

## A APPENDIX: PROOF OF THEOREM 1

Firstly, we recall some definitions. Denote  $x_r, x_g$  as the real training and generated samples, respectively.  $x$  are the population of all data, and  $x_r$  are sampled from  $p(x)$ .  $y_g$  represents the initial labels for the generator  $G$ , while  $\tilde{y}$  indicates the labels perturbed by  $\tilde{C}$  from  $y_g$ . The class-prior  $\pi_i$  meets  $\pi_i = P(y_g = i) = P(\mathcal{O}(x_r) = i)$ . For a rigorous proof of Theorem 1, we elaborate it again in the appendix.

**Theorem 1** We assume that the following three mild assumptions can be met: (a) PU classifier is not over-fitting on the training data, (b)  $P(PU_\theta(x_g)|\mathcal{O}(x_g), y_g) = P(PU_\theta(x_g)|\mathcal{O}(x_g))$ , (c) the conditional sample space is disjoint from each other class. Then,

(1)  $P^g$  is a permutation matrix if the generator  $G$  in CNI-CGAN is optimal, with the permutation, compared with an identity matrix, only happens on rows  $\mathbf{r}$  where corresponding  $\pi_r, r \in \mathbf{r}$  are equal.

(2) If  $P^g$  is an identity matrix and the generator  $G$  in CNI-CGAN is optimal, then  $p^r(x, y) = p^g(x, y)$  where  $p^r(x, y)$  and  $p^g(x, y)$  are the real and generating joint distribution, respectively.

### A.1 PROOF OF (1)

*Proof.* For a general setting, the oracle class of  $x_g$  given by label  $y_g$  is not necessarily equal to  $PU_\theta(x_g)$ . Thus, we consider the oracle class of  $x_g$ , i.e.,  $\mathcal{O}(x_g)$  in the proof.

**Optimal  $G$ .** In CNI-CGAN,  $G$  is optimal if and only if

$$p^r(x_r, PU_\theta(x_r)) = p^g(x_g, \tilde{y}). \quad (10)$$

The equivalence of joint probability distribution can further derive the equivalence of marginal distribution, i.e.,  $p^r(x_r) = p^g(x_g)$ . We define a probability matrix  $C$  where  $C_{ij} = P(PU_\theta(x) = j | \mathcal{O}(x) = i)$  where  $x$  are the population data. According to (c), we can apply  $\mathcal{O}(\cdot)$  on both  $x_r$  and  $x_g$  in Eq. 10. Then we have:

$$\begin{aligned} P(\mathcal{O}(x_r) = i, PU_\theta(x_r) = j) &\stackrel{(c)}{=} P(\mathcal{O}(x_g) = i, \tilde{y} = j) \\ P(\mathcal{O}(x_r) = i)P(PU_\theta(x_r) = j | \mathcal{O}(x_r) = i) &= \sum_{k=1}^{K+1} P(y_g = k, \mathcal{O}(x_g) = i)P(\tilde{y} = j | y_g = k, \mathcal{O}(x_g) = i) \\ \pi_i C_{ij} &\stackrel{(a)}{=} \sum_{k=1}^{K+1} P(\mathcal{O}(x_g) = i | y_g = k)P(y_g = k)P(\tilde{y} = j | y_g = k) \\ \pi_i C_{ij} &= \sum_{k=1}^{K+1} P_{ik}^{g\top} \pi_k \tilde{C}_{kj}, \end{aligned} \quad (11)$$

where assumption (a) indicates that  $PU_\theta(x_r)$  is close to  $PU_\theta(x)$  so that  $P(PU_\theta(x_r) = j | \mathcal{O}(x_r) = i) = P(PU_\theta(x) = j | \mathcal{O}(x) = i)$ . Then the corresponding matrix form follows as

$$\Pi C = P^{g\top} \Pi \tilde{C} \quad (12)$$

**Definition.** According to the definition of  $\tilde{C}$  and Law of Total Probability, we have:

$$\begin{aligned} P(y_g = i)P(PU_\theta(x_g) = j | y_g = i) &= \pi_i \sum_{k=1}^{K+1} P(\mathcal{O}(x_g) = k | y_g = i)P(PU_\theta(x_g) = j | \mathcal{O}(x_g) = k, y_g = i) \\ \pi_i \tilde{C}_{ij} &\stackrel{(b)}{=} \pi_i \sum_{k=1}^{K+1} P_{ik}^g P(PU_\theta(x_g) = j | \mathcal{O}(x_g) = k) \\ \pi_i \tilde{C}_{ij} &= \pi_i \sum_{k=1}^{K+1} P_{ik}^g C_{kj}, \end{aligned} \quad (13)$$

where the last equation is met as  $p(x_g)$  is close to  $p(x)$  when  $G$  is optimal, and thus  $P(PU_\theta(x_g) = j | \mathcal{O}(x_g) = k) = P(PU_\theta(x) = j | \mathcal{O}(x) = k)$ . Then we consider the corresponding matrix form as follows

$$\Pi \tilde{C} = \Pi P^g C \quad (14)$$

where  $\Pi$  is the diagonal matrix of prior vector  $\pi$ . Combining Eq. 14 and 12, we have  $P^{g\top} \Pi P^g = \Pi$ , which indicates  $P^g$  is a general orthogonal matrix. In addition, the element of  $P^g$  is non-negative and the sum of each row is 1. Therefore, we have  $P^g$  is a permutation matrix with permutation compared with the identity matrix only happens on rows  $\mathbf{r}$  where corresponding  $\pi_r, r \in \mathbf{r}$  are equal. Particularly, if all  $\pi_i$  are different from each other, then permutation operation will not happen, indicating the optimal conditional of  $P^g$  is the identity matrix.

## A.2 PROOF OF (2)

We additionally denote  $y_r$  as the real label of real sample  $x_r$ , i.e.,  $y_r = \mathcal{O}(x_r)$ . According to the optimal condition of  $G$  in Eq. 10, we have  $p^r(x_r) = p^g(x_g)$ . Since we have  $P^g$  is an identity matrix, then  $\mathcal{O}(x_g) = y_g$  a.e. Thus, we have  $p^g(x_g|y_g = i) = p^g(x_g|\mathcal{O}(x_g) = i), \forall i = 1, \dots, K + 1$ . According the assumption (c) and Eq. 10, we have  $p^r(x_r|\mathcal{O}(x_r) = i) = p^g(x_g|\mathcal{O}(x_g) = i)$ . In addition, we know that  $p^r(x_r|\mathcal{O}(x_r) = i) = p^r(x_r|y_r = i)$ , thus we have  $p^r(x_r|y_r = i) = p^g(x_g|y_g = i)$ . Further, we consider the identical class-prior  $\pi_i$ . Finally, we have

$$\begin{aligned} p^r(x_r|y_r = i)\pi_i &= p^g(x_g|y_g = i)\pi_i \\ p^r(x_r|y_r = i)p(\mathcal{O}(x_r) = i) &= p^g(x_g|y_g = i)p(y_g = i) \\ p^r(x_r|y_r = i)p(y_r = i) &= p^g(x_g|y_g = i)p(y_g = i) \\ p^r(x_r, y_r) &= p^g(x_g, y_g). \end{aligned} \tag{15}$$

□

## B APPENDIX: RELATED WORKS

**Positive-Unlabeled (PU) Learning.** Positive and Unlabeled (PU) Learning is the setting where a learner only has access to positive examples and unlabeled data. Early work (Bekker & Davis, 2018) did a survey around this. Non-Negative Risk Estimator (Kiryu et al., 2017) has been proposed to alleviate the overfitting, and thus it allows to utilize very flexible model, such as deep neural networks. Similarly, Hou et al. (2017) employed GANs (Goodfellow et al., 2014) to recover both positive and negative data distribution to step away from overfitting. Kato et al. (Kato et al., 2018) focused on remedy the selection bias in the PU learning, and Besides, Multi-Positive and Unlabeled Learning (Xu et al., 2017) extended the binary PU setting to the multi-class version, therefore adapting to more practical applications. Our Multi-Positive Unlabeled method, by contrast, is more intuitive and tailored for the deep neural networks optimization.

**CGAN on Few Labels Data.** To attain high-quality images with both fidelity and diversity, the training of generative models requires a large dataset. To reduce the need of huge amount of data, the vast majority of methods (Noguchi & Harada, 2019; Yamaguchi et al., 2019; Zhao et al., 2020) attempted to transfer prior knowledge of the pre-trained generator. Another branch (Lucic et al., 2019) is to leverage self- and supervised learning to add pseudo labels on the in-distribution unlabeled data in order to expand labeled dataset. Compared with this approach, our strategy can be viewed to automatically “pick” useful in-distribution data from total unknown unlabeled data via PU learning framework, and then constructs robust CGANs to generate clean data distribution out of predicted label noise.

**Robust GANs.** Existing Robust GANs can be mainly categorised into two types: ones robust to noisy labels and the others robust to noisy inputs. Robust Conditional GANs (Thekumparampil et al., 2018; Kaneko et al., 2019) were proposed to class-dependent noisy labels. The main idea of these approaches is to corrupt the label of generated sample before feeding to the adversarial discriminator, forcing the generator to produce sample with clean labels. As supplementary investigation, Koshy Thekumparampil et al. (2019) explored the scenario when CGANs get exposed to missing or ambiguous labels, while Chrysos et al. (2018) leveraged structure in the target space of the model to address this issue. Moreover, Noise RCGAN (Kaneko & Harada, 2019) focused on the robust generation against noisy inputs. Different from these works, the noise in our model mainly stems from the prediction error of existing PU classifier. We employ the imperfect classifier to estimate the label confusion noise, yielding a new branch of Robust CGANs against “classifier” noise.

**Semi-Supervised Learning (SSL).** There is a recent wave of approaches for semi-supervised learning, e.g., Virtual Adversarial Training (VAT) (Miyato et al., 2018) and its variants (Yu et al., 2019), Mix-Match (Berthelot et al., 2019) and its variant (Sun et al., 2019). One crucial issue in SSL is how to tackle with the mismatch of unlabeled and labeled data. *Augmented Distribution Alignment* (Wang et al., 2019) was proposed to leverage adversarial training to alleviate the bias, but they focus on the empirical distribution mismatch owing to the limited number of labeled data. Further, Yanbei Chen (2019) concentrated on this under-studied problem and designed a *Uncertainty Aware Self-Distillation* to guarantee the effectiveness of learning. In contrast, our approach leverage PU learning to construct the “open world” classification, which can be further investigated to cope with this issue in the future.

**Out-Of-Distribution (OOD) Detection** OOD Detection is one classical but always vibrant machine learning problem. PU learning can be used for the detection of outliers in an unlabeled dataset with knowledge only from a collection of inlier data (Hido et al., 2008; Smola et al., 2009). Another interesting and related work is *Outlier Exposure* (Hendrycks et al., 2018), an approach that leveraged an auxiliary dataset to enhance the anomaly detector based on existing limited data. This problem is similar to our generation task, the goal of which is to take better advantage of extra dataset, especially out-of-distribution data, to boost the generation.

**Learning from Noisy Labels** Rotational-Decoupling Consistency Regularization (RDCR) (Tsung Wei Tsai, 2019) was designed to integrate the consistency-based methods with the self-supervised rotation task to learn noise-tolerant representations. Ge et al. (2020) proposed Mutual Mean-Teaching that averages the parameters of two neural networks to refine the soft labels on person re-identification task. In addition, the data with noisy labels can also be viewed as bad data. Guo et al. (2019) provided a worst-case learning formulation from bad data, and designed a data-generation scheme in an adversarial manner, augmenting data to improve the current classifier.

## C APPENDIX: DETAILS ABOUT ALGORITHM 1

Similar in (Kiryu et al., 2017), we utilize the sigmoid loss  $\ell_{\text{sig}}(t, y) = 1/(1 + \exp(ty))$  in the implementation of the PU learning. Besides, we denote  $r_i = \hat{R}_u^-(g; \mathcal{X}_u^i) - \pi_p \hat{R}_p^-(g; \mathcal{X}_p^i)$  in the  $i$ -th mini-batch. Instructed by the algorithm in (Kiryu et al., 2017), if  $r_i < 0$  we turn to optimize  $-\nabla_{\theta} r_i$  in order to make this mini-batch less overfitting, which is slightly different from Eq. 4.

## D APPENDIX: DETAILS ABOUT EXPERIMENTS

**PU classifier and GAN architecture** For the PU classifier, we employ 6 convolutional layers with different number of filters on MNIST, Fashion-MNIST and CIFAR 10, respectively. For the GAN architecture, we leverage the architecture of generator and discriminator in the tradition conditional GANs (Mirza & Osindero, 2014). To guarantee the convergence of RCGAN-U, we replace Batch Normalization with Instance Batch Normalization. The latent space dimensions of generator are 128, 128, 256 for the three datasets, respectively. As for the optimization of GAN, we deploy the avenue same as WGAN-GP (Gulrajani et al., 2017) to pursue desirable generation quality. Specifically, we set update step of discriminator as 1.

**Choice of Hyper-parameters** We choose  $\kappa$  as 0.75,  $\beta$  as 5.0 and  $\lambda = 0.99$  across all the approaches. The learning rates of PU classifier and CGAN are 0.001 and 0.0001, respectively. In the alternate minimization process, we set the update step as 1 for PU classifier after updating the CGAN, and  $L_0$  as 5 in Algorithm 1.

**Further Evaluation of CGAN-P and Ours from the Aspect of Inception Score** To better verify our approach can generate more pleasant images than CGAN-P, we additionally compare the Inception Score these two methods attain. Specifically, we trained a (almost) perfect classifier with 99.21 % and 91.33% accuracy for MNIST and Fashion-MNIST respectively. Then we generate 50,000 samples from the two approaches to compute Inception Score, the results of which are exhibited in Table 2. It turns out that our method attain the consistent superiority against CGAN-P on the Inception Score for MNIST, even though the generator label accuracy of these two approaches are comparable. Note that the two method obtains the similar Inception Score on Fashion-MNIST, but our strategy outperforms CGAN-P significantly from the perspective of generator label accuracy. Overall, we can claim that our method is better than CGAN-P.

Table 2: Further evaluation of CGAN-P and Ours from the perspective of Inception Score on MNIST and Fashion-MNIST datasets.

Positive Rates		0.75%	1.0%	3.0%	5.0%	10.0%
Inception Score ( $\pm$ Standard Deviation)						
MNIST	CGAN-P	5.08 $\pm$ 0.02	5.10 $\pm$ 0.03	5.09 $\pm$ 0.02	5.14 $\pm$ 0.03	5.10 $\pm$ 0.04
	Ours	<b>5.60<math>\pm</math>0.01</b>	<b>5.59<math>\pm</math>0.02</b>	<b>5.65<math>\pm</math>0.02</b>	<b>5.52<math>\pm</math>0.01</b>	<b>5.63<math>\pm</math>0.02</b>
Fashion-MNIST	CGAN-P	4.95 $\pm$ 0.03	5.01 $\pm$ 0.03	5.04 $\pm$ 0.04	5.02 $\pm$ 0.04	5.00 $\pm$ 0.03
	Ours	4.99 $\pm$ 0.02	5.01 $\pm$ 0.02	5.03 $\pm$ 0.01	5.07 $\pm$ 0.02	5.04 $\pm$ 0.02

## E APPENDIX: MORE IMAGES

We additionally show some generated images on other datasets generated by baselines and CNI-CGAN, shown in Figure 6. Note that we highlight the erroneously generated images with red boxes. Specifically, on Fashion-MNIST our approach can generate images with more accurate labels compared with CGAN-A and RCGAN-U. Additionally, the quality of generated images from our approach are much better than those from CGAN-P that only leverages limited supervised data, as shown in Figure 7 on CIFAR-10.

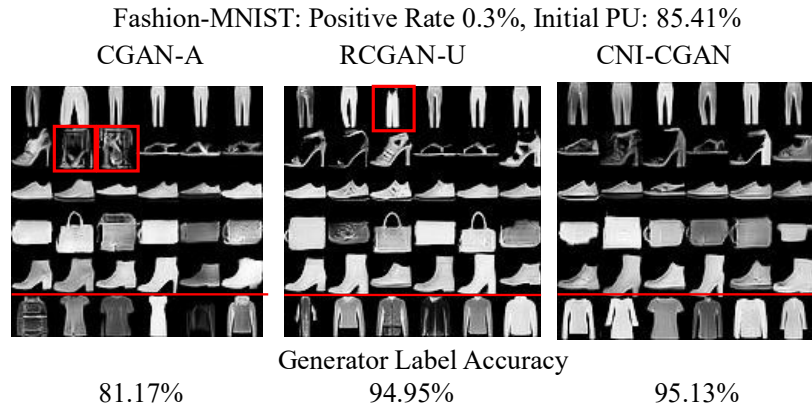


Figure 6: Visualization of generated samples from several baselines and ours on Fashion-MNIST.

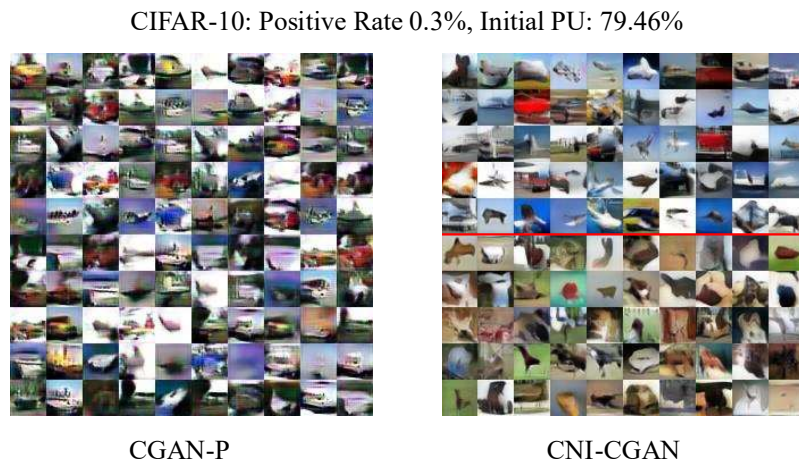


Figure 7: Visualization of generated samples from CGAN-P and ours on CIFAR-10.