Creating Ensembles of Generative Adversarial Network Discriminators for One-class Classification

Mihai Ermaliuc[1], Daniel Stamate[1], George D. Magoulas[2], Ida Pu[1]

[1] Data Science & Soft Computing Lab, and
Department of Computing,
Goldsmiths, University of London

[2] Department of Computer Science and Information Systems,
Birkbeck, University of London

Abstract. *We introduce an algorithm for one-class classification based on binary classification of the target class against synthetic samples. We use a process inspired by Generative Adversarial Networks (GANs) in order to both acquire synthetic samples and to build the one-class classifier. The first objective is achieved by leading the generator's output into close vicinities of the target class region. For the second objective, we obtain a one-class classifier by generating an ensemble of discriminators obtained from the GAN's training process. Our approach is tested on publicly available datasets producing promising results when compared to other methods.*

Keywords: One-Class Classification, Generative Adversarial Networks

1 Introduction

Classification is a major topic in machine learning, dealing with algorithms that learn to assign pre-defined discrete values (categories, classes) to unseen objects (vectors), based on a dataset of observed objects with known classes. From the perspective of the number of classes involved, one could identify three categories. Two-class, or binary classification, is arguably the most intensively studied with a wide range of methods available. Given a dataset of observations from two classes, an algorithm will learn to separate them by finding a decision boundary in the feature space that minimizes a cost function. Secondly, when more than two classes exist, the so-called multi-class classification problem can be reduced to multiple binary classification problems [1]. Finally, if the algorithm learns using observations from a single class only, we have one-class classification (OCC). In this case the algorithm will also learn a decision boundary, this time trying to separate the known class from everything else.

One-class classification stands out as different, more challenging and more versatile than its binary or multi-class counterparts [2]. In the latter cases we have reasonably balanced observations from two or more classes. This helps an algorithm identify those feature values that are distinctive to each class and use them to find an optimal separation boundary between them. One downside of this is that the final product will only

distinguish between classes seen during the learning phase. If presented with observations coming from a new class, it would wrongly categorize them as one of the known classes [3]. Conversely, in one-class classification we have information from one class only and the algorithm therefore must learn a decision boundary that separates the known class from the rest of the world. This allows it to work well regardless of what classes of objects it is presented with at runtime. On the other hand, finding a decision boundary becomes more difficult [2].

As categorized by [4], one-class classification methods fall into 3 large categories. Density methods make assumptions on the probability density function (PDF) of the known class and use a threshold value to accept or reject new samples. Reconstruction methods learn a mapping of data to and from a latent space using the available target class data. For new observations it is expected that the reconstruction error will be low if the object belongs to the class, and high otherwise. Boundary methods estimate a border around the observed single-class data and perform classification by evaluating the distance from new observations to that border.

From a training data perspective, traditional algorithms like SVDD [5] perform reasonably well on low-dimensional datasets but degrade or become intractable in higher dimensions. With the advent of deep learning, reconstruction methods based on autoencoders have been researched intensively and they achieve relatively good results on such datasets. Generative adversarial networks [6] have also been employed to improve properties of the latent space in autoencoder-based methods [7] or as a means to train the discriminator as a one-class classifier [8].

In this paper we propose an adversarial training framework for progressively building a one-class classifier. We employ a generative model to create out-of-class data in a region as near to the in-class data space as possible. We then create a discriminator that separates in-class from out-of-class data. In the next step we modify the generative model to change its output region, while keeping close to the in-class space. We repeat this several times and ensemble all obtained discriminators together. The resulting classifier achieves promising performance in one-class classification when compared to other methods.

2 Related work

Generative Adversarial Networks (GANs) [6] sparked a lot of interest in adversarial training, albeit they focused mostly on the creation of generative models. There is, however, significant work carried out on the classification side as well. To the best of our knowledge, [9] and [10] are the first attempts to describe GANs use in semi-supervised learning tasks. Given a dataset of K classes, the discriminator is trained as a (K+1)-class classifier, where the extra class contains synthetic images. This can result in increased classification performance compared to traditional classifiers, particularly when the number of class samples is small. Regarding one-class classification, [8] uses a pre-trained GAN for anomaly detection in medical images by comparing a query image versus the closest similar image that can be obtained via the generative process. In

[7] an encoder-decoder novelty detection architecture is augmented with adversarial training of generative processes to minimize pockets of out-of-class samples in the latent space. The one-class classification algorithm proposed in [7], called OCGAN, seeks to minimize the chances of drawing false positives from the latent space. Autoencoders learn to map the data manifold by looking at discrete samples drawn from it. There is however an infinite number of paths to transition between any two such points and these transitions may pass through points that do not belong to the manifold. Even worse, these points may represent shapes that are out-of-class but occur in the real world. A relevant example given by the authors of [7] is a transition between 2 shapes representing digit 8 that passes through a shape representing digit 1. These cases will normally be labelled by the system as belonging to the target class, hence they are false positives.

To overcome the above issue, additional constraints are placed on the latent space of the autoencoder to enforce the representation of in-class samples only. Firstly, the uniform distribution of the target data onto the latent space is enforced. This is done by bounding the latent space between (-1, +1) and by training the encoder E in an adversarial manner against a latent discriminator, using uniformly distributed data as the real dataset. Secondly, it is enforced that the decoded version of samples drawn from the latent space resemble the target dataset. This is done by training the decoder, D, adversarially versus a visual discriminator and target data samples. Lastly, another mechanism reduces regions from the latent space that produce out-of-class examples. Such regions are actively searched for by using a discriminator to identify the most unlikely samples drawn uniformly and decoded from the latent space, versus encoded-decoded samples from the target data. Once this discriminator is trained, gradient descent is used to find the zones.

3 A novel approach to one-class classification using GANs

3.1 Motivation

The problem of one-class classification can also be formulated as the classification of the target class against *everything else*. While this idea is not challenged by any framework to our knowledge, the focus so far has still been on the target class to inform the build of a one-class classifier. We aim to shift this focus to the '*everything else'* part and we can do so by our choice of method: we generate out-of-class samples around the in-class subspace, then we create our one-class classifier as a binary classifier between in-class and out-of-class samples (see Figure 1).

Generating the out-of-class samples is not trivial though. The quality of the model depends on the quality of these samples. Intuitively, we would require them to reside in the feature space as closely as possible to the real data manifold and surround it completely, thus constraining the classifier to build a tight border around the in-class space. To generate such samples, we turn to GANs and make use of one of their otherwise undesirable characteristics – the difficulty to achieve convergence. GANs are arguably the most successful generative models currently available, but tend to converge only on very narrow ranges of hyperparameters, while in the general case they either

end up in mode collapse or cycle indefinitely through output from outside the real data manifold [11]. Our hypothesis is that a non-converging GAN can still get close to that manifold and thus might be used as a provider of good quality samples for our binary classifier.

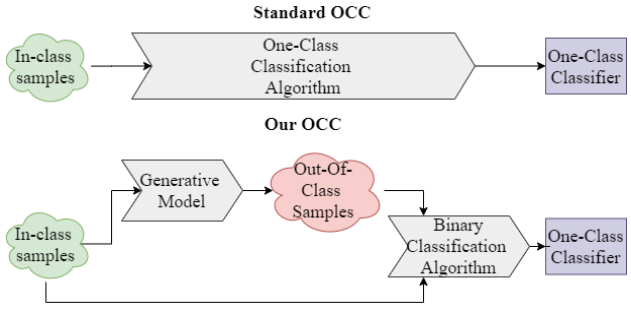

Figure 1. The top side illustrates the traditional process of building a one-class classifier, considering in-class data only. The bottom side illustrates our approach: we use the in-class samples to derive out-of-class samples then we create the one-class classifier as a binary classifier.

Lastly, we consider using the GAN's discriminator to build the classifier. The traditional recipe for successful GAN training is to train the generator and discriminator alternatively for one step each. Also, the discriminator is supposed to provide only a weak separation between generated and real data at any step, such that the generator will be able to follow it easily. Moreover, the discriminator is being reused at each training step by updating its current position. All these make sense since the discriminator's role is to drive the generator's output towards the real data and a crisp separation between the two datasets would harm this goal by failing to provide the generator with reliable gradients [12]. In other words, the discriminator's border needs to pass through or be close to the generated samples.

On the other hand, there is no such constraint related to the real samples. We could therefore encourage the discriminator's border not to pass through their space. This would effectively make the discriminator a one-class classifier for our data with low type-2 error (low number of false negatives) and presumably high type-1 error (high number of false positives), while the discriminator would still be able to provide gradients to drive the generator. If we repeat this process throughout the model's training and we save the discriminator at each step then we end up with several classifiers, each separating the real data from (hopefully) a different region of space. By combining these classifiers together in an ensemble, we obtain our one-class classifier (see Figure 2). The next sub-section provides more details on the proposed algorithm named GANOCC – which is a Generative Adversarial Network based algorithm for One-Class Classification.

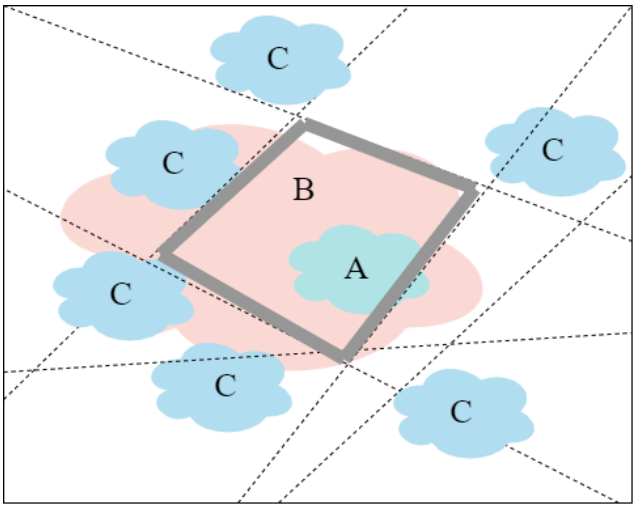

Figure 2. Illustration of our one-class classifier building process in a bidimensional feature space. A is our target class, B is real data we are interested in separating from A (invisible during training). A generative model creates a cloud of synthetic data C in a region near A. We build a classifier (dotted lines) to separate C and A. We use the classifier to further train the generative model so that it moves output past its border, and repeat. We combine all classifiers into a one-class classifier (the thick central polygon)

4 The GANOCC Algorithm

The algorithm we propose here is based on the following components: a generative model *G*, a variable number of discriminative models *D* and an ensemble *C* of all *D* models, which forms our one-class classifier. Training is performed in an adversarial framework inspired by GANs, but with some important changes. *G* is trained against *C*, a D_k is created at every epoch k and trained against *G*, then appended to the ensemble *C*. The paragraphs below provide detailed explanations.

The generator's role is to create synthetic samples. It is a vanilla DCGAN with transpose convolutional layers as defined by [13]. Training is performed against *C* with the objective to minimize:

$$\mathcal{L}_G = -E_{z\sim p_z}[\log(C(G(z)))]$$

For each training step, *G* will move its output into a new region of space. If we perform only one such step per epoch though, that region will overlap significantly with the previous region and only few novel samples will be generated. We try to reduce the overlap by training at each epoch until the loss crosses below a threshold. In the experiments reported in the next section, the threshold is 1.

The discriminators D_k are created one for each epoch and trained to separate data generated during the epoch from real data. Turning again to empirical observations, simpler models tend to provide better results in the end. We settled for a neural network

with one hidden layer and 8 neurons with LeakyRelu activations while the output's activation function is the identity. We add the constraint that each D_k must focus on classifying the real data as positive with low rejection rate. This is implemented indirectly by minimizing an objective function that places higher weight on the loss of positive samples:

$$\mathcal{L}_{D_k} = -\alpha_{data} E_{x\sim p_{data}}\left[\log\left(\sigma\left(D_k(x)\right)\right)\right] - \alpha_z E_{z\sim p_z}[\log\left(1-\sigma(D_k\left(G(z)\right))\right)]$$

where σ is the sigmoid function while α_{data} and α_z are positive real numbers scaling the influence of loss on in-class and out-of-class samples, respectively.

Training follows the same strategy as for the generative model *G*: at each epoch k we train D_k until the loss on real data and synthetic data drops below a threshold. We found 0.1 and 1, respectively, to be reasonable values in our experiments. The thresholds for *G* and D_k appear to be correlated. For instance, if we threshold D_k but train *G* for a single step/epoch then *G* will eventually be overpowered and stop making any progress.

The ensemble discriminator *C* is a composition of all D_k models created during training. We consider the ideal-case composition to be:

$$C(x) = \prod_{k=1}^{N_D} T_k(\sigma(D_k(x)))$$

where T_k is a threshold function defined as:

$$T_k(x) = \begin{cases} 1, x \geq t_k \\ 0, x < t_k \end{cases} \quad with \quad t_k \epsilon (0,1)$$

and t_k is the threshold value chosen such that

$$\sigma\left(D_k(x)\right) < t_k, \forall\, x \in X_s\ (synthetic\ data)$$

and

$$\sigma\left(D_k(x)\right) \geq t_k, \forall\, x \in X_r (real\ data)$$

and $N_D$is the total number of iterations.

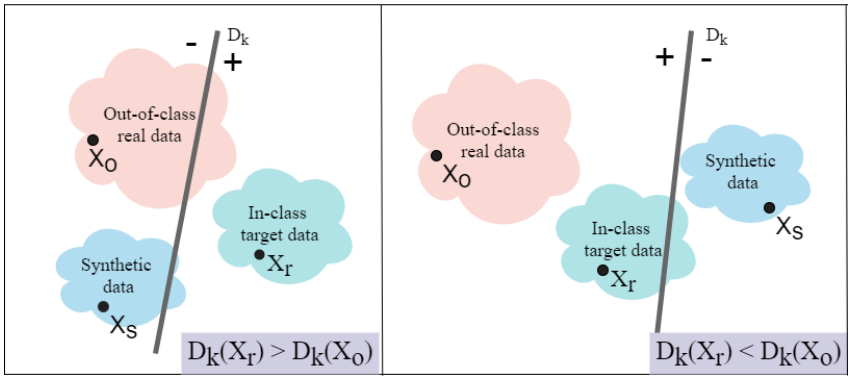

Figure 3. Simplified illustration of the effect of synthetic data position on the epoch discriminator $\boldsymbol{D_k}$. $\boldsymbol{X_o}$, $\boldsymbol{X_s}$ and $\boldsymbol{X_r}$ are arbitrary points belonging to real out-of-class, synthetic and real in-class regions, respectively. Left: synthetic data lies on the same side as out-of-class data and $\boldsymbol{D_k}$ assigns a higher score for $\boldsymbol{X_r}$ than for $\boldsymbol{X_o}$. Right: synthetic data lies opposite of real out-of-class data and $\boldsymbol{D_k}$ ends up assigning a higher score for $\boldsymbol{X_o}$ than for $\boldsymbol{X_r}$

In other words, assuming that D_k separates target data from synthetic data perfectly, we could threshold D_k and compose them such that a sample will be classified as out-of-class if it has at least one D_k classifying it as such. Conversely, a sample will be classified as in-class if all epoch discriminators D_k will classify it as such. The disadvantage of this formulation is that it does not allow the calculation of the AUC performance and it only leads to a crisp classification. As such, we reformulate the aggregation of D_k into the following formula which allows a soft classification framework:

$$C(x) = \prod_{k=1}^{N_D} \sigma\big(D_k(x)\big)$$

which produced better results in our experiments but also introduced an undesired phenomenon: D_k now assigns a score in the (0,1) range and some versions of D_k would assign higher scores to real out-of-class data than to in-class data. This corresponds to situations where the former lies further from the border than the latter (see Figure 3) and negatively impacts the system's performance by creating a slow and steady degradation trend as more epoch discriminators are added. This paper does not propose a solution for the issue, which is currently under investigation.

5 Experiments

5.1 Evaluation method

We tested our strategy on public multi-class datasets. We select each class in turn as the target class and consider all other classes to be novelty. The corresponding class training set is used for training while for testing we create a balanced dataset from the target class test set plus randomly chosen samples from other classes. Our performance measure is AUC - the Area Under the ROC Curve for separation between the target and novelty class. We also monitor how this measure evolves during training as more components are added to C, as well as the individual AUCs of these components. We take advantage of the work and extensive comparison performed by the authors of the OCGAN model [7] and compare our algorithm's performance with the figures presented by [7]. In our experiments, the number of iterations N_D has been set to 250.

5.2 Results on the MNIST dataset

This dataset [14] contains 10 classes of handwritten digits in 60K sample grayscale images of 28x28 pixels. We obtained a mean AUC of 0.9789 which is an insignificant 0.4% higher than OCGAN [7], and the best AUC on 5 out of 10 classes. Details on AUC performance comparisons of our method GANOCC with OCGAN [7] and other methods outperformed by the latter, are presented in Table 1.

Table 1. AUCs on the MNIST dataset by our method versus OCGAN and other methods which are outperformed by OCGAN. Compared to the latter, we get better results on 5/10 classes and achieve a minor performance improvement overall, mostly due to scores on digits 2 and 8.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**MEAN**
OCSVM [15]	0.988	0.999	0.902	0.950	0.955	0.968	0.978	0.965	0.853	0.955	0.9513
KDE [1]	0.885	0.996	0.710	0.693	0.844	0.776	0.861	0.884	0.669	0.825	0.8143
DAE [16]	0.894	0.999	0.792	0.851	0.888	0.819	0.944	0.922	0.740	0.917	0.8766
VAE [17]	0.997	0.999	0.936	0.959	0.973	0.964	0.993	0.976	0.923	0.976	0.9696
PixCNN [18]	0.531	0.995	0.476	0.517	0.739	0.542	0.592	0.789	0.340	0.662	0.6183
GAN [8]	0.926	0.995	0.805	0.818	0.823	0.803	0.890	0.898	0.817	0.887	0.8662
AND [19]	0.984	0.995	0.947	0.952	0.960	0.971	0.991	0.970	0.922	0.979	0.9671
AnoGAN [8]	0.966	0.992	0.850	0.887	0.894	0.883	0.947	0.935	0.849	0.924	0.9127
DSVDD [20]	0.980	0.997	0.917	0.919	0.949	0.885	0.983	0.946	0.939	0.965	0.9480
OCGAN [7]	0.998	0.999	0.942	0.963	0.975	0.980	0.991	0.981	0.939	0.981	0.9750
GANOCC	0.996	0.997	**0.969**	**0.97**	**0.976**	0.975	0.986	0.977	**0.961**	**0.982**	**0.9789**

5.3 Results on the CIFAR-10 dataset

This is also a 10-classes images dataset [21] containing 50K samples of 32x32 resolution and 3 color channels. Images are real-world photos of objects in that class. It is a more difficult set than MNIST not only due to its higher dimensionality but also because of the various backgrounds surrounding the subjects. This is reflected in the

AUCs we obtained, which are significantly lower than for the MNIST dataset. Nevertheless, our algorithm achieves a mean AUC of 0.742, which is a significant 13% increase when compared to OCGAN.

Table 2. AUCs on the CIFAR10 dataset, our method versus OCGAN and other methods which are outperformed by OCGAN. Compared to the latter, we get better results on 6/10 classes and a significant 13% higher mean AUC

	PLANE	CAR	BIRD	CAT	DEER	DOG	FROG	HORSE	SHIP	TRUCK	MEAN
OCSVM [15]	0.630	0.440	0.649	0.487	0.735	0.500	0.725	0.533	0.649	0.508	0.5856
KDE [1]	0.658	0.520	0.657	0.497	0.727	0.496	0.758	0.564	0.680	0.540	0.6097
DAE [16]	0.411	0.478	0.616	0.562	0.728	0.513	0.688	0.497	0.487	0.378	0.5358
VAE [17]	0.700	0.386	0.679	0.535	0.748	0.523	0.687	0.493	0.696	0.386	0.5833
PixCNN [18]	0.788	0.428	0.617	0.574	0.511	0.571	0.422	0.454	0.715	0.426	0.5506
GAN [8]	0.708	0.458	0.664	0.510	0.722	0.505	0.707	0.471	0.713	0.458	0.5916
AND [19]	0.717	0.494	0.662	0.527	0.736	0.504	0.726	0.560	0.680	0.566	0.6172
AnoGAN [8]	0.671	0.547	0.529	0.545	0.651	0.603	0.585	0.625	0.758	0.665	0.6179
DSVDD [20]	0.617	0.659	0.508	0.591	0.609	0.657	0.677	0.673	0.759	0.731	0.6481
OCGAN [7]	0.757	0.531	0.640	0.620	0.723	0.620	0.723	0.575	0.820	0.554	0.6566
GANOCC	0.778	0.649	0.655	**0.658**	**0.769**	**0.677**	**0.810**	**0.697**	**0.840**	0.709	**0.7420**

6 Discussion

In this paper we introduce a Generative Adversarial Network - based algorithm for one-class classification called GANOCC, that turns the problem into a binary classification problem making use of a generative model to create counter-samples. We use a modified adversarial training algorithm where the generator provides the counter-samples while the discriminators are progressively collected into the one-class classifier model. Our experiments show that the method surpasses the OCGAN model [7] which in turn is superior to other methods illustrated in Tables 1 and 2, and achieves significant improvements on real-world pictures. The same experiments also reveal some interesting aspects, which we discuss here.

Firstly, the authors of OCGAN [7] notice the relatively poor performance of all reconstruction-based methods on classes whose features significantly overlap those of other classes. Two relevant examples are digits 2 and 8 from the MNIST dataset – many other digits can be represented as a cut of digit 8, such as digits 3 and 9. Both their own and other results seem to be in line with this observation. The reason for this is, according to authors, that the latent space for complex digits will inherently learn to represent simpler digits as well. For our method though, this is exactly the setup where it tends to outperform all others. We assume this to be an effect of the same cause: during training, the generator will learn to output images that resemble distorted versions of the original digit and those simpler ones will be among them. Therefore, the discriminator will focus on separating them from the real data and produce higher performance in the end. This is empirically supported by results on MNIST, where our model outperforms OCGAN with the largest margin precisely on digits 2 and 8 (see Table 1).

Secondly, in our experiments the generator acts as a resource that gets exhausted as training progresses. In every experiment we noticed how the system improves sharply during the first steps, followed by a steady performance decay afterwards. We assume this is because, in our implementation, the generator eventually reaches an area where it creates samples that share more latent features with the in-class data than with the out-of-class data and this leads to the creation of discriminators on the "wrong side" (see Figure 3 right). A better control of the generator's output seems to be the key here to drive performance higher.

7 Conclusions and future work

We presented a novel one-class classification method based on binary classification of the target class against synthetic samples generated via adversarial training. We used a modified GAN training framework to create synthetic samples and provide the building blocks for our classifier. The generator is responsible for the creation of synthetic out-of-class data and is driven by the discriminators in various regions of the feature space that attempt to not overlap the real data manifold. We do not reuse discriminators but create a randomly initialized discriminator for each training epoch. Discriminators are progressively collected and combined in a multiplicative model. Experiments show that our method surpasses a similarly purposed recent method OCGAN [7] on two widely used benchmark datasets MNIST and CIFAR10, which in turn compares favorably with other methods for one-class classification as illustrated above by Tables 1 and 2. The most important contributions of this paper are: (1) introducing a classification training algorithm that behaves differently and does not match output with the traditional models, and (2) our approach achieves an overall favorable score when compared to other density and reconstruction-based methods on real-world images (see Tables 1 and 2).

Future work concerns the application of our new method and the study of its potential on other complex datasets including non-image data. In particular we plan to explore the application of an adapted form of the GANOCC algorithm on a large, and rich and complex clinical dataset formed of routine primary care records collected in UK, called CPRD (Clinical Practice Research Datalink https://www.cprd.com/), for predicting risk of dementia. This future work is to build on our ongoing research of using Machine Learning and Statistical Learning for building a tool for predicting risk of dementia. This condition, which has higher health and societal care costs compared to cancer, stroke and chronic heart disease taken together, represents a challenging to predict class within the primary care records data. The mentioned risk prediction problem could benefit of exploring novel and promising one-class classification techniques, including the GANOCC algorithm proposed here.

Moreover, as ongoing research work, we look also into investigating why our algorithm behaves differently from traditional classifiers and what is its full potential for extension and for further applicability.

References

[1] C. Bishop, "Pattern recognition and machine learning," Springer, 2006.

[2] D. Tax, "One-class classification. Concept-learning in the absence of counter-examples," Delft University of Technology, 2001.

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

[4] O. Mazhelis, "One-Class Classifiers: A Review and Analysis of Suitability in the Context of Mobile-Masquerader Detection" *Revue Africaine de la Recherche en Informatique et Mathématiques,* pp. 29-48, 2007.

[5] D. M. J. Tax and R. P. W. Duin, "Support Vector Data Description," *Machine Language,* pp. 45-66, 2004.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative Adversarial Networks," ARXIV 1406.2661, 2014.

[7] P. Perera, R. Nallapati and B. Xiang, "One-Class Novelty Detection Using GANs With Constrained Latent Representations," *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* pp. 2898-2906, 2019.

[8] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth and G. Langs, "Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery," *Niethammer M. et al. (eds) Information Processing in Medical Imaging. IPMI 2017. Lecture Notes in Computer Science, vol 10265.* , 2017.

[9] T. Salimans, I. Goodfellow, W. Zaremba, A. Radford and X. Chen, "Improved Techniques for Training GANs," ARXIV 1606.03498, 2016.

[10] A. Odena, "Semi-Supervised Learning with Generative Adversarial Networks," *arxix:1606.01583,* 2016.

[11] M. Arjovsky and L. Bottou, "Towards Principled Methods for Training Generative Adversarial Networks," in *ICLR*, 2017.

[12] M. Arjovsky, S. Chintala and L. Bottou, "Wasserstein Generative Adversarial Networks," in *Proceedings of the 34th International Conference on Machine Learning,* 2017.

[13] A. Radford, L. Metz and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," in *ICLR*, 2017.

[14] Y. LeCun, C. Cortes and C. Burges, "The MNIST Database of Handwritten Digits," http://yann.lecun.com/exdb/mnist/.

[15] B. Scholkopf, R. Williamson, A. Smola, J. Shawe-Taylor and J. Platt, "Support Vector Method for Novelty Detection," *NIPS,* vol. 12, pp. 582-588, 1999.

[16] R. Hadsell, S. Chopra and Y. Lecun, "Dimensionality reduction by learning an invariant mapping," in *10.1109/CVPR.2006.100*, 2006.

[17] D. Kingma and M. Welling, "Auto-encoding variational bayes," in *ICLR*.

[18] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves and K. Kavukcuoglu, "Conditional Image Generation with PixelCNN Decoders," in *NeurIPS*, 2016.

[19] D. Abati, A. Porello, S. Calderara and R. Cucchiara, "Autoregressive Novelty Detectors," *ArXIV*, no. 1807.01653, 2019.

[20] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, A. Muller and M. Kloft, "Deep one-class classification," in *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[21] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," University of Toronto.

[22] A. Odena, J. Buckman, C. Olsson, T. T. Brown, C. Olah, C. Raffel and I. Goodfellow, "Is Generator Conditioning Causally Related to GAN Performance?," arXiv:1802.08768, 2018.