Anomaly Detection with Normalizing Flows for Tabular Data

Denis Kuznedelev [*1] **Gleb Bazhenov** [*1] **Maxim Kurkin** [*1]

Abstract

Due to their ability to estimate probability density of the distribution of interest, normalizing flows are promising candidates for out-of-distribution (OOD) detection. However, as a number of recent works show, these models tend to learn spurious correlations and local structure instead of the genuine semantic structure of the target data. In this report, we study the problem of OOD detection for tabular data, where the underlying structure and patterns are not obvious. For this purpose, we conduct extensive experiments, considering several options of feature preprocessing and flow architectures. As a results, we demonstrate that, given proper model architecture and training procedure, flows can provide descent performance in OOD detection on tabular data. Our findings prove that there are no fundamental limitations for normalizing flows to learn accurate distributions over tabular data, but they have to be tuned in order to serve as reliable anomaly detectors.

1. Introduction

The problem of out-of-distribution (OOD) detection is very important both from theoretical and practical perspective. Solving it, one may design more reliable and sound machine learning systems which are important in high risk applications. Normalizing flows, being simple generative models that can provide exact likelihood, seem to be a natural choice for anomaly detection.

However, there is a strong evidence that these models often assign higher likelihood to the OOD data rather than the data which model was trained on. For instance, Kirichenko et al. (2020) show that normalizing flows tend to focus on the local structure of images rather on the overall semantics, and proposed a special masking approach to prevent a model from capturing local correlations. This enabled them to significantly improve the performance of flow models.

Since our study is focused on tabular data rather than images, it is not clear whether normalizing flows do suffer from this problem when trained on tabular data. In our work, we try to address this question and conduct extensive experiments, considering various options of feature preprocessing and flow architectures. This allows us to understand that there are no fundamental limitations that prevent normalizing flows from learning proper distributions over tabular data. In general, our contributions are as follows:

1. implement 4 normalizing flow layers (Planar, Glow, RealNVP and MAF) and 3 embedding layers (Simple, Periodic and VAE latent representations) that have been recently proposed to improve the performance of neural methods on tabular data;
2. construct an experimental framework that allows to both easily control the parameters of neural architectures or training procedure and extend it with new flow and embedding layers;
3. measure the performance of normalizing flows in OOD detection using the mentioned framework and draw conclusion about their capacity on tabular data based on the conducted experiments.

Further in this work, we provide a brief overview of related works in Section 2. Then, we describe the considered normalizing flows and embeddings for tabular data in Section 3. After that, we formulate the research questions and discuss our experimental results in Section 4. In conclusion, we outline a possible future research in Section 5.

2. Related work

2.1. Methods for anomaly detection

Unsupervised methods. These methods are proposed to follow some particular assumptions about the structure of data. For instance, OOD samples should be located in low-density regions compared to the in-distribution (ID) data. In such case, the AD performance depends on the agreement between the input data and the methods assumptions. Nu-

[*]Equal contribution [1]Skolkovo Institute of Science and Technology, Russia. Correspondence to: Denis Kuznedelev <denis.kuznedelev@skoltech.ru>, Gleb Bazhenov <gleb.bazhenov@skoltech.ru>, Maxim Kurkin <maxim.kurkin@skoltech.ru>.

Project report for the *Bayesian Methods of Machine Learning* course at Skoltech, 2022.

merous unsupervised methods have been proposed in the last few decades (Aggarwal, 2017; Pang et al., 2021; Ruff et al., 2021), which can be roughly grouped into shallow and deep neural methods. The former often provide better interpretability, while the latter can handle high-dimensional data more efficiently. Since no labeling is required for training normalizing flows, these models also belong to unsupervised methods based on neural networks.

Supervised methods. When ground truth labels are available, which is often not the case, supervised classifiers may be used to detect known anomalies at the risk of missing unknown types of OOD samples. The choice of supervised anomaly detection model depends on a particular application, so one may consider general purpose classifiers, such as random forest (Breiman, 2001) and neural networks (LeCun et al., 2015). One of the serious limitations of supervised methods is that ground truth labels are not necessarily sufficient to capture all types of anomalies that may appear on inference.

Semi-supervised methods. These methods are designed to use the supervision from partial labels, while keeping the ability to detect unseen types of anomalies. Some recent studies investigate using partially labeled data for improving anomaly detection performance and leveraging unlabeled data to improve representation learning. For instance, some semi-supervised models (Akcay et al., 2018; Akçay et al., 2019) are trained only on valid ID samples, and detect anomalies that deviate from the learned representations of these samples.

2.2. Benchmarks for anomaly detection

There are many notable works that make an effort to review and benchmark anomaly detection methods on tabular data (Campos et al., 2016; Domingues et al., 2018; Steinbuss & Böhm, 2021). However, for our study, we take the tabular datasets and experimental results from ADBench (Han et al., 2022), since they consider a great variety of different unsupervised anomaly detection methods and provide numerous tabular datasets that represent both real-world and synthetic data sources, and cover various domains such as healthcare, biology, physics and others.

3. Methods

3.1. Normalizing flows

There is a large variety of different normalizing flow architectures in the literature. It is important to make the training and generation process fast and cheap as possible, while, at the same time, having sufficiently expressive model, capable of capturing complicated data distributions. The first requirement restricts the possible choice of architectures to those which can compute the Jacobian of the underlying transformation in a simple way.

One of the first and simplest flows is **Planar** flow (Rezende & Mohamed, 2015):

$$\mathbf{y} = \mathbf{x} + \mathbf{v}\sigma(\mathbf{w}^\top \mathbf{x} + \mathbf{b}). \tag{1}$$

Here $\mathbf{v} \in \mathbb{R}^D, \mathbf{w} \in \mathbb{R}^D$ and $b \in \mathbb{R}$, σ is a *differentiable* activation function. This flow is simple and computationally inexpensive but has limited expressiveness.

RealNVP (Dinh et al., 2016) partitions the feature dimensions into two parts: one of them is modified during the pass and the other remains unchanged. Given a D-dimensional input and $d < D$, the output of a coupling layer has the following form:

$$\begin{aligned} \mathbf{y}_{1:d} &= \mathbf{x}_{1:d}, \\ \mathbf{y}_{d+1:D} &= \mathbf{x}_{d+1:D} \odot \exp(s(\mathbf{x}_{d+1:D})) + t(\mathbf{x}_{d+1:D}), \end{aligned} \tag{2}$$

where s and t are the learnable *scale* and *shift* transformations (usually expressed via MLP with learnable parameters). In order to apply transformation to all variables, one needs to stack multiple flow layers with alternating mask patterns. This flow architecture can be quite expressive and may fit even such complicated data as natural images.

Masked Autoregressive flows, **MAF** (Papamakarios et al., 2017) are the flows from the family of autoregressive models. Each layer of MAF is defined by a following recursion:

$$\begin{aligned} x_i &= u_i \exp(\alpha_i) + \mu_i, \\ \mu_i &= f_{\mu_i}(\mathbf{x}_{1:i-1}), \\ \alpha_i &= f_{\alpha_i}(\mathbf{x}_{1:i-1}), \end{aligned} \tag{3}$$

Functions f_{μ_i}, f_{α_i} are computed using MADE (Germain et al., 2015) (which in turn is based on MLP model) with Gaussian conditionals.

Glow (Kingma & Dhariwal, 2018) is a modification of the aforementioned ReaLNVP designed for generation of high-fidelity images. In addition to an affine coupling layer, it has two new components:

- `ActNorm` layer: $\mathbf{y} = \mathbf{s} \odot \mathbf{x} + \mathbf{b}$,
- `InvertibleLinear` layer: $\mathbf{y} = \mathbf{W}\mathbf{x}$,

where $\mathbf{W}$ is constructed in a way to be manifestly invertible. Thus, the Glow model represents a stack of ActNorm, Invertible linear and Affine coupling layers with alternating masks.

3.2. Processing tabular features

In addition to various normalizing flows, we consider some methods for encoding tabular features. They are expected to

Datasets/Algorithms	COPOD	ECOD	PCA	CBLOF	IForest	Glow		MAF		Planar		RealNVP	
						Features	VAE	Features	VAE	Features	VAE	Features	VAE
annthyroid	76.80	78.03	66.25	62.26	82.01	**86.84**	66.40	66.02	80.78	80.66	86.52	**90.68**	**86.78**
cover	88.64	93.42	**93.73**	89.30	86.74	86.17	92.78	**99.61**	86.20	**97.72**	86.90	89.50	93.32
donors	81.76	74.45	83.15	60.44	77.68	85.61	81.20	65.61	50.63	**86.50**	**88.60**	**97.50**	84.29
http	99.29	98.10	99.72	99.60	**99.96**	**99.99**	99.94	99.35	99.95	99.35	99.95	**100.00**	99.95
magic.gamma	68.33	64.36	67.22	**75.13**	73.25	67.05	68.71	**82.15**	72.62	71.33	73.85	**73.97**	73.02
mammography	**90.69**	**90.75**	88.72	83.74	86.39	62.13	76.73	**90.10**	82.60	89.44	83.95	83.20	83.47
PageBlocks	88.05	90.92	90.64	85.04	89.57	**93.55**	93.24	**93.85**	85.32	93.16	93.07	92.67	**93.98**
pendigits	90.68	91.22	**93.73**	90.40	**94.76**	79.45	73.76	**97.76**	84.03	92.94	83.90	87.98	86.11
shuttle	99.35	99.40	98.62	83.48	99.56	**99.84**	98.96	99.37	99.49	99.25	99.61	**99.85**	**99.68**
skin	47.55	39.09	45.26	69.49	68.21	**79.34**	71.84	75.75	71.86	40.73	63.33	**92.03**	**79.35**
smtp	79.09	71.86	88.41	79.68	89.73	**93.85**	84.17	**93.36**	88.20	35.15	89.59	89.21	**90.41**
thyroid	94.30	97.78	96.34	94.73	**98.30**	97.92	94.80	98.22	96.98	**98.27**	97.82	**98.69**	97.51
vowels	53.15	45.81	65.29	**89.92**	73.94	**84.87**	52.66	**98.50**	70.94	66.95	74.42	31.03	76.45
Wilt	33.40	39.43	20.39	32.54	41.94	56.06	**57.59**	53.19	54.59	49.05	**62.62**	**69.01**	56.23
yeast	36.99	39.61	41.15	44.85	37.76	51.33	47.62	**55.46**	**51.70**	41.76	**53.21**	41.31	51.52

Table 1. AUROC in anomaly detection on 15 benchmark tabular datasets. We mark the **first**, **second** and **third** best performing methods. Here, we take the best performing trainable embedding approach among *simple*, *periodic* or *none* and put them into the Features column, while the results with latent representations are put into the VAE column.

improve the performance of flow models by providing more meaningful geometry of samples in feature space.

In particular, we follow (Gorishniy et al., 2022) and implement *simple* and *periodic* encodings. For each feature j with corresponding value x_j, the *simple* enconding can be defined as some trainable direction $\mathbf{v}_j \in \mathbb{R}^k$, such that feature value x_j serves as the scaling coefficient along that direction:

$$\mathbf{z}_j = \texttt{simple}(x_j) = x_j \mathbf{v}_j.$$

After that, the obtained embeddings of distinct features are concatenated into one input vector. The *periodic* encoding slightly modifies this approach:

$$\texttt{periodic}(x_j) = \texttt{concat}\Big[\sin(2\pi\mathbf{z}_j), \cos(2\pi\mathbf{z}_j)\Big].$$

Moreover, we consider the latent representations from variational autoencoder (Rezende & Mohamed, 2015) trained to reconstruct the input features of the original tabular data. The final embeddings of samples are composed of the predicted mean values of variational distribution.

4. Experiments

In this work, we evaluate several standard normalizing flow architectures and try different types of tabular feature embeddings on various tabular datasets. We are focused on the following research questions:

Q1. Is processing of input features useful for anomaly detection on tabular data using normalizing flow models?

It is known (Gorishniy et al., 2022) that specific way of preprocessing numerical and categorical features may greatly boost the performance of deep learning models on tabular data in regression and classification tasks. One of the questions in our experiments is whether some embedding techniques can improve the performance on OOD detection.

For this purpose, we consider RealNVP normalizing flow on 5 representative tabular datasets, where the flow model manages to achieve high performance, and compare 3 different methods to process tabular features — *none*, *simple* and *periodic*. Our results in Table 2 show that learnable embeddings for tabular features often help to increase the anomaly detection performance.

Q2. How important are the hyperparameters of normalizing flow architectures for high anomaly detection performance on tabular data?

Another important topic that can have significant impact on the result is a proper recipe for training normalizing flows on data since at the experimental evidence below shows the results are quite sensitive to the particular choice of hyperparameters. Since flow models are invertable by construction, their input and output sizes must be the same. However, one can consider flow layers of different depth and width. A more detailed discussion on the hyperparameter optimisation is presented in Appendix B.

The optimal number of training steps is not straightforward to determine as well. Unlike the case of standard supervised learning, training curves are not very insightful, since the loss is not bounded from below, and usually both training loss and validation loss monotonously decrease. We found that one should train a normalizing flow model for a sufficient number of steps, but not too long — after some time, OOD performance begins to deteriorate. Sometimes, loss may take large negative values and training becomes unstable. A few examples of the training curves are presented on Figure 4 in Appendix C.

Q3. Are there fundamental limitations that prevent normalizing flows from learning proper distributions over tabular data in order to detect anomalies?

Our experimental results in Table 1 show that, by careful tuning of hyperparameters and choice of appropriate embeddings for the input features, one can achieve superior results in anomaly detection on the considered tabular datasets. In particular, flow models provide the first best result in terms of AUROC metric on all but one tabular datasets. This proves that the only challenge in making normalizing flows strong anomaly detectors is a proper choice of neural flow architecture and embeddings for tabular features.

5. Conclusion

Contrary to images and text, tabular data does not have apparent structure. Thus, there is no reason to assume that normalizing flows should fail in anomaly detection on tabular data. In fact, our experiments show that, given proper model architecture and training procedure, flows can provide descent results in OOD detection. However, we found that the performance of normalizing flows may vary significantly across various sets of hyperparameters and random initializations. Despite being capable of accurately capturing the target distribution and providing the best performance in anomaly detection, normalizing flows are not sufficiently robust, and certain effort is required to make them reliable in the task of interest.

References

Aggarwal, C. C. An introduction to outlier analysis. In *Outlier analysis*, pp. 1–34. Springer, 2017.

Akcay, S., Atapour-Abarghouei, A., and Breckon, T. P. Ganomaly: Semi-supervised anomaly detection via adversarial training. In *Asian conference on computer vision*, pp. 622–637. Springer, 2018.

Akçay, S., Atapour-Abarghouei, A., and Breckon, T. P. Skip-ganomaly: Skip connected and adversarially trained encoder-decoder anomaly detection. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2019.

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

Breiman, L. Random forests. *Machine learning*, 45(1): 5–32, 2001.

Campos, G. O., Zimek, A., Sander, J., Campello, R. J., Micenková, B., Schubert, E., Assent, I., and Houle, M. E. On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. *Data mining and knowledge discovery*, 30(4):891–927, 2016.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Domingues, R., Filippone, M., Michiardi, P., and Zouaoui, J. A comparative evaluation of outlier detection algorithms: Experiments and analyses. *Pattern recognition*, 74:406–421, 2018.

Germain, M., Gregor, K., Murray, I., and Larochelle, H. Made: Masked autoencoder for distribution estimation. In *International conference on machine learning*, pp. 881–889. PMLR, 2015.

Gorishniy, Y., Rubachev, I., and Babenko, A. On embeddings for numerical features in tabular deep learning. *arXiv preprint arXiv:2203.05556*, 2022.

Han, S., Hu, X., Huang, H., Jiang, M., and Zhao, Y. Adbench: Anomaly detection benchmark. In *Neural Information Processing Systems (NeurIPS)*, 2022.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

Kirichenko, P., Izmailov, P., and Wilson, A. G. Why normalizing flows fail to detect out-of-distribution data. *Advances in neural information processing systems*, 33: 20578–20589, 2020.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature*, 521(7553):436–444, 2015.

Pang, G., Shen, C., Cao, L., and Hengel, A. V. D. Deep learning for anomaly detection: A review. *ACM Computing Surveys (CSUR)*, 54(2):1–38, 2021.

Papamakarios, G., Pavlakou, T., and Murray, I. Masked autoregressive flow for density estimation. *Advances in neural information processing systems*, 30, 2017.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *International conference on machine learning*, pp. 1530–1538. PMLR, 2015.

Ruff, L., Kauffmann, J. R., Vandermeulen, R. A., Montavon, G., Samek, W., Kloft, M., Dietterich, T. G., and Müller, K.-R. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 109(5):756–795, 2021.

Steinbuss, G. and Böhm, K. Benchmarking unsupervised outlier detection with realistic synthetic data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15 (4):1–20, 2021.

A. Embeddings for tabular features

In Table 2, one can find comparison of various encodings for tabular features in terms of AUROC metric in anomaly detection performance, measures on 5 representative datasets using RealNVP flow model with the same hyperparameters in each experiment.

Datasets/Embeddings	None	Periodic	Simple	VAE
annthyroid	84.13	78.84	**90.68**	86.78
donors	**97.50**	87.92	90.20	84.29
http	99.36	70.57	**100.00**	99.95
shuttle	99.37	83.97	**99.85**	99.68
skin	66.53	78.94	**92.03**	79.35

Table 2. AUROC in anomaly detection on 5 benchmark tabular datasets with different embedding types for RealNVP flow. We mark the **first** best performing method. One can see that *simple* or *periodic* trainable embeddings often help to improve the anomaly detection performance. At the same time, VAE latent representations also significantly increase the AUROC metric in some cases.

B. Hyperparameter optimisation

Since the choice of hyperparameters can influence result dramatically an efficient procedure for their selection. We have adopted Optuna framework (Akiba et al., 2019) for hyperparameter selection employing Bayesian methods for efficient exploration. We have tuned following parameters:

- Number of flow layers (4 to 24);
- Hidden dimension of MLP layers in flows;
- Activation functions (ReLU, GELU, Tanh);
- Batch size;
- Learning rate;
- Number of training steps;
- Weight decay.

As one can see from Figures 1, 2 and 3, the most important parameters are usually learning rate and batch size. The optimal depth is usually 8 to 12 flow layers, i.e it is not optimal to use to shallow or deep models. Importantly, there is a large variance between different runs, so the OOD detection performance might be a matter of luck rather then the architecture.

C. Training dynamics

The loss curves unlike the common supervised training setup are hard to interpret since the loss is not bounded from below and usually monotonously decreases both on the train and test set. Sometimes the test loss during the flow training jumps to very high values and then decreases to moderate numbers. A possible explanation is that, in our experiments, we adopt the normalization of features. Both the ID and OOD data was normalized according to the statistics on the ID (i.e $\mathbf{x} \to (\mathbf{x} - \boldsymbol{\mu})/\boldsymbol{\sigma}$) with $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ being the mean and standard deviation on the ID data respectively.

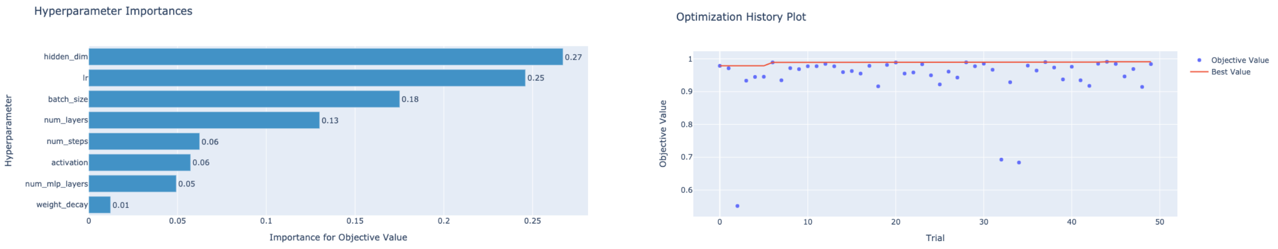
We have observed that on some datasets samples from OOD data may have values of very large magnitude in some dimensions (i.e some features can take very different values on the ID and OOD data). This fact can explain the emergence of large OOD loss values.

MAF turned out to be quite unstable during the training with a high probability of producing NaNs and infinities while transforming the input distribution. In order to make the training process more stable, we applied clipping of the flow output and NaN / inf removal. After this procedure, MAF can be trained normally, and in some cases this flow architecture appeared to yield the best results.

D. Visualization of distributions

In order to better understand what is going under the hood and whether normalizing flows do really separate the ID and OOD data either by assigning points to different clusters or OOD data to the regions with low estimated probability density, we plot two dimensional PCA projection of the input features (before feeding to the normalizing flow) and output features (after processing by normalizing flow layers). We have chosen one simple dataset (with high AUROC score) Shuttle and one hard (with high AUROC score) Yeast for visualization.

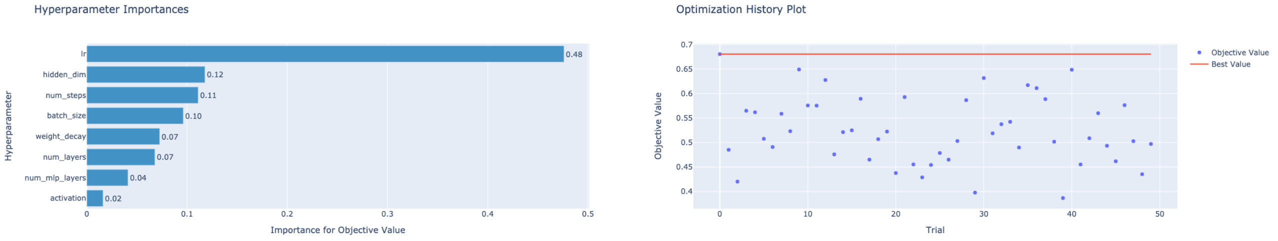
In Figures 5, one can observe that the ID points after traversing through the flow form a cluster of complicated form, whereas OOD data is scattered, but overlapping with this cluster. Regarding Figure 6, both distributions look indistinguishable in the original domain, and the difference is hardly discernible after traversal through the flow.



(a) Hyperparam importances

(b) Optimisation history

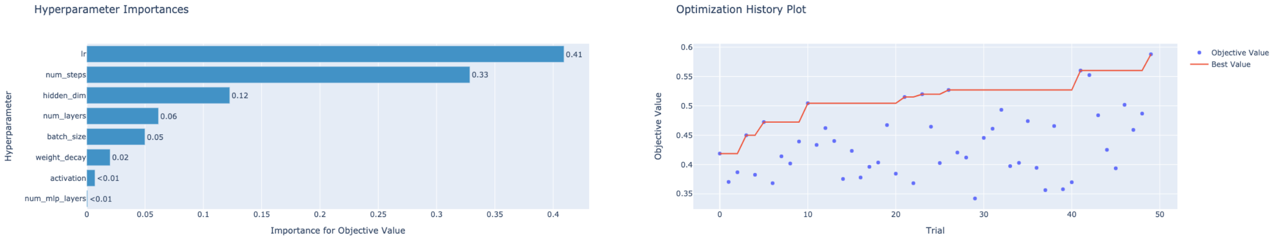
Figure 1. Hyperparameter optimisation on the Thyroid dataset.



(a) Hyperparam importances

(b) Optimisation history

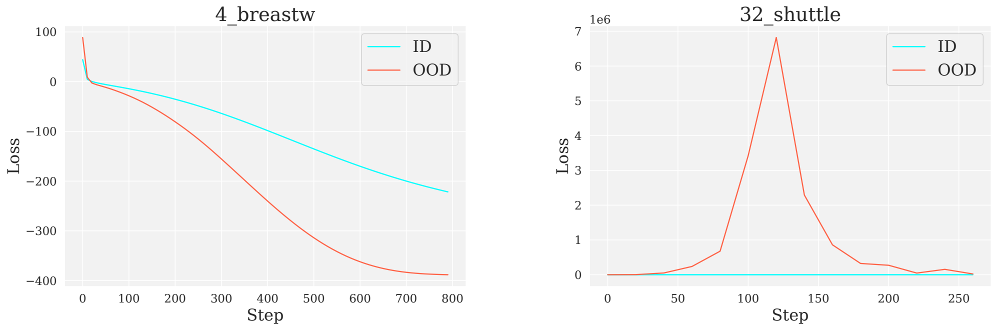
Figure 2. Hyperparameter optimisation on the Wilt dataset.



(a) Hyperparam importances

(b) Optimisation history

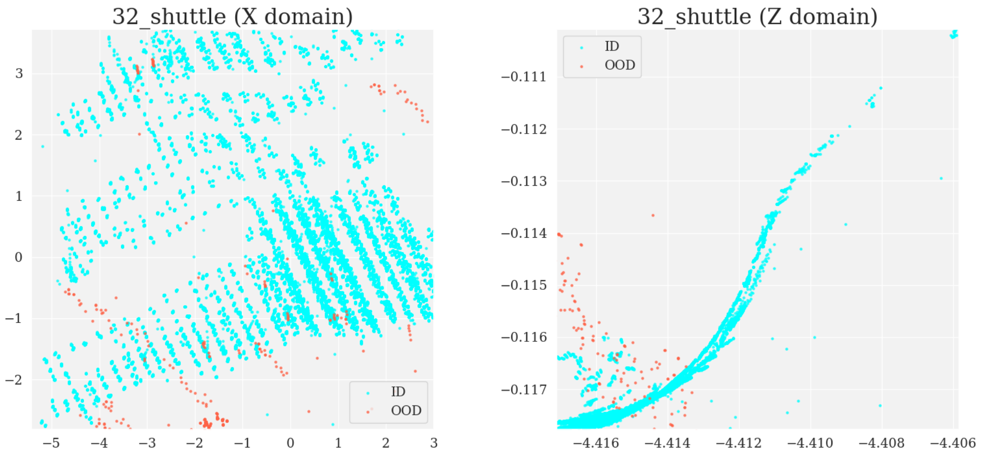
Figure 3. Hyperparameter optimisation on the Yeast dataset.



(a) Dynamics of ID and OOD loss on the Breast dataset.

(b) Optimisation history

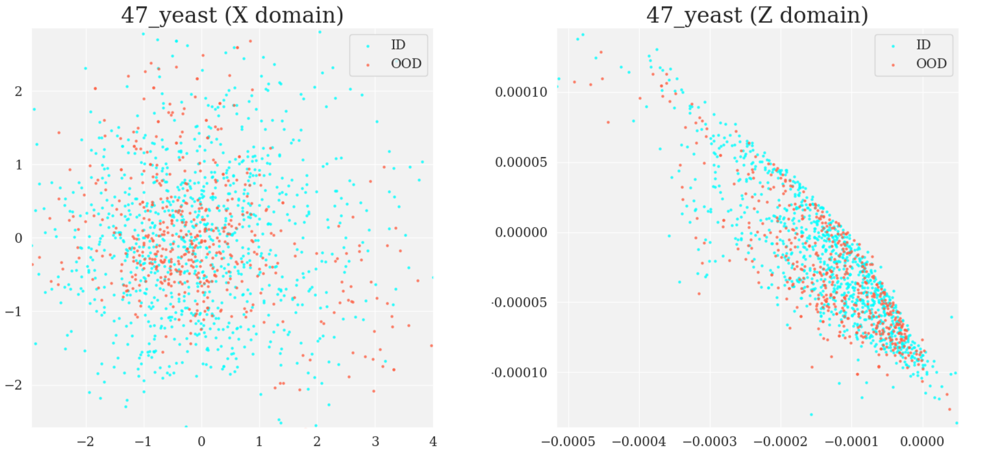
Figure 4. Dynamics of ID and OOD loss on the Shuttle dataset.



(a) original domain

(b) transformed domain

Figure 5. PCA projection of the Shuttle dataset.



(a) original domain

(b) transformed domain

Figure 6. PCA projection of the Yeast dataset.