One-Class Dense Networks for Anomaly Detection

Norman Karr
Lawrence Berkeley National Laboratory, Berkeley
University of California, Berkeley
nkarr11@berkeley.edu

Benjamin Nachman
Physics Division
Lawrence Berkeley National Laboratory, Berkeley
Berkeley Institute for Data Science
University of California, Berkeley
bpnachman@lbl.gov

David Shih
New High Energy Theory Center
Rutgers University, Piscataway
dshih@physics.rutgers.edu

Abstract

Unsupervised learning has been proposed as a tool for model agnostic anomaly detection (AD) in collider physics. While the goal of these approaches is usually to find events that are 'rare' under the Standard Model hypothesis, many approaches are governed by heuristics to strive towards an implicit density estimator. We study the simplest possible one-class classification method for unsupervised AD and show that it has similar properties to other unsupervised methods. The method is illustrated using a Gaussian dataset and a simulation of the events at the Large Hadron Collider (LHC). The simplicity of the one-class classification may enable a deeper understanding of unsupervised AD in the future.

1 Introduction

Given the lack of new particle discoveries at the Large Hadron Collider (LHC) and elsewhere, there is a growing need to broaden our sensitivity to new physics through search strategies that are less reliant on theoretical models. Machine learning (ML) based anomaly detection (AD) is a promising direction for broadening our sensitivity to unforeseen scenarios (see e.g. Ref. [1] for a recent review) and early results with collider data are now coming online [2, 3].

A large number of ML-based AD proposals are based on unsupervised learning. Generically, the goal of these approaches is to explicitly or implicitly estimate the density of the background-only data $p_B(x)$ for some features $x \in \mathbb{R}^N$. Then, anomalies in the target dataset are selected as events that are rare / low density[1]. A virtue and vice of these approaches is that their performance does not depend on the amount of signal in the target dataset [6, 7]. This is a desirable feature when there is little signal in the target dataset, but it also means that these models are only sensitive if the new physics happens to be concentrated in specific areas or directions relative to background-only training data.

There are many strategies for constructing unsupervised ML-based AD methods. Normalizing flows [8–11] can be used to explicitly learn the data probability density[2] while a number of implicit methods like autoencoders are also well-studied in collider physics (see e.g. Ref. [16–18] and papers that cite them). Approaches vary in their complexity and ease-of-use and since there is no universally optimal method, it is important to consider a variety of techniques in order to achieve broad coverage.

[1]This approach is coordinate-dependent [4, 5].

[2]Normalizing flows have also been proposed for weakly supervised learning [12–15].

For this reason, we explore possibly the simplest unsupervised deep-learning approach to AD that we call a one-class dense network (OCDN). We train a neural network classifier using the standard binary cross entropy loss function, but using only background (class 0) inputs. For a sufficiently flexible neural network architecture and a long enough training schedule, the classifier should be able to return a score of zero for all inputs. However, for a finite number of training epochs, the network will have a non-trivial functional form and the resultant scores can be used for AD. The simplicity of the OCDN may provide a platform for diving deeper into understanding unsupervised AD in the future. In this paper, we explore the OCDN on Gaussian and particle physics datasets.

2 One-class Methods

The 'new' method that we explore here is the one-class dense network (OCDN). The quotes around *new* in the previous sentence are because the OCDN is such a simple construction that surely it has been explored before[3]. However, its simplicity may have kept it hidden from further study. Using any relevant neural network architecture, the OCDN is trained using the usual binary cross entropy (BCE) loss function for classification. The difference with respect to normal classification is that only one class is provided during training. The function that minimizes this loss is the constant function 0 (for the input class labeled 0). For any finite epoch during training, the function will approach zero, but its deviation from a constant allows it to have some AD performance.

To see how the OCDN might be useful, consider the simplest neural network: $f(x) = 1/(1 + e^{-(ax+b)})$, i.e. logistic regression. Optimizing this neural network for the BCE loss L is a two-dimensional problem so one can even visualize the trajectory through function space. The partial derivatives are $\frac{\partial L}{\partial a} = \mathbb{E}[Xf(X)]$ and $\frac{\partial L}{\partial b} = \mathbb{E}[f(X)]$. Since $f(x) > 0$, $\frac{\partial \mathcal{L}}{\partial b} \to 0$ requires that $f(x) \to 0$ across the support of X. This means that $f(x) > 0$ away from where the training data have support and therefore may be able to detect out of distribution anomalies (at least in one direction).

We compare the OCDN to a variety of other, similar one-class methods. The first model we test is a one-class support vector machine (OC-SVM) [20]. This method finds a hyperplane that best separates data from the origin in a given feature space. Symbolically:

$$\min_{\boldsymbol{\omega},\rho,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{\omega}\| - \rho + \frac{1}{\nu n}\sum_{i=1}^{n} \xi_i\,, \qquad \langle \boldsymbol{\omega}, \phi_k(\mathbf{x}_i)\rangle \geq \rho - \xi_i\,,\ \xi_i \geq 0, \forall i\,, \tag{1}$$

where ϕ_k is a kernel function that transforms the data into a new feature space. In our experiments we use the radial basis function (RBF) kernel. $\boldsymbol{\omega}$ is the hyperplane and $\boldsymbol{\xi}$ is a slack variable. The ν variable is a hyperparameter in the range $(0, 1]$ that correlates with the fraction of datapoints that are permitted to lie outside of the hyperplane. We consider two choices of ν: ν close to zero (Outside Anomaly Detector, OAD) and ν close to unity (Inside Anomaly Detector, IAD).

The second model we tested was Deep Support Vector Data Description (Deep SVDD) [21]. SVDD works by finding a minimum volume hypersphere that encloses all or most data points [22]. At test time, points that fall on the outside of the hypersphere would be classified as anomalies. In Deep SVDD, a neural network is employed to transform the data into a new feature space in which to apply SVDD; The parameters of the network and the hypersphere are jointly learned to minimize the volume of the SVDD hypersphere.

We directly apply the Deep SVDD models, with minimal modifications, from Ref. [21] which defines two model objectives. The first is a *Soft-Boundary Deep SVDD* objective and the second is a simpler *One-Class Deep SVDD* objective. Symbolically:

$$\text{Soft-Boundary: } \min_{R,\mathcal{W}} R^2 + \frac{1}{\nu n}\sum_{i=1}^{N} \max\{0, \|\phi(\mathbf{x}_i;\mathcal{W}) - \mathbf{c}\|^2 - R^2\} + \frac{\lambda}{2}\sum_{l=1}^{L}\|W^l\|_F^2 \tag{2}$$

$$\text{One-Class: } \min_{R,\mathcal{W}} \frac{1}{n}\sum_{i=1}^{N}\|\phi(\mathbf{x}_i;\mathcal{W}) - \mathbf{c}\|^2 + \frac{\lambda}{2}\sum_{l=1}^{L}\|W^l\|_F^2, \tag{3}$$

[3]It is also a special case of other proposals, like Energy-based Out of Distribution Detection with $\lambda = 0$ [19].

The third model we tested was a bottleneck autoencoder, which consists of a deep neural network encoder and decoder that are simultaneously trained to reconstruct the data with a mean squared error (MSE) loss. Events are declared anomalous if their MSE (also called reconstruction error) is large.

3 Numerical Results

Training details can be found in Appendix 5.

3.1 Gaussian Example

To begin, we consider a simple Gaussian example in one dimension where the training data (background-only) is normally distributed with mean -2 and unit variance while the anomalous events have mean 0 (or -4) and also unit variance. We start with the logistic regression example discussed in the previous section. The results of this case are presented in Fig. 3.1. As training proceeds, the sigmoid morphs to the right in order to make the average score closer to zero for the background-only data. This results in an effective classifier when the signal happens to be to the right of the background and results in anti-tagging the signal when the signal happens to be on the other side of the background. In both cases, we observe universal behavior, where the performance is nearly constant over the training. Additionally, a similar but reflected behavior holds if either the model or data is reflected across the origin. The results of Fig. 3.1 are qualitatively similar to what we find with a multi-layer logistic regression model.

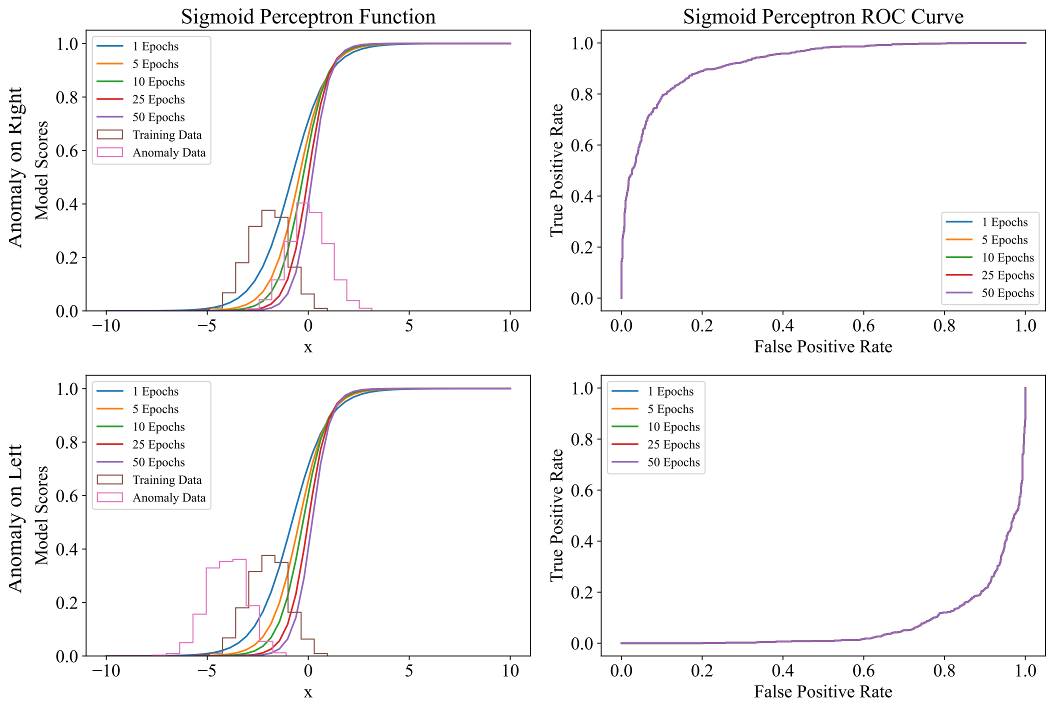

Figure 1: Left: histograms of the background and signal superimposed with the neural network score. The vertical axis corresponds to the network score; there is an arbitrary scaling applied to the histograms so that they fit on the same plot. Right: Receiver Operating Characterstic (ROC) curves. (Note: the ROC curves are completely identical for the different number of epochs, so that is why it appears only the 50 Epoch curve is shown.) Top: the anomaly is on the left of the background. Bottom: the anomaly is on the right of the background.

3.2 Particle Physics Example

For an example from collider physics, we use the LHC 2020 Olympics R&D dataset [23, 24] which includes 10^6 background (Standard Model) events and 10^5 signal events. Our setup closely follows previous resonant anomaly detection studies that use the LHC Olympics dataset [12–14, 25–28]. These events naturally live in a high-dimensional space (each containing hundreds of particles with a momentum), but we consider the same five-dimensional compressed version studied in Ref. [12–14, 27, 28], consisting of masses and angular moments of hadronic jets in the events.

We quantify the performance of various one-class methods in Fig. 3.2. Supervised classifiers are able to achieve a maximum Significance Improvement Characteristic (SIC) of about 17 and state-of-the-art weakly supervised methods are able to reach a maximum SIC of about 13 (see e.g. Ref. [13]). In contrast, we see a range of maximum SIC values between 1 and 2. This may be expected, since unsupervised methods are unaware of the signal properties unlike the weakly supervised approaches. Of the unsupervised methods, the IAD SVM and autoencoder have the best average performance on this benchmark signal model. OCDN performs much worse than these two methods (although it is not the worst). This is only one signal model and it is possible that there exist signal models for which the OCDN would outperform the other methods; this is a challenge for benchmarking unsupervised AD techniques.

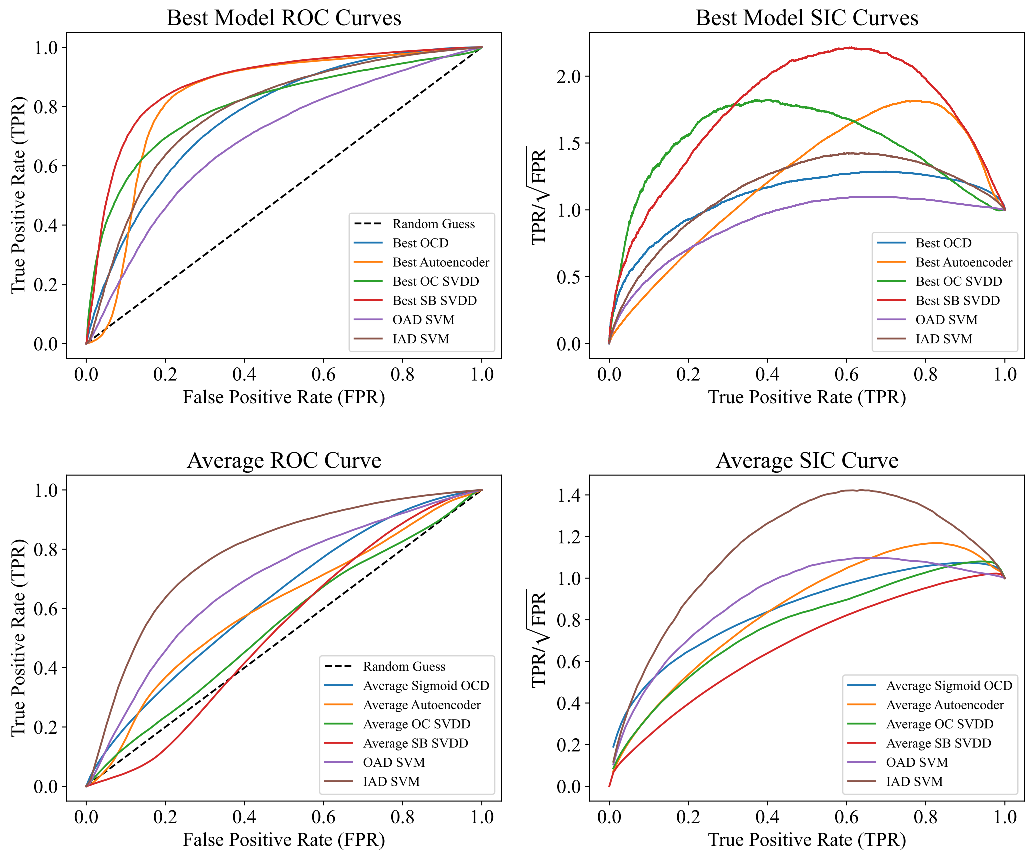

Figure 2: Left: ROC curves for various models. Right: Significance Improvement Characteristic (SIC) curves for various models. The SIC is defined as the true positive rate divided by the square root of the false positive rate. This metric indicates by how much the signal-to-noise (also called *significance*) increases with a given threshold cut specified by the horizontal axis value. Top: the best over 10 random network initializations. Bottom: the average performance over the 10 random network initializations.

4 Conclusions and Outlook

In this paper, we have explored a simple unsupervised method called the One-Class Dense Network. This technique uses a supervised classification setup, but only provides one class during training time. While the loss is minimized for the constant zero function, training for a finite number of epochs may drive the function to zero faster over the support of the background-only data and therefore, the score of the OCDN may be useful for AD. While our physics example does not show a large gain from using the OCDN over other approaches, it provides a complementary tool that may help broaden the sensitivity to new physics using unsupervised methods. Furthermore, the simplicity of this setup may provide a useful platform for analytic studies of AD performance in the future.

References

[1] G. Karagiorgi, G. Kasieczka, S. Kravitz, B. Nachman, and D. Shih, "Machine Learning in the Search for New Fundamental Physics," Dec. 2021. arXiv: 2112.03769 [hep-ph].

[2] ATLAS Collaboration, "Dijet resonance search with weak supervision using 13 TeV pp collisions in the ATLAS detector," 2020. DOI: 10.1103/PhysRevLett.125.131801. arXiv: 2005.02983 [hep-ex].

[3] "Anomaly detection search for new resonances decaying into a Higgs boson and a generic new particle X in hadronic final states using $\sqrt{s}$ = 13 TeV pp collisions with the ATLAS detector," CERN, Geneva, Tech. Rep., 2022, All figures including auxiliary figures are available at https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/CONFNOTES/ATLAS-CONF-2022-045. [Online]. Available: https://cds.cern.ch/record/2816323.

[4] C. Le Lan and L. Dinh, "Perfect density models cannot guarantee anomaly detection," *Entropy*, vol. 23, no. 12, 2021, ISSN: 1099-4300. DOI: 10.3390/e23121690. [Online]. Available: https://www.mdpi.com/1099-4300/23/12/1690.

[5] G. Kasieczka, R. Mastandrea, V. Mikuni, B. Nachman, M. Pettee, and D. Shih, "Anomaly Detection under Coordinate Transformations," Sep. 2022. arXiv: 2209.06225 [hep-ph].

[6] O. Amram and C. M. Suarez, "Tag N' Train: A Technique to Train Improved Classifiers on Unlabeled Data," Feb. 2020. DOI: 10.1007/JHEP01(2021)153. arXiv: 2002.12376 [hep-ph].

[7] J. H. Collins, P. Martın-Ramiro, B. Nachman, and D. Shih, "Comparing Weak- and Unsupervised Methods for Resonant Anomaly Detection," Apr. 2021. arXiv: 2104.02092 [hep-ph].

[8] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," *Proceedings of the 32nd International Conference on Machine Learning*, vol. 37, F. Bach and D. Blei, Eds., pp. 1530–1538, Jul. 2015. [Online]. Available: http://proceedings.mlr.press/v37/rezende15.html.

[9] T. Aarrestad *et al.*, "The Dark Machines Anomaly Score Challenge: Benchmark Data and Model Independent Event Classification for the Large Hadron Collider," May 2021. arXiv: 2105.14027 [hep-ph].

[10] R. Verheyen, "Event Generation and Density Estimation with Surjective Normalizing Flows," May 2022. arXiv: 2205.01697 [hep-ph].

[11] T. Buss *et al.*, "What's Anomalous in LHC Jets?," Feb. 2022. arXiv: 2202.00686 [hep-ph].

[12] B. Nachman and D. Shih, "Anomaly Detection with Density Estimation," *Phys. Rev. D*, vol. 101, p. 075 042, 2020. DOI: 10.1103/PhysRevD.101.075042. arXiv: 2001.04990 [hep-ph].

[13] A. Hallin *et al.*, "Classifying Anomalies THrough Outer Density Estimation (CATHODE)," Sep. 2021. arXiv: 2109.00546 [hep-ph].

[14] J. A. Raine, S. Klein, D. Sengupta, and T. Golling, "CURTAINs for your Sliding Window: Constructing Unobserved Regions by Transforming Adjacent Intervals," Mar. 2022. arXiv: 2203.09470 [hep-ph].

[15] A. Butter *et al.*, "Ephemeral Learning – Augmenting Triggers with Online-Trained Normalizing Flows," Feb. 2022. arXiv: 2202.09375 [hep-ph].

[16] J. Hajer, Y.-Y. Li, T. Liu, and H. Wang, "Novelty Detection Meets Collider Physics," 2018. DOI: 10.1103/PhysRevD.101.076015. arXiv: 1807.10261 [hep-ph].

[17] T. Heimel, G. Kasieczka, T. Plehn, and J. M. Thompson, "QCD or What?" *SciPost Phys.*, vol. 6, no. 3, p. 030, 2019. DOI: 10.21468/SciPostPhys.6.3.030. arXiv: 1808.08979 [hep-ph].

[18] M. Farina, Y. Nakai, and D. Shih, "Searching for New Physics with Deep Autoencoders," 2018. DOI: 10.1103/PhysRevD.101.075021. arXiv: 1808.08992 [hep-ph].

[19] W. Liu, X. Wang, J. D. Owens, and Y. Li, *Energy-based out-of-distribution detection*, 2020. DOI: 10.48550/ARXIV.2010.03759. [Online]. Available: https://arxiv.org/abs/2010.03759.

[20] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. Smola, and R. Williamson, "Estimating support of a high-dimensional distribution," *Neural Computation*, vol. 13, pp. 1443–1471, Jul. 2001. DOI: 10.1162/089976601750264965.

[21] L. Ruff *et al.*, "Deep one-class classification," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, 2018, pp. 4393–4402.

[22] D. Tax and R. Duin, "Support vector data description," *Machine Learning*, vol. 54, pp. 45–66, Jan. 2004. DOI: 10.1023/B:MACH.0000008084.60811.49.

[23] G. Kasieczka, B. Nachman, and D. Shih, *Official Datasets for LHC Olympics 2020 Anomaly Detection Challenge (Version v6) [Data set]*. https://doi.org/10.5281/zenodo.4536624, 2019.

[24] G. Kasieczka *et al.*, "The LHC Olympics 2020: A Community Challenge for Anomaly Detection in High Energy Physics," Jan. 2021. arXiv: 2101.08320 [hep-ph].

[25] J. H. Collins, K. Howe, and B. Nachman, "Anomaly Detection for Resonant New Physics with Machine Learning," *Phys. Rev. Lett.*, vol. 121, no. 24, p. 241 803, 2018. DOI: 10.1103/PhysRevLett.121.241803. arXiv: 1805.02664 [hep-ph].

[26] J. H. Collins, K. Howe, and B. Nachman, "Extending the search for new resonances with machine learning," *Phys. Rev.*, vol. D99, no. 1, p. 014 038, 2019. DOI: 10.1103/PhysRevD.99.014038. arXiv: 1902.02634 [hep-ph].

[27] A. Andreassen, B. Nachman, and D. Shih, "Simulation Assisted Likelihood-free Anomaly Detection," *Phys. Rev. D*, vol. 101, no. 9, p. 095 004, 2020. DOI: 10.1103/PhysRevD.101.095004. arXiv: 2001.05001 [hep-ph].

[28] K. Benkendorfer, L. L. Pottier, and B. Nachman, "Simulation-Assisted Decorrelation for Resonant Anomaly Detection," Sep. 2020. arXiv: 2009.02205 [hep-ph].

[29] Martín Abadi *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980.

[31] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[32] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[33] G. Kasieczka, B. Nachman, and D. Shih, *R&D Dataset for LHC Olympics 2020 Anomaly Detection Challenge*, version v5, Zenodo, Apr. 2019. DOI: 10.5281/zenodo.6466204. [Online]. Available: https://doi.org/10.5281/zenodo.6466204.

5 Potential Broader Impacts

Anomaly detection is used throughout science and industry and so the OCDN may be a useful setup well beyond collider physics.

Appendix

Training

The OCDN and bottleneck autoencoder models were both built in TensorFlow [29] and optimized with Adam [30]. For OCD, our model feeds the data through 3 linear layers each with 128 ReLU activation neurons followed by a singular sigmoid activation output neuron. The autoencoder encodes our data into 2-dimensional space through an encoding block composed of two hidden linear layers of 512 ReLU activation neurons with a batch normalization layer in between. The decoder block also consists of two hidden layers each with 512 ReLU activation neurons but does not contain a batch normalization layer. For the OC-SVM models, we used the OneClassSVM class from Scikit-Learn [31] and for Deep-SVDD, we used the PyTorch [32] framework to build out our model and used optimization methods directly from Ref. [21]. The Deep-SVDD model we employed transforms our data into, and fits the hypersphere, in a 128-dimensional feature space.

All results shown above utilized training data composed of 97,081 background only events and test data composed of 24,270 background and 75,298 signal events. Every network except for the OC-SVM's were trained on an Nvidia Quadro RTX 6000 GPU. The OCDN model trained exceptionally fast; on the order of milliseconds per epoch. The bottleneck autoencoder and Deep-SVDD networks both took around 10-15 seconds per epoch. Lastly, the OC-SVM's were run on cpu and took a few minutes to fit.

Code and Data

The code used for this paper can be found at `https://github.com/normankarr/lhc-oc-classifiers` and the physics data sets are hosted on Zenodo at [33].

Acknowledgements

We thank Aishik Ghosh for useful discussions. BN was supported by the U.S. Department of Energy (DOE), Office of Science under contract DE-AC02- 05CH11231 and DS was supported by DOE grant DOE-SC0010008.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section 5.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

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? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (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]
   (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)? [Yes]
   (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]
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? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (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]