Expert Systems With Applications 224 (2023) 120046

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Conditional generative positive and unlabeled learning

Aleš Papič *, Igor Kononenko, Zoran Bosnić

University of Ljubljana, Faculty of Computer and Information Science, Večna pot 113, Ljubljana, 1000, Slovenia

ARTICLE INFO

Dataset link: https://github.com/apapich/CGenPU

Keywords:
Positive and unlabeled learning
Partially supervised learning
Generative adversarial networks
Deep learning

ABSTRACT

The quantity of data generated increases daily, which makes it difficult to process. In the case of supervised learning, labeling training examples may represent an especially tedious and costly task. One of the aims of positive and unlabeled (PU) learning is to train a binary classifier from partially labeled data, representing a strategy for combining supervised and semi-supervised learning and alleviating the cost of labeling data fully. Still, the main strength of PU learning arises when the negative data are not directly available or too diverse. Although the generative approaches have shown promising results in this field, they also bring shortcomings, such as high computational cost, training instability, and inability to generate fully labeled datasets. In the paper, we propose a novel *Conditional Generative PU* framework (CGenPU) with a built-in auxiliary classifier. We develop a novel loss function to learn the distribution of positive and negative examples, which leads to a unique, desirable equilibrium under a nonparametric assumption. Our CGenPU is evaluated against existing generative approaches using both synthetic and real data. The characteristics of various methods, including ours, are depicted with different toy examples. The results demonstrate the *state-of-the-art* performance on standard positive and unlabeled learning benchmark datasets. Given only ten labeled CIFAR-10 examples, CGenPU achieves classification accuracy of 84%, while current *state-of-the-art* D-GAN framework achieves 54%. On top of that, CGenPU is the first single-stage generative framework for PU learning.

1. Introduction

The pace of data generation is rapidly increasing each year, which causes difficulties in data preparation for machine learning (Jaskie & Spanias, 2019), which demands the continuous improvement of existing methods. Binary supervised learning classification tasks require labeled training data, *e.g.* as positive and negative. Nevertheless, acquiring labeled data is not always trivial due to the lack of human experts, associated costs, time restrictions, or even the limitations of data acquisition processes, which motivates us for our work. The area of semi-supervised learning, called Positive and Unlabeled (PU) learning, is being increasingly studied considering these practical limitations. PU learning is a binary, positive and negative classification problem, where only labeled positive and unlabeled examples are available (Li & Liu, 2003; Liu, Lee, Yu, & Li, 2002). Because of a weak supervision, PU learning is considered more challenging than classical supervised and semi-supervised classification tasks. However, a comprehensive study by Bekker and Davis (2020) identified various applications where PU learning could prove a better alternative to existing methods. These applications are related to solving tasks in text analysis, biology, medicine, targeted marketing, and others.

Existing methods in the literature that address this problem can be divided into three categories, depending on how we handle unlabeled data. The first category of techniques first samples potential negative (and positive) examples used for further supervised learning (Northcutt, Wu, & Chuang, 2017). The sampling quality strongly depends on the chosen heuristic, which is often associated with the data knowledge (Yang, Liu, & Yang, 2017). However, when we have vast amounts of data, this can become a problem. There is also a danger that the learned classifier has seen a narrow subset of examples, which reduces its generalization capabilities (Chiaroni, Khodabandelou, Rahal, Hueber, & Dufaux, 2020). The second category of techniques works by weighting positive and unlabeled data (Du Plessis, Niu, & Sugiyama, 2015; Kiryo, Niu, Du Plessis, & Sugiyama, 2017). The loss for negative data is approximated by the weighted difference between unlabeled and positive examples. In practice, methods that approximate the loss for negative data achieve higher classification accuracy than sampling techniques. However, the main drawback is a requirement to know the distribution of data, which is often unknown, especially in large datasets. It also affects learning stability using stochastic optimization techniques because the distribution within the batch often differs from the distribution of the whole learning dataset (Chiaroni et al., 2020). The third category of techniques is based on generative adversary networks (GANs) (Goodfellow et al., 2014). They learn data distribution and generate an artificial set of labeled negative (and positive)

* Corresponding author.
E-mail addresses: ales.papic@fri.uni-lj.si (A. Papič), igor.kononenko@fri.uni-lj.si (I. Kononenko), zoran.bosnic@fri.uni-lj.si (Z. Bosnić).

https://doi.org/10.1016/j.eswa.2023.120046
Received 17 June 2022; Received in revised form 12 March 2023; Accepted 1 April 2023
Available online 6 April 2023
0957-4174/© 2023 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

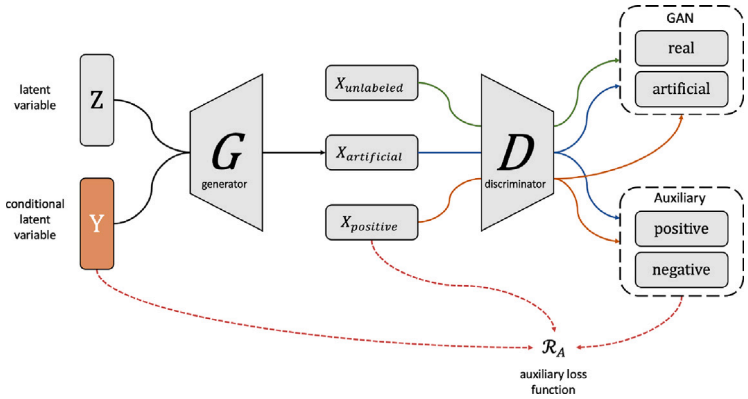


Fig. 1. The scheme of the proposed CGenPU learning framework, with a class-conditional generator, a discriminator and an auxiliary classifier (seen in dotted boxes on the right-hand side). Solid arrows indicate the flow of data within the framework, while the red dashed lines refer to the flow of labels. The generator takes as an input a latent vectors z and a class labels y to generate artificial examples (X_{art}). Next, the discriminator evaluates artificial examples for their quality and classifies them as positive or negative. The auxiliary loss $\mathcal{R}_A$ is then computed for artificial and labeled examples (X_{pos}), which is added to the standard GAN loss computed on both unlabeled (X_{unl}) and labeled examples. The framework's output is (i) a generator capable of generating artificial positive and negative data, and (ii) a binary classifier capable of classifying in positive or negative class.

examples to train the classifier. GANs can absorb large quantities of data if having enough expressive power. Therefore GANs are also being applied to the domain of PU learning. However, existing approaches include shortcomings such as high computational complexity, which leads to learning instability, and the requirement to know the data distribution (Hou, Chaib-Draa, Li, & Zhao, 2018). Moreover, some approaches cannot learn the distribution of positive examples, which is essential when labeled data is difficult to obtain or to label (Chiaroni et al., 2020; Chiaroni, Rahal, Hueber, & Dufaux, 2018).

In this work, we propose a novel *Conditional Generative PU* (CGenPU) framework, capable of learning the positive and negative data distributions without knowing the unlabeled data distribution. On top of that, CGenPU requires simple architecture (Fig. 1) and yields a trained classifier using a single-stage training. Our approach is based on the Auxiliary Classifier GAN (AC-GAN) (Odena, Olah, & Shlens, 2017), which generates class-conditional examples. We reuse its framework architecture and training scheme. However, to discriminate positive and negative examples, we introduce a novel PU auxiliary loss that enables learning from PU datasets. Exploiting AC-GANs architecture makes CGenPU applicable to a variety of known GAN architectures. We show its effectiveness through theoretical and experimental analysis, showing that the proposed CGenPU is significantly better than other similar approaches in this field.

Our work comprises the following four contributions:

1. We propose a novel GAN-based framework for PU learning, capable of conditional generation and binary classification;
2. We propose a novel PU auxiliary loss, capable of guiding the generator to separate the distribution of unlabeled data into positive and negative;
3. We prove that CGenPU's objective function leads to a unique, desirable equilibrium under a non-parametric assumption;
4. We evaluate the CGenPU on the MNIST and CIFAR-10 data and show that it can outperform existing GAN-based PU learning methods in One-vs-One and One-vs-Rest scenarios.

The paper is divided into the following sections. Section 2 describes the related methodology, while Section 3 provides a short background about GANs. Section 4 describes the proposed approach, followed by experimental evaluation in Section 5. Finally, in Section 6 and Section 7, we discuss our work's limitations and future opportunities.

2. Related work

In this section, we review existing work in the field of PU learning and highlight both their advantages and limitations. We begin with unbiased techniques and continue with generative approaches, which are increasingly studied lately. Finally, we summarize the limitations, which are the grounds for our work.

2.1. Unbiased PU learning

Du Plessis, Niu, and Sugiyama (2014) showed that PU learning could be solved with cost-sensitive learning if the class prior of unlabeled data is known, but the solution is often biased, which a non-convex loss can cancel. However, training a classifier with non-convex loss is difficult in practice; thus, Du Plessis et al. (2015) provided a formulation for convex PU learning (uPU). Kiryo et al. (2017) have later shown that complex models overfit as the empirical risk becomes negative. Therefore, a non-negative version (nnPU) was proposed, which constraints empirical risk to the positive interval during optimization. Both uPU and nnPU require prior knowledge in the empirical loss, which requires experts analysis or estimation (Christoffel, Niu, & Sugiyama, 2016). Either way, with the abundance of data getting reliable prior is tricky. Further, Chiaroni et al. (2020) highlighted that the prior varies per batch, which can make stochastic optimization techniques unstable. Further, the issue of overfitting for nnPU remains (Hou et al., 2018). Recent research focuses on novel strategies for PU learning using generative adversarial networks. Their goal is to replace unlabeled examples with labeled artificial that are used for further supervised learning. They can also absorb large quantities of data, which makes them particularly interesting.

2.2. Generative Adversarial Networks

Generative Adversarial Networks (GANs) are novel class methods that train generative models, which generate realistic artificial examples, enabling the augmentation of existing data sources (Goodfellow et al., 2014). The training scheme contains the generator and the discriminator that learn through an adversarial competition. The generator creates examples that exemplify the distribution of real data. The discriminator, on the other hand, assesses whether the given examples are real or artificial. GANs, as proposed, use unsupervised learning and treat data as unlabeled. However, having more information, *e.g.* class labels, can improve structure of the latent space. Mirza and Osindero (2014) proposed a strategy that supplies both the generator and discriminator with class labels to produce class conditional examples, using Conditional GAN (CGAN). Odena et al. (2017) proposed an alternative variant of CGAN called *Auxiliary Classifier* GAN (AC-GAN), where the discriminator has the auxiliary task to reconstruct conditional latent information. While the objective function remains

unchanged for CGAN, the AC-GAN introduces the auxiliary objective function to learn a joint hidden representation. The downside of both is that they require a fully labeled dataset, which is not available in the PU setting. Still, many relevant problems in machine learning are facing a shortage of labeled examples for training generative models (Ren et al., 2023). However, novel generative approaches for PU learning can address these problems.

2.3. Generative PU learning

The large quantities of unlabeled data cause learning challenges for existing PU methods. Recently, several GAN-based PU methods have shown encouraging results. Hou et al. (2018) proposed a generative positive and unlabeled framework (GenPU) that incorporates a series of generators and discriminators, each with a different role that simultaneously generate realistic positive and negative examples. The scheme of GenPU contains five neural networks; two generators, the task of which is to generate artificial positive and negative examples, and three discriminators that evaluate generators from the perspective of positive, negative and unlabeled distributions. However, learning a set of five adversarial networks is computationally demanding and increases the framework's instability. This makes the GenPU less applicable to complex datasets, such as CIFAR-10, which contains color images of high diversity and many details. It also requires prior knowledge, which is not always available. Chiaroni et al. (2018) proposed another GAN-based PU framework called Positive-GAN (PGAN). PGAN, in contrast with GenPU, uses the original GAN architecture. The framework is based on the optimistic assumption that unlabeled data contain mostly negative examples, making it prone to overfitting. Addressing the disadvantages of PGAN and GenPU frameworks, Chiaroni et al. (2020) proposed another GAN-based PU framework called Divergent-GAN (D-GAN). D-GAN introduces a biased PU risk within the standard GAN's objective function, forcing the generator to diverge from the distribution of positive examples. However, it is still limited to generating negative examples only.

Na et al. (2020) proposed a VAE-PU approach that combines the variational autoencoder and GAN for positive and unlabeled learning. The advantage of VAE-PU is that it does not depend on the *selected completely at random* assumption, making it less prone to convergence issues if labeled data is not randomly selected. However, training the classifier requires pre-training VAE-PU. PURE by Zhou et al. (2021) is another GAN-based approach that incorporates the unbiased positive and unlabeled risk into the discriminator to identify the relevant user–item pairs. The generator's task is to improve the robustness of the discriminator, which has to learn to recognize high-quality non-relevant examples. Another application of generative PU learning is the SRPUGAN-Charbon method, designed for super-resolution image reconstruction (Xu et al., 2022). Although PURE and SRPUGAN-Charbon are intriguing approaches, they are designed for domain-specific tasks, making them challenging to apply to other problems. Hu et al. (2021) proposed Predictive Adversarial Network (PAN), which adapts the GAN and replaces the generator with the classifier. The classifier competes with the discriminator and tries to replicate its predictions. The adaptation of GAN allowed the formulation of PAN's objective function without needing prior data distribution knowledge. Nevertheless, there is still space for further improving PAN's accuracy. Cho, Kim, Khan, and Choo (2022) proposed a novel single-to-multi-label (S2M) sampling technique, which enables existing unconditional and conditional GANs to learn from PU data. However, on top of training GAN, S2M requires three classifiers to compute the acceptance probability used in the Markov chain Monte Carlo method, significantly increasing the computational requirements.

Table 1
Comparison of the advantages of presented *state-of-the-art* PU methods (based on Chiaroni et al., 2020). The final column describes advantages of the proposed CGenPU framework. Check-mark (✓) denotes that the method carries the respective advantage from the leftmost column.

Advantage\Methods	GenPU	PGAN	D-GAN	*CGenPU*
Prior knowledge not required		✓	✓	✓
Applicable to complex datasets		✓	✓	✓
Generates relevant positive examples	✓			✓
Generates relevant negative examples	✓		✓	✓
Training stability using SGD		✓	✓	✓
Single-stage training				✓

2.4. Summary

Generative PU learning leverages large quantities of data, whereas former methods usually fail. However, current frameworks demand a lot of computational resources. Additionally, many assume that a large number of positive examples are available, which may not be achievable. We address these limitations in our novel framework, called *CGenPU*, which we compare to existing GAN methods for PU learning in Table 1, which is based upon (Chiaroni et al., 2020). Novel CGenPU is a computationally more efficient version of the existing GenPU framework. By reducing the complexity, it is not only more stable but can also be used on less sophisticated hardware. Also, when learning is complete, the resulting auxiliary classifier can be immediately used in production. Thus, CGenPU is the first GAN-based single-stage approach for PU learning.

3. Background

GANs can generate realistic artificial examples that resemble the input data distribution. This ability comes through an adversarial competition between generator G and discriminator D. The G aims to generate examples that exemplify the distribution of real data. The discriminator, on the other hand, assesses whether the given examples are real or artificial. Adversarial competition is guided by the objective value function *w.r.t.* D's output, where D maximizes the probability of assigning the correct label to both real and artificial data. Simultaneously, G aims to minimize D's probability of assigning the correct label to the artificial data. Formally, adversarial competition is guided by the minmax objective value function V, Eq. (1), where $p_u(x)$ denotes the distribution of real data, $p_z(x)$ is a straightforward prior distribution, such as $\mathcal{N}(0,1)$ or $\mathcal{U}(0,1)$, of the latent vector z, and G represents the transformation $G(z) : z \mapsto x$.

$$\min_G \max_D V(G,D) = \min_G \max_D \Big\{ \mathbb{E}_{x\sim p_u(x)}[\log(D(x))] + \\ + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \Big\} \tag{1}$$

Optimizing Eq. (1), specifically minimizing G, is equivalent to minimizing the distance between the distribution of real and artificial data (Goodfellow et al., 2014). The learning process runs by updating the D and G alternately through stochastic gradient descent. Nonetheless, the idea is to slowly improve G to avoid its collapse, where many latent representations yield a small diversity in G's output. Assuming that GAN has sufficient time and capacity to converge, then $p_g(x) = p_u(x)$.

4. Conditional generative PU learning

Our work addresses the limitations of GenPU (Hou et al., 2018), such as high computational requirements and training instability resulting from the adversarial competition of five neural networks (two generators and three discriminators). Generators learn a positive or

negative data distribution, one each. The discriminator evaluates generators from the perspective of positive, negative and unlabeled distributions. We replace two generators with a single one that provides artificial examples conditioned by a class label. This feature allows us to use a single discriminator, whose task is to output two scalars: the first scalar represents the probability that the given example is real rather than artificial, and the second represents the probability that the given example is positive rather than negative. The implemented learning framework is based on an Auxiliary Classifier GAN (AC-GAN) by Odena et al. (2017). AC-GAN provides additional structure for the generator's latent space, such as class information. The discriminator contains an auxiliary classifier network tasked to decode the class information. However, AC-GAN requires labeled examples that are not available in the PU domain. Thus, we propose a novel auxiliary objective function, $\mathcal{R}_{\mathcal{A}}$, that enables learning from PU data. The modified architecture preserves the ability to generate positive examples, which is not the case for PGAN and D-GAN. Having this ability is necessary for situations where positive examples are difficult to obtain or to label. We address these limitations in a novel *Conditional* generative PU framework, which we call CGenPU.

CGenPU adopts the idea of AC-GAN, although not to stabilize the training but to learn the distribution of positive and unknown negative data. The CGenPU is structurally the same as the existing AC-GAN. The generator takes as an input a latent vector and a class label to generate artificial examples, which are then evaluated for their quality and classified as positive or negative by the discriminator. The auxiliary loss is then computed for artificial and labeled examples, which is added to the standard GAN loss. However, to train the auxiliary classifier using PU data, we have developed a novel objective function $\mathcal{R}_{\mathcal{A}}$. The $\mathcal{R}_{\mathcal{A}}$ maximizes the similarity between positive and negative posterior distributions. In other words, the auxiliary classifier learns to classify labeled and artificial positive examples into the same class while artificial negative examples into the opposite class. Since labeled negative examples are unavailable, the classifier relies on the generator to split the positive and negative distributions using unlabeled data and the feedback from the classifier through $\mathcal{R}_{\mathcal{A}}$. Fig. 1 depicts the training scheme of the CGenPU framework, where colors illustrate how data and labels flow in the framework. The aim is to derive the positive $p^p(x)$ and negative $p^n(x)$ distributions from PU data, where the majority of training examples are unlabeled.

CGenPU is a single-stage approach, although we can use the generated data to train any existing binary classifier. CGenPU is also computationally less demanding, *i.e.*, requires less parameters to optimize, and has higher stability compared to GenPU (see Section 5). Additionally, it does not require prior knowledge of unlabeled data distribution, which represents an important advantage of the proposed approach.

4.1. The objective of the CGenPU

We first define the notation used in this section. Let $X \in \mathbb{R}^m$ be a random input and $Y \in \{0,1\}$ a random output variable. The input random variable X can either be positive $X^p \sim p^p(x)$, negative $X^n \sim p^n(x)$ or unlabeled $X_u \sim p_u(x)$ *w.r.t.* joint densities $p^p(x) = P(x|Y=1)$, $p^n(x) = P(x|Y=0)$ and marginal density $p_u(x) = \pi_p p^p(x) + (1-\pi_p)p^n(x)$, where $\pi_p \in (0,1)$ describes the fraction of unknown positive examples in unlabeled data. Additionally, let D: $\mathbb{R}^m \to [0,1]$, and A: $\mathbb{R}^m \to [0,1]$ be the arbitrary decision functions and $\ell(\hat{y}, y)$ an arbitrary cost function such that ℓ: $[0,1] \times [0,1] \to \mathbb{R}$, where $\hat{y}$ and y denote predicted and true class labels, respectively. Lastly, we define $p_d(x)$ as $p^p(x) + p_u(x)$, and $p_g(x)$ as $G(z,y)$, where z is a straightforward prior distribution and $G(z,y) : z, y \mapsto x$ a mapping function, and $x_g^y \sim p_g^y(x)$.

Unlike GAN, where the main goal is to generate realistic-looking examples, we additionally want the examples to be separable as positive and negative. Thus, we would like auxiliary classifier to separate them with a large margin between the probability distributions of positive and negative classes. Note, the discriminator and the classifier optimize different constraints, thus in order for the generator to successfully converge, it has to generate data similar to $p_d(x)$, such that it satisfies

$$p_g(x) = p_d(x) = \pi_p p^p(x) + (1-\pi_p)p^n(x), \tag{2}$$

with the proof given in Appendix A. In order to simplify the Eq. (2), we use the following interpretation of $\pi_p p^p(x)$, which represents both labeled positive and unlabeled positive examples since the number of labeled examples represents the minority. The Eq. (2) denotes that the generated data resembles the input dataset, which is a mixture of positive and negative distributions.

CGenPU's objective is defined as a three-player game, combining GAN and $\mathcal{R}_{\mathcal{A}}$ objectives. Formally, it is written as follows:

$$\min_{G,A}\max_{D} V(A,G,D) = \min_{G,A}\max_{D} \Big\{ \mathbb{E}_{x\sim p_d(x)}[\ell(D(x),1)] + \\ + \mathbb{E}_{x\sim p_g(x,y)}[\ell(D(x),0)] + \lambda \mathcal{R}_{\mathcal{A}} \Big\}, \tag{3}$$

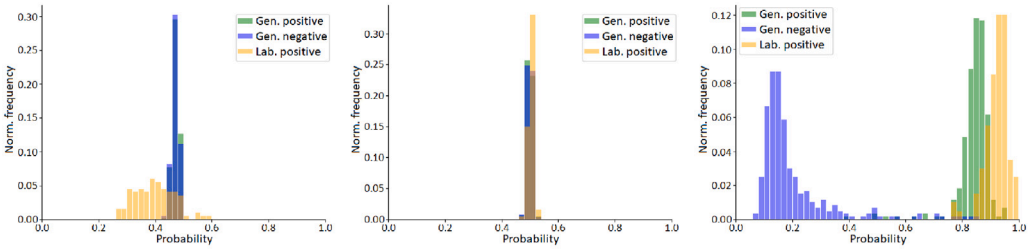
where λ is a constant that regulates the importance of classification. During the experimental evaluation, we set the parameter λ to 1, meaning that adversarial and auxiliary tasks are equally important. The objective function $\mathcal{R}_{\mathcal{A}}$ does not require prior knowledge about the data distribution, π_p, which simplifies the training procedure.

To achieve the desired goal, we propose an auxiliary objective $\mathcal{R}_{\mathcal{A}}$, which controls the similarity between positive and negative posterior distributions. We employ the Kullback–Leibler (KL) divergence as the similarity measure for this task (Kullback & Leibler, 1951). Intuitively, the classifier assigns the probability of being positive to each example in the batch. We also know that the labeled examples are positive and that artificial examples are generated from a *conditional* latent representation. Thus, we set constraints such that the distributions of labeled and generated positive examples are similar, and that the posterior distributions of generated positive and negative examples are dissimilar. To satisfy the given constraints, the similarity measure has to be non-negative and equal to zero if and only if the two distributions are equal. We define auxiliary objective function $\mathcal{R}_{\mathcal{A}}$ as follows:

$$\begin{aligned} \mathcal{R}_{\mathcal{A}} =& \mathbb{E}_{x^p\sim p^p(x), x_g^p\sim p_g^p(x)} \underbrace{[KL(A(x^p) \parallel A(x_g^p))]}_{\text{I}} \\ &+ \mathbb{E}_{x_g^p\sim p_g^p(x), x_g^n\sim p_g^n(x)} \underbrace{[KL(A(x_g^p) \parallel \widetilde{A}(x_g^n))]}_{\text{II}} \\ &+ \mathbb{E}_{x^p\sim p^p(x)} \underbrace{[KL(\mathbb{1} \parallel A(x^p))]}_{\text{III}} \end{aligned} \tag{4}$$

where the term marked by I minimizes the KL divergences of distributions $A(x^p)$ and $A(x_g^p)$, guiding the auxiliary classifier to give the same probability to the generated and labeled positive examples. This ensures that artificial positive examples come from $p^p(x)$. The term II minimizes the KL divergences of distributions $A(x_g^p)$ and $\widetilde{A}(x_g^n)$, guiding the auxiliary classifier to give different probabilities to artificial positive and negative examples. Consequently, $p_g^n(x)$ moves towards an unknown negative distribution $p^n(x)$. To achieve this goal, we have to measure the dissimilarity, which is done by computing the opposite distribution. Hu et al. (2021) compute it by subtracting classifiers prediction from probability one, *i.e.* $1 - A(x)$, which we denote as $\widetilde{A}(x)$. Finally, $\mathbb{1}$ is a constant distribution with all values equal to 1.

Terms I and II enable CGenPU to approximate $p^p(x)$ and $p^n(x)$. However, in practice, the auxiliary classifier is parameterized as a neural network, with an initial set of randomly selected weights. We show that terms I and II are not enough for successful convergence of the auxiliary classifier. Thus, we introduce the term III, which enables optimization of $\mathcal{R}_{\mathcal{A}}$. The study shows the effectiveness of term III (see Section 4.2).



a) After initialization b) Without term III c) With term III

Fig. 2. The distribution of probabilities after initialization, prior to introducing term III in $\mathcal{R}_{\mathcal{A}}$, and afterwards. The illustration shows that after adding the term III, the classifier starts to separate posterior probabilities and can afterward distinguish between positive and negative training examples.

4.2. Optimization of $\mathcal{R}_{\mathcal{A}}$

GANs are, unlike in theory, implemented as a series of neural networks, learning with the adversarial competition. Neural networks require an initial set of weights, which are randomly selected. Let us have an auxiliary classifier A, with *sigmoid* activation function on the output layer and a weight initialization, such that the inputs to the output activation function are distributed $N(0, \sigma_0)$. When applying the final activation, the probabilities will result in distribution $N(0.5, \sigma_1)$. Further, if we split the positive and the negative class probabilities, the resulting distributions are again $N(0.5, \sigma_1)$. The terms I and II in such a scenario return zero loss, resulting in zero gradients. Thus, the classifier will not learn. However, in practice, the initial distribution of the probabilities is not the same for labeled and generated positive examples. In this case, the classifier is going to adjust distributions so that the similarity is maximized. However, this causes the distribution of probabilities for positive examples having shape $N(0.5, 0)$ instead of $N(1, 0)$. The issue is related to term II and the way how we compute the opposite distribution. Computing $1 - A(x)$, where x comes from $N(0.5, \sigma_1)$, will result in the same distribution. Thus, minimizing the term II will result in probabilities for negative examples having shape $N(0.5, 0)$ instead of $N(0, 0)$.

We can mitigate this phenomenon by making the initial distribution of probabilities closer to zero or one. However, the convergence is still not guaranteed, *e.g.* the mode collapse (Thanh-Tung & Tran, 2020) may cause this phenomenon to appear. Luckily, we can eliminate this issue by providing an anchor for a positive class. This yields the term III, which guides the classifier to assign the probability equal to one for labeled positive examples. Term III can also be regarded as a predetermined label, $Y = 1$, for a positive class. Consequently, terms I and II follow, by moving probabilities towards one and zero for artificial positive and artificial negative examples, respectively. Assigning probability one to the positive class achieves smooth computation of $\mathcal{R}_{\mathcal{A}}$ by avoiding the occurrence of division by zero and the logarithm of zero in the KL divergence. Fig. 2 depicts the difference in the behavior before and after introducing the term III to the $\mathcal{R}_{\mathcal{A}}$.

4.3. Theoretical analysis

We now provide a formal theoretical analysis of CGenPU, which shows that the objective function leads to a unique, desirable equilibrium under a nonparametric assumption. For clarity of the paper, we refer to the proof details in Appendix A.

Lemma 1. *For G fixed, the output of the optimal auxiliary classifier A^* is:*

$$\forall x[x \in p^p \Leftrightarrow A^*(x) = 1] \wedge \forall x[x \in p^n \Leftrightarrow A^*(x) = 0].$$

Proof. Given G, the $\mathcal{R}_{\mathcal{A}}$ can be rewritten as

$$\int_x p^p(x) p_g^p(x) A(x^p) \log \frac{A(x^p)}{A(x_g^p)} dx +$$
$$+ \int_x p_g(x) A(x_g^p) \log \frac{A(x_g^p)}{1 - A(x_g^n)} dx +$$
$$+ \int_x p^p(x) \mathbb{1} \log \frac{\mathbb{1}}{A(x^p)} dx.$$

For any $(a, b, c) > 0$ and $(w, y, z) \in [0, 1]$, the function $f(w, y, z) = acy \log \frac{y}{z} + bz \log \frac{z}{1-w} + a \log \frac{1}{y}$ achieves its minimum at $f(0, 1, 1)$, concluding the proof. □

Theorem 1. *The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_d$, $p_g^p = p^p$ and $p_g^n = p^n$. At this point, $C(G)$ achieves the value $-\log 4$.*

Proof of Theorem 1. For $p_g = p_d$, $p_g^p = p^p$ and $p_g^n = p^n$, we derive a backward direction proof, where $C(G) = -\log 4$. To see that $-\log 4$ is not only a candidate but an actual global minimum, observe the forward direction proof.

$$C(G) = -\log 4 + \mathrm{KL}(p_d \| \frac{p_d + p_g}{2}) + \mathrm{KL}(p_g \| \frac{p_d + p_g}{2}) +$$
$$+ p^p p_g^p \mathrm{KL}(p_a^p \| p_{ga}^p) + p_g \mathrm{KL}(p_{ga}^p \| 1 - p_{ga}^n) + p^p \mathrm{KL}(\mathbb{1} \| p_a^p).$$

Knowing that $\mathrm{KL}(P \| Q) \geq 0$, $\mathrm{KL}(P \| Q) = 0 \iff P = Q$, and $\{p^p, p_g^p, p_g^n\} > 0$, the global minimum of virtual training criterion $C(G) = -\log 4$, concluding the proof. □

4.4. Training algorithm for CGenPU

Given the objective functions, we optimize the three networks alternately by the minibatch stochastic gradient descent (SGD). The expected input includes labeled positive and unlabeled examples. The output comprises a binary classifier, which can assign a positive or a negative label to yet unseen examples. We apply the non-saturating loss for better convergence (Goodfellow et al., 2014). Initially, we sample a batch of training examples from unlabeled and labeled data and generate artificial data using a generator. Next, we update parameters, starting with the discriminator and ending with the generator. Algorithm 1 shows the detailed training procedure.

5. Datasets and experimental evaluation

In this section, we first briefly present the datasets and experimental settings. Next, we show the classification and data quality results for various learning settings. The implementation of CGenPU is available at GitHub.[1]

[1] https://github.com/apapich/CGenPU.

Algorithm 1 Minibatch SGD training of CGenPU.

for $k = 1 \ldots$ #training iterations **do**

- Sample minibatch of m noise samples $\{z_{(1)}, \ldots, z_{(m)}\}$ from noise prior p_z and labels $\{y_{(1)}, \ldots, y_{(m)}\}$ from p_y.
- Sample minibatch of m training examples $\{x_{(1)}, \ldots, x_{(m)}\}$ from data distribution p_{data}.
- Sample minibatch of m labeled positive examples $\{x^p_{(1)}, \ldots, x^p_{(m)}\}$ from data distribution p^p.
- Update D by descending its stochastic gradient:

$$\nabla_{\theta^D} \frac{1}{m} \sum_{i=1}^{m} \left[-\log D(x_{(i)}) - \log(1 - D(G(z_{(i)}|y_{(i)}))) \right].$$

- Update A by descending its stochastic gradient:

$$\nabla_{\theta^C} \frac{\lambda}{m} \sum_{i=1}^{m} \left[-\log \frac{1}{A(x^p_{(i)})} - A(x^p_{(i)}) \log \frac{A(x^p_{(i)})}{A(G(z_{(i)}|y^p_{(i)}))} - A(G(z_{(i)}|y^p_{(i)})) \log \frac{A(G(z_{(i)}|y^p_{(i)}))}{1 - A(G(z_{(i)}|y^p_{(i)}))} \right].$$

- Update G by descending its stochastic gradient:

$$\nabla_{\theta^G} \frac{1}{m} \sum_{i=1}^{m} \left[-\log D(G(z_{(i)}|y_{(i)})) - \lambda A(x^p_{(i)}) \log \frac{A(x^p_{(i)})}{A(G(z_{(i)}|y^p_{(i)}))} - \lambda A(G(z_{(i)}|y^p_{(i)})) \log \frac{A(G(z_{(i)}|y^p_{(i)}))}{1 - A(G(z_{(i)}|y^n_{(i)}))} \right].$$

end for

5.1. Datasets and settings

We evaluate CGenPU using toy data and widely adopted handwritten digit dataset MNIST (LeCun, Cortes, & Burges, 2010) and computer-vision dataset used for object recognition CIFAR-10 (Krizhevsky, 2009). The toy data is synthesized of Gaussian mixtures, concentric circles, and half-moons, as shown in Fig. 3. The training data consist of 1000 unlabeled and 100 labeled examples, with $\pi_p = \frac{1}{2}$. Each dataset was randomly perturbed, using Gaussian noise with $\mu = 0$, and $\sigma = 0.1$. MNIST consists of handwritten digits from 0 to 9 with 60,000 training and 10,000 test examples. The digits are of size 28×28 pixels, encoded with black or white color representing background and foreground, respectively. The CIFAR-10 dataset consists of color images with ten mutually exclusive classes, each containing 6000 examples. The classes represent means of transport, such as airplanes, cars, ships and trucks, and different animals, like birds, cats, deer, dogs, frogs and horses. The dataset is split in 50,000 training and 10,000 test images, each of size $32 \times 32 \times 3$ pixels. We further split 10,000 training examples from MNIST and CIFAR-10 to validate the chosen model's parameters. Both MNIST and CIFAR-10 datasets are fully labeled; therefore, we transform them into a PU structure. First, the positive class is selected and sampled, such that we have N_p labeled positive examples. The unselected examples are used for unlabeled data with π_p positive and $1 - \pi_p$ negative examples. In One-vs-One setting, negative examples come from a predetermined class. We repeat the process for all possible class pairs. In One-vs-Rest setting, negative examples are sampled from the remaining classes in the dataset. Finally, we scale the images to the interval between -1 and 1 for all experiments to ensure better convergence.

We report the binary classification accuracy and assess the similarity of real and generated data based on clustering similarity (Robnik-Šikonja, 2018). For comparison of the proposed approach, we select existing GAN-based PU learning methods; PGAN (Chiaroni et al., 2018), D-GAN (Chiaroni et al., 2020) and GenPU (Hou et al., 2018). Because the implementations from corresponding authors are not publicly available, we implement them using Tensorflow,[2] to the best of our understanding. The generator and the discriminator architectures of PGAN, D-GAN, and CGenPU are the same (see Appendix B), while the architecture of GenPU is from Hou et al. (2018) due to convergence issues. We train each GAN once for every pair of positive and negative classes. Additionally, we run the classifier's validation on the test set five times, each time with a new set of random weights and samples of artificial training data. The clustering is performed on the embedded test and artificial data using openTSNE (Poličar, Stražar, & Zupan, 2019), which we pre-train on the training dataset. openTSNE is an implementation of t-Distributed Stochastic Neighbor Embedding (t-SNE), which incorporates the latest improvements, including adding new data points to existing embeddings. We exploit this feature to embed both test and artificial data in the latent space that describes the training dataset. This step is crucial to obtain accurate clusterings that we can compare. We measure the adjusted rand index (ARI), which describes the similarity of two joint clusterings. ARI close to one indicates high similarity and low when close to zero.

Additionally, we use a critical difference (CD) graph to compare the performance of multiple methods on different datasets (Demšar, 2006). The CD is a visualization of a statistical test, which considers only average ranks of methods using significance level $\alpha = 5\%$ and is robust to the numerical variations. This work ranks methods according to the calculated classification accuracies.

5.2. Results and analysis

5.2.1. Toy data

We begin our evaluation with toy examples for visual comparison of various GAN-based PU learning methods. Fig. 3 depicts the final convergence state of selected methods for various problems. All methods, in all scenarios, successfully converge to the respective true data distributions given a limited amount of labeled data. Based on the obtained visual results, we can distinguish two groups of the observed methods. The first group (PGAN and D-GAN) focuses on identifying unknown negative distribution. Such methods are useful in environments where labeling positive examples does not represent a challenge. However, the results indicate that PGAN cannot successfully recover the true negative distribution. In contrast, the second group (GenPU and *proposed* CGenPU) includes methods capable of identifying both positive and negative distributions. The evaluation confirms the properties summarized in Table 1.

5.2.2. MNIST

Results for the One-vs-One task. We report the results in Tables 2 and 3, which provide the classification accuracy and the generated data quality, respectively. The tables provide mean and standard deviation of five repetitions. The classification accuracy of PGAN, D-GAN, GenPU, and CGenPU is measured after the second learning stage on the MNIST dataset. Given the dataset characteristics (π_p, N_p), we evaluate classification accuracy for all digit pairs, which define the positive and negative classes. The accuracy is measured on the independent test data provided by the Tensorflow library. The details of the model's architecture, including parameters used in the evaluation, are given in Appendix B, Table B.7.

We observe that CGenPU outperforms existing methods, regardless of the structure of the PU data and the number of labeled examples. Among the existing methods, PGAN performs the worst, which is expected due to a lack of supervision in the training process. While D-GAN incorporates novel PU loss, the results are only marginally better than PGAN's. The baseline, GenPU, shows a comparable performance to D-GAN. However, when 50 or more labeled examples are available, the

[2] https://www.tensorflow.org/, version 2.4.0.

Table 2
The *classification accuracy* for the One-vs-One task using MNIST data. The column CGenPU-AC shows the accuracy of the auxiliary classifier. The dataset characteristics are available in the columns π_p and N_p, which vary throughout the evaluation. π_p describes the fraction of unknown positive examples in the unlabeled data, and N_p the number of labeled examples. Bolded cells indicate the best result.

π_p	N_p	PGAN	D-GAN	GenPU	*CGenPU*	*CGenPU-AC*
0.3	100	0.70 ± 0.23	0.81 ± 0.22	**0.87 ± 0.08**	0.82 ± 0.20	0.85 ± 0.18
	50	0.74 ± 0.21	0.78 ± 0.24	0.81 ± 0.13	0.83 ± 0.23	**0.88 ± 0.16**
	25	0.74 ± 0.20	0.81 ± 0.21	0.76 ± 0.15	0.76 ± 0.22	**0.82 ± 0.19**
	10	0.77 ± 0.18	0.76 ± 0.18	0.69 ± 0.17	0.75 ± 0.22	**0.82 ± 0.16**
	1	0.58 ± 0.10	0.60 ± 0.12	0.55 ± 0.11	0.69 ± 0.21	**0.70 ± 0.22**
0.5	100	0.69 ± 0.23	0.77 ± 0.22	0.86 ± 0.12	0.87 ± 0.19	**0.89 ± 0.18**
	50	0.67 ± 0.22	0.70 ± 0.24	0.80 ± 0.13	0.86 ± 0.22	**0.89 ± 0.20**
	25	0.70 ± 0.22	0.72 ± 0.22	0.71 ± 0.14	0.85 ± 0.18	**0.88 ± 0.17**
	10	0.60 ± 0.22	0.69 ± 0.23	0.59 ± 0.12	0.79 ± 0.21	**0.84 ± 0.18**
	1	0.57 ± 0.22	0.57 ± 0.17	0.59 ± 0.14	0.71 ± 0.25	**0.72 ± 0.23**
0.7	100	0.59 ± 0.17	0.57 ± 0.22	0.72 ± 0.18	0.89 ± 0.15	**0.91 ± 0.15**
	50	0.61 ± 0.21	0.60 ± 0.24	0.68 ± 0.18	0.86 ± 0.18	**0.87 ± 0.17**
	25	0.55 ± 0.19	0.57 ± 0.19	0.62 ± 0.16	0.85 ± 0.20	**0.86 ± 0.19**
	10	0.55 ± 0.16	0.57 ± 0.18	0.57 ± 0.11	0.81 ± 0.21	**0.84 ± 0.21**
	1	0.55 ± 0.11	0.56 ± 0.09	0.56 ± 0.11	**0.64 ± 0.23**	**0.64 ± 0.23**

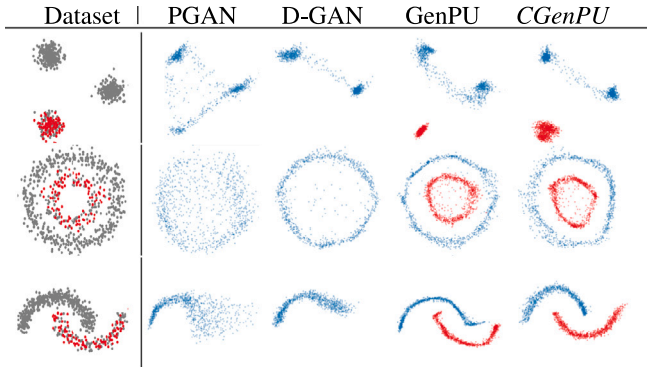


Fig. 3. The result of experiments on toy data shows a visual comparison between different GAN-based approaches. Each row represents a toy problem, and each column denotes one of the four methods. The Dataset column visualizes the training data, while individual cells show artificial positive and negative data.

obtained classifier has substantially higher accuracy, even by 10%. The results also show that the existing approaches suffer more from the lack of labeled examples than CGenPU, specifically when using the auxiliary classifier. The results imply that the proposed CGenPU is stable in extreme scenarios with ten or less labeled examples. Additionally, CGenPU achieves comparable classification performance for various distributions of unlabeled data, while existing methods obtain the lower accuracy when the positive class presents the majority ($\pi_p > 0.5$). Given that CGenPU does not need the prior data distribution knowledge (π_p), this represents an additional advantage of the approach. Further, Fig. C.8 shows the CD graph for the One-vs-One task on MNIST data, where our proposed method is significantly better than all other approaches. The best average ranking is achieved by CGenPU-AC followed by CGenPU, while there is no significant difference between GenPU and D-GAN. Lastly, PGAN achieves the worst ranking among all methods.

The data quality results show a superior performance of GenPU, which can produce realistic-looking data. However, GenPU is evaluated using a fully connected feedforward neural network due to convergence issues when introducing a convolutional architecture, which we assume it may be the reason for favorable results. Because, the remaining methods, including ours, use the architecture based on convolution the generated images contain less noise. As the fully connected feedforward architecture is easier to train, we assume that the choice of the architecture is the main reason for the GenPU's results. PGAN and D-GAN have ARI less than 0.5, which implies low similarity between the original and the artificial data. The proposed CGenPU approach

Table 3
The adjusted rand index for the One-vs-One task indicating the *quality of generated data* for MNIST dataset. The dataset characteristics are available in the columns π_p and N_p, which vary throughout the evaluation. π_p describes the fraction of unknown positive examples in the unlabeled data, and N_p the number of labeled examples. Bolded cells indicate the best result. We omit the *CGenPU-AC* column from this table since it uses the same generator as *CGenPU*.

π_p	N_p	PGAN	D-GAN	GenPU	*CGenPU*
0.3	100	0.60 ± 0.37	0.66 ± 0.37	**0.94 ± 0.13**	0.63 ± 0.45
	50	0.50 ± 0.37	0.43 ± 0.34	**0.93 ± 0.14**	0.68 ± 0.43
	25	0.40 ± 0.35	0.30 ± 0.27	**0.91 ± 0.16**	0.60 ± 0.45
	10	0.27 ± 0.25	0.31 ± 0.34	**0.80 ± 0.30**	0.53 ± 0.45
	1	0.24 ± 0.29	0.39 ± 0.36	**0.62 ± 0.33**	0.50 ± 0.44
0.5	100	0.52 ± 0.35	0.59 ± 0.37	**0.85 ± 0.21**	0.80 ± 0.37
	50	0.44 ± 0.38	0.51 ± 0.40	**0.88 ± 0.20**	0.81 ± 0.36
	25	0.37 ± 0.34	0.35 ± 0.32	**0.75 ± 0.32**	**0.75 ± 0.39**
	10	0.42 ± 0.39	0.28 ± 0.34	**0.62 ± 0.36**	0.54 ± 0.46
	1	0.40 ± 0.38	0.35 ± 0.40	**0.65 ± 0.32**	0.55 ± 0.47
0.7	100	0.43 ± 0.37	0.34 ± 0.40	**0.85 ± 0.28**	0.83 ± 0.33
	50	0.42 ± 0.37	0.34 ± 0.37	**0.83 ± 0.25**	0.71 ± 0.42
	25	0.41 ± 0.39	0.33 ± 0.36	0.70 ± 0.34	**0.76 ± 0.38**
	10	0.36 ± 0.38	0.33 ± 0.35	0.65 ± 0.36	**0.70 ± 0.42**
	1	0.31 ± 0.36	0.39 ± 0.36	**0.44 ± 0.38**	0.39 ± 0.44

produces similar data with an ARI = 0.5 or more. All methods produce the higher-quality data with more labeled examples available. D-GAN, GenPU and the proposed CGenPU use labeled data in the process of generators training, which affects the quality of the generated data. Greater quantity of the labeled data usually results in a more representative sample of the population. Given that PGAN is the only method that does not use labeled data while training the GAN, it makes the drop in data quality surprising.

Results for the One-vs-Rest task. The One-vs-Rest task is a binary classification problem, where the goal is to separate the class of interest from all the others. In our case, negative data represent examples from multiple classes. During the evaluation, a number of labeled examples was $N_p = 50$, and the architecture was the same as for the One-vs-One task (see Appendix B, Table B.7). Table 4 reports the classification accuracy of the auxiliary classifier trained with CGenPU for different unlabeled data distributions, *i.e.* different prior π_p. The results show high classification accuracy on test data, which we arranged such that the number of positive and negative examples in the test data is equal. Moreover, the results are comparable between various distributions of unlabeled data. Nonetheless, in the case of $\pi_p = 0.3$ and the positive class being 7, the generator failed to separate positive and negative distributions. Thus the classification accuracy is only 0.59. The issue is unclear since it was not recorded during the training. However, it is most likely related to the mode collapse.

Table 4
The *classification accuracy* of CGenPU-AC for the One-vs-Rest task on the MNIST data. The column denotes the selected positive class. We vary the number parameter π_p throughout the evaluation, which describes the fraction of unknown positive examples in the unlabeled data. $\bar{x}$ denotes the average result.

π_p	0	1	2	3	4	5	6	7	8	9	$\bar{x}$
0.3	0.95 ± 0.04	0.98 ± 0.02	0.79 ± 0.04	0.79 ± 0.03	0.89 ± 0.03	0.82 ± 0.02	0.97 ± 0.01	0.59 ± 0.05	0.81 ± 0.02	0.85 ± 0.03	0.84 ± 0.11
0.5	0.89 ± 0.01	0.97 ± 0.00	0.78 ± 0.03	0.78 ± 0.01	0.93 ± 0.00	0.80 ± 0.00	0.87 ± 0.01	0.93 ± 0.00	0.80 ± 0.01	0.83 ± 0.01	0.86 ± 0.07
0.7	0.99 ± 0.00	0.89 ± 0.00	0.95 ± 0.01	0.83 ± 0.06	0.91 ± 0.02	0.84 ± 0.04	0.96 ± 0.01	0.93 ± 0.04	0.80 ± 0.02	0.81 ± 0.01	0.89 ± 0.06
$\bar{x}$	0.94 ± 0.02	0.95 ± 0.01	0.84 ± 0.03	0.80 ± 0.03	0.91 ± 0.02	0.82 ± 0.02	0.93 ± 0.01	0.82 ± 0.03	0.81 ± 0.02	0.83 ± 0.02	

Table C.9, Table C.10 and Table C.11 show results for PGAN, D-GAN and GenPU, respectively. We see that the results for PGAN and D-GAN are similar and show low classification accuracy. In the PGAN method, we often observe that the distribution of artificial negative examples is not clean, especially when the π_p value is high. D-GAN has much less trouble with this and generates a clean distribution of negative examples when negative examples dominate the unlabeled data ($\pi_p \leq 0.5$). This problem contributes to lower classification accuracy, although it is not the leading cause. The biggest problem of PGAN and D-GAN is that they cannot generate positive examples, so it is impossible to sample an infinite number of negative examples to train the classifier. Since the distribution of negative examples is highly variable in this experiment, this further degrades the classification accuracy. We can also observe that the classification accuracy decreases as π_p increases, similar to the One-vs-One setting. Additionally, both PGAN and D-GAN fail to converge when the positive class represented the digit 8.

GenPU, on the other hand, achieves higher classification accuracy than PGAN and D-GAN. In most cases, the final classification accuracy exceeds 0.7. When accuracy is lower than 0.7, we observe signs of mode collapse or failed convergence, which is noticeable by the amount of noise on generated images. Further, the GenPU method shows that the classification accuracy decreases as the value of π_p increases, similar to the One-vs-One results. However, compared to CGenPU, all three approaches achieve lower classification accuracy. The CD graph for the One-vs-Rest task on MNIST data shows a significant difference between all methods (see Fig. C.9). CGenPU-AC achieves the best result again, followed by GenPU, D-GAN, and PGAN, respectively.

5.2.3. CIFAR-10

We repeat the evaluation of the proposed CGenPU and existing GAN-based methods for PU learning on the CIFAR-10 dataset. CIFAR-10 is considered challenging because of the distribution of data that resembles a real-world image dataset. To reduce the computational load, we set the prior distribution $\pi_p = 0.5$, and selected classes *car* and *ship* as positive and negative, respectively. Due to their foreground similarity, we focus exclusively on pair of classes *car* and *ship*. This increase the task difficulty and the time complexity for training and evaluation. Compared to Section 5.2.2, we do not compare all possible pairs because of the time required for the evaluation. Additionally, we measure the classification accuracy depending on different amounts of labeled examples. The details of the model's architecture, including parameters used in the evaluation, are available in Appendix B, Table B.8.

Fig. 4 depicts the artificial positive (*car*) and artificial negative (*ship*) examples generated by CGenPU. The red rectangles indicate out-of-class-distribution examples, *i.e.*, false positives and false negatives. We can observe that with less labeled examples the false positive and the false negative rates increase. With at least 25 labeled examples, the distributions look clean. The significance of the results comes from the fact that two corresponding positive and negative examples share the same latent vector z and have different labels y. This implies that the vector z only encodes features of the object, but not the object itself. In other words, CGenPU learns a hidden, latent representation for z that is independent of the class label y, similarly as AC-GAN (Odena et al., 2017).

In Table 5 we report the classification accuracy. The GenPU fails to successfully converge and learn both data distributions, which shows

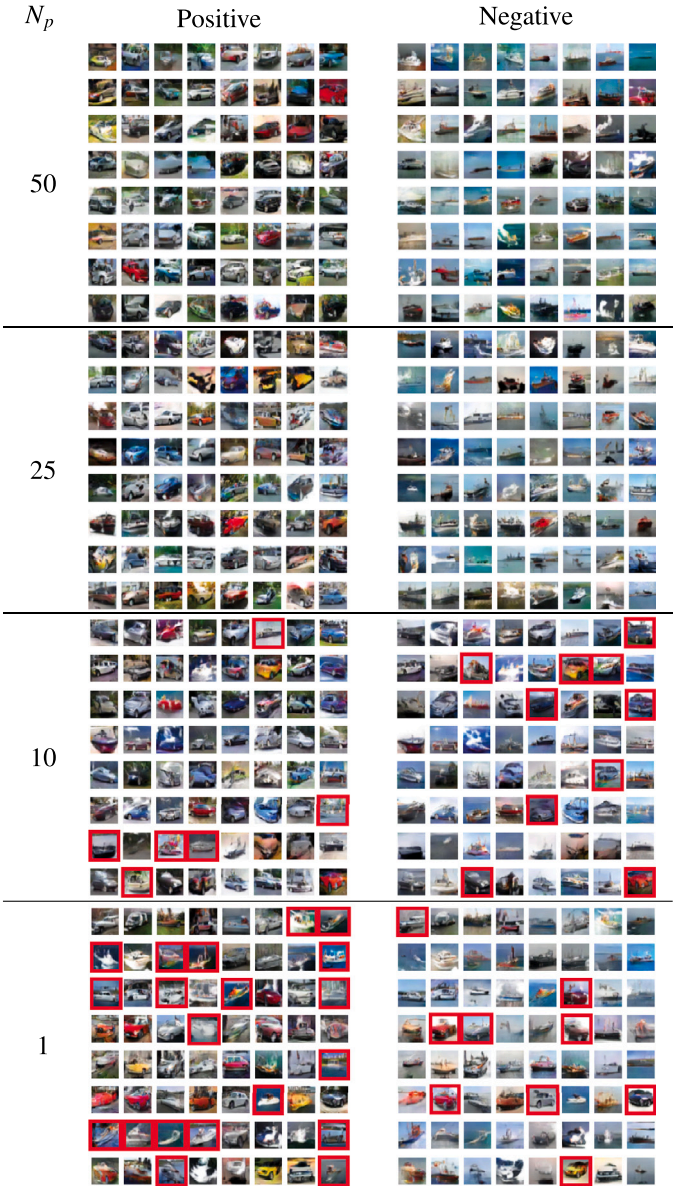


Fig. 4. A visualization of artificial positive (*car*) and artificial negative (*ship*) images generated by CGenPU. N_p indicates the number of labeled training examples. Red boxes indicate out-of-class-distribution examples.

the instability problem on complex data (Chiaroni et al., 2020). Both PGAN and D-GAN do not have a convergence problem. However, to train a reliable classifier, we need more than 50 labeled examples. Chiaroni et al. (2020) mention a required number of at least 100 labeled examples. Our approach obtains a classification accuracy of 0.84 with only ten labeled examples, indicating the significant advantage of the proposed approach. When only one labeled example is available, none of the approaches, including CGenPU, can converge. However, this is an extreme scenario that does not describe the distribution of positive

Table 5
The *classification accuracy* for car-vs-ship task on the CIFAR-10 data. We vary the number parameter N_p throughout the evaluation, which describes the number of labeled examples. Bolded cells indicate the best result.

N_p	PGAN	D-GAN	GenPU	*CGenPU*	*CGenPU-AC*
50	0.62 ± 0.02	0.70 ± 0.03	0.50 ± 0.01	0.70 ± 0.02	**0.80 ± 0.01**
25	0.66 ± 0.02	0.63 ± 0.02	0.53 ± 0.02	0.72 ± 0.02	**0.81 ± 0.02**
10	0.58 ± 0.02	0.54 ± 0.01	0.46 ± 0.07	0.73 ± 0.02	**0.84 ± 0.02**
1	**0.50 ± 0.00**	**0.50 ± 0.01**	0.48 ± 0.01	0.35 ± 0.02	0.34 ± 0.05

Table 6
The study of the importance of the auxiliary weight in the CGenPU framework. We vary the number parameter λ throughout the evaluation, which describes the influence of the auxiliary classifier and measure the classification accuracy. Bolded cell indicates the best result.

λ	0.001	0.1	1	10	1000
ACC	0.50 ± 0.07	**0.92 ± 0.01**	0.87 ± 0.02	0.66 ± 0.02	0.52 ± 0.02

data. The results visualized in the CD graph show three groups of methods that differ significantly (see Fig. C.10). The first group consists of CGenPU-AC and CGenPU, with no significant difference. The second group contains CGenPU, PGAN, and D-GAN, while the third group contains only the GenPU method. The statistical analysis proved again that our proposed CGenPU is significantly better than other approaches.

Table 5 also shows that CGenPU achieves higher classification performance when ten labeled examples are available compared to fifty. Further analysis shows signs of mode collapse when having fifty labeled examples, which could have contributed to lower classification accuracy. We suspect the same happened in other cases, except they appear between the two saved states. Another reason may be that the sample of labeled examples, which was randomly selected, affected the sample representativeness.

5.3. Influence of the auxiliary weight λ

The role of the auxiliary weight λ is to balance the importance between generating realistic data and classification performance, which influences identifying positive and negative data distributions. AC-GAN gives equal importance to both tasks, which is reasonable because the data is labeled (Odena et al., 2017). CGenPU does not have labeled data except for a few positive examples, making learning less supervised and more prone to failure.

We provide a systematical study, which shows the effect of λ on CGenPU's learning ability. Table 6 shows the classification accuracy for 3 and 5 digits pair, while $\pi_p = 0.5$. Simply setting λ to a small value, *e.g.* $\lambda = 0.001$, leads the generator to generate realistic data, but the distributions of positive and negative data will overlap. Consequently, the classification accuracy is low. The opposite setting of λ to a high value, *e.g.* $\lambda = 1000$, prevents the generator from learning the input data distribution, again leading to low classification accuracy. When both tasks are somewhat equally important, *e.g.* $\lambda \sim 1$, we can expect the generator to provide artificial data that looks realistic and with clean distributions. Fig. 5 depicts the behavior of CGenPU for various choices of λ.

Given our experience during the development of the CGenPU framework, the suitable value for λ lies on the interval $[0.1, 1]$. However, the most appropriate weight is domain-dependent. The rule of thumb for optimal selecting the value would be to start with $\lambda = 1$, gradually decreasing it, until the generated data look realistic with clean positive and negative distributions. Usually, giving more importance to the discriminator by reducing λ increases the convergence speed.

5.4. Comparison with the GenPU

The proposed CGenPU approach uses a fundamentally different structure than the existing GenPU framework. We build CGenPU on top

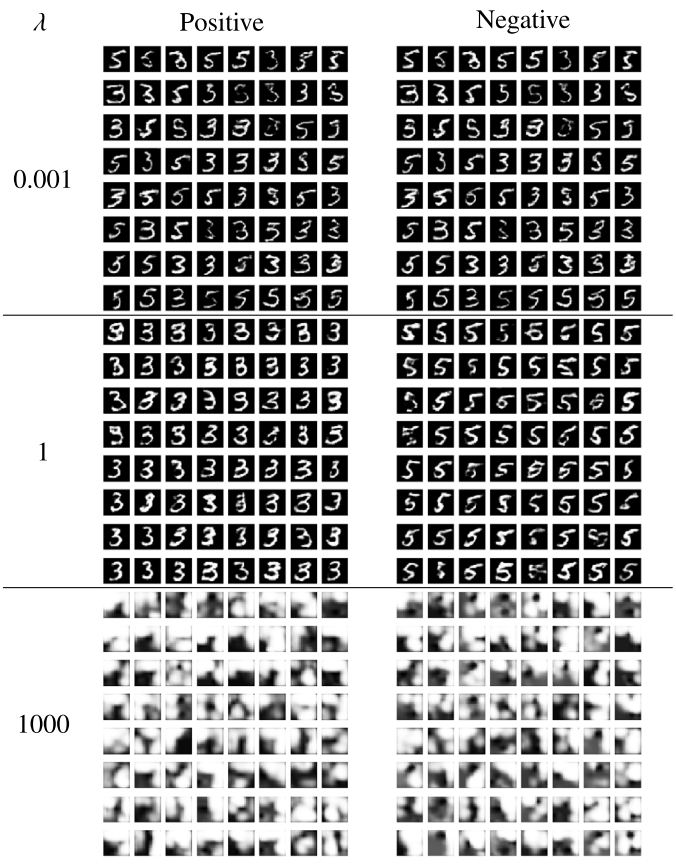


Fig. 5. A visualization of artificial positive (*3*) and artificial negative (*5*) digits generated by CGenPU. λ indicates the auxiliary classifier's weight.

of the AC-GAN, which through adversarial learning, trains three neural networks instead of GenPU's five. We perform a study where we compare both frameworks' computational performance and convergence speed to see the effect of structural modification.

Computational performance. We design the experiment to estimate the number of batch updates per second using feed-forward (FNN) and convolutional neural network (CNN) architectures. The estimation takes into account three stages: forward propagation, loss computation, and backward propagation. Additionally, for this study, we initialize all models with an equal number of parameters ranging from 10^5 to 10^7. Fig. 6 depicts the normalized number of iterations (batch updates) per second. We plot both the mean and the standard error.

The results show that CGenPU performs more batch updates in a second than GenPU using feed-forward and convolutional architectures. The speed difference reflects in the number of neural networks built within the frameworks, which relates to the number of parameters that need to be updated. On the average, CGenPU performs between 0.15 and 0.20 updates per second more than GenPU. The exception is an experiment with CNN architecture which contains 10^7 parameters, where CGenPU performed roughly 0.45 iterations more. This difference occurred because the GenPU framework did not fit into the GPU's memory anymore, while CGenPU had no such problem. The experiment shows that CGenPU's training speed decreases less rapidly with larger models, usually required for complex datasets.

Convergence speed. With this experiment, we measure the changes in classification accuracy during the training phase. To ensure a fair comparison, we take the neural network architectures and hyperparameters from the GenPU paper (Hou et al., 2018). The CGenPU's discriminator architecture is the same as the GenPU's discriminator of unlabeled data. At the same time, we establish the auxiliary classifier

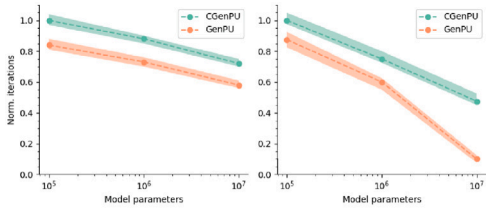


Fig. 6. A visualization of the mean value of the normalized number of iterations with the standard error for various values of model parameters.

on the architecture of the discriminator of positive/negative data. We set the auxiliary weight λ to 1. In Fig. 7, we show the classification accuracy for every five epochs on 3 and 5 digits pair from the MNIST dataset and various data distributions π_p. We plot both the mean and the standard deviation of five repetitions.

The results show that the CGenPU converges faster towards the highest classification accuracy than the GenPU, while both achieve a similar result at the end of training. In the first 20 epochs, GenPU achieves extremely low accuracy compared to CGenPU, which reaches its best accuracy already at the tenth epoch. This means that in case of a limited training time, CGenPU would result in significantly higher classification accuracy. The results confirm that the proposed three-player minimax game is more efficient than the five-player minimax game. Additionally, we conclude that the distribution of the data does not have a significant effect on the convergence speed.

6. Discussion

The proposed CGenPU is based on the auxiliary loss, which relies on unlabeled data to contain both positive and negative examples. The definition of PU learning is built on the assumption that unlabeled data is a mixture of positive and negative data distributions, which CGenPU relies upon as well. However, we question whether there is a lower bound for π_p, where CGenPU fails to converge or experiences a significant accuracy drop. This work already depicts the behavior for various distributions of unlabeled data. The future work shall provide the theoretical analysis of the lower bound of π_p.

A limitation not directly observed in the experimental evaluation is related to the KL divergence, which is non-negative and has no upper bound. In the case of mode collapse, the distribution of probabilities from the auxiliary classifier collapses, resulting in a spike of auxiliary loss value. In such cases, the discriminator's response can become negligible, preventing the generator from recovering quickly. Thus, future research shall address this problem by dynamically adjusting the classifier's importance or using a different bounded similarity measure.

7. Conclusion

This paper presents a novel CGenPU framework for PU learning. By building upon the AC-GAN's architecture we formulate a novel PU auxiliary objective function that enables learning from the PU data, making the proposed approach the first single-stage GAN-based framework. With the CGenPU, we address the fundamental shortcomings of existing methods, such as computational constraints, the requirement to know the prior distribution of unlabeled data, and the inability to generate positive examples. The experimental evaluation demonstrates the CGenPU's effectiveness on complex data, such as CIFAR-10, even with a small set of labeled examples. Moreover, CGenPU successfully converges using various batch sizes, confirming the training stability using SGD. The results show a potential to alleviate the human experts burdening with a tedious labeling process. In the future, we shall apply the CGenPU on various domains from Section 1 to show its real-world applicability and address limitations presented in Section 6.

Note that we evaluated CGenPU on images, but it is not limited to such datasets. CGenPU can be applied to any dataset as long as the underlying architecture of neural networks fits the data and allows convergence. Thus, CGenPU is applicable to various domains mentioned in Section 1, specifically for its ability to converge with a few labeled examples. This makes it ideal for tasks with limited labeled data, where identifying negative examples from unlabeled data is not enough to obtain a reliable classifier. CGenPU allows us to learn both positive and negative data distributions and enables us to increase the size of the dataset with examples from both classes, similarly to GenPU. Contrary to GenPU, it does not need prior information about unlabeled data distribution, which is a significant advantage.

This paper combines several advantages of existing GAN-based PU approaches to tackle their individual limitations. Our work considers a promising aspect of developing single-stage algorithms, which is of great importance for the machine learning community.

CRediT authorship contribution statement

Aleš Papič: Conceptualization, Methodology, Software, Formal analysis, Investigation, Visualization, Writing – original draft. **Igor Kononenko:** Supervision, Writing – review & editing. **Zoran Bosnić:** Supervision, Funding acquisition, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The code is available in the GitHub repository: https://github.com/apapich/CGenPU. The data used for the experimental analysis is publicly available and cited in the paper.

Acknowledgments

This work is supported by the Slovene Research Agency: A Junior Researcher Grant (53630) awarded to Aleš Papič, and the research programme P2-0209.

Appendix A. Detailed theoretical analysis

Lemma 1. For G fixed, the output of the optimal auxiliary classifier A^* is:

$$\forall x[x \in p^p \Leftrightarrow A^*(x) = 1] \wedge \forall x[x \in p^n \Leftrightarrow A^*(x) = 0].$$

Proof. Given G, the $\mathcal{R}_{\mathcal{A}}$ can be rewritten as

$$\begin{aligned}
&\int_x p^p(x) p_g^p(x) \mathrm{KL}(A(x^p) \parallel A(x_g^p)) dx + \\
&+ \int_x p_g(x) \mathrm{KL}(A(x_g^p) \parallel 1 - A(x_g^n)) dx + \\
&+ \int_x p^p(x) \mathrm{KL}(\mathbb{1} \parallel A(x^p)) dx \\
&= \int_x p^p(x) p_g^p(x) A(x^p) \log \frac{A(x^p)}{A(x_g^p)} dx + \\
&+ \int_x p_g(x) A(x_g^p) \log \frac{A(x_g^p)}{1 - A(x_g^n)} dx + \\
&+ \int_x p^p(x) 1 \log \frac{1}{A(x^p)} dx.
\end{aligned}$$

For any $(a, b, c) > 0$ and $(w, y, z) \in [0, 1]$, the function $f(w, y, z) = acy \log \frac{y}{z} + bz \log \frac{z}{1-w} + a \log \frac{1}{y}$ achieves its minimum at $f(0, 1, 1)$, concluding the proof. □

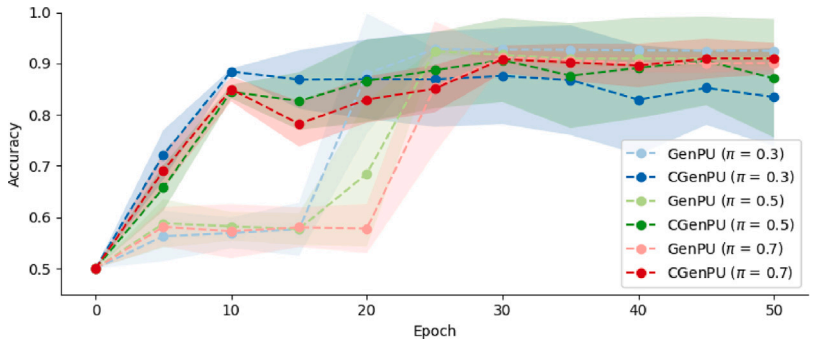


Fig. 7. A visualization of mean classification accuracy with the standard deviation on 3 and 5 digits pair from the MNIST dataset. The π_p indicates the data distribution.

Table B.7
GAN model architecture used for experiments on the MNIST dataset. The table consists of two parts, the first being the model's architecture and the second the learning parameters. Auxiliary classifier (A) and discriminator (D) shared the first two layers of weights.

Model	Operation	Kernel	Strides	Units/Filters	Nonlinearity
$G_x(z, y)$	$100 \times 1 \times 1$, 1 (input)				
	Linear			128	ReLU
	Conv2D	3×3	1×1	8	ReLU
	UpSampling2D				
	Conv2D	3×3	1×1	8	ReLU
	UpSampling2D				
	Conv2D	3×3	1×1	16	ReLU
	UpSampling2D				
	Conv2D	3×3	1×1	1	Tanh
$D(x)/A(x)$	$28 \times 28 \times 1$ (input)				
	Linear			512	Leaky ReLU
	Linear			512	Leaky ReLU
$A(x)$ *(only)*	Linear			128	Leaky ReLU
	Linear			2	Sigmoid
	Optimizer	Adam(α = 1e−3, β_1 = 0.5, β_2 = 0.999)			
	Batch size	32			
	Leaky ReLU slope	0.2			
	Auxiliary weight λ	1			
	Epochs	100			
	Weight, bias initialization	Xavier, Constant 0			
	Weight sharing	YES			

Table B.8
GAN model architecture used for experiments on the CIFAR-10 dataset. The table consists of two parts, the first being the model's architecture and the second the learning parameters. Auxiliary classifier (A) and discriminator (D) shared the first four layers of weights.

Model	Operation	Kernel	Strides	Units/Filters	Nonlinearity
$G_x(z, y)$	$100 \times 1 \times 1$, 1 (input)				
	Linear			$4 \times 4 \times 256$	Leaky ReLU
	Conv2DTranspose	4×4	2×2	128	Leaky ReLU
	Conv2DTranspose	4×4	2×2	128	Leaky ReLU
	Conv2DTranspose	4×4	2×2	128	Leaky ReLU
	Conv2D	3×3	1×1	3	Tanh
$D(x)/A(x)$	$32 \times 32 \times 3$ (input)				
	Conv2D	3×3	1×1	64	Leaky ReLU
	Conv2D	3×3	2×2	128	Leaky ReLU
	Conv2D	3×3	2×2	128	Leaky ReLU
	Conv2D	3×3	2×2	256	Leaky ReLU
$A(x)$ *(only)*	Linear			256	Leaky ReLU
	Linear			2	Sigmoid
	Optimizer	Adam(α = 1e−3, β_1 = 0.5, β_2 = 0.999)			
	Batch size	128			
	Leaky ReLU slope	0.2			
	Auxiliary weight λ	1			
	Epochs	500			
	Weight, bias initialization	Xavier, Constant 0			
	Weight sharing	YES			

Table C.9
The *classification accuracy* using generated data by *PGAN* for the One-vs-Rest task on the MNIST data. The column denotes the selected positive class. We vary the number parameter π_p throughout the evaluation, which describes the fraction of unknown positive examples in the unlabeled data. $\bar{x}$ denotes the average result.

π_p	0	1	2	3	4	5	6	7	8	9	$\bar{x}$
0.3	0.75 ± 0.02	0.69 ± 0.03	0.63 ± 0.03	0.66 ± 0.03	0.64 ± 0.03	0.62 ± 0.04	0.68 ± 0.01	0.64 ± 0.02	0.59 ± 0.00	0.64 ± 0.01	0.65 ± 0.02
0.5	0.72 ± 0.02	0.63 ± 0.06	0.65 ± 0.02	0.64 ± 0.04	0.66 ± 0.02	0.62 ± 0.03	0.66 ± 0.03	0.63 ± 0.02	0.56 ± 0.01	0.64 ± 0.03	0.64 ± 0.03
0.7	0.70 ± 0.03	0.61 ± 0.03	0.59 ± 0.01	0.60 ± 0.00	0.62 ± 0.01	0.57 ± 0.02	0.61 ± 0.03	0.61 ± 0.02	0.57 ± 0.00	0.60 ± 0.06	0.61 ± 0.02
$\bar{x}$	0.72 ± 0.02	0.64 ± 0.04	0.62 ± 0.02	0.64 ± 0.02	0.64 ± 0.02	0.60 ± 0.03	0.65 ± 0.02	0.63 ± 0.02	0.57 ± 0.01	0.63 ± 0.03	

Table C.10
The *classification accuracy* using generated data by *D-GAN* for the One-vs-Rest task on the MNIST data. The column denotes the selected positive class. We vary the number parameter π_p throughout the evaluation, which describes the fraction of unknown positive examples in the unlabeled data. $\bar{x}$ denotes the average result.

π_p	0	1	2	3	4	5	6	7	8	9	$\bar{x}$
0.3	0.74 ± 0.03	0.72 ± 0.05	0.62 ± 0.04	0.69 ± 0.06	0.66 ± 0.03	0.58 ± 0.07	0.68 ± 0.03	0.66 ± 0.05	0.57 ± 0.05	0.68 ± 0.04	0.66 ± 0.05
0.5	0.73 ± 0.01	0.66 ± 0.08	0.62 ± 0.02	0.69 ± 0.02	0.65 ± 0.02	0.57 ± 0.05	0.70 ± 0.02	0.63 ± 0.03	0.57 ± 0.06	0.67 ± 0.03	0.65 ± 0.03
0.7	0.74 ± 0.03	0.66 ± 0.07	0.60 ± 0.05	0.60 ± 0.07	0.63 ± 0.05	0.59 ± 0.05	0.64 ± 0.03	0.63 ± 0.02	0.56 ± 0.04	0.66 ± 0.04	0.63 ± 0.05
$\bar{x}$	0.74 ± 0.02	0.68 ± 0.06	0.61 ± 0.04	0.66 ± 0.05	0.65 ± 0.03	0.57 ± 0.06	0.67 ± 0.02	0.64 ± 0.03	0.57 ± 0.05	0.67 ± 0.04	

Table C.11
The *classification accuracy* using generated data by *GenPU* for the One-vs-Rest task on the MNIST data. The column denotes the selected positive class. We vary the number parameter π_p throughout the evaluation, which describes the fraction of unknown positive examples in the unlabeled data. $\bar{x}$ denotes the average result.

π_p	0	1	2	3	4	5	6	7	8	9	$\bar{x}$
0.3	0.88 ± 0.02	0.73 ± 0.02	0.80 ± 0.03	0.71 ± 0.02	0.78 ± 0.06	0.76 ± 0.04	0.84 ± 0.02	0.73 ± 0.02	0.75 ± 0.05	0.67 ± 0.04	0.77 ± 0.03
0.5	0.86 ± 0.03	0.56 ± 0.06	0.78 ± 0.04	0.67 ± 0.03	0.75 ± 0.05	0.74 ± 0.05	0.80 ± 0.04	0.78 ± 0.04	0.68 ± 0.04	0.71 ± 0.01	0.73 ± 0.04
0.7	0.85 ± 0.01	0.62 ± 0.05	0.72 ± 0.05	0.71 ± 0.02	0.70 ± 0.05	0.67 ± 0.01	0.79 ± 0.06	0.79 ± 0.05	0.68 ± 0.05	0.73 ± 0.03	0.73 ± 0.04
$\bar{x}$	0.86 ± 0.02	0.64 ± 0.04	0.77 ± 0.04	0.70 ± 0.02	0.74 ± 0.05	0.72 ± 0.03	0.81 ± 0.04	0.77 ± 0.04	0.70 ± 0.05	0.70 ± 0.03	

Theorem 1. The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_d$, $p_g^p = p^p$ and $p_g^n = p^n$. At this point, $C(G)$ achieves the value $-\log 4$.

Proof of Theorem 1. For $p_g = p_d$, $p_g^p = p^p$ and $p_g^n = p^n$, we derive a backward direction proof, where $C(G) = -\log 4$.

$$
\begin{aligned}
&C(G) = \max_D V(G,D) + \min_A V(G,A) = \\
&= \int_x p_d(x) \log \frac{1}{2} + p_g(x) \log(1 - \frac{1}{2}) dx + \\
&+ \int_x p^p(x) p_g^p(x) 1 \log \frac{1}{1} dx + \\
&+ \int_x p_g(x) 1 \log \frac{1}{1-0} dx + \\
&+ \int_x p^p(x) 1 \log \frac{1}{1} dx = -\log 4
\end{aligned}
$$

To see that $-\log 4$ is not only candidate but rather actual global minimum, observe the forward direction proof.

$$
\begin{aligned}
&C(G) = \max_D V(G,D) + \min_A V(G,A) + \\
&= \int_x p_d(x) \log \frac{p_d(x)}{p_d(x) + p_g(x)} + p_g(x) \log \frac{p_g(x)}{p_d(x) + p_g(x)} dx + \\
&+ \int_x p^p(x) p_g^p(x) \mathrm{KL}(p^p(x) \parallel p_g^p(x)) dx + \\
&+ \int_x p_g(x) \mathrm{KL}(p_g^p(x) \parallel 1 - p_g^n(x)) dx + \\
&+ \int_x p^p(x) \mathrm{KL}(\mathbb{1} \parallel p^p(x)) dx = \\
&\overset{[*]}{=} -\log 4 + \mathrm{KL}(p_d \| \frac{p_d + p_g}{2}) + \mathrm{KL}(p_g \| \frac{p_d + p_g}{2}) + \\
&+ p^p p_g^p \mathrm{KL}(p_a^p \| p_{ga}^p) + p_g \mathrm{KL}(p_{ga}^p \parallel 1 - p_{ga}^n) + p^p \mathrm{KL}(\mathbb{1} \| p_a^p)
\end{aligned}
$$

Note that to avoid confusion in $[*]$*, we introduce notation of* p_{ga}^p *and* p_{ga}^n *instead of* p_a^p *and* p_a^n*, to emphasize that the probabilities are dependent from the* G*.*

Knowing that

- $\mathrm{KL}(P \parallel Q) \geq 0$, and $\mathrm{KL}(P \parallel Q) = 0 \iff P = Q$,
- $p^p > 0$, since by the definition of PU learning, we have at least one labeled positive example, and
- $\{p_g^p, p_g^n\} > 0$, since by the definition of PU learning, we have unlabeled data $p_u = \pi_p p^p + (1 - \pi_p) p^n$, where $\pi_p \in (0, 1)$,

$-\log 4$ is the global minimum of virtual training criterion $C(G)$, and $p_{ga}^p = 1 - p_{ga}^n \iff p_g^n = p^n$. Thus G^* is a single unique solution, which concludes the proof. □

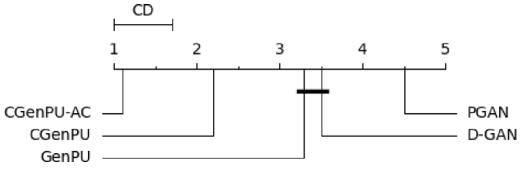


Fig. C.8. *Critical difference* (CD) graph for One-vs-One task on the MNIST data computed using average rankings of methods and 0.05 significance level.

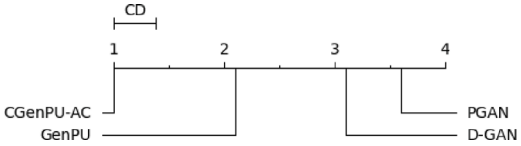


Fig. C.9. *Critical difference* (CD) graph for One-vs-Rest task on the MNIST data computed using average rankings of methods and 0.05 significance level.

Appendix B. Hyperparameters

See Tables B.7 and B.8.

Appendix C. Additional experiments

See Figs. C.8, C.9 and C.10 and Tables C.9, C.10 and C.11.

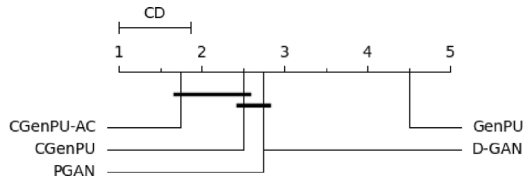


Fig. C.10. *Critical difference* (CD) graph for car-vs-ship task on the CIFAR-10 data computed using average rankings of methods and 0.05 significance level.

References

Bekker, J., & Davis, J. (2020). Learning from positive and unlabeled data: A survey. *Machine Learning*, *109*(4), 719–760.

Chiaroni, F., Khodabandelou, G., Rahal, M.-C., Hueber, N., & Dufaux, F. (2020). Counter-examples generation from a positive unlabeled image dataset. *Pattern Recognition*, *107*, Article 107527.

Chiaroni, F., Rahal, M.-C., Hueber, N., & Dufaux, F. (2018). Learning with a generative adversarial network from a positive unlabeled dataset for image classification. In *IEEE ICIP* (pp. 1368–1372). IEEE.

Cho, Y., Kim, D., Khan, M. A., & Choo, J. (2022). Mining multi-label samples from single positive labels. In *Advances in neural information processing systems*.

Christoffel, M., Niu, G., & Sugiyama, M. (2016). Class-prior estimation for learning from positive and unlabeled data. In *Asian conference on machine learning* (pp. 221–236). PMLR.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1–30.

Du Plessis, M. C., Niu, G., & Sugiyama, M. (2014). Analysis of learning from positive and unlabeled data. In *NIPS* (pp. 703–711).

Du Plessis, M., Niu, G., & Sugiyama, M. (2015). Convex formulation for learning from positive and unlabeled data. In *ICML* (pp. 1386–1394).

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., et al. (2014). Generative adversarial nets. In *NIPS* (pp. 2672–2680).

Hou, M., Chaib-Draa, B., Li, C., & Zhao, Q. (2018). Generative adversarial positive-unlabeled learning. In *IJCAI* (pp. 2255–2261). AAAI Press.

Hu, W., Le, R., Liu, B., Ji, F., Ma, J., Zhao, D., et al. (2021). Predictive adversarial learning from positive and unlabeled data. In *Proceedings of the AAAI conference on artificial intelligence, Vol. 35* (pp. 7806–7814).

Jaskie, K., & Spanias, A. (2019). Positive and unlabeled learning algorithms and applications: A survey. In *IISA* (pp. 1–8). IEEE.

Kiryo, R., Niu, G., Du Plessis, M. C., & Sugiyama, M. (2017). Positive-unlabeled learning with non-negative risk estimator. In *NIPS* (pp. 1675–1685).

Krizhevsky, A. (2009). *Learning multiple layers of features from tiny images*. University of Toronto.

Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, *22*(1), 79–86.

LeCun, Y., Cortes, C., & Burges, C. (2010). MNIST handwritten digit database. (p. 2). ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist.

Li, X., & Liu, B. (2003). Learning to classify texts using positive and unlabeled data. In *IJCAI, Vol. 3* (pp. 587–592).

Liu, B., Lee, W. S., Yu, P. S., & Li, X. (2002). Partially supervised classification of text documents. In *ICML, Vol. 2* (pp. 387–394).

Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784.

Na, B., Kim, H., Song, K., Joo, W., Kim, Y.-Y., & Moon, I.-C. (2020). Deep generative positive-unlabeled learning under selection bias. In *Proceedings of the 29th ACM international conference on information & knowledge management* (pp. 1155–1164).

Northcutt, C. G., Wu, T., & Chuang, I. L. (2017). Learning with confident examples: Rank pruning for robust classification with noisy labels. In *Proceedings of the thirty-third conference on uncertainty in artificial intelligence* (p. 10). Sydney, Australia: AUAI Press, Retrieved from http://auai.org/uai2017/proceedings/papers/35.pdf.

Odena, A., Olah, C., & Shlens, J. (2017). Conditional image synthesis with auxiliary classifier gans. In *ICML* (pp. 2642–2651).

Poličar, P. G., Stražar, M., & Zupan, B. (2019). openTSNE: a modular python library for t-SNE dimensionality reduction and embedding. http://dx.doi.org/10.1101/731877, bioRxiv.

Ren, Z., Li, Q., Cao, K., Li, M. M., Zhou, Y., & Wang, K. (2023). Model performance and interpretability of semi-supervised generative adversarial networks to predict oncogenic variants with unlabeled data. *BMC Bioinformatics*, *24*(1), 1–16.

Robnik-Šikonja, M. (2018). Dataset comparison workflows. *International Journal of Data Science*, *3*(2), 126–145.

Thanh-Tung, H., & Tran, T. (2020). Catastrophic forgetting and mode collapse in GANs. In *2020 international joint conference on neural networks* (pp. 1–10). IEEE.

Xu, S., Qi, M., Wang, X., Zhao, H., Hu, Z., & Sun, H. (2022). A positive-unlabeled generative adversarial network for super-resolution image reconstruction using a charbonnier loss.. *Traitement du Signal*, *39*(3).

Yang, P., Liu, W., & Yang, J. (2017). Positive unlabeled learning via wrapper-based adaptive sampling. In *Proceedings of the 26th international joint conference on artificial intelligence* (pp. 3273–3279).

Zhou, Y., Xu, J., Wu, J., Taghavi, Z., Korpeoglu, E., Achan, K., et al. (2021). Pure: Positive-unlabeled recommendation with generative adversarial network. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining* (pp. 2409–2419).