Classification d'images en apprenant sur des échantillons positifs et non labélisés avec un réseau antagoniste génératif

F. Chiaroni[12] M-C. Rahal[1] F. Dufaux[2] N. Hueber[3]

[1] Institut VEDECOM, équipe Perception du véhicule à conduite déléguée
[2] L2S, CNRS, CentraleSupelec, Univ Paris-Sud, Univ Paris-Saclay
[3] Institut Saint-Louis Franco-Allemand (ISL), équipe ELSI

florent.chiaroni@l2s.centralesupelec.fr

Abstract

In this article, we suggest a novel approach for image classification task from a positive unlabeled datased. Its proper functioning is based on generative adversarial networks (GANs) abilities. These allow us to generate fake images whose distribution is close to the distribution of the negative samples included in the unlabelled dataset available, while remaining different from the distribution of positive samples that are not labeled. Then we train a CNN classifier with the positive samples and the fake samples generated, as it would have been done with a classical Positive Negative dataset. Tests performed on three different image classification datasets show that the system is stable in its behavior with a non negligible fraction of positive samples present in the unlabeled dataset. Although very different, this method outperforms the state of the art in PU learning on the RGB CIFAR-10 dataset.

Résumé

Dans ce document, nous proposons une nouvelle approche répondant à la tâche de classification d'images à partir d'un apprentissage sur données positives et non-labélisées. Son bon fonctionnement repose sur certaines particularités des réseaux antagonistes génératifs (GANs). Ces derniers nous permettent de générer des fausses images dont la distribution se rapproche de la distribution des échantillons négatifs inclus dans le jeu de données non labélisé disponible, tout en restant différente de la distribution des échantillons positifs non labélisés. Ensuite, nous entraînons un classifieur convolutif avec les échantillons positifs et les faux échantillons générés, tel que cela aurait été fait avec un jeu de données classique de type Positif Négatif. Les tests réalisés sur trois jeux de données différents de classification d'images montrent que le système est stable dans son comportement jusqu'à une fraction conséquente d'échantillons positifs présents dans le jeu de données non labélisé. Bien que très différente, cette méthode surpasse l'état de l'art PU learning sur le jeu de données RVB CIFAR-10.

Mots Clef

Apprentissage Positif Non Labélisé (PU learning), Classification d'Images, Apprentissage Profond, Apprentissage de Représentations, Modèles Génératifs.

1 Introduction

Les méthodes d'apprentissage utilisant des filtres à noyaux de convolution ont démontré de bonnes performances de prédiction dans le domaine du traitement d'image, et plus particulièrement pour la tâche de classification d'images. Pour réaliser de telles performances, de grands jeux de données entièrement labélisés sont requis. De nos jours, plusieurs jeux de données distincts peuvent être amenés à être fusionnés pour cette raison afin d'augmenter la capacité de généralisation d'un modèle d'apprentissage tel que cela est proposé dans YOLO9000 [18]. Par ailleurs, pour atténuer ce besoin de grands jeux de données labélisés, des méthodes d'apprentissage semi-supervisé existent [16]. Mais, si un objet n'appartenant à aucune classe labélisée du jeu de données d'entraînement doit être traité, il reste difficile de prédire le comportement du modèle entraîné à son égard.

Néanmoins, une idée pouvant répondre à ce problème consiste à se focaliser principalement sur les données qui nous intéressent. Cela est le cas pour les méthodes de One-Class Classification (OCC) [8], détection de nouveauté [14] où il est utilisé uniquement des échantillons de la classe d'intérêt ; la classe positive. Cependant, à notre connaissance, les méthodes OCC ont une performance limitée lorsqu'elles sont appliquées à des tenseurs de données de grande dimensionalité tels que des images. De plus, il est souvent facile d'acquérir des échantillons non labélisés susceptibles de contenir des informations pertinentes à propos des contre-exemples de la classe d'intérêt. De cette manière, nous abordons le problème d'apprentissage Positif Non labélisé (apprentissage PU). Il se trouve que les méthodes d'apprentissage Positif Non labélisé ont été appliquées récemment à des données de type images tel que la méthode Rank Pruning (RP) [13]. Cette méthode est la plus performante de l'état de l'art dans un contexte où l'on n'a pas de connais-

sances à priori sur les fractions d'échantillons bruités. Elle est cependant coûteuse en calculs car elle consiste à réaliser plusieurs entraînements consécutifs du même classifieur de manière à éliminer les échantillons les moins pertinents pendant la phase d'entraînement. De plus, selon [12], ces méthodes deviennent compétitives lorsque le nombre d'échantillons non labélisés dans le jeu de données d'entraînement augmente considérablement. Cela est un avantage lorsque l'on peut obtenir facilement des données non labélisées.
Par ailleurs, les réseaux génératifs antagonistes (GANs) ont attiré notre attention en raison de leur capacité à générer de faux échantillons x_F qui ont une distribution $p_G(x_F)$ qui tend vers la distribution $p_{data}(x_R)$ des échantillons réels x_R utilisés pendant son entraînement. Le GAN originel [5] contient un modèle génératif G et un modèle discriminatif D. Ces deux modèles possèdent une structure de type perceptron multi-couche. Un vecteur de bruit z, composé de variables aléatoires continues, est placé en entrée de G. D est entraîné à distinguer les échantillons réels des faux échantillons générés par G, pendant que ce dernier est entraîné à produire des faux échantillons qui doivent sembler réels au possible. Cet entraînement adversaire consiste à utiliser la fonction d'évaluation minimax $V(G, D)$:

$$\min_G \max_D V(G,D) = \mathbb{E}_{x_R \sim p_{data}(x_R)} log D(x_R) + \mathbb{E}_{z \sim p_z(z)} log[1 - D(G(z))].$$

Lorsque D ne peut plus distinguer les vrais échantillons des faux, nous obtenons la propriété suivante, avec y_D le scalaire de sortie prédit :

$$p_G(x_F) \xrightarrow[y_D \to \frac{1}{2}]{} p_{data}(x_R).$$

D'autres variantes du GAN sont apparues telles que le DCGAN [15], qui adapte sa structure au traitement d'images en intégrant des couches convolutives. Le Wasserstein GAN (WGAN) [1] utilise d'une part la distance $Earth - Mover$ (EM) dans sa fonction de coût, et d'autre part limite les valeurs des poids de son modèle à un certain intervalle, afin de rectifier le problème d'instabilité des précédentes versions du GAN.
En raison de leur capacité à apprendre des représentations pertinentes d'un point de vue sémantique et de leur efficacité déjà démontrée en apprentissage semi-supervisé [20], nous avons décidé d'exploiter d'une certaine manière leurs avantages pour une application d'apprentissage PU. Parallèlement à notre étude, l'approche [7] est apparue pour répondre à la même problématique en utilisant un modèle d'apprentissage de type GAN. [7] requière deux modèles génératifs et trois discriminateurs pour l'étape générative contre un générateur et un discriminateur pour notre approche. Cela devrait nous confèrer un temps de calcul moindre et un apprentissage mieux maîtrisé. Mais leur étude s'est arrêtée à la description fonctionnelle de leur modèle [1]. Ici, l'approche proposée que l'on nomme Positive-GAN ("PGAN" par la suite) a été testée sur trois jeux de données différents et dont les résultats sont très prometteurs en termes robustesse et de performance de prédiction pour le traitement d'images complexes. Il surpasse l'état de l'art sur le jeu de données le plus difficile que nous avons testé.
Le document est organisé tel que ci-dessous. Dans la section suivante nous présentons la méthode. Les expérimentations et les résultats sont présentés dans la troisième section. Pour finir, une conclusion est faite sur notre approche et de futures directions de recherche sont suggérées.

2 Méthode d'apprentissage proposée : Le Positive-GAN

Dans cette section, nous décrivons notre système d'apprentissage PU de manière générique et focalisons la description sur la méthode d'entraînement. Notre méthode d'apprentissage Positive-GAN (PGAN) consiste à substituer l'absence d'échantillons négatifs labélisés x_N avec les faux échantillons x_F générés par notre GAN, dont la distribution est proche au possible de celle des x_N, tout en étant différente de celle des échantillons positifs x_P. La figure 1 illustre le fonctionnement du système.

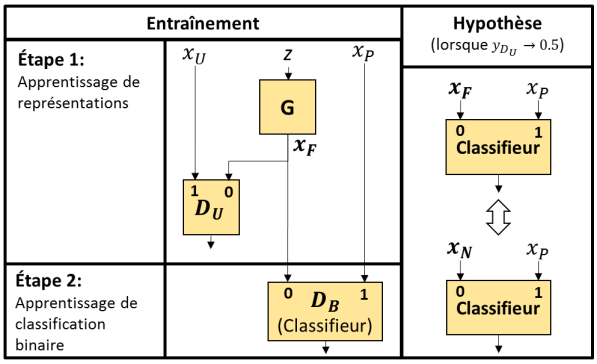


FIGURE 1 – Système d'apprentissage PU proposé : Positive-GAN.

Lors de l'étape 1, le GAN est entraîné avec les échantillons non labélisés x_U à partir du jeu de données d'entraînement

1. À noter qu'après la date de soumission de notre article à CNIA-RJCIA, une nouvelle version de la publication [7] est apparue, en version pre-print, où l'étude de leur méthode a été complétée : Leur nouvelle version datant du 4 avril 2018 inclue une partie théorique et quelques tests comparatifs sur les jeux de données USPS et MNIST auxquels il aurait été pertinent de se comparer, s'il n'y avait pas eu de conflits de dates. Leur nouvelle version nécessite la connaissance à priori de la fraction d'échantillons positifs inclus dans le jeu de données non labélisé pour fonctionner. Les tests ont été réalisés avec une faible proportion d'échantillons positifs labélisés, ce qui met en avant l'intérêt des méthodes génératives pour l'augmentation de dataset. Cependant, des détails sur l'intialisation des hyper-paramètres λ_P, λ_N et λ_U auraient été appréciés, ainsi que la réalisation de tests sur des bases de données plus complexes telles que CIFAR-10 avec des structures de type réseaux convolutifs. En effet, bien que leur méthode semble fonctionnelle avec une structure de type peceptron multicouche, les étapes de convolution rendent plus difficile l'effondrement d'un générateur et donc la divergence de leur générateur G_N pour l'apprentissage de la distribution des échantillons négatifs à partir d'échantillons non labélisés et d'échantillons positifs labélisés.

PU qui contient une fraction $\pi \in (0,1)$ d'échantillons positifs et une fraction $1-\pi$ d'échantillons négatifs x_N. Le système Positif-Non labélisé inclue trois modèles convolutifs avec différents rôles respectifs :

— Le modèle discriminateur D_U est entraîné à distinguer les vrais échantillons x_U des faux non labélisés générés x_F, avec $y_{D_U} \in (0,1)$ sa valeur de sortie prédite.

— Le modèle génératif G prend en entrée un vecteur de bruit z constitué de variables aléatoires continues, et fournit en sortie, dans le même format que x_U, les faux échantillons $x_F = G(z)$. G est entraîné de manière antagoniste à D_U afin de générer des faux échantillons tels que leur distribution $p(x_F)$ tend vers $p(x_U)$.

— Lors de la deuxième étape, une fois que l'entraînement du GAN est considéré comme terminé, le classifieur binaire convolutif D_B est entraîné à distinguer les échantillons réels positifs x_P des faux échantillons x_F.

Les explications présentées ci-dessous ont pour objectif de développer l'intuition derrière le système proposé.

Nous rappelons que le jeu de données non labélisé est composé d'une fraction π d'échantillons positifs x_P et d'une fraction $1-\pi$ d'échantillons négatifs x_N. Ainsi, si le GAN est correctement entraîné sur les échantillons non labélisés x_U, on peut en déduire que :

$$p(x_F) \xrightarrow[y_{D_U} \to \frac{1}{2}]{} p(x_U)$$

$$\Leftrightarrow p(x_F) \xrightarrow[y_{D_U} \to \frac{1}{2}]{} \pi\, p(x_P) + (1-\pi)\, p(x_N),$$

et l'on admet alors comme forte hypothèse pour la suite que $p(x_F) = \pi\, p(x_{FP}) + (1-\pi)\, p(x_{FN})$, avec :

$$\begin{cases} p(x_{FP}) \xrightarrow[y_{D_U} \to \frac{1}{2}]{} p(x_P) \\ p(x_{FN}) \xrightarrow[y_{D_U} \to \frac{1}{2}]{} p(x_N). \end{cases}$$

Lorsque $y_{D_U} \to \frac{1}{2}$, nous démarrons la deuxième étape du PGAN. Par ailleurs, un GAN n'est pas parfait dans son fonctionnement lorsqu'il se voit être appliqué à des tenseurs de grandes dimensions, ainsi :

$$p(x_{FP}) \neq p(x_P), \text{ et } p(x_{FN}) \neq p(x_N). \tag{1}$$

Il est donc alors possible d'estimer une distance d non nulle dans la fonction de coût du classifieur D_B, tel que :

$$d(p(x_P), p(x_F)) \Leftrightarrow \begin{cases} d(p(x_P), p(x_{FP})) \\ d(p(x_P), p(x_{FN})). \end{cases}$$

Mais, bien que calculée, la distance $d(p(x_P), p(x_{FP}))$ n'est pas exploitée dans l'application finale où nous traitons uniquement des échantillons réels avec le classifieur D_B. Ainsi, lorsque $p(x_{FN}) \xrightarrow[y_{D_U} \to \frac{1}{2}]{} p(x_N)$ et que D_B a été également correctement entraîné, nous obtenons l'équivalence :

$$d(p(x_P), p(x_{FN})) \Leftrightarrow d(p(x_P), p(x_N)). \tag{2}$$

Nous sommes donc capable de calculer la distance qui nous intéresse. En transférant ce raisonnement dans notre méthode PU, cela revient à affirmer les équivalences suivantes à la sortie de la fonction de coût L_{D_B} du classifieur D_B lorsque $y_{D_U} \to \frac{1}{2}$:

$$\begin{aligned} L_{D_B} &= \mathbb{E}_{x_P \sim p(x_P)} log D_B(x_P) \\ &+ \mathbb{E}_{z \sim p_z(z)} log[1 - D_B(G(z))] \end{aligned}$$

$$\begin{aligned} \Leftrightarrow L_{D_B} &= \mathbb{E}_{x_P \sim p(x_P)} log D_B(x_P) \\ &+ \mathbb{E}_{x_N \sim p(x_N)} log[1 - D_B(x_N)]. \end{aligned}$$

Ainsi, grâce à l'hypothèse de fonctionnement proposée ci-dessus on peut affirmer que la méthode PGAN devient similaire à un entraînement sur des échantillons positifs et négatifs respectivement labélisés, tout en s'éloignant d'un entraînement de type apprentissage PU, malgré le fait que le jeu de données que nous utilisons contienne uniquement des échantillons positifs labélisés et des échantillons non labélisés. De plus, intuitivement, calculer $d(p(x_P), p(x_{FP}))$ peut favoriser dans une certaine mesure, l'apprentissage des frontières de $p(x_P)$. Cependant, deux risques peuvent survenir avec cette méthode :

— Si les échantillons x_U contiennent majoritairement des échantillons x_P, alors il est possible que G ne soit plus apte à générer suffisamment de faux échantillons similaires aux échantillons x_N.

— Si G génère des faux échantillons ayant une distribution égale à celle des vrais échantillons, contrairement à l'inégalité 1, alors le PGAN deviendrait équivalent en termes de performances à un entraînement classique PU. Mais lorsque la dimensionalité des images à traiter devient large, alors ce risque disparaît. De plus, il peut être atténué en utilisant une structure pour le classifieur D_B dont les performances en prédiction sont meilleures que celles de la structure utilisée pour le discriminateur D_U.

Trouver une solution permettant d'éviter ces deux risques de se produire reste une question ouverte très intéressante pour améliorer la fiabilité de la méthode.

3 Expérimentations

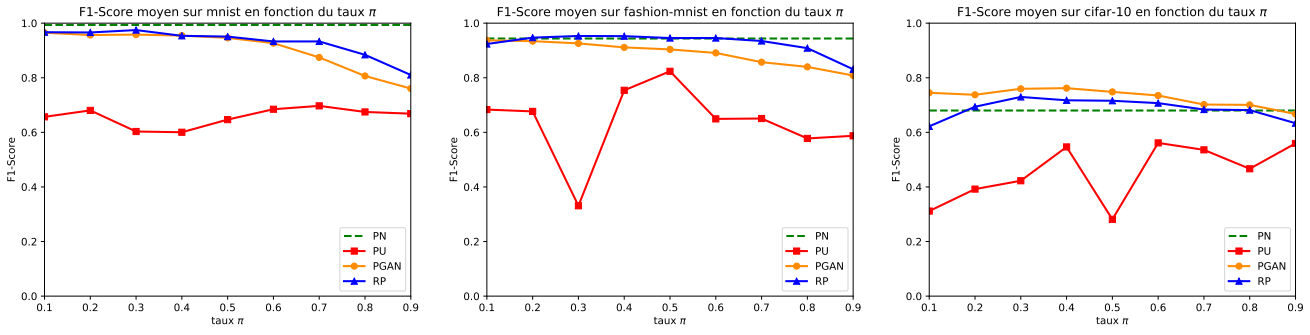
3.1 Réglages des tests

Les expériences ont été réalisées sur les trois jeux de données MNIST [10], Fashion-MNIST [21] et CIFAR-10 [9]. Nous avons comparé notre approche à RP [13], qui est à notre connaissance la meilleure méthode d'apprentissage bruité (noisy learning) et PU ne nécessitant pas de connaissances à priori de la fraction π. De plus, l'implémentation de l'auteur est disponible [2]. Nous indiquons aussi la performance du classifieur entraîné sur le jeu de données d'entraînement initial contenant des échantillons entièrement labélisés positifs et négatifs, et nous appelons évidemment cette méthode PN, que nous considérons comme la référence du cadre idéal. Nous comparons aussi le PGAN à un

2. *https : //github.com/cgnorthcutt/rankpruning*

TABLE 1 – Résultats comparatifs en fonction des F1-Scores mesurés sur MNIST, Fashion-MNIST et CIFAR-10 avec le classifieur entraîné sur 20 époques.

	ref	$\rho = 0.5, \pi = 0.1$			$\rho = 0.5, \pi = 0.3$			$\rho = 0.5, \pi = 0.5$			$\rho = 0.5, \pi = 0.7$		
Jeux de données	PN	PU	PGAN	RP	PU	PGAN	RP	PU	PGAN	RP	PU	PGAN	RP
0	0.997	0.633	0.974	**0.992**	0.445	**0.973**	0.955	0.320	0.973	**0.991**	0.689	**0.902**	0.880
1	0.998	0.774	0.971	**0.995**	0.642	0.979	**0.996**	0.851	0.958	**0.994**	0.884	0.863	**0.993**
2	0.990	0.395	0.972	**0.975**	0.658	**0.959**	0.923	0.795	**0.947**	0.936	0.692	0.914	**0.987**
3	0.996	0.716	0.963	**0.991**	0.620	0.953	**0.991**	0.766	**0.934**	0.882	0.729	**0.885**	0.829
4	0.997	0.512	0.964	**0.972**	0.802	0.952	**0.995**	0.717	**0.945**	0.933	0.551	0.914	**0.977**
5	0.993	0.701	0.974	**0.985**	0.725	**0.950**	0.943	0.799	**0.949**	0.910	0.626	0.873	**0.973**
6	0.992	0.708	**0.962**	0.928	0.758	0.959	**0.992**	0.699	0.971	**0.993**	0.613	0.944	**0.990**
7	0.995	0.603	**0.962**	0.947	0.433	0.960	**0.991**	0.620	0.926	**0.988**	0.783	0.737	**0.979**
8	0.995	0.741	**0.949**	0.929	0.506	0.941	**0.982**	0.339	0.922	**0.941**	0.651	**0.849**	0.818
9	0.981	0.785	**0.959**	0.954	0.442	0.956	**0.979**	0.561	0.939	**0.941**	0.750	0.865	**0.904**
AVG_{MNIST}	0.993	0.657	0.965	**0.967**	0.603	0.958	**0.975**	0.647	0.946	**0.951**	0.697	0.875	**0.933**
T-shirt/top	0.908	0.724	**0.926**	0.899	0.206	0.91	**0.937**	0.821	0.873	**0.947**	0.695	0.802	**0.91**
Trouser	0.993	0.815	0.983	**0.993**	0.247	0.969	**0.989**	0.938	0.953	**0.99**	0.681	0.911	**0.984**
Pullover	0.932	0.635	**0.9**	0.887	0.29	0.885	**0.925**	0.695	0.865	**0.917**	0.657	0.842	**0.888**
Dress	0.952	0.601	0.941	**0.948**	0.312	0.925	**0.955**	0.852	0.893	**0.914**	0.631	0.853	**0.882**
Coat	0.882	0.614	**0.909**	0.847	0.252	0.889	**0.942**	0.788	0.845	**0.92**	0.686	0.83	**0.918**
Sandal	0.995	0.793	0.945	**0.977**	0.444	0.964	**0.98**	0.819	0.923	**0.985**	0.67	0.919	**0.981**
Shirt	0.818	0.446	**0.852**	0.758	0.398	0.846	**0.847**	0.797	0.819	**0.873**	0.554	0.792	**0.853**
Sneaker	0.983	0.73	**0.973**	**0.973**	0.271	0.952	**0.979**	0.865	0.943	**0.967**	0.64	0.922	**0.977**
Bag	0.989	0.772	**0.978**	0.976	0.536	0.947	**0.99**	0.837	0.96	**0.965**	0.685	0.757	**0.977**
Ankle boot	0.985	0.704	0.964	**0.979**	0.354	0.973	**0.986**	0.824	0.963	**0.976**	0.609	0.942	**0.976**
$AVG_{\text{F-MNIST}}$	0.944	0.683	**0.937**	0.924	0.331	0.926	**0.953**	0.824	0.904	**0.945**	0.651	0.857	**0.935**
Plane	0.727	0.341	**0.818**	0.669	0.557	0.784	**0.795**	0.295	**0.758**	0.743	0.621	**0.731**	0.718
Auto	0.78	0.506	**0.801**	0.695	0.492	0.737	**0.829**	0.414	0.789	**0.798**	0.521	0.734	**0.783**
Bird	0.447	0.175	**0.688**	0.56	0.439	**0.744**	0.68	0.184	**0.694**	0.644	0.359	**0.688**	0.542
Cat	0.5	0.125	**0.658**	0.384	0.272	**0.722**	0.651	0.249	**0.718**	0.67	0.446	0.69	**0.698**
Deer	0.698	0.272	**0.68**	0.605	0.232	**0.708**	**0.708**	0.3	**0.708**	0.64	0.43	**0.633**	0.602
Dog	0.567	0.2	**0.632**	0.539	0.37	**0.756**	0.648	0.258	**0.746**	0.733	0.514	0.678	**0.712**
Frog	0.691	0.35	**0.837**	0.666	0.418	0.793	**0.794**	0.256	**0.788**	0.769	0.693	**0.75**	0.749
Horse	0.786	0.373	**0.693**	0.653	0.515	**0.757**	0.723	0.26	0.751	**0.759**	0.611	0.675	**0.711**
Ship	0.832	0.313	**0.821**	0.764	0.565	0.809	**0.831**	0.324	0.775	**0.785**	0.623	0.716	**0.755**
Truck	0.771	0.462	**0.822**	0.685	0.367	**0.786**	0.637	0.272	**0.754**	0.617	0.539	**0.724**	0.564
$AVG_{\text{CIFAR-10}}$	0.680	0.312	**0.745**	0.622	0.423	**0.760**	0.730	0.281	**0.748**	0.716	0.536	**0.702**	0.684



(a) (b) (c)

FIGURE 2 – F1-Scores moyens après 20 époques d'entraînement pour le classifieur en fonction du taux π qui varie entre 0.1 et 0.9 avec un pas de 0.1, pour PN (vert), PU (rouge), RP (bleu) et PGAN (orange) sur MNIST (a), Fashion-MNIST (b) et CIFAR-10 (c).

entraînement que l'on nomme PU, qui est équivalent à PN, mais avec une substitution des échantillons négatifs par des échantillons non labélisés.
Pour ces expérimentations, les méthodes PN, PU, RP et PGAN sont testées avec exactement le même classifieur convolutif afin d'être impartial. Nous avons utilisé le modèle convolutif de classification d'imagettes proposé par tensorflow [3] pour rester générique. Il contient deux couches convolutives successives, respectivement suivies d'une étape de max-pooling, et se finit par deux couches entièrement connectées consécutives. La fonction d'activation en sortie de chaque couche est ReLU, mis à part pour la dernière où softmax est appliquée. Nous avons uniquement modifié la dimension de sortie de la dernière couche que l'on fait passer de 10 neurones à 2, afin d'être adapté à notre tâche de classification binaire. Le classifieur est entraîné sur 20 époques. Pour les images 32x32x3 de CIFAR-10, les largeurs et longueurs des tenseurs d'entrée et de sortie des deux couches convolutives sont adaptées, et la profondeur des filtres à noyaux de la première couche convolutive est établie à 3 afin de correspondre aux trois cannaux de ces images RVB. Mais le nombre de filtres et leurs largeur et longueur restent inchangés.
Pour l'étape générative du PGAN, nous avons associé la méthode d'entraînement du WGAN [1] à l'architecture du DCGAN [15] en raison de leurs performances. À noter qu'avec la distance EM, $p(x_F)$ tend vers $p(x_U)$ lorsque y_{D_U} tend vers 0. Bien que cela ne soit pas une nécessité, dans le cadre de ces expériences, le vecteur de bruit d'entrée z est constitué de variables aléatoires continues de distributions uniformes. La durée d'entraînement de notre modèle génératif dépend de la complexité du jeu de données à traiter : 10 époques pour MNIST, 20 pour Fashion-MNIST, et 100 pour CIFAR-10. Pour ce dernier, nous faisons les mêmes modifications dans la structure de D_U et G, tel que cela a été expliqué précédemment pour le classifieur.
Au sujet de la création de notre jeu de données d'entraînement, ρ correspond à la fraction d'échantillons positifs du jeu de données initial qui contient n_P échantillons positifs. Ces $\rho \times n_P$ échantillons collectés sont ensuite introduits dans notre jeu de données non-labellisé U_{train}, qui contient initiallement uniquement des échantillons négatifs N dont le nombre total est n_N. π est la fraction d'échantillons positifs P que l'on impose dans le jeu de données non labellisé d'entraînement U_{train}. Pour se faire, nous retirons de U_{train} un certain nombre d'échantillons négatifs que nous n'exploitons pas, de manière à respecter π. U_{train} contient alors à la fois des N et des P selon les paramètres ρ et π. Nous établissons qu'avec $\pi \in [\frac{1}{\frac{n_N}{\rho\ n_P}+1}, 1)$ et $\rho \in (0, 1)$, nous pouvons alors obtenir consécutivement, avec P_{train} l'ensemble d'échantillons positifs d'entraînement, les deux jeux de données d'entraînement suivants :

$$P_{train} = \{(1-\rho)\ n_P\ P\ ;\ 0\ N\},$$

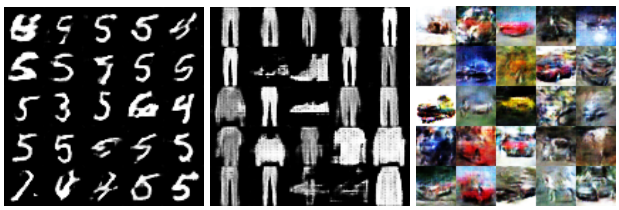
$$U_{train} = \{\rho\ n_P\ P\ ;\ \frac{1-\pi}{\pi}\ \rho\ n_P\ N\},$$

où les notations $a\ P$ et $b\ N$ désignent respectivement a éléments positifs et b éléments négatifs.
Pour trouver les équations définissant U_{train} selon les paramètres π et ρ, et l'intervalle des valeurs possibles pour π, nous utilisons n_U qui représente le nombre total d'échantillons non labélisés contenus dans l'ensemble U_{train}, tel que :

$$\begin{aligned} U_{train} &= \{\pi\ n_U\ P\ ;\ (1-\pi) n_U\ N\} \\ &= \{\rho\ n_P\ P\ ;\ (1-\pi)\frac{\rho\ n_P}{\pi}\ N\}, \end{aligned}$$

car nous imposons $\rho\ n_P = \pi\ n_U$.
Or, pour que cela soit réalisable, il faut que $(1-\pi)\frac{\rho\ n_P}{\pi}$ soit inférieur ou égal à n_N. Cela revient donc à dire que $\pi \in [\frac{1}{\frac{n_N}{\rho\ n_P}+1}, 1)$.
Les résultats présentés ci-dessous sont tous réalisés avec $\rho = 0.5$ et pour plusieurs valeurs de π.

3.2 Résultats



(a) (b) (c)

FIGURE 3 – Images générées par G avec $\rho = 0.5$ et $\pi = 0.5$ après 10 époques sur MNIST (a), 20 sur Fashion-MNIST (b), et 100 sur CIFAR-10 (c). Les classes positives respectives sont ici "5", "$trouser$" and "$automobile$".

Dans la figure 3, nous présentons quelques fausses images générées par G, respectivement pour MNIST, Fashion-MNIST et CIFAR-10. Nous pouvons remarquer que les images générées par G semblent visuellement acceptables, ce qui indique d'un point de vue qualitatif le bon fonctionnement du modèle génératif. Afin d'obtenir un tel résultat, plus les images sont grandes et complexes, et plus il faut entraîner le GAN sur un grand nombre d'époques.
Pour calculer le F1-Score, la fonction ArgMax est appliquée aux deux neurones de sortie du classifieur. Si l'indice du premier neurone est renvoyé par ArgMax, alors l'échantillon traité est classifié comme négatif. Sinon, il est considéré comme positif. De plus, étant donné que les jeux de données de test contiennent 9 fois plus d'échantillons négatifs que d'échantillons positifs, une fois toutes les prédictions de test réalisées, nous adaptons les proportions des échantillons négatifs à celle des positifs de manière à obtenir un F1-Score pertinent. Le tableau 1 montre une partie des F1-Scores comparatifs mesurés pour chaque classe pour chacun des trois jeux de données exploités, et respectivement pour chaque méthode testée. Sur Fig. 2, il peut être

3. *https : //github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/tutorials/mnist/mnist_softmax.py*

observé que la méthode PN est une bonne référence sur MNIST et Fashion-MNIST. Nous trouvons que l'efficacité de la méthode d'apprentissage PGAN est équivalente à celle de la méthode RP jusqu'à $\pi = 0.5$ sur MNIST et $\pi = 0.3$ sur Fashion-MNIST. Son efficacité décline ensuite un peu plus vite que pour RP, mais tout en conservant un score acceptable. Sur CIFAR-10 le F1-Score moyen est systématiquement meilleur pour notre méthode PGAN. Aussi, notre méthode présente de meilleurs résultats que la référence PN jusqu'à $\pi = 0.8$, ce qui est très intéressant. Cela est probablement dû au fait que les échantillons générés représentent une plus grande variété de distributions d'échantillons négatifs que celles inclues dans le jeu de données initial. De plus, le F1-Score du PGAN est significativement et systématiquement meilleur que la méthode PU sur l'ensemble des trois jeux de données, même avec seulement 10% d'échantillons positifs parmis les échantillons non labélisés, autrement dit avec $\pi = 0.1$.

La figure 4 présente l'étude de la robustesse de l'approche PGAN. Les figures 4.a et 4.b montrent que la méthode PGAN a comparativement à RP une meilleure stabilité dans son fonctionnement de manière à permettre de prédire plus facilement l'évolution de son F1-Score en fonction de π pour chaque classe du jeu de données. La figure 4.c nous montre que le classifieur se stabilise et converge après 10 époques d'entraînement. Pour réaliser l'histogramme de la figure. 4.d, nous avons récupéré la valeur du deuxième neurone de sortie du classifieur qui correspond à la probabilité prédite pour une image d'appartenir à la classe positive. On peut observer que les distributions respectives des échantillons de test positifs et négatifs estimées par le PGAN sont de forme gaussienne, ce qui est une caractéristique intéressante pour des applications réelles.

TABLE 2 – Stabilité moyenne des performances (F1-Scores) des méthodes PGAN et RP en fonction de π sur les jeux de données MNIST, Fashion-MNIST et CIFAR-10

jeux de données	PGAN	RP	$\frac{E_{RP}}{E_{PGAN}}$
MNIST	**0.00039**	0.00172	4.410
Fashion-MNIST	**0.00016**	0.00025	1.563
CIFAR-10	**0.00044**	0.00182	4.136

En complément aux figures 4.a et 4.b, le tableau 2 présente la quantification des robustesses réalisée pour RP et PGAN en fonction de π en ce qui concerne leurs performances de prédiction, pour chacun des trois jeux de données de test. Pour ce faire, nous avons lissé respectivement la courbe $s(\pi)$ de chaque classe représentant l'évolution du F1-Score en fonction de π. Les courbes lissées $\tilde{s}(\pi)$ ont été obtenues en appliquant un filtre moyen avec un noyau de taille 3. Ensuite, l'erreur quadratique moyenne MSE pour chaque classe est calculée entre $\tilde{s}(\pi)$ et $s(\pi)$ tel que ci-dessous, avec k le nombre d'échantillons de $\tilde{s}(\pi)$:

$$MSE = \frac{1}{k}\sum_{i=1}^{k}(\tilde{s}^{(i)} - s^{(i+1)})^2. \quad (3)$$

Puis, nous calculons les erreurs moyennes E_{PGAN} et E_{RP} pour chaque jeu de données, ainsi que le ratio E_{RP} : E_{PGAN}. On peut de cette manière constater que notre méthode a systématiquement un comportement plus stable, qui est d'un facteur 4 sur MNIST et CIFAR-10.

4 Conclusion

Ainsi, nous avons démontré que l'approche d'apprentissage PU proposée surpasse l'état de l'art sur les images RVB complexes du jeu de données CIFAR-10, et a un comportement plus stable sur l'ensemble des jeux de données testés jusqu'à une fraction acceptable π d'échantillons positifs dans le jeu de données non labélisé d'entraînement. Ces résultats sont en cohérence avec le raisonnement formulé et permettent ainsi d'envisager des applications PU sur des données de plus grandes dimensions. Le PGAN ne nécessite pas de connaissances à priori sur la fraction d'échantillons positifs non labélisés. Cependant, il reste à étudier plus en profondeur les risques de fonctionnement indiqués dans la section 1, afin de garantir un fonctionnement idéal pour cette approche.

L'optimisation du système peut se prolonger en testant d'autres récentes variantes du GAN tels que le BEGAN [3], le WGAN-GP [6], ou bien d'autres modèles génératifs de type auto-encodeurs variationnels (VAEs) par exemple, afin de généraliser l'approche proposée aux réseaux génératifs. Une autre idée peut être d'exploiter le vecteur latent z du GAN pour réaliser des opérations arithmétiques linéaires, tel que dans [4], afin de générer des faux échantillons dont on pourrait peut-être ainsi mieux gérer la distribution. Dans cette même idée, trouver un moyen d'exploiter les données positive labélisées pour la phase d'entraînement du générateur est envisagé.

Étant données les performances prometteuses obtenues, une future orientation certaine est d'étendre cette méthode à l'analyse de plus grandes images et donc permettre la réalisation de tâches plus complexes telles que la détection d'objets [19], [11], [17] ou la segmentation sémantique [2].

Références

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223, 2017.

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet : A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12) :2481–2495, 2017.

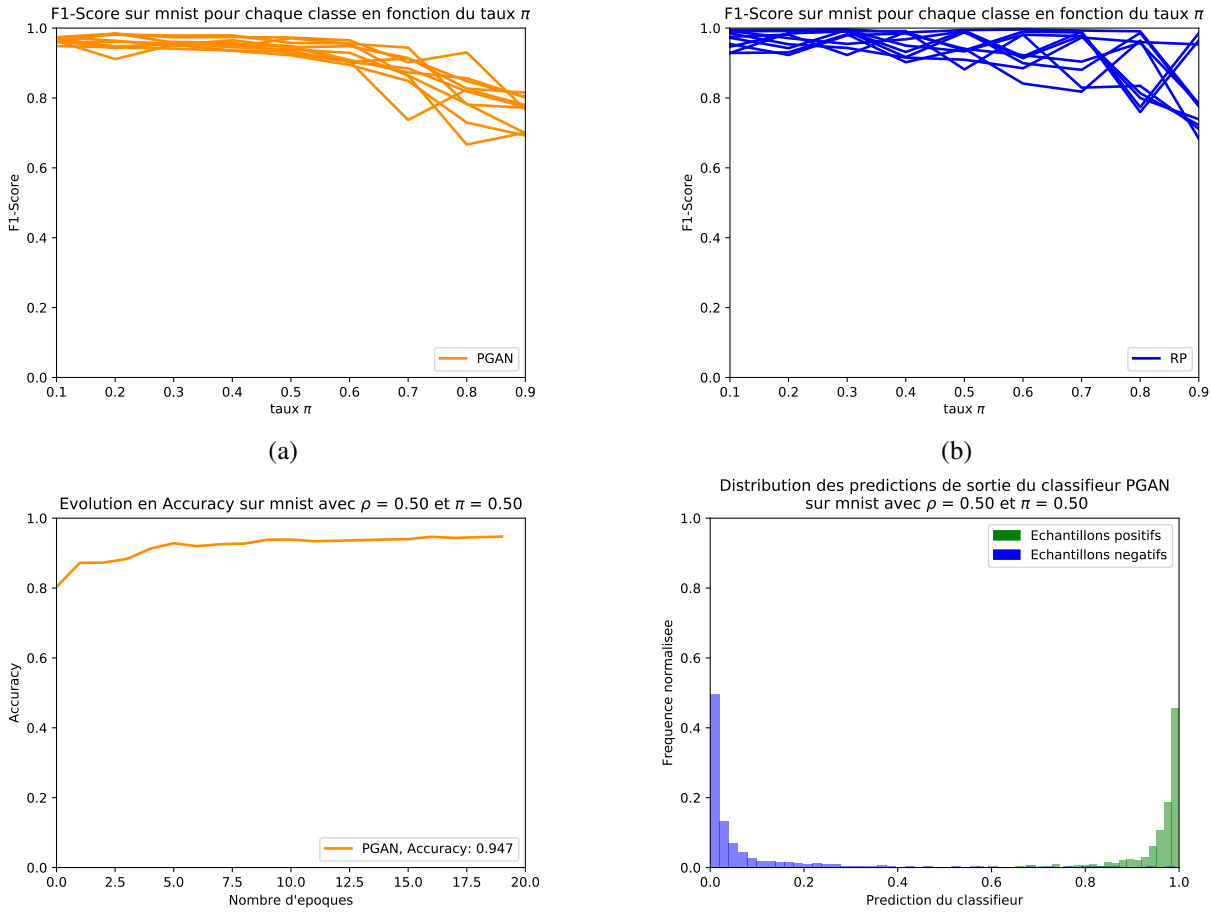


FIGURE 4 – Analyse de la robustesse sur MNIST. évolution du F1-Score pour chaque classe en fonction de π pour PGAN (a), et pour RP [13] (b). (c) montre l'évolution de l'Accuracy pendant l'entraînement du PGAN avec la classe positive "5" et $\pi = 0.5$. (d) est l'histogramme des distributions des valeurs de sortie du deuxième neurone du classifieur à son 20ème époque d'entraînement de (c) pour les échantillons de test positifs (vert) et négatifs (bleu).

[3] D. Berthelot, T. Schumm, and L. Metz. Began : Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv :1703.10717*, 2017.

[4] P. Bojanowski, A. Joulin, D. Lopez-Paz, and A. Szlam. Optimizing the latent space of generative networks. *arXiv preprint arXiv :1707.05776*, 2017.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[6] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5769–5779, 2017.

[7] M. Hou, Q. Zhao, C. Li, and B. Chaib-draa. A generative adversarial framework for positive-unlabeled classification. *arXiv preprint arXiv :1711.08054*, 2017.

[8] S. S. Khan and M. G. Madden. One-class classification : taxonomy of study and review of techniques. *The Knowledge Engineering Review*, 29(3) :345–374, 2014.

[9] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

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

[11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD : Single shot multibox detector. In *European Conference on Computer Vision*, pages 21–37. Springer, 2016.

[12] G. Niu, M. C. du Plessis, T. Sakai, Y. Ma, and M. Sugiyama. Theoretical comparisons of positive-unlabeled learning against positive-negative learning. In *Advances in Neural Information Processing Systems*, pages 1199–1207, 2016.

[13] C. G. Northcutt, T. Wu, and I. L. Chuang. Learning with confident examples : Rank pruning for robust classification with noisy labels. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence*, UAI'17. AUAI Press, 2017.

[14] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko. A review of novelty detection. *Signal Processing*, 99 :215–249, 2014.

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

[16] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, pages 3546–3554, 2015.

[17] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once : Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[18] J. Redmon and A. Farhadi. YOLO9000 : Better, Faster, Stronger. *arXiv preprint arXiv :1612.08242*, 2016.

[19] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN : Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[20] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[21] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist : a novel image dataset for benchmarking machine learning algorithms, 2017.