Triple Generative Adversarial Nets

Chongxuan Li LICX14@MAILS.TSINGHUA.EDU.CN
Kun Xu XU-K16@MAILS.TSINGHUA.EDU.CN
Jun Zhu DCSZJ@MAILS.TSINGHUA.EDU.CN
Bo Zhang DCSZB@MAILS.TSINGHUA.EDU.CN
Tsinghua University, China

Abstract

Generative adversarial nets (GANs) are good at generating realistic images and have been extended for semi-supervised classification. However, under a two-player formulation, existing work shares competing roles of identifying fake samples and predicting labels via a single discriminator network, which can lead to undesirable incompatibility. We present triple generative adversarial net (Triple-GAN), a flexible game-theoretical framework for classification and class-conditional generation in semi-supervised learning. Triple-GAN consists of three players—a generator, a discriminator and a classifier, where the generator and classifier characterize the conditional distributions between images and labels, and the discriminator solely focuses on identifying fake image-label pairs. With designed utilities, the distributions characterized by the classifier and generator both concentrate to the data distribution under nonparametric assumptions. We further propose unbiased regularization terms to make the classifier and generator strongly coupled and some biased techniques to boost the performance of Triple-GAN in practice. Our results on several datasets demonstrate the promise in semi-supervised learning, where Triple-GAN achieves comparable or superior performance than state-of-the-art classification results among DGMs; it is also able to disentangle the classes and styles and transfer smoothly on the data level via interpolation on the latent space class-conditionally.

1. Introduction

Deep generative models (DGMs) can capture the underlying distributions of the input data and synthesize new samples. Recently, significant progress has been made on generating realistic images based on Generative Adversarial Nets (GANs) (Goodfellow et al., 2014; Denton et al., 2015; Radford et al., 2015). GAN is formulated as a two-player game, where the generator G takes a random noise z as input and produces a sample $G(z)$ in the data space while the discriminator D identifies whether a certain sample comes from the true data distribution $p(x)$ or the generator. Both G and D are parameterized as deep neural networks and the training procedure is to solve a minimax problem:

$$\min_G \max_D U(D,G) = E_{x\sim p(x)}[\log(D(x))] + E_{z\sim p_g(z)}[\log(1-D(G(z)))],$$

where $p_g(z)$ is a simple distribution (e.g., uniform or normal) and $U(\cdot)$ denotes the utilities. Given a generator and the defined distribution p_g, the optimal discriminator is $D(x) = \frac{p(x)}{p_g(x)+p(x)}$ under the assumption of infinite capacity, and the global equilibrium of this game achieves if and only if $p_g(x) = p(x)$ (Goodfellow et al., 2014), which is desired in terms of image generation.

GANs and DGMs in general have also proven effective in semi-supervised learning (Kingma et al., 2014), while retaining the generative capability. Under the same two-player game framework, Cat-GAN (Springenberg, 2015) generalizes GANs with a categorical discriminative network and an objective function that minimizes the conditional entropy of predictions given data while maximizes the conditional entropy of predictions given generated samples. Odena (2016) and Salimans et al. (2016) augment the categorical discriminator with one more class, corresponding to the fake data generated by the generator. Salimans et al. (2016) further propose two alternative training objectives that work well for either semi-supervised classification or image generation, but not both. Specifically, the objective of *feature matching* works well in semi-supervised

classification but fails to generate indistinguishable samples (See Sec.4.2 for examples), while the other objective of *minibatch discrimination* is good at realistic image generation but cannot predict labels accurately.

This incompatibility essentially arises from their two-player formulation, where a single discriminator network has to play two competing roles—identifying fake samples and predicting labels. Such a shared architecture can prevent the model from achieving a desirable equilibrium. Assume that both G and D converge finally and G concentrates to the true data distribution, namely $p(x) = p_g(x)$. Given a generated sample, as a discriminator, D should identify it as fake data with probability $\frac{1}{2}$; while as a classifier, D should also predict it as the correct class confidently. Obviously, the two roles conflict, hence either G or D could not achieve its optimum. In addition, disentangling meaningful physical factors like object category from latent representations with limited supervision is of general interest (Yang et al., 2015). However, existing semi-supervised GANs focus on the marginal distribution of data and hence G cannot generate data in a specific class.

We address these issues by presenting Triple-GAN, a flexible game-theoretical framework for both classification and class-conditional image generation in semi-supervised learning, where the data is characterized by a joint distribution $p(x, y)$ of input x and label y. Based on the alternative factorizations $p(x, y) = p(x)p(y|x)$ and $p(x, y) = p(y)p(x|y)$, we build two separate conditional models–a generator and a classifier, to characterize the conditional distributions $p(x|y)$ and $p(y|x)$, respectively. By mildly assuming that samples can be drawn from the marginal distributions $p(x)$ and $p(y)$, we can draw input-label pairs (x, y) from our generator and classifier. Then, we define an adversarial game by introducing a discriminator which has the single role of determining whether a sample (x, y) is from the model distribution or the data distribution. Consequently, we naturally arrive at a game with tripartite correlations of a generator, a classifier and a discriminator. In Triple-GAN, the shared discriminator enforces the mixture distributions defined by the classifier and generator to be similar to the true data distribution, and the equilibrium is unique under a proper regularizer with a supervised loss and a nonparametric assumption (See Sec. 3.2). Therefore, a good classifier will result in a good generator via teaching the common discriminator and vice versa. We further introduce some unbiased regularization terms and biased techniques to boost the performance of Triple-GAN. Especially, we propose a *pseudo discriminative loss*, which enforces the classifier to predict the generated images correctly, to improve the classifier via the generative model explicitly (See details in Sec. 3.3).

Our results on the widely adopted MNIST (LeCun et al., 1998), SVHN (Netzer et al., 2011) and CIFAR10 (Krizhevsky & Hinton, 2009) datasets demonstrate that Triple-GAN can make accurate predictions without sacrificing generation quality. We advance the state-of-the-art semi-supervised classification results on MNIST and SVHN among DGMs. We further show that Triple-GAN is able to disentangle classes and styles and perform class-conditional interpolation given partially labeled data.

2. Related Work

Various approaches have been developed to learn DGMs, including MLE-based models such as Variational Autoencoders (VAEs) (Kingma & Welling, 2013; Rezende et al., 2014), Generative Moment Matching Networks (GMMNs) (Li et al., 2015; Dziugaite et al., 2015) and Generative Adversarial Nets (GANs) (Goodfellow et al., 2014), which can be viewed as an instance of Noise Contrastive Estimation (NCE) (Gutmann & Hyvärinen, 2010). These criteria are systematically compared in (Theis et al., 2015).

One primal goal of DGMs is to generate realistic samples, for which GANs have proven effective. Specifically, LAPGAN (Denton et al., 2015) leverages a series of GANs to upscale the generated samples to high resolution images through the Laplacian pyramid framework (Burt & Adelson, 1983). DCGAN (Radford et al., 2015) adopts (fractionally) strided convolution networks and batch normalization (Ioffe & Szegedy, 2015) in GANs and generate realistic natural images. The architecture of DCGAN is widely adopted in various GANs, including ours.

Recent work has introduced inference networks in GANs, which can infer latent variables given data and train the whole model jointly in an adversarial process. For instance, InfoGAN (Chen et al., 2016) learns interpretable latent codes from unlabeled data by regularizing the original GANs via variational mutual information maximization. In ALI (Dumoulin et al., 2016; Donahue et al., 2016), the inference network infers the latent variables from true data and the discriminator estimates the probability that a pair of latent variable and data comes from the inference network instead of the generator, which make the joint distributions defined by the generator and inference network to be same. We draw inspiration from the architecture of ALI but there exist two main differences in global equilibria and utilities: (1) Triple-GAN focuses on learning good generator and classifier given partially labeled data, while ALI aims to learn latent features from unlabeled data; (2) both the classifier (inference network) and generator attempt to fool the discriminator in Triple-GAN, while the discriminator would like to accept the samples from the inference network but reject samples from the generator in ALI.

To handle partially labeled data, the class-conditional VAE (Kingma et al., 2014) treats the missing labels

as latent variables and infer them for unlabeled data. ADGM (Maaløe et al., 2016) introduces auxiliary variables to build a more expressive variational distribution and improve the predictive performance. The Ladder Network (Rasmus et al., 2015) employs lateral connections between a variation of denoising autoencoders and obtains excellent semi-supervised classification results. CatGAN (Springenberg, 2015) generalizes GANs with a categorical discriminative network and an objective function. Salimans et al. (2016) propose empirical techniques to stabilize the training of GANs and improve the performance on semi-supervised learning and image generation under incompatible learning criteria. Triple-GAN differs significantly from these methods, as stated in the introduction.

3. Method

We consider learning deep generative models in the semi-supervised setting,[1] where we have a partially labeled dataset with x denoting input and y denoting the output label. The goal is to predict the labels y for unlabeled data as well as to generate new samples x. This is different from the unsupervised learning setting for pure generation, where the primary goal is to identify whether a sample x is generated from the true distribution $p(x)$ of input or not; thus a two-player game with an input generator and a discriminator is sufficient to describe the process as in GANs. In our setting, as the label information y is incomplete (thus uncertain), our density model should characterize the uncertainty of both x and y, therefore a joint distribution $p(x, y)$ of input-label pairs.

A straightforward application of the two-player GAN is infeasible because of the missing values on y. Unlike the previous work on semi-supervised GANs (Springenberg, 2015; Salimans et al., 2016), which is restricted to the two-player framework and can lead to incompatible objectives, we build our game-theoretic objective based on the insight that the joint distribution can be factorized in two ways, namely, $p(x, y) = p(x)p(y|x)$ and $p(x, y) = p(y)p(x|y)$, and that the conditional distributions $p(y|x)$ and $p(x|y)$ are of interest for classification and class-conditional generation, respectively. To jointly estimate these conditional distributions, which are characterized by a class-conditional generator network and a classifier network, we define a single discriminator network which has the sole role of distinguishing whether a sample is from the true data distribution or the models. Hence, we naturally extend GANs to Triple-GAN, a three-player game to characterize the process of semi-supervised classification and class-conditional generation, as detailed below.

[1] Supervised learning is an extreme case, where the training set is fully labeled while the testing set is unlabeled.

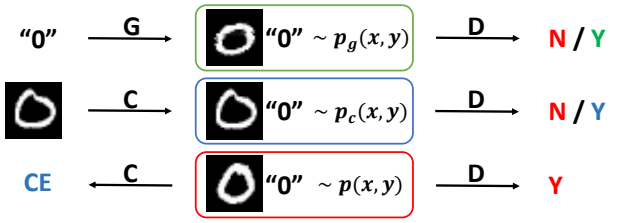


Figure 1. An illustration of Triple-GAN (best view in color). The objectives of C, G and D are colored in blue, green and red respectively, with "N" denoting rejection, "Y" denoting acceptance and "CE" denoting the cross entropy loss for supervised learning.

3.1. A Game with Three Players

Triple-GAN consists of three components: (1) a classifier C that (approximately) characterizes the conditional discriminative distribution $p_c(y|x) \approx p(y|x)$; (2) a class-conditional generator G that (approximately) characterizes the conditional distribution in the other direction $p_g(x|y) \approx p(x|y)$; and (3) a discriminator D that distinguishes whether a pair of data (x, y) comes from the true distribution $p(x, y)$. All the components are parameterized as neural networks. Our desired equilibrium is that the joint distributions defined by the classifier and generator will concentrate to the true data distributions. To this end, we design a game with compatible utilities for triple players as follows.

We make the mild assumption that the samples from both $p(x)$ and $p(y)$ can be easily obtained.[2] In the game, after a sample x is drawn from $p(x)$, the classifier C produces a pseudo label y given x following the conditional distribution $p_c(y|x)$. Hence, the pseudo input-label pair is a sample from the joint distribution $p_c(x, y) = p(x)p_c(y|x)$. Similarly, a pseudo input-label pair can be sampled from the generator G by first drawing $y \sim p(y)$ and then drawing $x|y \sim p_g(x|y)$; hence from the joint distribution $p_g(x, y) = p(y)p_g(x|y)$. For $p_g(x|y)$, we assume that x is transformed by the latent style variables z given the label y, namely, $x = G(y, z), z \sim p_g(z)$, where $p_g(z)$ is a simple distribution (e.g., uniform or standard normal). Then, the pseudo input-label pairs (x, y) generated by both C and G are sent to the single discriminator D for judgement. The discriminator D can also access the input-label pairs from the true data distribution as positive samples. The utilities can be formulated as a minimax game:

$$\min_{C,G} \max_{D} U(C, G, D) = E_{(x,y)\sim p(x,y)}[\log D(x, y)] + (1-\alpha)E_{y\sim p(y), z\sim p_g(z)}[\log(1 - D(G(y, z), y))] + \alpha E_{x\sim p(x), y\sim p_c(y|x)}[\log(1 - D(x, y))], \quad (1)$$

[2] In semi-supervised learning, $p(x)$ is the empirical distribution of inputs and $p(y)$ is assumed same to the distribution of labels on labeled data, which is uniform in our experiment.

where $\alpha \in (0,1)$ is a constant that controls the relative importance of generation and classification. Note that $p_c(y|x)$ should be a deterministic mapping (or delta distribution) to allow the training signal from D back-propagated to C and we simply choose the label with the maximum probability instead of sampling one in our experiment.

The minimax game defined in Eqn. (1) achieves the equilibrium if and only if $p(x,y) = (1-\alpha)p_g(x,y)+\alpha p_c(x,y)$ (See details in Sec. 3.2). The equilibrium indicates that if one of C and G tends to the data distribution, the other will become better. However, unfortunately, it cannot guarantee that $p(x,y) = p_g(x,y) = p_c(x,y)$ is the unique global optimum, which is not desirable. To address this problem, we add the standard supervised loss (i.e., cross-entropy loss) for the classifier C, $\mathcal{R}_{\mathcal{L}} = E_{(x,y)\sim p(x,y)}[-\log p_c(y|x)]$, which is equivalent to the KL-divergence between $p_c(x,y)$ and $p(x,y)$. Consequently, we define the game as:

$$\begin{aligned}\min_{C,G}\max_{D} \tilde{U}(C,G,D) &= E_{(x,y)\sim p(x,y)}[\log D(x,y)] \\ +(1-\alpha)&E_{y\sim p(y),z\sim p_g(z)}[\log(1-D(G(y,z),y))] \\ +\alpha&E_{x\sim p(x),y\sim p_c(y|x)}[\log(1-D(x,y))] + \mathcal{R}_{\mathcal{L}}. \quad (2)\end{aligned}$$

It will be proven that the game trained under $\tilde{U}$ has the unique global optimum for C and G. See Fig. 1 for an illustration of the whole model.

3.2. Theoretical Analysis

We now provide a formal theoretical analysis of Triple-GAN under nonparametric assumptions. For clarity of the main text, we defer the proof details to Appendix A.

First, we can show that the optimal D balances between the true data distribution and the mixture distribution defined by C and G, as summarized in Lemma 3.1.

Lemma 3.1. *For any fixed C and G, the optimal discriminator D of the game defined by the utility function $U(C,G,D)$ is:*

$$D^*_{C,G}(x,y) = \frac{p(x,y)}{p(x,y)+p_\alpha(x,y)}, \quad (3)$$

where $p_\alpha(x,y) := (1-\alpha)p_g(x,y) + \alpha p_c(x,y)$ is a valid distribution.

Given $D^*_{C,G}$, we can omit the discriminator and reformulate the minimax game with value function U as:

$$V(C,G) = \max_D U(C,G,D),$$

whose optimum is summarized as in Lemma 3.2.

Lemma 3.2. *The global minimum value of $V(C,G)$ is $-\log 4$ and it is achieved if and only if $p(x,y) = p_\alpha(x,y)$.*

We can further show that C and G can at least capture the marginal distribution of data, especially for $p_g(x)$, even there may exist multiple global equilibria, as summarized in Corollary A.

Corollary 3.2.1. *Given $p(x,y) = p_\alpha(x,y)$, the marginal distributions are the same for any pairs of p, p_c and p_g.*

Given the above result that $p(x,y) = p_\alpha(x,y)$, C and G do not compete explicitly as in the two-player based formulation and it is easy to verify that $p(x,y) = p_c(x,y) = p_g(x,y)$ is a global equilibrium point. However, it may not be unique and we should minimize an additional objective to ensure the uniqueness. In fact, this is true for the utility function $\tilde{U}(C,G,D)$ in problem (2), as stated below.

Theorem 3.3. *The equilibrium of $\tilde{U}(C,G,D)$ is achieved if and only if $p(x,y) = p_g(x,y) = p_c(x,y)$ with $D^*_{C,G}(x,y) = \frac{1}{2}$ and the optimum value is $-\log 4$.*

The conclusion essentially motivates our design of Triple-GAN, as we can ensure that both C and G will concentrate to the true data distribution if the model has been trained to achieve the optimum.

We can further show another nice property of $\tilde{U}$, which allows us to regularize our model for stable and better convergence in practice without bias, as summarized below.

Corollary 3.3.1. *Any additional regularization on the distances between the marginal, conditional and joint distributions of any two players, will not change the global equilibrium of $\tilde{U}$.*

3.3. Unbiased Regularization

Despite the discriminative loss for the classifier, we add other unbiased regularization terms to boost the performance of our algorithm, following Corollary A.

Pseudo discriminative loss We treat the samples generated by the generator as labeled data and optimize the classifier on the pseudo data to make the generator and classifier strongly coupled. Intuitively, a good generator can provide meaningful labeled data beyond the training set as extra side information for the classifier, which will boost the predictive performance. We refer to this regularization as *Pseudo discriminative loss* and it is in the form:

$$\mathcal{R}_{\mathcal{P}} = E_{p_g}[-\log p_c(y|x)],$$

and the right hand side can be rewritten as:

$$D_{KL}(p_g(x,y)||p_c(x,y)) + H_{p_g}(y|x) - D_{KL}(p_g(x)||p(x)),$$

where $H_{p_g}(y|x)$ denotes the conditional entropy over p_g (See Appendix A for proof). Note that we do not train the generator to minimize this loss and the last two terms can be viewed as constants with respect to C. Therefore, minimizing $\mathcal{R}_{\mathcal{P}}$ is equivalent to minimizing the KL-divergence between $p_g(x,y)$ and $p_c(x,y)$; hence Corollary A applies to ensure that the global equilibrium is unchanged.

Algorithm 1 Minibatch stochastic gradient descent training of Triple-GAN in semi-supervised learning.

for number of training iterations **do**

- Sample a batch of labels $y_g \sim p(y)$ and a batch of noise $z_g \sim p_g(z)$ of size m_g, and generate pseudo data $x_g \sim p_g(x|y) = G(z_g, y_g)$.
- Sample a batch of unlabeled data $x_c \sim p(x)$ of size m_c, and compute pseudo labels $y_c = \arg\max_y p_c(y|x)$ for x_c.
- Sample a batch of labeled data $(x_l, y_l) \sim p(x, y)$ and a batch of unlabeled data $x_d \sim p(x)$.
- Get pseudo labels $y_d = \arg\max_y p_c(y|x)$ for x_d and concatenate (x_l, y_l) and (x_d, y_d) together as a batch of size m_d.
- Update D by ascending along its stochastic gradient:

$$\nabla_{\theta_d}\left[\frac{1}{m_d}\Big(\sum_{(x_l,y_l)}\log D(x_l,y_l)+\sum_{(x_d,y_d)}\log D(x_d,y_d)\Big)+\frac{\alpha}{m_c}\sum_{(x_c,y_c)}\log(1-D(x_c,y_c))+\frac{1-\alpha}{m_g}\sum_{(x_g,y_g)}\log(1-D(x_g,y_g))\right].$$

- Compute the unbiased estimators $\tilde{\mathcal{R}}_{\mathcal{L}}$, $\tilde{\mathcal{R}}_{\mathcal{P}}$ and $\tilde{\mathcal{R}}_{\mathcal{U}}$ of the corresponding losses $\mathcal{R}_{\mathcal{L}}$, $\mathcal{R}_{\mathcal{P}}$ and $\mathcal{R}_{\mathcal{U}}$ respectively.
- Update C by descending along its stochastic gradient:

$$\nabla_{\theta_c}\left[\frac{\alpha}{m_c}\sum_{(x_c,y_c)}\log(1-D(x_c,y_c))+\tilde{\mathcal{R}}_{\mathcal{L}}+\alpha_{\mathcal{P}}\tilde{\mathcal{R}}_{\mathcal{P}}+\alpha_{\mathcal{U}}\tilde{\mathcal{R}}_{\mathcal{U}}\right].$$

- Update G by descending along its stochastic gradient:

$$\nabla_{\theta_g}\left[\frac{1-\alpha}{m_g}\sum_{(x_g,y_g)}\log(1-D(x_g,y_g))\right].$$

end for

Regularization on the generator It is natural to regularize the generator because it will in turn benefit the classifier. In our early experiment, we tried the maximum mean discrepancy loss (Gretton et al., 2012) between the marginal distributions $p_g(x)$ and $p(x)$. However, we did not observe significant improvement and hence omited it for efficiency. Corollary A may explain this as Triple-GAN ensures that the players share same marginal distributions.

3.4. Practical Techniques

In this subsection we introduce several practical techniques for Triple-GAN, which may lead to a biased solution theoretically but work well in practice.

One crucial problem of Triple-GAN in semi-supervised learning is that the discriminator may collapse to the delta distribution on the labeled data of small size, and reject other types of samples, which may come from the true data distribution indeed. Consequently, the generator would also concentrate to the empirical distribution and generate certain labeled data no matter what the latent variable is. To address this problem, we sample some unlabeled data and generate pseudo labels through the classifier and use these data as true labeled data for the training of the discriminator. Note that we just use $p_c(y|x)$ to approximate $p(y|x)$ but do not train C to optimize this term. This approach introduces slight bias to Triple-GAN because the target distribution shifts a little towards $p_c(x, y)$. However, it remains that a good classifier will result in a good generator and this technique works well in practice.

As only C can leverage the unlabeled data directly, the performance of the whole system highly relies on the goodness of C. Consequently, it is necessary to regularize C heuristically as in recent advances (Springenberg, 2015; Laine & Aila, 2016) to make more accurate predictions. Note that the separated pathway of C and D in Triple-GAN provides extra freedom to regularize C biased or not without any negative effect on D intuitively. We consider two alternative losses on the unlabeled data as follows.

Confidence and balance loss Springenberg (2015) minimizes the conditional entropy of $p_c(y|x)$ and the cross entropy between $p(y)$ and $p_c(y)$, weighted by a hyperparameter $\alpha_{\mathcal{B}}$, as follows:

$$\mathcal{R}_{\mathcal{U}} = H_{p_c}(y|x) + \alpha_{\mathcal{B}} E_p\big[-\log p_c(y)\big],$$

which encourages the classifier to make predictions confidently and be balanced on unlabeled data. The similar idea has been proven effective in (Li et al., 2016) with a large margin classifier.

Consistency loss Laine & Aila (2016) penalize the network if it predicts the same unlabeled data inconsistently given different noise ϵ, e.g., dropout masks, as follows:

$$\mathcal{R}_{\mathcal{U}'} = E_{x\sim p(x)}||p_c(y|x,\epsilon) - p_c(y|x,\epsilon')||^2,$$

where $||\cdot||^2$ is the square of the l_2-norm, which is chosen for its smoothness but can also be replaced with other choices.

We use the confidence and balance loss by default except on the CIFAR10 dataset because the consistency loss works better for complicated data. The whole training procedure of Triple-GAN is presented in Alg. 1.

4. Experiments

We now present results on the widely adopted MNIST (LeCun et al., 1998), SVHN (Netzer et al., 2011), and CIFAR10 (Krizhevsky & Hinton, 2009) datasets. MNIST consists of 60,000 training samples and 10,000 testing samples of handwritten digits of size 28×28 pixels. SVHN consists of 73,257 training samples and 26,032 testing samples and each is a colored image of size 32×32, containing a sequence of digits with various backgrounds. CIFAR10 consists of colored images distributed across 10 general classes—*airplane*, *automobile*, *bird*, *cat*, *deer*, *dog*, *frog*, *horse*, *ship* and *truck*. There are 50,000 training samples and 10,000 samples of size 32×32 in CIFAR10. We follow (Salimans et al., 2016) to rescale the pixels of SVHN and CIFAR10 data into $(-1, 1)$. The labeled data is distributed equally across classes and the results are averaged over 10 times with different random splits of training data, following (Springenberg, 2015; Salimans et al., 2016).

We implement our method on Theano (Theano Development Team, 2016) and here we briefly summarize our experimental settings.[3] Our network architectures are highly referred to existing work on GANs (Radford et al., 2015; Springenberg, 2015; Salimans et al., 2016) and the details are listed in Appendix E. We train G to maximize $\log D(G(y, z), y)$ instead of minimizing $\log(1 - D(G(y, z), y))$ as suggested in (Goodfellow et al., 2014; Radford et al., 2015). We optimize all of our networks with Adam (Kingma & Ba, 2014) where the first order of momentum is 0.5 and others are default values. The model is trained for 1000 epochs with a global learning rate in $\{0.0003, 0.001\}$, which is annealed by a factor of 0.995 after 300 epochs. Each batch of data for C contains an equal number of labeled and unlabeled data for fast convergence. However, for the training of D, the unlabeled data (with labels generated by the classifier) is more than the labeled data in a batch to avoid collapsing and the ratio between them is nearly proportional to that of the whole dataset.

For the hyperparameters in the objectives, we simply set $\alpha = 1/2$, which means that D treats C and G equally. We refer (Li et al., 2016) to set $\alpha_{\mathcal{B}} = 1/300$ and $\alpha_{\mathcal{U}} = 0.3$ and fix them across all experiments if not mentioned. The pseudo discriminative loss is not applied until a threshold that the generator could generate meaningful data. The threshold is chosen from $\{200, 300\}$ and $\alpha_{\mathcal{P}}$ is chosen from $\{0.1, 0.03\}$, depending on the data. We keep the moving average of the parameters in the classifier for stable evaluation as in (Salimans et al., 2016). In our experiments, we find that these training techniques for the original two-player GANs are sufficient to stabilize the optimization of Triple-GAN and the convergence speed is comparable to previous work (Salimans et al., 2016).

[3] We will release our source code recently.

4.1. Classification

We first evaluate our method with 20, 50, 100 and 200 labeled samples on MNIST for a systematical comparison with previous methods. We pretrain C solely with $\alpha_{\mathcal{B}} = 1$ for 30 epochs to reduce the variance given no more than 100 labels. Table 1 summarizes the quantitative results. Given 100 labels, our method is competitive to the state-of-the-art among a large body of approaches. Under other settings, Triple-GAN consistently outperforms Improved-GAN, as shown in the last two rows. Besides, we can see that Triple-GAN achieves more significant improvement as the number of labeled data decreases, suggesting the effectiveness of the pseudo discriminative loss. We also evaluate our Triple-GAN in supervised learning, where we omit all regularization terms and techniques except the pseudo discriminative loss. The result again confirms that the compatible utilities help Triple-GAN converge better than two-player GANs, as shown in the last column in Table 1.

Table 2 presents the results on SVHN with 1,000 labels. For a fair comparison with previous GANs, we do not leverage the extra unlabeled data while some baselines (e.g., ADGM, SDGM and MMCVA) do. In this setting, Triple-GAN substantially outperforms the state-of-the-art methods (e.g., ALI and Improved-GAN).

Table 3 compares the Triple-GAN with baselines on the CIFAR10 dataset with 4,000 labels. We perform ZCA whitening on the data for C following (Laine & Aila, 2016) but still generate and estimate the raw images using G and D. We found that the consistency loss used works better for Triple-GAN and we implement a simple version of the Π model in (Laine & Aila, 2016) with the learning rate annealing strategy mentioned in the general setting as the baseline, which achieves an error rate of 19.2%. The small margin between Triple-GAN and the baseline may indicate that the generator on CIFAR10 is far from optimal and cannot help the classifier as much as it does on MNIST and SVHN. Nevertheless, Triple-GAN is still competitive to the state-of-the-art DGMs.

4.2. Generation

We demonstrate that Triple-GAN tends to achieve the desirable equilibrium by generating samples in various ways with the exact models used in the semi-supervised classification. The generative model and the number of labels are the same to the previous method (Salimans et al., 2016).

Table 1. Error rates (%) on the (partially) labeled MNIST dataset.

Algorithm	$n = 20$	$n = 50$	$n = 100$	$n = 200$	All
M1+M2 (Kingma et al., 2014)			3.33 ($\pm$0.14)		0.96
VAT (Miyato et al., 2015)			2.33		0.64
Ladder (Rasmus et al., 2015)			1.06 ($\pm$0.37)		0.57
Conv-Ladder (Rasmus et al., 2015)			**0.89** ($\pm$0.50)		
ADGM (Maaløe et al., 2016)			0.96 ($\pm$0.02)		
SDGM (Maaløe et al., 2016)			1.32 ($\pm$0.07)		
MMCVA (Li et al., 2016)			1.24 ($\pm$0.54)		**0.31**
CatGAN (Springenberg, 2015)			1.39 ($\pm$0.28)		0.48
Improved-GAN (Salimans et al., 2016)	16.77 ($\pm$4.52)	2.21 ($\pm$1.36)	**0.93** ($\pm$0.07)	0.90 ($\pm$0.04)	
Triple-GAN (**ours**)	**5.40** ($\pm$6.53)	**1.59** ($\pm$0.69)	**0.92** ($\pm$0.58)	**0.66** ($\pm$0.16)	**0.32**

Table 2. Error rates (%) on the partially labeled SVHN dataset. The results with † are trained with more than 500,000 extra unlabeled data.

Algorithm	$n = 1000$
M1+M2 (Kingma et al., 2014)	36.02 ($\pm$0.10)
VAT (Miyato et al., 2015)	24.63
ADGM (Maaløe et al., 2016)	22.86 †
SDGM (Maaløe et al., 2016)	16.61($\pm$0.24)†
MMCVA (Li et al., 2016)	**4.95** ($\pm$0.18) †
Improved-GAN (Salimans et al., 2016)	8.11 ($\pm$1.3)
ALI (Dumoulin et al., 2016)	7.3
Triple-GAN (**ours**)	**5.83** ($\pm$0.20)

Table 3. Error rates (%) on the partially labeled CIFAR10 dataset.

Algorithm	$n = 4000$
Ladder (Rasmus et al., 2015)	20.40 ($\pm$0.47)
CatGAN (Springenberg, 2015)	19.58 ($\pm$0.58)
Improved-GAN (Salimans et al., 2016)	18.63 ($\pm$2.32)
ALI (Dumoulin et al., 2016)	**18.3**
Triple-GAN (**ours**)	18.82 ($\pm$0.32)

In Fig. 6, we first compare the quality of images generated by Triple-GAN on SVHN and the Improved-GAN with feature matching (Salimans et al., 2016)[4], which works well for semi-supervised classification. We can see that Triple-GAN outperforms the baseline by generating fewer meaningless samples and clearer digits. Furthermore, Triple-GAN avoids repeating to generate strange patterns as shown in the figure. The comparison on MNIST and CIFAR10 is presented in Appendix B. We also evaluate the samples on CIFAR10 quantitatively via inception score following (Salimans et al., 2016). The value of Triple-GAN is 5.08 ± 0.09 while that of the Improved-GAN trained without minibatch discrimination (Salimans et al., 2016) is 3.87 ± 0.03, which agrees with the visual comparison.

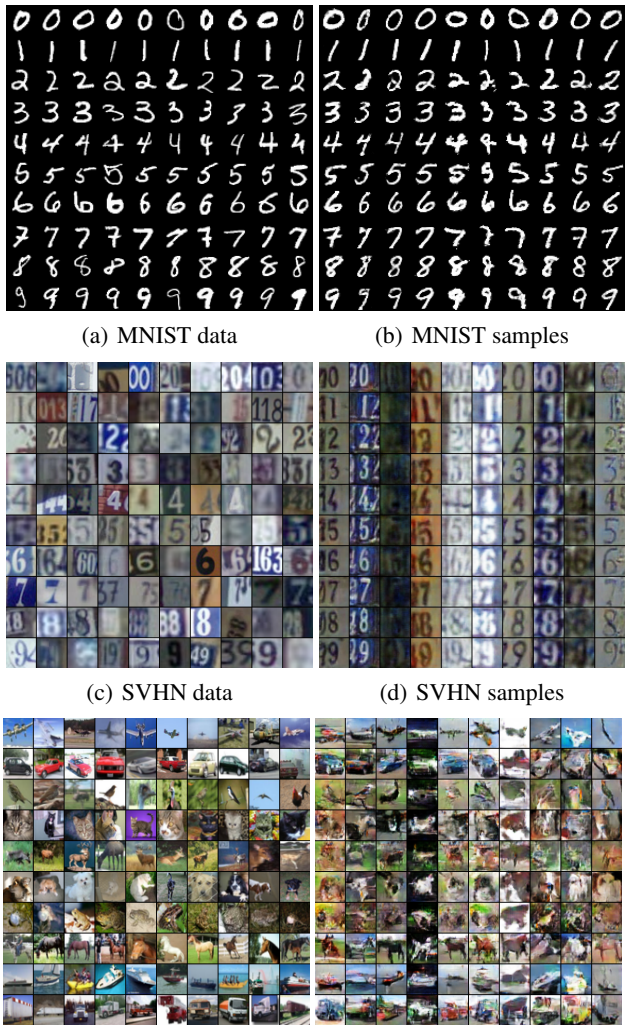
[4]Though the Improved-GAN trained with minibatch discrimination (Salimans et al., 2016) can generate good samples, it fails to predict labels accurately.

(a) Feature Matching (b) Triple-GAN

Figure 2. (a) Samples generated from Improved-GAN trained with feature matching. The strange pattern labeled with the red rectangle appears four times in the generation. (b) Samples generated from Triple-GAN. We randomly sample equally distributed labels and unshared latent variables for fair comparison in vision.

Then, we show the ability of Triple-GAN to disentangle classes and styles in Fig. 3. It can be seen that Triple-GAN can generate realistic data in a specific class and the latent factors encode meaningful physical factors like: scale, intensity, orientation, color and so on. DCGAN (Radford et al., 2015) and ALI (Dumoulin et al., 2016) can generate data class-conditionally given full labels, while Triple-GAN can do similar thing given incomplete label information. We illustrate images generated from four specific classes on CIFAR10 in Fig. 4 and see more in Appendix C. In most cases, Triple-GAN is able to generate meaningful images with correct semantics.

Finally, we demonstrate the generalization capability of our Triple-GAN on class-conditional latent space interpolation as in Fig. 5. Triple-GAN can transit smoothly from one sample to another with totally different visual factors without losing label semantics, which proves that Triple-GANs can learn meaningful latent spaces class-conditionally instead of overfitting to the training data, especially labeled



(a) MNIST data (b) MNIST samples

(c) SVHN data (d) SVHN samples

(e) CIFAR10 data (f) CIFAR10 samples

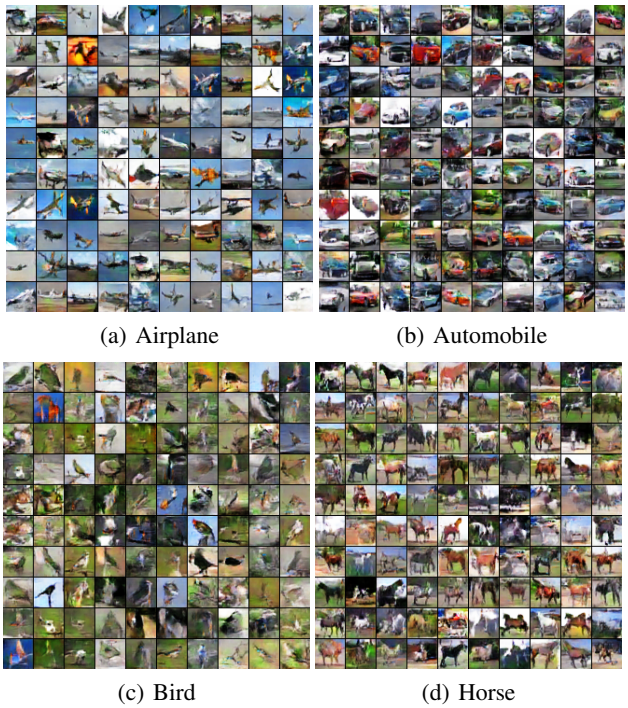
Figure 3. (a), (c) and (e) are randomly selected labeled data. (b), (d) and (f) are samples from Triple-GAN, where each row shares the same label and each column shares the same latent variables.

data (See results on MNIST in Appendix D).

Overall, these results confirm that Triple-GAN avoid the competition between C and G and can lead to a situation where both the generation and classification are good in semi-supervised learning.

5. Conclusions

We present triple generative adversarial networks (Triple-GAN), a unified game-theoretical framework with three players—a generator, a discriminator and a classifier, to do semi-supervised learning with compatible utilities. The distributions characterized by the classifier and the class-conditional generator concentrate to the data distribution under nonparametric assumptions. The classifier benefits from the generator by the enforce of the discriminator implicitly and the pseudo discriminative loss explicitly, and the generator generates images class-conditionally in semi-supervised learning thanks to the classifier, which can infer the labels for unlabeled data. Our empirical results on several datasets demonstrate the promise—Triple-GAN achieves comparable or superior classification performance than state-of-the-art DGMs, while retaining the capability to generate meaningful images when the label information is incomplete. Moreover, Triple-GAN can disentangle styles and classes and transfer smoothly on the data level via interpolation on the latent space.



(a) Airplane (b) Automobile

(c) Bird (d) Horse

Figure 4. Samples from four specific classes on CIFAR10.

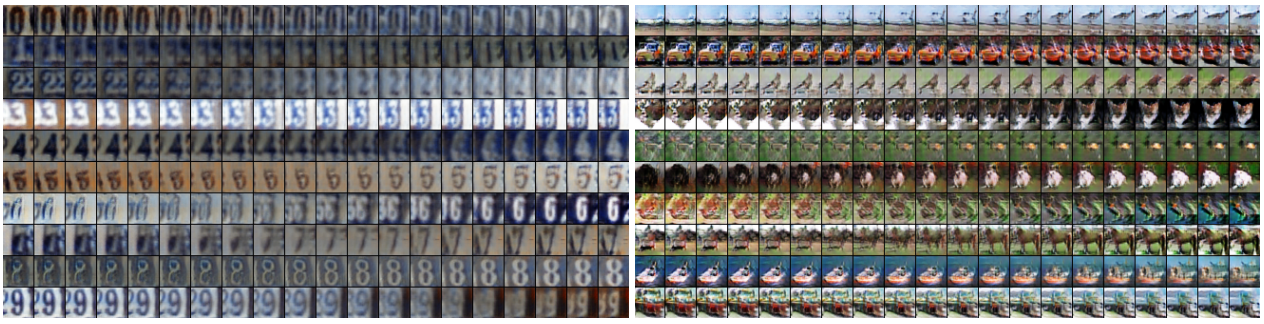
References

Burt, P. and Adelson, E. The Laplacian pyramid as a compact image code. *IEEE Transactions on communications*, 1983.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS*, 2016.

Denton, E. L., Chintala, S., and Fergus, R. Deep generative image models using a Laplacian pyramid of adversarial networks. In *NIPS*, 2015.

Donahue, J., Krähenbühl, P., and Darrell, T. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.



(a) SVHN (b) CIFAR10

Figure 5. Class-conditional latent space interpolation. We first sample two random vectors in the latent space and interpolate linearly from one to another. Then, we map these vectors to the data level given a fixed label for each class. Totally, 20 images are shown for each class. We select two endpoints with clear semantics on CIFAR10 for better illustration.

Dumoulin, V., Belghazi, I., Poole, B., Lamb, A., Arjovsky, M., Mastropietro, O., and Courville, A. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

Dziugaite, G. K., Roy, D. M., and Ghahramani, Z. Training generative neural networks via maximum mean discrepancy optimization. *arXiv preprint arXiv:1505.03906*, 2015.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *NIPS*, 2014.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. A kernel two-sample test. *JMLR*, 13 (Mar):723–773, 2012.

Gutmann, M. and Hyvärinen, A. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, 2010.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. Semi-supervised learning with deep generative models. In *NIPS*, 2014.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. *Citeseer*, 2009.

Laine, S. and Aila, T. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, C., Zhu, J., and Zhang, B. Max-margin deep generative models for (semi-) supervised learning. *arXiv preprint arXiv:1611.07119*, 2016.

Li, Y., Swersky, K., and Zemel, R. S. Generative moment matching networks. In *ICML*, 2015.

Maaløe, L., Sønderby, C. K., Sønderby, S. K., and Winther, O. Auxiliary deep generative models. *arXiv preprint arXiv:1602.05473*, 2016.

Miyato, T., Maeda, S.-i., Koyama, M., Nakae, K., and Ishii, S. Distributional smoothing with virtual adversarial training. *arXiv preprint arXiv:1507.00677*, 2015.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, 2011.

Odena, A. Semi-supervised learning with generative adversarial networks. *arXiv preprint arXiv:1606.01583*, 2016.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Rasmus, A., Berglund, M., Honkala, M., Valpola, H., and Raiko, T. Semi-supervised learning with ladder networks. In *NIPS*, 2015.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training GANs. In *NIPS*, 2016.

Springenberg, J. T. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*, 2015.

Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. URL http://arxiv.org/abs/1605.02688.

Theis, L., Oord, A. v. d., and Bethge, M. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

Yang, J., Reed, S. E., Yang, M.-H., and Lee, H. Weakly-supervised disentangling with recurrent transformations for 3d view synthesis. In *NIPS*, 2015.

A. Detailed Theoretical Analysis

Lemma 3.1. *For any fixed C and G, the optimal discriminator D of the game defined by the utility function $U(C,G,D)$ is*

$$D^*_{C,G}(x,y)=\frac{p(x,y)}{p(x,y)+p_\alpha(x,y)}, \tag{4}$$

where $p_\alpha(x,y):=(1-\alpha)p_g(x,y)+\alpha p_c(x,y)$ is a valid distribution .

Proof. Given the classifier and generator, the utility function can be rewritten as

$$\begin{aligned}U(C,G,D)&=\iint p(x,y)\log D(x,y)dydx\\ +(1-\alpha)&\iint p(y)p_g(z)\log(1-D(G(z,y),y))dydz\\ +\alpha&\iint p(x)p_c(y|x)\log(1-D(x,y))dydx\\ &=\iint p_\alpha(x,y)\log(1-D(x,y))dydx\\ +&\iint p(x,y)\log D(x,y)dydx=f(D(x,y)).\end{aligned}$$

Note that the function $f(D(x,y))$ achieves the maximum at $\frac{p(x,y)}{p(x,y)+p_\alpha(x,y)}$.

□

Lemma 3.2. *The global minimum value of $V(C,G)$ is $-\log 4$ and it is achieved if and only if $p(x,y)=p_\alpha(x,y)$.*

Proof. Given $D^*_{C,G}$, we can reformulate the minimax game with value function U as:

$$\begin{aligned}V(C,G)=&\max_D U(C,G,D)\\ =&\iint p(x,y)\log\frac{p(x,y)}{p(x,y)+p_\alpha(x,y)}dydx\\ &+\iint p_\alpha(x,y)\log\frac{p_\alpha(x,y)}{p(x,y)+p_\alpha(x,y)}dydx.\end{aligned}$$

Following the proof in GAN, the $V(C,G)$ can be rewritten as

$$V(C,G)=-\log 4+2JSD(p(x,y)||p_\alpha(x,y)), \tag{5}$$

where JSD is the Jensen-Shannon divergence, which is always non-negative and the unique optimum is achieved if and only if $p(x,y)=p_\alpha(x,y)=(1-\alpha)p_g(x,y)+\alpha p_c(x,y)$. □

Corollary 3.2.1. *Given $p(x,y)=p_\alpha(x,y)$, the marginal distributions are the same for any pairs of p, p_c and p_g.*

Proof. Remember that $p_g(x,y)=p(y)p_g(x|y)$ and $p_c(x,y)=p(x)p_c(y|x)$. Take integral with respect to x on both sides of $p(x,y)=p_\alpha(x,y)$ to get

$$\int p(x,y)dx=(1-\alpha)\int p_g(x,y)dx+\alpha\int p_c(x,y)dx,$$

which indicates that

$$p(y)=(1-\alpha)p(y)+\alpha p_c(y),\ \text{i.e. } p_c(y)=p(y)=p_g(y).$$

Similarly, it can be shown that $p_g(x)=p(x)=p_c(x)$ by taking integral with respect to y. □

Theorem 3.3. *The equilibrium of $\tilde{U}(C,G,D)$ is achieved if and only if $p(x,y)=p_g(x,y)=p_c(x,y)$ with $D^*_{C,G}(x,y)=\frac{1}{2}$ and the optimum value is $-\log 4$.*

Proof. According to the definition, $\tilde{U}(C,G,D)=U(C,G,D)+\mathcal{R}_\mathcal{L}$, where

$$\mathcal{R}_\mathcal{L}=E_p[-\log p_c(y|x)],$$

which can be rewritten as:

$$D_{KL}(p(x,y)||p_c(x,y))+H_p(y|x).$$

Namely, minimizing $\mathcal{R}_\mathcal{L}$ is equivalent to minimizing $D_{KL}(p(x,y)||p_c(x,y))$, which is always non-negative and zero if and only if $p(x,y)=p_c(x,y)$. Besides, the previous lemmas can also be applied to $\tilde{U}(C,G,D)$, which indicates that $p(x,y)=p_\alpha(x,y)$ at the global equilibrium, concluding the proof. □

Corollary 3.3.1. *Any additional regularization on the distances between the marginal, conditional and joint distributions of any two players, will not change the global equilibrium of $\tilde{U}$.*

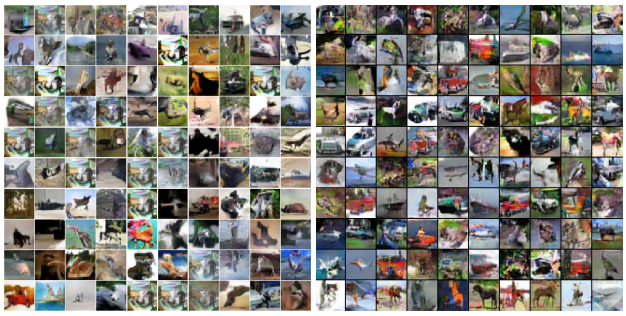
Proof. This conclusion is straightforward derived by the global equilibrium point of $\tilde{V}$. □

Pseudo data loss We prove the equivalence of the two forms of pseudo data loss in the main text as follows:

$$\begin{aligned}&D_{KL}(p_g(x,y)||p_c(x,y))+H_{p_g}(y|x)-D_{KL}(p_g(x)||p(x))\\ =&\iint p_g(x,y)\log\frac{p_g(x,y)}{p_c(x,y)}+p_g(x,y)\log\frac{1}{p_g(y|x)}dxdy\\ &-\int p_g(x)\log\frac{p_g(x)}{p(x)}dx\\ =&\iint p_g(x,y)\log\frac{p_g(x,y)}{p_c(x,y)p_g(y|x)}dxdy\\ &-\iint p_g(x,y)\log\frac{p_g(x)}{p(x)}dxdy\\ =&\iint p_g(x,y)\log\frac{p_g(x,y)p(x)}{p_c(x,y)p_g(y|x)p_g(x)}dxdy\\ =&E_{p_g}[-\log p_c(y|x)].\end{aligned}$$

Note that the last equality holds as $p_c(x)=p(x)$.

(a) Feature Matching (b) Triple-GAN



(c) Feature Matching (d) Triple-GAN

Figure 6. (a) and (c): Samples generated from Improved-GAN trained with feature matching on MNIST and CIFAR10 datasets. Strange patterns repeat on CIFAR10. (b) and (d): Samples generated from Triple-GAN.

B. Unconditional Generation

We compare the samples generated from Triple-GAN and Improved-GAN on the MNIST and CIFAR10 datasets as in Fig. 6, where Triple-GAN shares the same architecture of generator and number of labeled data with the baseline. It can be seen that Triple-GAN outperforms the GANs that are trained with the feature matching criterion on generating indistinguishable samples.

C. Class-conditional Generation on CIFAR10

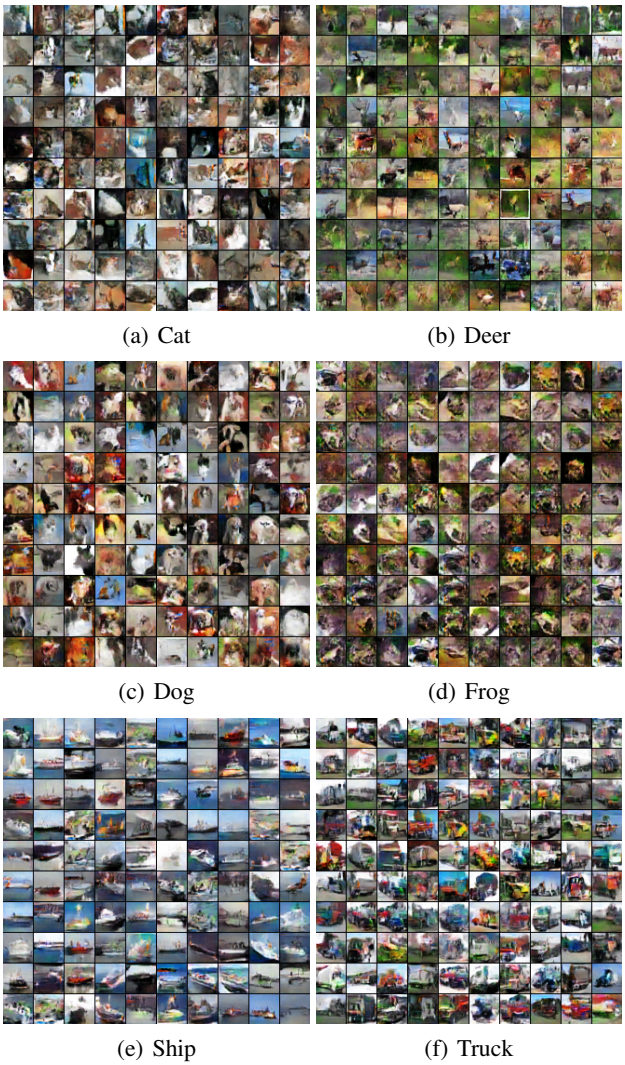
We show more class-conditional generation results on CIFAR10 in Fig. 7. Again, we can see that Triple-GAN can generate meaningful images in specific classes.

D. Interpolation on the MNIST dataset

We present the class-conditional interpolation on the MNIST dataset as in Fig. 8. We have the same conclusion as in main text that Triple-GAN is able to transfer smoothly on the data level with clear semantics.

E. Detailed Architectures

We list the detailed architectures of Triple-GAN on MNIST, SVHN and CIFAR10 datasets in Table 4, Table 5 and Table 6, respectively.



(a) Cat (b) Deer

(c) Dog (d) Frog

(e) Ship (f) Truck

Figure 7. Samples from Triple-GAN given certain class on CIFAR10.

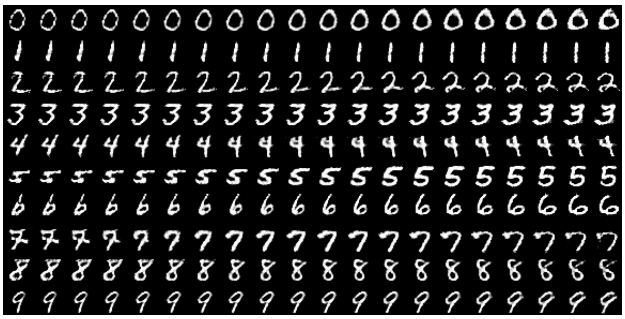


Figure 8. Class-conditional interpolation for Triple-GAN on MNIST.

Table 4. **MNIST**

Classifier C	**Discriminator** D	**Generator** G
Input 28×28 Gray Image	Input 28×28 Gray Image, Ont-hot Class representation	Input Class y, Noise z
5×5 conv. 32 ReLU	MLP 1000 units, lReLU, gaussian noise, weight norm	MLP 500 units,
2×2 max-pooling, 0.5 dropout	MLP 500 units, lReLU, gaussian noise, weight norm	softplus, batch norm
3×3 conv. 64 ReLU	MLP 250 units, lReLU, gaussian noise, weight norm	
3×3 conv. 64 ReLU	MLP 250 units, lReLU, gaussian noise, weight norm	MLP 500 units,
2×2 max-pooling, 0.5 dropout	MLP 250 units, lReLU, gaussian noise, weight norm	softplus, batch norm
3×3 conv. 128 ReLU	MLP 1 unit, sigmoid, gaussian noise, weight norm	
3×3 conv. 128 ReLU		MLP 784 units, sigmoid
Global pool		
10-class Softmax		

Table 5. **SVHN**

Classifier C	**Discriminator** D	**Generator** G
Input: 32×32 Colored Image	Input: 32×32 colored image, class y	Input: Class y, Noise z
0.2 dropout	0.2 dropout	MP 8192 units,
3×3 conv. 128 lReLU, batch norm	3×3 conv. 32, lReLU, weight norm	ReLU, batch norm
3×3 conv. 128 lReLU, batch norm	3×3 conv. 32, lReLU, weight norm, stride 2	Reshape 512×4×4
3×3 conv. 128 lReLU, batch norm		5×5 deconv. 256. stride 2,
2×2 max-pooling, 0.5 dropout	0.2 dropout	ReLU, batch norm
3×3 conv. 256 lReLU, batch norm	3×3 conv. 64, lReLU, weight norm	
3×3 conv. 256 lReLU, batch norm	3×3 conv. 64, lReLU, weight norm, stride 2	
3×3 conv. 256 lReLU, batch norm		5×5 deconv. 128. stride 2,
2×2 max-pooling, 0.5 dropout	0.2 dropout	ReLU, batch norm
3×3 conv. 512 lReLU, batch norm	3×3 conv. 128, lReLU, weight norm	
NIN, 256 lReLU, batch norm	3×3 conv. 128, lReLU, weight norm	
NIN, 128 lReLU, batch norm		
Global pool	Global pool	5×5 deconv. 3. stride 2,
10-class Softmax, batch norm	MLP 1 unit, sigmoid	sigmoid, weight norm

Table 6. **CIFAR10**

Classifier C	**Discriminator** D	**Generator** G
Input: 32×32 Colored Image	Input: 32×32 colored image, class y	Input: Class y, Noise z
Gaussian noise	0.2 dropout	MLP 8192 units,
3×3 conv. 128 lReLU, weight norm	3×3 conv. 32, lReLU, weight norm	ReLU, batch norm
3×3 conv. 128 lReLU, weight norm	3×3 conv. 32, lReLU, weight norm, stride 2	Reshape 512×4×4
3×3 conv. 128 lReLU, weight norm		5×5 deconv. 256.stride 2
2×2 max-pooling, 0.5 dropout	0.2 dropout	ReLU, batch norm
3×3 conv. 256 lReLU, weight norm	3×3 conv. 64, lReLU, weight norm	
3×3 conv. 256 lReLU, weight norm	3×3 conv. 64, lReLU, weight norm, stride 2	
3×3 conv. 256 lReLU, weight norm		5×5 deconv. 128. stride 2
2×2 max-pooling, 0.5 dropout	0.2 dropout	ReLU, batch norm
3×3 conv. 512 lReLU, weight norm	3×3 conv. 128, lReLU, weight norm	
NIN, 256 lReLU, weight norm	3×3 conv. 128, lReLU, weight norm	
NIN, 128 lReLU, weight norm		
Global pool	Global pool	5×5 deconv. 3. stride 2
10-class Softmax wieh weight norm	MLP 1 unit, sigmoid, weight norm	tanh, weight norm