A Unified View of cGANs with and without Classifiers

Si-An Chen
National Taiwan University
d09922007@csie.ntu.edu.tw

Chun-Liang Li
Google Cloud AI
chunliang@google.com

Hsuan-Tien Lin
National Taiwan University
htlin@csie.ntu.edu.tw

Abstract

Conditional Generative Adversarial Networks (cGANs) are implicit generative models which allow to sample from class-conditional distributions. Existing cGANs are based on a wide range of different discriminator designs and training objectives. One popular design 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 shortcoming and reach state-of-the-art performance without having classifiers. Somehow it remains unanswered 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 cGANs and classification as a unified framework. The framework, along with a classic energy model to parameterize distributions, justifies the use of classifiers for cGANs in a principled manner. It explains several popular cGAN variants, such as ACGAN, ProjGAN, and ContraGAN, as special cases with different levels of approximations, which provides a unified view and brings new insights to understanding cGANs. Experimental results demonstrate that the design inspired by the proposed framework outperforms state-of-the-art cGANs on multiple benchmark datasets, especially on the most challenging ImageNet. The code is available at https://github.com/sian-chen/PyTorch-ECGAN.

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, 37, 29, 25, 36, 26, 38], backbone architectures [41, 50], and regularization techniques [13, 35, 51] 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, 39, 34, 16]. The capability of conditional generation extends the application horizon of GANs to conditional image generation based on labels [39] or texts [43], speech enhancement [32], and image style transformation [18, 53].

One representative cGAN is Auxiliary Classifier GAN [ACGAN; 39], 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

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

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 to generate 1000 classes of ImageNet images [6]. That is, ACGAN used to be a leading cGAN design. While the classifier in ACGAN indeed improves the quality of conditional generation, deeper studies revealed that the classifier biases the generator to generate easier-to-classify images [45], which in term decreases the capability to match the target distribution.

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 accurately generates images in target classes 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 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 an 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 **E**nergy-based **C**onditional **G**enerative **A**dversarial **N**etworks (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 [39], 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 backbone architectures on benchmark datasets, including the most challenging ImageNet.

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 backbone 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 [39], 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 outputs higher values for real data. The choice of D leads to different types 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 [39]. 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 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 Optimizing Conditional Discriminators

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 [48] 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 [48] 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 $H(q_y) = -\mathbb{E}_{xf\sim q_y(x)}\left[\log q_y(x)\right]$ is the entropy of q_y. The derivation is provided in Appendix A. 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 Distributions via Optimizing Unconditional Discriminators & Classifiers

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 $h_\theta(x) = \log \sum_y \exp(f_\theta(x)[y])$:

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

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

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

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:

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

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 [46, 42, 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 with 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],

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

is an empirical estimate of a proper lower bound of $H(X)$ [40], where $d(a, b) = \exp(a^\top b/t)$ is a distance function with a temperature t. We provide the proof in Appendix B.

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

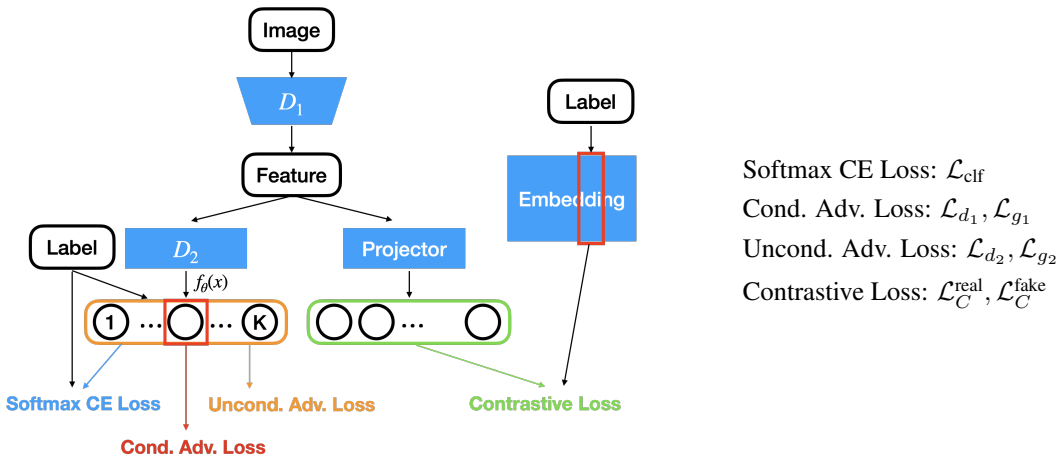

Figure 1: The discriminator's design of ECGAN. D_1 can be any network backbone such as DCGAN, ResGAN, BigGAN. D_2 is a linear layer with K outputs, where K is the number of classes.

practice. Therefore, the loss function of Eq. (6) can be written as:

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

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 [11, 28]. Therefore, we propose a framework named Energy-based Conditional Generative Adversarial Network (ECGAN), which combines 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. The discriminator's design is illustrated in Fig 1.

Here we discuss the intuition of the mechanisms behind each component 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 increases the y-th output and decreases the other outputs for data belonging 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 C. The overall training procedure of ECGAN is presented in Appendix E.

3 Accommodation to Existing cGANs

In this section, we show that our framework covers several representative cGAN algorithms, including ACGAN [39], ProjGAN [35], and ContraGAN [16]. Through the ECGAN framework, we obtain a unified view of cGANs, which allows 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. We summarize the settings in Table 1 and illustrate the discriminator designs in Appendix F.

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

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

3.1 ProjGAN

ProjGAN [34] is the most representative cGAN design that is commonly used in state-of-the-art research [3, 50]. Let the output of the penultimate layer in the discriminator be $g(x)$. The output of ProjGAN's discriminator is:

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

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 [39] is the most well-known cGAN algorithm that leverages 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 class. 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 suffix 'C' for classification loss. 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 objectives.

Dataset	# training	# test	# classes	Resolution	# training data per class
CIFAR-10	50,000	10,000	10	32×32	5,000
Tiny ImageNet	100,000	10,000	200	64×64	500
ImageNet	1,281,167	50,000	1,000	128×128	1,281

Table 2: Datasets for evaluation.

3.3 ContraGAN

ContraGAN [16] proposed 2C loss, which we mentioned in Eq. (10), to capture 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:

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

The loss functions are similar to ones in ECGAN with $\alpha = \lambda_{\text{clf}} = 0$ and $\lambda_c > 0$. We call it **ECGAN-E**, where 'E' means entropy estimation. The main difference 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.

We keep Eq. (11) and Eq. (12) as simple as possible to reduce the burden of hyperparameter tuning. Under the simple equations of the current framework, ECGAN-C and ECGAN-E are the closest counterparts to ACGAN and ContraGAN. The subtle difference (in addition to the underlying network architecture) is that ACGAN uses $\mathcal{L}_{d_2}$ instead of $\mathcal{L}_{d_1}$ (ECGAN-C); ContraGAN uses $\mathcal{L}_{d_2}, \mathcal{L}_{g_2}$ instead of $\mathcal{L}_{d_1}, \mathcal{L}_{g_1}$ (ECGAN-E). One future direction is to introduce more hyperparameters in Eq. (11) and Eq. (12) to get closer counterparts.

4 Experiment

We conduct our experiments on CIFAR-10 [20] and Tiny ImageNet [22] for analysis, and ImageNet [6] for large-scale empirical study. Table 2 shows the statistics of the datasets. All datasets are publicly available for research use. They were not constructed for human-related study. We do not specifically take any personal information from the datasets in our experiments.

In our experiment, we use two common metrics, Frechet Inception Distance [FID; 14] and Inception Score [IS; 44], 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.

4.1 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 architectures and techniques. To provide reliable evaluation, we conduct experiments on CIFAR-10 and Tiny ImageNet 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 backbone 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. The experiments take 1-2 days on single GPU (Nvidia Tesla V100) machines for CIFAR-10, Tiny ImageNet, and take 6 days on 8-GPU machines for ImageNet. More details are described in Appendix D.

4.2 Ablation Study

We start our empirical studies by investigating the effectiveness of each component in ECGAN. We use symbols 'U' to represent unconditional GAN loss, 'C' to represent classification loss, and 'E'

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

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

Table 3: Ablation study of ECGAN on CIFAR-10 and Tiny ImageNet. ECGAN-0 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 ($\lambda_c > 0$).

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

4.3 Comparison with Existing cGANs

We compare ECGAN to several representative cGANs including ACGAN [39], ProjGAN [34], and ContraGAN [16], with three representative backbone architectures: DCGAN [41], ResNet [13], and BigGAN [3]. Table 4 compares the results of each combinations of cGAN algorithms and backbone architectures. The results show that ECGAN-UC outperforms other cGANs significantly with all backbone architectures on both CIFAR-10 and Tiny ImageNet. We also noticed that ContraGAN, though achieves decent image quality and diversity, learns a conditional generator that 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 situations.

4.4 Comparisons between Existing cGANs and their ECGAN Counterpart

Table 5 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 5 shows that the subtle difference still brings significant improvement to the generation quality, especially on the harder Tiny ImageNet.

Compared to ContraGAN, ECGAN-E has the same loss but different design in the discriminator's output layer. While the discriminator of ContraGAN has only single output, ECGAN-E has multiple outputs for every class. The difference makes ECGAN-E solve the label mismatching problem of ContraGAN mentioned in Section 4.3 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.

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

Table 4: Comparison between cGAN variants with different backbone architectures on CIFAR-10 and Tiny ImageNet

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

Table 5: Compare between representative cGANs and their ECGAN counterparts.

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.

4.5 Evaluation on ImageNet

We compare our ECGAN-UC and ECGAN-UCE with BigGAN [3] and ContraGAN [16] on ImageNet. We follow all configurations of BigGAN with batch size 256 in StudioGAN. The results in Table 6 show that ECGAN-UCE outperforms other cGANs dramatically. The comparison between ECGAN-UC and ECGAN-UCE indicates that the 2C loss brings more significant improvement in the ECGAN framework than in ContraGAN. To the best of our knowledge, this is a state-of-the-art result of GANs with batch size 256 on ImageNet. Selected generated images are shown in Appendix G.

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 [39] proposes 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 [45] and Miyato and Koyama [34] pointed out

Method	FID(↓)	IS(↑)
BigGAN*	24.68	28.63
ContraGAN*	25.16	25.25
ECGAN-UC	30.05	26.47
ECGAN-UCE	12.16	56.33
ECGAN-UCE (40k step)	**8.49**	**80.69**

Table 6: Evaluation on ImageNet128×128. (*: Reported by StudioGAN.)

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. [12] 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 [11], which improves [12] 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 [12, 11] by studying a GAN perspective. We note that the studied cGAN approaches also result in better generation quality than its variational alternative [11].

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 into 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 outperforms state-of-the-art cGANs on benchmark datasets, especially on the most challenging ImageNet. Further investigation can be conducted 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 investigation on ImageNet. Ideally, more experiments and analysis on ImageNet can further strengthen the contribution. But training with such a large dataset is barely affordable for our computational resource, and we can only resort to the conclusive findings in the current results. 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 content, 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.

Acknowledgement

We thank the anonymous reviewers for valuable suggestions. This work is partially supported by the Ministry of Science and Technology of Taiwan via the grants MOST 107-2628-E-002-008-MY3 and 110-2628-E-002-013. We also thank the National Center for High-performance Computing (NCHC) of National Applied Research Laboratories (NARLabs) in Taiwan for providing computational resources.

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. 10

[2] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *CoRR*, abs/1701.07875, 2017. 1, 2, 5, 15

[3] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *ICLR*, 2019. 1, 6, 7, 8, 9

[4] Grigorios G. Chrysos, Jean Kossaifi, and Stefanos Zafeiriou. Rocgan: Robust conditional GAN. *Int. J. Comput. Vis.*, 128(10):2665–2683, 2020. 2

[5] Bo Dai, Zhen Liu, Hanjun Dai, Niao He, Arthur Gretton, Le Song, and Dale Schuurmans. Exponential family estimation via adversarial dynamics embedding. In *NeurIPS*, 2019. 10

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 2, 7

[7] Emily L. Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. In *NeurIPS*, 2015. 9

[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 *ICLR*, 2018. 2

[9] Zhe Gan, Liqun Chen, Weiyao Wang, Yunchen Pu, Yizhe Zhang, Hao Liu, Chunyuan Li, and Lawrence Carin. Triangle generative adversarial networks. In *NeurIPS*, 2017. 9

[10] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *CoRR*, abs/1406.2661, 2014. 1, 2

[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. 5, 10

[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 *ICLR*, 2020. 10

[13] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. In *NeurIPS*, 2017. 1, 8

[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 *NeurIPS*, 2017. 7

[15] Mingqi Hu, Deyu Zhou, and Yulan He. Variational conditional GAN for fine-grained controllable image generation. In *ACML*, 2019. 2

[16] Minguk Kang and Jaesik Park. Contragan: Contrastive learning for conditional image generation. In *NeurIPS*, 2020. 1, 2, 4, 6, 7, 8, 9, 15, 16

[17] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *ICLR*, 2018. 16

[18] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. In *ICML*, 2017. 1

[19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 10, 16

[20] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009. 7

[21] Rithesh Kumar, Anirudh Goyal, Aaron C. Courville, and Yoshua Bengio. Maximum entropy generators for energy-based models. *CoRR*, abs/1901.08508, 2019. 4

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

[23] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. 2006. 3

[24] Chongxuan Li, Taufik Xu, Jun Zhu, and Bo Zhang. Triple generative adversarial nets. In *NeurIPS*, 2017. 9

[25] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. MMD GAN: towards deeper understanding of moment matching network. In *NeurIPS*, 2017. 1

[26] Jae Hyun Lim and Jong Chul Ye. Geometric GAN. *CoRR*, abs/1705.02894, 2017. 1, 5, 15, 16

[27] Jae Hyun Lim, Aaron C. Courville, Christopher J. Pal, and Chin-Wei Huang. AR-DAE: towards unbiased neural entropy gradient estimation. In *ICML*, 2020. 4

[28] Hao Liu and Pieter Abbeel. Hybrid discriminative-generative training via contrastive learning. *CoRR*, abs/2007.09070, 2020. 5

[29] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *ICCV*, 2017. 1, 2

[30] Andreas Maurer, Massimiliano Pontil, and Bernardino Romera-Paredes. The benefit of multitask representation learning. *JMLR*, 17:81:1–81:32, 2016. 5

[31] Lars M. Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *ICML*, 2018. 16

[32] Daniel Michelsanti and Zheng-Hua Tan. Conditional generative adversarial networks for speech enhancement and noise-robust speaker verification. In *Interspeech*, 2017. 1

[33] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014. 1, 9

[34] Takeru Miyato and Masanori Koyama. cgans with projection discriminator. In *ICLR*, 2018. 1, 2, 6, 8, 9

[35] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018. 1, 6, 16

[36] Youssef Mroueh and Tom Sercu. Fisher gan. In *NeurIPS*, 2017. 1

[37] Youssef Mroueh, Tom Sercu, and Vaibhava Goel. Mcgan: Mean and covariance feature matching gan. In *ICML*, 2017. 1

[38] Youssef Mroueh, Chun-Liang Li, Tom Sercu, Anant Raj, and Yu Cheng. Sobolev gan. In *ICLR*, 2018. 1

[39] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In Doina Precup and Yee Whye Teh, editors, *ICML*, 2017. 1, 2, 6, 8, 9

[40] 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, *ICML*, 2019. 4, 15

[41] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016. 1, 8

[42] Rajesh Ranganath, Dustin Tran, and David M. Blei. Hierarchical variational models. In *ICML*, 2016. 4

[43] Scott E. Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *ICML*, 2016. 1

[44] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NeurIPS*, 2016. 7

[45] Rui Shu. Ac-gan learns a biased distribution. 2017. 2, 9

[46] Shashank Singh and Barnabás Póczos. Analysis of k-nearest neighbor distances with application to entropy estimation. *CoRR*, abs/1603.08578, 2016. 4

[47] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018. 15

[48] Martin J. Wainwright and Michael I. Jordan. Graphical models, exponential families, and variational inference. *Found. Trends Mach. Learn.*, 1(1-2):1–305, 2008. 3

[49] Yasin Yazici, Chuan-Sheng Foo, Stefan Winkler, Kim-Hui Yap, Georgios Piliouras, and Vijay Chandrasekhar. The unusual effectiveness of averaging in GAN training. In *ICLR*, 2019. 16

[50] Han Zhang, Ian J. Goodfellow, Dimitris N. Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *ICML*, 2019. 1, 6, 16

[51] Han Zhang, Zizhao Zhang, Augustus Odena, and Honglak Lee. Consistency regularization for generative adversarial networks. In *ICLR*, 2020. 1

[52] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient GAN training. In *NeurIPS*, 2020. 16

[53] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017. 1

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, Section 4.1, Appendix D
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Table 3, Table 4, Table 5
   (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 4.1
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, Section 4.1
   (b) Did you mention the license of the assets? [Yes] Section 4, Section 4.1
   (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, Section 4.1. 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
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]