Certifiable Metric One Class Learning with adversarially trained Lipschitz Classifier

Louis Béthune, †
Université Paul-Sabatier, IRIT
Toulouse, France

Mathieu Serrurier
Université Paul-Sabatier, IRIT
Toulouse, France

Abstract

We propose a new Novelty Detection and One Class classifier, based on the smoothness properties of orthogonal neural network, and on the properties of Hinge Kantorovich Rubinstein (HKR) function. The classifier benefits from robustness certificates against $l2$-attacks thanks to the Lipschitz constraint, whilst the HKR loss allows to provably approximate the signed distance function to the boundary of the distribution: the normality score induces by the classifier has a meaningful interpretation in term of distance to the support. Finally, gradient steps in the input space allows free generation of samples from the one class in a fashion that reminds GAN or VAE.

Methodology

We assume we are given samples that are independently and identically sampled from a distribution $\mathbb{P}_X$ with compact support $\mathcal{X} \subset \mathbb{R}^d$. Let $\partial\mathcal{X} = \overline{\mathcal{X}}/\mathring{\mathcal{X}}$ the boundary of the distribution. Our goal is to learn the Signed Distance Function [1] to the boundary.

Definition 1 (Signed Distance Function)
Let $\mathcal{S} : \mathbb{R}^d \to \mathbb{R}$ be such that:

$$\mathcal{S}(x) = \begin{cases} d(x, \partial\mathcal{X}) & \textit{if } x \in \mathcal{X}, \\ -d(x, \partial\mathcal{X}) & \textit{otherwise.} \end{cases} \tag{1}$$

Here $d(x, \partial\mathcal{X}) = \inf_{z \in \partial\mathcal{X}} \|x - z\|_2$.

The value $\mathcal{S}(x)$ can be used as a *normality score* for one class classification.

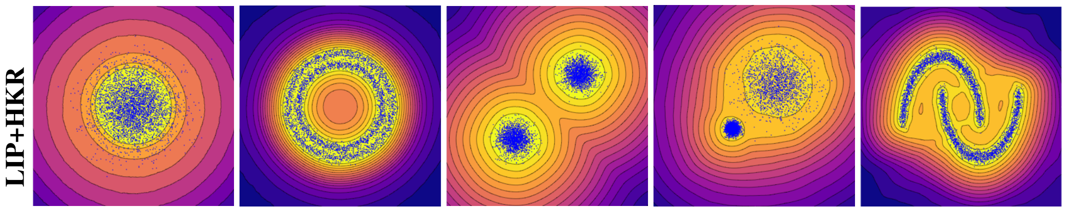
Property 1 (Regularity of SDF)**.** *$\mathcal{S}$ fulfills the Eikonal equation: $\|\nabla_x \mathcal{S}(x)\| = 1$. In particular $\mathcal{S}$ is 1-Lipschitz with respect to $l2$-nom : $\forall x, z \in \mathbb{R}^d, \|f(x) - f(z)\|_2 \leq \|x - z\|_2$.*

Orthogonal Neural Networks

Property 1 is the rational to parameterize $\mathcal{S}$ with an **orthogonal neural network** [2, 3, 4]. As noticed in [5, 6] those networks are naturally linked to the signed distance function. In particular they fulfill $\|\nabla_x f(x)\| \leq 1$ on the whole input space.

They boasts a rich literature, specially for convolutional neural networks [7, 8, 9, 10, 11, 12, 13]. In this work we use the Deel-Lip library[1]. They are based on the work of [14] that proved that universal approximation in the set of 1-Lipschitz function was possible with normalized matrices; they suggest to use orthogonal matrices in affine layers (i.e $W_i^T W_i = I$) and GroupSort activation function to counter vanishing gradient: GroupSort2$(x)_{2i,2i+1} = [\min(x_{2i}, x_{2i+1}), \max(x_{2i}, x_{2i+1})]$. Details about the architecture are given in appendix.

[1]https://github.com/deel-ai/deel-lip distributed under MIT license.



(a) One cloud. (b) Two circles. (c) Two blobs. (d) Blob and cloud. (e) Two moons.

Figure 1: Contour plots of our method with Lipschitz (LIP) orthogonal network and $\mathcal{L}^{\text{hkr}}_{m,\lambda}$ (HKR) loss on toy examples of Scikit-learn.

These networks benefit from several appealing properties: they are not subject to exploding nor vanishing gradients [11], they generalize well [15, 5], they are certifiably robust against adversarial attacks [16, 5], they are elegantly connected to optimal transport theory [17, 18], and their saliency map [19] behave like counterfactual explanations [20].

Metric One Class Learning as special case of Binary Classification

The function f_θ is an orthogonal neural network parameterized by θ. The idea is to learn by reformulating one class learning of $\mathbb{P}_X$ as a binary classification of $\mathbb{P}_X$ against a carefully chosen adversarial distribution $Q(\mathbb{P}_X)$. We show there that it enjoys great properties when applied to orthogonal neural networks, and that those theoretical results are corroborated by empirical performance.

Definition 2 ($\overset{B,\epsilon}{\sim}$ Complementary Distribution (informal))
Let Q be a distribution of compact support included in B, with disjoint support from $\mathbb{P}_X$ that "fill" the remaining space, with 2ϵ gap between $\mathcal{X}$ and supp Q. Then we write $Q \overset{B,\epsilon}{\sim} \mathbb{P}_X$.

Definition 2 captures the idea of a "complementary distribution", that can be made formal in appendix with Definition 3. Binary classification between $\mathbb{P}_X$ and any $Q \overset{B,\epsilon}{\sim} \mathbb{P}_X$ allows to build the the optimal Signed Distance Function, with the use of hinge Kantorovich Rubinstein [18] loss.

Theorem 1
Signed Distance Learning with hKR loss. *Let $\mathcal{L}^{hkr}_{m,\lambda}(yf(x)) = \lambda \max(0, m - yf(x)) - yf(x)$ be the hinge Kantorovich Rubinstein loss, with margin $m = \epsilon$, regularization $\lambda > 0$, prediction $f(x)$ and label $y \in \{-1, 1\}$. Let Q be a probability distribution on B. Let $\mathcal{E}^{hkr}(f)$ be the average loss:*

$$\mathcal{E}^{hkr}(f, \mathbb{P}_X, Q) := \mathbb{E}_{x\sim\mathbb{P}_X}[\mathcal{L}^{hkr}_{m,\lambda}(f(x))] + \mathbb{E}_{z\sim Q}[\mathcal{L}^{hkr}_{m,\lambda}(-f(z))]. \tag{2}$$

Let f^* be such that:

$$f^* \in \underset{f\in Lip_1(\mathbb{R}^d,\mathbb{R})}{\arg\inf} \; \mathcal{E}^{hkr}(f, \mathbb{P}_X, Q). \tag{3}$$

Assume that $Q \overset{B,\epsilon}{\sim} \mathbb{P}_X$. Then f^* approximates the signed distance function over B:

$$\forall x \in \mathcal{X}, \mathcal{S}(x) = f^*(x) - m, \qquad \forall z \in \textit{supp } Q, \mathcal{S}(z) = f^*(z) - m. \tag{4}$$

Moreover $sign(f(x)) = sign(\mathcal{S}(x))$ for all $x \in$ supp $Q \cup \mathcal{X}$.

If $m = \epsilon \ll 1$, then we have $f^*(x) \approx \mathcal{S}(x)$. We propose to seek Q through an alternating optimization process: at every iteration t a proposal distribution Q_t is used to train a classifier f_t against $\mathbb{P}_X$ by minimizing empirical loss. Then, the proposal distribution is updated in Q_{t+1} based on the loss induced by the optimal classifier f_t.

Finding the complementary distribution by targeting the boundary

To ensure that supp $Q \subset B$ we suggest to start from the uniform distribution: $Q_0 = \mathcal{U}(B)$. Observe that in high dimension, due to the *curse of dimensionality*, it is unlikely that a sample $z \sim Q_0$ fulfills $z \in \mathcal{X}$. Indeed the data lies on a low dimensional manifold $\mathcal{X}$ for which Lebesgue measure is negligible compared to B. Hence, in the limit of small sample size $n \ll \infty$, a sample $Z_n \sim Q_0^{\otimes n}$

fulfills $Z_n \overset{B,\epsilon}{\sim} \mathbb{P}_X$. However this is merely an advantage since the *curse* works both ways and yield a high variance in samples Z_n. Consequently the variance of the associated minimizers f_0 of equation 3 will also exhibit a high variance, which may impede generalization and convergence speed.

Instead, the distribution Q_t must be chosen to produce higher density in the neighborhood of the boundary $\partial\mathcal{X}$. The boundary is unknown, however the level set $\mathbb{L}_t = f_t^{-1}(\{-\epsilon\})$ of the classifier can be used as a proxy to improve the initial proposal Q_0. We start from $z_0 \sim Q_0$. We look for a displacement $\delta \in \mathbb{R}^d$ such that $z + \delta \in \mathbb{L}_t$. Taking inspiration from multidimensional Newton–Raphson method we consider a linearization of f_t:

$$f_t(z_0 + \delta) \approx f_t(z_0) + \langle \nabla_x f_t(z_0), \delta \rangle. \tag{5}$$

Condition $f_t(z_0 + \delta) \in \mathbb{L}_t$ translates into $f_t(z_0 + \delta) = -\epsilon$. Hence we have:

$$\delta = -\frac{f_t(z_0) + \epsilon}{\|\nabla_x f_t(z_0)\|^2} \nabla_x f_t(z_0). \tag{6}$$

The optimal displacement follows the direction of the gradient $\nabla_x f_t(z_0)$ which coincides with the direction of an optimal transportation plan, since the $\mathcal{L}^{\text{hkr}}_{m,\lambda}$ loss yields optimal transportation plans as noticed in [18]. The term $\|\nabla_x f_t(z_0)\|$ enjoys an interpretation as a Local Lipschitz Constant (see [21]) of f_t around z_0, that we know fulfills $\|\nabla_x f_t(z_0)\| \leq 1$ when parametrized with an orthogonal neural network. When f_t is trained to perfection, the expression for δ simplifies to $\delta = -f_t(z_0)\nabla_x f_t(z_0)$ thanks to Property 2.

Property 2 (Minimizers of $\mathcal{L}^{\text{hkr}}_{m,\lambda}$ are Gradient Norm Preserving (from [18]))**.** *Let f^* be the solution of Equation 3. Then for almost every $z \in B$ we have $\|\nabla_x f_t(z)\| = 1$.*

In practice the exact minimizer f^* is not always retrieved so the equation 6 applies more broadly to imperfectly fitted classifiers. The final sample $z' \sim Q_t$ is obtained by generating a sequence of T small steps to smooth the generation. The procedure is given in algorithm 1. In practice T can be chosen very low (below 16) without significantly hurting the quality of generated samples. Finally, we pick a random "learning rate" $\eta \sim \mathcal{U}([0, 1])$ for each negative example in the batch to ensure they distribute evenly on the path toward the boundary.

Remark. *In high dimension $d \gg 1$, when $\|\nabla_x f_t(z)\| = 1$ and Vol$(B) \gg$ Vol$(\mathcal{X})$ the samples obtained with algorithm 1 are approximately uniformly distributed on the level sets of f_t. It implies that the density of Q increases exponentially fast (with factor d) with respect to the value of $-|f_t(\cdot)|$. This mitigates the adverse effects of the curse of dimensionality.*

This scheme of "generating samples by following gradient in input space" reminds of diffusion models [22], feature vizualization tools [23], or recent advances in VAE [24]. However no elaborated scheme is required for the training of f_t: orthogonal networks exhibit smooth and interpretable gradients [20] which allows sampling from $\mathcal{X}$ "for free" as illustrated in figure 2.

Alternating minimization for Metric One Class learning

The sequence of classifiers f_t does not need to be trained from scratch - that would be too expensive. Instead, the same architecture $f_.$ is kept throughout training, and the algorithm produces a sequence of parameters θ_t such that $f_t = f_{\theta_t}$. Each set of parameters θ_t is used as high quality initialization for the next one θ_{t+1}. Similarly, the distribution Q_t can be updated at high pace to take advantage of the fast convergence of f_t. In the limit, each mini-batch at time step t can be sampled from a different Q_t. The final procedure is depicted in algorithm 2.

Orthogonal neural networks benefit from structural robustness [5] and certificates against $l2$-attacks [11]; hence the approximation of the $\mathcal{S}$ is robust against $l2$-adversarial attacks *by design*.

Experiments

Toy examples from Scikit-Learn

We use the toy examples of Scikit-Learn library [25] with two dimensional examples. Results are shown in figure 1. We plot the contour of the decision function in resolution 300×300 pixels.

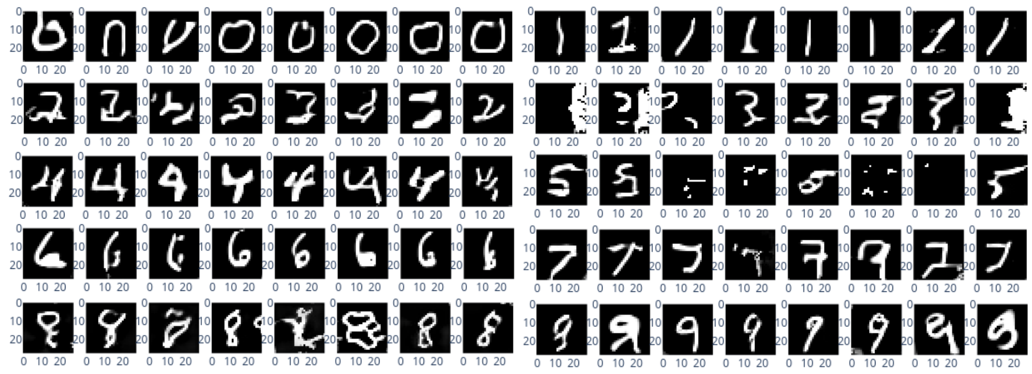


Figure 2: Synthetic examples obtained by running algorithm 1 with $T = 64$ and $\eta = 1$.

MNIST Digit	0	1	2	3	4	5	6	7	8	9
Accuracy (ours)	97.26	99.21	88.67	87.50	91.80	87.50	96.48	91.40	80.85	93.35
AUC (ours)	**99.44**	**99.84**	**94.85**	**92.94**	**95.25**	**92.64**	**98.25**	**97.11**	86.37	**96.54**
AUC DeepSVDD	98	**99.7**	91.7	91.9	**94.9**	88.5	**98.3**	94.6	**93.9**	**96.5**
AUC OCSVM	98.6	**99.5**	82.5	88.1	**94.9**	77.1	96.5	93.7	88.9	93.1
AUC IF	98.0	97.3	88.6	89.9	92.7	85.5	95.6	92.0	89.9	93.5

Table 1: AUC score and Test accuracy on the test set of MNIST in a *one versus all* fashion. We also report the AUC of DeepSVDD [28] for completeness, along with the other AUC scores of Isolation Forest (IF) and One Class SVM (OCSVM) reported in [28]. We highlight the highest ROC-AUC. When the confidence intervals overlap, we highlight both.

We compare the level sets of the classifier against the ones of One Class SVM [26] and Isolation Forest [27]. We also show the contour plot of a conventional network trained with Binary Cross Entropy against complementary distribution Q_t, and we show it struggles to learn a meaningful decision boundary. Moreover its Local Lipschitz Constant [21] increases uncontrollably, as shown in table 3. This make the conventional network prone to adversarial attacks. Moreover there is no natural interpretation of the prediction of the conventional network in term of distance: the magnitude $|f(\cdot)|$ of the predictions quickly grows above $1e-3$ whereas for orthogonal networks it is approximately equal to the signed distance function $\mathcal{S}$, as show in figure 4.

One Class on Mnist

We train a classifier on each of the classes of Mnist, and we evaluate it on test set in a *one-versus-all* fashion. Note that the *out-of-distribution* examples are not seen during training, but more importantly, the *in-distribution* examples from the test set are not seen either. Hence the task evaluates both the generalization capacity (new example from the *in-distribution*) and the discriminative capacity (against *out-of-distribution*). This setting is more challenging because of the curse of dimensionality. The AUC score is reported in table 1. The accuracy is computed by choosing the optimal threshold on decision function. The histograms are given in figure 5.

One Class Classifier Explainability with Image synthesis

Interestingly, the smoothness of the gradients of the classifier (with respect to the input) are easy to interpret. In particular, the decision boundary learned by the classifier can be materialized by generating adversarial examples with algorithm 1. The forward computation graph is a classifier based on optimal transport, and the backward computation graph an image generator. Indeed, the back-propagation through a convolution is a transposed convolution, a popular layer in the generator of GANs. In overall the algorithm behave like a WGAN [17] with a single network fulfilling both roles.

References

[1] Mikael Rousson and Nikos Paragios. Shape priors for level set representations. In *European Conference on Computer Vision*, pages 78–92. Springer, 2002.

[2] Bartłomiej Stasiak and Mykhaylo Yatsymirskyy. Fast orthogonal neural networks. In *International Conference on Artificial Intelligence and Soft Computing*, pages 142–149. Springer, 2006.

[3] Shuai Li, Kui Jia, Yuxin Wen, Tongliang Liu, and Dacheng Tao. Orthogonal deep neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 43(4):1352–1368, 2019.

[4] Jiahao Su, Wonmin Byeon, and Furong Huang. Scaling-up diverse orthogonal convolutional networks with a paraunitary framework. *arXiv preprint arXiv:2106.09121*, 2021.

[5] Louis Béthune, Thibaut Boissin, Mathieu Serrurier, Franck Mamalet, Corentin Friedrich, and Alberto González-Sanz. Pay attention to your loss: understanding misconceptions about 1-lipschitz neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[6] Fabio Brau, Giulio Rossolini, Alessandro Biondi, and Giorgio Buttazzo. Robust-by-design classification via unitary-gradient neural networks. *arXiv preprint arXiv:2209.04293*, 2022.

[7] Alexander V Gayer and Alexander V Sheshkus. Convolutional neural network weights regularization via orthogonalization. In *Twelfth International Conference on Machine Vision (ICMV 2019)*, volume 11433, page 1143326. International Society for Optics and Photonics, 2020.

[8] Jiayun Wang, Yubei Chen, Rudrasis Chakraborty, and Stella X Yu. Orthogonal convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11505–11515, 2020.

[9] Sheng Liu, Xiao Li, Yuexiang Zhai, Chong You, Zhihui Zhu, Carlos Fernandez-Granda, and Qing Qu. Convolutional normalization: Improving deep convolutional network robustness and training. *Advances in Neural Information Processing Systems*, 34:28919–28928, 2021.

[10] El Mehdi Achour, François Malgouyres, and Franck Mamalet. Existence, stability and scalability of orthogonal convolutional neural networks. *arXiv preprint arXiv:2108.05623*, 2021.

[11] Qiyang Li, Saminul Haque, Cem Anil, James Lucas, Roger B Grosse, and Jörn-Henrik Jacobsen. Preventing gradient attenuation in lipschitz constrained convolutional networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, Cambridge, MA, 2019. MIT Press.

[12] Asher Trockman and J Zico Kolter. Orthogonalizing convolutional layers with the cayley transform. In *International Conference on Learning Representations*, 2021.

[13] Sahil Singla and Soheil Feizi. Skew orthogonal convolutions. In *International Conference on Machine Learning*, pages 9756–9766. PMLR, 2021.

[14] Cem Anil, James Lucas, and Roger Grosse. Sorting out lipschitz function approximation. In *International Conference on Machine Learning*, pages 291–301. PMLR, 2019.

[15] Peter L Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research*, 20:63–1, 2019.

[16] Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 31, pages 6541–6550. Curran Associates, Inc., 2018.

[17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[18] Mathieu Serrurier, Franck Mamalet, Alberto González-Sanz, Thibaut Boissin, Jean-Michel Loubes, and Eustasio Del Barrio. Achieving robustness in classification using optimal transport with hinge regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 505–514, 2021.

[19] K Simonyan, A Vedaldi, and A Zisserman. Deep inside convolutional networks: visualising image classification models and saliency maps. pages 1–8. ICLR, 2014.

[20] Mathieu Serrurier, Franck Mamalet, Thomas Fel, Louis Béthune, and Thibaut Boissin. When adversarial attacks become interpretable counterfactual explanations. *arXiv preprint arXiv:2206.06854*, 2022.

[21] Matt Jordan and Alexandros G Dimakis. Exactly computing the local lipschitz constant of relu networks. *Advances in Neural Information Processing Systems*, 33:7344–7353, 2020.

[22] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[23] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization.

[24] Victor Boutin, Aimen Zerroug, Minju Jung, and Thomas Serre. Iterative vae as a predictive brain model for out-of-distribution generalization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[25] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[26] 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.

[27] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE, 2008.

[28] 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*, pages 4393–4402. PMLR, 2018.

[29] Mohammad Sabokrou, Mohammad Khalooei, Mahmood Fathy, and Ehsan Adeli. Adversarially learned one-class classifier for novelty detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3379–3388, 2018.

[30] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. 2018.

[31] Åke Björck and Clazett Bowie. An iterative algorithm for computing the best estimate of an orthogonal matrix. *SIAM Journal on Numerical Analysis*, 8(2):358–364, 1971.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [No]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Code is not included but appendix contains enough details to allow reproducibility.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Hardware is given.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes]

(b) Did you mention the license of the assets? [Yes]

(c) Did you include any new assets either in the supplemental material or as a URL? [No]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Proofs and comments

Complementary distribution

Definition 3 ($\overset{B,\epsilon}{\sim}$ Complementary Distribution)
Let $\mathbb{P}_X$ a distribution with compact support $\mathcal{X} \subset B$, with $B \subset \mathbb{R}^d$ a bounded measurable set. Q is said to be (B,ϵ) disjoint from $\mathbb{P}_X$ if (i) its support supp $Q \subset B$ is compact (ii) $d(\text{supp } Q, \mathcal{X}) \geq 2\epsilon$ (iii) for all measurable sets $M \subset B$ such that $d(M, \mathcal{X}) \geq 2\epsilon$ we have $Q(M) > 0$. It defines a symmetric but irreflexive binary relation denoted $Q \overset{B,\epsilon}{\sim} \mathbb{P}_X$.

The idea is to learn one class classifier by reformulating one class learning of $\mathbb{P}_X$ as a binary classification of $\mathbb{P}_X$ against a carefully chosen adversarial distribution $Q(\mathbb{P}_X)$. This simple idea had already occurred repeatedly in the related literature [29]. Note that $\mathcal{L}^{\text{hkr}}_{m,\lambda}$ benefit from generalization guarantees as proved in [5]: the optimal classifier on the train set and on the test set are the same in the limit of big samples.

Theorem 1
Signed Distance Learning with hKR loss. *Let $\mathcal{L}^{hkr}_{m,\lambda}(yf(x)) = \lambda \max(0, m - yf(x)) - yf(x)$ be the hinge Kantorovich Rubinstein loss, with margin $m = \epsilon$, regularization $\lambda > 0$, prediction $f(x)$ and label $y \in \{-1, 1\}$. Let Q be a probability distribution on B. Let $\mathcal{E}^{hkr}(f)$ be the average loss:*

$$\mathcal{E}^{hkr}(f, \mathbb{P}_X, Q) := \mathbb{E}_{x\sim\mathbb{P}_X}[\mathcal{L}^{hkr}_{m,\lambda}(f(x))] + \mathbb{E}_{z\sim Q}[\mathcal{L}^{hkr}_{m,\lambda}(-f(z))]. \tag{2}$$

Let f^* be such that:

$$f^* \in \operatorname*{arg\,inf}_{f \in Lip_1(\mathbb{R}^d,\mathbb{R})} \mathcal{E}^{hkr}(f, \mathbb{P}_X, Q). \tag{3}$$

Assume that $Q \overset{B,\epsilon}{\sim} \mathbb{P}_X$. Then f^* approximates the signed distance function over B:

$$\forall x \in \mathcal{X}, \mathcal{S}(x) = f^*(x) - m, \qquad \forall z \in \text{supp } Q, \mathcal{S}(z) = f^*(z) - m. \tag{4}$$

Moreover sign$(f(x))$ = sign$(\mathcal{S}(x))$ for all $x \in$ supp $Q \cup \mathcal{X}$.

Proof. The results follows from the properties of $\mathcal{L}^{\text{hkr}}_{m,\lambda}$ loss given in Proposition 2 of [18]. If $Q \overset{B,\epsilon}{\sim} \mathbb{P}_X$ then by definition the two datasets are 2ϵ separated. Consequently the hinge part of the loss is null: $\max(0, m - yf(x))$ for all pairs $(x, +1)$ and $(z, -1)$ with $x \sim \mathbb{P}_X$ and $z \sim Q$. We deduce that:

$$\forall x \in \mathcal{X}, f(x) \geq m \qquad \forall z \in \text{supp } Q, f(z) \leq -m \tag{7}$$

Since $m = \epsilon$ we must have $f(\partial x) = m$ for all $x \in \partial\mathcal{X}$ and $f(\partial z) = -m$ for all $\partial z \in \partial(\text{supp } Q)$. Whereas $Sdf(\partial x) = 0$ and $Sdf(\partial z) = -2m$. Now observe that thanks to the $-yf(x)$ term in $\mathcal{L}^{\text{hkr}}_{m,\lambda}$ loss maximizes the amplitude $|f(x)|$, and any other choice is sub-optimal without violating the 1-Lipschitz constraint. Hence for every $x \in \mathcal{X}$ we have $f(x) = f(\partial x) + \|x - \partial x\|$ where $\partial x =$

$\arg\min_{\bar{x}\in\partial\mathcal{X}} \|x-\bar{x}\|$ is the projection of x onto the boundary $\partial\mathcal{X}$. Similarly $f(z) = f(\partial z)-\|z-\partial z\|$. Notice that $\mathcal{S}(x) = \mathcal{S}(\partial x) + \|x - \partial x\|$ and $\mathcal{S}(z) = \mathcal{S}(\partial z) - \|z - \partial z\|$. This allows to conlude:

$$\forall x \in \mathcal{X}, \mathcal{S}(x)) = f^*(x) - m, \qquad \forall z \in \text{supp } Q, \mathcal{S}(z) = f^*(z) - m. \tag{8}$$

□

Finding the right distribution $Q \overset{B,\epsilon}{\sim} \mathbb{P}_X$ with small ϵ is also challenging.

We propose to seek Q through an alternating optimization process. Consider a sample $z \in \mathbb{L}_t$. If z is a *false positive* (i.e $f_t(z) > 0$ and $z \notin \mathcal{X}$), by training f_{t+1} on the pair $(z, -1)$ it will incentive f_{t+1} to fulfill $f_{t+1}(z) < 0$, and that will have for consequence to reduce the volume of false positive associated to f_{t+1}. If z is a *true negative* (i.e $f_t(z) < 0$ and $z \notin \mathcal{X}$) it already exhibit the wanted properties. The case of *false negative* (i.e $f_t(z) < 0$ and $z \in \mathcal{X}$) is more tricky: the density of $\mathbb{P}_X$ around z will play an important role to ensure that $f_{t+1}(z) > 0$.

Hence we assume that samples from the target $\mathbb{P}_X$ are *significantly* more frequent than the ones obtained from pure randomness. This is a very reasonable assumption (specially for images for example), and most distributions from real use-cases fall under this setting.

Assumption 1 ($\mathbb{P}_X$ samples are more frequent than pure randomness (informal))
For any measurable set $M \subset \mathcal{X}$ we have $\mathbb{P}_X(M) \gg \mathcal{U}(M)$.

Algorithm 1: Adapted Newton–Raphson for Complementary Distribution Generation

```
Input: orthogonal neural network $f_t$
Parameter: number of steps $T$
Output: sample $z' \sim Q_t(f)$
1: sample learning rate $\eta \sim \mathcal{U}([0,1])$
2: $z_0 \sim \mathcal{U}(B)$ { Initial approximation.}
3: for each step $t = 1$ to $T$ do
4:     $z_{t+1} \leftarrow z_t - \frac{\eta}{T}\frac{\nabla_x f(z^t)}{\|\nabla_x f(z^t)\|_2^2}(f(z_t) + \epsilon)$ {Refine estimation.}
5:     $z_{t+1} \leftarrow \Pi_B(z_{t+1})$ { Ensure it remains in feasible set.}
6: end for
7: return $z_T$
```

To ensure that property (iii) of definition 3 is fulfilled, we also introduce stochasticity in algorithm 1 by picking a random "learning rate" $\eta \sim \mathcal{U}([0,1])$. The learning rate is sampled independently for each example in the batch, in order to decorrelate samples.

The final procedure depicted in algorithm 2 benefit from the mild guarantee of proposition 1. This guarantees that once the complementary distribution has been found, the algorithm will continue to produce a sequence of complementary distributions, and a sequence of classifiers f_t that approximates $\mathcal{S}$.

One Class learning framed as Adversarial Training

Proposition 1 (Complementary distributions are fix points)**.** *Let Q^t be such that $Q^t \overset{B,\epsilon}{\sim} \mathbb{P}_X$. Assume that Q^{t+1} is obtained with algorithm 2. Then we have $Q^{t+1} \overset{B,\epsilon}{\sim} \mathbb{P}_X$.*

Proof. The proof also follows from the properties of $\mathcal{L}^{\text{hkr}}_{m,\lambda}$ loss given in Proposition 2 of [18] Since $Q_t \overset{B,\epsilon}{\sim} \mathbb{P}_X$ all examples $z \sim Q_t$ generated fulfill (by definition) $d(z, \mathcal{X}) \geq 2\epsilon \geq 2m$. Indeed the 1-Lipschitz constraint (in property 2) guarantees that no example z_t can "overshoot" the boundary. Hence for the associated minimizer f_{t+1} of $\mathcal{L}^{\text{hkr}}_{m,\lambda}$ loss, the hinge part of the loss is null. This guarantees that $f_{t+1}(z) \leq -m$ for $z \sim Q$. We see that by applying algorithm 1 the property is preserved: for all $z \sim Q_{t+1}$ we must have $f_{t+1}(z) \leq -m = -\epsilon$. Finally notice that because $z_0 \sim \mathcal{U}(B)$ and $\eta \sim \mathcal{U}([0,1])$ the support supp Q covers the whole space B (except the points that are less than 2ϵ apart from $\mathcal{X}$). Hence we have $Q_{t+1} \overset{B,\epsilon}{\sim} \mathbb{P}_X$ as expected. □

Algorithm 2: Alternating Minimization for Metric One Class learning

```
Input: orthogonal neural network architecture $f_\circ$
Input: initial weights $\theta_0$, learning rate $\alpha$
1: repeat
2:     $f_t \leftarrow f_{\theta_t}$
3:     Generate batch $z \sim Q_t$ of negative samples with algorithm 1
4:     Sample batch $x \sim \mathbb{P}_X$ of positive samples
5:     Compute loss on batch $\mathcal{L}(\theta_t) := \mathcal{E}^{\text{hkr}}(f_{\theta_t}, x, z)$
6:     Learning step $\theta_{t+1} \leftarrow \theta_t + \alpha\nabla_\theta\mathcal{L}(\theta_t)$
7: until convergence of $f_t$.
```

The procedure of algorithm 2 bears numerous similarities with the adversarial training of Madry [30]. In our case the adversarial examples are obtained by starting from noise $\mathcal{U}(B)$ and relabeled are negative examples. In their case the adversarial examples are obtained by starting from $\mathbb{P}_X$ itself and relabeled as positive examples.

Experimental setting

Orthogonal Neural Networks

We ensure that the kernel remains orthogonal by re-parametrizing them: $\Theta_i = \Pi(W_i)$ where W_i is a set of unconstrained weights, and Θ_i the orthogonal projection of W_i on the Stiefel manifold - i.e the set of orthogonal matrices. The projection Π is made with Björck algorithm [31] which is differentiable and be included in computation graph during forward pass. Unconstrained optimization is performed on W_i directly.

Toy experiment in 2D

All datasets are normalized to have zero mean and unit variance across all dimensions. The domain B is chosen to be the ball of radius 5. This guarantees that $\mathcal{X} \subset B$ for all datasets. The plots of figure1 are squares of sizes $[-5, 5]$ for **(a)**, $[-3, 3]$ for **(b)(c)(e)** and $[-4, 4]$ for **(d)** to make the figure more appealing.

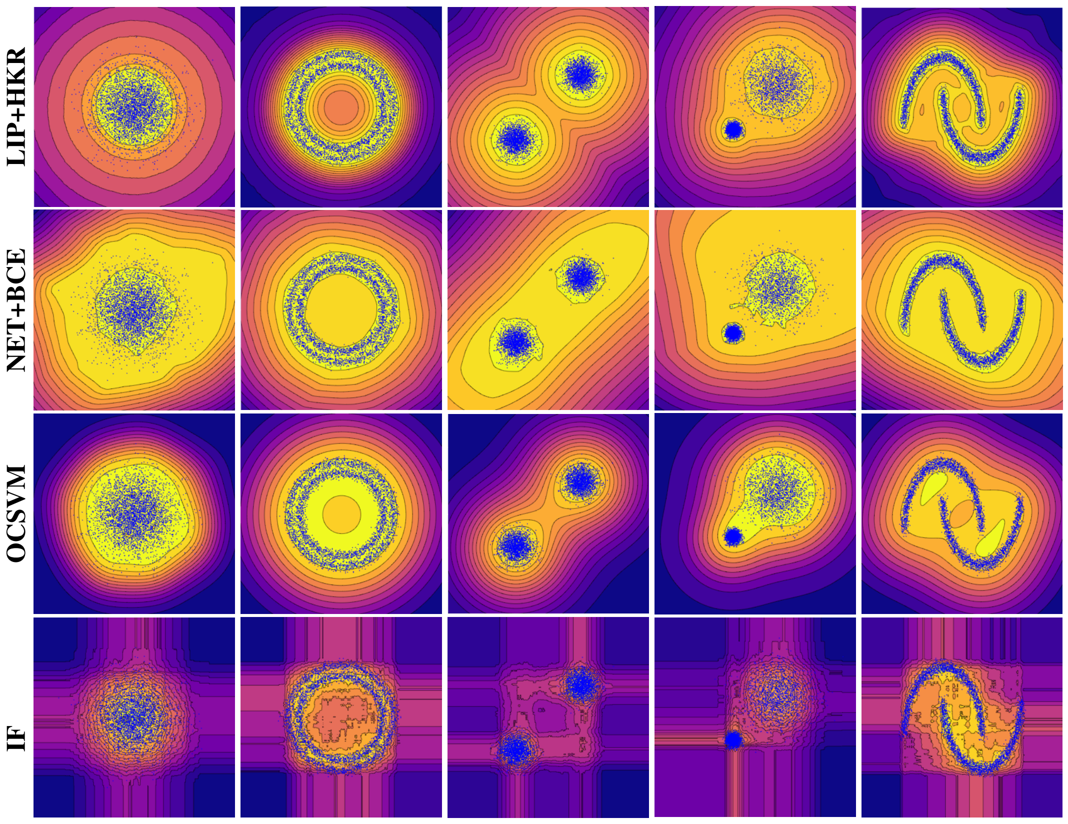
Metric One Class Learning

All the toy experiments of figure 1 uses a $2 \rightarrow 512 \rightarrow 512 \rightarrow 512 \rightarrow 512 \rightarrow 1$ neural network. All the squares matrices are constrained to be orthogonal. The last layer is a unit norm column vector. The first layer consists of two unit norm columns that are orthogonal to each other. The optimizer is Rmsprop with default hyper-parameters. The batch size is $b = 256$ and the number of steps $T = 4$ is small. We chose a margin $m = 0.05$ except for "blob and cloud" dataset where we used $m = 0.1$ instead. We take $\lambda = 100$. The networks are trained for a total of $10,000$ gradient steps.

Other benchmarks

For One Class SVM we chose a parameter $\nu = 0.05$, $\gamma = \frac{1}{2}$ (which corresponds to the "scale" behavior of scikit-learn for features in 2D with unit variance) and the popular RBF kernel. For Isolation Forest we chose a default contamination level of 0.05.

The conventional network (without orthogonality constraint) is trained with Binary Cross Entropy (also called log-loss) and Adam optimizer. It shares the same architecture, with ReLU instead of GroupSort. It is also trained for a total of $10,000$ gradient steps for fair comparison. It would not make sense to use $\mathcal{L}^{\text{hkr}}_{m,\lambda}$ loss for conventional network since it diverges during as noticed in [5]. The Lipschitz constant of conventional network grows uncontrollably during training even with $\mathcal{L}^{\text{hkr}}_{m,\lambda}$ loss, which is also compliant with the results of [5].



(a) One cloud. (b) Two circles. (c) Two blobs. (d) Blob and cloud. (e) Two moons.

Figure 3: Toy examples of Scikit-learn. **Top row**: our method with Lipschitz (LIP) orthogonal network and $\mathcal{L}^{\text{hkr}}_{m,\lambda}$ (HKR) loss. **Second row**: conventional network (NET) trained with Binary Cross Entropy (BCE). **Third row**: One Class SVM. **Fourth row**: Isolation Forest.

Mnist experiment

The MNIST images are normalized such that pixel intensity lies in $[-1, 1]$ range. The set B is chosen to be the image space, i.e $B = [-1, 1]^{28\times 28} = [-1, 1]^{784}$. The optimizer is Adam with default hyper-parameters. We chose $m = 0.02 \times \sqrt{(28 \times 28 \times (1 - (-1))} \approx 0.79$: this corresponds to modification of 2% of the maximum norm of an image. We take $\lambda = 200$. We use a batch size $b = 128$ a number of steps $T = 16$. The network is trained for a total of 1000 epochs over the one class (size of the support: ≈ 5000 examples).

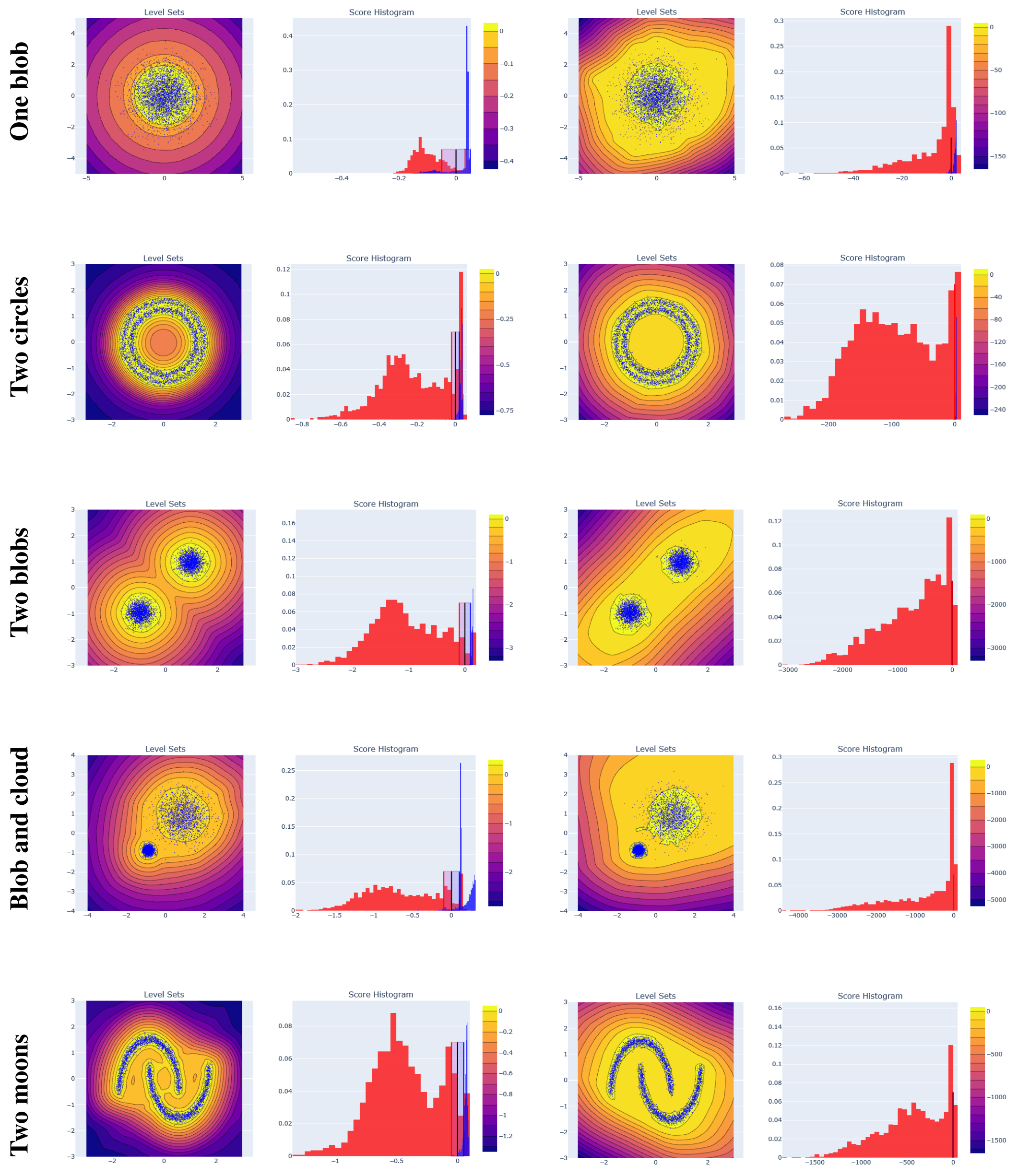
All the experiments use a VGG-like architecture depicted in table 2. Convolutions are parametrized using layers of Deel-Lip library [18], that uses orthogonal kernel with a corrective constant on the upper bound of the Lipschitz constant of the kernel. Dense layers also use orthogonal kernel. We use *l2-norm-pooling* layer with windows of size 2×2 that operates as: $(x_{11}, x_{12}, x_{21}, x_{22}) \mapsto \|[x_{11}, x_{12}, x_{21}, x_{22}]\|$.

We see in table 1 that it yields competitive results against other naive baselines such as Isolation Fortest (IF) and One Class SVM (OCSVM), and against the popular deep learning algorithm *Deep SVDD* [28].

We report the histogram of predictions for the train set (One class), the set test (One class) and Out Of Distribution (OOD) examples (the classes of the test set) in figure 5.

Image synthesis from one class classifier

We perform a total of $T = 64$ steps with algorithm 1 by targeting the level set $f^{-1}(\{2m\})$ (to ensure we are inside the distribution). Moreover during image synthesis we follow a deterministic



(a) Lipschitz Network and $\mathcal{L}^{\text{hkr}}_{m,\lambda}$.

(b) Conventional Network and BCE.

Figure 4: Histograms of score functions for orthogonal network (left) and conventional neural network. The blue bars correspond to the distribution of $f(x), x \sim \mathbb{P}_X$ and the red bars to the distribution $f(z), z \sim \mathcal{U}(B)$.

path by setting $\eta = 1$. The images generated were picked at random (with respect to initial sample $z_0 \sim \mathcal{U}(B)$) without cherry picking. Interestingly, we see that the algorithm recovers some *in-distribution* examples successfully. The examples for which the image is visually deceptive are somewhat correlated with low AUC score. Those failure cases are also shared by concurrent methods, which suggests that some classes are harder than other to distinguish. Notice that sometimes OOD MNIST digits, from *other classes not seen during training*, are sometimes generated. This is the case of the class "7" for example inside which we find images that look like a "9" or a "2". This suggests

network architecture
conv-3x3-128 (fullsort)
l2 norm pooling - 2x2
conv-3x3-128 (fullsort)
l2 norm pooling - 2x2
conv-3x3-128 (fullsort)
l2 norm pooling - 2x2
conv-3x3-128 (fullsort)
global l2 norm pooling
dense-128 (fullsort)
dense-128 (fullsort)
dense-128 (fullsort)
dense-1 (linear)

Table 2: Architecture of the ConvNet used for MNIST experiment. 872K parameters.

Daatset	One cloud	Two circles	Two blobs	Blob and cloud	Two moons
LLC conventional	26.66	122.84	1421.41	53.90	258.73

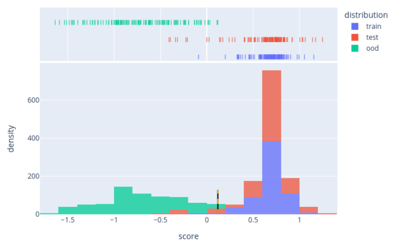
Table 3: Lower bound on the Local Lipschitz Constant (LLC) of conventional network after $10,000$ training steps. It is obtained by computing the maximum of $\|\nabla_{x_i} f(x_i)\|$ over the train set.

that most MNIST digits can be build from a small sets of elementary features that are combined during generation of Q_t.

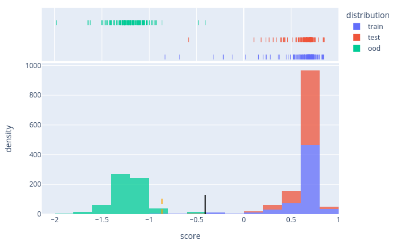
Finally note that are our method is not tailored for example generation: this is merely a side effect of the training process of the classifier. There is no need a the encoder-decoder pair of VAE nor the discriminator-generator pair of a GAN. Moreover no hyper-parameters other than m and T are required.

Hardware

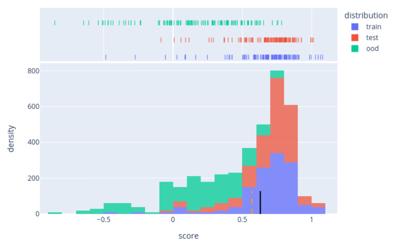
The hardware consist on a workstation with NVIDIA 1080 GPU with 8GB memory and a machine with 32GB RAM.



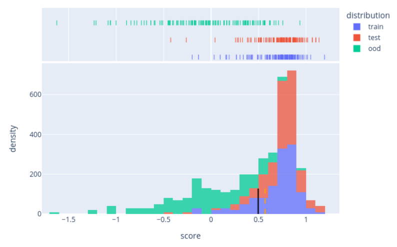
(a) Class “0”.



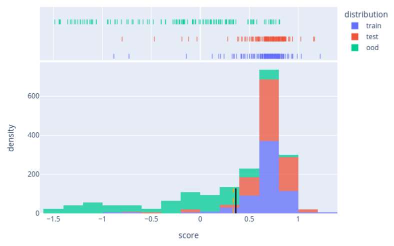
(b) Class “1”.



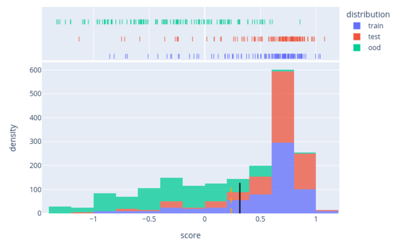
(c) Class “2”.



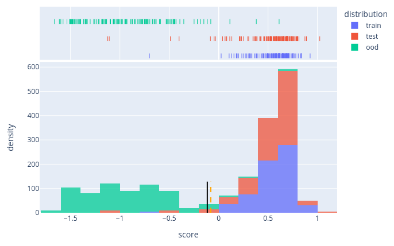
(d) Class “3”.



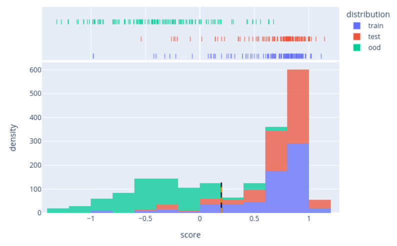
(e) Class “4”.



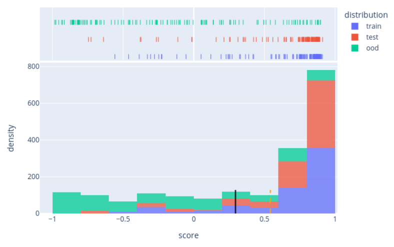
(f) Class “5”.



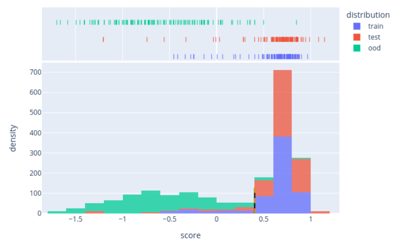
(g) Class “6”.



(h) Class “7”.



(i) Class “8”.



(j) Class “9”.

Figure 5: Histogram of scores predicted by the classifier at the end of training for **train** examples, **test** examples, and **OOD Test** examples.