CONCEPTRON: A PROBABILISTIC DEEP ONE-CLASS CLASSIFICATION METHOD

Anonymous authors
Paper under double-blind review

ABSTRACT

One-class learning through deep architectures is a particularly challenging task; in this scenario the crasis of kernel methods and deep networks can represent a viable strategy to empower already effective methods. In this contribution we present Conceptron, a probabilistic and deep one-class classification method. The proposed algorithm is a hybridization of the Nyström version of the Import Vector Domain Description (IVDD) to deep learning layers rendering the approach highly scalable (via batch stochastic gradient optimization) and automatically learning the underlying feature space. Further we modify the cost function to allow to get a Laplace distribution of the samples probabilities. Experiments on MNIST, CIFAR-10 and other benchmark datasets show that Conceptron (and/or variations) performs comparably or better with competing state-of-the-art methods with the additional capability of providing probabilities (through a logistic model) and avoiding any degeneracy in the training process.

1 INTRODUCTION

A widely important and studied problem in computational intelligence is one-class learning. The objective of one-class learning is detecting data samples which are unusual with respect to the expected behaviour (Chandola et al., 2009) or, in more abstract terms, to understand a *concept* and to separate it from the rest of the Universum (Weston et al., 2006). In one-class learning the training process is performed only on ”normal” samples (the ones respecting the expected behaviour) and no a priori knowledge is given about the unusual samples (Moya et al., 1993). The main challenge for this type of methods, and more in general for unsupervised learning techniques, is to learn the intrinsic structure of the data, in particular the lack of samples belonging to the anomalous class, makes the training harder as only the data belonging to one class delivers information. Another crucial aspect in this type of methods is hyper-parameters selection, namely the choice of the best performing model. While in supervised classification tasks cross-validation strategies can be used for model selection, here other types of heuristics have to be adopted.

Typical domains where one-class classification methods are used include: network security (Tuor et al., 2017), fraud detection (Adewumi & Akinyelu, 2017), medical diagnosis (Ronneberger et al., 2015) and in general the monitoring of correct functioning of facilities/plants/devices (Stojanovic et al., 2016). Several different algorithms have been proposed and some of the most effective ones are in the kernel and deep learning realm. One-Class Support Vector Machine (Schölkopf et al., 2001) (OC-SVM), the Support Vector Domain Description (Tax & Duin, 2004) (SVDD) in its original and deep version (Ruff et al., 2018) and the Import Vector Domain Description (Decherchi & Rocchia, 2016) (IVDD) are examples of effective one-class learning methods. IVDD uses an hypersphere as SVDD, but additionally it provides the probability estimation for each sample to be normal (based on the distance with respect to the hypersphere).

Two main problems arise however when using vanilla kernel-based approaches: they don’t properly scale (in memory and computing time) to big problems and being shallow they don’t learn features representations, offloading this task to the design of the input space. To cope with the scaling issue the Nyström method can represent a valid solution also endowing regularization properties (Erfani et al., 2016; Rudi et al., 2015). In the one-class case this strategy was applied to IVDD (Decherchi & Cavalli, 2020) where a fast and memory-efficient version was presented. Despite this improvement, the effectiveness to cope with high-dimensional datasets is limited as it may require a substantial feature engineering effort.

Deep learning approaches, on the other side, directly provide computational models that learn representations of data with multiple levels of abstraction (LeCun et al., 2015). Their effectiveness in disparate supervised and unsupervised learning contexts have been widely confirmed (Shrestha & Mahmood, 2019) reaching state of the art accuracy. In anomaly detection domain (AD) deep learning methods are often used to learn a low-dimensional feature representation which is then used in a disjoint and independent anomaly scoring step (Pang et al., 2021; Chalapathy & Chawla, 2019). This approach proved to be effective in extracting semantic features and, as expected, superior to linear dimensionality reduction methods (e.g. Principal Component Analisys (PCA) (Candès et al., 2011; Schölkopf et al., 1997; Zou et al., 2006) or random projections (Li et al., 2006; Pang et al., 2018; Pevnỳ, 2016) among others (Bengio et al., 2013)). Other methods couple feature learning with anomaly scoring. Typical architectures used for this purpose are Autoencoders (AEs) (Hinton & Salakhutdinov, 2006), which aim to learn a low-dimensional feature representation space from which the input samples can be properly reconstructed. Clearly, Variational AEs (VAEs) (Kingma & Welling, 2013) represent a further possibility. Another emergent method belonging to this category is AnoGAN (Schlegl et al., 2017), which adopts Generative Adversarial Networks (GANs) to learn a latent space capable to properly describe the normal samples. The idea behind all these approaches is that, after the training step, they have learned how to reconstruct normal input samples or how to represent normal samples in the latent feature space, therefore they will fail in reconstructing anomalous samples or in generating them. In this case, the anomaly score is computed on the reconstruction error. In Ruff et al. (2018) a new method for the one-class classification, dubbed Deep SVDD, has recently been proposed which obtained state of the art performances. The Deep SVDD algorithm learns a useful feature representation of the data together with the one-class classification objective. However, even if the method is very effective compared to the other competing ones, it have some weaknesses. The training process, as Authors discuss, can degenerate to trivial uninformative solutions if carried without a proper management. This is a structural problem and hence a change on the functional form would be desirable. Additionally, it does not provide any anomaly probability, but only a score can be retrieved.

In this paper we introduce *Conceptron* (and some variations on the theme), a probabilistic and deep one-class method. It is a hybridization of the IVDD method (Decherchi & Rocchia, 2016) to deep learning layers through the Nyström method (Decherchi & Cavalli, 2020). In this way the advantages of deep learning are combined with the advantages of the kernel-based approaches (radial basis functions locality control) obtaining a scalable and non degenerate one-class method. As the IVDD method, Conceptron provides the probability estimation for each sample to be normal. Additionally it utilizes a regularization term employing a Laplace prior that allows to obtain smooth probability distributions; to avoid overfitting we detail an early stopping strategy.

In the following, in Section 2 we describe the IVDD method, Conceptron and some variations. In Section 3, we challenge the devised methods with well-known data sets comparing the results with the directly competing method, namely deep SVDD. Finally, in Sections 4 we present conclusions and future works.

2 Methods

The following notation will be used throughout the text.

- $\mathbf{X}$ is the $n \times d$ matrix of samples, where n is the number of samples and d is the dimension of the input space.
- $\phi(\cdot) : \mathbb{R}^d \to \mathbb{R}^{d'}$ is the, possibly nonlinear, transformation mapping the d-dimensional input space into a d'-dimensional transformed one.
- $\mathbf{K}$ is the kernel matrix.
- r is the radius of the sphere, and $\Gamma = r^2$.
- $\mathbf{a}$ is the center of the sphere.
- f_i is the decision function, i.e., the function that tells if a sample is inside or outside the sphere. If $f_i > 0$, the ith pattern is outside the sphere, otherwise is inside.
- p_i is the probability of the i-th sample to be inside the sphere.
- C is a inverse regularization coefficient, and $\hat{C} = C/n$, where $C \geq 0$.

- $\hat{\mathbf{x}}_i = g(h(\mathbf{x}_i; \mathcal{W}_h); \mathcal{W}_g)$ is the reconstructed sample in a VAE (or AE) and h and g are respectively the coder and decoder functions parameterized by their respective weights.

2.1 Import Vector Domain Description

Import Vector Domain Description (IVDD) is a one-class classification kernel method (Decherchi & Rocchia, 2016) inspired by OC-SVM (Schölkopf et al., 2001) and SVDD methods (Tax & Duin, 1999). IVDD not only performs one-class classification using a hypersphere but also additionally provides a probability estimation. IVDD is solved by the following minimization problem:

$$\min_{\Gamma, a} \Gamma^2 - \hat{C} \sum_{i=1}^{n} \log(p_i) \tag{1}$$

where Γ is the square of the radius of the hypersphere, the constant $\hat{C}$ represents the trade-off between the radius size and the error minimization and p_i is the probability defined by a logistic model:

$$p_i = \frac{1}{1 + \exp(\beta f_i)} \tag{2}$$

where f_i is the decision function defined as:

$$f_i = ||\phi(\mathbf{x}_i) - \mathbf{a}||^2 - \Gamma \tag{3}$$

and β is a fixed coefficient. If the center $\mathbf{a}$ in the cost (Eq. 1) is expressed as a linear combination of the input patterns:

$$\mathbf{a} = \sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}_i) \tag{4}$$

than it is easy to show that the method can be set in the framework of kernel methods as only dot products involving $\phi(\mathbf{x}_i)$ are needed (Decherchi & Rocchia, 2016). In a reproducing kernel Hilbert space the dot product $\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$ can be implicitly computed by a single function evaluation $k(\mathbf{x}_i, \mathbf{x}_j)$ where k is a positive definite kernel function; hence the knowledge of $\phi(\cdot)$ is not required. Thanks to this property one can expand the distance between the samples and the center as:

$$d(\mathbf{x}_i, \mathbf{a}) = \left|\left| \phi(\mathbf{x}_i) - \sum_{k=1}^{n} \alpha_k \phi(\mathbf{x}_k) \right|\right|^2 = \boldsymbol{\alpha}^t \mathbf{K} \boldsymbol{\alpha} - 2\mathbf{K}_{i,(\cdot)} \boldsymbol{\alpha} + k_{ii} \tag{5}$$

where $\mathbf{K}$ is the kernel matrix of size $n \times n$, $\mathbf{K}_{i,(\cdot)}$ is the i-th row of the kernel matrix $\mathbf{K}$ of size $n \times n$, and k_{ii} is the self-similarity of the i-th sample.

In Decherchi & Rocchia (2016) an efficient optimization algorithm is also described that can be ascribed to the class of sequential minimal optimization (SMO) methods (Zeng et al., 2008) combined with features typical of expectation maximization (EM) algorithms (Dempster et al., 1977) as the sub-minimization problem is solved via self consistent iterations. The method has been shown to be very effective, however it suffers from the usual limits of kernel methods namely the scalability problem as the expansion is carried over all the samples n. To improve the scaling, the Nyström approximation can be used (Decherchi & Cavalli, 2020). In particular, a restricted number of samples can be selected a-priori from the original samples as landmarks. Then, one can assume that the decision function lives in the space spanned by these landmarks, instead of the whole set of samples in the training set. As a consequence, the center $\mathbf{a}$ can be expanded in the landmarks subset as:

$$\mathbf{a} = \sum_{k=1}^{n_l} \alpha_k \phi(\mathbf{x}_k) \tag{6}$$

where n_l is the number of landmarks.

The cost to be minimized in this case is therefore the following:

$$\Gamma^2 - \hat{C} \sum_{i=1}^{n} \log \left(1 + \exp \left(\beta \left(\left|\left| \phi(\mathbf{x}_i) - \sum_{k=1}^{n_l} \alpha_k \phi(\mathbf{x}_k) \right|\right|^2 - \Gamma \right) \right) \right) \tag{7}$$

again kernel properties lead to:

$$d(\mathbf{x}_i, \mathbf{a}) = \left|\left|\phi(\mathbf{x}_i) - \sum_{k=1}^{n_l} \alpha_k \phi(\mathbf{x}_k)\right|\right|^2 = \boldsymbol{\alpha}^t \hat{\mathbf{K}}_r \boldsymbol{\alpha} - 2\hat{\mathbf{K}}_{i,(\cdot)}\boldsymbol{\alpha} + k_{ii} \tag{8}$$

where $\hat{\mathbf{K}}_r$ is the kernel matrix of size $n_l \times n_l$, $\hat{\mathbf{K}}_{i,(\cdot)}$ is the i-th row of the kernel matrix $\hat{\mathbf{K}}$ of size $n \times n_l$, and k_{ii} is the self-similarity of the i-th sample. This version is much faster, bears a limited memory footprint, and also shows interesting regularization properties (Decherchi & Cavalli, 2020) confirming for the unsupervised scenario the findings in (Rudi et al., 2015) for the supervised setting.

2.2 Conceptron

In this contribution we introduce Conceptron (and variants), a cost-modified and deep version of the IVDD method. Conceptron is obtained starting from the Nyström version of the IVDD method, which, as already discussed, is rather scalable even for large scale datasets, but that still it is not a deep paradigm.

The simple, and general, observation is that a kernel method when gets approximated via a limited set of landmarks can be interpreted as the terminal layer of a neural network, where the kernels represent the activation function. For instance, it is not difficult to apply this approach to Regularized Least Squares (Rifkin et al., 2003) and realize that such machine becomes an *old-school* radial basis function network where the centers are provided by the landmarks samples.Through a further approximation one could even avoid the landmarks samples and use random values (Gastaldo et al., 2016). On the light of this observation we can define this objective function:

$$\min_{\Gamma, a, \mathcal{W}} \Gamma^2 - \hat{C} \sum_{i=1}^{n} \log(\hat{p}_i) + \lambda \mathcal{L}(\mathbf{x}_i, \hat{\mathbf{x}}_\mathbf{i}) \tag{9}$$

where the third term is the loss function of an AE or a VAE and λ is a regularization coefficient. Here we are jointly optimizing a VAE/AE and the IVDD Nyström loss function. This cost function for λ different from zero is inherently not degenerate as it prevents any collapse of the feature space. We will refer to this functional as Deep IVDD. Here $\hat{p}_i$ is a modification of the original probability definition; it is still a logistic model but the decision function is now scaled with respect to the distance of a sample from the center:

$$\hat{f}_i = \frac{||\phi(h(\mathbf{x}_i; \mathcal{W})) - \mathbf{a})||^2 - \Gamma}{||\phi(h(\mathbf{x}_i; \mathcal{W})) - \mathbf{a})||} = \frac{\boldsymbol{\alpha}^t \hat{\mathbf{K}}_r \boldsymbol{\alpha} - 2\hat{\mathbf{K}}_{i,(\cdot)}\boldsymbol{\alpha} + k_{ii} - \Gamma}{\sqrt{\boldsymbol{\alpha}^t \hat{\mathbf{K}}_r \boldsymbol{\alpha} - 2\hat{\mathbf{K}}_{i,(\cdot)}\boldsymbol{\alpha} + k_{ii}}}. \tag{10}$$

This modification helps in obtaining smoother probability distributions with respect to the IVDD method; the result is that the probability histogram covers better all the possible probability values in the range between zero and one. We will call Conceptron a functional similar to this one, where now instead of controlling the radius directly as in the SVDD approach, we control the adherence of the probability distribution p to a reference probability distribution q via the Kullback-Leibler divergence:

$$\min_{\Gamma, a, \mathcal{W}} D_{KL}(q||\hat{p}) - \hat{C} \sum_{i=1}^{n} \log(\hat{p}_i) + \lambda \mathcal{L}(\mathbf{x}_i, \hat{\mathbf{x}}_i). \tag{11}$$

This variation of the cost function allows to obtain a control of the probabilities distribution and hence indirectly controlling the radius size as a consequence (a schematic representation of Conceptron can be found in Appendix A). In this work we set the prior q as a Laplace distribution:

$$q = \frac{1}{2b} \exp\left(-\frac{|x - \mu|}{b}\right). \tag{12}$$

We suggest this distribution to be reasonable for several reasons, beyond empirical evidence. First we observe that learning a concept is meaningful only if this learning process is robust. In other words we expect that the majority of normal samples have high probability values, hence the norm-one induced sharp peak. Second, we prefer this distribution to a Gaussian, for instance, as we expect that the confidence reduction with respect to the distance from the center is a monotone function and there is no reason to have a convexity change in this probability shift process. These cost functions (and the following variants we will discuss) are always optimized via the Batch Stochastic Gradient Descent (SGD) and its variants (e.g. Adam (Kingma & Ba, 2014)), whereas for IVDD we used the original batch algorithm. In particular, when we train by SGD we take advantage of the fact that, with a proper initialization, and a sufficiently big C ($C = 1$ proves always to be sufficient) the size of the hypersphere is monotonically increasing with respect to the epochs. We fix a priori a range $[\pi_{low}, \pi_{high}]$ where we accept the percentage of inner samples and we train the network until for the first time this range is hit. If learning is too fast and the range is skipped as the hypersphere is growing too fast, we step back to the previous epoch parameters and decrease the learning rate. This procedure is similar to what it was done for IVDD where one reaches the prescribed range by bisection on C (Decherchi & Cavalli, 2020).

The Conceptron training procedure hence overcomes the IVDD sensitivity problem to the C value, as its value is never changed, and we take advantage of the epochs to reach the desired range; this strategy proved to be very fast and always hitting the desired range. Here we detail the pseudo-code where η indicates the learning rate, $takestep$ is a gradient update step and π is the fraction of inner samples.

Algorithm 1 Conceptron algorithm

```
Require: $\mathbf{X}, \sigma, \eta$
Ensure: $\boldsymbol{\alpha}, \mathbf{W}, \Gamma$
  $\mathbf{W} = \mathbf{W}_0, \eta = \eta_0, \boldsymbol{\alpha} = 1/n_l, \pi = 0.0$
  while $\pi \notin [\pi_{low}, \pi_{high}]$ do
      for epoch do                                            ▷ $\pi$ is the ratio of inner samples
          $\mathbf{V}, \pi, \hat{\boldsymbol{\alpha}}, \hat{\Gamma} = takestep(\mathbf{X}, \mathbf{W}, \eta, \Gamma, \boldsymbol{\alpha})$
          if $\pi > \pi_{high}$ then                                  ▷ step back
              $\mathbf{V} = \mathbf{W}$
              $\eta = \eta/10.0$
              $\hat{\boldsymbol{\alpha}} = \boldsymbol{\alpha}$
              $\hat{\Gamma} = \Gamma$
          end if
          $\mathbf{W} = \mathbf{V}$
          $\boldsymbol{\alpha} = \hat{\boldsymbol{\alpha}}$
          $\Gamma = \hat{\Gamma}$
      end for
  end while
```

The range $[\pi_{low}, \pi_{high}]$ is an input hyper-parameter and may vary according to the reliability of the training dataset, playing the same role of ν in OC-SVM. In fact, the lowest π_{high}, the smaller is the radius and the more conservative is the final solution. In general, a conservative approach should be used when the training dataset contains many heterogeneous and imperfect samples.

For the original IVDD we apply an analogous back-tracking strategy but this time not on the epochs but on C. This method is needed for IVDD as it is highly sensitive to the C value.

3 Results

The implementation (except the original IVDD Matlab code) was written in Python and all the experiments can be reproduced using the code available at the following link in the references: Conceptron-URL. To fully understand the sources of inaccuracies/improvements and the good or bad behaviour of the probability distributions we tested several combinations and variations over the original IVDD and the just introduced methods. In detail we will indicate by IVDD the original shallow IVDD, by IVDD-Nys the Nyström version, by Fixed-VAE when the VAE is pre-trained only and no joint optimization is performed. Further the suffix -S and -K will indicate respectively when the scaled decision function and the Kullback-Leibler divergence is used. Together with these variations we finally also tested what we termed Deep IVDD and Conceptron methods.

We tested the methods on the well-known MINIST (LeCun et al.) and CIFAR-10 (Krizhevsky et al., 2009) datasets. Both MNIST and CIFAR-10 have ten different classes. MNIST is composed by 60000 images for training (≈ 6000 for each class) and 10000 images for testing (≈ 1000 for each

Table 1: Mean results (AUC and BER) over 10 classes on MNIST and CIFAR-10 datasets.

	MNIST AUC	**CIFAR AUC**	**MNIST BER**	**CIFAR BER**
Deep SVDD	95.01 ±0.8	**60.9** ±2.0	12.0 ±1.1	45 ±1.3
IVDD	76.1 ±1.9	57.4 ±0.0	33.2 ±2.5	**44.8** ±0.0
IVDD Nys	89.5 ±0.2	57.6 ±0.2	19.5 ±0.5	45.2 ±0.3
Fixed-VAE IVDD	**99.9** ±0.0	41.7 ±1.0	**7.9** ±1.2	50.8 ±0.4
Fixed-VAE IVDD-Nys	96.1 ±0.1	55.1 ±0.2	11.4 ±0.4	45.6 ±0.7
Fixed-VAE IVDD-Nys-S	96.1 ±0.1	55.1 ±0.0	11.1 ±0.3	45.5 ±0.0
Fixed-VAE IVDD-Nys-SK	96.1 ±0.0	55.1 ±0.2	10.9 ±0.5	45.9 ±0.8
Deep IVDD	96.7 ±0.1	56.2 ±0.3	10.8 ±0.4	46.0 ± 0.6
Conceptron	96.2 ±0.6	54.2 ±0.6	11.0 ±1.1	47.4 ±0.8

class) each of shape 28x28. On the other hand, CIFAR-10 is composed by 50000 images for training (5000 for each class) and 10000 for testing (1000 for each class) each with size 32x32. All images were pre-processed by rescaling to the domain [0, 1] via min-max scaling.

Here, we consider just one class at a time for training and all the test set for testing. In this way, ten different experiments can be run for each dataset. In addition, to evaluate the generalization ability of the model, we repeated ten times the experiments with different seeds and results have been averaged accordingly. We evaluate the results quantitatively via the Balanced Error Rate (BER) metric (the average of the per class error) and the Area Under the Curve (AUC) by using the ground truth labels in testing.

In both experiments on MNIST and CIFAR-10 we use a VAE as network architecture. Further details on the network structure can be found in Appendix B. Both the networks for the two datasets are pre-trained for 100 epochs with a batch size of 256 and the Adam optimizer (all the parameters are as recommended in the original work (Kingma & Ba, 2014)). After the pre-training step, both the Deep IVDD and the Conceptron cost functions are optimized using a reference range $[\pi_{low}, \pi_{high}]$ and the previously presented algorithm. In all the run experiments the additional guarding term $-\min(0, \Gamma)$ in the cost function has been added. We found however that this term is always zero during the minimization process. The following parameters have been adopted: batch size $= 32$, $n_l = 50$, kernel RBF, $\sigma = \max_{ij}(d_{ij})/\log(n_l)$ (where d_{ij} is the distance between i and j landmarks samples), and $\beta = 25$. The value of $\hat{C}$ is set at the value 1 and never changed. On MNIST experiments, $\lambda = 100$ for both Deep IVDD and the Conceptron methods, while on CIFAR-10 $\lambda = 0.1$. The λ values were chosen such that the IVDD and VAE cost components have analogous orders of magnitude and the IVDD component is always slightly more dominant. The Laplace parameters are set as: $\mu = 1$ and $b = 0.2$. Finally, in all the experiments, we used the range $[80\%, 90\%]$.

For the sake of comparison, the Deep SVDD method (Ruff et al., 2018) as well as the IVDD method (Decherchi & Rocchia, 2016) are used, which represent the competing state-of-the-art methods. We run Deep SVDD with the same configuration proposed in: Deep-SVDD-URL. IVDD has been run with $\beta = 25$, $\sigma = \max_{ij}(d_{ij})/\log(n_l)$, and $\hat{C}$ automatically updated and tuned for including inside the sphere the $[80\% - 90\%]$ of samples. All the other parameters remains unchanged and are described in Decherchi & Rocchia (2016).

In addition to these experiments, we investigated other variations, which incrementally modify the original IVDD method until Deep IVDD and Conceptron methods are obtained. The detailed results of all these experiments are in Table 2 and in Table 3 in Appendix C whereas in Table 1 we present a summary of the results.

Appendix D shows a t-SNE projection of the feature spaces for the training and test data.

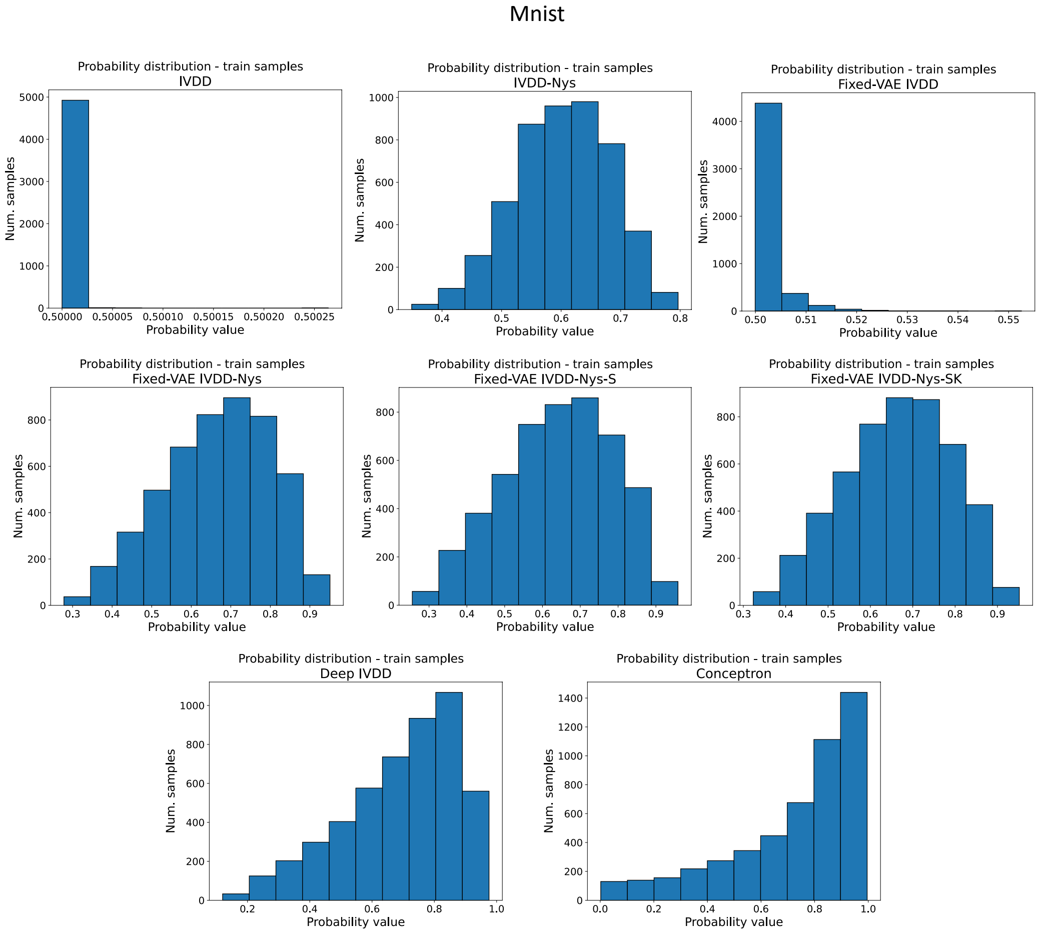

Figure 1: Probability distributions of the training samples (class 0, seed 0) for the MNIST dataset.

As expected, the performances of IVDD are not on par when dealing with high-dimensional datasets. However, the Fixed-VAE IVDD method interestingly outperforms in most cases the other tested methods and proves to be significantly superior on average to all the methods. This result confirms that using deep approaches to learn a derived feature representation in a completely disjoint step is a valid strategy to address one-class classification problems. Moreover using a kernel method as terminus of the network with a RBF kernel proves to be very effective overall. The MNIST result particularly is almost perfect. However, this approach is not scalable with large-scale datasets as, again, the training and evaluation step are both expensive. Nevertheless, in terms of AUC and BER only, this is the best approach. Conceptron obtains results close to Deep SVDD and additionally bears informative probability distributions.

Analysing more in depth the various solutions, particularly the probability distributions of the training samples (see Figure 1), one can observe that they progressively get better and that the best performances are obtained with Conceptron. These results confirm that using a more robust decision function (Eq. 10), the benefits gained with the feature space induced by the VAEs and particularly the new Laplace regularizer, allow to obtain good results in terms of BER and AUC and provide, at the same time, well distributed probabilities in the entire range between 0 and 1. The testing probability distributions on MNIST are reported, for completeness, in Appendix E. Training and testing probability distributions on CIFAR-10 are in Appendix F.

On CIFAR-10 the situation is less clear as all the methods show pretty poor performances. The BER and the AUC scores are in general modest meaning that, when complex colors images are used none of the methods is capable to obtain satisfying performances. Results in the Appendix F show that in CIFAR-10 the Conceptron distribution is not optimal; this is a quite hard task and this may deteriorate the probability modeling attempt.

A qualitative check on the most normal and most anomalous samples obtained with Deep SVDD, Deep IVDD and Conceptron on MNIST and CIFAR-10 points to some interesting features. One can observe in Figure 2 that the most normal samples found by Deep SVDD, are, by visual inspection less prototypical than the ones found by the competing methods. In fact, the numbers look more well rounded, less sloped and more similar to each other. Here both the VAE and the finely tuned probability model are contributing to this result. For the training anomalous cases, visual inspection shows that the most anomalous samples detected by Conceptron and variations are similar to the ones detected by Deep SVDD. For the CIFAR-10 cases, performances are everywhere poor and can be found in Figure 3.

Eventually, Deep SVDD and Conceptron obtain similar AUC and BER performances but thanks to a probability model and the VAE, Conceptron extracts better the central *concepts*.

Additional experiments on some UCI datasets can be found in Appendix G.

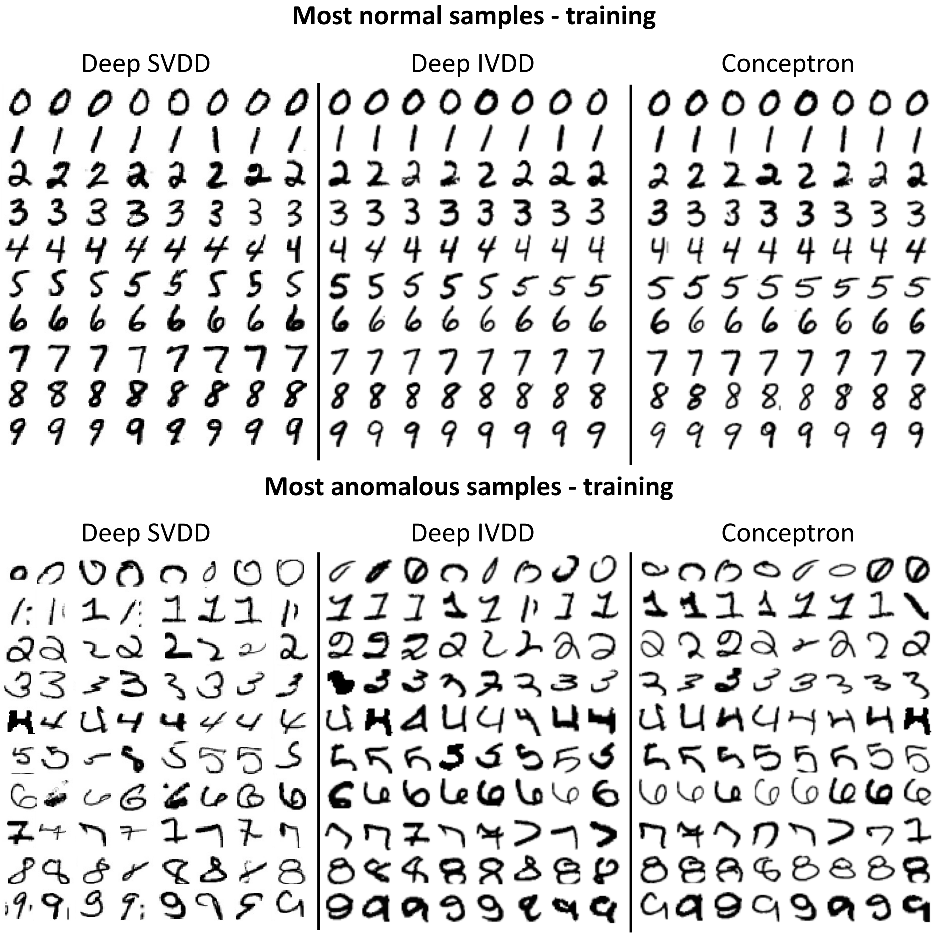
Most normal samples - training
Deep SVDD
Deep IVDD
Conceptron
Most anomalous samples - training
Deep SVDD
Deep IVDD
Conceptron

Figure 2: Analysis of the most normal and anomalous samples obtained with different one-class classification methods on the standard MNIST dataset.

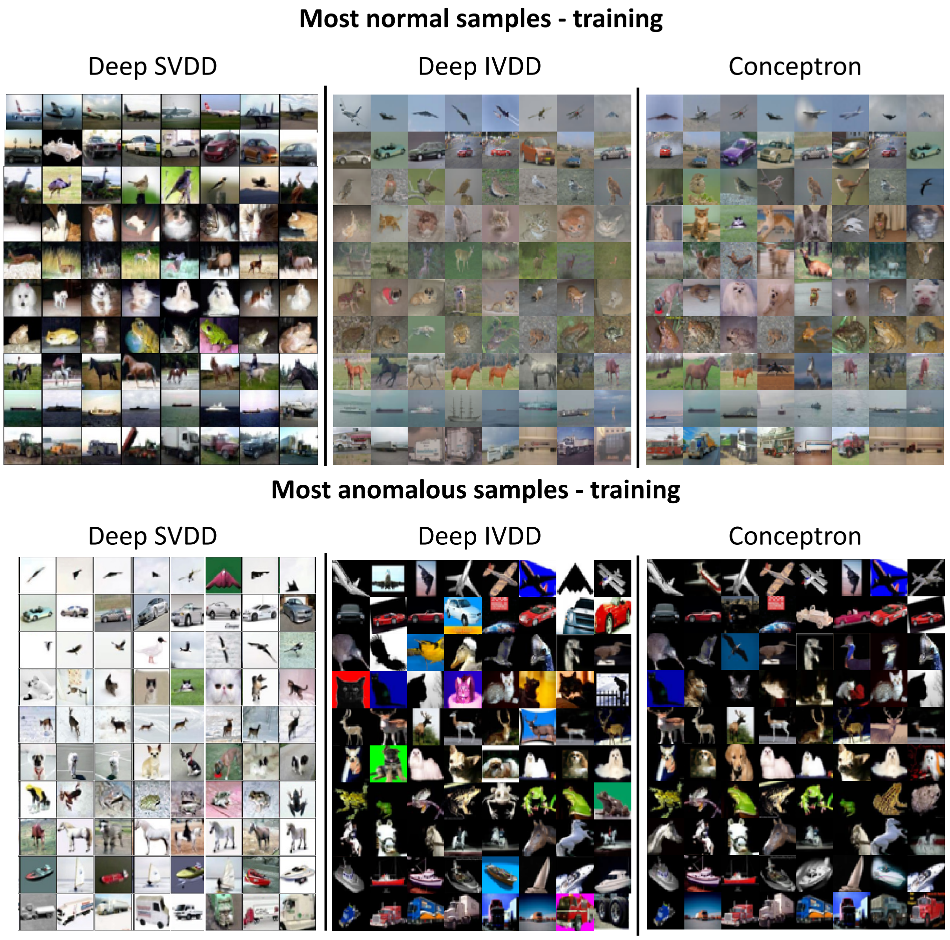
Most normal samples - training
Deep SVDD
Deep IVDD
Conceptron
Most anomalous samples - training
Deep SVDD
Deep IVDD
Conceptron

Figure 3: Analysis of the most normal and anomalous samples obtained with different one-class classification methods on the standards CIFAR-10 dataset.

4 Conclusion and future works

In this paper we introduced Conceptron and variations, a set of unsupervised deep one-class classification methods. The most significant advantage of the presented approach is that it hybridizes the capability of the Import Vector Domain Description to deliver a probability to deep learning layers. Using the SGD as optimization method and the Nyström approximation, the solution is scalable. Additionally, the VAE architecture combined with the benefits of the local control of the RBF kernel and the Nyström approximation, allows to obtain a stable solution avoiding any degeneracy in the training procedure.

Even though the AE architecture used in Eq. 9 and Eq. 11 is not mandatory, the Variational Autoencoders (VAEs) (Kingma & Welling, 2014; Rezende et al., 2014) approach provides an effective starting point as the latent space of the VAEs is induced in reproducing a Gaussian distribution, something that is more akin to be embedded into a sphere. Even though Conceptron obtained results analogous to the others competing methods, the performances with complex, colorful images can be still improved. Moreover, despite we defined defaults for many parameters, we would like to understand if a model selection strategy is possible, obviously a genuine unsupervised one. In future, we will examine in depth the benefits of Conceptron in the life sciences/clinical context where taking decisions is both crucial and expensive and one cannot avoid to estimate a probability before taking action. Efforts in these areas are already underway.

5 Reproducibility Statement

The complete source code, the results and some how-to can be found at the link Conceptron-URL in the references.

References

Aderemi O Adewumi and Andronicus A Akinyelu. A survey of machine-learning and nature-inspired based credit card fraud detection techniques. *International Journal of System Assurance Engineering and Management*, 8(2):937–953, 2017.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):1–37, 2011.

Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), July 2009. ISSN 0360-0300. doi: 10.1145/1541880.1541882. URL `https://doi.org/10.1145/1541880.1541882`.

Conceptron-URL. URL `https://anonymous.4open.science/r/Conceptron-FCB5/readme.md`. Accessed: 2021-10-04.

Sergio Decherchi and Andrea Cavalli. Fast and memory-efficient import vector domain description. *Neural Processing Letters*, 52:511–524, 2020.

Sergio Decherchi and Walter Rocchia. Import vector domain description: A kernel logistic one-class learning algorithm. *IEEE transactions on neural networks and learning systems*, 28(7): 1722–1729, 2016.

Deep-SVDD-URL. Lukas ruff. deep-svdd. URL `https://github.com/lukasruff/Deep-SVDD`.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1): 1–22, 1977.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Sarah M Erfani, Sutharshan Rajasegarar, Shanika Karunasekera, and Christopher Leckie. High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning. *Pattern Recognition*, 58:121–134, 2016.

Paolo Gastaldo, Federica Bisio, Sergio Decherchi, and Rodolfo Zunino. Sim-elm: Connecting the elm model with similarity-function learning. *Neural Networks*, 74:22–34, 2016.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL `http://arxiv.org/abs/1312.6114`.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. Mnist handwritten digit database. URL `http://yann.lecun.com/exdb/mnist/`. Accessed: 2021-10-04.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Ping Li, Trevor J Hastie, and Kenneth W Church. Very sparse random projections. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 287–296, 2006.

Mary M Moya, Mark W Koch, and Larry D Hostetler. One-class classifier networks for target recognition applications. *NASA STI/Recon Technical Report N*, 93:24043, 1993.

Guansong Pang, Longbing Cao, Ling Chen, and Huan Liu. Learning representations of ultrahigh-dimensional data for random distance-based outlier detection. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2041–2050, 2018.

Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. Deep learning for anomaly detection: A review. *ACM Computing Surveys (CSUR)*, 54(2):1–38, 2021.

Tomáš Pevnỳ. Loda: Lightweight on-line detector of anomalies. *Machine Learning*, 102(2):275–304, 2016.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In Eric P. Xing and Tony Jebara (eds.), *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 1278–1286, Bejing, China, 22–24 Jun 2014. PMLR.

Ryan Rifkin, Gene Yeo, Tomaso Poggio, et al. Regularized least-squares classification. *Nato Science Series Sub Series III Computer and Systems Sciences*, 190:131–154, 2003.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Alessandro Rudi, Raffaello Camoriano, and Lorenzo Rosasco. Less is more: Nyström computational regularization. In *NIPS*, pp. 1657–1665, 2015.

Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *International conference on machine learning*, pp. 4393–4402. PMLR, 2018.

Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International conference on information processing in medical imaging*, pp. 146–157. Springer, 2017.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *International conference on artificial neural networks*, pp. 583–588. Springer, 1997.

Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

Ajay Shrestha and Ausif Mahmood. Review of deep learning algorithms and architectures. *IEEE Access*, 7:53040–53065, 2019.

Ljiljana Stojanovic, Marko Dinic, Nenad Stojanovic, and Aleksandar Stojadinovic. Big-data-driven anomaly detection in industry (4.0): An approach and a case study. In *2016 IEEE international conference on big data (big data)*, pp. 1647–1652. IEEE, 2016.

David M.J Tax and Robert P.W Duin. Support vector domain description. *Pattern Recognition Letters*, 20(11):1191–1199, 1999. ISSN 0167-8655. doi: https://doi.org/10.1016/S0167-8655(99)00087-2.

David MJ Tax and Robert PW Duin. Support vector data description. *Machine learning*, 54(1): 45–66, 2004.

Aaron Tuor, Samuel Kaplan, Brian Hutchinson, Nicole Nichols, and Sean Robinson. Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Jason Weston, Ronan Collobert, Fabian Sinz, Léon Bottou, and Vladimir Vapnik. Inference with the universum. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pp. 1009–1016, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933832. doi: 10.1145/1143844.1143971. URL https://doi.org/10.1145/1143844.1143971.

Zhi-Qiang Zeng, Hong-Bin Yu, Hua-Rong Xu, Yan-Qi Xie, and Ji Gao. Fast training support vector machines using parallel sequential minimal optimization. In *2008 3rd international conference on intelligent system and knowledge engineering*, volume 1, pp. 997–1001. IEEE, 2008.

Hui Zou, Trevor Hastie, and Robert Tibshirani. Sparse principal component analysis. *Journal of computational and graphical statistics*, 15(2):265–286, 2006.

A CONCEPTRON: A SCHEMATIC REPRESENTATION

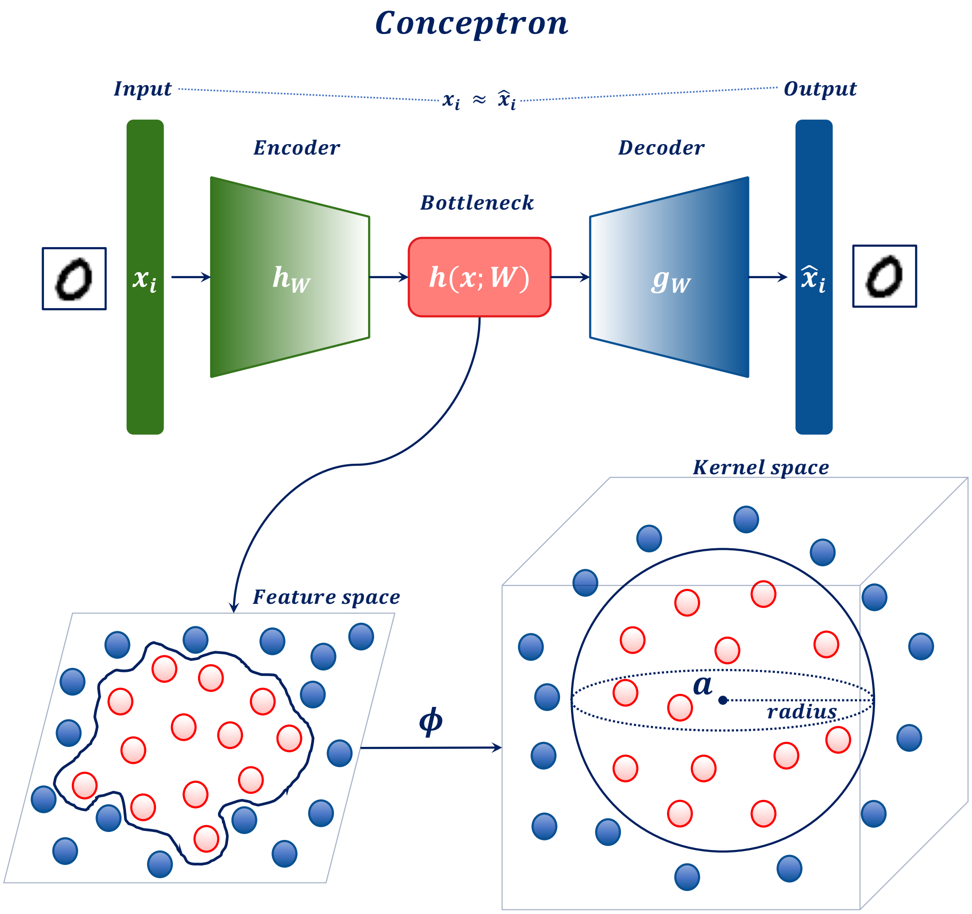

Figure 4: Conceptron and variations schematic representation.

B VAEs ARCHITECTURE ON MNIST AND CIFAR-10 EXPERIMENTS

In this appendix we report the VAEs architectures used for the experiements. The kernel size is 3, while the stride is 2. Each layer is followed by ReLU activations except for the last layer of the decoder, which is followed by the sigmoid activation function.

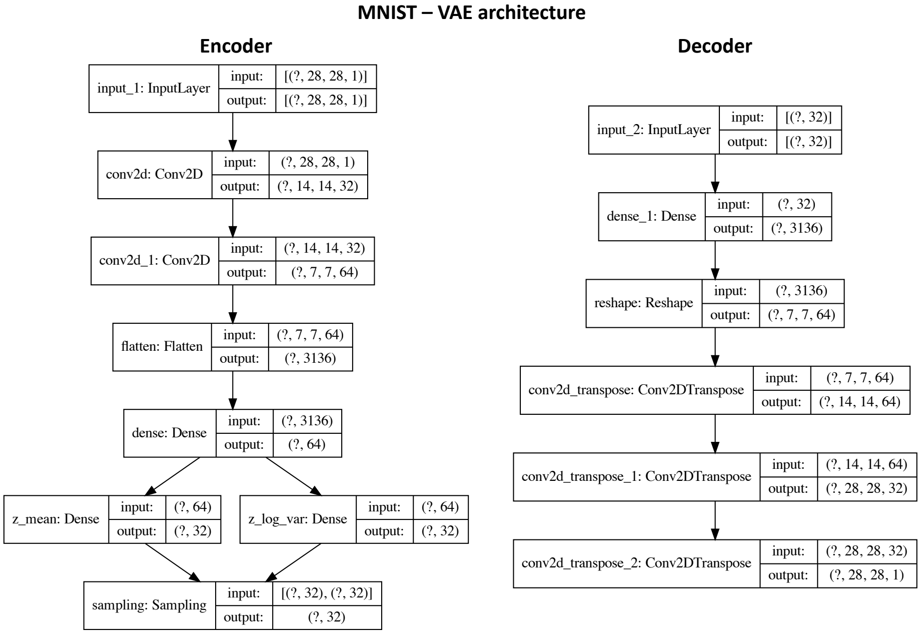
MNIST – VAE architecture
Encoder
input_1: InputLayer | input: [(?, 28, 28, 1)] | output: [(?, 28, 28, 1)]
conv2d: Conv2D | input: (?, 28, 28, 1) | output: (?, 14, 14, 32)
conv2d_1: Conv2D | input: (?, 14, 14, 32) | output: (?, 7, 7, 64)
flatten: Flatten | input: (?, 7, 7, 64) | output: (?, 3136)
dense: Dense | input: (?, 3136) | output: (?, 64)
z_mean: Dense | input: (?, 64) | output: (?, 32)
z_log_var: Dense | input: (?, 64) | output: (?, 32)
sampling: Sampling | input: [(?, 32), (?, 32)] | output: (?, 32)
Decoder
input_2: InputLayer | input: [(?, 32)] | output: [(?, 32)]
dense_1: Dense | input: (?, 32) | output: (?, 3136)
reshape: Reshape | input: (?, 3136) | output: (?, 7, 7, 64)
conv2d_transpose: Conv2DTranspose | input: (?, 7, 7, 64) | output: (?, 14, 14, 64)
conv2d_transpose_1: Conv2DTranspose | input: (?, 14, 14, 64) | output: (?, 28, 28, 32)
conv2d_transpose_2: Conv2DTranspose | input: (?, 28, 28, 32) | output: (?, 28, 28, 1)

Figure 5: VAE architecture for MNIST experiments.

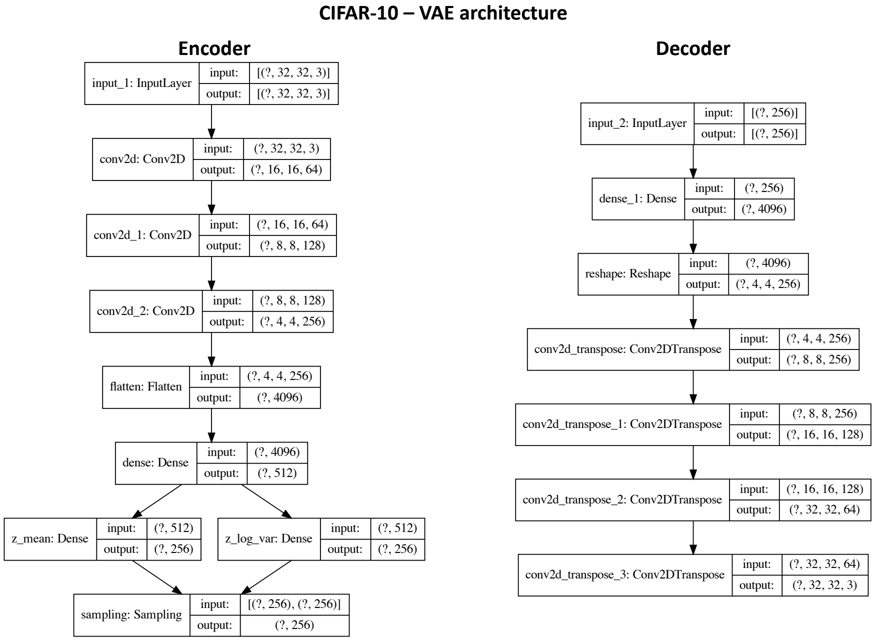
CIFAR-10 – VAE architecture
Encoder
input_1: InputLayer | input: [(?, 32, 32, 3)] | output: [(?, 32, 32, 3)]
conv2d: Conv2D | input: (?, 32, 32, 3) | output: (?, 16, 16, 64)
conv2d_1: Conv2D | input: (?, 16, 16, 64) | output: (?, 8, 8, 128)
conv2d_2: Conv2D | input: (?, 8, 8, 128) | output: (?, 4, 4, 256)
flatten: Flatten | input: (?, 4, 4, 256) | output: (?, 4096)
dense: Dense | input: (?, 4096) | output: (?, 512)
z_mean: Dense | input: (?, 512) | output: (?, 256)
z_log_var: Dense | input: (?, 512) | output: (?, 256)
sampling: Sampling | input: [(?, 256), (?, 256)] | output: (?, 256)
Decoder
input_2: InputLayer | input: [(?, 256)] | output: [(?, 256)]
dense_1: Dense | input: (?, 256) | output: (?, 4096)
reshape: Reshape | input: (?, 4096) | output: (?, 4, 4, 256)
conv2d_transpose: Conv2DTranspose | input: (?, 4, 4, 256) | output: (?, 8, 8, 256)
conv2d_transpose_1: Conv2DTranspose | input: (?, 8, 8, 256) | output: (?, 16, 16, 128)
conv2d_transpose_2: Conv2DTranspose | input: (?, 16, 16, 128) | output: (?, 32, 32, 64)
conv2d_transpose_3: Conv2DTranspose | input: (?, 32, 32, 64) | output: (?, 32, 32, 3)

Figure 6: VAE architecture for CIFAR-10 experiments.

C Per class details on MNIST and CIFAR-10 experiments

Table 2: Average AUC (over 10 seeds) on standard MNIST and CIFAR datasets.

MNIST	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Deep SVDD	98.2 ±0.4	99.5 ±0.1	91.3 ±1.6	92.5 ±1.2	95.1 ±0.7	89.2 ±0.8	98.3 ±0.5	94.6 ±0.9	94.4 ±1.0	96.9 ±0.4
IVDD	71.9 ±0.6	99.1 ±0.0	65.7 ±1.3	78.9 ±3.9	75.8 ±2.5	57.1 ±1.5	73.8 ±5.2	83.6 ±0.1	77.2 ±1.9	78.3 ±2.5
IVDD Nys	98.3 ±0.1	99.6 ±0.01	82.4 ±0.3	88.3 ±0.1	89.4 ±0.2	80.4 ±0.4	91.4 ±0.2	93.2 ±0.1	81.8 ±0.2	90.5 ±0.1
Fixed-VAE IVDD	**99.9** ±0.0	99.9 ±0.0	**99.8** ±0.0	**99.9** ±0.2	**99.9** ±0.0	**99.9** ±0.0	**99.9** ±0.0	**99.9** ±0.0	**99.9** ±0.0	**99.8** ±0.0
Fixed-VAE IVDD-Nys	98.7 ±0.0	**99.9** ±0.0	92.8 ±0.1	96.3 ±0.1	94.6 ±0.1	94.8 ±0.1	98.2 ±0.0	94.7 ±0.1	94.4 ±0.1	96.8 ±0.1
Fixed-VAE IVDD-Nys-S	98.7 ±0.0	**99.9** ±0.0	92.8 ±0.6	96.3 ±0.1	94.6 ±0.1	94.7 ±0.1	98.2 ±0.0	94.7 ±0.1	94.3 ±0.1	96.8 ±0.0
Fixed-VAE IVDD-Nys-SK	98.7 ±0.0	**99.9** ±0.0	92.8 ±0.1	96.3 ±0.0	94.6 ±0.1	94.7 ±0.0	98.2 ±0.1	94.7 ±0.0	94.3 ±0.1	96.8 ±0.0
Deep IVDD	99.2 ±0.0	99.9 ±0.0	94.4 ±0.1	96.7 ±0.1	95.3 ±0.1	95.4 ±0.1	98.4 ±0.1	95.5 ±0.1	95.0 ±0.2	97.1 ±0.1
Conceptron	99.3 ±0.0	99.8 ±0.0	92.5 ±3.44	96.0 ±0.1	94.9 ±0.2	93.9 ±1.5	98.5 ±0.1	95.6 ±0.1	94.8 ±0.4	97.2 ±0.1

CIFAR-10	**air**	**auto**	**bird**	**cat**	**deer**	**dog**	**frog**	**horse**	**sheep**	**truck**
Deep SVDD	62.3 ±2.2	**60.0** ±1.7	49.4 ±0.8	**58.4** ±1.6	53.5 ±2.2	**63.3** ±2.0	56.5 ±3.5	**61.9** ±3.2	**75.2** ±1.4	**68.6** ±1.2
IVDD	61.7 ±0.0	43.3 ±0.0	62.1 ±0.0	49.5 ±0.0	71.2 ±0.0	50.0 ±0.0	67.7 ±0.0	49.7 ±0.0	66.0 ±0.0	53.8 ±0.0
IVDD Nys	63.3 ±0.3	40.4 ±0.1	63.5 ±0.1	49.0 ±0.1	73.3 ±0.1	51.3 ±0.1	**69.0** ±0.4	51.1 ±0.1	65.5 ±0.3	49.2 ±0.1
Fixed-VAE IVDD	46.4 ±0.2	37.2 ±0.2	41.8 ±0.2	42.9 ±2.4	42.3 ±1.0	40.3 ±1.1	39.6 ±1.3	38.2 ±1.4	44.6 ±0.8	43.0 ±1.4
Fixed-VAE IVDD-Nys	60.2 ±0.2	37.9 ±0.2	**64.6** ±0.2	49.7 ±0.2	**73.7** ±0.1	49.4 ±0.2	65.9 ±0.1	48.4 ±0.2	64.7 ±0.2	37.0 ±0.2
Fixed-VAE IVDD-Nys-S	60.1 ±0.0	38.0 ±0.0	**64.6** ±0.0	49.7 ±0.0	**73.7** ±0.0	49.5 ±0.0	65.7 ±0.0	48.3 ±0.0	64.4 ±0.0	37.1 ±0.0
Fixed-VAE IVDD-Nys-SK	60.0 ±0.2	38.0 ±0.2	**64.6** ±0.1	49.6 ±0.2	**73.7** ±0.2	49.6 ±0.2	65.7 ±0.3	48.3 ±0.3	64.3 ±0.3	37.0 ±0.4
Deep IVDD	63.8 ±1.0	39.0 ±0.1	63.9 ±0.2	49.3 ±0.2	73.4 ±0.2	49.7 ±0.2	68.0 ±0.3	50.3 ±0.4	64.7 ±0.2	39.8 ±0.6
Conceptron	**66.0** ±0.3	40.5 ±0.4	62.5 ±0.3	44.6 ±0.4	67.5 ±1.1	47.6 ±0.3	58.0 ±1.7	49.0 ±0.3	63.7 ±0.7	42.3 ±0.3

Table 3: Average BER (over 10 seeds) on standard MNIST and CIFAR datasets.

MNIST	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Deep SVDD	**7.1** ±0.8	**5.1** ±0.2	18.1 ±2.4	16.4 ±2.0	10.9 ±0.7	20.5 ±1.1	**7.0** ±0.4	12.5 ±1.1	13.3 ±1.7	8.8 ±0.9
IVDD	42.5 ±2.3	9.2 ±1.0	44.4 ±1.4	31.2 ±3.5	32.1 ±3.0	48.9 ±0.9	36.2 ±6.5	24.1 ±0.6	34.4 ±2.4	29.5 ±3.3
IVDD Nys	9.0 ±1.7	8.0 ±0.3	27.2 ±0.5	22.6 ±0.4	18.0 ±0.4	32.1 ±0.4	15.4 ±0.3	16.5 ±0.3	29.5 ±0.3	17.2 ±0.3
Fixed-VAE IVDD	7.9 ±1.8	8.4 ±1.1	**7.3** ±0.9	**7.2** ±0.8	**7.8** ±1.1	**7.2** ±1.2	10.8 ±0.9	**7.8** ±1.6	**6.8** ±1.6	**7.3** ±1.1
Fixed-VAE IVDD-Nys	8.1 ±1.0	8.5 ±0.8	15.9 ±0.1	10.1 ±0.1	12.8 ±0.2	12.6 ±0.3	8.7 ±0.2	13.5 ±0.4	15.4 ±0.4	8.5 ±0.2
Fixed-VAE IVDD-Nys-S	9.8 ±0.2	6.0 ±0.3	15.7 ±0.1	10.1 ±0.2	12.0 ±0.3	12.5 ±0.19	9.0 ±0.2	13.3 ±0.49	12.8 ±0.3	9.9 ±0.3
Fixed-VAE IVDD-Nys-SK	8.5 ±1.7	6.0 ±0.3	16.5 ±0.2	10.1 ±0.15	12.88 ±0.3	12.5 ±0.1	7.8 ±0.3	13.6 ±0.3	12.2 ±0.3	9.1 ±0.9
Deep IVDD	9.3 ±0.4	9.2 ±0.3	14.0 ±0.2	9.9 ±0.3	11.6 ±0.6	12.1 ±0.3	9.7 ±0.8	12.0 ±0.2	11.3 ±0.8	9.3 ±0.5
Conceptron	9.4 ±0.7	8.1 ±0.8	14.9 ±4.7	10.1 ±0.2	12.1 ±0.5	13.1 ±2.3	9.9 ±0.47	12.0 ±0.3	11.5 ±0.8	8.9 ±0.3

CIFAR-10	**air**	**auto**	**bird**	**cat**	**deer**	**dog**	**frog**	**horse**	**sheep**	**truck**
Deep SVDD	48.7 ±1.6	**43.0** ±1.1	50.9 ±0.5	**45.7** ±0.9	48.8 ±0.6	**44.4** ±1.3	46.3 ±1.9	**43.0** ±1.7	**39.4** ±1.9	**40.0** ±1.2
IVDD	46.0 ±0.0	51.4 ±0.0	44.6 ±0.0	48.7 ±0.0	36.7 ±0.0	47.8 ±0.0	**37.7** ±0.0	48.1 ±0.0	41.6 ±0.0	45.3 ±0.0
IVDD Nys	46.5 ±0.7	52.0 ±0.1	46.0 ±0.4	50.3 ±0.2	36.4 ±0.6	47.4 ±0.1	38.0 ±0.8	47.2 ±0.1	41.9 ±0.4	46.1 ±0.1
Fixed-VAE IVDD	52.5 ±0.2	51.7 ±0.2	53.3 ±0.2	49.9 ±0.6	50.1 ±0.4	50.8 ±0.4	50.9 ±0.5	50.8 ±0.4	49.7 ±0.3	47.8 ±0.6
Fixed-VAE IVDD-Nys	47.8 ±0.8	50.7 ±0.5	**43.8** ±0.7	49.0 ±0.4	**34.0** ±1.4	49.1 ±0.3	40.4 ±1.1	48.4 ±0.3	41.6 ±1.1	51.4 ±0.5
Fixed-VAE IVDD-Nys-S	46.6 ±0.0	50.7 ±0.0	**43.8** ±0.0	48.8 ±0.0	34.7 ±0.0	48.9 ±0.0	40.1 ±0.0	48.5 ±0.0	42.0 ±0.0	51.4 ±0.0
Fixed-VAE IVDD-Nys-SK	46.9 ±1.0	50.7 ±0.3	44.6 ±1.0	49.3 ±0.5	34.8 ±1.7	48.8 ±0.1	40.7 ±1.5	48.6 ±0.2	42.9 ±1.2	51.5 ±0.5
Deep IVDD	45.2 ±0.6	50.6 ±0.3	45.7 ±1.3	49.7 ±0.2	37.1 ±1.1	49.2 ±0.2	42.0 ±1.4	47.7 ±0.3	42.1 ±0.7	50.9 ±0.4
Conceptron	**42.3** ±1.5	51.4 ±0.4	45.9 ±1.3	52.2 ±0.4	41.9 ±1.6	50.6 ±0.3	48.3 ±1.0	48.6 ±0.3	44.5 ±0.9	48.4 ±0.4

D CONCEPTRON ON MNIST: A SCHEMATIC REPRESENTATION

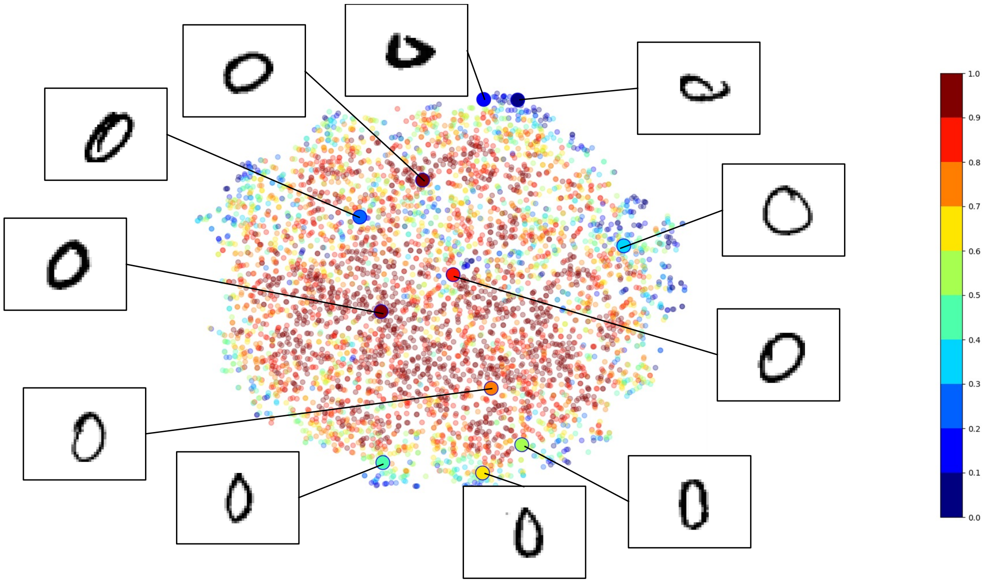

Figure 7: VAE training feature space projected in 2d via t-SNE. The color of each sample encodes the probability to be normal.

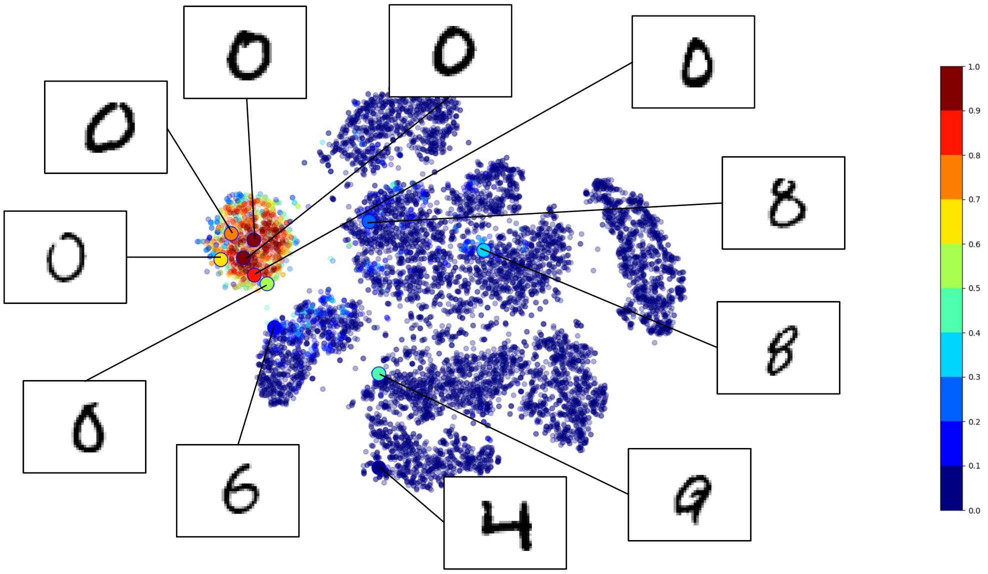

Figure 8: VAE test feature space projected in 2d via t-SNE. The color of each sample encodes the probability to be normal.

E Testing probability distributions on the MNIST dataset

In this experiment, the majority of the samples assumes low probability scores, while fewer obtain high probability scores. This is completely coherent with the unbalanced distribution of the test set, formed by many anomalous samples and a reduced part of correct samples.

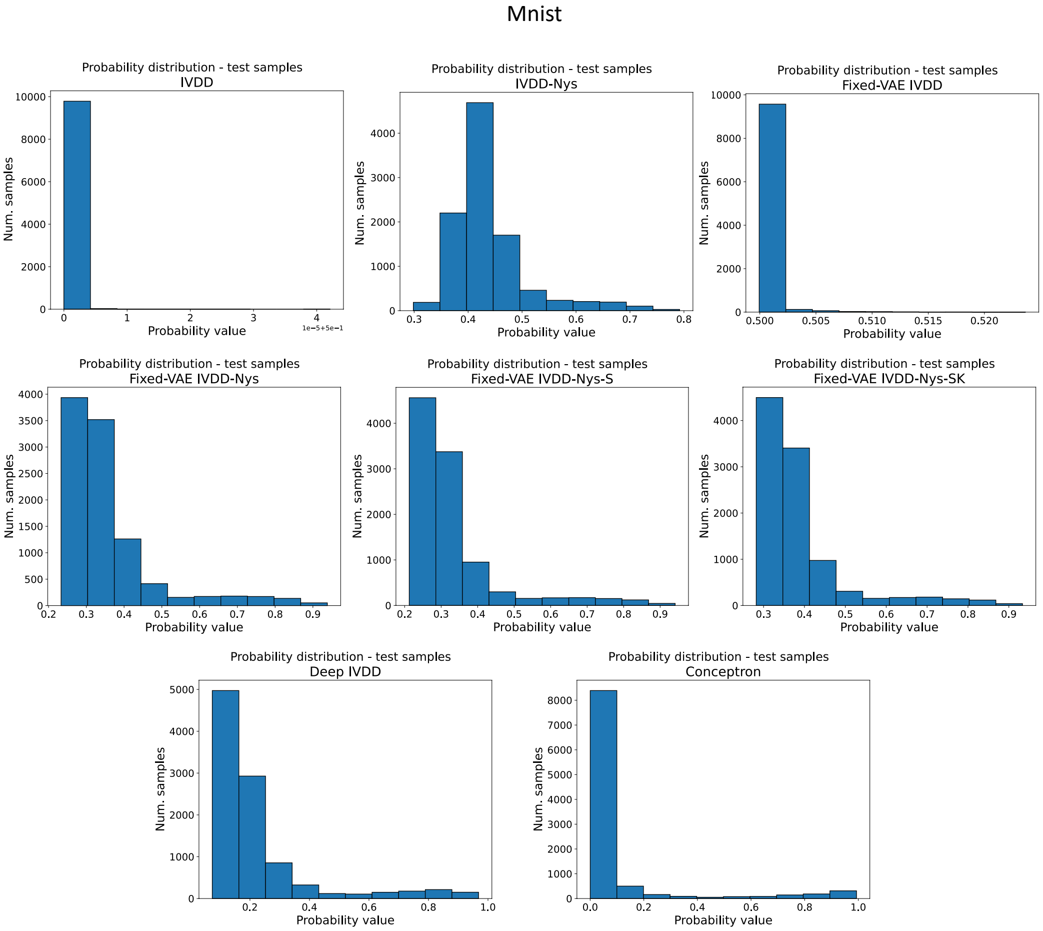

Figure 9: Probability distributions of the test samples (class 0, seed 0) for the MNIST dataset.

F TRAINING AND TESTING PROBABILITY DISTRIBUTIONS ON THE CIFAR-10 DATASET

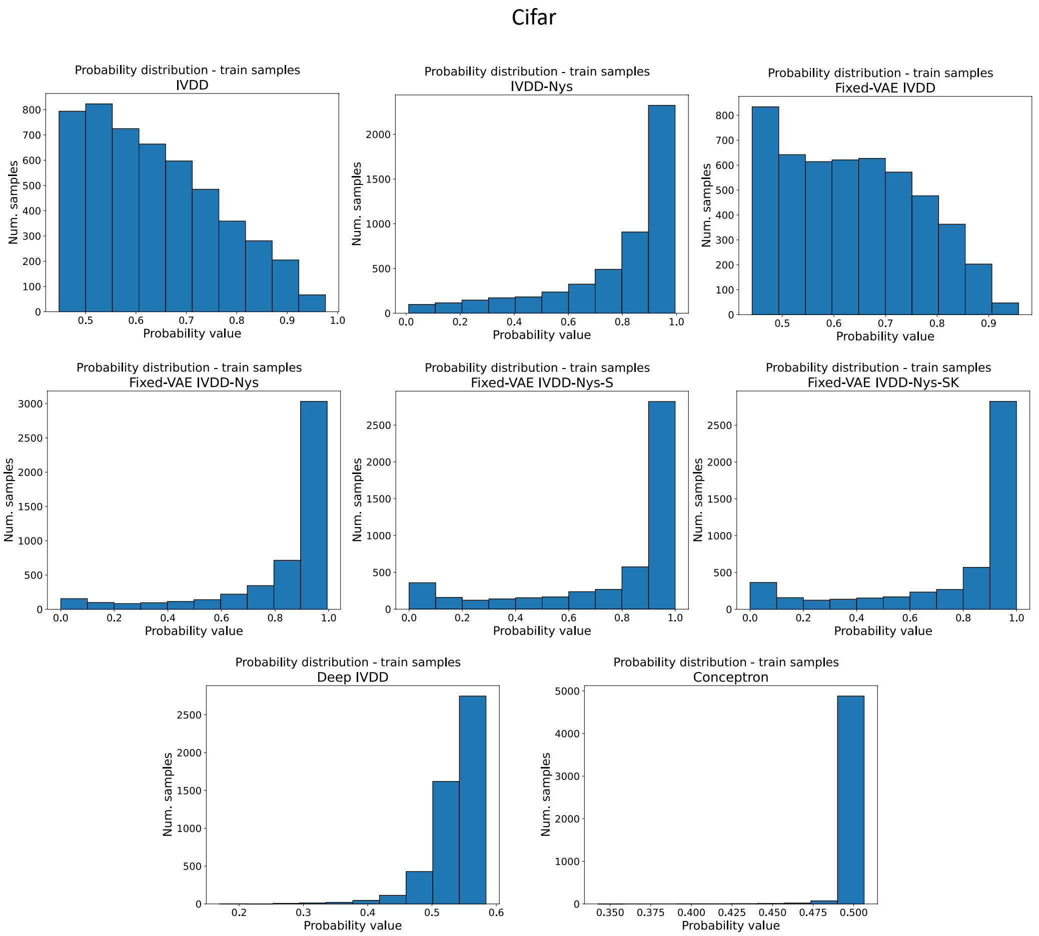

Figure 10: Probability distributions of the training samples (class airplane, seed 0) for the CIFAR-10 dataset.

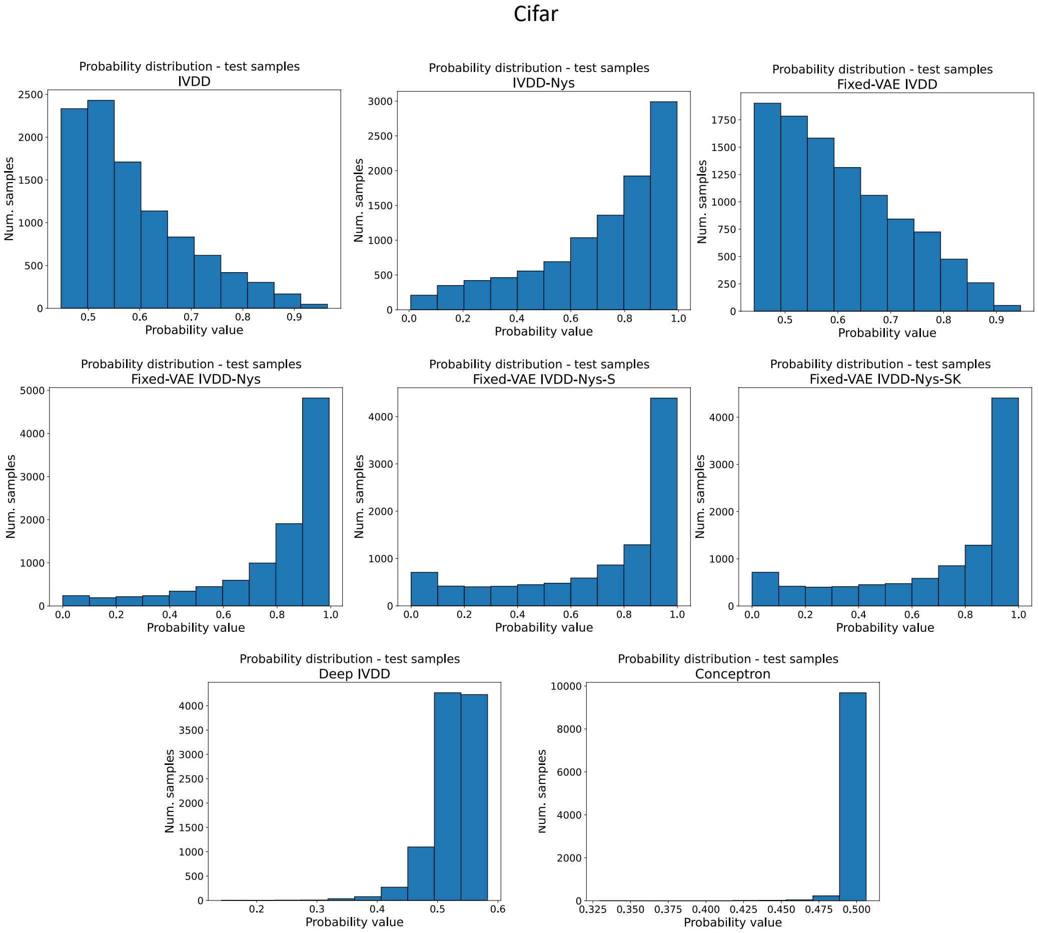

Figure 11: Probability distributions of the test samples (class airplane, seed 0) for the CIFAR-10 dataset.

G ONE-CLASS CLASSIFICATION EXPERIMENTS ON UCI DATASETS

Here we report the results of experiments on few datasets from the well-known UCI repository (Dua & Graff, 2017). We evaluate the results quantitatively via the Balanced Error Rate (BER) metric and the Area Under the Curve (AUC) metric by using the ground truth labels in testing. This metrics can be computed since all the datasets used for the experiments are classification datasets, therefore the labels are available.

Parameters are identical to MNIST experiments. To evaluate the generalization ability of the model, we repeated ten times the experiments with different seeds and results have been averaged accordingly. For each of the following experiments the class 1 is used as normal class. The other samples are considered as anomalous. A schematic representation of the samples distribution is reported in Table 4.

Table 4: Description of the standard UCI datasets.

	Training samples	**Testing samples**	**Features**
Brest cancer	443	240	10
Ionosphere	225	126	34
Musk	207	269	166
O	753	19247	16
Ozone	128	1719	72
Sonar	97	111	60

All the selected datasets are in a low-dimension space hence the deep architecture is not needed/used. This means that $\lambda = 0$ and both the training samples $\mathbf{x}_i$ and the center $\mathbf{a}$ are in the input space. In this way, it is possible to validate the scaled decision function presented in Eq. 10, combined with the SGD optimization procedure, in a fully controlled environment. Additionally, we can directly compare the methods with the IVDD method (Decherchi & Rocchia, 2016). The IVDD has been run with $\beta = 25$, $\sigma = \max_{ij}(d_{ij})/\log(n)$, and C automatically updated and tuned for including inside the sphere the 80%-90% of samples. All the other parameters remains unchanged and are described in Decherchi & Rocchia (2016).

Conceptron is run with the following parameter configuration: $n_l = 50$, $lr = 0.01$ and batch size $= n$. Both the Deep IVDD and the Conceptron decision functions are used. The results of all these experiments are summarized in Table 5 and in Table 6.

Table 5: Average AUC (over 10 seed) on standard UCI datasets.

	IVDD	**Conceptron** (Eq. 9) $n_l = 50$	**Conceptron** (Eq. 11) $n_l = 50$
Brest cancer	96.22	**97.75**	**96.75**
Ionosphere	**99.75**	99.60	99.60
Musk	98.16	**98.25**	**98.25**
O	**99.67**	99.66	99.66
Ozone	98.13	**98.20**	**98.20**
Sonar	**96.50**	**96.50**	**96.50**

From these experiments one can observe that the Conceptron results are comparable to the IVDD ones and the performances are not reduced when the Nyström approximation is used. In Figures from 12 to 23 we provide the probability distribution of the training and the testing samples for all the UCI experiments (seed 0). Similar results have been obtained for the other seeds values and can be found in the Conceptron-URL link in the references, together with all the code and output files.

Table 6: Average BER (over 10 seed) on standard UCI datasets.

	IVDD	**Conceptron**	**Conceptron**
		(Eq. 9)	(Eq. 11)
		$n_l = 50$	$n_l = 50$
Brest cancer	13.24	**12.13**	**12.13**
Ionosphere	**10.37**	11.72	11.72
Musk	**11.37**	11.38	11.39
O	9.65	**7.99**	**7.99**
Ozone	**13.81**	13.91	13.91
Sonar	**16.52**	18.93	18.93

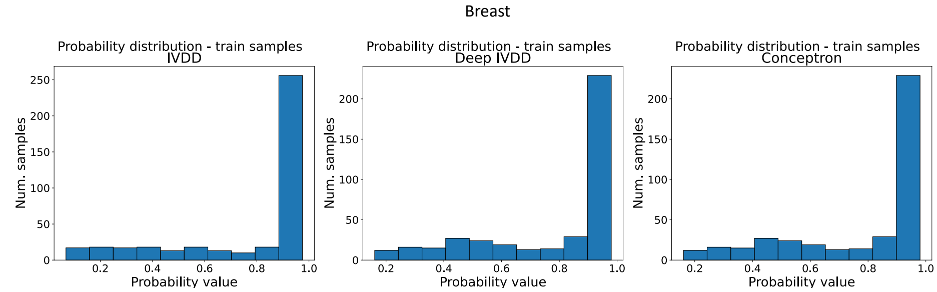

Figure 12: Probability distributions of the training samples (seed 0) for the Breast dataset.

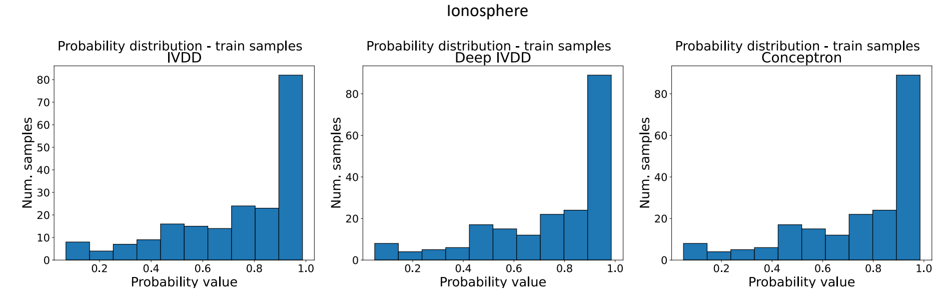

Figure 13: Probability distributions of the training samples (seed 0) for the Ionosphere dataset.

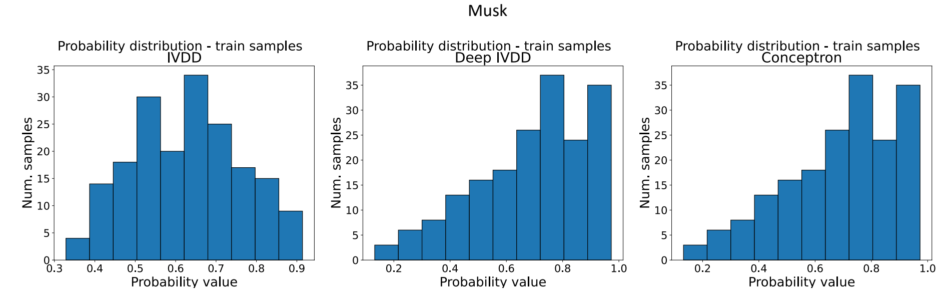

Figure 14: Probability distributions of the training samples (seed 0) for the Musk dataset.

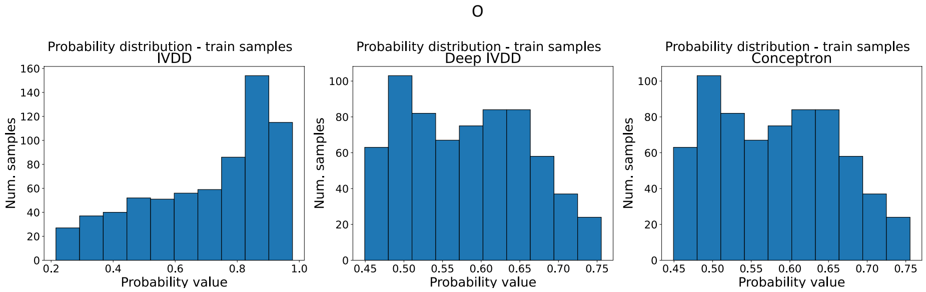

Figure 15: Probability distributions of the training samples (seed 0) for the O dataset.

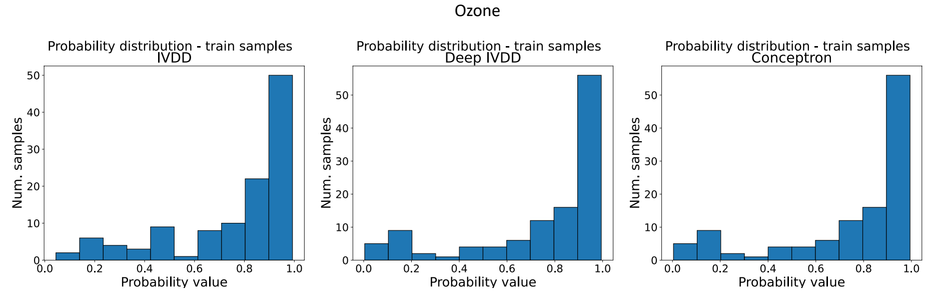

Figure 16: Probability distributions of the training samples (seed 0) for the Ozone dataset.

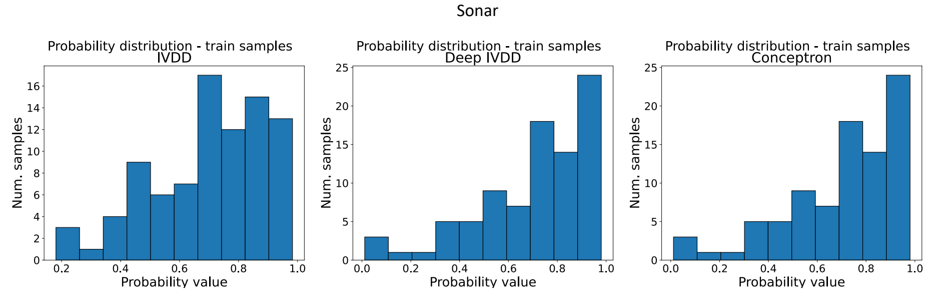

Figure 17: Probability distributions of the training samples (seed 0) for the Sonar dataset.

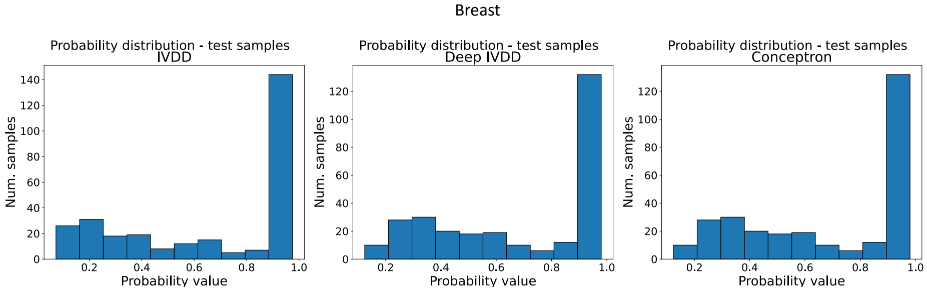

Figure 18: Probability distributions of the test samples (seed 0) for the Breast dataset.

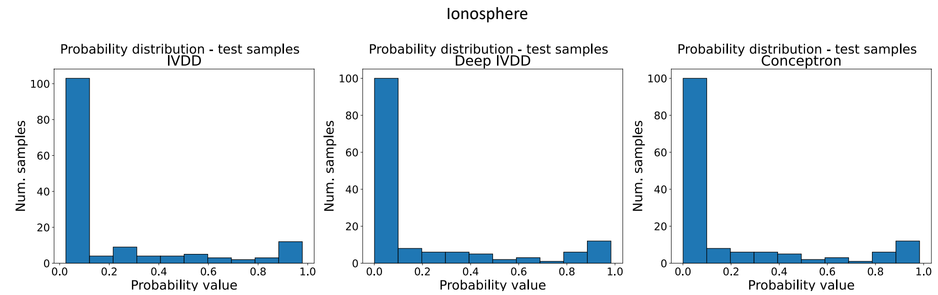

Figure 19: Probability distributions of the test samples (seed 0) for the Ionosphere dataset.

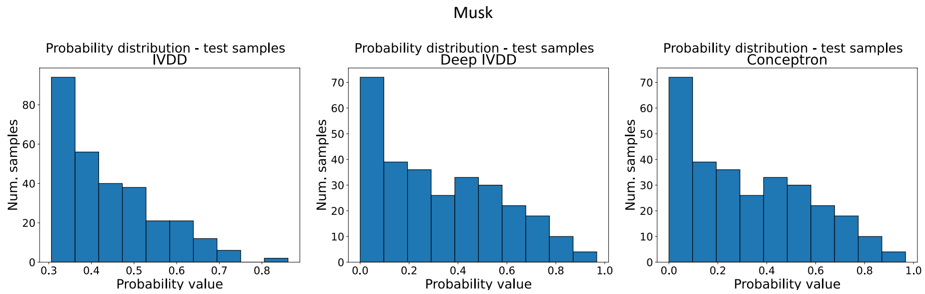

Figure 20: Probability distributions of the test samples (seed 0) for the Musk dataset.

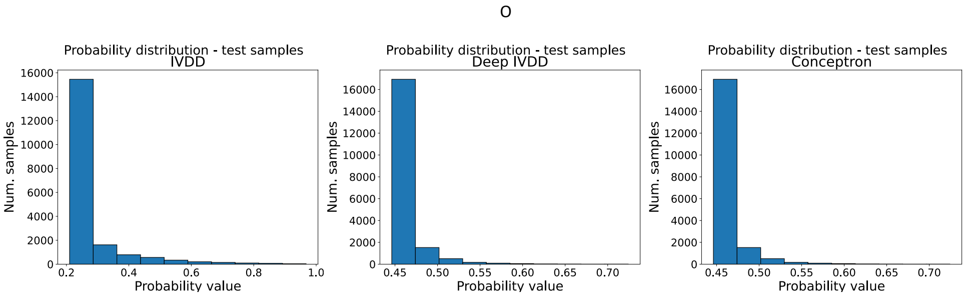

Figure 21: Probability distributions of the test samples (seed 0) for the O dataset.

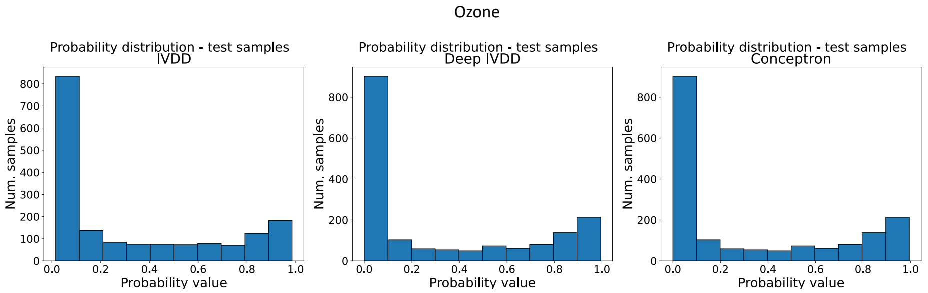

Figure 22: Probability distributions of the test samples (seed 0) for the Ozone dataset.

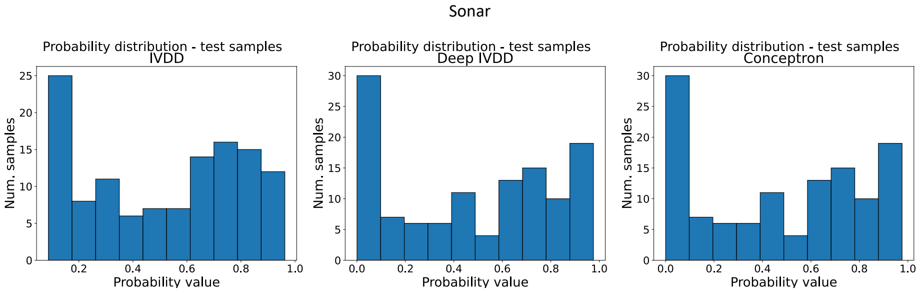

Figure 23: Probability distributions of the test samples (seed 0) for the Sonar dataset.