D-GAN: DIVERGENT GENERATIVE ADVERSARIAL APPROACH FOR POSITIVE-UNLABELED LEARNING AND COUNTER-EXAMPLES GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Positive Unlabeled learning consists in learning to distinguish samples of our class of interest, the positive class, from the counter-examples, the negative class, by using positive labeled and unlabeled samples during the training. Recent approaches suggest to exploit the GANs (Goodfellow et al., 2014) abilities to address the PU learning problem by using relevant generated counter-examples. In this paper, we propose a new GAN-based PU learning approach named Divergent-GAN (D-GAN). It consists in asking to the generator to converge towards the unlabeled samples distribution while diverging from the positive samples distribution, by incorporating a standard Positive Unlabeled learning risk inside the GAN discriminator loss function. This results in the generator convergence towards the unlabeled counter-examples distribution without using prior knowledge, while keeping the standard adversarial GAN architecture. Experimental results show that this proposed framework achieves state of the art PU learning performances in terms of prediction, stability, and its ability to work on both simple and complex images.

1 INTRODUCTION

Nowadays, the number of available labeled datasets dedicated to perception applications has considerably augmented (Russakovsky et al., 2015), (Yu et al., 2015), (Cordts et al., 2016). However, when the learning methods are trained on these datasets in order to be applied on real application data, generalization problem is intensified. Indeed, the prediction performances of these methods deteriorate when the real data on which they are applied do not follow the same statistical distribution as the training dataset. Consequently, it is necessary to use a dataset specialized for the given target application. It turns out that it can be easy to get unlabeled data from the real world such as the autonomous driving domain. If we consider these unlabeled data as counter-examples of our classes of interest, then this enables a better adaptability of these learning methods to real world applications, without the need to label additional data. This is known as Positive Unlabeled (PU) learning. (Sansone et al., 2018) enumerates several learning poblems which can be adressed in this way. The PU learning task consists in learning to distinguish positive samples, considered as the samples of our class of interest, from negative samples considered as the counter-examples of our class of interest, by using only positive and unlabeled samples during the training phase. The interest is that unlabeled data can contain relevant counter-examples. The difficulty is that unlabeled data can also contain a fraction π_p of unlabeled positive samples.

Several PU learning methods exist and some of them are adapted for image classification. They are generally classified into two categories: censoring PU learning formalized by Elkan & Noto (2008), recently improved by Northcutt et al. (2017) and case-control PU learning methods introduced by Ward et al. (2009), formalized by du Plessis et al. (2014), and then consecutively improved by Du Plessis et al. (2015) and Kiryo et al. (2017). However, in order to highlight GAN-based approaches such that their main goal is to prepare a PN dataset, in this article we classify PU learning approaches into two groups suggested by Kiryo et al. (2017): One-stage and two-stage PU methods. One-stage ones such as the unbiased PU method (uPU) (Du Plessis et al., 2015) and the non-negative PU method (nnPU) (Kiryo et al., 2017) consist in training a classifier directly with the PU dataset. These methods have the advantage to need only one training of the classifier. But they require dataset prior knowledge. In practice uPU and nnPU have to be combined with an approach able to estimate

prior knowledge (Jain et al., 2016). Two-stage methods prepare during the first stage a PN dataset by keeping only the samples considered during the training as the most negative among the unlabeled ones. Rank Pruning method (RP) (Northcutt et al., 2017) acchieves two-stage state of the art performances. Several consecutive trainings of a binary classifier are performed. After each training, only the samples considered as the most negative are kept in order to substitute the unlabeled samples for the next training of the same classifier. RP achieves good prediction performances without prior knowledge, but with a high training computational cost and is limited in its prediction performances on complex RGB image datasets like CIFAR-10.

This article addresses the PU learning problem with a new GAN-based PU learning approach: the Divergent-GAN. Two-stage state of the art PU learning performances are achieved on simple MNIST and complex CIFAR-10 image datasets. Our contributions are the following:

- a direct incorporation of a PU learning risk (equation 1) inside the standard GAN framework (Goodfellow et al., 2014). The intuition behind is to ask the generative model to solve the PU learning problem formulated in the discriminator loss function. In this way the generator learns the distribution of the samples which are both unlabeled and not positive: the negative ones included in the unlabeled dataset;
- highlight of a Batch Normalization (BN) (Ioffe & Szegedy, 2015) critical issue: A learning model which manipulates different minibatches distributions should not use BN. Alternative normalization techniques are discussed (section 3.2) and experimented (section 4.1) in the context of the proposed framework which manages positive, unlabeled and generated minibatches.

From a practical point of view, several previous PU methods benefits are combined in the proposed framework as listed in the table 1 of the next section. The D-GAN code is available at [1].

The article is structured as follow. Section 2 presents GAN-based PU learning approaches. Section 3 describes the proposed method. Section 4 presents the D-GAN ability to generate relevant counter-examples on the celebA dataset, its prediction performances, and learning stability for the PU learning task compared to state of the art PU learning methods on MNIST and CIFAR-10 datasets. Then, we finish by a conclusion where we also discuss perspectives.

2 Related work

GAN-based PU learning approaches are a recent subcategory of two-stage PU methods. Both existing are PGAN (Chiaroni et al., 2018) and GenPU (Hou et al., 2018) . The interest of using GANs is double: GANs enable relevant data augmentation and they use high-level feature metrics to evaluate generated samples quality, thanks to the use of a discriminator. The generate samples considered as replace the unlabeled ones by learning on these latter (Chiaroni et al., 2018), or on both unlabeled and positive labeled ones as (Hou et al., 2018) during the first step. Both methods (Hou et al., 2018) and (Chiaroni et al., 2018) exploit GANs (Goodfellow et al., 2014) benefits, but they are not suitable under the same datasets conditions and the reasonings are different:

- GenPU method is based on the original GAN convergence (Goodfellow et al., 2014), such that: $\pi \cdot p_{G_P} + (1-\pi) \cdot p_{G_N} \rightarrow p_U$, with p_{G_P} the distribution of positive samples generated by the generator G_P, p_{G_N} the distribution of the negative samples generated by G_N, and p_U the distribution of real unlabeled samples. The same idea can be expressed by using one single generator G_N such that we replace the generated positive samples by the positive labeled samples initially present in a PU dataset. Training adversarially five learning models to address standard PU learning challenge is not necessary. In practice, GenPU seems to be the most interesting PU method on simple datasets when we own few positive labeled samples, and it generates relevant counter-examples. However, GenPU uses prior knowledge while its training optimization functions need three additional hyper-parameters. Its architecture composed of five learning models, instead of two for the standard GANs, is more computational demanding.

[1] The code will be publicly available after the anonymous reviewing period.

- PGAN is trained to converge towards the unlabeled dataset distribution during the first step. The PGAN exploits GANs imperfections such that the generated distribution at the adversarial equilibrium is still separable from the unlabeled samples distribution by a classifier. The PGAN does not use a PU learning risk to train its GAN part. Positive-GAN (PGAN) method presents a relatively steadier behaviour than RP and better results on the complex RGB image dataset CIFAR-10, in terms of prediction performances, and without prior knowledge for varying π_P fractions. However, it appears to be less suited for relatively simpler datasets such as MNIST. The generated samples are all considered as negative samples by the classifier. But this is true only if the generated samples distribution converges close enough towards the unlabeled samples distribution, while not matching with it. In other words, if the GAN works as expected theoretically by Goodfellow et al. (2014), then the PGAN will fall back into the initial PU learning problem. However, it provides a practical framework based on standard GAN architecture and it is suited for complex datasets, contrary to previous methods. As far as the classification step is concerned, it only needs a standard GAN model, such as (Goodfellow et al., 2014) or (Arjovsky et al., 2017), composed of one generator and one discriminator.

The proposed approach merges the advandages of previously cited methods to address the standard PU learning task with a different reasoning presented in the next section.

Table 1: Practical D-GAN proposed approach advantages.

Methods	D-GAN (proposed)	PGAN (Chiaroni et al., 2018)	GenPU (Hou et al., 2018)	RP (Northcutt et al., 2017)	nnPU (Kiryo et al., 2017)	uPU (Du Plessis et al., 2015)
No need of priori knowledge	✓	✓	×	✓	×	×
Practical PU framework	✓	✓	×	✓	✓	✓
Simple PU dataset analysis	✓	×	✓	✓	✓	✓
Complex PU dataset analysis	✓	✓	×	×		
Generation of relevant counter-examples	✓	×	✓			
Noisy Labeled learning (section 6.2)	✓	×	×	✓	×	×

3 METHOD

This section starts by highlighting some particularities of a standard PU learning risk (equation 1). Then we incorporate this risk into a generic GAN framework in order to guide the generator convergence towards the distribution of the negative samples included in the unlabeled dataset. In addition, we discuss normalization techniques in the context of this framework which manipulates three different types of minibatches: positive, unlabeled and generated ones.

3.1 A DIVERGENT APPROACH FROM THE DISCRIMINATOR POINT OF VIEW

The D-GAN intuition can be expressed as follow: *Show me what IS unlabeled AND NOT positive.* The discriminator D addresses this riddle to the generator G. It turns out that negative samples included in the unlabeled dataset are both unlabeled and not positive. Consequently G adresses this riddle by learning to show the negative samples distribution to D.

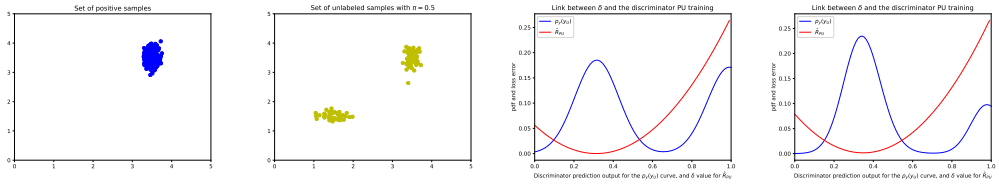
The Positive Unlabeled (PU) learning problem consists in trying to distinguish positive samples from negative samples by using a PU dataset. We start by detailing the expected PU functionality that we want to incorporate into the GAN discriminator. Let $X \in \mathbb{R}^m$ be the input random variable and $y \in [0, 1]$ its associated label. X can be a positive x_P, negative x_N or unlabeled x_U sample which respectively follow the distributions $p_P = p(X|y = 0)$, $p_N = p(X|y = 1)$ and $p_U = (1 - \pi_P) \cdot p_N + \pi_P \cdot p_P$, with $\pi_P \in (0, 1)$ the unknown prior which represents the fraction of unlabeled positive samples included in the unlabeled dataset. Let $D : \mathbb{R}^m \to (0, 1)$ be the decision function which is then considered as the discriminator network of the proposed framework. We have $l(\hat{y}, y)$ such that $l : (0, 1) \times [0, 1] \to \mathbb{R}$ is the arbitrary cost function with the predicted output $\hat{y}$ of D for a given sample X and the corresponding label y as inputs. D is trained with a PU risk R_{PU} to predict the label value $y = 1$ for the unlabeled samples, and the label value $y = 0$ for the positive labeled ones. Given the composition of the unlabeled dataset, we develop $R_{PU}(D)$ such that:

$$
\begin{aligned}
R_{PU}(D) &= \mathbb{E}_{X\sim \boldsymbol{p_U}}[l(D(X), \boldsymbol{y=1})] + \mathbb{E}_{X\sim \boldsymbol{p_P}}[l(D(X), \boldsymbol{y=0})] \\
&= (1-\pi_P)\cdot \mathbb{E}_{X\sim \boldsymbol{p_N}}[l(D(X), \boldsymbol{y=1})] + \pi_P \cdot \mathbb{E}_{X\sim \boldsymbol{p_P}}[l(D(X), \boldsymbol{y=1})] \\
&+ \mathbb{E}_{X\sim \boldsymbol{p_P}}[l(D(X), \boldsymbol{y=0})].
\end{aligned} \tag{1}
$$

Negative samples are associated exclusively to the label value $y = 1$ for any π_P value. Labeled and unlabeled positive samples follow the same distribution p_P. So p_P is associated to both contradictory labels $y = 0$ and $y = 1$. Thus if D does not overfit labeled and unlabeled positive samples, let $\delta \in (0, 1)$ be the label associated to all positive samples such that we propose the Positive Negative risk $R_{PN}(D, \delta) = R_{PU}(D)$. δ represents the intermediate label value equivalent of associating, in the risk R_{PU}, both contradictory labels $y = 0$ and $y = 1$ respectively to labeled and unlabeled positive samples. We develop the risk R_{PN} as follow:

$$
R_{PN}(D, \delta) = (1-\pi_P)\cdot \mathbb{E}_{X\sim p_N}[l(D(X), y=1)] + (1+\pi_P)\cdot \mathbb{E}_{X\sim p_P}[l(D(X), y=\delta)]. \tag{2}
$$

As illustrated in figure 1, in practice we can train D with the risk R_{PU} to output a value $\hat{y}$ close to δ when the input is a positive sample x_P, and close to 1 when the input is a negative sample x_N. We can observe that the global minimum of the approximated risk $\hat{R}_{PN}(D, \delta)$ corresponds to the global maximum of the distribution predicted by D in output for positive unlabeled samples. Moreover, the distribution of the negative samples is centered on the label associated to the unlabeled samples during the training, which coincides also with the equation 2 formulation. This illustrates experimentally the equality $R_{PN}(D, \delta) = R_{PU}(D)$.



(a) (b) (c) (d)

Figure 1: Link between the PN loss function suggested (equation 2) and the distribution of the discriminator output predictions for an unlabeled minibatch. For this experiement, l is the mean squared error (MSE) and D is a multi-layer perceptron. D has been trained to distinguish a 2D gaussian distribution to another one by using the risk R_{PU} on a PU dataset. (a) Shows a set of 2D points considered as positive samples. (b) Shows a set of 2D points considered as unlabeled samples. Both curves in (c) and (d) have been normalized to get a better visualization. For (c), $p_Y(D(x_U))$ (in blue) represents the probability distribution of D predicted outputs for a minibatch of unlabeled samples, with $\pi_P = 0.5$. $\hat{R}_{PN}(D, \delta)$ (in red) represents the PN risk computed in function of δ with the equation 2 on a minibatch of positive and negative labeled samples, once D is trained with R_{PU} risk. (d) shows the same curves as in (c) but by giving in input a concatenation of an unlabeled minibatch with a positive labeled minibatch. Unlabeled and labeled positive samples provide a unified prediction output distribution.

Next, let us consider the decision function D as the discriminator of a generative adversarial network. We propose to incorporate the risk R_{PU} into the value function of the original GAN model (Goodfellow et al., 2014) by adding the loss function term $log[1 - D(x_P)]$. Consequently, the loss function L_D of D is:

$$
\begin{aligned}
L_D &= \mathbb{E}_{x_U\sim p_U(x_U)}[logD(x_U)] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))] \\
&+ \mathbb{E}_{x\sim p_P(x_P)}[log(1-D(x_P))],
\end{aligned} \tag{3}
$$

with z the input random vector of the generative model G such that $G(z)$ is a generated sample. z can follow a uniform or normal distribution. The equation 3 is justified intuitively as follow. Instead of asking to G to converge towards the distribution p_U, we want that G learns to converge towards the distribution p_N of negative samples. Thus we propose to replace the standard GAN task *show*

me what is unlabeled by the task *show me what is both unlabeled and not positive*. This results in addressing directly the risk R_{PU} to D such that L_D is as follow:

$$L_D = R_{PU} + \mathbb{E}_{z \sim p_z(z)}[l(D(G(z)), y = 0)], \tag{4}$$

with $y = 0$ the label associated to the samples $G(z)$ generated by G. Let the cost function l used in the proposed D-GAN framework be the binary cross-entropy H such that $l = -H$. So:

$$\begin{aligned} L_D = \mathbb{E}_{x_U \sim p_U(x_U)}[-H(D(x_U), y = 1)] + \mathbb{E}_{x \sim p_P(x_P)}[-H(D(x_P), y = 0)] \\ + \mathbb{E}_{z \sim p_z(z)}[-H(D(G(z)), y = 0)], \end{aligned} \tag{5}$$

We recall that the binary cross-entropy H is defined as below:

$$H(D(X), y) = -y \cdot log(D(X)) - (1 - y) \cdot log(1 - D(X)), \tag{6}$$

with y the label associated to the D input samples X. Thus $H(D(X), y = 1) = -log(D(X))$ and $H(D(X), y = 0) = -log(1 - D(X))$. Consequently, L_D can be developed as in the equation 3. This shows the incorporation of R_{PU} risk (equation 1) inside the D-GAN discriminator loss function (equation 3).

On the other hand, the role of G during the adversarial training is to generate samples considered by D as 1. So as suggested by Goodfellow et al. (2014) in practice, the training loss function L_G of G is such that $L_G = \mathbb{E}_{z \sim p_z(z)}[log(D(G(z)))]$. Only negative samples are considered as 1 by D thanks to the R_{PU} risk, as shown previously. This justifies intuitively the G convergence towards the negative samples distribution p_N. The corresponding implementation algorithm 1 of this first stage is presented in annexe 6.

Once the D-GAN training is finished, we can start the second step which consists in using a classifier C to distinguish fake generated samples $x_{FN} = G(z)$, which are ideally equivalent to the real negative samples, from real positive labeled samples as illustrated in figure 2. The divergent GAN

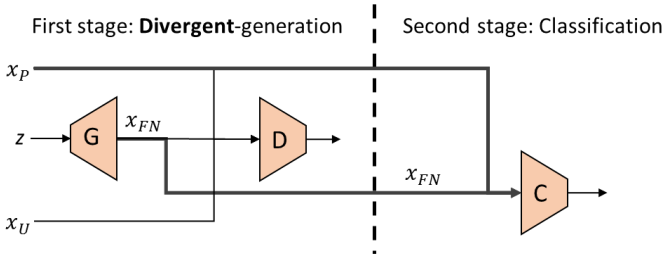


Figure 2: Proposed Positive Unlabeled system: Divergent-GAN Learning model. x_{FN} represents the fake generated samples which are similar to real negative samples x_N. G is the generative model. D is the discriminator. C is the classifier which is used to perform a binary Positive-Negative classification.

behaviour is only possible if the discriminator D is able to distinguish positive and negative samples. Nonetheless, in the worst-case scenario where the discriminator is not able to sufficiently encode the complexity of the boundary between positive and negative samples, the D-GAN will then converge towards the behaviour of the PGAN, which seems to be the best solution in this situation. This is observed experimentally in the table 3 on the CIFAR-10 dataset.

3.2 Without discriminator batch normalization

Nowadays, Batch Normalization (BN) (Ioffe & Szegedy, 2015), is considered as a relevant regularization technique commonly used in deep neural networks architectures. Its utility for GANs training has been highlighted by (Radford et al., 2015) for the DCGAN architecture in order to stabilize the adversarial training. Other variants like the Wasserstein-GAN (Arjovsky et al., 2017) or the Loss-Sensitive GAN (Qi, 2017) confirmed its interest. Indeed, as developped in (Ioffe & Szegedy, 2015),

BN addresses issues like vanishing, exploding gradient problems, and the risk of getting stuck in a poor local minima. BN suggests to reduce the $internal\ covariate\ shift$ problem of the learning model. In this way, a higher learning rate can be used and it can improve significantly the training speed.

But BN regularizes the model, such that a training sample from a given minibatch is considered in conjunction with other examples of this minibatch. This is the consequence of estimating the mean and variance normalization parameters one time per minibatch, and then applying them on each minibatch sample. It turns out that it does not allow to link labeled positive samples to the unlabeled positive samples and to produce a gap between positive and negative samples, when positive samples x_P and unlabeled samples x_U are not in the same training minibatch. To counter this problem, we could imagine to apply BN on a unified minibatch which contains a fraction of each distribution x_P, x_U and x_F. But the BN effect is influenced a lot by the content of the minibatch on which it is applied. So the fraction π_P of positive samples included in x_U will affect the batchnorm effect.

As illustrated experimentally in figure 5, with BN, we observe that the classifier only learns to distinguish minibatches of positive samples from minibatches of unlabeled samples. On the contrary, without the use of BN, both distributions appear in the unlabeled minibatch from the discriminator output point of view. In other words, the negative samples can be distinguished from the positive samples without BN.

However, BN benefits in a more traditional training are not negligible, so we suggest to use another type of normalization which could replace BN role in a PU learning task. Layer Normalization (LN) (Ba et al., 2016), is a frequently used technique with sequential networks as it can be applied for each sequential sample separately. Indeed, with LN the normalization for a given sample is computed on its resulting output feature map layers and the mean and variance are computed separately for each sample of a minibatch. For this reason, we propose to apply LNs instead of BNs inside our discriminative model D. As we can observe in figure 5, it allows the discriminator to keep the behaviour that interests us which is the ability to distinguish unlabeled negative samples to unlabeled positive samples. Additional qualitative experiments in annexe section 6 show that the LN can substitute BN inside the discriminative model structure to generate acceptable images.

The next section presents experimental results which demonstrate the good performance of the proposed approach.

4 EXPERIMENTAL RESULTS

This section demonstrates experimentally the D-GAN abilities on simple and complex datasets: We present qualitative results for the counter-example generation, the discriminator normalization effect, and quantitative results for the PU learning image classification task.

Experimental settings are as follow. Minibatches of labeled positive samples, unlabeled samples, and generated samples have the same size. The convolutional classifier C used [2] in the subsection 4.2 is described in the PGAN article (Chiaroni et al., 2018) experiments. C is always trained during 20 epochs. We use the DCGAN (Radford et al., 2015) architecture for all the comparative experiments except for celebA and CIFAR-10 datasets where we respectively adapt the LS-GAN (Qi, 2017) and WGAN-GP (Gulrajani et al., 2017) variants to the D-GAN framework. The D-GAN uses LN instead of BN concerning D. Concerning the PU dataset initialization from a standard PN dataset, ρ is the fraction of the positive labeled samples of the initial PN dataset that we unlabel such that they are included into the unlabeled dataset. π_P is the fraction that represents these unlabeled positive samples among the unlabeled dataset.

From a qualitative point of view, and on the contrary to the PGAN model, the D-GAN generates samples which are only similar to the counter-examples for diverse data types, as illustrated in figure 4 for a 2D point cloud dataset and in figure 3 for a real images dataset. In order to enable the proposed approach reproducibility, a D-GAN implemented version corresponding to the figure 3 results is available [3] and is applied on the LSGAN model (Qi, 2017). Our code includes also the

[2] $https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/tutorials/mnist/mnist_softmax.py$

[3] after the anonymous reviewing period

method implementation, proposed by Chiaroni et al. (2018), to establish a PU training dataset from a fully labeled dataset.

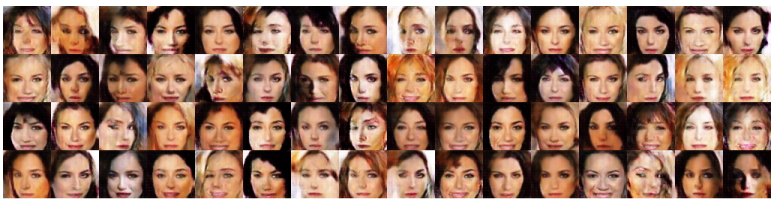
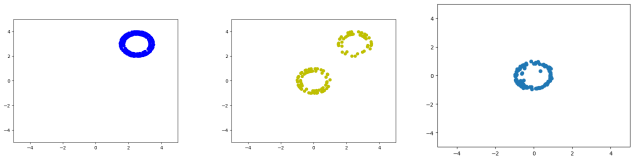


Figure 3: This is a minibatch of fake negative images x_{FN} generated, after 20 epochs iteration, with the D-GAN based on the Loss-Sensitive GAN (Qi, 2017) and by replacing BN with LN. The PU training dataset uses the celebA dataset images with settings $\rho = 0.5$, $\pi_P = 0.3$ and "Male" the positive class chosen arbitrarily. The images size is cropped to 64*64*3. All generated samples are qualitatively relevant fake counter-examples.

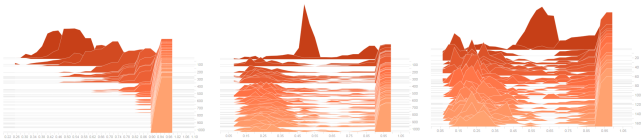


(a) (b) (c)

Figure 4: D-GAN functioning on clusters of 2D points such that D and G have a multilayer-perceptron structure. (a), (b) and (c) are respectively sets of labeled positive, unlabeled, and generated samples. The generated samples only follow the distribution of the counter-examples.

4.1 Normalization effects on the Discriminator

We compare in the figure 5 the ability of D to distinguish the positive from the negative samples distribution when it is trained on a PU dataset with BN, LN layers or without any normalization technique. Smaller is surface intersection between the discriminator output histogram distribution predictions on positive and negative samples included in the unlabeled dataset, better it is.



(a) (b) (c)

Figure 5: Normalization effect on the histogram distribution of output values predicted by D on an unlabeled samples minibatch. (a) is with BN, (b) without any normalization, and (c) with LN. The axe in depth represents the training iterations. The lighter histogram corresponds to the last training iteration. Settings are with positive class 8 and negative class 3 of MNIST dataset, with $\rho = 0.5$ and $\pi_P = 0.3$. We observe that BN in the figure 5(a) does not enable to distinguish positive from negative samples when the discriminator is trained on a PU dataset. On the contrary, LN (figure 5(c)) enables it for the reasons specified in the subsection 3.2. Thus with LN in D, G can converge exclusively towards the unlabeled negative samples distribution p_N, as expected.

4.2 DIVERGENT-GAN FOR POSITIVE UNLABELED LEARNING

We compare the D-GAN accuracy prediction to GenPU on the table 2 for the simple One vs. One task with few positive labeled samples. The GenPU method presents better results for fewer labeled positive samples. The D-GAN presents Accuracy performances better than the GenPU method for one hundred of positive labeled samples after 40 epochs training iterations for the D-GAN counter-examples generation step.

Moreover, it is interesting to observe that the proposed approach globally slightly outperforms nnPU and uPU methods without using prior knowledge.

Table 2: Comparative results of Accuracy prediction performances on MNIST in a One vs. One mode. N_P is the number of positive labeled samples, and N_U is the number of unlabeled samples which mixes all the rest of positive and negative samples. GenPU, uPU and nnPU results come from (Hou et al., 2018).

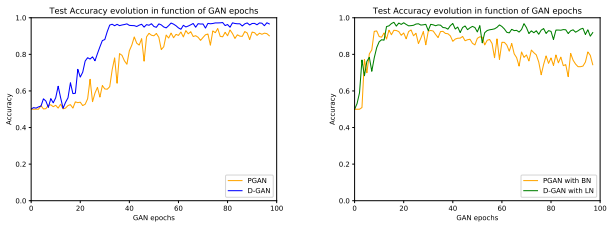
One vs. One	'3'vs.'5'				'8'vs.'3'			
Dataset: MNIST	D-GAN (proposed)	GenPU	nnPU	uPU	D-GAN (proposed)	GenPU	nnPU	uPU
No need of prior knowledge	✓	×	×	×	✓	×	×	×
N_P=100 : N_U=9900	**0.987**	0.983	0.969	0.914	**0.989**	0.982	0.974	0.932
N_P=50 : N_U=9950	0.964	**0.982**	0.966	0.854	0.974	**0.979**	0.965	0.873

Concerning the relatively more challenging One vs. Rest task, the D-GAN globally outperforms the PGAN and RP methods in terms of F1-Score predictions performances on the table 3 both on MNIST and CIFAR-10 datasets.

Moreover, the D-GAN demonstrates a steadier behaviour than PGAN against the overfitting problem as illustrated in figure 6(b).

Table 3: Comparative results of average F1-Score prediction performances in a One vs. Rest mode on MNIST and CIFAR-10 with respectively 35 Divergent-GAN epochs on MNIST and 275 Divergent-WGAN-GP epochs on CIFAR-10. PGAN and RP results come from (Chiaroni et al., 2018). PNGAN represents a PN (PU when $\pi_P = 0$) training such that the negative samples are replaced by WGAN generated ones. This highlights the GANs data augmentation effect on complex datasets like CIFAR-10.

One vs. Rest	ref		$\rho = 0.5, \pi_P = 0.1$			$\rho = 0.5, \pi_P = 0.3$			$\rho = 0.5, \pi_P = 0.5$			$\rho = 0.5, \pi_P = 0.7$		
Datasets	PN	PNGAN	D-GAN	PGAN	RP	D-GAN	PGAN	RP	D-GAN	PGAN	RP	D-GAN	PGAN	RP
AVG_{MNIST}	0.993	0.988	**0.99**	0.965	0.967	**0.984**	0.958	0.975	**0.972**	0.946	0.951	0.922	0.875	**0.933**
$AVG_{\text{CIFAR-10}}$	0.680	0.812	**0.749**	0.745	0.622	0.727	**0.760**	0.730	**0.749**	0.748	0.716	**0.713**	0.702	0.684



(a) (b)

Figure 6: Classifier test Accuracy evolution in function of the GAN epochs (first stage). 8 vs. Rest MNIST task, with $\rho = 0.5$ and $\pi_p = 0.5$. (a) D-GAN and PGAN are trained without normalization layers. (b) D-GAN and PGAN are respectively trained with LN and BN inside the discriminator.

In figure 6(a), without BN or LN, the D-GAN method gets a faster and better $Accuracy$ prediction performance than the PGAN when both are trained under the same conditions. In figure 6(b), the D-GAN with LN follows the learning training rythm of the PGAN with BN. The D-GAN also

has a steadier behaviour once the Accuracy progression is finished: it reduces the PGAN first step overfitting problem.

5 CONCLUSION

To conclude, we have incorporated into the discriminator D loss function a standard PU risk to address the PU learning challenge. In this way, the proposed model generates relevant counter-examples from a PU dataset without prior knowledge. It achieves two-stage state of the art PU learning performances by addressing previous GAN-based PU methods issues: -It reduces the over-fitting PGAN first stage problem. -It addresses the GenPU mode collapse issue by keeping the practical standard GANs architecture such that it is easily adaptable to conventional GANs variants. A side contribution of this article is to identify a discriminator batch-normalization critical issue appearing when the discriminator manipulates multiple minibatches distributions. Nonetheless, we strongly believe that its stability and prediction performances still have the potential to be improved by taking the best of the representation learning and PU learning domains. For example, recent promising GAN training approaches (Karras et al., 2018) not mandatorily using BN should be suitable to extend the proposed approache for large-scaled image datasets. In addition, premices of Noisy labeled learning results presented in annexe section 6.2 could extend the proposed approach to a unified generative framework which could output relevant generated labeled samples even from a weak, noisy labeled or incremental dataset with varying fractions of unlabeled positive samples.

REFERENCES

Martin Arjovsky, Soumith Chintala, and Lon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pp. 214–223, 2017.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Florent Chiaroni, Mohamed-Cherif Rahal, Nicolas Hueber, and Frederic Dufaux. Learning with a generative adversarial network from a positive unlabeled dataset for image classification. In *IEEE International Conference on Image Processing*, 2018.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

C. Craye, D. Filliat, and J. Goudou. Biovision: a biomimetics platform for intrinsically motivated visual saliency learning. *IEEE Transactions on Cognitive and Developmental Systems*, 2018.

Marthinus Du Plessis, Gang Niu, and Masashi Sugiyama. Convex formulation for learning from positive and unlabeled data. In *International Conference on Machine Learning*, pp. 1386–1394, 2015.

Marthinus C. du Plessis, Gang Niu, and Masashi Sugiyama. Analysis of learning from positive and unlabeled data. In *Advances in neural information processing systems*, pp. 703–711, 2014. URL `http://papers.nips.cc/paper/5509-analysis-of-learning-from-positive-and-unlabeled-data`.

Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 213–220. ACM, 2008.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pp. 5769–5779, 2017.

Raia Hadsell, Pierre Sermanet, Jan Ben, Ayse Erkan, Marco Scoffier, Koray Kavukcuoglu, Urs Muller, and Yann LeCun. Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics*, 26(2):120–144, 2009.

Ming Hou, Qibin Zhao, Chao Li, and Brahim Chaib-draa. A generative adversarial framework for positive-unlabeled classification. *arXiv preprint arXiv:1711.08054*, 2018.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456, 2015.

Shantanu Jain, Martha White, and Predrag Radivojac. Estimating the class prior and posterior from noisy positives and unlabeled data. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 29*, pp. 2693–2701. Curran Associates, Inc., 2016.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

Ryuichi Kiryo, Gang Niu, Marthinus C du Plessis, and Masashi Sugiyama. Positive-Unlabeled Learning with Non-Negative Risk Estimator. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30*, pp. 1675–1685. Curran Associates, Inc., 2017.

Curtis G. Northcutt, Tailin Wu, and Isaac L. Chuang. Learning with Confident Examples: Rank Pruning for Robust Classification with Noisy Labels. *arXiv preprint arXiv:1705.01936*, 2017.

Guo-Jun Qi. Loss-sensitive generative adversarial networks on lipschitz densities. *arXiv preprint arXiv:1701.06264*, 2017.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pp. 2234–2242, 2016.

Emanuele Sansone, Francesco GB De Natale, and Zhi-Hua Zhou. Efficient training for positive unlabeled learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

Gill Ward, Trevor Hastie, Simon Barry, Jane Elith, and John R. Leathwick. Presence-only data and the EM algorithm. *Biometrics*, 65(2):554–563, 2009.

Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. LSUN: construction of a large-scale image dataset using deep learning with humans in the loop. *CoRR*, abs/1506.03365, 2015. URL `http://arxiv.org/abs/1506.03365`.

6 ANNEXE

The algorithm 1 is the implementation of the proposed approach. It includes the option for the noisy labeled learning task.

The table 4 details for each class the average F1-Score results presented in Table 3.

The figure 9 presents the same minibatch generated as in the figure 3, but in a larger size.

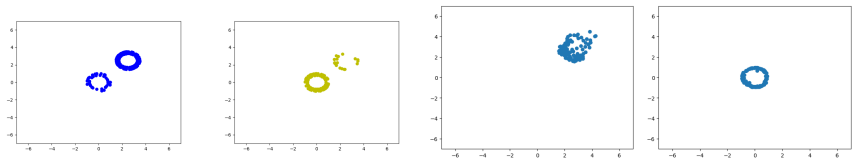
6.1 NEEDING PRIOR KNOWLEDGE MAY NOT BE SUITABLE FOR INCREMENTAL APPLICATIONS

In the context of real learning applications like incremental learning (Craye et al., 2018), (Hadsell et al., 2009), the fraction π_P of unlabeled positive samples can vary continuously at each new incremental unlabeled minibatch of samples to process. However, existing PU learning methods using prior knowledge, like uPU and nnPU implicitly do the assumption that the prior knowledge holds the same during all the training. A PU learning method which does not need prior knowledge as the D-GAN proposed approach could be extended to such real applications.

6.2 AN EXTENSION TO NOISY LABELED LEARNING AND FEW LABELED DATA RESSOURCES

If a generative model in an adversarial training only converges towards the subdivision of a distribution which is considered as the closest to a given label from the discriminator point of view, then we can extend the proposed approach to the noisy labeled learning task. More precisely, the D-GAN architecture can be modified to include a generative model G_P trained to generate fake positive samples. This is achieved by adding the loss function term $logD(G_P(z))$ in the minimax value function. Furthermore, this extension also allows to manage problems where we only have few labeled positive samples. Indeed, the positive generator G_P will converge towards the distribution considered as the most positive, the closest to zero in our case, which includes also the unlabeled positive samples perceived by the discriminator at the same location that the labeled ones as illustrated in figure 1 (d). We argue that it is indirectly the same phenomenon that allows the GenPU (Hou et al., 2018) to get relatively good accuracy predictions with few labeled positive samples. Because the discriminator task becomes more difficult, the weight α is applied on the unlabeled loss function $log(D(x_U))$ and on the positive one $log(1 - D(x_P))$. It ensures the guidelines that both generators G and G_P have to follow. The boolean variable $noisyLabelsMode$ of the algorithm 1 is set to $True$ to activate this functionality.

The figure 7 shows the results obtained on clusters of 2D point clouds with the D-GAN, when its noisy labeled mode is activated.



(a) (b) (c) (d)

Figure 7: D-GAN Noisy-labeled version functioning on sets of 2D points such that (a), (b), (c) and (d) are respectively sets of real noisy labeled positive (20 percents of noise), real noisy labeled negative (20 percents of noise), fake generated positive and fake generated negative samples. D, G and G_P have a multilayer-perceptron structure. The generated samples only follow the ditribution of the well-labeled samples of the input sets.

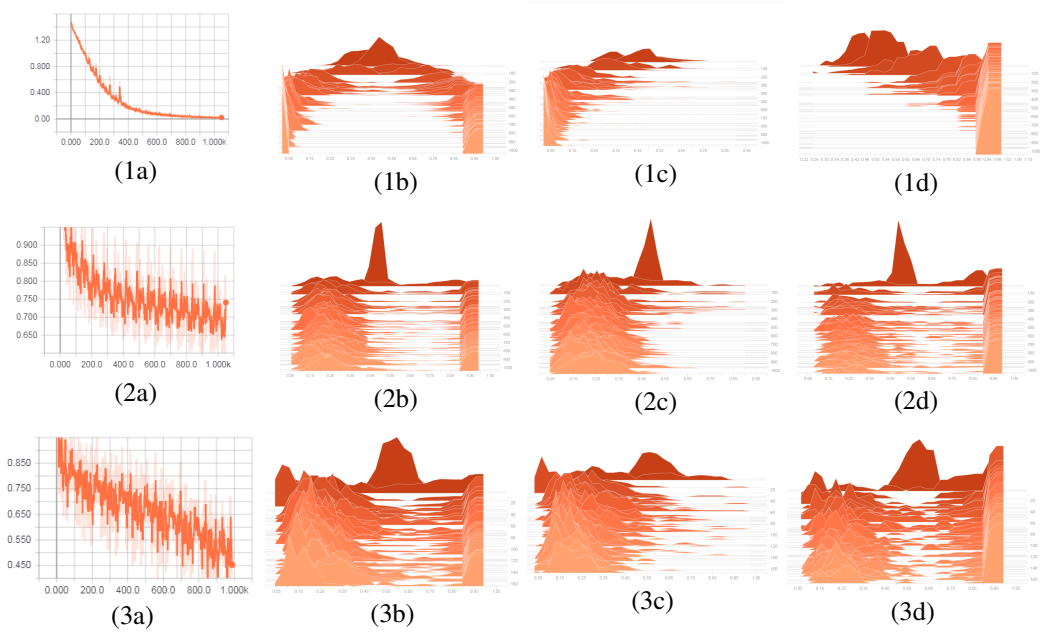


Figure 8: Details of figure 5 Batch Normalization effect on discriminator output predictions on distinct Positive and Unlabeled minibatches. Settings are with positive class 8 and negative class 3 of MNIST dataset, with $\rho = 0.5$ and $\pi_P = 0.3$. The results on the first row are done with BN, without on the second row, and with LN on the third row. The first column (a) is the mean D loss value estimated in function of training iterations. The second column (b) is the histogram distributions of output values predicted by D on the positive and unlabeled minibatches. The third column (c) is the histogram distribution of output values predicted by D on the positive minibatch. The fourth column (d) is the histogram distribution of output values predicted by D on the unlabeled minibatch. The axe in depth represents the training iteration axe on which the clearest histograms gradually correspond to the closest training iterations. Other alternative normalization techniques have been applied to GANs, such as weight normalization (Karras et al., 2018) or the Virtual Batch Normalization (VBN) (Salimans et al., 2016) where they essentially mention the same problem for BN. However, VBN parameters strongly depend to their "reference batch" composition. This is a severe limitation if the fraction of unlabeled positive samples changes during the training, as it can happen for example in a real incremental learning application. Moreover, VBN in (Salimans et al., 2016) is only used in the generator, which is not what we are looking for. For PU learning task, BN is not a problem inside the generator structure.

Algorithm 1 Minibatch stochastic gradient descent training of the Divergent-GAN

Initialize $noisyLabelsMode = false$.
Initialize $\alpha = 2$.
for number of training iterations **do**
  for k steps **do**
    Sample minibatch of m noise samples $\{z^{(1)}, ..., z^{(m)}\}$ from noise prior $p_G(z)$.
    Sample minibatch of m unlabeled examples $\{x_U^{(1)}, ..., x_U^{(m)}\}$ from data distribution p_U.
    Sample minibatch of m positive labeled examples $\{x_P^{(1)}, ..., x_P^{(m)}\}$ from data distribution p_P.

    Update the discriminator D by descending its stochastic gradient:
    if $noisyLabelsMode$ is $false$ **then**

$$\nabla_{\theta_D} \frac{1}{m} \sum_{i=0}^{m} logD(x_U^{(i)}) + log[1 - D(G(z^{(i)}))] + log[1 - D(x_p^{(i)})]$$

    else

$$\nabla_{\theta_D} \frac{1}{m} \sum_{i=0}^{m} \alpha \cdot \big(logD(x_U^{(i)}) + log[1 - D(x_p^{(i)})]\big) + log[1 - D(G(z^{(i)}))] + logD(G_P(z^{(i)}))$$

    end if
  end for
  Sample minibatch of m noise samples $\{z^{(1)}, ..., z^{(m)}\}$ from noise prior $p_G(z)$.
  Update the generator G by descending its stochastic gradient:

$$\nabla_{\theta_G} \frac{1}{m} \sum_{i=0}^{m} log[D(G(z^{(i)}))]$$

  if $noisyLabelsMode$ is $true$ **then**
    Update the generator G_P by descending its stochastic gradient:

$$\nabla_{\theta_{G_P}} \frac{1}{m} \sum_{i=0}^{m} log[1 - D(G_P(z^{(i)}))]$$

  end if
end for
The gradient-based updates can use any standard gradient-based learning rule. We use Adam in our experiments.

Table 4: Comparative results of F1-Score prediction performances on MNIST and CIFAR-10 in a One vs. Rest mode with 20 classifier epochs.

One vs. Rest	ref		$\rho = 0.5, \pi_P = 0.1$			$\rho = 0.5, \pi_P = 0.3$			$\rho = 0.5, \pi_P = 0.5$			$\rho = 0.5, \pi_P = 0.7$		
Datasets	PN	PNGAN	D-GAN	PGAN	RP	D-GAN	PGAN	RP	D-GAN	PGAN	RP	D-GAN	PGAN	RP
0	0.997	0.992	**0.996**	0.974	0.992	**0.994**	0.973	0.955	0.984	0.973	**0.991**	**0.965**	0.902	0.88
1	0.998	0.993	**0.995**	0.971	**0.995**	0.994	0.979	**0.996**	0.988	0.958	**0.994**	0.952	0.863	**0.993**
2	0.99	0.987	**0.992**	0.972	0.975	**0.99**	0.959	0.923	**0.98**	0.947	0.936	0.892	0.914	**0.987**
3	0.996	0.996	**0.993**	0.963	0.991	0.97	0.953	**0.991**	**0.957**	0.934	0.882	**0.914**	0.885	0.829
4	0.997	0.992	**0.989**	0.964	0.972	0.989	0.952	**0.995**	**0.972**	0.945	0.933	0.923	0.914	**0.977**
5	0.993	0.981	**0.988**	0.974	0.985	**0.976**	0.95	0.943	**0.969**	0.949	0.91	0.894	0.873	**0.973**
6	0.992	0.989	**0.992**	0.962	0.928	0.99	0.959	**0.992**	0.986	0.971	**0.993**	0.961	0.944	**0.99**
7	0.995	0.99	**0.982**	0.962	0.947	0.983	0.96	**0.991**	0.969	0.926	**0.988**	0.918	0.737	**0.979**
8	0.995	0.985	**0.985**	0.949	0.929	0.981	0.941	**0.982**	**0.966**	0.922	0.941	**0.88**	0.849	0.818
9	0.981	0.977	**0.986**	0.959	0.954	0.972	0.956	**0.979**	**0.947**	0.939	0.941	**0.921**	0.865	0.904
AVG_{MNIST}	0.993	0.988	**0.99**	0.965	0.967	**0.984**	0.958	0.975	**0.972**	0.946	0.951	0.922	0.875	**0.933**
Plane	0.727	0.803	0.759	**0.818**	0.669	0.749	0.784	**0.795**	0.736	**0.758**	0.743	0.715	**0.731**	0.718
Auto	0.78	0.865	0.766	**0.801**	0.695	0.749	0.737	**0.829**	0.793	0.789	**0.798**	0.757	0.734	**0.783**
Bird	0.447	0.775	0.686	**0.688**	0.56	0.706	**0.744**	0.68	0.692	**0.694**	0.644	0.68	**0.688**	0.542
Cat	0.5	0.698	**0.695**	0.658	0.384	0.713	**0.722**	0.651	0.707	**0.718**	0.67	**0.712**	0.69	0.698
Deer	0.698	0.8	**0.768**	0.68	0.605	0.681	**0.708**	**0.708**	**0.72**	0.708	0.64	**0.672**	0.633	0.602
Dog	0.567	0.805	**0.716**	0.632	0.539	0.68	**0.756**	0.648	0.74	**0.746**	0.733	0.704	0.678	**0.712**
Frog	0.691	0.812	0.758	**0.837**	0.666	0.786	0.793	**0.794**	0.78	**0.788**	0.769	0.723	**0.75**	0.749
Horse	0.786	0.845	**0.697**	0.693	0.653	0.685	**0.757**	0.723	0.74	0.751	**0.759**	**0.713**	0.675	0.711
Ship	0.832	0.879	**0.851**	0.821	0.764	0.784	0.809	**0.831**	**0.797**	0.775	0.785	0.728	0.716	**0.755**
Truck	0.771	0.839	0.794	**0.822**	0.685	0.741	**0.786**	0.637	**0.778**	0.754	0.617	**0.732**	0.724	0.564
$AVG_{\text{CIFAR-10}}$	0.680	0.812	**0.749**	0.745	0.622	0.727	**0.760**	0.730	**0.749**	0.748	0.716	**0.713**	0.702	0.684

Figure 9: Minibatch of counter-examples (women faces) generated after 20 epochs with the proposed D-GAN framework adapted to the LSGAN-GP variant. The D-GAN is applied on a PU celebA dataset (celebrity RGB images of size 64*64), with 30% of positive samples (men faces) among the unlabeled ones. We can visually observe that 100% of the generated samples follow the counter-examples (women faces) distribution. The LSGAN-GP code is available at: $https://github.com/maple-research-lab/lsgan-gp-alt$.