Construction d'espace latent pour la détection d'anomalies par apprentissage adversarial*

Elies Gherbi[1,2], Blaise Hanczar[1], Jean-Christophe Janodet[1] et Witold Klaudel[3]

[1]IBISC, Univ Evry, Université Paris-Saclay, 91025 Evry, France
[2]IRT SystemX, 8 avenue de la Vauve, 91120-Palaiseau
[3]Renault, 1 avenue du Golf 78288–Guyancourt

Résumé

La détection d'anomalies est un problème récurrent en *Machine Learning*. Des techniques récentes cherchent à exploiter le potentiel des GAN (*Generative Adversarial Networks*) pour détecter les anomalies de façon indirecte, dans l'espace des données ; elles se fondent sur l'idée que le générateur ne peut pas reconstruire une anomalie. Nous développons une approche alternative, basée sur un *Encoding Adversarial Network* (ANOEAN), qui projette les données dans un espace latent, où la détection d'anomalies se fait de façon directe, en calculant un score. Notre encodeur est appris par *adversarial learning*, en utilisant deux fonctions de perte, la première contraignant l'encodeur à modéliser les données normales dans un espace qui suit une distribution gaussienne, et la seconde, à projeter des données anormales en dehors de cette distribution. Nous conduisons une série d'expériences sur plusieurs bases standard, et montrons que notre approche dépasse l'état de l'art lorsqu'on utilise 10% d'anomalies lors de l'apprentissage.

Mots-clef : Détection d'anomalies, *Adversarial Learning*, GAN, EAN.

1 Introduction

La détection des anomalies a longtemps été une question de grand intérêt dans plusieurs applications [4, 9, 3]. On l'utilise dans des domaines comme la détection de fraudes, la cybersécurité, la vidéo-surveillance, la maintenance prédictive, *etc.* Les développements récents dans le *deep learning* ont permis de revisiter ce problème. Ainsi, selon la taxonomie de [8], il existe deux grandes familles de méthodes.

La première catégorie, *Representation learning*, consiste à apprendre une représentation des données normales qu'on utilise pour déclencher une alarme lorsqu'un comportement déviant est repéré. De nombreuses techniques d'apprentissage automatique ont été utilisées pour construire de tels modèles, dont les machines à vecteurs de support à une classe (OCSVM) [15] ; il s'agit dans ce cas de construire une boule (*cluster*) de faible rayon autour des données normales, et de décréter que toute anomalie est en dehors.

La deuxième catégorie, *Reconstruction based anomalie score*, part de l'hypothèse que les anomalies possèdent des caractéristiques différentes de celles des données normales. En conséquence, il est difficile de reconstruire une anomalie en utilisant un modèle appris pour la classe normale. Les *autoencoders* ont été utilisés en premier dans cette catégorie [18, 19], mais les techniques relevant du *adversarial learning* (GAN, BiGAN) sont de plus en plus populaires, plusieurs travaux ont adapté les réseaux génératifs adversariaux (GAN) pour la detection d'anomalies[14, 21, 20, 2, 8, 13, 6].

Un GAN (*Generative Adversarial Network*) est composé de deux réseaux : un générateur G, transformant les vecteurs z d'un espace latent en données artificielles, et un discriminateur D dont le rôle est de différencier les données artificielles des données réelles. On entraîne un tel modèle de façon concurrente : le générateur doit leurrer le discrimateur en apprenant à approximer la distribution des données réelles.

Dans le contexte de la détection d'anomalies, ANOGAN [14] et ses variantes [6], apprennent un GAN à partir de données normales, puis pour un exemple x de test, l'algorithme recherche un point z dans l'espace latent tel que $G(z) \approx x$; si on n'échoue à trouver un tel z,

*This work has been partially carried out in IRT SystemX, and therefore granted with public funds within the scope of the French program Investissements d'avenir. This work is part of Project "Cybersecurity for Intelligent Transport".

alors on déclare que l'exemple x est une anomalie. En d'autres termes, on cherche à calculer $z = G^{-1}(x)$ en partant du principe que G n'est inversible que pour les exemples normaux. Pour celà, on introduit une fonction de perte en reconstruction (*reconstruction loss*), typiquement $\mathcal{L}_r = \|x - G(z)\|$, qu'on cherche à minimiser en fonction de z. Cette technique est très coûteuse en temps car pour chaque exemple de test, on doit faire une descente de gradient.

Aussi, d'autres algorithmes comme EGBAD et ALAD [20, 21] se basent sur des BiGAN plutôt que des GAN, c'est-à-dire des architectures composées de trois réseaux : un générateur G, un discriminateur D ainsi qu'un encodeur E, dont le rôle est de projeter les données vers l'espace latent avec $E = G^{-1}$. Dans ce cas, chercher z tel que $G(z) \approx x$ est immédiat : il suffit de prendre $z = E(x)$. Mais en pratique, le générateur et l'encodeur sont rarement inverses l'un de l'autre [5], ceci peut conduire à des scores de reconstruction élevés pour des données normales.

Dans cet article, nous proposons une technique alternative, appelée AnoEAN pour *Anomaly detection with Encoding Adversarial Network*. L'idée est de travailler directement dans un espace latent et l'utiliser comme espace de décision.Pour se faire, nous entraînons un encodeur, par *adversarial learning*, à projeter les données normales dans une région restreinte de l'espace latent, et les anomalies en dehors de cette région. Nous supposons disposer d'une grande quantité de données normales, et d'un petit nombre d'anomalies. C'est un cadre de travail fréquent en détection d'anomalies. Ce faisant, nous éliminons les problèmes liés à l'estimation de la fonction de perte en reconstruction dont souffrent les GAN et BiGAN. Nous identifions les anomalies directement dans l'espace latent, à l'aide d'une distance de Mahalanobis calculée sur la distribution induite par les exemples normaux. Nous conduisons une série d'expériences prouvant qu'AnoEAN donne de meilleurs résultats que les techniques classiques de détection d'anomalies, notamment celles fondées sur les GAN, en utilisant à la fois la base MNIST de chiffres manuscrits [10] et deux bases de détection d'intrusions dans des réseaux (KDD'99, NSL-KDD) [12, 7].

2 Algorithme AnoEAN

Nous considérons un problème dans lequel nous surveillons les données $x \in \mathbf{R}^p$ décrivant l'état d'un système avec p variables. Notre objectif est de déclencher une alerte lorsque les données montrent que le système sort de son comportement normal. Pour cela, nous construisons un score d'anomalie $a(x) : \mathbf{R}^p \to [0, +\infty[$ mesurant le degré d'anomalie d'une donnée x, les données normales devant avoir un score proche de 0. Pour apprendre cette fonction, nous disposons d'un ensemble d'apprentissage contenant N données correspondant à un comportement normal $\{x_{n_i} \mid i = 1..N\}$ et M données correspondant à des anomalies $\{x_{a_i} \mid i = 1..M\}$ avec $M \ll N$. Nous notons P_{x_n} (resp. P_{x_a}) la distribution des données normales (resp. anomalies). A noter que contrairement aux données normales, les anomalies ne forment pas une classe homogène.

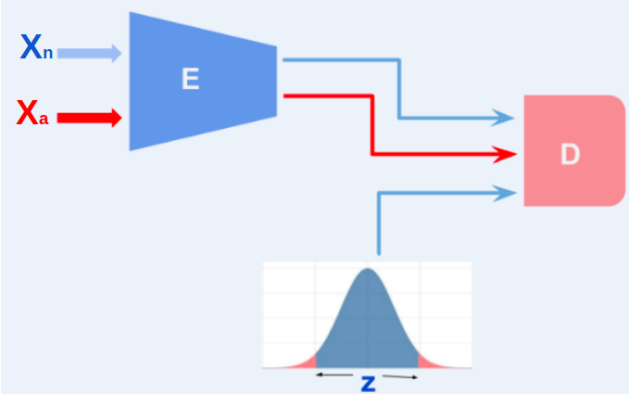

FIGURE 1 – Architecture de AnoEAN.

Notre méthode, nommée AnoEAN pour *Anormaly detection by Encoding Adversarial Network*, consiste à projeter les données x_n et x_a dans un espace de petite dimension, appelé espace de décision, dans lequel la distribution des données normales est gaussienne. Nous appelons *encodeur*, le réseau de neurones qui projette les données de l'espace d'origine dans l'espace de décision $E(x) : \mathbf{R}^p \to \mathbf{R}^d$ avec $d \ll p$. Soit p_z une distribution gaussienne $\mathcal{N}(0, I)$ dans l'espace de décision. L'objectif de l'encodeur est de projeter les donnnées normales dans p_z et les anomalies en dehors de p_z.

Pour cela, de la même manière que les GAN, nous utilisons un deuxième réseau appelé *discriminateur* qui reçoit en entrée un vecteur de l'espace de décision et prédit si ce vecteur provient de p_z ou de l'encodeur. L'encodeur doit donc à la fois tromper le discriminateur sur la projection de données normales pour lui faire croire qu'elle proviennent de p_z et aider le discriminateur à différencier p_z de la projection des anomalies. L'architecture de notre modèle AnoEAN est représentée dans la Fig. 1. Le discriminateur et l'encodeur doivent respectivement minimiser les fonctions de coût L_D et L_E suivantes :

$$
\begin{aligned}
L_D = & -\mathbf{E}_{z\sim p_z}[\log(D(z))] \\
& -\mathbf{E}_{x_n\sim p_{x_n}}[\log(1-D(E(x_n)))] \\
& -\mathbf{E}_{x_a\sim p_{x_a}}[\log(1-D(E(x_a)))]
\end{aligned}
$$

$$
\begin{aligned}
L_E = & -\mathbf{E}_{x_n\sim p_{x_n}}[\log(D(E(x_n)))] \\
& -\mathbf{E}_{x_a\sim p_{x_a}}[\log(1-D(E(x_a)))]
\end{aligned}
$$

L'apprentissage de ce modèle suit la procédure suivante. On prend aléatoirement des exemples normaux x_n et des anomalies x_a de la base d'apprentissage et on les projette dans l'espace de décision avec l'encodeur (*i.e.*, on calcule chaque $E(x_a)$ et $E(x_n)$). On ajoute à ces exemples projetés des vecteurs aléatoires tirés de la distribution p_z afin de former un *batch* que l'on présente à l'entrée du discriminateur. Le discriminateur est modifié par descente du gradient afin de minimiser L_D ; l'encodeur est gelé pendant cette étape. Ensuite c'est au tour de l'encodeur d'être modifié afin de minimiser L_E, le discriminateur étant gelé pendant cette étape. Cette procédure est itérée jusqu'à convergence du modèle.

Une fois l'apprentissage du modèle terminé, la prédiction des anomalies ne nécessite que l'utilisation de l'encodeur. Le score d'anomalie d'un exemple correspond à la distance entre sa projection dans l'espace de décision $E(x)$ et la distribution de la projection des exemples normaux. Cette distribution est censée tendre vers $p_z = \mathcal{N}(0, I)$ au cours de l'apprentissage du modèle. Nous pourrions donc utiliser la norme de $E(x)$ comme score d'anomalie. Cependant, nos expérimentations ont montré que la distribution de la projection des exemples normaux pouvait diverger légèrement de p_z. Aussi, à la fin de l'apprentissage, nous représentons cette distribution par une distribution gausienne $\mathcal{N}(\mu, \Sigma)$ dont nous estimons les paramètres sur une base de validation. Le score d'anomalie est finalement la distance de Mahalanobis entre $E(x)$ et μ :

$$a(x) = \sqrt{(E(x)-\mu)^T\Sigma^{-1}(E(x)-\mu)}$$

3 Expérimentations et résultats

Nous présentons dans cette partie les expérimentations que nous avons conduites et leurs résultats montrant l'efficacité de notre méthode comparée à l'état de l'art.

3.1 Protocole expérimental

Dans nos expérimentations nous avons utilisé trois jeux de données : MNIST, KDD99 et NSL-KDD. MNIST [10] est une base de chiffres manuscrits qui est courament utilisée pour les problèmes de classification d'images. Bien que MNIST ne contient pas à l'origine de classe normale et d'anomalies, elle est souvent utilisée en détection d'anomalies, afin d'analyser le comportement des algorithmes et de visualiser leurs prédictions. Dans nos expérimentations, nous avons utilisé ce jeu de données de deux façons :
1) **1 contre tous** : nous considérons qu'un certain chiffre représente la classe normale et les 9 restant sont des anomalies. Pour les méthodes OCSVM, AnoGAN, ALAD, EGBAD, l'ensemble d'apprentissage est constitué de 5000 données normales ; l'ensemble de test comprend 1700 exemples dont 80% de données normales et 20% d'anomalies choisies aléatoirement parmi les 9 autres classes. Pour AnoEAN et SVM, nous injectons en plus 10% d'anomalies dans l'ensemble d'apprentissage ; celles-ci sont choisies parmi 4 classes sur 9, de sorte que certains chiffres (anomalies) n'apparaissent pas pendant l'apprentissage.
2) **n contre m** : on se base sur le même principe que le "1 contre tous", la seule différence étant que la classe normale est composée de plusieurs chiffres. Nous regroupons n classes de chiffres en une seule classe normale et traitons les m chiffres restants comme des exemples anormaux. Nous utilisons les mêmes répartitions d'exemples dans les ensembles d'apprentissage et de tests que précédemment. L'objectif est d'analyser le comportement des algorithmes dans le cas où la classe normale est hétérogène et peut se décomposer en plusieurs sous-classes.

Les deux autres jeux de données utilisés, KDD99 [12] et NSL-KDD [7], sont très courament utilisés pour évaluer les performances des algorithmes de détection d'anomalies, l'objectif étant de détecter des intrusions dans des réseaux surveillés. Nous utilisons les mêmes répartitions d'exemples dans les ensembles d'apprentissage et de tests que précédemment.

Nous comparons notre méthode avec deux approches classiques et trois méthodes basées sur les GAN.

OCSVM. Les *one-class* SVM permettent de construire un cluster à partir de données normales [15]. Nous utilisons un noyau RBF. Nous les avons également essayés avec quelques anomalies et une marge douce de 10%, mais cela n'aporte pas d'amélioration aux résultats.

SVM. Les SVMs sont sensibles aux ensembles de données déséquilibrés [16, 17, 1]. Donc de façon clas-

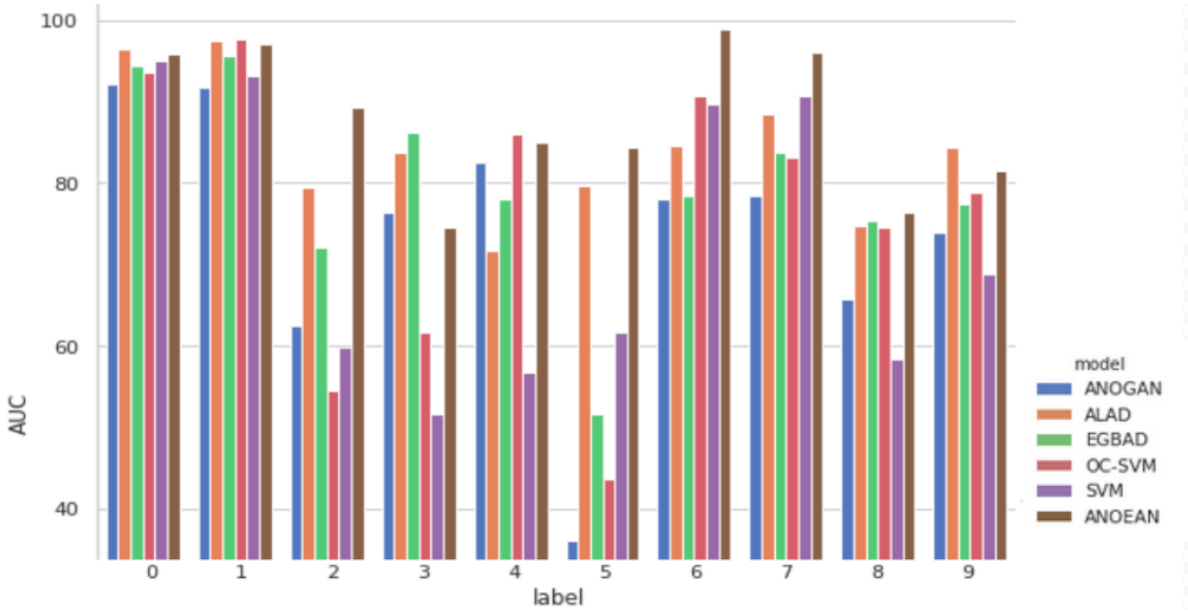

FIGURE 2 – Performance des modèles suivant la métrique AUC sur chaque chiffre dans le cas du problème MNIST [1 contre tous]

sique [16], nous avons rééquilibré les classes en donnant un poids plus élevé à la classe la moins fréquente (celle des anomalies).

AnoGAN. AnoGAN est utilisé avec les mêmes paramètres que les auteurs de [14]. Le critère de détection d'anomalies tient compte de la fonction de perte en reconstruction, et des sorties de la dernière couche cachée du discriminateur.

EGBAD. L'encodeur est utilisé pour projeter une donnée test x en un point z de l'espace latent, puis le générateur pour reconstruire une donnée artificielle x' à partir de z. Le critère d'anomalie compare x et x' d'une façon similaire à AnoGAN [20].

ALAD. Basé sur l'architecture ALICE [11], ALAD [21] améliore EGBAD en forçant explicitement l'encodeur et le générateur à être inverse l'un de l'autre au cours de l'apprentissage (par *cycle consistency*). C'est un des discrimateurs d'ALAD qui est utilisé pour détecter les anomalies.

La détection d'anomalies avec OCSVM, ALAD, AnoGAN et EGBAD sont des méthodes *one-class*. On utilise uniquement des données normales dans la phase d'apprentissage. Dans AnoEAN et SVM, on introduit en plus quelques exemples d'anomalies ce qui entraîne un déséquilibre des classes.

Les performances des algorithmes sont évaluées principalement par leur aire sous la courbe précision-rappel (AUC). Nous calculons également la mesure F1 optimale, ainsi que le taux de fausse alarme et le rappel associé. Pour chaque modèle, nous sélectionnons les paramètres d'apprentissage qui maximisent l'AUC calculée à partir du score d'anomalie $a(x)$ établi avec un ensemble de validation.

AnoEAN possède deux réseaux de neurones, l'encodeur et le discriminateur. Dans le cas de MNIST, l'encodeur est un réseau convolutif $(4*2*32, 4*2*64, 4*2*128, d)$ et le discriminateur est un réseau multi-couches $(32, 16, 1)$. Pour les bases NSL-KDD et KDD99, l'encodeur et le discriminateur sont des réseaux multi-couches $(128, 64, 32, d)$ et $(32, 16, 1)$ respectivement. La taille des *batch* est de 200. Le nombre d'*epoch* est fixé par *early stopping* et la descente du gradient sera avec Adam ($lr = 10^{-3}$).

3.2 Les résultats

Dans la Fig. 2, nous présentons nos résultats sur MNIST [1 contre tous] : chaque chiffre est successivement considéré comme la classe normale, les autres étant vus comme des anomalies. AnoEAN surpasse toutes les autres méthodes dans 5 cas sur 10 ; il a des performances comparables aux meilleures méthodes sur les 5 chiffres restants. Remarquons que dans l'absolu, toutes les méthodes ont des performances faibles sur les classes 2, 3, 4 et 5, car ces classes ont des caractéristiques visuelles similaires à celles des anoma-

lies. Nous observons aussi que les approches à base de reconstruction (ALAD et EGBAD) sont compétitives. Elles réussissent à recontruire correctement les chiffres, ce qui leur permet d'avoir un score d'anomalie important sur les éléments qui ne sont pas visuellement similaires à la classe normale.

Dans la Fig. 3, nous montrons les résultats sur MNIST [n contre m]. Nous nous intéressons au problème de l'identification des anomalies dans un ensemble de données où la classe normale est hétérogène (composée de plusieurs chiffres). Dans cette configuration, AnoEAN surpasse largement l'état de l'art, même si ses performances se dégradent avec l'hétérogénéité croissante de la classe normale. On remarque que les performances des approches à base de reconstruction (ALAD et EGBAD) s'effondrent. En effet, pour qu'elles fonctionnent, il est nécessaire que le générateur et l'encodeur soient inverses l'un de l'autre, ce qui est particulièrement difficile à assurer lorsque la classe normale est hétérogène. En revanche, AnoEAN réussit à encoder des distributions complexes car il ne necessite pas de calculer l'inverse de l'encodeur.

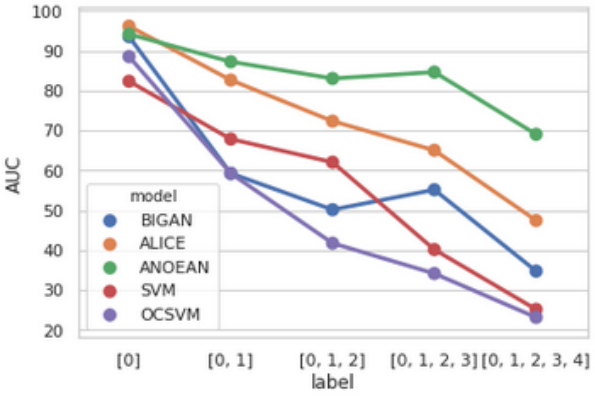

FIGURE 3 – Performances des différentes méthodes dans le cas du problème MNIST[n contre m]. La classe normale est constituée de 1 à 5 chiffres.

Les résultats sur les bases NSL-KDD et KDD99 sont présentés dans les Tables 1 et 2 . Outre l'AUC, nous montrons le taux de fausses alarmes (FA) qui est la proportion de fausses anomalies par rapport au nombre d'anomalies détectées, et le rappel qui est la proportion d'anomalies détectées par rapport à l'ensemble des anomalies de la base de test. Concernant le taux de fausses alarmes, la méthode SVM est celle qui a les meilleurs résultats sur KDD99 (car c'est un problème de classification), mais AnoEAN a des performances assez proches. Concernant le rappel, AnoEAN surpasse les autres méthodes, car nous avons explicitement codé dans sa fonction de coût le fait qu'il devait projeter toute anomalie en dehors de la zone des données normales dans l'espace latent.

AUC	F1	FA	Rappel	Modèle
79.6	80.5	19.6	79.6	AnoGAN
96.0	92.1	8.0	92.2	EGBAD
94.7	87.8	13.5	89.2	ALAD
94.1	87.9	13.1	89.0	OCSVM
97.5	**94.8**	**5.0**	94.6	**AnoEAN**
97.3	93.3	8.5	**95.2**	SVM

TABLE 1 – Résultats des différentes méthodes sur la base NSL-KDD

AUC	F1	FA	Rappel	Modèle
72.9	73.4	23.3	70.1	AnoGAN
89.1	93.9	5.5	93.3	EGBAD
95.4	96.6	3.6	96.6	ALAD
86.0	94.3	7.4	96.2	OCSVM
98.9	97.6	3.9	**99.0**	**AnoEAN**
98.6	98.7	**0.6**	97.6	SVM

TABLE 2 – Résultats des différentes méthodes sur la base KDD99

4 Conclusions et perspectives

Nous avons présenté une nouvelle méthode de détection d'anomalies qui utilise une projection dans un espace latent de faible dimension dans lequel exemples normaux et anomalies sont séparés. Pour cela, nous entrainons un encodeur par *adversarial learning*. La fonction de coût permet de modéliser les données normales dans dans un espace qui suit une distribution gaussienne, et les rares exemples d'anomalies en dehors de cette distribution. Notre méthode est moins couteuse, en terme de temps d'apprentissage, que les méthodes classiques basées sur les GAN, car elle ne nécessite pas d'apprendre un générateur. Pour la prédiction, notre méthode est également beaucoup plus rapide et légère en mémoire que les autres car elle necessite uniquement d'utiliser l'encodeur, ceci est un avantage pour toute application dans des systèmes embarqués. Nous avons montré expérimentalement que notre modèle peut faire face à des types d'anomalies non rencontrés au moment de l'apprentissage. De plus, notre technique a des meilleures performances que

l'état de l'art lorsqu'elle est confrontée à une classe normale hétérogène. Enfin, sur les bases habituelles utilisées en détection d'attaques dans des réseaux, notre technique déclenche peu de fausses alarmes. On note que les modèles entraînés d'une façon adversarial souffrent de problème de convergence, des nouvelles fonctions de coût et régularisation peuvent être rajouté pour stabilisé l'apprentissage. Il existe aussi un autre phénomène dit *Mode collapse*(problème d'effondrement), dans notre cas le *mode collapse* n'est pas contraignant, car cela signifie que l'encodeur projette toutes données normale dans un point point dans l'espace latent et et toutes données anormale dans un autre point éloigné dans le même espace, Dans se cas la distance entre les exemples normaux et les anomalies restera importante (score important). Parmi les pistes d'amélioration, nous travaillons à la suppression de tout exemple d'anomalie lors de l'apprentissage. Pour ce faire, une idée est d'apprendre un générateur d'anomalies, en utilisant l'encodeur et le discriminateur actuels comme adversaires.

Références

[1] R. Akbani, S. Kwek, and N. Japkowicz. Applying support vector machines to imbalanced datasets. In *Proc. ECML 2004*, pages 39–50, 2004.

[2] S. Akcay, A. A. Abarghouei, and T. P. Breckon. Ganomaly : Semi-supervised anomaly detection via adversarial training. *CoRR*, abs/1805.06725, 2018.

[3] R. Chalapathy and S. Chawla. Deep learning for anomaly detection : A survey. *CoRR*, abs/1901.03407, 2019.

[4] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection : A survey. *ACM Comput. Surv.*, 41(3) :15 :1–15 :58, 2009.

[5] A. Creswell and A. A. Bharath. Inverting the generator of A generative adversarial network (II). *CoRR*, abs/1802.05701, 2018.

[6] L. Deecke, R. A. Vandermeulen, L. Ruff, S. Mandt, and M. Kloft. Image anomaly detection with generative adversarial networks. In *Proc. ECML 2018*, pages 3–17, 2018.

[7] L. Dhanabal and S. Shantharajah. A study on NSL-KDD dataset for intrusion detection system based on classification algorithms. 2015.

[8] I. Golan and R. El-Yaniv. Deep anomaly detection using geometric transformations. In *Proc. NIPS 2018*, pages 9781–9791, 2018.

[9] B. R. Kiran, D. M. Thomas, and R. Parakkal. An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos. *J. Imaging*, 4(2) :36, 2018.

[10] Y. LeCun. The MNIST database of handwritten digits. 1998.

[11] C. Li, H. Liu, C. Chen, Y. Pu, L. Chen, R. Henao, and L. Carin. ALICE : towards understanding adversarial learning for joint distribution matching. In *Proc. NIPS 2017*, pages 5501–5509, 2017.

[12] M. Lichman. UCI machine learning repository,. 2013.

[13] M. Sabokrou, M. Khalooei, M. Fathy, and E. Adeli. Adversarially learned one-class classifier for novelty detection. In *Proc. CVPR 2018*, pages 3379–3388, 2018.

[14] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. *CoRR*, abs/1703.05921, 2017.

[15] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt. Support vector method for novelty detection. *Advances in neural information processing systems*, 12(3) :582–588, 2000.

[16] K. Veropoulos, C. Campbell, and N. Cristianini. Controlling the sensitivity of support vector machines. *Proc. IJCAI 1999*, 1999.

[17] G. Wu and E. Y. Chang. Adaptive feature-space conformal transformation for imbalanced-data learning. In *Proc. ICML 2003*, pages 816–823, 2003.

[18] Y. Xia, X. Cao, F. Wen, G. Hua, and J. Sun. Learning discriminative reconstructions for unsupervised outlier removal. In *Proc. 2015 IEEE International Conference on Computer Vision (ICCV'15)*, pages 1511–1519, 2015.

[19] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, J. Chen, Z. Wang, and H. Qiao. Unsupervised anomaly detection via variational auto-encoder for seasonal KPIs in web applications. *CoRR*, abs/1802.03903, 2018.

[20] H. Zenati, C. S. Foo, B. Lecouat, G. Manek, and V. R. Chandrasekhar. Efficient GAN-based anomaly detection. *CoRR*, abs/1802.06222, 2018.

[21] H. Zenati, M. Romain, C. Foo, B. Lecouat, and V. Chandrasekhar. Adversarially learned anomaly detection. In *Proc. ICDM 2018*, pages 727–736, 2018.