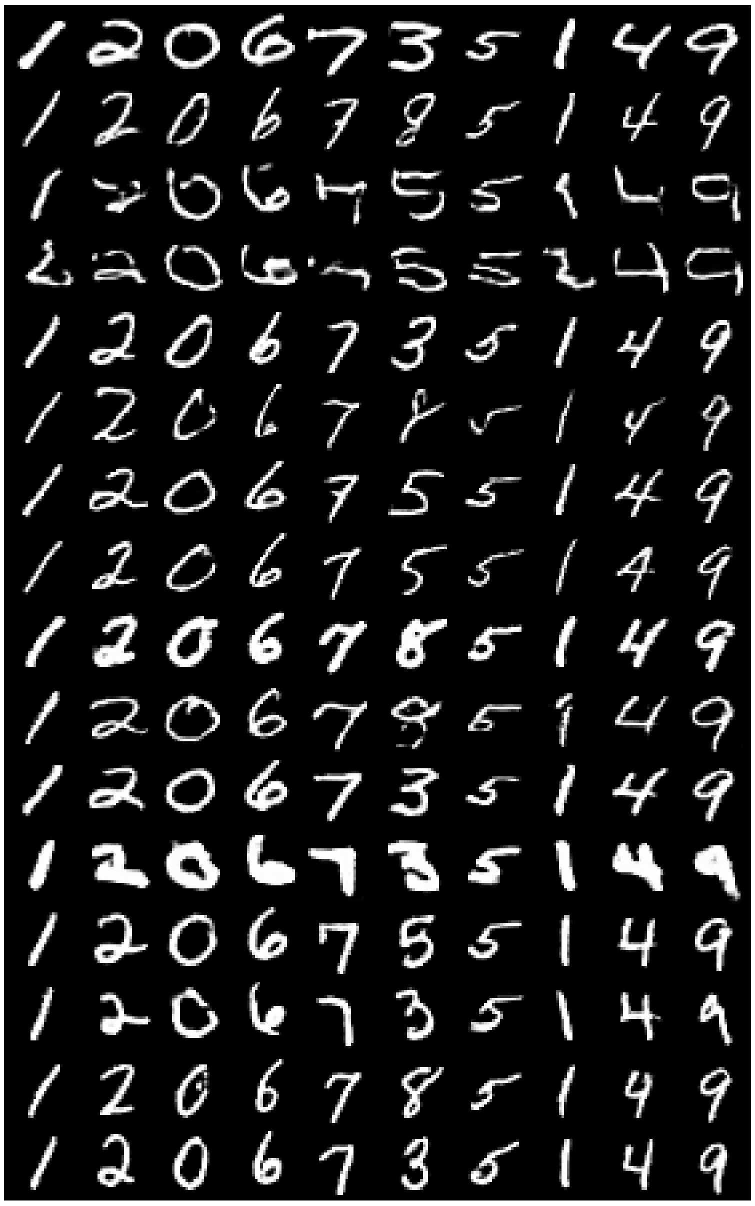

Figure 6: MNIST generated data with $\lambda_{creg} = 5$. Each row has the same continuous latent code, and each column has the same categorical latent code. Out of d_c categories, those with a probability less than 1% were excluded. The probabilities for each category are $[0.0346, 0.1033, 0.0985, 0.0984, 0.1045, 0.2786, 0.0116, 0.0778, 0.0965, 0.0962]$. The entropy of categorical latent distribution is 2.1015.

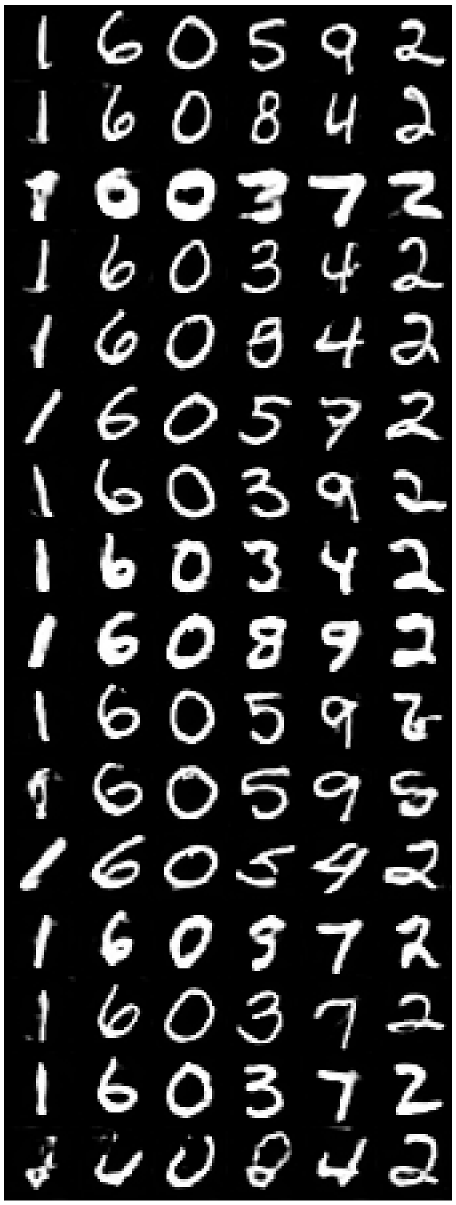

Figure 7: MNIST generated data with $\lambda_{creg} = 10$. Each row has the same continuous latent code, and each column has the same categorical latent code. Out of d_c categories, those with a probability less than 1% were excluded. The probabilities for each category are $[0.1077, 0.0985, 0.0997, 0.2920, 0.3016, 0.1004]$. The entropy of categorical latent distribution is 1.6515.

5 Conclusion

In this paper, we introduced VCAGAN, a self-supervised class-conditional GAN. VCAGAN uses CAGAN loss, classification loss, and classifier gradient penalty loss. Also, the categorical latent distribution is updated to approximate the classifier output distribution of the real data.

The classifier gradient penalty loss weight of VCAGAN controls the sensitivity of each cluster. The entropy of the categorical latent distribution gradually decreases and converges to the appropriate value.

VCAGAN does not require a label of data, optimal categorical latent distribution, and a good metric to calculate the distance between data. This means that VCAGAN can be used in most situations regardless of the data domain.

VCAGAN performed better than Vanilla GAN or InfoGAN with categorical latent distribution in Gaussian clusters generation experiments. VCAGAN could also perform self-supervised class-conditional data generation on the MNIST experiment.

References

[1] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative Adversarial Nets. In Commun. ACM, vol. 63, no. 11, pp. 139-144, Nov. 2020. https://doi.org/10.1145/3422622

[2] Mirza, M., Osindero, S.: Conditional Generative Adversarial Nets. In arXiv preprint, 2014, arXiv:1411.1784. https://arxiv.org/abs/1411.1784

[3] A. Odena, C. Olah, J. Shlens, "Conditional Image Synthesis with Auxiliary Classifier GANs," in proceedings of the 34th International Conference on Machine Learning, PMLR 70:2642-2651, 2017. https://proceedings.mlr.press/v70/odena17a.html

[4] Cho, J., Yoon, K.: Conditional Activation GAN: Improved Auxiliary Classifier GAN. In IEEE Access, vol. 8, pp. 216729-216740, 2020. https://doi.org/10.1109/ACCESS.2020.3041480

[5] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. In NIPS proceedings, 2016. https://papers.nips.cc/paper/2016/hash/7c9d0b1f96aebd7b5eca8c3edaa19ebb-Abstract.html

[6] Lucic, M., Kurach, K., Michalski, M., Gelly, S., Bousquet, O.: Are GANs Created Equal? A Large-Scale Study. In NIPS, 2018. https://papers.nips.cc/paper/2018/hash/e46de7e1bcaaced9a54f1e9d0d2f800d-Abstract.html

[7] Mescheder, L., Geiger, A., Nowozin S.: Which Training Methods for GANs do actually Converge? In PMLR, 2018. https://proceedings.mlr.press/v80/mescheder18a.html

[8] Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive Growing of GANs for Improved Quality, Stability, and Variation. In ICLR conference, Vancouver, Canada, Apr. 30-May 3, 2018. https://openreview.net/forum?id=Hk99zCeAb

[9] Lecun, Y., Cortes, C., and Burges, C.: MNIST handwritten digit database, 2010. In ATT Labs. http://yann.lecun.com/exdb/mnist