Triple Generative Adversarial Nets

1st Kaiqi Wang 16337233
School of Data and Computer Sci.
Sun Yat-sen University, Guangzhou, 510275, China
wangkq3@mail2.sysu.edu.cn

***Abstract*—Omitted.**
***Index Terms*—Omitted.**

I. Introduction

Omitted.

II. Related Work

Omitted.

III. Method

A. Problem Definition

Given partially labeled dataset with x denoting the input data and y denoting the output label. Our goal is to learn 2 functions $G : x \to y$ to predict the labels y for unlabeled data as well as $D : (x, y) \to \{0, 1\}$ to generate new samples x conditioned on y.

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. 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 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 generator 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 classification and class-conditional generation in SSL, as detailed below.

We consider GANs as a semi-supervised learning model, because it uses a small amount of labeled data together with a large amount of unlabeled data.

B. Overview

Triple-GAN consists of three components: (1) a classifier C that (approximately) characterizes the conditional 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 the generator both converge to the true data distribution. To this end, we design a game with compatible utilities for the three players as follows.

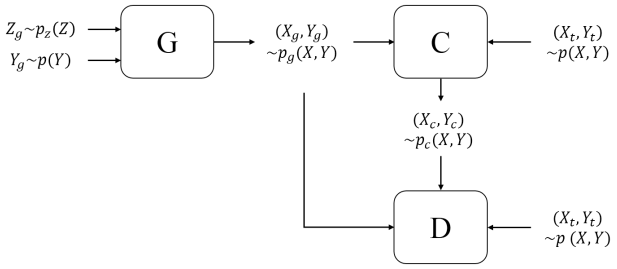

Fig. 1. A detailed structure of Triple-GANs.

The detailed structure of the Triple-GANs is shown in Figure 1. X_t, Y_t denotes labeled images and their corresponding labels, X_c denotes unlabeled images, and Z denotes noise vector uniformly distributed in $[0, 1]$. p_g, p_c and p are the distributions defined by the generator, classifier and true data generating process, respectively. The input images are $N \times N$ RGB images $X_t \in \mathbb{R}^{N \times N \times 3}$ and the input labels are Y_t. We also have unlabeled images which are $N \times N$ RGB images $X_c \in \mathbb{R}^{N \times N \times 3}$.

We make the mild assumption that the samples from both $p(x)$ and $p(y)$ can be easily obtained. In the game, after a sample x is drawn from $p(x)$, C produces a pseudo label y given x followingthe conditional distribution $p_c(y|x)$. Similarly, a pseudo input-label pair can be sampled from G by first drawing $y \sim p(y)$ and then drawing $x|y \sim p_g(x|y)$; hence from the joint distribution $p_g(x, y) = p(y)p_g(x|y)$. Similaryly, a pseudo input-label pair can be sampled from 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, $x = G(y, z), x \sim p_z(z)$, where $p_z(z)$ 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. D can also access the input-label pairs from the true data distribution as positive samples.

We refer the utilities in the process as adversarial losses,

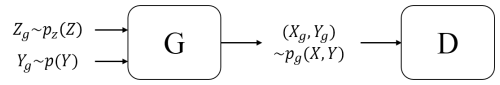

Fig. 2. The procedure of training generator: fix discriminator D. Generator G generates and labels images and sends to discriminator D. Discriminator D judges whether these labels are corresponding to the images. Generator G gets the feedback and update its model.

which can be formulated as a minimax game:

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

where $\alpha \in (0,1)$ is a constant that controls the relative importance of generation and classification and we focus on the balance case by fixing it as 1/2 throughout the paper.

The game defined in Eqn.1 archieves its equilibrium if and only if $p(x,y) = (1-\alpha)p_g(x,y) + \alpha p_c(x,y)$. The equilibrium indicates that if one of C and G tends to the data distribution, the other will also go towards the data distribution, which addresses the competing problem. However, unforunately, 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 introduce the standard supervised loss (i.e., cross-entropy loss) to C, $R_L = E_{(x,y)\sim p(x,y)}[-\log p_c(y|x)]$, which is equivalent to the KL-divergence between $p_c(x,y)$ and $p(x,y)$. Consequently, we define the game as:

$$\begin{aligned}\min_{C,G}\max_{D} \tilde{U}(C,G,D) &= E_{(x,y)\sim p(x,y)}[\log D(x,y)] \\ &+ \alpha E_{(x,y)\sim p_c(x,y)}[\log 1 - D(x,y)] \\ &+ (1-\alpha)E_{(x,y)\sim p_g(x,y)}[\log(1-D(G(y,z),y))] + R_L\end{aligned} \tag{2}$$

It will be proven that the game with utilities $\tilde{U}$ has the unique global optimum for C and G.

C. Objective Function

1) Generator: Generator G generates as well as labels images, and sends them to the discriminator. Figure 2 illustrates the procedure of training generator. Only discriminator's response will affect generator's parameters. Hence, we can write a cross-entropy loss function,

$$\nabla_{\theta_g}\left[\frac{1-\alpha}{m_g}\sum_{(x_g,y_g)}\log(1-D(x_g,y_g))\right] \tag{3}$$

where θ_g denotes learning rate.

2) Classifier: Classifier C accepts true labeled images as well as generated images, and sends them to the discriminator. And our goal is to fool discriminator. Figure 3 illustrates the procedure of training classifier. Hence, we can write a cross-entropy loss function:

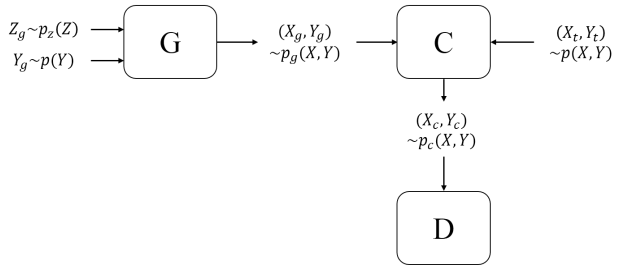

Fig. 3. The procedure of training classifier: fix discriminator D and generator G. Classifier C accepts true labeled images and generated images and sends to discriminator D. Discriminator D judges whether these labels are corresponding to the images. Classifier C gets the feedback and tries to improve its classification accuracy to let discriminator accept its verdict.

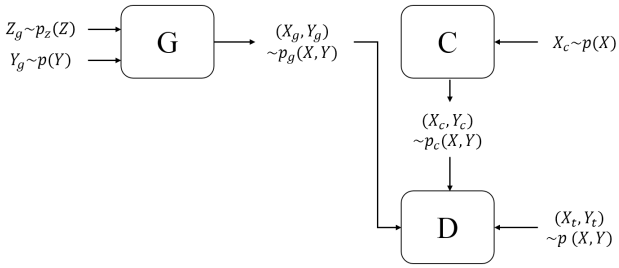

Fig. 4. The procedure of training discriminator: fix generator G and classifier C. Discriminator D accepts true images with true labels from true data, generated images with generated labels from generator G, and true/generated images with generated labels from classifier C. Discriminator D gets the source of images and labels, and updates the parameters.

$$\nabla_{\theta_c}\left[\frac{\alpha}{m_c}\sum_{(x_c,y_c)} p_c(y_c|x_c)\log(1-D(x_c,y_c))\right] \tag{4}$$

3) Discriminator: Discriminator D accepts true images with true labels from true data, generated images with generated labels from generator G, and true/generated images with generated labels from classifier C. Figure 4 illustrates the procedure of training discriminator. Hence, we can write a cross-entropy loss function:

$$\begin{aligned}\nabla_{\theta_d}[\frac{1}{m_d}(&\sum_{(x_d,y_d)}\log D(x_d,y_d)) \\ &+ \frac{\alpha}{m_c}\sum_{(x_c,y_c)}\log(1-D(x_c,y_c)) \\ &+ \frac{1-\alpha}{m_g}\sum_{(x_g,y_g)}\log(1-D(x_g,y_g))]\end{aligned} \tag{5}$$

D. Theoretical Analysis

Omitted.

IV. Pratical Techniques

Omitted.

V. Experiments

Omitted.

VI. Conclusions

Omitted.