Unsupervised Image Generation with Infinite Generative Adversarial Networks Supplementary Material

Hui Ying[1], He Wang[2], Tianjia Shao[1]*, Yin Yang[3], Kun Zhou[1]
[1]Zhejiang University [2]University of Leeds [3]Clemson University
huiying@zju.edu.cn, H.E.Wang@leeds.ac.uk, tjshao@zju.edu.cn, yin5@clemson.edu, kunzhou@acm.org

1. Algorithm Details

The training of MIC-GANs is split into two stages, the initialization stage and the Adversarial Chinese Restaurant Process (ACRP) Stage.

1.1. Initialization Stage

The initialization stage is to initialize the generator, enabling it to produce images of good quality. At the same time, we require the generator to produce conditioned output without supervised class labels.

The detailed algorithm for initialization is shown in Algorithm 1. The training procedure is the same as the ordinary GANs, except that the generator is given a conditioned input. Note that the discriminator is not conditional, so it is not a conditional GAN. $Categ(\alpha_1, ..., \alpha_K)$ is the category distribution where index number k is sampled according to the probability proportional to α_k.The conditional input for generator is uniformly sampled, i.e. $\alpha_1, ..., \alpha_K = \frac{1}{K}$.

After initialization, the generator can produce conditioned outputs. Generally, the outputs from one condition are more likely to be close to one class of images. However, because K is different from the ground-truth class number, the outputs from one condition often either contain only a part of one class or multiple classes. Taking MNIST for example, when using $K = 20$ which is larger than the ground-truth class number, after initialization there may be two modes generating '6' and one mode generating both '7' and '9'. However, it will be resolved in the later ACRP stage.

1.2. ACRP Stage

The main algorithm of ACRP is shown in Algorithm 2, and the Chinese Restaurant Process Sampling algorithm is shown in Algorithm 3. In practice, the encoder network in Q is initialized before training in every epoch to prevent overfitting. The parameters of the GMMs need to be initialized before training. We let the covariance matrices Σ_ks be the identity matrix, and initialize the mean μ_ks in the way that they become the vertices of a high-dimensional simplex and are equidistant to each other. This is to ensure that the Gaussians are distinctive. Next, the dimensionality of the latent space of the encoder in Q needs to be decided. Theoretically, it can be any dimension that is smaller than that of the data space. In practice, we set the dimension of both the image embedding and GMM to K which is the number of modes, so that conveniently the μ_ks are the basis vectors of such a K-dimensional space. One straightforward solution is to use the one-hot K-vectors as the μ_ks' initialization. As for the encoder loss $\mathcal{L}_Q(e, \mu_c)$, we maximize the log likeli-

Variable	Meaning
$\mathcal{X}$	the whole real images
K	the number of modes
N	the total number of real images
G_{ϕ^G}	generator parameterized by ϕ^G
D_{ϕ^D}	discriminator parameterized by ϕ^D
Q_{ϕ^Q}	classifier parameterized by ϕ^Q
z	the input noise for generator
α_k	the sampling probability of mode k
c_i	the picked mode index for each real image i
μ_k, Σ_k	the parameters of the kth Gaussian
N_k	the number of real images associated with mode k
$p_{i,k}$	the likelihood of real image i on mode k
e	the embedding of an image, from encoder Q

Table 1. Symbols of MIC-GANs Training

Algorithm 1 Initialization

Require:
$epochs$ - the number of total training epochs;
N_{init} - the number of images for initialization training;

for $n = 1$ to N_{init} **do**
  Sample $x \sim \mathcal{X}$
  Sample $z \sim \mathcal{N}(0, 1)$, $c \sim Categ(\alpha_1, ..., \alpha_K)$
  Generate fake image $\hat{x} = G(z, c)$
  Optimize ϕ^g and ϕ^d via GAN loss
end for

*Corresponding author. The authors from Zhejiang University are affiliated with the State Key Lab of CAD&CG.

Algorithm 2 Adversarial Chinese Restaurant Process

Require:

$epochs$ - the number of total training epochs;
N_Q - the number of images for training encoder in each epoch;
N_{GD} - the number of images for training generator and discriminator in each epoch;
$iters_1$ - the number of iterations for CRP sampling and GMM updating;
Initialize() ;

1: **for** $epoch = 1$ to $epochs$ **do**
2: **for** $n = 1$ to N_Q **do** ▷ Train Q
3: Sample $z \sim \mathcal{N}(0, 1)$, $c \sim Categ(\alpha_1, ..., \alpha_K)$
4: Get embedding $e = Q(G(z, c))$
5: Optimize ϕ_Q via encoder loss $L_Q = \mathcal{L}_Q(e, \mu_c)$
6: **end for**
7: **for** $iter = 1$ to $iters_1$ **do** ▷ Classify x
8: **for** x_i in $\mathcal{X}$ **do** ▷ Computing likelihood
9: $e_i = Q(x_i)$
10: $p_{i,k} = Gauss(e_i|\mu_k, \Sigma_k)$ for $k = 1$ to K
11: **end for**
12: Sample $\{c_i\}_{i=1}^N$, $\{N_k\}_{k=1}^K$ via CRP (Alg. 3)
13: $\mathbb{E}_k \leftarrow \emptyset$ for $k = 1$ to K
14: $\mathbb{E}_{c_i} \leftarrow \mathbb{E}_{c_i} \bigcup \{e_i\}$ for each e_i
15: **for** $k = 1$ to K **do** ▷ Update GMMs
16: Update μ_k and Σ_k with $\mathbb{E}_k$
17: **end for**
18: **end for**
19: $\alpha_k \leftarrow \frac{N_k}{N}$
20: **for** $n = 1$ to N_{GD} **do** ▷ Train GAN
21: Sample $x_i \sim \mathcal{X}$, and fetch corresponding c_i
22: Sample $z \sim \mathcal{N}(0, 1)$, $c \sim Categ(\alpha_1, ..., \alpha_K)$
23: Generate fake image $\hat{x} = G(z, c)$
24: Optimize ϕ^g and ϕ^d via conditional GAN loss
25: **end for**
26: **end for**

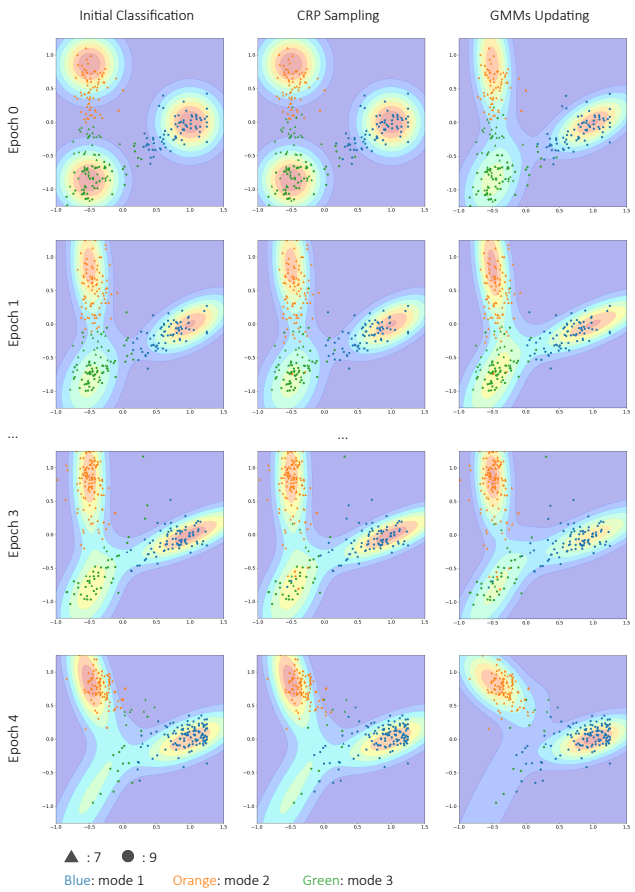


Figure 1. The transformation of classification results and GMMs in the ACRP stage. In each small figure, the small dots represent embeddings of training images from encoder Q, with the triangle dots representing number '7' and the circle dots representing number '9'. The colors of the dots represent the classifications to three modes. The background color visualizes the shape of GMMs. In each epoch, the left two figures show the classification results conducted directly from the gaussian probability and after CRP sampling and the right figure shows the updated GMMs based on the classification results after CRP sampling.

hood of e with respect to the Gaussian with μ_c as its mean.

Likelihoods in GANs. We do not solve the likelihood problem of GANs directly. MIC-GANs employ a surrogate density estimator. The design is due to the following reasons. First, MIC-GANs are designed to work essentially with any GANs. So the density estimation needs to be independent of the specific GAN architecture. Also, different GANs are designed for different tasks, e.g. StyleGAN, BigGAN, etc. They all contain specific architectures optimized for their aimed tasks. Therefore, we choose to keep any chosen GAN intact under MIC-GANs. Employing a surrogate density estimator is our current solution, and we are actively looking for a 'true' solution.

To help understand ACRP, we present a visualization (Figure 1) of the procedure of our algorithm with a simple dataset which only consists of number '7' and number '9' from MNIST. The figure visualizes line 7-18 of Algorithm 2. In the algorithm, we set the embedding dimension to be 2 with $K = 3, iters_1 = 1, iters_2 = 1$, and we fix the means of GMMs to be three vertices of an equilateral triangle for better visualization quality. As shown in Figure 1, at first, 'mode 1' contains most '9's and 'mode 2' contains most '7's while 'mode 3' contains both '9's and '7's. With the algorithm progressing, the number of images classified to 'mode 3' is gradually reduced, because more '9's that were originally classified to 'mode 3' are now classified to 'mode 1', and similarly '7's that are originally under 'mode 3' are now classified to 'mode 2'. In Epoch 4, most of the images are divided into two modes and the classification is almost correct. Meanwhile, the original 'mode 3' basi-

Algorithm 3 Chinese Restaurant Process Sampling

Require:

$iters_2$ - the number of iterations for CRP sampling;

1: $N_k \leftarrow 0$ for $k = 1$ to K
2: **for** x_i in $\mathcal{X}$ **do**
3: $c_i \leftarrow argmax(\{p_{i,k}\}_{k=1}^{K})$
4: $N_{c_i} = N_{c_i} + 1$
5: **end for**
6: **for** $iter = 1$ to $iters_2$ **do**
7: **for** x_i in $\mathcal{X}$ **do**
8: $N_{c_i} = N_{c_i} - 1$
9: $\beta_k \leftarrow N_k \cdot p_{i,k}$ for $k = 1$ to K
10: $\beta_k \leftarrow \frac{\beta_k}{\sum \beta_k}$ for $k = 1$ to K
11: Sample $c_i \sim Categ(\beta_1, ..., \beta_K)$
12: $N_{c_i} = N_{c_i} + 1$
13: **end for**
14: **end for**

cally disappears as the probability of it being sampled again becomes nearly zero. To see this, N_3 in line 12 in Algorithm 2 after CRP becomes very small and the probability of 'mode 3' being sampled again is proportional to N_3. Figure 1 shows the 'richer gets richer' property of Chinese Restaurant Process.

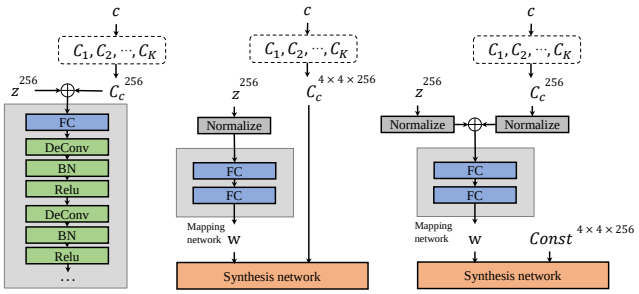
2. Implementation Details

2.1. Network Architecture

We adopt different GAN models including DCGAN [11], StyleGAN2 [4] and StyleGAN2-Ada [3] to validate our algorithm. Specifically, in order to achieve conditioned generation, we modify the input of the generators to take conditions. The detailed implementation of the conditional inputs is shown in Figure 2. For DCGAN and StyleGAN2-Ada, the condition of the generator is specified by adding the conditioned latent code C_c to the noise z. In StyleGAN2, we tried to control the condition by picking one of K constant inputs for the synthesis network.

The network architecture of discriminator needs to be handled differently in different stages because the discriminator needs to take conditions in ACRP stage but the conditions are not reliable in the initialization stage. So we keep the discriminator as the original one in DCGAN or StyleGAN2 during initialization, and in ACRP stage, modify it to take conditions. For the discriminator of DCGAN and StyleGAN2, we follow the approach of traditional cGAN [7], and for the discriminator of StyleGAN2-Ada, we follow the approach in [8]. After initialization and at the beginning of the ACRP stage, a condition input is added to the discriminator.

For the network architecture of encoder in Q, we simply



(a) DCGAN (b) StyleGAN2 (c) StyleGAN2-Ada

Figure 2. The conditional input heads of the generators for DCGAN, StyleGAN2 and StyleGAN2-Ada.

	training Q	sampling	training GAN
DCGAN	0.5mins	2.6mins	4.5mins
StyleGAN2	0.5mins	2.6mins	15mins

Table 2. Training time distribution for one epoch on MNIST.

epochs	1	5	9	13	19
purity	0.839	0.908	0.911	0.927	0.929

Table 3. Purity vs sampling epochs on MNIST with $K = 15$.

adopt a multi-layer convolutional network. For the dataset of MNIST, FashionMNIST, and Hybrid, we use a 4-layer CNN with a fully connected output layer, and for the dataset of CatDog, CIFAR and Tiny Imagenet, we use a 7-layer CNN with a fully connected output layer. Both of them do not use BatchNorm Layer.

Besides the DCGAN and StyleGAN, there are other GANs that are also suitable for mode separation, e.g., FiLM [10].In fact, our algorithm can be applied to any conditional GANs theoretically.

2.2. Training Details

Images in MNIST, FashionMNIST, Hybrid, CIFAR and Tiny Imagenet are resized to 32, and images in CatDog are resized to 64. When training, the batch size is set to 64 for CatDog and CIFAR, and 256 for the other datasets. During initialization, N_{init} is set to $2400k$ for the MNIST FashionMNIST, Hybrid dataset, $1000k$ for the CatDog dataset, $2000k$ for the CIFAR and Tiny Imagenet dataset. In ACRP stage, N_Q is set to $64k$, and N_{GD} is set to $300k$ for the MNIST FashionMNIST, Hybrid dataset, $100k$ for the CatDog dataset, $200k$ for the CIFAR and Tiny Imagenet dataset. We trained the MIC-GANs for totally 40 epochs in ACRP stage, as the classification results of the real images converge quickly, we stop CRP Sampling (re-classification) after 10 epochs, so the GAN can focus on improving the quality of image generation.

For the dataset of Tiny ImageNet, we picked 10 classes for the MIC-GANs training, which are 'goldfish', 'black widow', 'brain coral', 'golden retriever', 'monarch', 'beach wagon', 'beacon', 'bullet train', 'triumphal arch', 'lemon'.

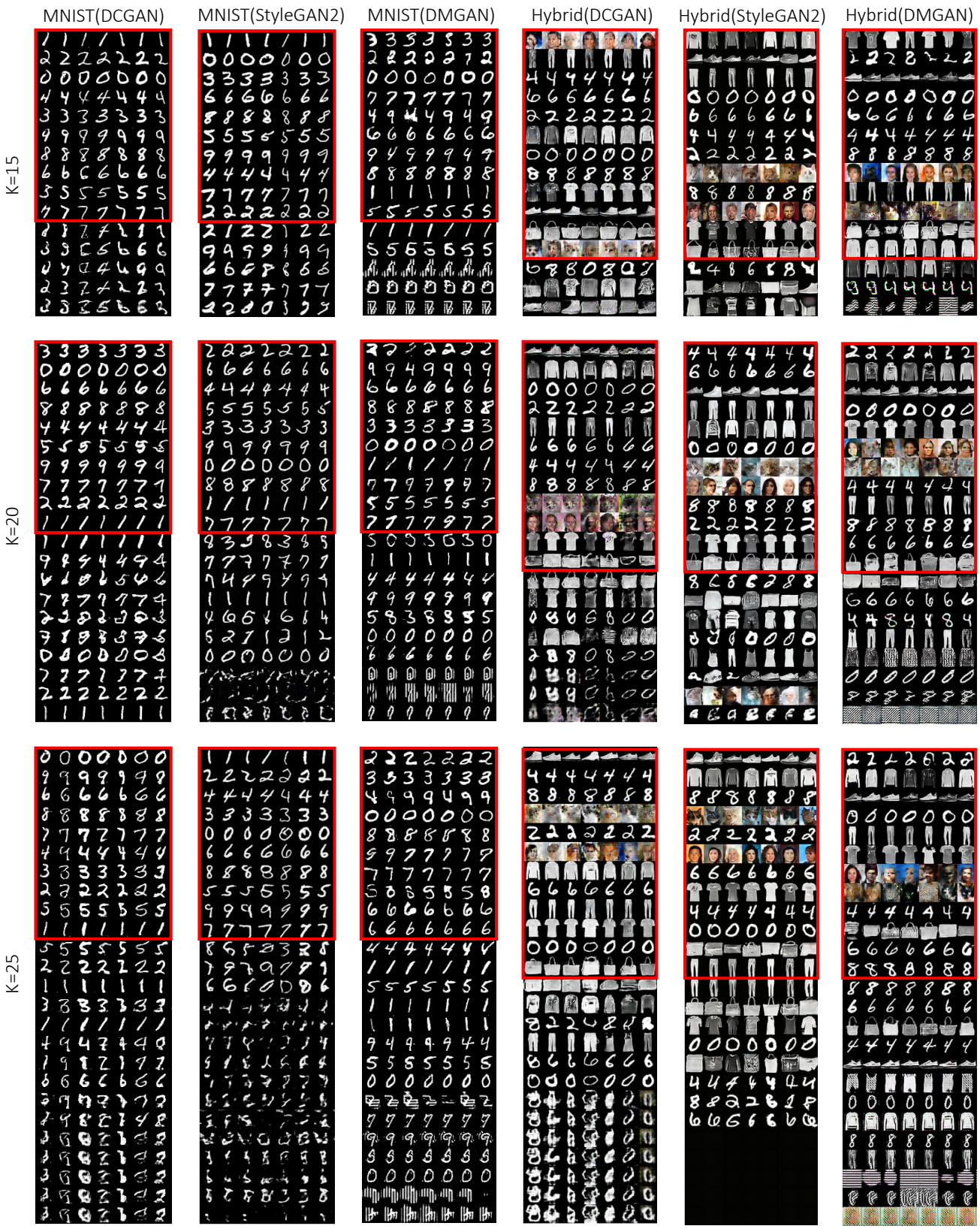


Figure 3. Our results on the MNIST and Hybrid dataset using DCGAN and StyleGAN2 with different Ks, compared to DMGAN. Each row is generated from a mode, and the rows are sorted by αs. The red boxes mark the top n modes in the results, where $n = 10$ for MNIST and $n = 12$ for Hybrid.

3. Quantitative Results

Table. 2 shows the training time distribution for one epoch on MNIST dataset. We find that the sampling in learning the prior is not the most time-consuming component, while the training of the GANs itself dominates the training time. And the situation is similar on all datasets.

In Table. 3, we show the relationship between the purity and sampling epochs on the MNIST dataset. We find that the purity converges quickly in the first few epochs (similar on other datasets). So we stop the CRP sampling after 10 epochs and use the stable classification results for GAN training.

4. Generation Results

Figure 3 visualizes the MNIST and Hybrid results of our method and DMGAN [5]. We can find that even using different Ks, our method can provide stable results, which demonstrates the ability of unsupervised clustering of our method. DMGAN achieves a similar effect as ours, but it often learns mixed modes, e.g., the confusion between '4' and '9' on MNIST. Furthermore, our method is flexible with the architecture of GAN without compromising the training speed much, which means that we can employ complex GANs such as StyleGAN2 on complex datasets. However, it will be prohibitively expensive for DMGAN to achieve the same because DMGAN requires K generators for K modes, while MIC-GANs only require K latent codes.

Figure 4 shows the results of InfoGAN [1], ClusterGAN [9] and DeliGAN [2] on Hybrid with different Ks. Obviously, these methods fail to perform correct clustering when the ground-truth $K = 12$ is unknown and the best way is to make multiple guesses. However, when $K < 12$, there will be mixed modes; when $K > 12$, there will be repetitive modes, as well as mixed modes. This shows that these methods either cannot produce good results or require a large number of guesses in the absence of the ground-truth, while MIC-GANs can generate satisfying results in one run.

Figure 5 shows the generation results of our method, Self-Conditioned GAN [6] and StyleGAN2-Ada [3] on CIFAR with different Cs and Ks. CIFAR is a difficult dataset for generation, and it is an even more challenging dataset for conditioned generation based on unsupervised clustering. In our method, some modes can generate images that are from clear-cutting single classes, e.g., 'automobile', 'airplane', 'horse'. In other cases, images generated from one mode consist of images from two or more classes. This reflects the fact that images can be clustered based on different criteria. This sometimes leads to different classification results between MIC-GANs and human labels. For example, images can be classified according to the colors or shapes or semantics. While human labels in CIFAR are primarily based on semantics (object identities), it is normal that MIC-GANs at times generate images from one mode that match several ground-truth classes. Nevertheless, we can still find some interesting similarities among the images generated from one mode. In addition, MIC-GANs improve the generation quality in general with lower FID scores shown in the paper. We also find that Self-Conditioned GAN suffers from mode collapse in several modes and the problem gets worse when K is small. StyleGAN2-Ada is able to generate images with diversities but ours still achieve better FID scores.

Figure 6 shows the generation results of our method on the Tiny Imagenet dataset. Without any class supervision, our algorithm still generates several reasonable conditional results. For example, line 1, 2, 3, 5, 11, 12 correctly produce the images of lemon, triumphal arch, beacon, brain coral, monarch and beach wagon, while several modes generate the mixtures of classes, like line 4 and 7. Another interesting observation is that line 9 mostly generates bullet trains facing the right while line 10 genearates buleet trains facing the left. Tiny Imagenet is a difficult dataset, so the generation is less ideal on some modes, but still covers most of them.

References

[1] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS 2016*, pages 2172–2180, 2016. 5

[2] Swaminathan Gurumurthy, Ravi Kiran Sarvadevabhatla, and R. Venkatesh Babu. Deligan: Generative adversarial networks for diverse and limited data. In *CVPR 2017*, pages 4941–4949, 2017. 5

[3] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In *NIPS 2020*, 2020. 3, 5

[4] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *CVPR 2020*, pages 8107–8116, 2020. 3

[5] Mahyar Khayatkhoei, Maneesh K Singh, and Ahmed Elgammal. Disconnected manifold learning for generative adversarial networks. In *NIPS 2018*, pages 7343–7353, 2018. 5

[6] Steven Liu, Tongzhou Wang, David Bau, Jun-Yan Zhu, and Antonio Torralba. Diverse image generation via self-conditioned gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14286–14295, 2020. 5

[7] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014. 3

[8] Takeru Miyato and Masanori Koyama. cgans with projection discriminator. In *ICLR 2018*, 2018. 3

[9] Sudipto Mukherjee, Himanshu Asnani, Eugene Lin, and Sreeram Kannan. Clustergan: Latent space clustering in

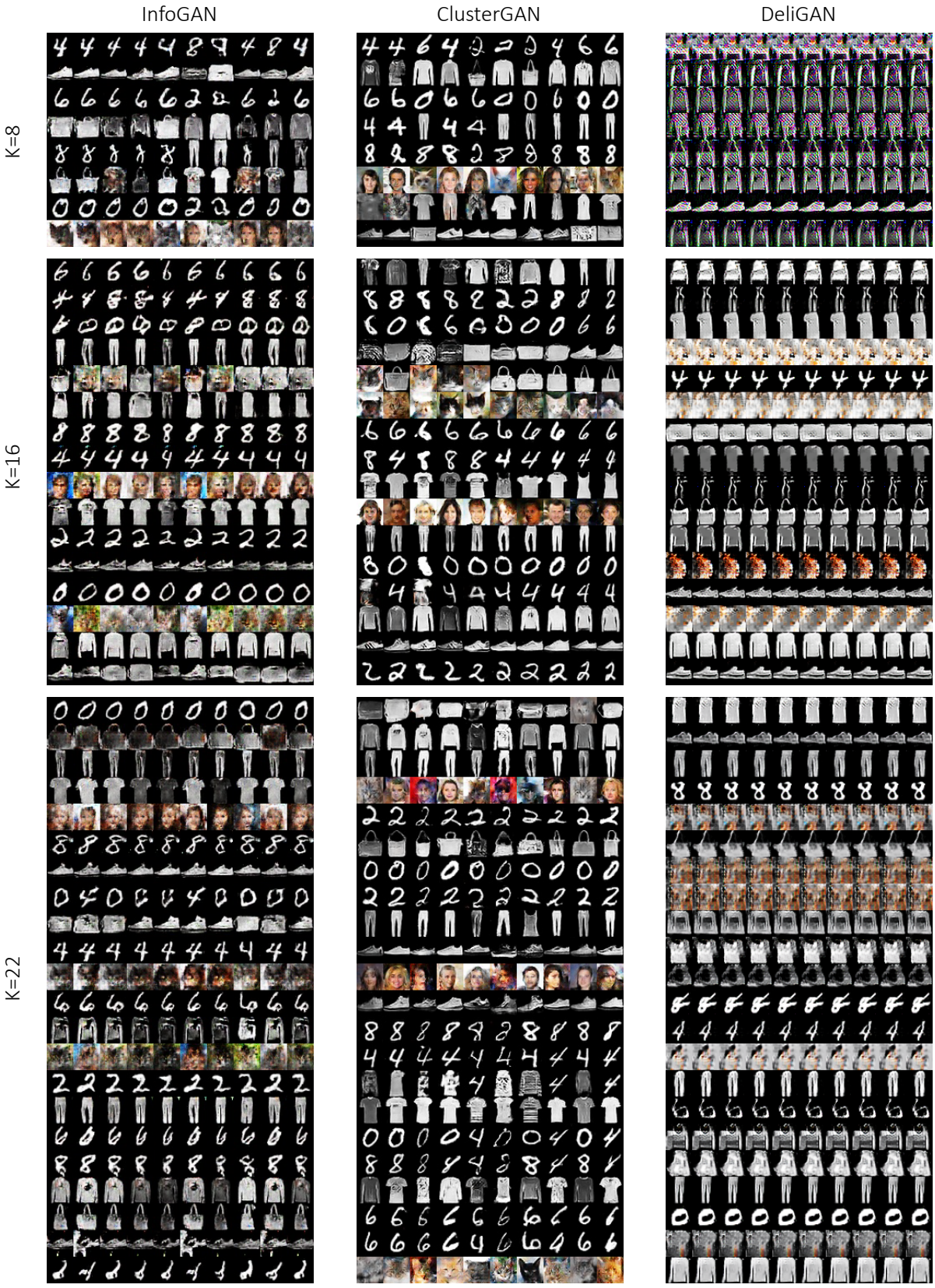


Figure 4. Results of InfoGAN, ClusterGAN and DeliGAN on the Hybrid dataset with different Ks. Each row is generated from a mode.

Figure 5. Results of Our method, Self-Conditioned GAN and StyleGAN2-Ada on the CIFAR with different Cs and Ks. 'C' means the ground-truth class number in the dataset. Note that StyleGAN2-Ada is trained without conditions. For our method and Self-Conditioned GAN, each row is generated from a mode, and the number on the right of each row indicates the distribution of the mode.

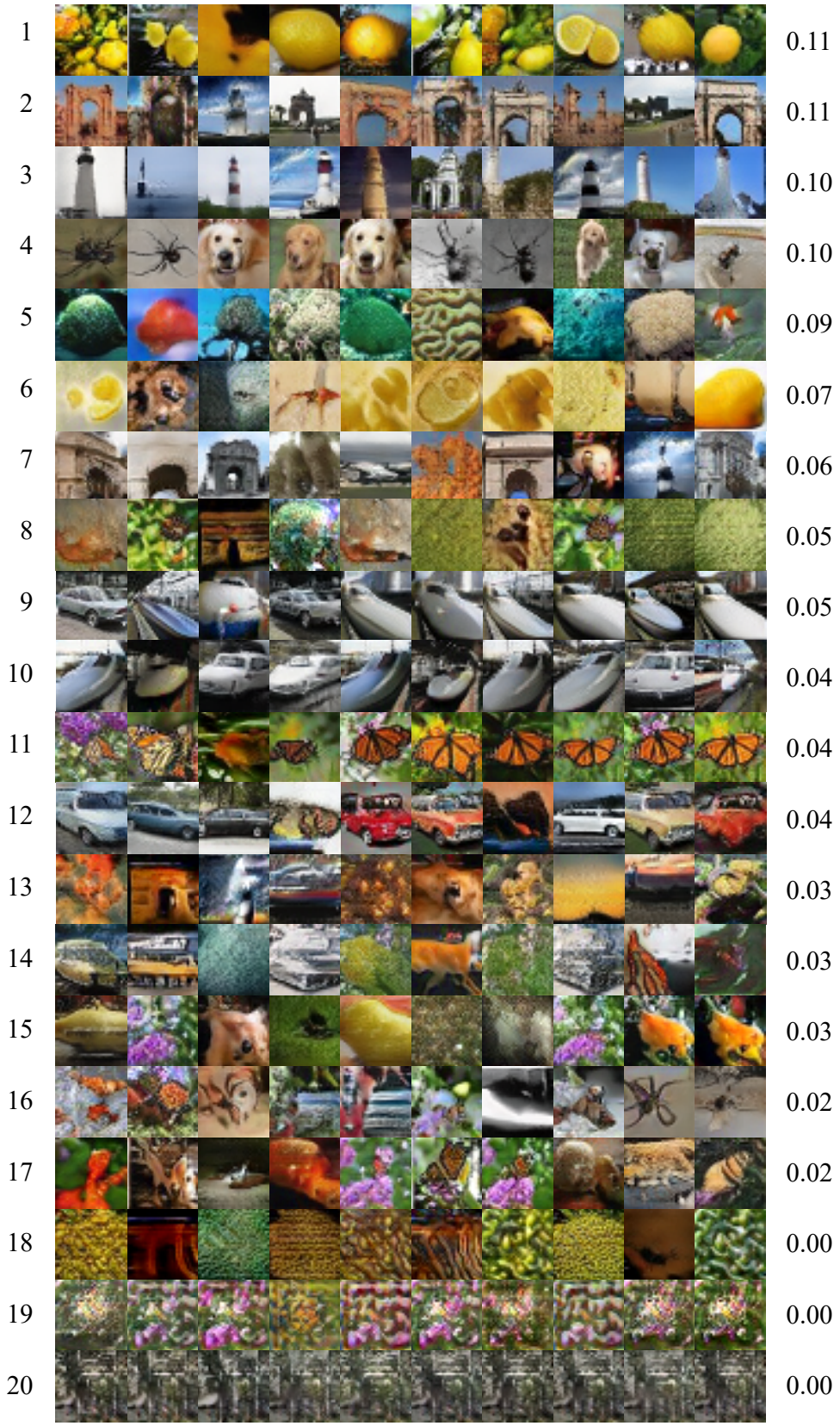


Figure 6. Results of Our method on the Tiny Imagenet with different $C = 10$s and $K = 20$s. Each row is generated from a mode,. The number on the left of each row indicates the line number and on the right indicates the weight of the mode.

generative adversarial networks. In *AAAI 2019*, volume 33, pages 4610–4617, 2019. 5

[10] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI 2018*, 2018. 3

[11] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR 2016*, 2016. 3