1. Datasets

Here we describe the datasets we use for benchmarks. See Table S1 for a summary of this information.

The **annthyroid** dataset is derived from the "Thyroid Disease" dataset from the UCIMLR. The original data has 7200 instances with 15 categorical attributes and 6 real-valued attributes. The class labels are "normal", "hypothyroid", and "subnormal". For anomaly detection, the "hypothyroid" and "subnormal" classes are combined into 534 outlier instances, and only the 6 real-valued attributes are used.

The **arrhythmia** dataset is derived from the "Arrhythmia" dataset from the UCIMLR. The original dataset contains 452 instances with 279 attributes. There are five categorical attributes which are discarded, leaving this as a 274-dimensional dataset. The instances are divided into 16 classes. The eight smallest classes collectively contain 66 instances and are combined into the outlier class.

The **breastw** dataset is also derived from the "Breast Cancer Wisconsin (Original)" dataset. This is a 9-dimensional dataset containing 683 instances of which 239 represent malignant tumors and are treated as the outlier class.

The **cardio** dataset is derived from the "Cardiotocography" dataset. The dataset is composed of measurements of fetal heart rate and uterine contraction features on cardiotocograms. The are each labeled "normal", "suspect", and "pathologic" by expert obstetricians. For anomaly detection, the "normal" class forms the inliers, the "suspect" class is discarded, and the "pathologic" class is downsampled to 176 instances forming the outliers. This leaves us with 1831 instances with 21 attributes in the dataset.

The **cover** dataset is derived from the "Covertype" dataset. The original dataset contains 581,012 instances with 54 attributes. The dataset is used to predict the type of forest cover solely from cartographic variables. The instances are labeled into seven different classes. For outlier detection, we use only the 10 quantitative attributes as the features. We treat class 2 (lodgepole pine) as the inliers, and class 4 (cottonwood/willow) as the outliers. The remaining classes are discarded. This leaves us with a 10-dimensional dataset with 286,048 instances of which 2,747 are outliers.

The **glass** dataset is derived from the "Glass Identification" dataset. The study of classification of types of glass was motivated by criminological investigations where glass fragments left at crime scenes were used as evidence. This dataset contains 214 instances with 9 attributes. While there are several different types of glass in this dataset, class 6 is a clear minority with only 9 instances and, as such, points in class 6 are treated as the outliers while all other classes are treated as inliers.

The **http** dataset is derived from the original "KDD Cup 1999" dataset. It contains 41 attributes (34 continuous and 7 categorical) which are reduced to 4 attributes (service, duration, src_bytes, dst_bytes). Only the "service" attribute is categorical, dividing the data into {http, smtp, ftp, ftp_data, others} subsets. Here, only the "http" data is used. The values of the continuous attributes are centered around 0, so they have been log-transformed far away from 0. The original data contains 3,925,651 attacks in 4,898,431 records. This smaller dataset is created with only 2,211 attacks in 567,479 records.

The **ionosphere** dataset is derived from the "Ionosphere" dataset. It consists of 351 instances with 34 attributes. One of the attributes is always 0 and, so, is discarded, leaving us with a 33-dimensional dataset. The data come from radar measurements of the ionosphere from a system located in Goose Bay, Labrador. The data are classified into "good" if the radar returns evidence of some type of structure in the ionosphere, and "bad" otherwise. The "good" class serves as the inliers and the "bad" class serves as the outliers.

The **lympho** dataset is derived from the "Lymphography" dataset. The data contain 148 instances with 18 attributes. The instances are labeled "normal find", "metastases", "malign lymph", and "fibrosis". The two minority classes only contain a total of six instances, and are combined to form the outliers. The remaining 142 instances form the inliers.

The **mammography** dataset is derived from the original "Mammography" dataset provided by Aleksandar Lazarevic. Its goal is to use x-ray images of human breasts to find calcified tissue as an early sign of breast cancer. As such, the "calcification" class is considered as the outlier class while the "non-calcification" class is the inliers. We have 11,183 instances with 6 attributes, of which 260 are "calcifications."

The **mnist** dataset is derived from the classic "MNIST" dataset of handwritten digits. Digit-zero is considered the inlier class while 700 images of digit-six are the outliers. Furthermore, 100 pixels are randomly selected as features from the original 784 pixels.

The **musk** dataset is derived from its namesake in the UCIMLR. It is created from molecules that have been classified by experts as "musk" or "non-musk". The data are downsampled to 3,062 instances with 166 attributes. The "musk" class forms the outliers while the "non-musk" class forms the inliers.

The **optdigits** dataset is derived from the "Optical Recognition of Handwritten Digits" dataset. Digits 1–9 form the inliers while 150 samples of digit-zero form the outliers. This gives us a dataset of 5,216 instances with 64 attributes.

The **pendigits** dataset is derived from the "Pen-Based Recognition of Handwritten Digits" dataset from the UCI Machine Learning Repository. The original collection of

handwritten samples is reduced to 6,870 points, of which 156 are outliers.

The **pima** dataset is derived from the "Pima Indians Diabetes" dataset. The original dataset presents a binary classification problem to detect diabetes. This subset was restricted to female patients at least 21 years old of Pima Indian heritage.

The **satellite** dataset is derived from the "Statlog (Landsat Satellite)" dataset. The smallest three classes (2, 4, and 5) are combined to form the outlier class while the other classes are combined to form the inlier class. The train and test subsets are combined to produce a of 6,435 instances with 36 attributes.

The **satimage-2** dataset is also derived from the "Satlog (Landsat Satellite)" dataset. Class 2 is downsampled to 71 instances that are treated as outliers, while all other classes are combined to form an inlier class. This gives us 5,803 instances with 36 attributes.

The **shuttle** dataset is derived from the "Statlog (Shuttle)" dataset. There are seven classes in the original dataset. Here, class 4 is discarded, class 1 is treated as the inliers and the remaining classes, which are comparatively small, are combined into an outlier class. This gives us 49,097 instances with 9 attributes, of which 3,511 are outliers.

The **smtp** is also derived from the "KDD Cup 1999" dataset. It is pre-processed in the same way as the **http** dataset, except that the "smtp" service subset is used. This version of the dataset only contains 95,156 instances with 3 attributes, of which 30 instances are outliers.

The **thyroid** dataset is also derived from the "Thyroid Disease" dataset. The attribute selection is the same as for the **annthyroid** dataset but only the 3,772 training instances are used in this version. The "hyperfunction" class, containing 93 instances, is treated as the outlier class, while the other two classes are combined to form an inlier class.

The **vertebral** dataset is derived from the "Vertebral Column" dataset. 6 attributes are derived to represent the shape and orientation of the pelvis and lumbar spine. These attributes are: pelvic incidence, pelvic tilt, lumbar lordosis angle, sacral slope, pelvic radius and grade of spondylolisthesis. Each instance comes from a different patient. The "Abnormal (AB)" class of 210 instances are used as inliers while the "Normal (NO)" class is downsampled to 30 instances to be used as outliers.

The **vowels** dataset is derived from the "Japanese Vowels" dataset. The UCIMLR presents this data as a multivariate time series of nine speakers uttering two Japanese vowels. For outlier detection, each frame of each time-series is treated as a separate point. There are 12 features associated with each time series, and these translate as the attributes for each point. Data from speaker 1, downsampled to 50 points, form the outlier class. Speakers 6, 7, and 8 form the inlier class. The rest of the points are discarded. This leaves is with 1,456 points in 12 dimensions, of which 50 are outliers.

The **wbc** dataset is derived from the "Wisconsin-Breast Cancer (Diagnostics)" dataset. The dataset records measurements for breast cancer cases. The benign class is treated as the inlier class, while the malignant class is downsampled to 21 points and serves as the outlier class. This leaves us with 278 points in 30 dimensions.

The **wine** dataset is a collection of results of a chemical analysis of several wines from a region in Italy. The data contain 129 samples having 13 attributes, and divided into 3 classes. Classes 2 and 3 form the inliers while class 1, downsampled to 10 instances, is the outlier class.

2. Complexity of CHAODA

Here we provide short proofs for the time complexity and space complexity of the CHAODA algorithms. For each algorithm, we have a dataset X with $n = |X|$ points and a graph $G(V, E)$ of clusters/vertices V and edges E between overlapping clusters.

2.1. CLAM Clustering

We use CLAM to build the cluster-tree and the induced graphs. The time complexity of clustering is the same as for clustering in CHESS (Ishaq et al., 2019); i.e., expected $\mathcal{O}(nlogn)$ and worst-case $\mathcal{O}(n^2)$ where n is the size of the dataset.

The cost for inducing graphs depends on whether it is a layer-graph or an optimal graph. For both types of graphs, we first have to select the right clusters, and then find neighbors based on cluster overlap.

We implemented CLAM in Python and the language does not have tail-call optimization for recursive functions. Therefore we implement partition to, instead of recursing until reaching leaves, iteratively increase the depth of the tree. During the course of this partition, we store a map from tree-depth to a set of clusters at that depth. Therefore, selecting all cluster at a given depth costs $\mathcal{O}(1)$ time and $\mathcal{O}(|V|)$ space where V is the set of selected clusters. Selecting clusters for optimal-graphs is more expensive. First, we use a trained meta-ml model to predict the AUC contribution from each cluster in a tree; this costs $\mathcal{O}(n)$ time and $\mathcal{O}(n)$ space. Next, we sort the clusters by this predicted value; this costs $\mathcal{O}(nlogn)$ time and $\mathcal{O}(n)$ space. Finally, we perform a linear pass over the clusters to select the best for the graph, while discarding the ancestors and descendants of any cluster that has already been selected; this costs $\mathcal{O}(n)$ time and

Table S1. Datasets used for Benchmarks

Dataset	Cardinality	# Dim.	# Outliers	% Outliers
annthyroid	7,200	6	534	7.42
arrhythmia	452	274	66	15
breastw	683	9	239	35
cardio	1,831	21	176	9.6
cover	286,048	10	2,747	0.9
glass	214	9	9	4.2
http	567,479	4	2,211	0.4
ionosphere	351	33	126	36
lympho	148	18	6	4.1
mammography	11,183	6	260	2.32
mnist	7603	100	700	9.2
musk	3,062	166	97	3.2
optdigits	5,216	64	150	3
pendigits	6,870	16	156	2.27
pima	768	8	268	35
satellite	6,435	36	2036	32
satimage-2	5,803	36	71	1.2
shuttle	59,097	9	3,511	7
smtp	95,156	3	30	0.03
thyroid	3,772	6	93	2.5
vertebral	240	6	30	12.5
vowels	1,456	12	50	3.4
wbc	278	30	21	5.6
wine	129	13	10	7.7

$\mathcal{O}(|V|)$ space. Therefore, the total cost of selecting clusters for optimal graphs is $\mathcal{O}(nlogn)$ time and $\mathcal{O}(n)$ space.

Once the clusters have been selected for a graph, we have to find every pair of clusters with overlapping volumes. Naïvely, this can be done with an all-pairs distance computation for a cost of $\mathcal{O}(|V|^2)$ for time and space. However, our implementation is superior to the naïve method although the proof is beyond the scope of this supplement.

2.2. Relative Cluster Cardinality

This algorithm performs a single linear pass over the vertices in the graph. The cardinalities of clusters are cached during the tree-building phase of clam. Each lookup from this cache costs $\mathcal{O}(1)$. For a graph $G(V, E)$ the time-complexity is trivially $\mathcal{O}(|V|)$. Since each cluster stores its cardinality, the space complexity is also $\mathcal{O}(|V|)$.

2.3. Relative Component Cardinality

This method first finds the components of the graph. This costs $\mathcal{O}(|E|)$ time because we have to check each edge once. The cardinality of each component is cached when traversing the clusters to find components, thus the space complexity is $\mathcal{O}(|C|)$ where C is the set of distinct connected components in the graph. With this done, the algorithm performs a single linear pass over each component. This brings the total worst-case cost to $\mathcal{O}(|E| + |V|)$.

2.4. Graph Neighborhood

This algorithm performs a linear pass over the clusters in the graph and first computes the eccentricity of each cluster. Finding the eccentricity of a vertex in a graph is worst-case $\mathcal{O}(|E|)$ time when the graph consists of a single component. This brings the total cost up to $\mathcal{O}(|E| \cdot |V|)$, with space complexity $\mathcal{O}(|V| + |E|)$. Next, the algorithm per-

forms a traversal from each cluster. This adds a constant factor of $\mathcal{O}(|E|)$ to the time complexity and $\mathcal{O}(|V|)$ to the space complexity, which can be ignored. The total time-complexity of this algorithm is thus $\mathcal{O}(|E| \cdot |V|)$ and its space complexity is $\mathcal{O}(|V|)$ (because only the size of each graph-neighborhood needs to be stored).

2.5. Child-Parent Cardinality Ratio

While building the tree with CLAM, we cache the child-parent cardinality ratios of every cluster, because it proved useful for purposes other than anomaly detection. This method performs a single linear pass over the clusters in the graph and looks-up the cached child-parent ratios as needed. The time-complexity is thus $\mathcal{O}(|V|)$. Since the ratios are cached with their respective clusters, the space complexity is $\mathcal{O}(|V|)$.

2.6. Stationary Probabilities

This method starts by computing a transition matrix for each component in the graph. We set the transition probability from a cluster to a neighbor to be inversely proportional to the distance between their centers, normalized by all possible neighbors of the cluster. We successively square this matrix until it converges. The transition matrices from our graphs obey the criteria required for convergence as proven in (Levin & Peres, 2017). Matrix multiplication for square matrices costs $\mathcal{O}(|V|^{2.373})$ with the Coppersmith-Winograd algorithm (Coppersmith & Winograd, 1990). Thus the worst-case time complexity is the same as that for the matrix-multiplication algorithm employed. For space, we need only store a single $|V| \times |V|$ matrix, giving us a space complexity of $\mathcal{O}(|V|^2)$.

In practice, $|V| \ll n$ and graphs only rarely consist of only one component. Thus, the average run-time performance is much better than that suggested by the quadratic space time-complexity.

2.7. Vertex Degree

Since we already have a graph with vertices and edges, calculating the degree of each vertex only costs $\mathcal{O}(1)$ time. Thus, the complexity of this algorithm is $\mathcal{O}(|V|)$.

2.8. Normalization

Normalizing the outlier scores requires finding the mean and standard deviation of the raw scores, followed by a linear pass over the set of scores. Thus the time-complexity of this step is $\mathcal{O}(n)$. Since we need to store a score for each point, the space complexity is $\mathcal{O}(n)$.

The algorithm is presented in Algorithm 1.

Algorithm 1 Gaussian Normalization

Require: X, a dataset
Require: S, a set of outlier scores for each point in X
$erf : x \mapsto \frac{2}{\sqrt{\pi}} \int_0^x e^{-u^2}\, du$
$\mu \leftarrow mean(S)$
$\sigma \leftarrow std(S)$
for point $p \in X$ **do**
    $S[p] \leftarrow \frac{1}{2}\left(1 + erf\left(\frac{S[p]-\mu}{\sigma \cdot \sqrt{2}}\right)\right)$
end for

2.9. Ensemble

Given the normalized scores from the individual methods, we combine the scores by voting among them in an ensemble. There is a small, constant, number of scores for each point; each score is from a different graph built using the meta-ml models. We simply take the mean of all scores for each point. Thus the time-complexity of voting among these scores is $\mathcal{O}(n)$ for the entire dataset. Since we need to store a score for each point, the space complexity is $\mathcal{O}(n)$.

3. UMAP Visualization

A visualization in Figure S1 using UMAP illustrates a handful of different examples; the anomalies in the Cardio and OptDigits datasets, where CHAODA outperforms other methods, appear to be at the edges of a complex manifold (though, clearly, the UMAP projection has distorted the manifold). In the Mnist dataset, where several methods perform fairly well, the distribution is less interesting. Most anomalies are off to one side but there are several interspersed among the inliers.

In Figure S2, we show UMAP visualizations of the Pima dataset. The inliers and outliers seem inseparable, and so all the methods perform poorly.

4. Performance

4.1. Run-time performance on Test set of Datasets

Tables S2 and S3 report the running time, in seconds, of CHAODA and each competitor. The fastest methods on each dataset are presented in bold face.

4.2. AUC and Runtime performance on Train set of Datasets

Tables S4 and S5 report the the AUC performance and running time, respectively, of CHAODA and each competitor on the train set of datasets.

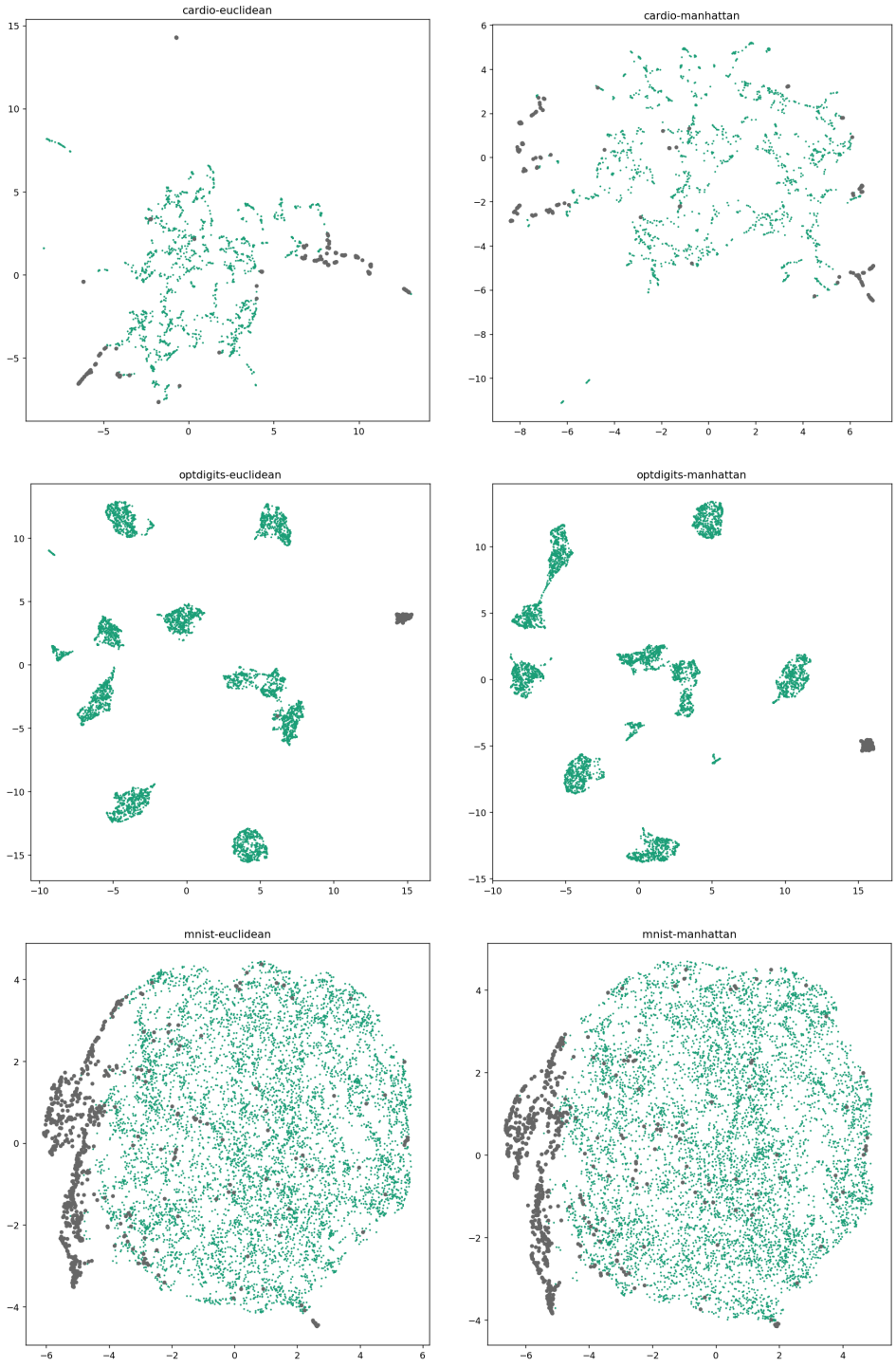

Figure S1. UMAP projections of Cardio (first row), Optdigits (second row) and Mnist (third row). The distance metrics used are Euclidean (left column) and Manhattan (right column). Anomalies are in gray. Note that for MNIST, the UMAP projection does not find much structure, though most of the anomalies congregate to one side. For Cardio, there is a single main component to the manifold, and anomalies tend to be at the edges of that manifold. For OptDigits, there are several distinct pieces to the manifold, perhaps corresponding to different digits. Most algorithms perform comparably on MNIST, while CHAODA outperforms others on Cardio and OptDigits.

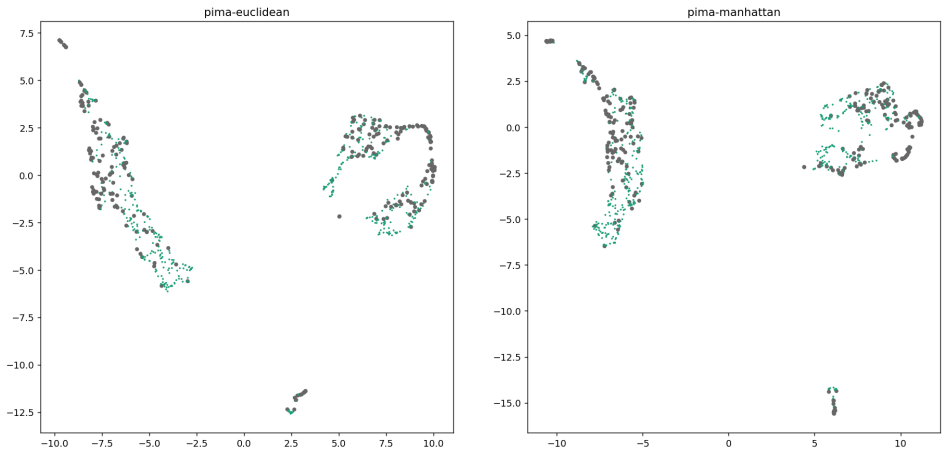

Figure S2. UMAP projections of the Pima dataset. All algorithms performed poorly on Pima. This may be because the anomalies and the outliers seem inseparable in the projection.

Table S2. Time taken, in seconds, on the first half of the Test Datasets

Model	Arrhy	BreastW	Cardio	Cover	Glass	Http	Iono	Lympho	Mammo
CHAODA-fast	8.06	5.01	42.37	829.62	1.73	5e3	3.71	1.78	29.92
CHAODA	27.43	14.00	344.27	6e3	7.44	2e4	22.13	1.74	244.81
ABOD	0.34	0.20	0.72	24.02	**0.07**	19.08	0.13	**0.05**	3.82
AutoEncoder	8.00	6.18	9.05	183.70	3.99	154.20	4.99	3.79	35.26
CBLOF	0.16	0.13	0.17	*EX*	**0.06**	*EX*	**0.07**	**0.05**	0.20
COF	1.24	1.84	12.63	2e4	0.24	2e5	0.60	0.14	513.46
HBOS	**0.08**	**0.00**	**0.01**	1.13	**0.00**	**0.01**	**0.01**	**0.01**	**0.01**
IFOREST	0.43	0.34	0.43	4.61	0.30	3.95	0.33	0.30	0.95
KNN	0.19	**0.07**	0.30	11.46	**0.02**	7.47	**0.05**	**0.02**	1.58
LMDD	14.85	2.62	22.12	1e4	0.57	4e3	1.73	0.42	243.01
LOCI	307.04	*TO*	*TO*	*TO*	25.68	*TO*	120.92	9.78	*TO*
LODA	**0.04**	**0.03**	**0.05**	0.81	**0.03**	0.66	**0.03**	**0.03**	0.17
LOF	0.16	**0.01**	0.18	10.24	**0.00**	1.93	**0.01**	**0.00**	0.59
MCD	5.08	0.64	0.89	28.65	**0.05**	8.21	0.15	**0.05**	2.11
MOGAAL	46.46	42.09	116.73	*TO*	40.67	*TO*	40.86	37.84	*TO*
OCSVM	**0.10**	**0.02**	0.23	257.95	**0.00**	257.28	**0.01**	**0.00**	6.37
SOD	1.19	1.89	13.39	*TO*	0.31	*TO*	0.76	0.20	521.25
SOGAAL	6.70	5.03	14.43	591.14	3.95	597.17	4.71	3.96	92.27
SOS	0.69	47.40	4.11	*TO*	0.18	*TO*	0.33	0.11	*TO*
VAE	10.00	7.75	10.89	223.09	5.27	175.59	6.66	5.21	40.87

4.3. Performance of Individual CHAODA Algorithms

The ensemble of CHAODA algorithms is discussed extensively in the main paper, but we did not have room to discuss or present the performance of the individual algorithms. Due to the large numbers of graphs generated for the ensemble and with each method being applied to each graph, we cannot provide these intermediate results as a table in this document. We instead provide a .csv file which will be available for download.

References

Coppersmith, D. and Winograd, S. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251 – 280, 1990. ISSN 0747-7171. Computational algebraic complexity editorial.

Ishaq, N., Student, G., and Daniels, N. M. Clustered hierarchical entropy-scaling search of astronomical and biological data. In *2019 IEEE International Conference on Big Data (Big Data)*, pp. 780–789. IEEE, 2019.

Table S3. Time taken, in seconds, on the second half of the Test Datasets

Model	Musk	OptDigits	Pima	SatImg-2	Smtp	Vert	Vowels	WBC	Wine
CHAODA-fast	40.22	75.81	8.16	58.63	156.97	2.59	29.96	4.71	0.89
CHAODA	601.12	8e3	24.50	4e3	413.86	2.32	214.64	7.21	1.08
ABOD	4.39	6.81	0.26	3.20	18.00	**0.08**	0.51	0.14	**0.04**
AutoEncoder	23.40	24.29	4.65	23.73	195.61	2.85	6.98	4.70	3.31
CBLOF	0.27	0.44	**0.08**	0.31	0.65	**0.06**	**0.10**	**0.08**	**0.05**
COF	45.60	119.57	2.40	141.84	2e4	0.28	8.06	0.68	0.11
HBOS	**0.07**	**0.04**	**0.00**	**0.02**	**0.01**	**0.00**	**0.01**	**0.01**	**0.00**
IFOREST	0.93	0.92	0.35	0.79	3.90	0.30	0.40	0.33	0.29
KNN	3.50	5.53	**0.09**	1.84	6.83	**0.03**	0.18	**0.05**	**0.01**
LMDD	375.11	421.22	2.86	302.94	4e3	0.59	9.47	1.91	0.33
LOCI	*TO*	*TO*	*TO*	*TO*	*TO*	35.97	*TO*	154.49	6.84
LODA	**0.07**	**0.10**	**0.04**	**0.10**	0.73	**0.03**	**0.05**	**0.04**	**0.03**
LOF	3.59	5.49	**0.02**	1.50	1.36	**0.00**	**0.06**	**0.01**	**0.00**
MCD	84.74	7.24	0.69	6.83	13.78	**0.05**	0.81	0.11	**0.05**
MOGAAL	267.93	415.02	40.89	463.18	*TO*	39.86	80.35	40.65	38.10
OCSVM	2.51	3.71	**0.03**	3.14	252.52	**0.00**	0.11	**0.01**	**0.00**
SOD	56.94	131.49	2.48	210.39	*TO*	0.54	13.92	1.03	0.19
SOGAAL	29.55	44.72	5.20	48.79	592.27	4.69	10.38	4.76	4.01
SOS	9.37	26.09	1.01	34.96	*TO*	0.21	2.81	0.36	**0.10**
VAE	30.38	30.58	5.67	30.41	176.84	4.03	10.46	5.33	4.62

Table S4. Performance on Train Datasets

Model	Annthy	Mnist	PenDigits	Satellite	Shuttle	Thyroid
CHAODA-fast	0.64	0.67	0.83	0.63	**0.97**	**0.89**
CHAODA	**0.85**	**0.71**	**0.87**	**0.77**	0.86	**0.91**
ABOD	0.50	0.60	0.53	0.51	0.54	0.50
AutoEncoder	0.69	0.67	0.58	0.63	0.94	0.88
CBLOF	0.59	0.62	0.59	0.68	**0.99**	0.87
COF	0.59	0.56	0.53	0.56	0.52	0.49
HBOS	**0.84**	0.53	0.52	0.62	0.74	0.86
IFOREST	0.70	0.61	0.63	0.70	0.91	**0.91**
KNN	0.65	0.65	0.51	0.56	0.53	0.56
LMDD	0.52	0.59	0.56	0.42	0.92	0.70
LOCI	*TO*	*TO*	*TO*	*TO*	*TO*	*TO*
LODA	0.63	0.66	0.57	0.65	0.96	**0.90**
LOF	0.60	0.57	0.52	0.57	0.53	0.49
MCD	0.72	0.57	0.53	0.58	0.96	0.85
MOGAAL	0.46	*TO*	0.67	0.59	*TO*	0.49
OCSVM	0.62	0.63	0.59	0.62	**0.97**	0.78
SOD	0.64	0.55	0.52	0.52	*TO*	0.53
SOGAAL	0.46	0.57	0.57	0.60	0.96	0.49
SOS	0.50	0.52	0.52	0.47	*TO*	0.50
VAE	0.69	0.67	0.58	0.63	0.94	0.88

Levin, D. A. and Peres, Y. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

Table S5. Time taken, in seconds, on Train Datasets

Model	**Annthy**	**Mnist**	**PenDigits**	**Satellite**	**Shuttle**	**Thyroid**
CHAODA-fast	35.92	93.81	58.03	54.60	17.07	35.51
CHAODA	112.86	732.15	481.12	314.60	1e4	108.99
ABOD	3.42	16.39	2.83	3.53	21.58	1.25
AutoEncoder	22.80	40.54	23.87	26.74	158.72	12.84
CBLOF	1.01	1.14	0.23	0.30	0.71	0.21
COF	215.63	277.91	199.03	176.35	1e4	54.82
HBOS	1.58	**0.06**	**0.01**	**0.02**	**0.03**	**0.00**
IFOREST	0.73	1.41	0.80	0.84	3.24	0.54
KNN	0.84	14.30	1.25	2.05	9.13	0.42
LMDD	107.05	1e3	197.48	371.63	6e3	32.69
LOCI	*TO*	*TO*	*TO*	*TO*	*TO*	*TO*
LODA	0.13	0.13	0.12	0.11	0.59	**0.08**
LOF	0.26	14.79	0.90	1.71	6.30	**0.09**
MCD	1.50	20.56	3.01	10.39	10.78	1.03
MOGAAL	594.69	*TO*	561.40	513.84	*TO*	280.80
OCSVM	2.70	11.30	3.04	3.83	162.18	0.68
SOD	262.80	281.86	195.81	253.71	*TO*	91.48
SOGAAL	61.14	68.03	55.40	50.07	460.66	31.43
SOS	101.82	54.22	46.02	42.62	*TO*	42.10
VAE	27.95	50.16	29.63	33.51	182.23	16.18