A Unified View of cGANs with and without Classifiers

Anonymous Author(s)
Affiliation
Address
email

Abstract

Conditional Generative Adversarial Networks (cGANs) are implicit generative models which allow us to sample from class-conditional distributions. Existing cGAN works are based on a wide range of different architectures and objectives. One popular architecture in earlier works is to include a classifier during training with the assumption that good classifiers can help eliminate samples generated with wrong classes. Nevertheless, including classifiers in cGANs often comes with a side effect of only generating easy-to-classify samples. Recently, some representative cGANs avoid the side effect and reach state-of-the-art performance without having classifiers. Somehow it is not clear whether the classifiers can be resurrected to design better cGANs. In this work, we demonstrate that classifiers can be properly leveraged to improve cGANs. We start by using the decomposition of the joint probability distribution to connect the goals of the cGAN and classifier as a unified framework. The framework, along with a classic energy model to parameterize the distribution, justifies the use of classifiers for cGANs in a principled manner. In addition, it explains several popular cGAN variants, such as ACGAN, ProjGAN, and ContraGAN, as special cases with different levels of approximations. Experimental results demonstrate that the framework outperforms state-of-the-art cGANs on benchmark datasets, especially on the harder Tiny ImageNet.

1 Introduction

Generative Adversarial Networks [GANs; 10] is a family of generative models that are trained from the duel of a generator and a discriminator. The generator aims to generate data from a target distribution, where the fidelity of the generated data is "screened" by the discriminator. Recent studies on the objectives [2, 29, 25, 26], architectures [38, 46], and regularization techniques [13, 35, 47] for GANs have achieved impressive progress on image generation, making GANs the state-of-the-art approach to generate high fidelity and diverse images [3]. Conditional GANs (cGANs) extend GANs to generate data from class-conditional distributions [33, 36, 34, 16]. The capability of conditional generation extends the application horizon of GANs to conditional image generation based on labels [36] or texts [40], speech enhancement [32], and image style transformation [18, 49].

One representative cGAN is Auxiliary Classifier GAN [ACGAN; 36], which decomposes the conditional discriminator to a classifier and an unconditional discriminator. The generator of ACGAN is expected to generate images that convince the unconditional discriminator while being classified to the right class. The classifier plays a pivotal role in laying down the law of conditional generation for ACGAN, making it the very first cGAN that can learn from ImageNet to generate 1000 classes of images [6]. That is, ACGAN used to be a leading cGAN architecture. While the classifier in ACGAN indeed improves the accuracy of conditional generation, deeper studies revealed that the classifier biases the generator to generate easier-to-classify images [42], which in term decreases the capability to match the target distribution.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Unlike ACGAN, most state-of-the-art cGANs are designed without a classifier. One representative cGAN without a classifier is Projection GAN [ProjGAN; 34], which learns an embedding for each class to form a projection-based conditional discriminator. ProjGAN not only generates higher-quality images than ACGAN, but also achieves superior accuracy for conditional generation without relying on an explicit classifier. In fact, it was found that ProjGAN usually cannot be further improved by adding a classification loss [34]. The finding, along with the success of ProjGAN and other cGANs without classifiers [15, 4], seem to suggest that including a classifier is not helpful for improving the generation quality or accuracy of cGANs.

In this work, we challenge the belief that classifiers are not helpful for cGANs, with the conjecture that leveraging the classifiers appropriately can benefit conditional generation. We propose a framework that pins down the roles of the classifier and the conditional discriminator by first decomposing the joint target distribution with Bayes rule. We then model the conditional discriminator as an energy function, which is a unnormalized log probability. Under the energy function, we derive the corresponding optimization term for the classifier and the conditional discriminator with the help of Fenchel duality to form the unified framework. The framework reveals that a jointly generative model can be trained via two routes, from the aspect of the classifier and the conditional discriminator, respectively. We name our framework Energy-based Conditional Generative Adversarial Networks (ECGAN), which not only justifies the use of classifiers for cGANs in a principled manner, but also explains several popular cGAN variants, such as ACGAN [36], ProjGAN [34], and ContraGAN [16] as special cases with different approximations. After properly combining the objectives from the two routes of the framework, we empirically find that ECGAN outperforms other cGAN variants across different network architectures on benchmark datasets. Most importantly, ECGAN achieves superior quality and accuracy on Tiny ImageNet, which is notably difficult because of its scarcity of data in each class.

We summarize the contributions of this paper as:

- We justify the principled use of classifiers for cGANs by decomposing the joint distribution.
- We propose a cGAN framework, Energy-based Conditional Generative Adversarial Networks (ECGAN), which explains several popular cGAN variants in a unified view.
- We experimentally demonstrate that ECGAN consistently outperforms other state-of-the-art cGANs across different network architectures on benchmark datasets.

The paper is organized as follows. Section 2 derives the unified framework that establishes the role of the classifiers for cGANs. The framework is used to explain ACGAN [36], ProjGAN [34], and ContraGAN [16] in Section 3. Then, we demonstrate the effectiveness of our framework by experiments in Section 4. We discuss related work in Section 5 before concluding in Section 6.

2 Method

Given a K-class dataset $(x, y) \sim p_d$, where $y \in \{1 \dots K\}$ is the class of x and p_d is the underlying data distribution. Our goal is to train a generator G to generate a sample $G(z, y)$ following $p_d(x|y)$, where z is sampled from a known distribution such as $\mathcal{N}(0, 1)$. To solved the problem, a typical cGAN framework can be formulated by extending an unconditional GAN as:

$$\max_D \min_G \sum_y \mathop{\mathbb{E}}_{p_d(x|y)} D(x, y) - \mathop{\mathbb{E}}_{p(z)} D(G(z, y), y) \tag{1}$$

where G is the generator and D is a discriminator that output higher values for real data. The choice of D leads to different type of GANs [10, 2, 29, 8].

At first glance, there is no classifier in Eq. (1). However, because of the success of leveraging label information via classification, it is hypothesized that a better classifier can improve conditional generation [36]. Motivated by this, in this section, we show how we bridge classifiers to cGANs by Bayes rule and Fenchel duality.

2.1 Bridge Classifiers to Discriminators with Joint Distribution

A classifier, when viewed from a probabilistic perspective, is a function that approximates $p_d(y|x)$, the probability that $\mathbf{x}$ belongs to class y. On the other hand, a conditional discriminator, telling

whether x is real data in class y, can be viewed as a function approximate $p_d(x|y)$. To connect $p_d(y|x)$ and $p_d(x|y)$, an important observation is through the joint probability:

$$\log p(x,y) = \log p(x|y) + \log p(y) \tag{2}$$
$$= \log p(y|x) + \log p(x). \tag{3}$$

The observation illustrates that we can approximate $\log p(x,y)$ in two directions: one containing $p(x|y)$ for conditional discriminators and one containing $p(y|x)$ for classifiers. The finding reveals that by sharing the parameterization, updating the parameters in one direction may optimize the other implicitly. Therefore, we link the classifier to the conditional discriminator by training both objectives jointly.

2.2 Learning Joint Distribution via Optimization of Conditional Discriminator

Since $p(y)$ is usually known a priori (e.g., uniform) or able to easily estimated (e.g., empirical counting), we focus on learning $p(x|y)$ in Eq.(2). Specifically, since $\log p(x,y) \in \mathbb{R}$, we parameterize it via $f_\theta(x)$, such as a neural network with K real value outputs, where $\exp(f_\theta(x)[y]) \propto p(x,y)$. Similar parameterization is also used in exponential family [44] and energy based model [23]. Therefore, the log-likelihood $\log p(x|y)$ can be modeled as:

$$\log p_\theta(x|y) = \log\left(\frac{\exp\left(f_\theta(x)[y]\right)}{Z_y(\theta)}\right) = f_\theta(x)[y] - \log Z_y(\theta), \tag{4}$$

where $Z_y(\theta) = \int_{x'} \exp\left(f_\theta(x')[y]\right)\, dx'$.

Optimizing Eq. (4) is challenging because of the intractable partition function $Z_y(\theta)$. Here we introduce the Fenchel duality [44] of the partition function $Z_y(\theta)$:

$$\log Z_y(\theta) = \max_{q_y}\left[\mathop{\mathbb{E}}_{q_y(x)}\left[f_\theta(x)[y]\right] + H(q_y)\right]$$

where q_y is a distribution of x conditioned on y and $H(q_y) = -\mathbb{E}_{x\sim q_y(x)}\left[\log q_y(x)\right]$ is the entropy of q_y. By the Fenchel duality, we can obtain our maximum likelihood estimation in Eq. (4) as:

$$\max_\theta\left[\mathop{\mathbb{E}}_{p_d(x,y)}\left[f_\theta(x)[y]\right] - \max_{q_y}\left[\mathop{\mathbb{E}}_{q_y(x)}\left[f_\theta(x)[y]\right] + H(q_y)\right]\right]. \tag{5}$$

To approximate the solution of q_y, in additional to density models, we can train an auxiliary generator q_ϕ as in cGANs to estimate $\mathbb{E}_{q_y(x)}$ via sampling. That is, we can sample x from q_ϕ by $x = q_\phi(z,y)$, where $z \sim \mathcal{N}(0,1)$. The objective (5) then becomes:

$$\max_\theta \min_\phi \sum_y \mathop{\mathbb{E}}_{p_d(x|y)}\left[f_\theta(x)[y]\right] - \mathop{\mathbb{E}}_{p(z)}\left[f_\theta(q_\phi(z,y))[y]\right] - H(q_\phi(\cdot,y)), \tag{6}$$

which is almost in the form of Eq (1) except the entropy $H(q_\phi(\cdot,y))$. We leave the discussion about the entropy estimation in Section 2.4. Currently, the loss function to optimize the objective without the entropy can be formulated as:

$$\mathcal{L}_{d_1}(x,z,y;\theta) = f_\theta(x)[y] - f_\theta(q_\phi(z))[y]$$
$$\mathcal{L}_{g_1}(z,y;\phi) = -f_\theta(q_\phi(z,y))[y]$$

2.3 Learning Joint Distribution via Optimization of Unconditional Discriminator and Classifier

Following Eq. (3), we can approximate $\log p(x,y)$ by approximating $\log p(y|x)$ and $\log p(x)$. With our energy function f_θ, $p_\theta(y|x)$ can be formulated as:

$$p_\theta(y|x) = \frac{p_\theta(x,y)}{p_\theta(x)} = \frac{\exp(f_\theta(x)[y])}{\sum_{y'}\exp(f_\theta(x)[y'])},$$

which is equivalent to the y'th output of $\texttt{SOFTMAX}(f_\theta(x))$. Therefore, we can maximize the log-likelihood of $p_\theta(y|x)$ by consider f_θ as a softmax classifier minimizing the cross entropy loss:

$$\mathcal{L}_{\text{clf}}(x, y; \theta) = -\log\left(\texttt{SOFTMAX}\left(f_\theta(x)\right)[y]\right)$$

On the other hand, to maximize the log-likelihood of $p(x)$, we introduce a reparameterization $h_\theta(x) = \log \sum_y \exp(f_\theta(x)[y])$:

$$\begin{aligned}
\log p_\theta(x) &= \log\left(\sum_y p_\theta(x, y)\right) = \log\left(\sum_y \frac{\exp(f_\theta(x)[y])}{\int_{x'} \sum_{y'} \exp(f_\theta(x')[y'])\, dx'}\right) \\
&= \log\left(\frac{\exp(\log(\sum_y \exp(f_\theta(x)[y])))}{\int_{x'} \exp(\log(\sum_{y'} \exp(f_\theta(x')[y'])))\, dx'}\right) = \log\left(\frac{\exp(h_\theta(x))}{\int_{x'} \exp(h_\theta(x'))\, dx'}\right) \\
&= h_\theta(x) - \log(Z'(\theta)), \qquad (7)
\end{aligned}$$

where $Z'(\theta) = \int_x \exp(h_\theta(x))\, dx$. Similar to Eq. (5), we can rewrite $\log Z'(\theta)$ by its Fenchel duality:

$$\log Z'(\theta) = \max_q \left[\mathop{\mathbb{E}}_{q(x)}\left[h_\theta(x)\right] + H(q)\right] \qquad (8)$$

where q is a distribution of x and $H(q)$ is the entropy of q.

Combining Eq. (7) and Eq. (8) and reusing the generator in Section 2.2, we obtain the optimization problem:

$$\max_\theta \min_\phi \mathop{\mathbb{E}}_{p_d(x,y)}\left[h_\theta(x)\right] - \mathop{\mathbb{E}}_{p(z)}\left[h_\theta(q_\phi(z, y))\right] - H(q_\phi) \qquad (9)$$

Similar to Eq. (6), the objective of the unconditional discriminator is equivalent to typical GANs augmented with an entropy term. The loss function without considering the entropy can be formulated as:

$$\begin{aligned}
\mathcal{L}_{d_2}(x, z, y; \theta) &= h_\theta(x) - h_\theta(q_\phi(z)) \\
\mathcal{L}_{g_2}(z, y; \phi) &= -h_\theta(q_\phi(z, y))
\end{aligned}$$

2.4 Entropy Approximation in cGANs

In Section 2.2 and Section 2.3, we propose two approaches to train cGANs with and without classification. Unsolved problems in Eq. (6) and Eq. (9) are the entropy terms $H(q_\phi(\cdot, y))$ and $H(q_\phi)$. In previous work, various estimators have been proposed to estimate entropy or its gradient [43, 39, 21, 27]. One can freely choose any approach to estimate the entropy in the proposed framework. In this work, we consider two entropy estimators, and we will show how they connect with existing cGANs.

The first approach is the naive constant approximation. Since entropy is always non-negative, we naturally have the constant zero as a lower bound. Therefore, we can maximize the objective by replacing the entropy term to its lower bound, which is zero in this case. This approach is simple but we will show its effectiveness in Section 4 and how it links our framework to ProjGAN and ContraGAN in Section 3.

The second approach is estimating a variational lower bound. Informally, given a batch of data $\{(x_1, y_1), \ldots, (x_m, y_m)\}$, an encoder function l, and a class embedding function $e(y)$, the negative 2C loss used in ContraGAN [16],

$$\mathcal{L}_C(x_i, y_i; t) = \log\left(\frac{d(l(x_i), e(y_i)) + \sum_{k=1}^m [\![y_k = y_i]\!]\, d(l(x_i), l(x_k))}{d(l(x_i), e(y_i)) + \sum_{k=1}^m d(l(x_i), (l(x_k))}\right), \qquad (10)$$

is an empirical estimate of a proper lower bound of $H(X)$ [37], where $d(a, b) = \exp(a^\top b/t)$ is a distance function with a temperature t. The intuitive behind the 2C loss is pulling the data with the same class closer and pushing the data in difference classes away.

The 2C loss heavily relies on the embeddings $l(x)$ and $e(y)$. Although we only need to estimate the entropy of generated data in Eq. (6) and Eq. (9), we still rely on true data to learn the embeddings in practice. Therefore, the loss function of Eq. (6) can be written as:

$$\begin{aligned}
\mathcal{L}_{D_1}(x, z, y; \theta) &= \mathcal{L}_{d_1}(x, z, y; \theta) + \lambda_c \mathcal{L}_C^{\text{real}} \\
\mathcal{L}_{G_1}(z, y; \phi) &= \mathcal{L}_{g_1}(x, y; \phi) + \lambda_c \mathcal{L}_C^{\text{fake}},
\end{aligned}$$

where λ_c is a hyperparameter controlling the weight of the contrastive loss, and $\mathcal{L}_C^{\text{real}}, \mathcal{L}_C^{\text{fake}}$ are the contrastive loss calculated on a batch of real data and generated data respectively.

Similarly, the loss function of Eq. (9) becomes:

$$\mathcal{L}_{D_2}(x, z, y; \theta) = \mathcal{L}_{d_2}(x, z, y; \theta) + \lambda_c \mathcal{L}_C^{\text{real}}$$
$$\mathcal{L}_{G_2}(z, y; \phi) = \mathcal{L}_{g_2}(x, y; \phi) + \lambda_c \mathcal{L}_C^{\text{fake}},$$

The introduction of 2C loss allows us to accommodate ContraGAN into our framework.

2.5 Energy-based Conditional Generative Adversarial Network

Previous work has shown that multitask training benefits representation learning [30] and training discriminative and generative models jointly outperforms their purely generative or purely discriminative counterparts [12, 28]. Therefore, we propose a framework named Energy-based Conditional Generative Adversarial Network (ECGAN) which combining the two approaches in Section 2.2 and Section 2.3 to learn the joint distribution better. The loss function can be summarized as:

$$\mathcal{L}_D(x, z, y; \theta) = \mathcal{L}_{d_1}(x, z, y; \theta) + \alpha \mathcal{L}_{d_2}(x, z, y; \theta) + \lambda_c \mathcal{L}_C^{\text{real}} + \lambda_{\text{clf}} \mathcal{L}_{\text{clf}}(x, y; \theta) \tag{11}$$
$$\mathcal{L}_G(z, y; \phi) = \mathcal{L}_{g_1}(z, y; \phi) + \alpha \mathcal{L}_{g_2}(z, y; \phi) + \lambda_c \mathcal{L}_C^{\text{fake}} \tag{12}$$

where α is a weight parameter for the unconditional GAN loss.

Here we discuss the intuition of the mechanisms behind each components in Eq. (11). $\mathcal{L}_{d_1}$ is a loss function for conditional discriminator. It updates the y-th output when given a data pair (x, y). $\mathcal{L}_{d_2}$ guides to an unconditional discriminator. It updates all outputs according to whether x is real. $\mathcal{L}_{\text{clf}}$ learns a classifier. It increase the y-th output and decrease the other outputs for data belong to class y. Finally, $\mathcal{L}_C^{real}$ and $\mathcal{L}_C^{fake}$ play the roles to improve the latent embeddings by pulling the embeddings of data with the same class closer.

Previously, we derive the loss functions $\mathcal{L}_{d_1}$ and $\mathcal{L}_{d_2}$ as the loss in Wasserstein GAN [2]. In practice, we use the hinge loss as proposed in Geometric GAN [26] for better stability and convergence. We use the following combination of $\mathcal{L}_{d_1}$ and $\mathcal{L}_{d_2}$:

$$\text{Hinge}(f_\theta(x_{\text{real}}, y) + \alpha \cdot h_\theta(x_{\text{real}}), f_\theta(x_{\text{fake}}, y) + \alpha \cdot h_\theta(x_{\text{fake}})). \tag{13}$$

For more discussion of the implementation of hinge loss, please check Appendix A. The overall training procedure of ECGAN is presented in Appendix C.

3 Accommodation to Existing cGANs

In this section, we show that our framework covers several representative cGAN algorithms, including ACGAN [36], ProjGAN [35], and ContraGAN [16]. Through the ECGAN framework, we obtain a unified view of cGANs, which allowing us to fairly compare and understand the pros and cons of existing cGANs. We name the ECGAN counterparts ECGAN-0, ECGAN-C and ECGAN-E, corresponding to ProjGAN, ACGAN, and ContraGAN, respectively. A summary can be found in Table 1.

Table 1: A summary of cGANs and their closest ECGAN counterpart.

Existing cGAN	ECGAN Counterpart	α	λ_{clf}	λ_c
ProjGAN	ECGAN-0	0	0	0
ACGAN	ECGAN-C	0	> 0	0
ContraGAN	ECGAN-E	0	0	> 0

3.1 ProjGAN

ProjGAN [34] is the most representative cGAN architecture that is commonly used in state-of-the-art research [3, 46]. Let the output of the penultimate layer in the discriminator be $g(x)$. The output of ProjGAN's discriminator is:

$$D(x, y) = w_u^T g(x) + b_u + w_y^T h(x) = (w_u + w_y)^T g(x) + b_u \tag{14}$$

where w_u, b_u are the parameters for the unconditional linear layer, and w_y is the class embedding of y. On the other hand, the output of a discriminator in ECGAN is:

$$D(x, y) = f(x)[y] = (\mathbf{W}^T g(x) + \mathbf{b})[y] = w_y^T g(x) + b_y \tag{15}$$

where $\mathbf{W}, \mathbf{b}$ are the parameters of the linear output layer in f_θ. As shown in Eq. (14) and Eq. (15), the architectures of ProjGAN and ECGAN are almost equivalent. In addition, the loss function of ProjGAN can be formulated as:

$$\begin{aligned}\mathcal{L}_G &= -D(G(z), y)\\ \mathcal{L}_D &= D(x, y) - D(G(z), y),\end{aligned}$$

which is a special case of ECGAN while $\alpha = \lambda_c = \lambda_{\text{clf}} = 0$. We name this case **ECGAN-0**, which is the simplest version of ECGAN. Compared with ProjGAN, ECGAN-0 has additional bias terms for the output of each class.

3.2 ACGAN

ACGAN [36] is the most well-known cGAN algorithm that leverage a classifier to achieve conditional generation. Given a K-class dataset, the discriminator of ACGAN is parameterized by a network with $K + 1$ outputs. The first output, denoted as $D(x)$, is an unconditional discriminator distinguishing between real and fake images. The remaining K outputs, denoted as $C(x)$, is a classifier that predicts logits for every classes. The loss function of ACGAN can be formulated as:

$$\begin{aligned}\mathcal{L}_G &= -D(G(z)) + \lambda_g \mathcal{L}_{\text{clf}}(G(z), y; C)\\ \mathcal{L}_D &= D(x) - D(G(z)) + \lambda_d(\mathcal{L}_{\text{clf}}(x, y; C) + \mathcal{L}_{\text{clf}}(G(z), y; C))\end{aligned}$$

where G is the generator, λ_g and λ_d are hyperparameters to control the weight of cross entropy loss.

The formulation of ACGAN is similar to our ECGAN when $\alpha = \lambda_c = 0$ and $\lambda_{\text{clf}} > 0$. We call the special case as **ECGAN-C**, with a siffix 'C' for classification loss. The main difference between ACGAN and ECGAN-C is that ECGAN-C uses a conditional discriminator which plays the role of a classifier at the same time. Hence the generator in ECGAN-C learns from the conditional discriminator rather than the cross entropy loss which is biased for generative objective.

3.3 ContraGAN

ContraGAN [16] proposed 2C loss, which we mentioned in Eq. (10), to improve ProjGAN by capturing the data-to-data relationship and data-to-label relationship. The 2C loss is applied in both discriminator and generator to achieve conditional generation. That is:

$$\begin{aligned}\mathcal{L}_G &= -D(G(z), y) + \lambda_c \mathcal{L}_C^{\text{fake}}\\ \mathcal{L}_D &= D(x, y) - D(G(z), y) + \lambda_c \mathcal{L}_C^{\text{real}}\end{aligned}$$

The loss functions are the same as ECGAN with $\alpha = \lambda_{\text{clf}} = 0$ and $\lambda_c > 0$. We call it **ECGAN-E**, where 'E' means entropy estimation. The main discrepancy between ContraGAN and ECGAN-E is the output layer of their discriminators. While ContraGAN uses a single-output network, ECGAN uses a K-output network f_θ which has higher capacity.

4 Experiment

4.1 Dataset and Evaluation Metric

We conduct our conditional generation experiments with two dataset: CIFAR-10 [20] as a common benchmark and Tiny ImageNet [22] as a higher resolution and more challenging condition generation task. Both datasets are distributed under the MIT license and do not contain any human-related class hence no personally identifiable information or offensive content in them.

CIFAR-10 [20] is a popular benchmark dataset in image tasks. It contains 50,000 images for training and 10,000 images for testing. Each image contains 32×32 pixels with 3 color channels. The dataset is categorized into 10 different classes.

Table 2: Ablation study of ECGAN on CIFAR-10 and Tiny ImageNet. The label cGAN means the vanilla version of ECGAN where $\alpha = \lambda_{\text{clf}} = \lambda_c = 0$. The label U stands for unconditional gan loss ($\alpha > 0$). C means classification loss ($\lambda_{\text{clf}} > 0$). E means entropy estimation loss via contrastive learning ($\lambda_c > 0$).

Dataset	ECGAN Variant	FID (↓)	IS (↑)	Intra-FID (↓)
CIFAR-10	ECGAN-0	8.049 ± 0.092	9.759 ± 0.061	41.708 ± 0.278
	ECGAN-U	**7.915** ± 0.095	9.967 ± 0.078	41.430 ± 0.326
	ECGAN-C	7.996 ± 0.120	9.870 ± 0.157	41.715 ± 0.307
	ECGAN-UC	7.942 ± 0.041	**10.002** ± 0.120	41.425 ± 0.221
	ECGAN-UCE	8.039 ± 0.161	9.898 ± 0.064	**41.371** ± 0.278
Tiny ImageNet	ECGAN-0	24.077 ± 1.660	16.173 ± 0.671	214.811 ± 3.627
	ECGAN-U	20.876 ± 1.651	15.318 ± 1.148	215.117 ± 7.034
	ECGAN-C	24.853 ± 3.902	16.554 ± 1.500	212.661 ± 8.135
	ECGAN-UC	**18.919** ± 0.774	**18.442** ± 1.036	**203.373** ± 5.101
	ECGAN-UCE	24.728 ± 0.974	17.935 ± 0.619	209.547 ± 1.968

Tiny ImageNet [22] provides 120,000 color images with 200 classes. Each image contains 64×64 pixels. Each class contains 500, 50, 50 images for training, validation, testing, respectively. We use this dataset to demonstrate the effectiveness of ECGAN in a much more challenging conditional generation task with higher resolution images and the scarcity of training data in each class.

In our experiment, we use two common metrics, **Frechet Inception Distance** [FID; 14] and **Inception Score** [IS; 41], to evaluate our generation quality and diversity. Besides, we use **Intra-FID**, which is the average of FID for each class, to evaluate the performance of conditional generation.

4.2 Experimental Setup

We use StudioGAN[1] [16] to conduct our experiments. StudioGAN is a PyTorch-based project distributed under the MIT license that provides implementation and benchmark of several popular GAN architecture and techniques. To provide reliable evaluation, we conduct all experiments with 4 different random seeds and report the means and standard deviations for each metric. We evaluate the model with the lowest FID for each trial. The default architecture is BigGAN [3]. We fix the learning rate for generators and discriminators to 0.0001 and 0.0004, respectively, and tune λ_{clf} in $\{1, 0.1, 0.05, 0.01\}$. We follow the setting $\lambda_c = 1$ in [16] when using 2C loss, and set $\alpha = 1$ when applying unconditional GAN loss. All experiments are run on single GPU (Nvidia Tesla V100) machines. Each process takes about 36 GPU hours. The total amount of computation takes about 14,400 GPU hours. For more detail of the experimental setting, please check Appendix B.

4.3 Ablation Study

We start our empirical studies from investigating the effectiveness of each component in ECGAN. We use symbols 'U' to represent unconditional GAN loss, 'C' to represent classification loss and 'E' to represent entropy estimation loss, which is 2C loss in our implementation. The concatenation of the symbols indicates the combination of losses. For example, ECGAN-UC means ECGAN with both unconditional GAN loss and classification loss ($\alpha > 0$ and $\lambda_{\text{clf}} > 0$). Table 2 shows the results of ECGAN from the simplest ECGAN-0 to the most complicated ECGAN-UCE. On CIFAR-10, ECGAN-0 already achieves decent results. Adding unconditional loss, classification loss or contrastive loss provides slightly better or on-par performance. On the harder Tiny Imagenet, the benefit of unconditional loss and classification loss become more significant. While ECGAN-U already shows advantages to ECGAN-0, adding classification loss to ECGAN-U further improves all metrics considerably. We can also observe that directly adding classification loss is not sufficient to improve cGAN, just the same to the results mentioned in [34]. The fact reveals that the unconditional GAN loss is a crucial component to bridge classifiers and discriminators in cGANs. We also observe that adding contrastive loss doesn't improve ECGAN-UC. An explanation is that the entropy estimation lower bound provided by the contrastive loss is too loose to benefit the training. Furthermore, the additional parameters introduced by 2C loss make the optimization problem more complicated. As

[1] https://github.com/POSTECH-CVLab/PyTorch-StudioGAN

Table 3: Comparison between cGAN variants with different architectures on CIFAR-10 and Tiny ImageNet

Dataset	Backbone	method	FID (↓)	IS (↑)	Intra-FID (↓)
CIFAR-10	DCGAN	ACGAN	32.507 ± 2.174	7.621 ± 0.088	129.603 ± 1.212
		ProjGAN	21.918 ± 1.580	8.095 ± 0.185	68.164 ± 2.055
		ContraGAN	28.310 ± 1.761	7.637 ± 0.125	153.730 ± 9.965
		ECGAN-UC	**18.035** ± 0.788	**8.487** ± 0.131	**59.343** ± 1.557
	ResGAN	ACGAN	10.073 ± 0.274	9.512 ± 0.050	48.464 ± 0.716
		ProjGAN	10.195 ± 0.203	9.268 ± 0.139	46.598 ± 0.070
		ContraGAN	10.551 ± 0.976	9.087 ± 0.228	138.944 ± 12.582
		ECGAN-UC	**9.244** ± 0.062	**9.651** ± 0.098	**43.876** ± 0.384
	BigGAN	ACGAN	8.615 ± 0.146	9.742 ± 0.041	45.243 ± 0.129
		ProjGAN	8.145 ± 0.156	9.840 ± 0.080	42.110 ± 0.405
		ContraGAN	8.617 ± 0.671	9.679 ± 0.210	114.602 ± 13.261
		ECGAN-UC	**7.942** ± 0.041	**10.002** ± 0.120	**41.425** ± 0.221
Tiny ImageNet	BigGAN	ACGAN	29.528 ± 4.612	12.964 ± 0.770	315.408 ± 1.171
		ProjGAN	28.451 ± 2.242	12.213 ± 0.624	242.332 ± 11.447
		ContraGAN	24.915 ± 1.222	13.445 ± 0.371	257.657 ± 3.246
		ECGAN-UC	**18.780** ± 1.291	**17.475** ± 1.052	**204.830** ± 5.648

a result, we use the combination ECGAN-UC as the default option of ECGAN in the following experiments.

4.4 Comparison with Existing cGANs

We compare ECGAN to several representative cGANs including ACGAN [36], ProjGAN [34], and ContraGAN [16], with three representative network architectures: DCGAN [38], ResNet [13], and BigGAN [3]. Table 3 compares the results of each combinations of cGAN algorithms and network architectures. The results show that ECGAN-UC outperforms other cGANs significantly with all architectures on both CIFAR-10 and Tiny ImageNet. We also noticed that ContraGAN, though achieves decent image quality and diversity, learns a conditional generator which interchanges some classes while generating, hence has low Intra-FID. Overall, the experiment indicates that ECGAN-UC can be a preferred choice for cGAN in general situation.

4.5 Compare Existing cGANs to their ECGAN Counterpart

Table 4 compares ProjGAN, ContraGAN, ACGAN to their ECGAN counterpart. As we described in Section 3, each of these representative cGANs can be viewed as special cases under our ECGAN framework. As mentioned in Section 3, ECGAN-0 has additional bias terms in the output layer compared to ProjGAN. The results in Table 4 shows that the subtle difference still bring significant improvement to the generation quality, especially on the harder Tiny ImageNet.

Compared to ContraGAN, ECGAN-E has the same loss design but different architecture in the discriminator's output layer. While the discriminator of ContraGAN has only single output, ECGAN-E has multiple output for every classes. The difference makes ECGAN-E solve the label mismatching problem of ContraGAN mentioned in Section 4.4 and benefits generation on CIFAR-10, but does not work well on Tiny ImageNet. It is probably because of the scarcity of training data in each class in Tiny ImageNet. Only 50 data are available for updating the parameters corresponding to each class.

Last, we compare ECGAN-C to ACGAN. Both of them optimize a GAN loss and a classification loss. However, ECGAN-C combines the discriminator and the classifier, so the generator can directly optimize cGAN loss rather than the classification loss. As a result, ECGAN-C demonstrates better performance on both CIFAR-10 and Tiny ImageNet. In sum, the comparisons show that through the unified view provided by ECGAN, we can improve the existing methods with minimal modifications.

5 Related Work

The development of cGANs started from feeding label embeddings to the inputs of GANs or the feature vector at some middle layer [33, 7]. To improve the generation quality, ACGAN [36] proposes

Table 4: Compare between representative cGANs and their ECGAN counterparts.

Dataset	method	FID (↓)	IS (↑)	Intra-FID (↓)
CIFAR-10	ProjGAN	8.145 $\pm$ 0.156	**9.840** $\pm$ 0.080	42.110 $\pm$ 0.405
	ECGAN-0	**8.049** $\pm$ 0.092	9.759 $\pm$ 0.061	**41.708** $\pm$ 0.278
	ContraGAN	8.617 $\pm$ 0.671	9.679 $\pm$ 0.210	114.602 $\pm$ 13.261
	ECGAN-E	**8.038** $\pm$ 0.102	**9.876** $\pm$ 0.036	**41.155** $\pm$ 0.277
	ACGAN	8.615 $\pm$ 0.146	9.742 $\pm$ 0.041	45.243 $\pm$ 0.129
	ECGAN-C	**8.102** $\pm$ 0.039	**9.980** $\pm$ 0.093	**41.109** $\pm$ 0.273
Tiny ImageNet	ProjGAN	28.451 $\pm$ 2.242	12.213 $\pm$ 0.624	242.332 $\pm$ 11.447
	ECGAN-0	24.077 $\pm$ 1.660	16.173 $\pm$ 0.671	214.811 $\pm$ 3.627
	ContraGAN	**24.915** $\pm$ 1.222	**13.445** $\pm$ 0.371	257.657 $\pm$ 3.246
	ECGAN-E	38.270 $\pm$ 1.174	12.576 $\pm$ 0.405	239.184 $\pm$ 2.628
	ACGAN	29.528 $\pm$ 4.612	12.964 $\pm$ 0.770	315.408 $\pm$ 1.171
	ECGAN-C	**24.853** $\pm$ 3.902	**16.554** $\pm$ 1.500	**212.661** $\pm$ 8.135

to leverage classifiers and successfully generates high resolution images. The use of classifiers in GANs can also be found in Triple GAN [24] for semi-supervised learning and Triangle GAN [9] for cross-domain distribution matching. However, Shu [42] and Miyato and Koyama [34] pointed out that the auxiliary classifier in ACGAN misleads the generator to generate images that are easier to be classified. Thus, whether classifiers can help conditional generation still remains questionable.

In this work, we connect cGANs with and without classifiers via an energy model parameterization from the joint probability perspective. [11] use similar ideas but focus on sampling from the trained classifier via Markov Chain Monte Carlo [MCMC; 1]. Our work is also similar to a concurrent work [12], which improve [11] by introducing Fenchel duality to replace computationally-intensive MCMC. They use a variational approach [19] to formulate the objective for tractable entropy estimation. In contrast, we study the GAN perspective and the entropy estimation via contrastive learning. Therefore, the proposed ECGAN can be treated a complements works compared with Grathwohl et al. [11, 12] by studying a GAN perspective. We note that, the studied cGAN approaches also results in better generation quality than its variational alternative [12].

Last, [5] study the connection between exponential family and unconditional GANs. Different from [5], we study the conditional GANs with the focus to provide a unified view of common cGANs and an insight to the role of classifiers in cGANs.

6 Conclusion

In this work, we present a general framework Energy-based Conditional Generative Networks (ECGAN) to train cGANs with classifiers. With the framework, we can explain representative cGANs including ACGAN, ProjGAN, and ContraGAN in a unified view. The experiments demonstrate that ECGAN-UC outperforms state-of-the-art cGANs with different network architectures on CIFAR-10 and Tiny ImageNet. Further investigation can be conduct to find a better entropy approximation or improve cGANs by advanced techniques for classifiers. We hope this work can pave the way to more advanced cGAN algorithms in the future.

7 Limitations and Potential Negative Impacts

There are two main limitations in the current study. One is the scale of the experiments. Ideally experimenting with a larger dataset such as the full ImageNet can further strengthen the contribution. But training with such a large dataset is not affordable for our computational resource, and we can only resort to the conclusive findings in the current benchmark datasets. The other limitation is whether the metrics such as FID truly reflect generation quality, but this limitation is considered an open problem to the community anyway.

As with any work on generative models, there is a potential risk of the proposed model being misused to create malicious contents, much like how misused technology can be used to forge bills. In this sense, more anti-forgery methods will be needed to mitigate the misuse in the future.

References

[1] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50(1):5–43, 2003.

[2] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *CoRR*, abs/1701.07875, 2017.

[3] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[4] Grigorios G. Chrysos, Jean Kossaifi, and Stefanos Zafeiriou. Rocgan: Robust conditional GAN. *Int. J. Comput. Vis.*, 128(10):2665–2683, 2020.

[5] Bo Dai, Zhen Liu, Hanjun Dai, Niao He, Arthur Gretton, Le Song, and Dale Schuurmans. Exponential family estimation via adversarial dynamics embedding. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 10977–10988, 2019.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[7] Emily L. Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1486–1494, 2015.

[8] William Fedus, Mihaela Rosca, Balaji Lakshminarayanan, Andrew M. Dai, Shakir Mohamed, and Ian J. Goodfellow. Many paths to equilibrium: Gans do not need to decrease a divergence at every step. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[9] Zhe Gan, Liqun Chen, Weiyao Wang, Yunchen Pu, Yizhe Zhang, Hao Liu, Chunyuan Li, and Lawrence Carin. Triangle generative adversarial networks. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5247–5256, 2017.

[10] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[11] Will Grathwohl, Jacob Kelly, Milad Hashemi, Mohammad Norouzi, Kevin Swersky, and David Duvenaud. No MCMC for me: Amortized sampling for fast and stable training of energy-based models. *CoRR*, abs/2010.04230, 2020.

[12] Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[13] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5767–5777, 2017.

[14] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6626–6637, 2017.

[15] Mingqi Hu, Deyu Zhou, and Yulan He. Variational conditional GAN for fine-grained controllable image generation. In Wee Sun Lee and Taiji Suzuki, editors, *Proceedings of The 11th Asian Conference on Machine Learning, ACML 2019, 17-19 November 2019, Nagoya, Japan*, volume 101 of *Proceedings of Machine Learning Research*, pages 109–124. PMLR, 2019.

[16] Minguk Kang and Jaesik Park. Contragan: Contrastive learning for conditional image generation. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[17] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[18] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1857–1865. PMLR, 2017.

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[20] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

[21] Rithesh Kumar, Anirudh Goyal, Aaron C. Courville, and Yoshua Bengio. Maximum entropy generators for energy-based models. *CoRR*, abs/1901.08508, 2019.

[22] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7:7, 2015.

[23] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. 2006.

[24] Chongxuan Li, Taufik Xu, Jun Zhu, and Bo Zhang. Triple generative adversarial nets. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4088–4098, 2017.

[25] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. MMD GAN: towards deeper understanding of moment matching network. *CoRR*, abs/1705.08584, 2017.

[26] Jae Hyun Lim and Jong Chul Ye. Geometric GAN. *CoRR*, abs/1705.02894, 2017.

[27] Jae Hyun Lim, Aaron C. Courville, Christopher J. Pal, and Chin-Wei Huang. AR-DAE: towards unbiased neural entropy gradient estimation. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 6061–6071. PMLR, 2020.

[28] Hao Liu and Pieter Abbeel. Hybrid discriminative-generative training via contrastive learning. *CoRR*, abs/2007.09070, 2020.

[29] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2813–2821. IEEE Computer Society, 2017.

[30] Andreas Maurer, Massimiliano Pontil, and Bernardino Romera-Paredes. The benefit of multitask representation learning. *J. Mach. Learn. Res.*, 17:81:1–81:32, 2016.

[31] Lars M. Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3478–3487. PMLR, 2018.

[32] Daniel Michelsanti and Zheng-Hua Tan. Conditional generative adversarial networks for speech enhancement and noise-robust speaker verification. In Francisco Lacerda, editor, *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, August 20-24, 2017*, pages 2008–2012. ISCA, 2017.

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

[34] Takeru Miyato and Masanori Koyama. cgans with projection discriminator. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[35] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[36] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 2642–2651. PMLR, 2017.

[37] Ben Poole, Sherjil Ozair, Aäron van den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5171–5180. PMLR, 2019.

[38] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[39] Rajesh Ranganath, Dustin Tran, and David M. Blei. Hierarchical variational models. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48, pages 324–333. JMLR.org, 2016.

[40] Scott E. Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1060–1069. JMLR.org, 2016.

[41] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2226–2234, 2016.

[42] Rui Shu. Ac-gan learns a biased distribution. 2017.

[43] Shashank Singh and Barnabás Póczos. Analysis of k-nearest neighbor distances with application to entropy estimation. *CoRR*, abs/1603.08578, 2016.

[44] Martin J. Wainwright and Michael I. Jordan. Graphical models, exponential families, and variational inference. *Found. Trends Mach. Learn.*, 1(1-2):1–305, 2008.

[45] Yasin Yazici, Chuan-Sheng Foo, Stefan Winkler, Kim-Hui Yap, Georgios Piliouras, and Vijay Chandrasekhar. The unusual effectiveness of averaging in GAN training. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[46] Han Zhang, Ian J. Goodfellow, Dimitris N. Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 7354–7363. PMLR, 2019.

[47] Han Zhang, Zizhao Zhang, Augustus Odena, and Honglak Lee. Consistency regularization for generative adversarial networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[48] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient GAN training. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[49] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2242–2251. IEEE Computer Society, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] Section 1
   (b) Did you describe the limitations of your work? [Yes] Section 7
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Section 7
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Supplemental material
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Section 4.1, Section 4.2, Appendix B
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Table 2, Table 3, Table 4
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Section 4.2
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] Section 4.1, Section 4.2

(b) Did you mention the license of the assets? [Yes] Section 4.1, Section 4.2

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Supplemental material

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] Section 4.1, Section 4.2. All assets are distributed under the MIT license.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] Section 4.1

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Implementation Issue of Hinge Loss

In Section 2.2 and Section 2.3, we derive the loss functions $\mathcal{L}_{d_1}$ and $\mathcal{L}_{d_2}$ as the loss in Wasserstein GAN [2]. In practice, we use the hinge loss as proposed in Geometric GAN [26] for better convergence. An intuitive combination of $\mathcal{L}_{d_1}$ and $\mathcal{L}_{d_2}$ can be as following:

$$\text{Hinge}(f_\theta(x_{\text{real}}, y), f_\theta(x_{\text{fake}}, y)) + \alpha \cdot \text{Hinge}(h_\theta(x_{\text{real}}), h_\theta(x_{\text{fake}})), \tag{16}$$

where $\text{Hinge}(\cdot)$ is the hinge loss function proposed in [26].

The property of the hinge loss encourages the output value of $f_\theta(x_{\text{real}}, y), h_\theta(x_{\text{real}})$ to 1, and $f_\theta(x_{\text{fake}}, y), h_\theta(x_{\text{fake}})$ to -1, which leads to better stability in optimization generally. However, since $h_\theta(x) = \log \sum_y \exp(f_\theta(x)[y])$ by definition, we notice that encouraging the output of both functions into the same scale harms the optimization. Therefore, we use the following combination instead:

$$\text{Hinge}(f_\theta(x_{\text{real}}, y) + \alpha \cdot h_\theta(x_{\text{real}}), f_\theta(x_{\text{fake}}, y) + \alpha \cdot h_\theta(x_{\text{fake}})). \tag{17}$$

The new formulation leads to more stable optimization and is less sensitive to the parameter α empirically.

B Experimental Setup Details

We use hinge loss [26] and apply spectral norm [35] on all models to stabilize the training. We adopt self-attention technique [46] and horizontal random flipping [48] to provide better generation quality. We apply moving average update [17, 31, 45] for generators after 1,000 generator updates for CIFAR-10 and 20,000 generator updates for Tiny ImageNet with decay rate 0.9999. We follow the setting of 2C-loss in [16], using $\lambda_c = 1$ and 512-dimension linear projection layer for CIFAR-10 and 768-dimension linear projection layer for Tiny ImageNet. We use Adam [19] optimizer with batch size 64 for CIFAR-10 and batch size 256 for Tiny ImageNet. The training takes 150,000 steps for CIFAR-10 and 100,000 steps for Tiny ImageNet.

C Training Algorithm

Input: Unconditional GAN loss weight: α. 2C loss weight: λ_c. Classification loss weight: λ_{clf}. Parameters of the discriminator and the generator: (θ, ϕ).
Output: (θ, ϕ)

Initialize (θ, ϕ)
for $\{1, \ldots, n_{iter}\}$ **do**
  for $\{1, \ldots, n_{dis}\}$ **do**
    Sample $\{(x_i, y_i)\}_{i=1}^m \sim p_d(x, y)$
    Sample $\{z_i\}_{i=1}^m \sim p(z)$
    Calculate $\mathcal{L}_D$ by Eq. (11)
    $\theta \longleftarrow \text{Adam}(\mathcal{L}_D, lr_d, \beta_1, \beta_2)$
  end for
  Sample $\{(y_i)\}_{i=1}^m \sim p_d(y)$ and $\{z_i\}_{i=1}^m \sim p(z)$
  Calculate $\mathcal{L}_G$ by Eq. (12)
  $\phi \longleftarrow \text{Adam}(\mathcal{L}_G, lr_g, \beta_1, \beta_2)$
end for

D Images Generated by ECGAN-UC

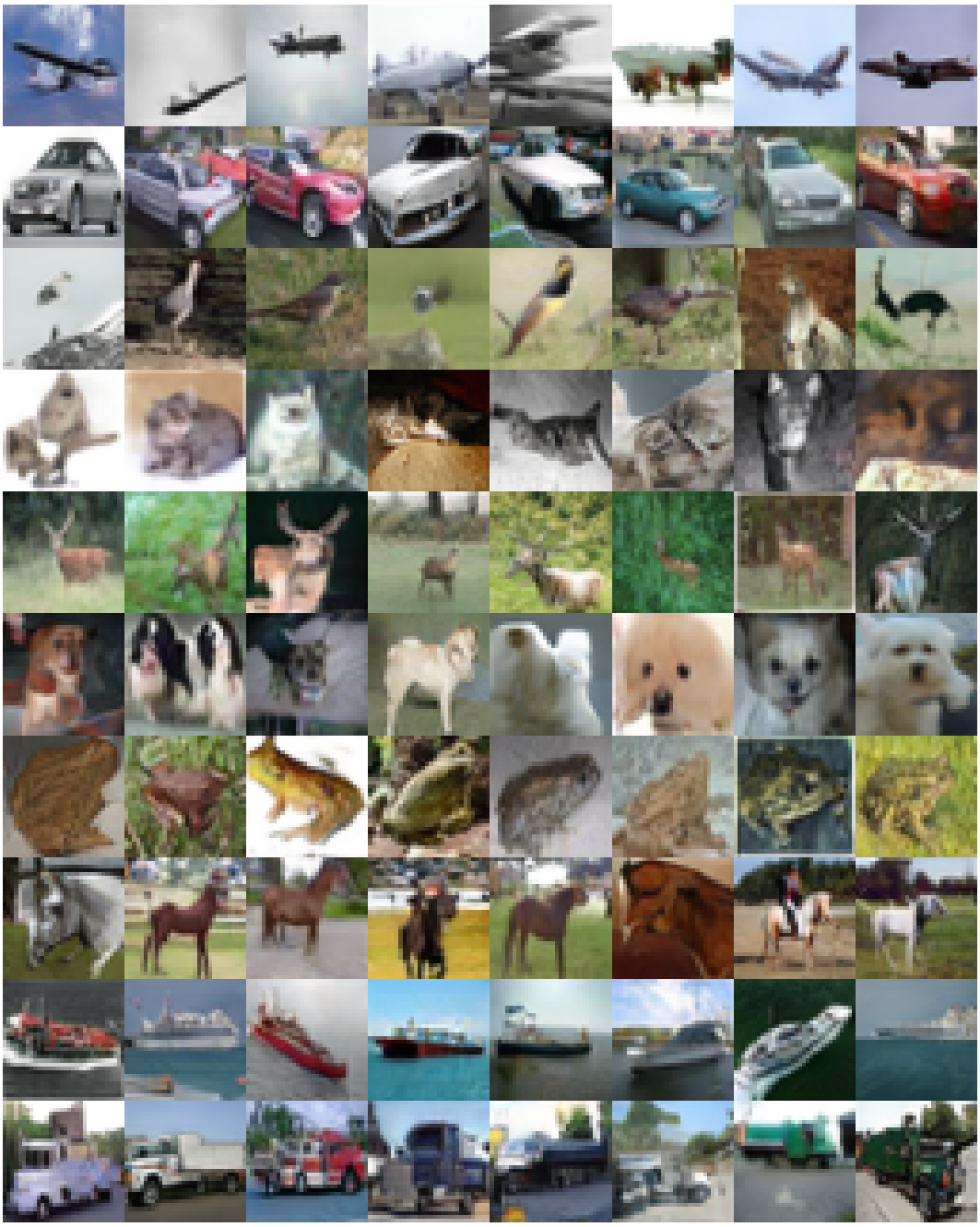

Figure 1: CIFAR-10 images generated by ECGAN (FID: 7.89, Inception Score: 10.06, Intra-FID: 41.42)

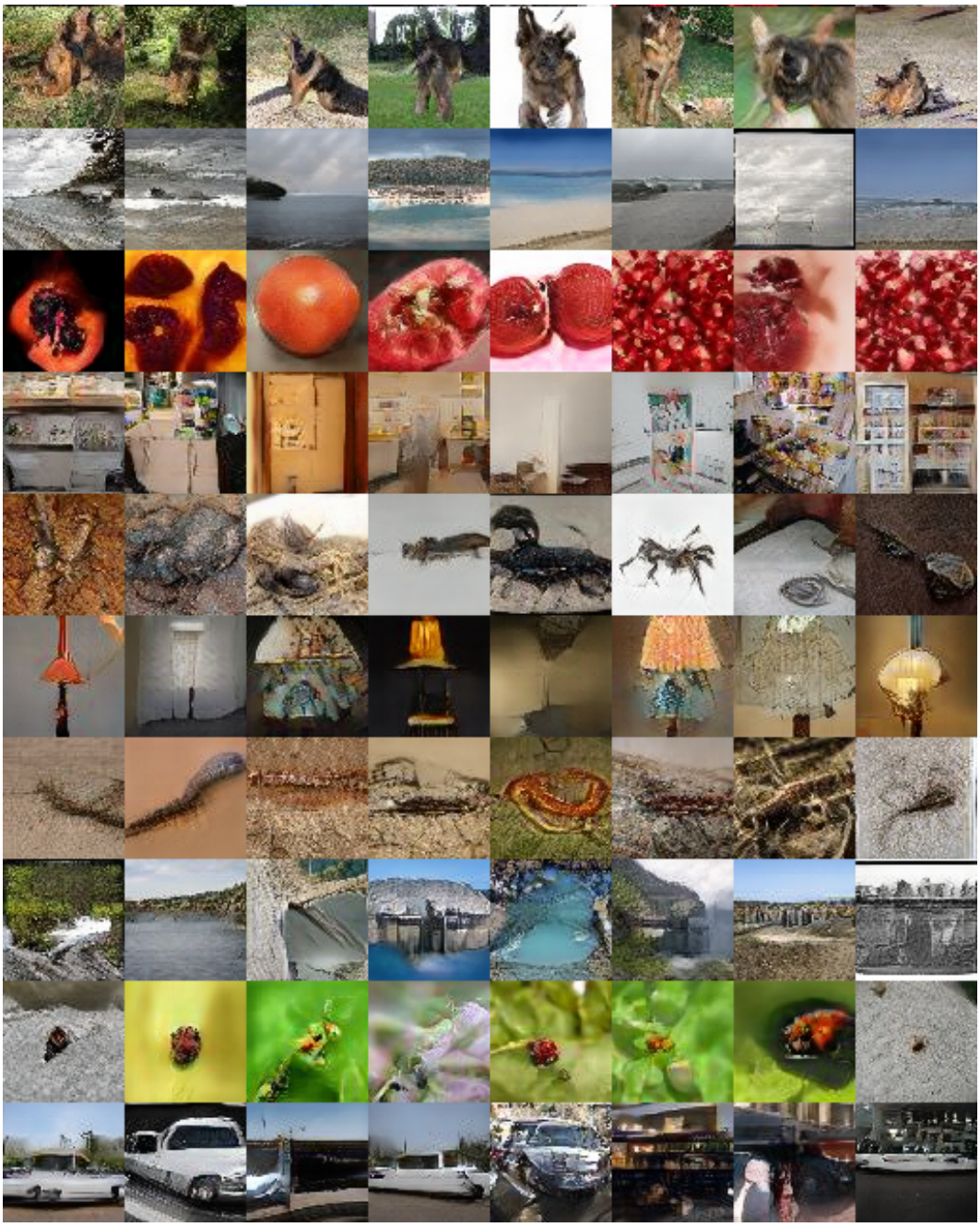

Figure 2: Tiny ImageNet images generated by ECGAN (FID: 17.16, Inception Score: 17.77, Intra-FID: 201.66)