Time series and anomaly detection
Master Digital Economics – Madalina OLTEANU

Literature Review

Kathleen ROGAN

&

César DENOST

Random Partitioning Forest for Point-Wise and Collective Anomaly Detection - Application to Intrusion Detection, Pierre-François Marteau (2021)

1. Description of the objective of the article

Anomaly detection is an important problematic in many fields, including cybersecurity, finance, medicine and industry. It involves identifying behaviours or patterns that deviate from ”normality”, thus helping to prevent major incidents such as intrusions into IT systems, financial fraud, or industrial equipment failures.

These anomalies can be considered as individual or collective, and have a wide range of characteristics, making them particularly complex to identify. Traditional anomaly detection algorithms have strengths and weaknesses depending on the application context. There are tree types of most used models in anomaly detection.

The Isolation Forest (IF) model, is widely used for its speed and ability to handle large dimensions, but suffers from limitations. In particular, it fails to detect collective anomalies and presents ”blind” zones, which are poorly-covered regions of the data space where anomalies can be misclassified. Auto-encoders, such as KitNET and Variational Auto-Encoder (VAE), while powerful for reconstructing normal data, are often resource-intensive and are more prone to overfitting. Classical supervised approaches, such as one-class SVMs face scalability issues with high dimensional data, they also require a fine hyperparameter tuning and might struggle more than other models with imbalanced datasets.

These limitations highlight the need for a new algorithm that combines the ability to detect various types of anomalies with high computational efficiency.

The Divergence Feature Representation Random Forest (DiFF-RF) algorithm, submitted in 2021 by Pierre-François Marteau, aims to improve anomaly detection performances while remaining computationally efficient. DiFF-RF follows in the footsteps of random forests, with key improvements that enable it to overcome the limitations of existing algorithms, notably those of Isolation Forest (IF). The main objective is to develop a method capable of efficiently detecting both point and collective anomalies while maintaining robustness and scalability adapted to large datasets. The aim of the paper was to propose an innovative semi-supervised algorithm based on random partitioning trees, incorporating additional criteria for better anomaly characterization.
The different models, as well as the DiFF-RF, are evaluated on synthetic data and sets from the UCI database, and cybersecurity-specific benchmarks datasets.

2. Methodologies

The main model of this paper is the DiFF-RF, but as its results are tested against other methodologies we briefly introduce them before introducing DiFF-RF in more detail.

2.1 Isolation forest (IF)

Isolation Forest (IF) was introduced by Liu et al; and was based on a probabilistic approach with random partitioning trees for anomaly detection. Anomalies are isolated from normal points using their remoteness and rarity.

Isolation trees are constructed recursively, using a partitioning of a random sample drawn from the data with n as the sample size. At each stage a random selection of a q dimension from the n dimensions is operated, as well as a random selection of a threshold τ between the minimum and maximum values of q in the current data. And a division of the data into two subsets based on threshold τ. This process continues until a single point or maximum depth $\lceil \log_2(n) \rceil$ is reached, guaranteeing trees of logarithmic depth.

The IF algorithm calculates an anomaly score for each point based on the path lengths and rank all the points.

The path length or average isolation depth, $h(x)$ is the average of the lengths of the paths leading to a leaf through all trees until an external node is reached:

$$h(x) = \frac{1}{t} \sum_{i=1}^{t} h_i(x) \tag{1}$$

One issue with this method for anomaly scoring is that the Isolation tree grows in order of n but the average height growth order is log n. Instead the average path length of unsuccessful search in a BTS can be used as Isolation Trees have the same structure.

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n} \tag{2}$$

The anomaly score of a point is thus :

$$s(x) = 2^{-\frac{h(x)}{c(n)}} \tag{3}$$

with $0 < s \leq 1$.

IF is very efficient as it has a time complexity of $\mathcal{O}(t \cdot n \log n)$ for the construction and $\mathcal{O}(t \log n)$ for evaluation. The fast processing time enables IF to be used for high dimensional datasets. However, it also has some limits, notably the blind-spot problem and a limited capacity to detect collective or contextual anomalies. These limitations have led to extensions such as DiFF-RF and EIF.[Liu et al.(2008)]

2.2 Extended Isolation Forest (EIF)

Extended Isolation Forest (EIF) is an extension of IF model. It addresses the two main drawbacks of IF, the "blind spot" effect, and the inability to model intricate correlations among features due to a bias related to the branching of the trees.

EIF employs hyperplanes in multiple dimensions to partition, thereby enhancing its ability to model more complicated data structures.The branch cuts have a random slope and a random intercept.
The selection of the random slope is done by defined by a normalized vector in

$$\mathbb{R}^n$$

where n is the dimensionality of the dataset. This approach enables EIF to account for correlations among features, allowing partitions that align more closely with the underlying data distribution. Consequently, EIF reduces the "blind spot" that IF is prone to—regions of the feature space that are sparsely populated or unoccupied yet erroneously interpreted as normal.

The anomaly scoring mechanism in EIF is the same as in IF. The final anomaly score $s(x)$ reflects the probability that a given instance is an outlier.

In terms of computational efficiency, EIF maintains the same time complexity as IF, requiring $\mathcal{O}(t \cdot n \log n)$ for training and $\mathcal{O}(t \log n)$ for testing, where t is the number of trees and n is the sample size. Although multidimensional splits are introduced, the computational overhead is minimal, making EIF both scalable and efficient for larger datasets. [Hariri et al.(2018)]

2.3 Support Vector Machines (SVM)

Support Vector Machines (SVM), first introduced by Cortes and Vapnik, is a powerful machine learning approach designed to find a hyperplane that best separates data points into distinct categories. In the paper a one-class SVM is used. It is designed to distinguish anomalies by learning a decision function that encapsulates the boundary of normal data in a given feature space.
The one-class SVM builds this boundary using the following optimization problem:

$\min_{w,\rho,\xi} \|w\|^2 + \frac{1}{\nu n} \sum_{i=1}^{n} \xi_i - \rho(1)$
subject to:

$$w \cdot \phi(x_i) \geq \rho - \xi_i, \quad \xi_i \geq 0, \quad \forall i \tag{2}$$

With w a weight vector defining the hyperplane, ρ the offset parameter, $\phi(x)$ the mapping function projecting data into a higher-dimensional feature space, ξ_i the slack variables to allow soft margins, and ν the parameter controlling the trade-off between boundary width and the fraction of anomalies.

The decision function for 1C-SVM is given by:

$$f(x) = w \cdot \phi(x) - \rho \tag{3}$$

A point is classified as normal if $f(x) > 0$ and anomalous if $f(x) \leq 0$.

A key strength of SVM lies in its ability to use kernel functions $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$, enabling mapping into high-dimensional spaces in non-linear space.

Training complexity for 1C-SVM ranges from $\mathcal{O}(n^2)$ to $\mathcal{O}(n^3)$, with n the number of samples, making it less scalable for large datasets. Additionally, its performance is sensitive to hyperparameter choices (ν, kernel parameters), which require careful cross-validation for optimal performance. However, 1C-SVM remains robust and effective, particularly in high-dimensional spaces with moderate dataset sizes. [Schölkopf et al.(1999)]

2.4 Variational Autoencoders (VAE)

Variational Autoencoders (VAE) can be seen as probabilistic generative model or variational Bayesian models that use neural networks. They were introduced in the seminal work "Auto-Encoding Variational Bayes" by Kingma and Welling (2022). VAE has an encoder, $q_\phi(z|x)$, that takes high dimensional inputs, x, and maps it to a low dimensional distribution, with a mean vector μ and a standard deviation vector σ, in a latent space z . These parameters define a multivariate Gaussian distribution $z \sim \mathcal{N}(\mu, \sigma^2)$.

It also has a decoder $p_\theta(x|z)$, that maps the latent space z to an input space. The reconstruction is probabilistic in nature. Thus, the VAE is able to model complex distributions of data very effectively.
VAE uses gradient-based optimization, to update the weights of the network the evidence lower bound (ELBO) is maximized. This allows the model to balance its reconstruction and latent space regularization.

$$argmax L(\theta, \phi; x) \tag{1}$$

$$L(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{\mathrm{KL}}(q_\phi(z|x) \| p(z)) \tag{2}$$

It is the difference between the measures of how well the reconstructed data matches the input and the distance loss.
For anomaly detection, VAE identify anomalies through failure to reconstruct inputs that greatly differ from the training distribution.
The reconstruction error, defined as:

$$\text{Reconstruction Error} = \|x - x'\|^2 \tag{3}$$

serves as the primary metric for detecting anomalies.
Higher reconstruction errors indicate a higher likelihood of the input being anomalous. Additionally, the likelihood of the latent representation z under the prior $p(z)$ provides a secondary anomaly signal, as anomalous inputs are often mapped to low-probability regions of the latent space. The computational complexity of VAE with a network with L layers and d parameters, for training is $\mathcal{O}(n \cdot d)$, where n is the number of training samples. Inference complexity is linear with respect to the number of layers and samples, making VAEs computationally efficient for moderate-sized datasets. Despite a high complexity, VAE offer a powerful and flexible framework for anomaly detection, particularly in high-dimensional and complex data distributions.[Kingma and Welling(2022)]

2.5 KitNET

KitNET is an unsupervised anomaly detection framework designed for real-time network intrusion detection based on an ensemble of neural networks. It uses an ensemble of lightweight auto-encoders to handle high-dimensional data efficiently, while a hierarchical approach ensures scalability and modularity. KitNET operates through two main components: the Feature Mapper (FM) that monitor the patterns of the network traffic, and the Anomaly Detector (AD) that detects patterns that are anomalous.

During the *train-mode*, KitNET learns correlations among input features and clusters them into subsets. It serves as inputs to the auto-encoders in the subsequent ensemble layer. FM minimizes inter-feature correlations within each subset, and maps instances x to

$$\mathcal{V} = \{v_1, v_2, \ldots, v_k\},$$

with v_i the set of feature indices assigned to the i-th auto-encoder, and k the total number of subsets.

The FM optimizes feature grouping using a correlation-based clustering algorithm. The correlation matrix $\mathbf{C}$ is incrementally updated with each input instance x, and a hierarchical clustering algorithm is applied to form the feature subsets.

The Anomaly Detector (AD) consists of two layers. Firstly, an Ensemble Layer where each subset v_i is processed by an independent auto-encoder $\mathcal{A}_i$, which reconstructs the input features and computes the reconstruction error:

$$S_i = \|x_{v_i} - \mathcal{A}_i(x_{v_i})\|^2 \tag{1}$$

x_{v_i}: The features in subset v_i, $\mathcal{A}_i(x_{v_i})$: The reconstructed output.
And secondly, an Output Layer, which consists of the reconstruction errors from the ensemble layer are aggregated by a final auto-encoder $\mathcal{A}_{\text{output}}$, which computes the global anomaly score:

$$\text{Anomaly Score} = \|S - \mathcal{A}_{\text{output}}(S)\|^2 \tag{2}$$

$S = [S_1, S_2, \ldots, S_k]$: The vector of reconstruction errors from the ensemble layer.

Once trained, KitNET evaluates the incoming data for anomalies based on the global anomaly score. High scores indicate deviations from normal behavior.

The computational complexity of KitNET is linear with respect to the number of auto-encoders k and the size of each feature subset m. The training complexity per instance is $\mathcal{O}(n)$ and the execution complexity per instance is $\mathcal{O}(k \cdot m)$.

This efficiently enables KitNET to process data streams in real-time, making it particularly suitable for dynamic environments with high-dimensional feature spaces.[Mirsky et al.(2018)]

2.6 DIFF-RF

The DiFF-RF model was inspired by the IF algorithm, as it is also a forest of binary partitioning trees. The main ambitions of DiFF-RF are to overcome the weaknesses of the IF model, mainly the blind spot problem. It does that by introducing a criterion based on expected distances to the centroid and relative frequencies of visits to tree leaves. It also offers a scalable and efficient solution, as DiFF-RF maintains the simplicity and scalability of random forests, while offering the ability to adapt to complex and important datasets.

The introduction of a distance based criterion, through the measures of the closeness between a unique observation and a cluster associated with the leaves improves the ability of DiFF-RF to classify if a single data point is normal or not. The model also enables one to classify a collection of points by taking into account the relative frequency of visits to a leaf. Two meta-parameters are used to build the trees. The first one is the size of the subsets and the second the number of trees t. The maximum height of the trees is set at $\log_2 \psi$.

The DiFF-RF forest is obtained by randomly selecting t samples in $\{S_1, \ldots, S_t\}$, with $|S_i|$ the size of subsets.

$$|Si| = \psi \tag{1}$$

The way the cuts in the algorithm differs from the IF algorithm, as it uses the empirical probability distribution D to select dimensions with low or medium entropy and those with high entropy are considered as noise. For each subset, the entropy H_q is calculated for each dimension q.

For the anomaly score, we consider two main cases, if it is a point-wise anomaly or collective anomalies.

For point-wise anomaly detection, with a given tree T, a data point x in leaf e, in subset S, the anomaly score is calculated using a Gaussian model that gives the weighted distance based on the aggregation of the difference between the data point and the centroid divided by the standard deviation of the training instances.
For T the anomaly score is calculated as:

$$\delta_T(x) = 2^{-\alpha \cdot \delta(M_S, \sigma_S, x)} \tag{2}$$

The point-wise anomaly score is:

$$pwas(x) = -\mathbb{E}(\delta_T(x)) \tag{3}$$

where α is the hyper-parameter used to scale the distance. The range of $pwas$ is $-1 < pwas < 0$.

For collective anomaly detection, which is when a subset of the data is abnormal in comparison to the training set, with abnormal co-occurrences, DiFF-RF considers the visiting frequencies in the leaf nodes to construct the collective anomaly score.
The frequency of visit of a specific leaf during training is evaluated using $f_n = \frac{|S|}{\psi}$ which is the ratio between the number of instances attached to this leaf and the number of training instances. And during testing, $f_X = \frac{|S_x|}{|X|}$ is used, where S_x is the subset of X in the leaf.

Thus, we can compare the relative train/test frequencies for each leaf with $v_T(X) = \frac{f_n}{f_X}$. The collective anomaly score is the aggregation of the distance score and the relative frequency score.

$$c_T(x, X) = \delta_T(x) \cdot v_T(X) \tag{4}$$

And for X the collective anomaly score is:

$$cas(x, X) = -\mathbb{E}(c_T(x, X)) \tag{5}$$

When collective anomalies are overlapping with normal data points this scoring is not effective.

DiFF-RF has the same complexity as the IF algorithm, with time complexities of $\mathcal{O}(t \cdot (\psi) \cdot log(\psi))$ during training and $\mathcal{O}(n \cdot t \cdot \log(\psi))$ during testing.

The goal of DiFF-RF is to solve the blind spot effect in the Isolation Forest and both point-wise and collective DiFF-RF are able to address these blind spots, with collective DiFF-RF performing better.

Moreover, collective DiFF-RF is effective at detecting concept drift which happens when there is a change in the joint distribution of input variables X and target y. It can also detect variance increases or decreases, which can help detect attacks that happen over long periods.

DiFF-RF is also efficient at detecting different types of intrusion. For poisoning attacks, where the training data is modified by the attacker, if a reference distribution for the normal data is known, DiFF-RF can identify drift. For evasion attacks, where a malicious pattern that is known is modified utile it is very difficult to detect it, DiFF-RF's design to handle blind spots enables it to detect these patterns. And finally, for flooding attacks with data points that are on the decision boundary thus increasing the rate of false positive, DiFF-RF's anomaly detection results are not impacted, demonstrating the robustness of the approach. [Marteau(2021)]

3. Performances of the Models

3.1 Synthetic Experiment

In the paper, a synthetic experiment was conducted to showcase the difference in results between the IF model and the DiFF-RF. The normal data is a 2D-torus centered in (0,0). Training and testing sets are uniformly drawn with each having n= 1000 instances. Two anomaly sets are drawn from a normal distribution and are represented in red and green in Figure 1, where the 2D-torus dataset displays clusters of anomalous data in specific regions. We can see that when using the IF model, we get the following results when detecting the anomalies.

In Figure 2, we can see that the anomalies closer to the normal data are being miss-detected as normal data with the blind-spot clearly visible as the green anomalies are not being discriminated against, and only the red anomalies are being detected.

In Figure 3, graph (b), we see the point-wise anomaly DiFF-RF is able to discriminate between the red and green anomalies, but has difficulty with the green ones.

When only using the ratio of visiting frequencies we can see in graph (c) that the anomaly clusters are more likely to be in leave of training at the limit of the torus. When using the collective anomaly DiFF-RF we can see that we achieve the best results, that separates the two anomalies from the normal data.

3.2 Experimentation and analysis of the results

Pierre-François Marteau carried out a study using 13 different datasets from the UCI repository that followed different criteria. They needed to be suitable for binary classification, with numerical data, and have distinct fields of application which enables to test the different models on wide range of problematic. Seven semi-supervised models have been evaluated: a SVM, a VAE, KitNET, IF, EIF, and DiFF-RF, both point-wise and collective anomaly detections. For IF and SVM the implementation of sklearn are used, for VAE the keras tensorflow implementation is used. A custom EIF was implemented.

The models were then evaluated on the 13 different datasets in the UCI repository. It comprised datasets on banknote authentication , cardiography , default of credit card clients, high time resolution universe survey, gamma telescope signals, particle identification, molecule conformation, occupency detection, sensorless drive diagnosis, spam detection, steel plates faults, and TV news commercial. Four benchmark datasets are used to evaluate the different models on intrusion detection and one toy dataset in 5 dimensions.

For the evaluation of the models the Area under the ROC curve (AUC) was used. It represents how well a model distinguishes between classes, with the best model closer to

1.[Narkhede(2021)]

The Average Precision (AP) is also used to access the performance of the model. It is a summary of precision-recall curve into a scalar that represents the average of all precisions. [Anwar(2022)]

On average, the Diff-RF model is the best ranked method for both of its versions. There are no significant differences between the DiFF-RF (PW) and DiFF-RF (C), DiFF-RF(PW) and KitNET also perform similarly, KitNET,VAE,IF and EIF have close results for most of the datasets and 1C-SVM is the only model that has significant difference from the other models. The rest is considered significant for the other groups. SVM has the worst results, it mainly performs poorly because it does not scale well, has a long computing time, and is difficult to select good parameters to optimise. EIF is worse than IF on average due to the blind spot problem still being there. PW DiFF-RF outperforms both IF and EIF as well as VAE except for some specific cases. On average DiFF-RF point-wise is better than KiNET except for three benchmarks sets, and other sets. For collective anomaly DiFF-RF outperforms all the other approaches, except for particularly difficult detection anomaly tasks. And alpha seems to be tuned accurately for DiFF-RF to perform well.

The two best ranked methods DiFF-RF and KitNet are complementary, and most of the time their performance is not significantly different even though they are very different in structure as Kitnet is a ensemble of auto-encoders. DiFF-RF does not provide satisfactory detection in some datasets. This is due to the complexity of the datasets and should be further analyzed to get some insights about the miss-detection.

A way to improve the DiFF-RF model would be to combine it with auto-encoders. This could lead to a more robust detector, that compensate their weaknesses. and would be more resilient to drift for example.

4. Reproduction of study

We decided to reproduce the study done on the 12 datasets from the UCI's machine learning repository only. The other datasets were used as benchmarks for intrusion detection. Moreover, the ISCX dataset is not available anymore due to a piracy risk when trying to download it.

As we saw in part 3.2, the datasets are from various domains ranging from industry to television commercials. Some of them required further cleaning steps and preprocessing to be used properly as they could not be used in models directly. All the datasets that needed those steps are available in the GitHub repository. [Rogan(2024)]

It is also important to note that the dataset on Steel Plates Data Set (SPF) is constituted of only faulty entries from 7 fault categories. We decided to treat each fault separately as a target and compute the mean of each of the AUC and AP scores. The TV News commercial detection dataset is constituted of four separate datasets, we used two of them, TVCD-BBC and TVCD-CNN.

For the models we used the isolation forest IF algorithm [Pedregosa et al.(2011a)], and one-class SVM classifier 1C-SVM [Pedregosa et al.(2011b)] implementations from SKLearn, a deep variational auto-encoder VAE implementation for tensorflow with Keras [Kingma and Welling(2022)], an ensemble of auto-encoders KitNET available on github [**?**], and the DiFF-RF with two modes from the paper [Marteau(2021)]. We were not able to use the extended isolation forest due to a problem between the C++ compiler and the construction of wheels in Python.

For base code to calculate the AUC the testDiFF_RF_Donuts.py [Marteau(2021)] was used as an example and modified to work with other datasets and models.

We can compare the results we obtain in the following tables to the ones in Tables I and II in the paper. All the code and results are available on the GitHub repository. [Rogan(2024)]

Table 1: AUC AND AP VALUES OBTAINED BY IF, KITNET, AND VAE ON THE 13 DATASETS

Datasets	IF		KitNET		VAE	
	AUC	AP	AUC	AP	AUC	AP
BNA	0.941	0.98	0.64	0.856	0.873	0.961
CTG	0.835	0.87	0.889	0.924	0.843	0.881
DCCC	0.613	0.668	0.53	0.595	0.632	0.672
HTRU2	0.951	0.939	0.921	0.92	0.943	0.933
MAGIC	0.759	0.897	0.713	0.891	0.69	0.869
MiniBooNE	0.75	0.834	0.734	0.803	0.778	0.855
MUSK	0.433	0.42	0.267	0.343	0.333	0.364
Occupancy	0.994	0.995	0.983	0.985	0.993	0.993
SDD	0.995	0.996	1.0	1.0	0.967	0.965
SPAM	0.809	0.919	0.725	0.88	0.797	0.923
TVCD-BBC	0.585	0.817	0.622	0.844	0.592	0.836
TVCD-CNN	0.585	0.817	0.622	0.844	0.597	0.837
SPF	0.495	0.832	0.537	0.843	0.484	0.827
Mean Rank	0.75	0.845	0.706	0.825	0.733	0.84

In Table 1 we can see that the results obtained for the IF algorithm are quite similar to the ones obtained in the paper. Our model performs slightly better on average with a mean AUC score of 0.75 compared to an average of 0.733 for the paper. The same can be said for KitNET with an average AUC of 0.71 compared to 0.75, with the model underperforming on some datasets like DCCC or MUSK. And for VAE our model had on average better results with an average AUC of 0.733 compared to 0.713. For those three models no important discrepancies were observed.

Table 2: AUC AND AP VALUES OBTAINED BY 1C-SVM, DIFF-RF (PW), AND DIFF-RF (CL) ON THE 13 DATASETS

Datasets	1C-SVM		DIFF-RF(PW)		DIFF-RF(CL)	
	AUC	AP	AUC	AP	AUC	AP
BNA	0.976	0.993	1.0	1.0	1.0	1.0
CTG	0.83	0.868	0.83	0.872	0.902	0.924
DCCC	0.643	0.679	0.65	0.709	0.735	0.795
HTRU2	0.948	0.939	0.953	0.942	0.963	0.956

Continued on next page

Datasets	1C-SVM		DIFF-RF(PW)		DIFF-RF(CL)	
	AUC	AP	AUC	AP	AUC	AP
MAGIC	0.742	0.895	0.834	0.933	0.881	0.951
MiniBooNE	0.766	0.847	0.751	0.835	0.943	0.966
MUSK	0.205	0.325	0.722	0.73	0.793	0.81
Occupancy	0.997	0.997	0.999	0.999	0.999	0.999
SDD	1.0	1.0	1.0	1.0	1.0	1.0
SPAM	0.807	0.928	0.884	0.963	0.957	0.986
TVCD-BBC	0.6	0.839	0.617	0.847	0.66	0.867
TVCD-CNN	0.6	0.839	0.618	0.848	0.662	0.868
SPF	0.522	0.839	0.401	0.784	0.378	0.772
Mean Rank	0.741	0.845	0.789	0.882	0.836	0.915

In Table 2, we can see that both point-wise DiFF-RF and collective DiFF-RF had results that were similar to the ones obtained in the main experimentation. Our point-wise model obtained on average an AUC of 0.79 compared to 0.78, and our collective model obtained an average AUC of 0.84 compared to 0.83. The main difference we could observe in terms of results was with the 1C-SVM model. Our model outperformed the results of the paper's model, with an average AUC of 0.74 compared to 0.64 which is quite significant. This could be explained by a recent update of the source code that changed unnecessary Cython import and some general upgrades solving errors.

Overall, our best-performing model is also the collective DiFF-RF and then the point-wise version. However, the worst performing algorithm is KitNET which is surprising as in the paper its results are better than the IF, VAE and SVM models.

Moreover, we wanted to see if combining the DiFF-RF model with auto-encoders, using the Ensemble Learning method called ”stacking” would improve its performance. With this method, the outputs of the first model are used as the inputs for the second. In this case, we used the output of the encoder from the VAE model as the input of the DiFF-RF model. The features learned by the VAE are used in the binary classification of the DiFF-RF. In Table 3 of the Appendix, we can observe that the results obtained by the VAE-DiFF-RF of both the point-wise and collective version had less accurate results.

Appendix

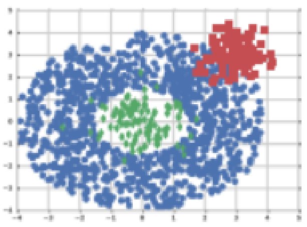

Figure 1: 2D-torus 'normal' data set in blue round dots, with a cluster of anomaly data in red square dots at the top right side of the torus, with an additional cluster of anomaly data in green diamond dots at the center of the to

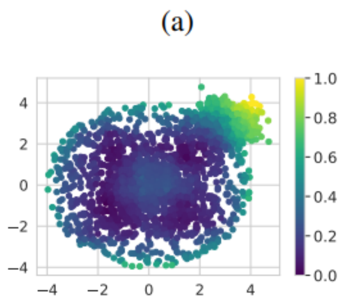

Figure 2: 2D-torus heat map corresponding to the IF

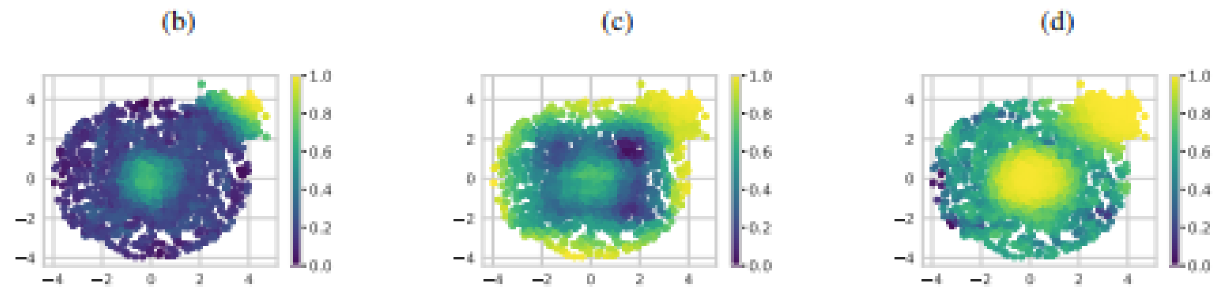

Figure 3: 2D-torus heat map corresponding to the point-wise anomaly score of the DiFF-RF (b), to the expectation of the ratio of visiting frequencies in the DiFF-RF (c), to the collective anomaly score of the DiFF-RF (d)

Table 3: AUC AND AP VALUES OBTAINED BY AE-DIFF-RF (PW), AND AE-DIFF-RF (CL) ON THE 13 DATASETS

Datasets	VAE-DIFF-RF(PW)		VAE-DIFF-RF(CL)	
	AUC	AP	AUC	AP
BNA	0.405	0.78	0.997	0.999
CTG	0.746	0.803	0.935	0.953
DCCC	0.558	0.62	0.666	0.736
HTRU2	0.936	0.904	0.962	0.951
MAGIC	0.696	0.866	0.787	0.909
MiniBooNE	0.731	0.82	0.895	0.94
MUSK	0.614	0.615	0.746	0.743
Occupancy	0.969	0.974	0.996	0.996
SDD	0.991	0.995	1.0	1.0
SPAM	0.765	0.917	0.842	0.946
TVCD-BBC	0.651	0.866	0.668	0.879
TVCD-CNN	0.664	0.868	0.7	0.899
SPF	0.468	0.816	0.315	0.755
Mean Rank	0.707	0.834	0.808	0.9

References

[Anwar(2022)] Aqeel Anwar. 2022. What is average precision in object detection localization algorithms and how to calculate it? https://towardsdatascience.com/what-is-average-precision-in-object-detection-localization-algorithms-and-how-to-

[Hariri et al.(2018)] Sahand Hariri, Matias Carrasco Kind, and Robert J. Brunner. 2018. Extended Isolation Forest. *CoRR* abs/1811.02141 (2018). arXiv:1811.02141 http://arxiv.org/abs/1811.02141

[Kingma and Welling(2022)] Diederik P Kingma and Max Welling. 2022. Auto-Encoding Variational Bayes. arXiv:1312.6114 [stat.ML] https://arxiv.org/abs/1312.6114

[Liu et al.(2008)] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation Forest. In *2008 Eighth IEEE International Conference on Data Mining*. 413–422. https://doi.org/10.1109/ICDM.2008.17

[Marteau(2021)] Pierre-François Marteau. 2021. Random Partitioning Forest for Point-Wise and Collective Anomaly Detection - Application to Network Intrusion Detection. *IEEE Transactions on Information Forensics and Security* (Jan. 2021), 1–16. https://doi.org/10.1109/TIFS.2021.3050605

[Mirsky et al.(2018)] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. 2018. Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. *(NDSS'18* (2018).

[Narkhede(2021)] Sarang Narkhede. 2021. Understanding AUC - roc curve. https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

[Pedregosa et al.(2011a)] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011a. , 2825–2830 pages. https://scikit-learn.org/1.5/modules/generated/sklearn.ensemble.IsolationForest.html

[Pedregosa et al.(2011b)] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011b. , 2825–2830 pages. https://scikit-learn.org/1.5/modules/generated/sklearn.svm.SVC.html

[Rogan(2024)] Kathleen Rogan. 2024. Tiny-boot/ts project. https://github.com/Tiny-boot/TS_project

[Schölkopf et al.(1999)] Bernhard Schölkopf, Robert Williamson, Alex Smola, John Shawe-Taylor, and John Platt. 1999. Support Vector Method for Novelty Detection. *NIPS* 12 (01 1999), 582–588.