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, class-conditional InfoGAN can generate class-conditional data through a self-supervised (unsupervised) method without a labeled dataset. Instead, class-conditional 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 (prior probability). The proposed model consists of a discriminator, a classifier, and a generator, and uses three losses.

The first loss is the cross-entropy classification loss to predict the conditional vector 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 conditional vector 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 local optimum over a wide region.

Additionally, the proposed method updates the categorical latent distribution with a predicted conditional vector of real data. As training progresses, the entropy of the categorical latent distribution gradually decreases 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 measure the distance between data.

1 Introduction

Among recent deep generative models, GAN [1] and diffusion model [13] have shown state-of-the-art performance. In general, it is known that the generative performance of diffusion models is higher [14], but diffusion models are not easy to analyze the latent space due to their high latent dimension and recurrent inference. GANs, on the other hand, have a low latent dimension and require only one inference for sampling, making it easier to analyze the latent space and create applications. For example, InterFaceGAN [16] showed that the label of data is linearly separable in latent space. InterFaceGAN also proposed a method to continuously change the attributes of the input face image using these characteristics.

Among variations of GANs, a conditional GAN [2] is a GAN that generates conditional data. CGAN's generator takes an unconditional latent vector and a conditional vector as input and generates data corresponding to the conditional vector. To generate data corresponding to a conditional vector, the training dataset should consist of real data and the corresponding conditional vector, which can be continuous or discrete. Pix2Pix [17] is an example of a conditional GAN using a continuous conditional vector. Pix2Pix takes an image that is a continuous conditional vector and generates a corresponding conditional image. For example, Pix2Pix can be trained to take a grayscale image as input and output a corresponding color image.

On the other hand, a class-conditional GAN is a conditional GAN where the conditional vector is a discrete categorical vector. ACGAN [3] and CAGAN [4] are examples of class-conditional GANs. ACGAN and CAGAN take one or multiple discrete categorical vectors as input and generate data corresponding to the categorical vectors. In ACGAN, a classifier is trained to predict the label of real data, and a generator is trained so that the fake data generated with the discrete categorical vector is correctly classified by the classifier. CAGAN is a composite of multiple GANs, where each GAN is trained to generate each class. Therefore, CAGAN does not use a classifier, but only multiple adversarial losses to generate class-conditional data.

However, these class-conditional GANs can only be trained given the labels (class-conditional vector) of data. Therefore, these methods cannot be used with unlabeled datasets.

Unlike ACGAN or CAGAN, class-conditional InfoGAN [5] can generate class-conditional data distribution even if the data is not labeled. In class-conditional InfoGAN, the classifier and generator are trained so that the conditional vector of the fake data is correctly classified by the classifier. InfoGAN has shown that if the generator and the classifier are trained with classification loss, it is possible to generate class-conditional data without knowing the conditional vector of the real data. This is because the generator generates class-conditional data that is easy to be classified by the classifier. For example, the MNIST handwritten digits dataset [9] consists of handwritten images of 10 different digits, each with a proportion of 0.1. If one trains class-conditional InfoGAN to generate the MNIST dataset with a 10-dimensional categorical conditional vector with each category having a probability of 0.1, each conditional vector will represent each digit.

Although InfoGAN does not require a conditional vector of the real data, it requires knowing the prior probability or the data distribution. That is, in the previous MNIST example, one should know that a 10-dimensional conditional vector with a probability of 0.1 for each category is a good representation of the MNIST dataset.

Elastic InfoGAN [12] is proposed for class-conditional data generation even when one does not know the optimal categorical latent distribution. Elastic InfoGAN updates categorical latent probability through gradient descent to minimize generator loss. Elastic InfoGAN also restricted each class to have the same identity by using contrastive loss [15] with identity preserving transformations.

In this paper, we analyze the problems of previous works and propose a novel self-supervised (unsupervised) class-conditional GAN, Virtual Conditional Activation GAN (VCAGAN) to address them. VCAGAN can be used under the following conditions:

1. The labels of all data are unknown.
2. Optimal categorical latent distribution is unknown.
3. Metric to measure the distance between the data is unknown.

A VCAGAN consists of a discriminator, a classifier, and a (class-conditional) generator. 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 to minimize 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 allows the categorical latent distribution of the fake data to approximate that of the real data.

InfoGAN cannot be used under condition 2 because it requires an optimal categorical latent distribution. Elastic InfoGAN cannot be used under condition 3 because it requires a metric for identity preserving transformation. Also, in InfoGAN or Elastic-infoGAN, the generator is trained to minimize classification loss, which causes a conflict between classification loss and adversarial loss. On the other hand, VCAGAN does not have this conflict because it uses CAGAN loss [4] as the class-conditional GAN loss. Also, Elastic InfoGAN cannot adjust the sensitivity of each cluster, while VCAGAN can adjust the sensitivity of each cluster through the classifier gradient penalty loss weight.

2 Class-conditional Data Generation

Typically, when training a GAN, everything is assumed to be continuous. It 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 should be differentiable continuous functions to be trained).

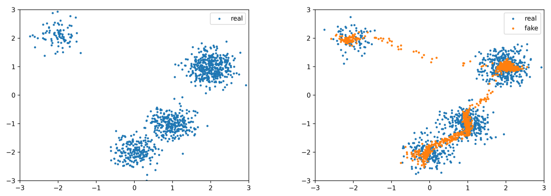

Figure 1: Left part: Two-dimensional dataset consisting of four Gaussian clusters. The centers and probabilities of each cluster are $(-2, 2), (0, -2), (1, -1), (2, 1)$ and $[0.1, 0.2, 0.3, 0.4]$, respectively. The standard deviation for all clusters is 0.3. Right part: Samples generated by GAN trained only with a continuous latent distribution.

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 four Gaussian clusters. There is no perfect discrete part in this data distribution (i.e., the probability density function is still continuous), but one can see that it is easier to represent this data distribution with a 4-dimensional 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 the clusters. 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 data distribution (i.e., entangled data representation).

For datasets with discrete parts, using a discrete latent distribution is more appropriate for model training and disentangled 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. This allows the discrete part of the dataset to be represented appropriately. However, ACGAN or CAGAN cannot be trained if there is no conditional vector (label) for real data.

Class-conditional 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. The following equations show losses for class-conditional InfoGAN.

$$L_q = \lambda_{cls} L_{cls} \tag{1}$$

$$L_d = L^d_{adv} \tag{2}$$

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

$$L_{cls} = \mathbb{E}_{z,c}\left[cross\ entropy(c, Q(G(z,c)))\right] \tag{4}$$

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

$$L^g_{adv} = \mathbb{E}_{z,c}\left[f_g(D(G(z,c)))\right] \tag{6}$$

In Eqs. 1, 2, and 3, L_q, L_d, and L_g represent classifier loss, discriminator loss, and generator loss of InfoGAN, respectively. L_{cls} and λ_{cls} represent classification loss and classification loss weight, respectively. In Eqs. 4, 5 and 6, Q, D, and G represent classifier, discriminator, and generator, respectively. In Eq. 4, L_{cls} is cross entropy between categorical latent code c and predicted label of generated data $Q(G(z,c))$. z represents continuous latent code sampled from the continuous latent distribution Z. In Eqs. 5 and 6, f_d and f_g represent adversarial loss function [6] for GAN training. In InfoGAN, a classifier Q can share hidden layers with a discriminator D for efficiency.

From the above equations, one can see that a classifier Q and a generator G are trained to minimize classification loss L_{cls}. 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 prior information about categorical latent distribution C. Without knowing the appropriate categorical latent distribution C, InfoGAN 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. Specifically, the density of fake data is always lower than the density of real data near the decision boundary of classifier Q in InfoGAN. This is because the generator is trained to move the fake data away from the decision boundary due to classification loss.

Elastic InfoGAN uses Gumbel softmax [18, 19] and contrastive loss in addition to class-conditional InfoGAN loss. Gumbel softmax is used to perform gradient descent on the categorical latent distribution probability. The categorical latent distribution probability is trained to minimize generator loss. Contrastive loss is the loss that ensures that augmented data is classified into the same class through an identity preserving transformation. By training the classifier with contrastive loss, generators are constrained to generate data with the same identity if they are of the same class. For example, Elastic InfoGAN has used rotation, zoom, flip, crop, and gamma change as identity preserving transformations for image data.

However, there are still several problems in Elastic InfoGAN. First, contrastive loss can only be used if a good transformation that preserves the identity of the data is known. Therefore, it cannot be used in data domains where an identity preserving transformation is not known. Second, such transformation can prevent clustering on that transformation. For example, on the MNIST handwritten digits dataset, if 180-degree rotation is used for identity preserving transformation, Elastic InfoGAN will consider the digits 6 and 9 as the same class. Also, like InfoGAN, Elastic InfoGAN uses classification loss in its generators, which causes conflict between adversarial loss and classification loss.

3 Virtual Conditional Activation GAN

In this paper, we introduce VCAGAN which can perform class-conditional data generation and clustering under more general conditions than InfoGAN. VCAGANs can be used under the following very general conditions:

1. The labels of all data are unknown.
2. Optimal categorical latent distribution is unknown.
3. Metric to measure the distance between the data is unknown.

InfoGAN cannot be used under condition 2 because it requires an optimal categorical latent distribution and Elastic InfoGAN cannot be used under condition 3 because it requires a metric for identity preserving transformation.

A VCAGAN consists of a discriminator D, classifier Q, and (class-conditional) generator G. The discriminator has d_c-dimensional output like CAGAN. d_c represents the dimensionality of the categorical latent distribution. The classifier is trained to predict d_c-dimensional conditional vector of fake data. Also, the predicted conditional vector of real data is used for adversarial training of the generator and discriminator. CAGAN loss is used for adversarial loss in VCAGAN to prevent conflict between adversarial loss and classification loss. The generator G takes d_z-dimensional continuous latent distribution and d_c-dimensional categorical latent distribution as inputs to generate class-conditional data.

The following equations show the losses to train VCAGAN.

$$L_q = \lambda_{cls} L_{cls} + \lambda_{creg} L_{creg} \tag{7}$$

$$L_d = L_{adv}^d \tag{8}$$

$$L_g = L_{adv}^g \tag{9}$$

$$L_{creg} = \mathbb{E}_x \left[\|\nabla((1 - Q(x) \cdot argmax\ onehot(Q(x)))^2)\|_2^2 \right] \tag{10}$$

$$L_{adv}^{d} = \mathbb{E}_{x,z,c} \left[f_d(D(x) \cdot argmax\ onehot(Q(x)), D(G(z,c)) \cdot c) \right] \tag{11}$$

$$L_{adv}^{g} = \mathbb{E}_{z,c} \left[f_g(D(G(z,c)) \cdot c) \right] \tag{12}$$

In Eq. 7, λ_{creg} and L_{creg} represent classifier gradient penalty loss weight and classifier gradient penalty loss, respectively. Classification loss L_{cls} is the same as InfoGAN's classification loss (Eq. 4). In Eq. 9, one can see that there is no classification loss L_{cls} in generator loss L_g. Since VCAGAN's generator is trained with adversarial losses only, there is no conflict between L_{cls} and L_{adv}^{g} as in InfoGAN.

Eqs. 11 and 12 show CAGAN adversarial losses for VCAGAN. 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. Instead, 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. 10 shows classifier gradient penalty loss for VCAGAN. As with adversarial loss, $argmax\ onehot(Q(x))$ is used as the label of real data x. Minimizing the classification loss L_{cls} can be done by simply classifying the input data well, but it can also be done by increasing the slope of the decision boundary or decreasing the density of data near the decision boundary. Since the generator is not trained with a classification loss L_{cls} in VCAGAN, the classification loss does not lower the density of fake data near the decision boundary, unlike InfoGAN. Instead, the decision boundary will naturally move to the local optimum which minimizes the density of the fake data, and the slope of the decision boundary will increase. If the slope of the decision boundary is very large and the probability density function of the data is lumpy, the decision boundary will converge to a local optimum in a very narrow region that minimizes the density of the data.

To avoid classifier decision boundary converging local optimum in a narrow region 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}. By relaxing the slope of the classifier's decision boundary, the decision boundary can converge to a local optimum in a wider region. To give a higher gradient penalty for being closer to the decision boundary, VCAGAN uses the gradient over 1 minus probability squared as L_{creg} (Eq. 10). In the classifier, the larger the classifier gradient penalty loss L_{creg}, the more the decision boundary converges to the local optimum in a larger region. Therefore, VCAGAN can adjust the sensitivity of each cluster through the weight of the classifier gradient penalty loss.

Additionally, VCAGAN updates the probability of the categorical latent distribution $P(C)$ during the training with $\mathbb{E}_x[Q(x)]$ (i.e., $P(C) \approx \mathbb{E}_x[Q(x)]$). Through this approximation, VCAGAN can approximate $P(C)$ without knowing it. However, updating $P(C)$ early in the training can make VCAGAN converge to a trivial solution (i.e., one category has a probability of 1 and the other has a probability of 0). To avoid converging a trivial solution and ensure that the ratio of real to fake data in each category is similar, VCAGAN normalizes $Q(x)$ by the batch distribution only at the beginning of training.

Algo. 1 shows the training step of VCAGAN.

The training step of VCAGAN requires X (data random variable), Z (continuous latent random variable), C (categorical latent random variable), D (discriminator), Q (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 lines 5 and 6, D and Q takes a fake data point x' as input and outputs a_f (fake adversarial vector) and c_f' (fake categorical latent code prediction), respectively. Similarly, in lines 7-8, D and Q take a real data point x as input and outputs a_r (real adversarial vector) and c_r' (real categorical latent code prediction).

In lines 9-11, the real categorical latent code prediction c_r' is normalized for stable training only at the beginning of the training. In line 10, the *prob normalize* function forces the real categorical latent code c_r to approach a uniform distribution. This ensures that the ratio of real data to fake data in each category is similar, allowing for stable training in the early stages of VCAGAN training.

Algorithm 1 Training step of VCAGAN

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

 4: $x' \leftarrow G(z, c_f)$
 5: $a_f \leftarrow D(x')$
 6: $c'_f \leftarrow Q(x')$
 7: $a_r \leftarrow D(x)$
 8: $c'_r \leftarrow Q(x)$
 9: if $early\,in\,training$ then
10:     $c'_r = prob\,normalize(c'_r)$
11: end if
12: $c_r \leftarrow argmax\ onehot(c'_r)$

13: $L_{cls} \leftarrow cross\ entropy(c_f, c'_f)$
14: $L_{creg} \leftarrow \|gradient((1 - c_r \cdot c'_r)^2, x)\|_2^2$

15: $L_d \leftarrow f_d(a_r \cdot c_r, a_f \cdot c_f)$
16: $L_q \leftarrow \lambda_{cls} L_{cls} + \lambda_{creg} L_{creg}$
17: $L_g \leftarrow f_g(a_f \cdot c_f)$

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

19: $return\ L_d, L_q, L_g, C$
```

$$prob\,normalize(\mathbf{c}) = \mathbf{c} - batch\ average(\mathbf{c}) + \frac{1}{d_z} \tag{13}$$

Eq. 13 shows the function to normalize the categorical latent codes $\mathbf{c}$, where $\mathbf{c}$ is a $b \times d_z$ matrix, where b represents the batch size. *batch average* is a function that computes the element-wise average vector of $\mathbf{c}$. Therefore, $batch\ average(\mathbf{c})$ is d_z-dimensional vector. One can see that $batch\ average(prob\,normalize(\mathbf{c}))$ is always $[\frac{1}{d_z}, \frac{1}{d_z}, ..., \frac{1}{d_z}]$. However, the *prob normalize* function restricts the representation of the real categorical latent code c_r, so it is not used after some training.

In line 12, real categorical latent code c_r is calculated from c'_r.

In line 13, L_{cls} represents classification loss. *crossentropy* is a function that calculates cross-entropy loss. In line 14, $gradient(y, x)$ function calculates slope dy/dx.

In lines 15-17, L_d, L_q, and L_g represent discriminator loss, classifier loss, and generator loss, respectively. f_d and f_g represent adversarial functions for GAN. If one uses a gradient penalty for adversarial loss, we recommend using R2 regularization for adversarial loss, where the true label is known, instead of R1 regularization [7].

In line 18, $P(C)$ 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. In VCAGAN, $P(C)$ is initialized with $[\frac{1}{d_z}, \frac{1}{d_z}, ..., \frac{1}{d_z}]$, and c'_r is normalized at the beginning of the training (line 10). Thus, at the beginning of training, $P(C)$ will always be $[\frac{1}{d_z}, \frac{1}{d_z}, ..., \frac{1}{d_z}]$. This makes C to not converge to a trivial solution.

After line 19, D, Q and G are updated with losses L_d, L_q, and L_g, respectively.

4 Experiments

We trained the models to generate two-dimensional Gaussian clusters and MNIST dataset [9]. In Gaussian clusters experiments, we compare the performance of Vanilla GAN [1], InfoGAN [5], Elastic InfoGAN [12], and our proposed VCAGAN. In MNIST, we compared the clustering of VCAGANs according to classifier gradient penalty loss weight λ_{creg}.

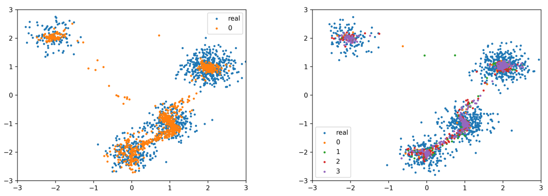

Figure 2: Vanilla GAN (trained only with adversarial loss) is trained with both continuous latent distribution and categorical latent distribution. Left: 1-dimensional categorical latent distribution ($P(C) = [1.0]$). Right: 4-dimensional optimal categorical latent distribution ($P(C) = [0.1, 0.2, 0.3, 0.4]$).

4.1 Gaussian clusters experiments

In Gaussian clusters experiments, we used the dataset consisting of four 2-dimensioanl Gaussian clusters as a training dataset. The 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. The generator, discriminator, and classifier consisting of four fully connected hidden layers with 512 units were used for training. The following hyperparameters were used for experiments.

$$Z \sim N(0, I_{64})$$
$$optimizer = Adam \begin{pmatrix} learning\,rate = 0.001 \\ \beta_1 = 0.0 \\ \beta_2 = 0.99 \end{pmatrix}$$
$$batch\ size = 32$$
$$\lambda_{r2} = 1$$
$$train\ step\ per\ epoch = 1000$$
$$epoch = 100$$

λ_{r2} represents R2 regularization [7] loss weight ($\lambda_{r2} = \gamma/2$). Classification loss weight $\lambda_{cls} = 1.0$ was used for InfoGAN, elastic InfoGAN, and VCAGAN. We used exponential moving average with $decay\,rate = 0.999$ as $update$ function for VCAGAN. In InfoGAN, Elastic InfoGAN, and VCAGAN, classifier Q and discriminator D do not share hidden layers. Equalized learning rate [8] was used for all weights. Also, exponential moving average [8] with $decay\,rate = 0.999$ was used for generator weights. In VCAGAN and Elastic InfoGAN, $d_c = 8$ was used, and $P(C)$ was updated after epoch 25 (i.e., in VCAGAN, $early\,in\,training$ in line 9 of Algo. 1 was $True$ until epoch 25). Since we assumed that there is no good metric to measure the distance between data, we did not use identity preserving transformations in Elastic InfoGAN. Only gradient descent on a categorical latent distribution with Gumbel softmax was used for Elastic InfoGAN.

Fig. 2 shows samples generated with vanilla GAN (trained only with adversarial loss). The left part of Fig. 2 shows data generated by a vanilla GAN trained with a one-dimensional categorical latent distribution. Since there is no discrete part in the latent distribution, one can see that the vanilla GAN generates lines between clusters. This shows that when training a GAN with only continuous latent vectors, the latent space is entangled and not suitable for representing data with discrete parts.

The right part of Fig. 2 shows data generated by a vanilla GAN trained with optimal categorical latent distribution ($P(C) = [0.1, 0.2, 0.3, 0.4]$). 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. This means that even if the generator takes a discrete categorical latent distribution as input, additional loss is required to make the generator use it meaningful.

Fig. 3 shows data generated by three InfoGAN trained with the optimal categorical latent distribution. Unlike the Vanilla GAN, one can see that the model generates class-conditional distribution with the categorical latent distribution. However, one can still see the problems of InfoGAN in this figure.

The first problem was that even though the categorical latent distribution was optimal, sometimes each conditional vector was not mapped to the correct cluster. One can see that in all iterations,

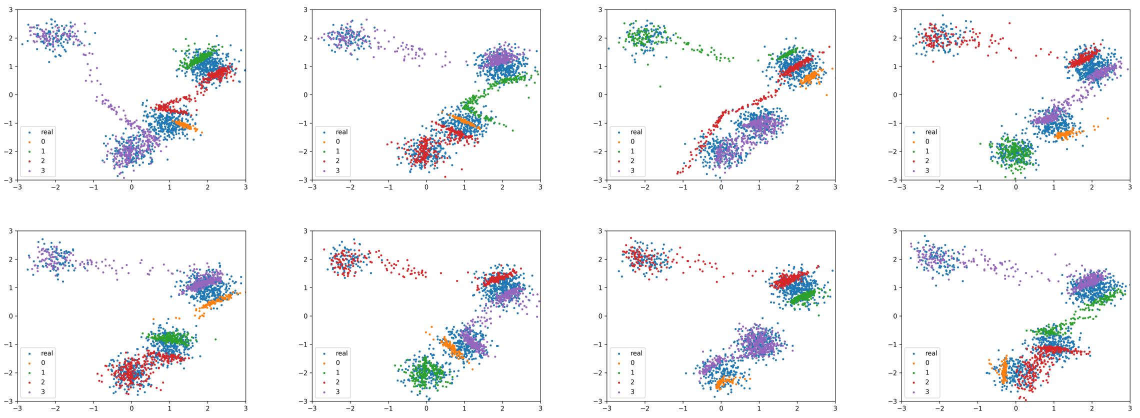

Figure 3: InfoGAN trained with 4-dimensional optimal categorical latent distribution ($P(C) = [0.1, 0.2, 0.3, 0.4]$). Eight times repeated.

InfoGAN did not map all categories to the correct cluster. This shows that even when InfoGAN is trained with the optimal categorical latent distribution, it may not be able to assign each cluster to the correct category.

The second problem is that the generator does not generate data near the decision boundary of the classifier. In Fig. 3, one can see that the generator of InfoGAN does not generate data near the decision boundary of the classifier. This is because classification loss and adversarial loss are in conflict in the generator of InfoGAN.

Fig. 4 shows data generated by elastic InfoGAN. We tested several combinations of hyperparameters, but Elastic InfoGAN could not generate class-conditional data correctly.

Fig. 5 shows samples generated by VCAGAN trained with different λ_{creg}. In column 1 of Fig. 5, one can see that some clusters were divided into more than one category. This is because there is no L_{creg}, and the estimated fake data distribution was not perfectly smooth, so the decision boundary of the classifier converged to a local optimum in a very narrow region.

However, in column 2, one can see that each category is assigned to each cluster correctly. The probability of each category is also very accurate. Also, unlike InfoGAN, one can see that there is a natural division between clusters in the middle. This is because VCAGAN's generator is only trained with adversarial loss, not classification loss, so there is no conflict between those losses.

In columns 3 and 4, multiple clusters were assigned to the same category. This is because λ_{creg} was too high, causing the classifier to converge to a local optimum in a wide region.

From this, one can see that VCAGAN can adjust the sensitivity of each category via λ_{creg}.

Fig. 6 shows the results of eight iterations of VCAGAN training with $\lambda_{creg} = 0.1$. One can see that VCAGAN is generating class-conditional data correctly over multiple iterations.

4.2 MNIST experiments

In this experiment, we trained VCAGAN to generate the MNIST handwritten digits dataset [9]. The generator, discriminator, and classifier simply consist of CNNs. *learning rate* $= 0.003$, $d_z = 256$, $d_c = 64$, *epoch* $= 300$ were used for the experiments. $P(C)$ was updated after epoch 50. Other hyperparameters are the same as in section 4.1. We used FID [10], precision & recall [11] for generative performance evaluation. All evaluation methods used the Inception model. 32k training samples were used for evaluation.

Figs. 7, 8, 9, and 10 show the difference in class-conditional data generation of VCAGAN according to λ_{creg}.

First, in Figs. 7 and 8, since λ_{creg} was too low, the classifier decision boundary converged on a local optimum in a narrow region. Thus, some clusters were split into two or more categories. For example, in Fig. 7, digit 2 was divided into categories 10 (column 10) and 21 (column 21). Notice the difference in the bottom left with and without the loop. One can see that the probability of category 10 is 3.52%, and category 21 is 6.76%, which together adds up to about 10%. Also, digit 7 was divided

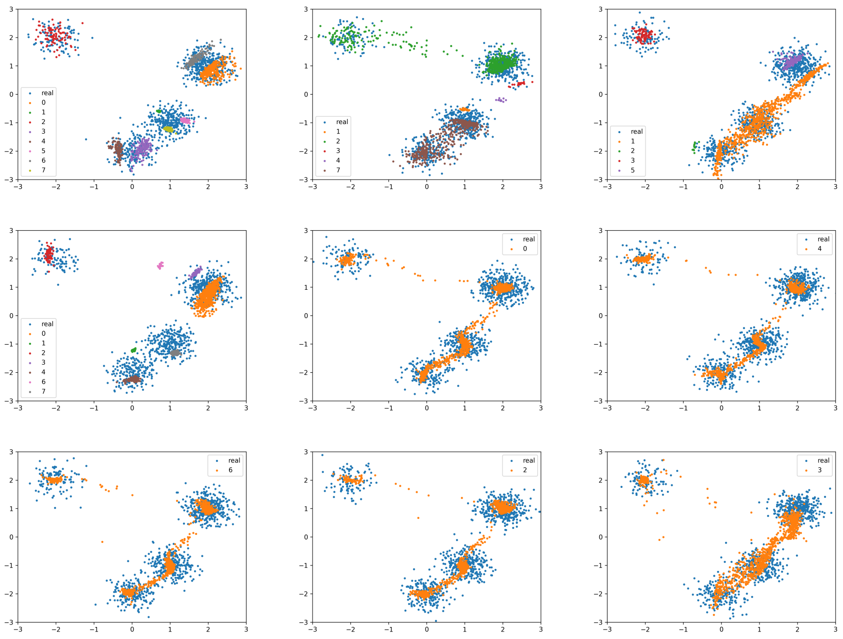
Figure 4: Elastic InfoGAN trained with different temperature and learning rate. Contrastive loss was not used. Row 1: $t = 0.01$, Row 2: $t = 0.1$, Row 3: $t = 1$, Column 1: $lr = 0.001$, Column 2: $lr = 0.003$, Column 3: $lr = 0.01$.

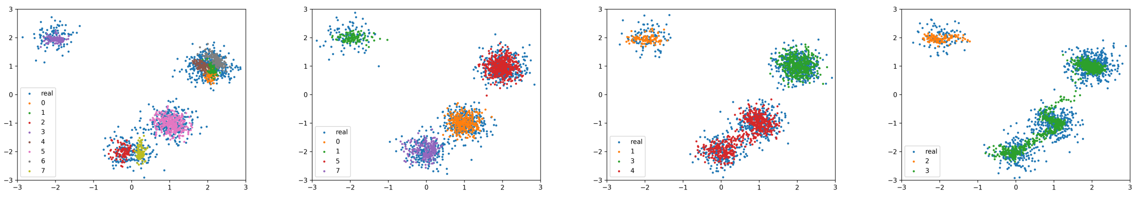
Figure 5: VCAGAN trained with different λ_{creg}. Categories with a probability of less than 1% were ignored. Column 1: $\lambda_{creg} = 0.0$, $P(C) = [0.0938, 0.0795, 0.0959, 0.0985, 0.0744, 0.3012, 0.1514, 0.1053]$. Column 2: $\lambda_{creg} = 0.1$, $P(C) = [0.2983, 0.0998, 0.3989, 0.2031]$. Column 3: $\lambda_{creg} = 1.0$, $P(C) = [0.0995, 0.3967, 0.5037]$. Column 4: $\lambda_{creg} = 10.0$, $P(C) = [0.1004, 0.8997]$.

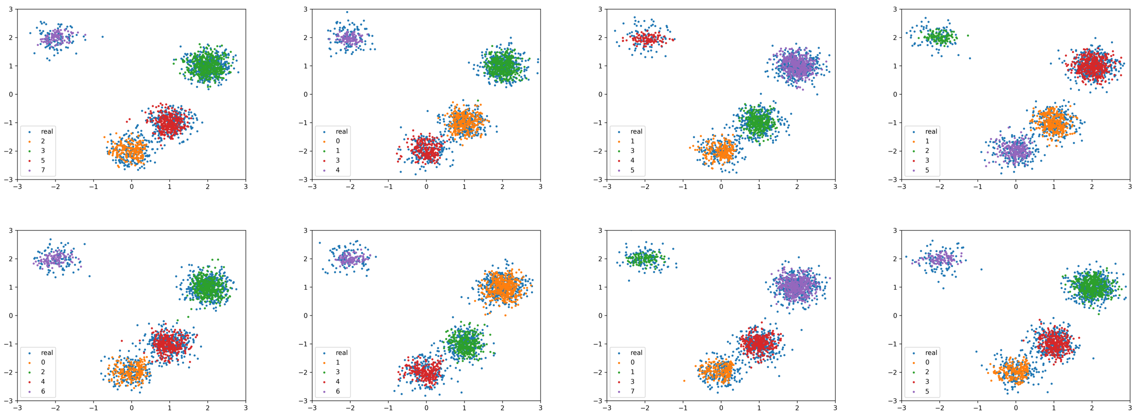
Figure 6: VCAGAN trained with $\lambda_{creg} = 0.1$. Categories with a probability of less than 1% were ignored. Eight times repeated.

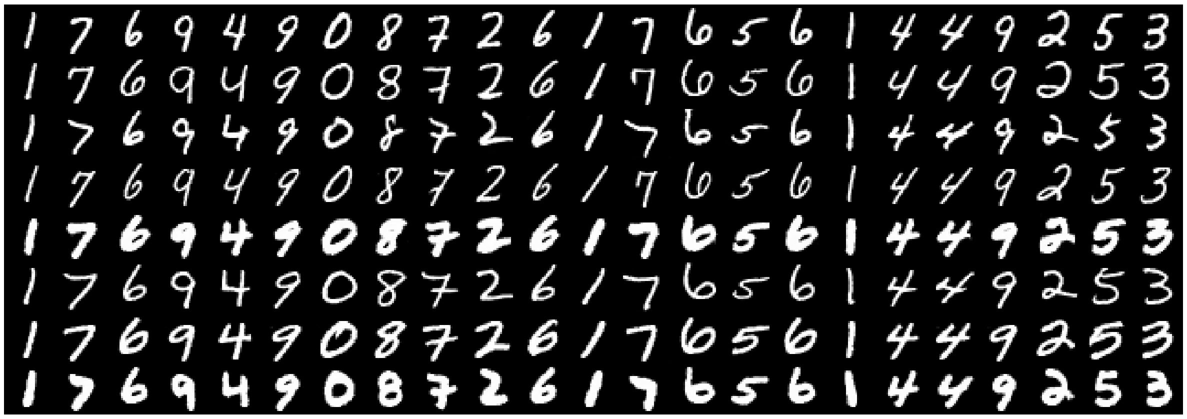

Figure 7: MNIST generated data with $\lambda_{creg} = 10$. Each row has the same continuous latent code, and each column has the same categorical latent code. Out of d_c categories, those with a probability less than 1% were excluded. The probabilities for each category are $[0.0166, 0.0567, 0.0386, 0.0493, 0.0584, 0.0191, 0.0970, 0.0961, 0.0127, 0.0352,$ $0.0146, 0.0474, 0.0296, 0.0136, 0.0170, 0.0292, 0.0455, 0.0241, 0.0105, 0.0244, 0.0676, 0.0743, 0.1020]$. FID: 3.1934, precision: 0.8224, recall: 0.6484.

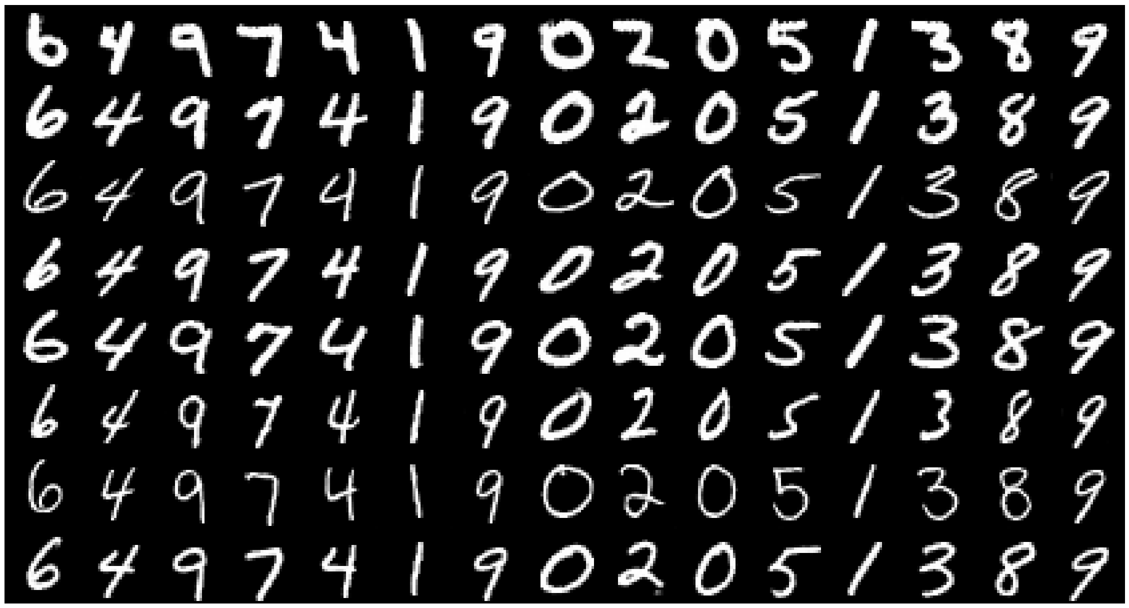

Figure 8: MNIST generated data with $\lambda_{creg} = 20$. Each row has the same continuous latent code, and each column has the same categorical latent code. Out of d_c categories, those with a probability less than 1% were excluded. The probabilities for each category are $[0.0985, 0.0357, 0.0421, 0.1017, 0.0565, 0.0628, 0.0425, 0.0439, 0.1014, 0.0541,$ $0.0899, 0.0469, 0.1020, 0.0970, 0.0127]$. FID: 3.0919, precision: 0.8195, recall: 0.6535.

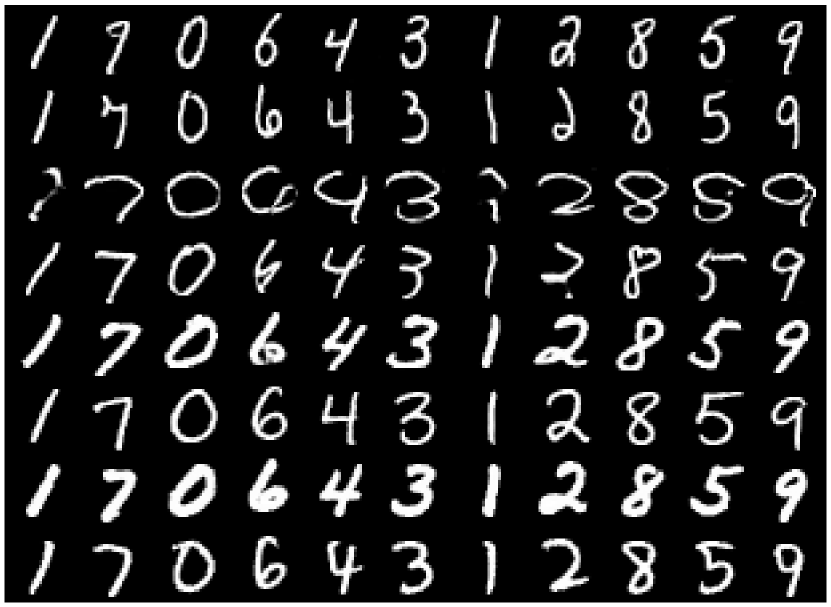

Figure 9: MNIST generated data with $\lambda_{creg} = 50$. Each row has the same continuous latent code, and each column has the same categorical latent code. Out of d_c categories, those with a probability less than 1% were excluded. The probabilities for each category are $[0.0434, 0.1054, 0.1017, 0.1003, 0.0985, 0.1046, 0.0656, 0.0998, 0.0978, 0.0879, 0.0951]$. FID: 2.7041, precision: 0.8206, recall: 0.6687.

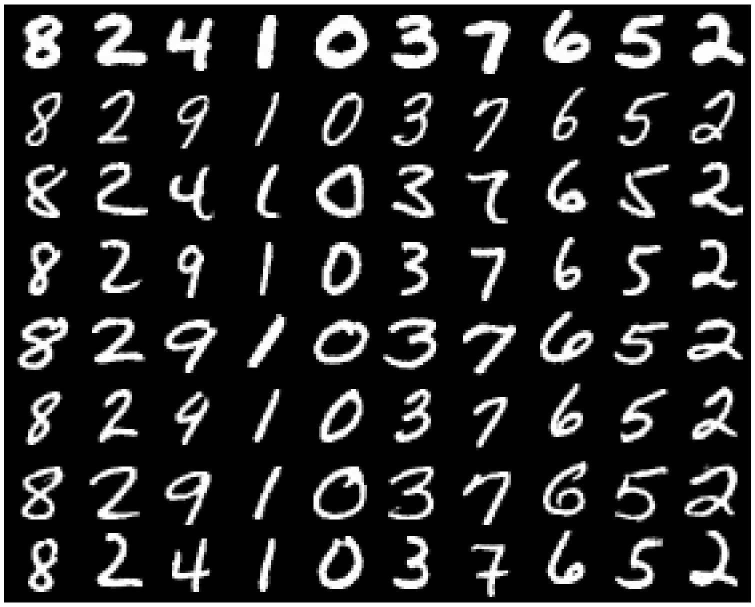

Figure 10: MNIST generated data with $\lambda_{creg} = 100$. Each row has the same continuous latent code, and each column has the same categorical latent code. Out of d_c categories, those with a probability less than 1% were excluded. The probabilities for each category are $[0.0969, 0.0267, 0.1931, 0.1091, 0.0987, 0.1057, 0.1040, 0.1016, 0.0908, 0.0735]$. FID: 2.9231, precision: 0.8223, recall: 0.6600.

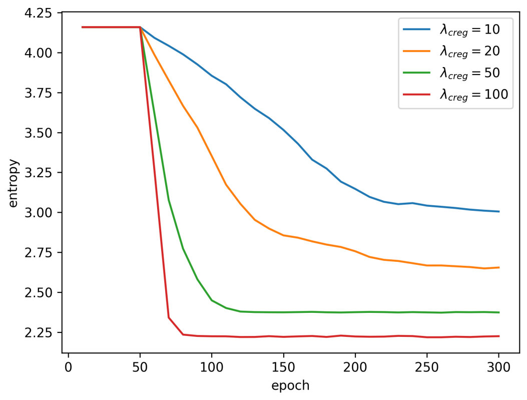

Figure 11: Entropy of a categorical latent distribution over epochs.

into categories 2, 9, and 13. Notice that digits 7 in category 9 have the center horizontal line. One can see that the probability of category 2 is 5.67%, category 9 is 1.27%, and category 13 is 2.96%, which together adds up to about 10%. Also, several other digits were divided into different categories. On the other hand, the digits 0 (category 7), 8 (category 8), and 3 (category 23) each correspond to only one category, and the probability of each category was about 10%.

Some digits were split into multiple categories, but this does not mean that VCAGAN performed an incorrect class-conditional data generation. For example, without any prior information, digit 2 with a loop and without a loop, and digit 7 with a horizontal line in the center and without a horizontal line can be considered different clusters. This means that the optimal clustering (and class-conditional data generation) may depend on the sensitivity of each cluster.

As λ_{creg} increases, the classifier decision boundary converges to the local optimum over a wider region. In Fig. 9, $\lambda_{creg} = 50$ was used for training. One can see that all digits are clustered into one category, except for digit 1. Digit 1 was split into two categories based on angle.

In Fig. 10, $\lambda_{creg} = 100$ was used for training. One can see that digit 2 has been divided into categories 2 and 10 based on loop, and digits 4 and 9 have been clustered into the same category. One can notice that digits 4 and 9 are relatively close to each other compared to the other digits. In other words, digits 4 and 9 can be considered the same cluster if the sensitivity of the cluster is low.

Fig. 11 shows the entropy of a categorical latent distribution over epochs. Entropy was calculated every 10 epochs. One can see that the larger λ_{creg} is, the faster the entropy decreases.

In this experiment, one can see that VCAGAN properly performed the class-conditional data generation of the MNIST handwritten digit by adjusting λ_{creg}. In particular, when λ_{creg} is low so the sensitivity of each cluster is high, VCAGAN found that there are different patterns within the same digit (e.g., digits 2 and 7). When λ_{creg} is high so the sensitivity of each cluster is low, VCAGAN found that digits 4 and 9 have a similar pattern.

Separately, one can see that all three VCAGANs have good unconditional generative performance from FID and precision & recall.

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.

Unlike previous works, 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's generator is trained with adversarial loss only, so there is no conflict between classification loss and adversarial loss. Therefore, VCAGAN can correctly generate data near the decision boundary of a classifier. Also, VCAGAN can VCAGAN can also adjust the sensitivity of each cluster with the classifier gradient penalty loss.

VCAGAN performed better than Vanilla GAN, InfoGAN, and Elastic InfoGAN in Gaussian cluster generation experiments. We also showed that VCAGAN could perform self-supervised class-conditional data generation on the MNIST experiments.

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

[10] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In NIPS, 2017. https://papers.nips.cc/paper/2017/hash/8a1d694707eb0fefe65871369074926d-Abstract.html

[11] Kynkäänniemi, T., Karras, T., Laine, S., Lehtinen, J., Aila, T.: Improved precision and recall metric for assessing generative models. In NIPS proceedings, 2019. https://proceedings.neurips.cc/paper/2019/hash/0234c510bc6d908b28c70ff313743079-Abstract.html

[12] Utkarsh Ojha, Krishna Kumar Singh, Cho-Jui Hsieh, Yong Jae Lee, "Elastic-InfoGAN: Unsupervised Disentangled Representation Learning in Class-Imbalanced Data," in NIPS, 2020. https://proceedings.neurips.cc/paper/2020/hash/d1e39c9bda5c80ac3d8ea9d658163967-Abstract.html

[13] Jonathan Ho, Ajay Jain, Pieter Abbeel, "Denoising Diffusion Probabilistic Models," in NIPS, 2020.https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html

[14] Zhisheng Xiao, Karsten Kreis, Arash Vahdat, ”Tackling the Generative Learning Trilemma with Denoising Diffusion GANs”, in ICLR 2022. https://openreview.net/pdf?id=JprM0p-q0Co

[15] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, ICML 2020. https://dl.acm.org/doi/abs/10.5555/3524938.3525087

[16] Yujun Shen, Jinjin Gu, Xiaoou Tang, Bolei Zhou, ”Interpreting the Latent Space of GANs for Semantic Face Editing,” in CVPR 2020.https://openaccess.thecvf.com/content_CVPR_2020/papers/Shen_Interpreting_the_Latent_Space_of_GANs_for_Semantic_Face_Editing_CVPR_2020_paper.pdf

[17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros, in CVPR 2017. https://openaccess.thecvf.com/content_cvpr_2017/papers/Isola_Image-To-Image_Translation_With_CVPR_2017_paper.pdf

[18] Eric Jang, Shixiang Gu, and Ben Poole, ”Categorical reparameterization with gumbel-softmax,” in ICLR, 2017. https://openreview.net/pdf?id=rkE3y85ee

[19] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh, ”The concrete distribution: A continuous relaxation of discrete random variables,” in ICLR, 2017. https://openreview.net/forum?id=S1jE5L5gl