Efficient Non-Parametric Neural Density Estimation and Its Application to Outlier and Anomaly Detection

Joseph Alejandro Gallego Mejia

Universidad Nacional de Colombia
Departamento de Ingeniería de Sistemas e Industrial
Bogotá, Colombia
2023

Efficient Non-Parametric Neural Density Estimation and Its Application to Outlier and Anomaly Detection

Joseph Alejandro Gallego Mejia

Submitted to the Engineering School of the Universidad Nacional de Colombia, in fulfillment of the requirements for the degree of:

Doctor of Engineering

Systems and Computer Engineering

Advisor:
Fabio A. Gonzalez Ph.D.

Research area:
Computer Science

Research Group:
MindLab Research Group

Universidad Nacional de Colombia
Departamento de Ingeniería de Sistemas e Industrial
Bogotá, Colombia
2023

Estimación neuronal no paramétrica eficiente de la densidad y su aplicación a la detección de valores atípicos y anomalías

Joseph Alejandro Gallego Mejia

Presentado a la Facultad de Ingeniería de la Universidad Nacional de Colombia, en cumplimiento de los requisitos para optar al título de:
Doctor en Ingeniería
Ingeniería de Sistemas y Computación

Director:
Fabio A. Gonzalez Ph.D.

Area de investigación:
Ciencias de la computación

Grupo de invetigación:
Grupo de investigación MindLab

Universidad Nacional de Colombia
Departamento de Ingeniería de Sistemas e Industrial
Bogotá, Colombia
2023

This work is dedicated to the individuals who have provided unwavering support and guidance throughout my life's journey: my parents, including my stepfather; my wife; my advisor; my siblings; and my friends.

This goal has been a lifelong aspiration, one that I have carried since my early years, and today, I proudly bring it to fruition.

In a fast-paced world filled with challenges, it's up to us as a society to ensure the wellbeing of all living things, both now and for future generations.

Acknowledgements

I would like to express my heartfelt gratitude to every individual who played a significant role in my journey towards attaining a doctorate in systems and computing engineering. Firstly, I am immensely grateful to my esteemed professor, Fabio González, whose invaluable guidance and mentorship paved the way for the completion of this thesis. Secondly, I extend my thanks to my beloved wife, who has been actively involved in every decision that allowed me to pursue my research aspirations. Their unwavering support has been instrumental in my success. Thirdly, I want to express my profound appreciation to my colleagues Juan Lara and Santiago Toledo, whose unwavering support have been with me throughout this journey. Additionally, I am indebted to Oscar Bustos and Juan Felipe Osorio for their invaluable assistance, which made this entire process more engaging and fulfilling. Their contributions have been truly impactful. Lastly, I would like to express my gratitude to this magnificent planet that we call home, as it provides us with the opportunity to exist and live our lives to the fullest.

Abstract

The main goal of this thesis is to propose efficient non-parametric density estimation methods that can be integrated with deep learning architectures, for instance, convolutional neural networks and transformers. A recent approach to non-parametric density estimation is neural density estimation. One advantage of these methods is that they can be integrated with deep learning architectures and trained using gradient descent. Most of these methods are based on neural network implementations of normalizing flows which transform an original simpler distribution to a more complex one. The approach of this thesis is based on a different idea that combines random Fourier features with density matrices to estimate the underlying distribution function. The method can be seen as an approximation of the popular kernel density estimation method but without the inherent computational cost. Density estimation methods can be applied to different problems in statistics and machine learning. They may be used to solve tasks such as anomaly detection, generative models, semi-supervised learning, compression, text-to-speech, among others. This thesis explores the application of the method in anomaly and outlier detection tasks such as medical anomaly detection, fraud detection, video surveillance, time series anomaly detection, industrial damage detection, among others.

Keywords:

Kernel density estimation, Kernel methods, Deep Learning, Random Fourier Features, Machine Learning, Deep Kernel, Large-scale learning, Kernel Density Estimation Approximations, Density Matrix, Neural Density Estimation.

Resumen

El objetivo principal de esta tesis es proponer métodos eficientes de estimación de densidad no paramétrica que puedan integrarse con arquitecturas de aprendizaje profundo, por ejemplo, redes neuronales convolucionales y transformadores. Una aproximación reciente a la estimación no paramétrica de la densidad es la estimación de la densidad usando redes neuronales. Una de las ventajas de estos métodos es que pueden integrarse con arquitecturas de aprendizaje profundo y entrenarse mediante gradiente descendente. La mayoría de estos métodos se basan en implementaciones de redes neuronales de flujos de normalización que transforman una distribución original más simple en una más compleja. El enfoque de esta tesis se basa en una nueva idea diferente que combina características aleatorias de Fourier con matrices de densidad para estimar la función de distribución subyacente. El método puede considerarse una aproximación al popular método kernel density estimation, pero sin el coste computacional inherente. Los métodos de estimación de la densidad pueden aplicarse a diferentes problemas en estadística y aprendizaje automático. Pueden ser utilizados para resolver tareas como la detección de anomalías, modelos generativos, aprendizaje semi-supervisado, compresión, texto a habla, entre otros. El presente trabajo se centra principalmente en la aplicación del método en tareas de detección de anomalías y valores atípicos como la detección de anomalías médicas, la detección de fraudes, la videovigilancia, la detección de anomalías en series temporales, la detección de daños industriales, entre otras.

Palabras claves:

Estimación de la densidad del núcleo, métodos del núcleo, aprendizaje profundo, características aleatorias de Fourier, aprendizaje automático, núcleo profundo, aprendizaje a gran escala, aproximaciones de la estimación de la densidad del núcleo, matriz de densidad, estimación de la densidad neuronal.

Content

. Acknowledgements v

Abstract vii

Resumen viii

1. Introduction 2
   1.1. Motivation . . . 2
   1.2. Problem Statement . . . 5
   1.3. Objectives . . . 6
      1.3.1. Main goal . . . 6
      1.3.2. Specific objectives . . . 6
   1.4. Contributions and Academic Products . . . 6
      1.4.1. Publications . . . 7
      1.4.2. Software . . . 8
      1.4.3. Datasets . . . 9
      1.4.4. Other Contributions . . . 9
   1.5. Organization of this thesis . . . 11

2. Background and Related Work 12
   2.1. Density Estimation . . . 12
      2.1.1. Density Estimation and Kernel Density Estimation . . . 13
      2.1.2. Neural Density Estimation . . . 15
      2.1.3. Density Estimation Applications . . . 17
      2.1.4. Efficient Kernel Density Estimation . . . 18
      2.1.5. Density Matrices . . . 19
   2.2. Anomaly Detection . . . 20
      2.2.1. Anomaly Detection and Outlier Detection . . . 20
      2.2.2. Streaming Anomaly Detection . . . 24
   2.3. Kernel Approximation Techniques - Random Fourier Features . . . 26
   2.4. Brief Summary and Perspectives . . . 28

I. Neural Density Estimation 32

3. DEMANDE: Density Matrix Neural Density Estimation 33
3.1. Introduction . . . 33
3.2. Density Matrix Neural Density Estimation (DEMANDE) . . . 35
3.2.1. Density Matrix Neural Density Estimation (DEMANDE) . . . 37
3.2.2. Adaptive Fourier feature learning . . . 41
3.2.3. Complexity Analysis . . . 44
3.3. Experimental Evaluation . . . 44
3.3.1. Fast Kernel Density Estimation . . . 45
3.3.2. Unconditional Density Estimation . . . 47
3.3.3. Unconditional Random Density Estimation in Higher Dimensions . . . 49
3.3.4. Conditional Density Estimation . . . 50
3.3.5. DEMANDE versus DEMANDE-SGD discussion . . . 52
3.4. Conclusion . . . 52

II. Anomaly Detection 58

4. Quantum Anomaly Detection through Density Matrices, Adaptive Fourier Features and Autoencoders 59
4.1. Introduction . . . 59
4.2. Quantum Latent Density Estimation for Anomaly Detections (LEAN-DMKDE) 61
4.2.1. Autoencoder . . . 62
4.2.2. Quantum Measurement Kernel Density Estimation . . . 62
4.2.3. Anomaly Detection Step . . . 64
4.2.4. Training Strategy . . . 64
4.3. Experimental Evaluation . . . 65
4.3.1. Experimental Setup . . . 65
4.3.2. Results and Discussions . . . 69
4.3.3. Ablation Study . . . 69
4.3.4. Dataset Description . . . 71
4.3.5. Statistical Analysis . . . 75
4.4. Conclusions . . . 75

5. Continuous and Incremental Quantum Anomaly Detection 78
5.1. Introduction . . . 78
5.2. Incremental Quantum Measurement Anomaly Detection (InQMAD) . . . 81
5.2.1. Adaptive Fourier Features . . . 81
5.2.2. Density Matrix Initialization . . . 82
5.2.3. Quantum Measurement . . . 83

5.2.4. Anomaly Detection Classification 83
5.2.5. Density Matrix Update 83
5.2.6. Complexity Analysis 84
5.3. Experimental Evaluation 85
5.3.1. Comparison to Streaming Methods 86
5.3.2. Ablation Study 92
5.3.3. Parameters 93
5.3.4. Statistical Tests 94
5.4. Conclusion 95

6. Conclusions and Perspectives **96**

Bibliography **98**

List of Figures

2-1. Taxonomy map for density estimation. 13
2-2. 1-D synthetic dataset. The gray zone is the area of true density. The estimated pdf calculated by KDE ($\sigma = 0,3535$) is shown. 14
2-3. Taxonomy map for density estimation applications. 29
2-4. Taxonomy map for efficient kernel density estimation literature. 29
2-5. Taxonomy map for density matrices used in machine learning literature. . . 30
2-6. Taxonomy map for anomaly detection. Based on the ideas exposed in [24]. . 30
2-7. Taxonomy map for random Fourier features approximation literature. . . . 31

3-1. DEMANDE explanation figure 37
3-2. Adaptive Fourier Features explanation figure 42
3-3. Comparison between Fourier features methods 43
3-4. We analyze nine synthetic datasets, each of which is labeled based on its probability and energy function. These datasets include: (1) Multivariate normal distribution, (2) Arc distribution, (3) Mixture Gaussian distribution, (4) Star, (5) Swiss Roll, (6) Potential 1, (7) Potential 2, (8) Potential 3, and (9) Potential 4. 45
3-5. Comparison of the efficacy of each algorithm on six synthetic data sets. The x-axis is a logarithmic scale of 10^i where $i \in \{1, \cdots, 5\}$. The y-axis represents the mean average error between the prediction of the algorithm and the true density. 53
3-6. Comparison of the efficiency of each algorithm on six synthetic data sets. The x-axis is a logarithmic scale of 10^i where $i \in \{1, \cdots, 5\}$. The y-axis represents the time consumed by each algorithm in milliseconds used by the central processing unit (CPU). 54
3-7. (Top) Randomly generated samples, (bottom) experimental results of unconditional random density estimation in higher dimensions. The x-axis is the dimensions, and the y-axis is the Spearman's correlation. 55
3-8. Each graph represents the density estimate obtained by a specific algorithm on a two-dimensional data set. From left to right the algorithms are Real Density, DEMANDE, DEMANDE-SGD, Made, Inverse Maf, Planar Flow, and Neural Splines. From top to bottom, the data sets are Arc, Bimodal Gaussian, two-dimensional Gaussian distribution, Potential 1-4, Star, and Swizz Roll. . . . 56

3-9. Each graph represents the comparison of Spearman's correlation between the real density for each data set and the density estimate obtained by each algorithm. From left to right the algorithms are DEMANDE, DEMANDE-SGD, Made, Inverse Maf, Planar Flow, and Neural Splines. From top to bottom, the data sets are Arc, Bimodal Gaussian, 2-dimensional Gaussian distribution, Moons, Potential 1-4, Star, and Swizz Roll. 57

4-1. Quantum Anomaly Detection through Density Matrices, Adaptive Fourier Features and Autoencoders (LEAN-DMKDE) method 61
4-2. Results for Friedman-Nemenyi test over the metrics of Auc-PR and Auc-ROC. 70

5-1. Incremental Quantum Measurement Anomaly Detection (InQMAD) method 81
5-2. Importance comparison between memory based against exponential decay. . 84
5-3. Results of Friedman-Nemenyi test, performed on AUCROC performances of all methods over all datasets. 95

List of Tables

3-1. Notation 36
3-2. Complexity analysis of DEMANDE with and without spectral decomposition. 44
3-3. Spearman correlation for Unconditional Density Estimation Experiment. . . 48
3-4. Mean average error for Unconditional Density Estimation Experiment. . . . 48
3-5. Data sets used for conditional density estimation. 50
3-6. Accuracy results in conditional density estimation experiment using neural density estimation methods. 51

4-1. Main features of the datasets. 67
4-2. Parameters of all the algorithms selected for grid search. 68
4-3. Ablation study: area under the Precision-Recall curve (AUC-PR) for all algorithms applied on all data sets. The highest values have been highlighted, the first in bold and the second in underline. 71
4-4. Range of all the parameters used for grid search. 72
4-5. Area under the Precision-Recall curve (AUC-PR) for all algorithms applied on all data sets. The highest values have been highlighted, the first in bold and the second in underline. 76
4-6. Area under the ROC curve (AUC-ROC) for all algorithms applied on all data sets. The highest values have been highlighted, the first in bold and the second in underline. 77

5-1. Main features of the datasets 89
5-2. Area under the ROC curve (AUCROC) for all algorithms over all datasets. The first, second and third positions are respectively highlighted in bold, underlined and italics. 91
5-3. Results for Ablation Study on InQMAD, including Adaptive and NoAdaptive versions 92
5-4. Best combinations of InQMAD parameters for each dataset 93
5-5. Best parameters for baseline methods 94

List of Algorithms

1. Density Matrix Neural Density Estimation prediction (DEMANDE) 40
2. Density Matrix Neural Density Estimation training (DEMANDE) 40
3. Density Matrix Neural Density Estimation training using gradient descent (DEMANDE-SGD) 41
4. Adaptive Fourier Feature learning 43
5. LEAN-DMKDE training process 66
6. InQMAD initialization process 86
7. InQMAD inference and density matrix update 87

1. Introduction

The primary objective of this thesis is to introduce novel non-parametric methods for density estimation, which will be applied to anomaly detection tasks. Density estimation is a crucial statistical task that involves estimating the underlying distribution function from observed data. In order to achieve high accuracy and precision in density estimation, it is necessary to develop robust methods that accurately capture the underlying distribution function, as certain methods may produce biased or high-variance estimations. Broadly speaking, density estimation methods can be classified into two categories: parametric and non-parametric. Parametric methods are limited by predefined functions with free parameters, while non-parametric methods do not make any assumptions about the distribution function. One of the most widely used non-parametric methods for density estimation is kernel density estimation, however, this method has certain drawbacks that are fully explained in Chapter 3. To address these limitations, this thesis proposes a novel non-parametric density estimation method that overcomes the issues associated with kernel density estimation. Additionally, we present two novel methods for anomaly detection, as well as a new approach to streaming anomaly detection. The proposed methods offer significant improvements over existing methods and are evaluated in detail in this thesis.

1.1. Motivation

In statistics, density estimation is a task in which an estimate of the true underlying distribution function is constructed from experimental or observed data. First, one must define random variables, which are measurable functions in a probability space that maps from the sample space to a real number. These random variables have a probability distribution function (PDF) that can be interpreted as the relative probability of an outcome. Often, we have data from an experiment in which the underlying PDF is unknown. Therefore, in this scenario, we have two possible use cases for density estimation; we can assume that our data come from a proposed parametric model, or we can use a nonparametric model approach. The former assumes that the process was generated from a specific type of q_θ density functions, where q_θ are the adjustable parameters of the model, such as the exponential family of distributions. In general, however, the underlying process is arbitrary and may not follow a parametric distribution. The latter assumes no particular distribution and uses the data points to estimate the probability density function. A parametric model is

an excellent choice when statistical goodness-of-fit tests, such as the Kolmogorov-Smirnov test, are available. Thus, we can apply these tests to compare the experimental samples with the reference probability distribution. However, in several experiments, it is not easy to propose a probability distribution. Moreover, parametric methods, such as multivariate Gaussian in a high-dimensional space, scale quadratically. In such cases, it is a better choice to use a nonparametric approach such as kernel density estimation (KDE), also known as the *Parzen-Rosenblatt window* [136, 155]. Another good reason for selecting a nonparametric approach is its ability to obtain an optimal fit to any input distribution as more training samples become available. In contrast, no model with parametric assumptions can have that property [166].

A variety of real problems can be solved using density estimation such as generative models, spatial analysis, classification, anomaly and outliersd́etection, among others. First, generative models attempt to generate new samples from a given probability distribution. This probability distribution can be estimated explicitly or implicitly. For example, autoregressive models and neural flow models use explicit maximum likelihood estimation to find an estimate of the underlying probability density function and, from there, are used to generate new samples such as images and audio [135, 151]. Second, density estimation can be used in spatial analysis to construct choropleth maps. Each shaded part of the choropleth maps represents areas of measurement variability [19, 169]. Third, density estimation can be used as an intermediate step for supervised and unsupervised learning, for example, to classify points into different classes [180]. Fourth, anomaly and outliersd́etection algorithms are used to recognize sample data points that deviate from the rest of the samples where, for instance, an explicit PDF can be used to classify points with very low probability as outliers [108].

Anomaly and outlierd́etection is a very common task in a variety of applications such as fraud detection [201, 143], video surveillance [92, 188], medical anomaly detection [122, 23], time series anomaly detection [119], industrial damage detection [119, 147], among others. Given a set of points $\{x_1, \cdots, x_n\}$, typically the task in anomaly detection is to categorize the data into normal and anomalous points. In this context, it is usually assumed that the anomalous points are generated by a different generative process than the normal points. However, this assumption is often not fulfilled, since, for example, normal data generated by a machine in an industrial factory may become anomalous due to a change in the underlying process. Another problem in anomaly detection is that in the vast majority of data sets the data are not labeled, so that both normal and abnormal data may appear indistinguishable. Therefore, three different approaches are followed to create anomaly detection algorithms. First, a supervised learning approach in which normal and anomalous data points are used in the training phase [89, 172, 87]. The problem with this approach is that anomalous data is rare, so it is difficult to generate such training data points. Therefore, this approach has the drawback of dealing with unbalanced data sets. Second, the *clean* approach in which only normal points are used in the training phase and in the testing phase the dataset is

contaminated with anomaly data points. Third, the unsupervised learning approach, in which both normal and anomaly points are used as training data points, and the idea is to separate normal and anomaly points without any labeling [186, 124, 107]. The problem with this approach is that commonly, in the literature, algorithms assume that the anomalous points have a well-defined anomaly distribution; however, in general, this assumption does not hold because, for example, adversaries always want to try to defeat our defense algorithm by showing them as similar as possible to the normal data points. In addition, a diverse set of potential causes, such as a new device failure model, may deviate slightly from the normal point distribution. Besides, there is an intrinsic problem with high-dimensional data sets, where we need large data sets to cover the entire space.

The KDE is perhaps the preferred nonparametric density estimation tool used by statisticians, engineers, and scientists. In this method, we want to approximate the underlying distribution function f based on the finite data samples. We assume no prior parametric distribution function. However, one of the drawbacks of KDE is its low efficiency, i.e., the time consumed to make a prediction is linear in the number of training data points. Moreover, it is a memory-based algorithm, i.e., the density prediction of new data points is done using all the training data points; therefore, we need to store each training data point to make a new prediction. Due to these problems, the approximate fast evaluation of nonparametric density estimation is an active research topic. Different approaches are proposed in the literature: higher-order divide-and-conquer methods [69], near- and far-field separation (pruning) [118], and hashing-based estimators (HBE) [27]. So far, hashing-based estimators seem to be the best state-of-the-art method for approximate kernel density estimation. In this thesis, a new algorithm for approximating KDE will be proposed and evaluated on anomaly detection problems.

This thesis addresses density estimation using a new perspective that combines density matrices, which is an important formalism in quantum mechanics, and random Fourier features, which explicitly compute an approximate feature space of certain kernels, to perform density estimation. Density matrices have not been widely used in machine learning; however, its combination of linear algebra and probability is a powerful tool that has a high potential impact in different machine learning tasks as has been shown by [64, 66]; therefore, in this thesis we started from the initial ideas presented in [64] and [66] and extended them and explored new algorithms on density matrices and random Fourier features. In addition, we aimed to design new competitive and efficient density estimation algorithms with additional properties such as being able to connect to deep learning architectures. New algorithms based on these ideas have been presented to solve anomaly detection tasks. One important anomaly detection task is anomaly detection for streaming data [199], where data is continuously generated. This task can be solved using the novel algorithm proposed, which utilizes density matrices based on the presented ideas. This approach is advantageous due to its fast speed and low-resource training phase.

1.2. Problem Statement

Anomaly and outlier detection is an important task in different research areas with various application domains. One of the main challenges in anomaly detection is that usually only normal data points are available. This problem can be solved using an unsupervised learning approach; however, its performance is lower than the supervised one, but it avoids the process of generating anomalous data where in some real tasks they are difficult or even impossible to obtain. Examples of unsupervised algorithms brought from the supervised world for anomaly detection are the one-class support vector machines or the isolated forest algorithm. These methods assume that the data come from a single class; it has been shown in the literature that these algorithms obtain inferior results to methods that can handle multimodal datasets. Other algorithms that solve anomaly detection tasks from the unsupervised world are density-based algorithms such as parametric density estimation or kernel density estimation. However, current density estimation algorithms suffer from the curse of dimensionality and their linear complexity in the number of training data is prohibitive in most application domains. Typically, this problem is solved by using a dimensionality reduction approach, such as principal component analysis, and a density estimation algorithm, such as kernel density estimation.

A challenge in density estimation is how to build efficient algorithms. Algorithms capable of producing an approximation to the underlying distribution function are applied to a variety of problems in machine learning. State-of-the-art algorithms, such as density neural networks, provide a useful tool for estimating density, and this estimate can be used to solve a variety of problems, such as anomaly and outlier detection problems. However, algorithms from literature, such as autoregressive models or latent models, are computationally heavy and time consuming. Other algorithms that are based on adversarial networks have the drawback of not explicitly estimating density, but compute it intrinsically; therefore, the search for new algorithms that can estimate density with maximum likelihood and are able to deal with high dimensionality using few resources compared to current solutions is of utmost importance. Finally, with the advent of deep learning and its powerful feature extraction capabilities, the proposed new algorithms must be able to connect with and train alongside other networks.

Recent research in the literature points to the need for new algorithms that predict the underlying density estimation to find anomaly patterns and outlier data points [129, 113]. Therefore, some open questions need to be addressed:

- How to design a nonparametric density estimation method that is efficient in terms of time and space?
- How to design a method capable of integrating with other deep learning methods for nonparametric density estimation?
- Is the approximation given by the proposed method better in terms of efficiency than the state-of-the-art approximation methods for kernel density estimation?

- How to design methods for anomaly and outlier detection using nonparametric density estimation?
- What is the impact of new methods for anomaly and outlier detection using nonparametric density estimation?

1.3. Objectives

1.3.1. Main goal

To develop a non-parametric method for density estimation that is efficient, can be integrable with other deep learning frameworks, and can be used for solving anomaly detection tasks.

1.3.2. Specific objectives

- To design a non-memory-based method for efficient kernel density estimation.
- To evaluate the efficiency of the method and to compare it against other state-of-the-art kernel density approximation methods.
- To propose new algorithms for anomaly detection using density matrices and random Fourier features.
- To evaluate the model in anomaly detection tasks.

1.4. Contributions and Academic Products

This PhD thesis is a body of work, encompassing a diverse range of contributions to the academic community. It includes four journal papers, two of which have already been accepted and two more that are currently under review. In addition, the thesis contains five international full-size conference papers, one student abstract paper, and one doctoral consortium paper, along with nine pre-print articles that have been made available to the wider research community. The candidate has also made contributions to open-source software, with the creation of six new public repositories and the creation of a public dataset for density estimation tasks. The candidate has presented their research on five international conference posters and presentations. The thesis also includes the supervision and support of an undergraduate statistics thesis and a Master's in computer science thesis.

1.4.1. Publications

Neural Density Estimation

- **Gallego, J.A.**, Osorio, J.F., González, F.A. (2022). Fast Kernel Density Estimation with Density Matrices and Random Fourier Features. In: Bicharra Garcia, A.C., Ferro, M., Rodríguez Ribón, J.C. (eds) Advances in Artificial Intelligence – IBERAMIA 2022. IBERAMIA 2022. Lecture Notes in Computer Science, vol 13788. Springer, Cham. https://doi.org/10.1007/978-3-031-22419-5_14

  We have demonstrated that the Density Matrix Kernel Density Matrix can effectively address the challenges associated with efficient kernel density estimation.

- **Gallego, J.A.**, & González, F.A. (2023). DEMANDE: Density Matrix Neural Density Estimation. IEEE Access (accepted). doi: https://10.1109/ACCESS.2023.3279123

  We have introduced a novel neural density estimation method called DEMANDE, which utilizes adaptive Fourier features in combination with density matrices.

- González, F. A., **Gallego, A.**, Toledo-Cortés, S., & Vargas-Calderón, V. (2022). Learning with density matrices and random features. Quantum Machine Intelligence, 4(2), 1-17. https://doi.org/10.1007/s42484-022-00079-9

  We introduced the Density Matrix Kernel Density Estimation framework, which forms the basis of this PhD thesis. Our work demonstrates that this method can be applied to a variety of tasks, including classification, regression, ordinal regression, density estimation, and more.

- **Gallego-Mejia, J.** (2023). Efficient Non-Parametric Neural Density Estimation and Its Application to Outlier and Anomaly Detection (Doctoral Consortium). Association for the Advancement of Artificial Intelligence (AAAI 2023)

  We have summarized all the methods developed during this PhD in a brief doctoral consortium paper, which we presented at the Doctoral Consortium hosted by the Association for the Advancement of Artificial Intelligence (AAAI 2023).

Anomaly Detection

- **Gallego-Mejia, J.**, Bustos-Brinez, O., & González, F. A. (2022). LANDM and ANDM: Quantum Inspired Density Matrices for Anomaly Detection. (To Be Submitted on Data Mining and Knowledge Discovery)
  arXiv:2211.08525.

  We have introduced two novel methods for anomaly detection that use an autoencoder to capture a feature representation and DMKDE to compute probabilities.

- **Gallego-Mejia, J.**, Bustos-Brinez, O., & González, F. "InQMAD: Incremental Quantum Measurement Anomaly Detection," 2022 IEEE International Conference on Data Mining Workshops (ICDMW), Orlando, FL, USA, 2022, pp. 787-796, doi: 10.1109/ICDMW58026.2022.00107.

  We have introduced a novel incremental anomaly detection method that utilizes density matrices and adaptive Fourier features. This method can be viewed as an exponential moving average method.

- **Gallego-Mejia, J.**, Bustos-Brinez, O., & González, F. "Streaming and Incremental Anomaly Detection through Density Matrices" 2023 Springer Neural Computing and Applications (To Be Submitted, Special Issue Invitation on Incremental Learning from ICDM Organization).

  We have been invited by ICDM Workshop organizers to make an extended version of the InQMAD article. In this new article, we have introduced a novel new incremental anomaly detection method using density matrices and adaptive Fourier features. With InQMAD, these methods can be considered a simple moving average and an exponential moving average, respectively.

- Bustos-Brinez, O., **Gallego-Mejia, J.**, & González, F. A. (2022). Anomaly Detection through Density Matrices and Kernel Density Estimation (AD-DMKDE). Neural Information Processing Systems Conference: LatinX in AI (LXAI) Research Workshop 2022, New Orleans, USA.

  We have introduced a novel method called AN-DMKDE, which utilizes density estimations as a measure of abnormality.

1.4.2. Software

- **Gallego-Mejia, J.**, Bustos-Brinez, O., González, F. A (2023). Joaggi/lean-dmkde: v1.0 (v1.0). Zenodo. https://doi.org/10.5281/zenodo.7709642
- **Gallego-Mejia, J.**, González, F. A (2022). Joaggi/demande: v1.0 (v1.0). Zenodo. https://doi.org/10.5281/zenodo.7709634
- **Gallego M., Joseph A.**, Osorio, Juan F., González, Fabio A. (2022). Fast Kernel Density Estimation with Density Matrices and Random Fourier Features Software (1.0.1). Zenodo. https://doi.org/10.5281/zenodo.6941020
- Bustos-Brinez, O., **Gallego-Mejia, J.**, González, F. A. (2022). Joaggi/anomaly-detection-density-matrix-kernel-density-estimation: v1.0.0 (v1.0.0). Zenodo. https://doi.org/10.5281/zenodo.7308904
- **Gallego-Mejia, J.**, Bustos-Brinez, O., González, F. A (2022). Joaggi/Incremental-

Anomaly-Detection-using-Quantum-Measurements: v1.0.0 (v1.0.0). Zenodo. https://doi.org/10.5281/zenodo.7183564

- **Gallego-Mejia, J.**, González, F. A (2023). Joaggi/Robust-kernels-for-robust-location-estimation: v1.0 (v1.0). Zenodo. https://doi.org/10.5281/zenodo.7709651

1.4.3. Datasets

- **Gallego-Mejia, Joseph A.**, Gonzalez, Fabio A. (2023). DEMANDE Dataset (V1.0) [Data set]. Zenodo. https://doi.org/10.5281/zenodo.7822851

1.4.4. Other Contributions

Papers

- **Gallego, J. A.**, González, F. A., & Nasraoui, O. (2021). Robust kernels for robust location estimation. Neurocomputing, 429, 174-186 [54].

  We demonstrated that the Gaussian kernel possesses inner robustness due to its relationship with the Welsh M-Estimator. Additionally, we introduced four novel robust kernels that share a relationship with M-Estimators.

- **Gallego, J. A.**, González, F. A., (2019). Robust Estimation in Reproducing Kernel Hilbert Space [Poster Presentation]. Neural Information Processing Systems Conference: LatinX in AI (LXAI) Research Workshop 2019, Vancouver, Canada. https://doi.org/10.52591/lxai2019120829

- Useche, D. H., Bustos-Brinez, O. A., **Gallego, J. A.**, & González, F. A. (2022). Computing expectation values of adaptive Fourier density matrices for quantum anomaly detection in NISQ devices. In arXiv: 2201.10006.

  We demonstrated that the novel DMKDE method can be implemented on a quantum computer. Our experiments involved using DMKDE for density estimation and anomaly detection tasks.

- **Gallego-Mejia, J.**, Bustos-Brinez, O., & González, F. A. (2023). LEAN-DMKDE: Quantum Latent Density Estimation for Anomaly Detection (Student Abstract). Association for the Advancement of Artificial Intelligence (AAAI 2023)

  We have introduced a novel method for anomaly detection that utilizes an autoencoder to capture a feature representation and DMKDE to calculate probabilities. It is published as a student abstract conference paper.

Coadvisor

- Osorio Ramírez, Juan Felipe (2021). On the performance of Kernel Density Estimation using Density Matrices. Undergraduate Thesis, Universidad Nacional de Colombia.
- Bustos-Brinez, O., (2023). Desarrollo de un algoritmo para detección de anomalías basado en estimación de densidad basada en kernels, matrices de densidad y medias cuánticas, Master's Thesis, Universidad Nacional de Colombia.

Posters

Gallego-Mejia, Joseph A., González, Fabio A. (2023). Efficient Non-Parametric Neural Density Estimation and Its Application to Outlier and Anomaly Detection (v1.0.0). Association for the Advancement of Artificial Intelligence (AAAI 2023). Zenodo. https://doi.org/10.5281/zenodo.7633808

Gallego-Mejia, Joseph A., Bustos-Brinez, Oscar, González, Fabio A. (2023). LEAND: Quantum Latent Density Estimation for Anomaly Detection (Student Abstract) (v1.0.0). Association for the Advancement of Artificial Intelligence (AAAI 2023), Washington DC, USA. Zenodo. https://doi.org/10.5281/zenodo.7633793

Gallego-Mejia, Joseph A., Bustos-Brinez, Oscar, González, Fabio A. (2022). Poster: Anomaly Detection through Density Matrices and Kernel Density Estimation (AD-DMKDE) (Version 1). LatinX in AI Research at NeurIPS 2022, New Orleans, USA. Zenodo. https://doi.org/10.5281/zenodo.7573566

Gallego-Mejia, Joseph A., Fabio A. González. (2019). Robust Estimation in Reproducing Kernel Hilbert Space. In Neurocomputing (Version 1, Vol. 429, Numbers 14 March 2021, pp. 174–186). Zenodo. https://doi.org/10.5281/zenodo.6604897

Presentations

- Efficient Non-Parametric Neural Density Estimation and Its Application to Outlier and Anomaly Detection - Association for the Advancement of Artificial Intelligence AAAI 2023 - Washington DC, USA - Winner of a travel award to present work - Doctoral consortium Feb. 2023 - Presentation of doctoral research
- InQMAD: Incremental Quantum Measurement Anomaly Detection - The IEEE International Conference on Data Mining (ICDM) - Orlando - Florida (2022), USA - Presentation of research in streaming anomaly detection - Winner of a travel award to present work
- Anomaly Detection Through Density Matrices and Kernel Density Estimation (AD-DMKDE) Nov. 2022 - LatinX in AI Research Workshop co-located with the Thirty-

Third Neural Information Processing Systems (NeurIPS) New Orleans, USA - Presentation of research in anomaly detection - Winner of a travel award to present work

- Fast Kernel Density Estimation with Density Matrices and Random Fourier Features - Nov. 2022 - Sociedad Iberoamericana de Inteligencia Artificial (IBERAMIA) New Orleans, USA - Presentation of the research named Fast Kernel Density Estimation - Winner of a travel award to present work
- LatinX in AI Research Workshop co-located with the Thirty-Third Neural Information Processing Systems (NeurIPS) - Vancouver, Canada - Robust Estimation in Reproducing Kernel Hilbert Dec. 2019 - Presentation of research in robust estimation - Winner of a travel award to present work

1.5. Organization of this thesis

This PhD thesis is focused on the development of novel techniques for density estimation and anomaly detection. Above, we presented an introduction that outlines the motivation for the research, the problem statement, and the objectives. The contributions of the research are also presented, which include neural density estimation and anomaly detection, among others. In the Chapter 2, we provide an overview of density estimation, kernel density estimation, and related topics, such as efficient kernel density estimation and anomaly detection. The first contribution is presented in Chapter 3, which proposes DEMANDE: Density Matrix Neural Density Estimation. The second main contribution is presented in Chapter 4, which proposes AN-DMKDE and LEAN-DMKDE as methods to stationary anomaly detection. The third main contribution is presented in Chapter 6, which proposes InQMAD, a method for streaming anomaly detection. Overall, this thesis presents important new techniques for density estimation and anomaly detection, with potential applications in a range of fields, including finance, healthcare, and security.

2. Background and Related Work

This chapter serves as a comprehensive introduction to the background and related work utilized in this thesis. We start by delving into the fundamentals of density estimation and Kernel Density estimation, followed by the groundbreaking concept of neural density estimation and the novel methods employed in neural flows. To provide a more thorough understanding, we demonstrate various applications of density estimation, and share several efficient kernel density estimation techniques. We then dive into the approximation kernel techniques, with a detailed exploration of the random Fourier feature method. The density matrix formalism utilized in this thesis is introduced, providing a deeper understanding of the subject matter. Moreover, we tackle the anomaly detection problem by defining it and showcasing multiple shallow and deep learning methods that are pertinent to this problem. Finally, we provide an overview of streaming anomaly detection and present the latest state-of-the-art techniques.

2.1. Density Estimation

In this section, our goal is to provide readers with a comprehensive understanding of density estimation and its importance in various fields. We begin by providing a concise explanation of density estimation and its fundamental importance in statistical modeling and data analysis. To further elaborate on the topic, we present the kernel density estimation method, highlighting its strengths and weaknesses. We discuss how this approach uses kernel functions to estimate the underlying probability density function, allowing flexible modeling of data distributions. However, we also acknowledge the limitations of kernel density estimation, such as sensitivity to the choice of kernel bandwidth and its computational complexity. In addition, we explore interesting advances in density estimation techniques known as neural density estimation. These methods take advantage of neural networks to estimate density functions and offer several advantages. In particular, these new approaches can be trained end-to-end using deep learning methodologies, allowing for more efficient and accurate density estimation. In the following, we delve into the various applications of density estimation in various domains. We show how density estimation plays a crucial role in fields such as anomaly detection, image synthesis, and generative modeling, among others. By illustrating these applications, we highlight the practical relevance and versatility of density estimation techniques. In addition to the above content, we also present state-of-the-art

methods to address the challenges and limitations of kernel density estimation. We explore novel approaches that propose innovative solutions to improve the complexity of kernel density estimation. We conclude this section including an explanation of the density matrix, a formalism used in quantum mechanics. This formalism serves as a fundamental concept for the novel methods presented in the following chapters, showing the interdisciplinary nature of density estimation. By enhancing this section with a more complete and coherent narrative, we strive to provide readers with a comprehensive understanding of density estimation, its various methodologies, applications, and ongoing advances in the field.

2.1.1. Density Estimation and Kernel Density Estimation

Density estimation is a branch of statistics in which the idea is to reconstruct the underlying probability density function (PDF) given a set of data points $\{x_1, \cdots, x_n\} \subset \mathbb{R}^d$ [67]. Figure **2-1** shows a perspective of different approaches for density estimation. Three main topics are shown: parametric, nonparametric and neural network density estimation. Parametric density estimation has a variety of literature generated in the last century and is not our main focus in the present thesis. For nonparametric density estimation, the most commonly used algorithm is KDE. Density estimation using neural networks has three main subtopics: autoregressive, adversarial and variational networks.

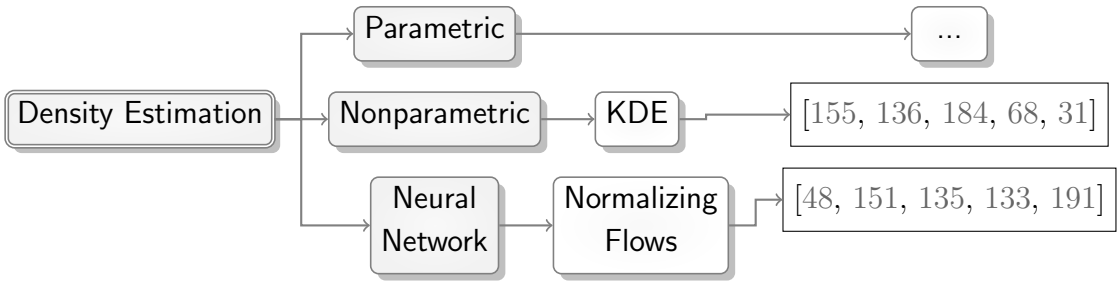

Figure 2-1.: Taxonomy map for density estimation.

In statistics there are several parametric strategies to obtain the PDF, where some parametric density function is commonly assumed, for instance, the Gaussian, Beta or Gamma distribution among others [17]. The problem with this approach is that, in general, the underlying density function is possibly not even known [154]. Therefore, to solve this problem, nonparametric strategies for density estimation, such as kernel density estimation, are used. Kernel Density Estimation (KDE) or also known as Parzen window is a nonparametric estimation of the probability density function [155, 136]. This method makes no particular assumptions about the underlying probability density function. The smooth estimation of the Parzen window has the form:

$$\hat{f}_X(x) = \frac{1}{N\lambda} \sum_{i=1}^{N} k_\lambda(x, x_i) \tag{2-1}$$

where $k_\lambda(\cdot)$ is a kernel function and λ is the smoothing bandwidth parameter of the estimate. A small λ-parameter implies a small degree of smoothing, leading to capture some purely random structures [68, 31]. Conversely, a high λ-parameter implies a high degree of smoothing, possibly leaving behind some important structures [184]. Obtaining the perfect band parameter is a data-dependent optimization problem; hence, various solutions are proposed in the literature, such as ad-hock solutions [13], adaptive functions [171], particle swarm optimization [168], among others.

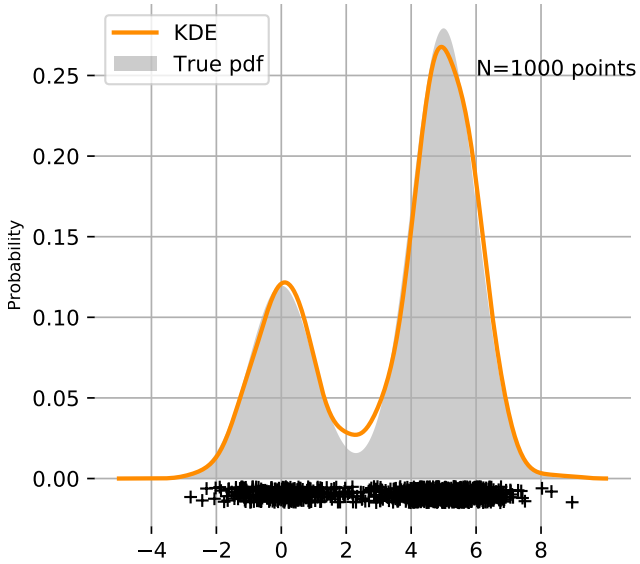

Figure 2-2.: 1-D synthetic dataset. The gray zone is the area of true density. The estimated pdf calculated by KDE ($\sigma = 0,3535$) is shown.

Figure 2-2 shows a synthetic example using KDE. This dataset corresponds to a mixture of univariate Gaussians. The mixture weights are 0.3 and 0.7 respectively and the parameters are $(\mu_1 = 0, \sigma = 1)$ and $(\mu_1 = 5, \sigma = 1)$. It was generated with 10,000 samples for training and use as test dataset 1,000 samples equally spaced in the interval $[-5, 10]$. The gray zone is the area of true density. The estimate pdf calculated by KDE is shown as the orange curve. This figure shows that the KDE is a good estimator for estimating a uni-variate bi-modal probability distribution function, and it can be shown that it can be used in higher dimensions.

In [136], Parzen showed that Eq. 2-1 is an unbiased estimator of the pdf f. When k_λ is the Gaussian kernel, Eq. 2-1 takes the form:

$$\hat{f}_{\gamma,X}(x) = \frac{1}{N(\pi/\gamma)^{\frac{d}{2}}} \sum_{i=1}^{N} e^{-\gamma(\|x_i - x\|)^2} \tag{2-2}$$

The kernel density function (KDE) is perhaps the most widely used nonparametric density estimation tool [31]. However, KDE has the drawback of suffering from the curse of dimensionality, i.e., the time consumed to make a prediction is at least linear in terms of the training data points. This problem has been studied in the last two decades, with hash data structures being one of the most recent solution attempts, see Subsection 2.1.4 for further details. A more efficient density estimation algorithm could be a solution to this difficult problem, in which an approximation is made with the help of a quantum feature map, in particular using a random Fourier feature map.

In recent years, new state-of-the-art methods called neural density estimates have been proposed [48, 151]. With these new methods, prior knowledge of the data can be combined with the flexibility and learning capability of neural networks [135]. Furthermore, these methods differ from other neural networks architectures as they provide an accurate estimate of the probability distribution function [63]. Neural density estimates transform a simple initial distribution such as the Gaussian distribution into a more complex and richer distribution. Some caution must be taken with the transformation because the probability distribution function must integrate to one given the second axiom of Kolmogorov probability. Therefore, given a set of points $x_1, \cdots, x_n$, define $P_X(\mathbf{x})$ as the density function of the random variable X. A transformation $\mathbf{y} = f(\mathbf{x})$ has to integrate to one; therefore, the probability density function is defined as $P_Y(\mathbf{y}) = P_X(f^-1(\mathbf{y}))|det\frac{\partial f^-1}{\partial \mathbf{y}}|$. This transformation could be chained as $f_1 \circ \cdots \circ f_n$ [133, 191].

2.1.2. Neural Density Estimation

Three main approaches have been used in state-of-the-art neural density estimation: autoregressive models, normalizing flows and generative adversarial networks. Autoregressive methods have their origin in the restricted Boltzman machine, which is a Markov random field with bipartite substructure, where a connection is established between the weights $\boldsymbol{W}$ and the observations $\boldsymbol{v}$. One issue of this kind of method is their intractable Z partition function who ensure a valid distribution and sums to 1 [95, 52, 14].

Normalizing Flow models were proposed in the last decades as an improvement of autoregressive flows models, whose strength is based on the change of variables [44, 152]. This change of variable can be composed in a series of differentiable and invertible transformations of

a known density function, for instance, the normal distribution. The change of variables needs to preserve the volume, which imposes a constraint on the availability of base density functions. Nonetheless, these normalizing flow algorithms are difficult to tune, thus their convergence is not always guaranteed [112].

In [112], the authors propose a new algorithm for density estimation using deep generative neural networks. In the case of the discriminators, the z discriminator is used to distinguish the generated latent variable $\hat{z}$ from the real latent variable z. The other discriminator is used to discern the true data x from the generated data $\hat{x}$. This method can use complex deep neural networks as discriminators, e.g., convolutional neural networks or transformers.

The model presented in this chapter follows a different approach to neural density estimation which is based on density matrices and kernel-approximating Fourier features. One of the main advantages of this approach is its simplicity as well as its good performance in some benchmark tasks as shown by the experimental evaluation in Section 5.3. In that section different state-of-the-art neural density methods were used as baselines. These methods are described next:

Normalizing flows models data $\boldsymbol{x}$ using a sequence of invertible and differentiable transformation of a generally simple f-function.

The method uses a change of variables where Z and X are defined as random variables, such that $X = f_\theta(Z)$, $Z = f_\theta^{-1}(X)$, and $f : \mathbb{R}^n \to \mathbb{R}^n$. Then,

$$p_X(\boldsymbol{x}; \theta) = p_Z(f_\theta^{-1}(\boldsymbol{x})) \left| \det \left(\frac{\partial f_\theta^{-1}(\boldsymbol{x})}{\boldsymbol{x}} \right) \right|$$

The volume is conserved in this transformation due to the calculation of the determinant. Generative adversarial networks are based on two generator networks and two discriminators. One of the generators is used to map from z latent space to x space similar to the process performed by normalization flow algorithms. The other generator is used to perform the opposite transformation from x-space to z-space. Some prominent algorithms as presented next:

- Masked Autoregressive Flow (MAF): use the following recursions for each layer: $x_i = u_i \exp \alpha_i + \mu_i$ where $\mu_i = f_{\mu_i}(\boldsymbol{u}_{1:i-})$ [135]. MAF is a generalization of RealNVP.
- Inverse Autoregressive Flow (IAF) [91]: In [135], the authors show that inverse autorregresive flow is a generalization of RealNVP. Define $z_0 = (z'_0 - \mu_0)/\sigma_0$ and $z_i = (z'_i - \mu(z'_{1:i-1})/\sigma(z'_{1:i-1})$, then the Jacobian is lower triangular. This implies that the determinant$|d\boldsymbol{z}/d\boldsymbol{z'}|$ can be computed as $\prod_{i=1}^{D} 1/\sigma_i(z_{1:i-1})$ who is not dependent of z'_i.
- Planar Flow [152]: this normalized flow uses a family of transformation of the form $f(\boldsymbol{z}) = \boldsymbol{z} + \boldsymbol{u}h(\boldsymbol{w}^T\boldsymbol{z} + b)$ where $\boldsymbol{u}$, $\boldsymbol{w}$, b are free parameters and $h(\cdot)$ is an element-wise function. This transformation has a triangular Jacobian.

- Real NVP [45]: this method uses coupling layers as follows $y_{1:d} = x_{1:d}$ and $y_{d+1:D} = x_{d+1:D} \bigodot exp(s(x_{1:d})) + t(x_{1:d})$ where s and t means scale and translation respectively, and $\bigodot$ is the Hadamard product or element-wise product. Also, the inverse of such transformation does not involve the computation of the inverse of neither s or t, therefore these functions can be arbitrarily complex and difficult to invert in particular can be multilayer neural networks.
- Neural Spline Flow [47]: Neural Splines Flow uses a partition of K *nodes* of the space between (-B,-B) and (B,B). The out-of-range transformation is mapped as the identity. This makes the overall transformation linear out-of-range, so it can take unrestricted inputs. Each knot uses a monotone rational-quadratic function. The authors claim that rational-quadratic functions are easy to derive and, due to their monotonic behavior, are also analytically invertible.

2.1.3. Density Estimation Applications

Density estimation models can be used for a wide variety of machine learning tasks, such as anomaly detection, image processing, text-to-speech, and semi-supervised learning, among others. Anomaly detection, outlier detection, and novelty detection algorithms will be presented in Section 2.2.1. Figure **2-3** shows a variety of density estimation applications. Kernel density estimation, parametric models, and neural networks are the main approaches in those applications. KDE has several applications, such as estimation of the underlying probability density function, estimation of confidence intervals and confidence bands [49, 34], local mode finding for geometric feature estimation [29, 32], for estimating ridge density function [61], for constructing cluster trees [12], for estimating cumulative distribution function [130], for estimating receiver operating characteristic (ROC) curves [121], among others.

Density estimation is used to construct clustering algorithms, e.g., Fraley and Raftery [51] used an expectation maximization algorithm with a mixture model to propose a hierarchical agglomerative clustering, Nakaya and Yano [131] used the kernel density estimation algorithm with a spatio-temporal modification to visualize clusters of crime areas, Anderson [6]. Density estimation is also used in spatial analysis, e.g., Borruso [19] used network density estimation to estimate the use of some insurance banks in European cities, Anderson [6] used kernel density estimation to assess injury-related traffic accidents in London, UK, Downs [46] proposed a method to generate the intensity surface in a spatial environment for moving objects using kernel density estimation and time geography. Super-resolution imaging is another use case of density estimation, e.g., Gatopoulos et al. [60] used variational autoencoders and likelihood estimation to generate new super-resolution images, Sejong et al. [167] used a normalizing flow-based model and a large stack of convolutional layers to provide better super-resolution images, Cheng et al. [33] used a mixture model of a Dirichlet process and a Gaussian process regression to estimate the distribution of training patches, Guardnaccia

et al. [70] used kernel density estimation to make a spatial analysis of city noise with information collected from a French mobile app installed on citizens' smartphones using their GPS location data.

Kamalov [88] used kernel density estimation to generate new samples from a given data set, dealing with imbalanced data sets. Lee and Park [100] used kernel density estimation to process the image by subtracting the background in a pixel-based method. This method can dynamically remove backgrounds in the video, using few resources. Density estimation is also used in classification, e.g., [93] proposed an end-to-end pipeline for classification using kernel density estimation for online classification, [66] proposed a classification method using density estimation and Fourier random features. Semi-supervised classification is another task that can be solved using density estimation, e.g., Ji et al. [84] used manifold kernel density estimation to propose a first phase with unlabeled samples where the manifold is extracted and a second phase with labeled samples. Text-to-speech conversion is currently an active research topic. The density estimation obtained by inverse autoregressive flow (IAF) is used in Peng et al. [138] to generate a sequence-to-sequence model thus converting text to speech.

2.1.4. Efficient Kernel Density Estimation

Kernel density estimation is a memory-based algorithm, i.e., it stores each data point in the training phase and uses it to calculate an average kernel distance. This process has a drawback given by the memory space and the time footprint of the process; therefore, new algorithms are proposed in the literature to solve this problem. It should be noted that the time complexity of the original KDE is linear as a function of the training data points. Figure **2-4** shows several approaches for efficient kernel density estimation.

Indyk and Motwd [78] proposed a method for hashing buckets called *Locality-Sensitive Hashing*. This method differs from normal hashing in that it maximizes collisions instead of minimizing them. This method can be considered a dimensionality reduction technique. Using this method, the authors proposed two algorithms to reduce the processing time and realized a sub-linear query time. Gray and Moore [69] proposed a new space partitioning tree that uses the idea of grid search but with a divide-and-conquer approach. Each point in the tree is contained in several hyper-rectangles. The tree is constructed using only a small percentage or number of points. The problem with the latter two approaches is that they scale exponentially according to the number of dimensions. In physics and statistics, this is not a real problem because typical applications have several d dimensions less than or equal to three. However, in recent data sets where the number of dimensions is on the order of tens, better approaches are needed to solve this problem.

Recently, March et al. [117] have proposed a new method for pruning far points and approximating near fields. This algorithm performs fast kernel sums by separating points between

near and far fields. A *tree* data structure is used to perform fast tree traversal summation. The most promising tool in state-of-the-art methods are hashing-based estimators for kernel density estimation, where a hash structure is used to construct close distance boxes that allow the computation of few distances in the prediction step [27, 173, 11]. Here, a sampling scheme is used, where points are sorted into buckets thanks to a hash function whose main objective is to send similar objects to the same hash value.

2.1.5. Density Matrices

One of the main building blocks used in quantum physics to capture classical and quantum probability in a given physical system is the density matrix. This formalism was conceived by Von Neumann [187] as the foundation of quantum statistical mechanics. The density matrix describes the states of the quantum system and explains the relationship between the pure state and the mixed states of the system. Given the Heisenberg uncertainty principle, the position and velocity of a given particle cannot be measured at the same time, even in theory. Therefore, the pure state allows the density matrix to capture the quantum probability of a given particle. However, when the preparation of the system is not fully known, mixed states are required to provide the classical probability in the density matrix. Another possibility is when the particle system has quantum entanglement, in this case, mixed states are used to provide the quantum probability in the density matrix [178]. In this thesis, a density matrix will be used as a component for new quantum-inspired machine learning algorithms combined with kernel density estimation.

Density matrices had been used in some works in machine learning. Figure **2-5** shows some applications of density matrices in machine learning. Density matrices can be used as a building block for machine learning algorithms. According to our research, Wolf [193] showed the first attempt to use density matrix to solve some machine learning problems, including clustering, feature selection, set similarity, and classification. Chatzis et al. [28] proposed a combination between the Gaussian mixture model and density matrices. They showed that the density matrix can represent a linear combination of the simple Gaussian mixture model as a diagonal density matrix. Tiwari and Melucci [182] proposed a new algorithm for information retrieval using the density matrix. They used $|x\rangle$ as the document representation and $|y\rangle$ as the input query. Both elements, non-relevant and relevant documents, were used to compute the density operators ρ_0 and ρ_1 respectively. Sato et al. [161] study proposed a variational Bayes inference based on simulated annealing and the density matrix. Jankovic and Sugiyama [81], Jankovic [79], Janković et al. [82] proposed a robust and non-robust probabilistic PCA using Born's rule. They showed that the algorithm can be solved offline as a sequential and online optimization problem with two different time scales. Jankovic [80] proposed a new quantum-inspired machine learning algorithm called quantum low entropy-based associative reasoning based on quantum Tsallis entropy, the

nearest neighbor algorithm, and support vector machines.

2.2. Anomaly Detection

In this section, we provide an in-depth look at the anomaly detection task in the context of machine learning. We define the task itself and also explore various methods that have been developed to solve the problem. We cover a wide range of techniques, including both traditional approaches rooted in classical ideas and state-of-the-art methodologies that leverage deep learning. In addition to discussing these methods, we delve into the concept of streaming anomaly detection, which is of significant importance when dealing with massive data sets arriving as a data stream. We recognize that in scenarios where acquiring the complete training set from scratch is difficult or even infeasible, streaming anomaly detection becomes a crucial aspect to consider. We examine the unique challenges posed by streaming data and highlight strategies and algorithms that enable effective anomaly detection in such dynamic environments. By incorporating these topics, we aim to provide a comprehensive exploration of the task of anomaly detection. We cover its various methodologies, ranging from classical to state-of-the-art approaches, and highlight the critical role of streaming anomaly detection in effectively analyzing large-scale datasets.

2.2.1. Anomaly Detection and Outlier Detection

The idea in anomaly detection is to classify normal versus abnormal patterns, i.e., to construct a segmentation between normal and abnormal patterns of the data points. Figure **2-6** shows the different methods and application of anomaly detection. In anomaly detection, the normal and abnormal behavior of data points could be given by two different underlying distribution functions; however, it could be a modification of the underlying process. In this case, the task is referred to in the literature as novelty detection, i.e., the algorithm will search for anomalous patterns within the data sets and if there is a change in the process, the anomaly detection model will be updated. An outlier is a data point that differs significantly from other observations; therefore, outlier detection is a task to discriminate those outliers. Anomaly and outlier detection can be used in various applications such as fraud detection, insurance care, intrusion detection for cybersecurity, industrial damage detection, among others.

Let α be the ratio of the anomaly points. When α is high, the most prominent approach for anomaly detection will be to use a supervised learning approach. However, when α is low, the best approach will be to use outlier detection algorithms, e.g., in a fraud detection problem, it will be easier to model the distribution of normal behaviors first and treat the outlier points as outliers later. The advantage of this approach is that we do not have to make any assumptions about the distribution. However, the drawback of this approach is

that the anomaly may not necessarily be an outlier. Moreover, if the data set has a long-tailed distribution due to the high number of outlier data points, it will not work either. The importance of anomaly detection is due to the fact that the recognition of anomalous behavior enables decision making. For example, anomalous traffic patterns on a network could mean that sensitive data is being transmitted over the network [77]. Anomalous detection in MRI images could mean cancer detection [7]. Anomalous user behavior in payment transactions could mean credit card fraud [143]. Anomaly detection is an actionable task in contrast to outlier detection which is commonly used to locate outliers and subsequently remove them [141]. An efficient density estimation model can serve as the basis for a new proposed algorithm for anomaly and outlier detection problems capable of yielding a measure of the uncertainty given by the density prediction [24]. Lately, deep neural networks are used to predict whether a point is an outlier showing great performance in anomaly detection tasks. They work best when a large amount of data is available and can perform implicit feature engineering. For instance, Lv et al. [113] proposed a two-step anomaly detection algorithm using a variational autoencoder as the first step and a kernel density algorithm as the second step.

Anomaly detection can be solved using supervised [89, 172, 87], unsupervised [186, 124, 107], hybrid and one-class approaches [2], among others. Anomaly detection can solve problems such as medical anomaly detection [122, 23], fraud detection [201, 143], video surveillance [92, 188], time series anomaly detection [119], and industrial damage detection [119, 147]. New algorithms in deep neural networks are able to provide better approximations to very complex problems such as medical images or sequencing datasets. Moreover, voluminous datasets are increasing rapidly nowadays, where this feature is frequently encountered in a variety of applications in anomaly detection such as sensor networks; therefore, deep neural networks are a good selection for their good performance on huge datasets. In addition, neural networks can learn intrinsically complex features by avoiding manual feature generation. Also, neural networks can be mixed with classical methods like one-class support vector machines [25].

In the upcoming subsections, we present a comprehensive overview of eleven state-of-the-art anomaly detection methods that represent the most prevalent types of techniques in the field. These methods are classified into three categories: classical methods, recent shallow methods, and deep learning methods. Each algorithm takes into account the proportion of outliers in the data as a parameter for determining threshold values. Moreover, specific parameters for each algorithm are elaborated upon in their respective sections. In Chapter 4, we utilized these methods as baseline models and compared them against novel proposed methods to evaluate their effectiveness in detecting anomalies.

Classical Methods

- Isolation Forest [111]: this method profiles the anomalous data points from the beginning, assuming that some anomalous points can be found in the training data set. It divides the space into lines orthogonal to the origin. The points that require less tree breaking are considered anomalous. The method requires three hyperparameters: number of decision trees, number of samples for each decision tree and the proportion of anomalous data points.
- Minimum Covariance Determinant [156]: this algorithm embeds the data points into a hyper ellipsoid using robust estimators of the mean and covariance of the points, and then uses the Mahalanobi distance of each point to score its anomaly. This algorithm has only one hyperparameter of proportion of anomalous points in the data set.
- One Class Support Vector Machines [165]: basically, the idea is to generate a maximal margin from the origin such that all the training data points (normal points) lie within the origin and the margin. The method supposes that anomalous points are far away from normal points, i.e., they reside in the subspace generated by the maximization of the hyperplane that does not contain the origin. A Gaussian Kernel was used for all the experiments given their good properties.
- Local Outlier Factor [20]: the algorithm computes the k-nearest neighbors of each point. It then computes the local reachability density using the distance between the points and the k-nearest neighbor. Next, the algorithm compares the local reachability density of each point and computes the LOF. When the LOF values are equal or less than 1, the data point is considered an inlier, otherwise an outlier. The hyperparameter is the type of distance used in the k-nearest neighborhood algorithm and the proportion of anomalous data points.
- K-nearest neighbors (KNN) [146]: this method calculates the k-nearest neighbor distances of each point and computes a function over this distance. The three main functions to compute the distances are: the largest distance, the mean distance and the median distance. According to the distance computed function, the scientist selects an anomaly threshold given the percentage of anomalous data points. There are several hyperparameters: k nearest neighbor, distance metric to the k-nearest neighbor algorithm, distance function applied to the distance metric, and proportion of anomalous data points.

Recent Shallow Methods

- SOS [83]: the algorithm is inspired by T-SNE. It computes a dissimilarity matrix which can be the Euclidean distance. The dissimilarity matrix is then used to compute an affinity matrix using a perplexity parameter. With stochastic selection of sub-graphs,

a probability metric of being an outlier point is calculated using the link probabilities between points in the subgraph. Given the link probability, a threshold value is selected to classify outlier points from normal ones. It has two parameters: the perplexity and the threshold value.

- COPOD [104]: this algorithm uses the copula, which is a function that allows the joint multivariate distribution function to be expressed in its marginal distribution functions. The copula is calculated and then a skewness correction is calculated to decide whether to use the left tail or the right tail of the copula. Finally, a threshold value is used to determine whether a point is anomalous or normal. The only parameter considered is the anomaly threshold value.
- LODA [139]: it is an ensemble algorithm based on histograms applied to each dimension. Each histogram is projected using $\boldsymbol{w}$ random parameters. The joint probability is computed under the strong assumption of independence between the projected vectors. The logarithm function is applied to avoid the curse of dimensionality. The method has three hyperparameters, nevertheless, they are obtained automatically in the training phase.

Deep Learning Methods

- VAE [90]: the model is based on Bayesian statistics. It uses an autoencoder with stochastic variational inference to reconstruct the original sample point. With a variational lower bound reparameterization trick, the model can be trained by stochastic gradient descent. The model has several parameters like the number of neurons in each hidden layer, the number of hidden layers, the number of iterations of the optimization, the activation function of each neuron, among others. In addition, a threshold value is needed to decide whether a point is anomalous or not.
- DeepSVDD [157]: this method is based on Support Vector Data Description (SVDD), which surrounds normal points on a hypersphere in a reproducing Hilbert space. The anomalous points are assumed to be outside the hypersphere. The parameter v controls the proportion of anomalous points found by the algorithm. It has several parameters including: v, number of hidden layers, number of neurons, epochs, among others.
- LAKE [113]: it is based on the union of two anomaly detection methods. First, a variational autoencoder (VAE) is used as a dimensionality reduction algorithm and as a measure of reconstruction error. Second, a kernel density estimation (KDE) is attached to the last hidden layer of the variational autoencoder and concatenates with the reconstruction error in terms of mean squared error and cosine similarity. The KDE uses the reconstruction error and the dimensionality reduction to construct an estimate of the density of the given point. The method has several parameters such as the number of epochs of the VAE, the γ parameter of the Gaussian kernel used in

KDE, and the ratio of anomalous points.

- Adversarially Learned Anomaly Detection (ALAD) [198]: is a method that combines a generative adversarial network (GAN) framework with anomaly detection. It involves training a generator network to learn the underlying distribution of normal data and generate synthetic samples, while simultaneously training a discriminator network to distinguish between real and synthetic samples. During anomaly detection, new samples are passed through the discriminator, which assigns a score indicating the likelihood of being anomalous. A threshold is set on these scores to differentiate between normal and anomalous samples. ALAD leverages adversarial training to learn a representation of normal data and effectively detect anomalies that deviate significantly from the learned distribution.
- Anomaly Detection through Density Matrices and Fourier Features (AD-DMKDE) [22]: is a powerful method that combines adaptive Fourier features and density matrices to detect anomalies in data. By leveraging adaptive Fourier features, AD-DMKDE can effectively approximate the Gaussian kernel, capturing complex patterns and variations in the dataset. Incorporating density matrices allows AD-DMKDE to capture the distribution function of the data points, providing a comprehensive representation of the underlying structure. Through a quantum measurement-based approach and a threshold computed from the training data, AD-DMKDE assigns a normality score to each data point, indicating its likelihood of being normal. This integrated approach enables accurate and reliable anomaly detection, facilitating effective data analysis and decision-making processes.

2.2.2. Streaming Anomaly Detection

In streaming anomaly detection, the data points $\mathcal{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t\}$ arrive as a d-feature-dimension sequence. This sequence can be the sequence of transactions for a given credit card or the temperature recorded by an IOT sensor. The challenge in this configuration is that the concept of "normality" evolves over time, meaning that a point that was considered normal behavior can drift and become anomalous behavior. To solve this problem, there is a need to develop algorithms that can learn on-line with high speed and low memory consumption.

Next, we will present an explanation of incremental anomaly detection methods found in the state of the art. These twelve methods present a wide variety of techniques and are the following STORM, HS-Tree, iForestASD, RS-Hash, RCF, LODA, Kitsune, DILOF, xStream, MStream, Ex. IF and MemStream. These methods employ different techniques such as Isolation Forest, hashing, autoencoders and random cutting trees to detect anomalies in the streaming data. These methods are used as a baseline in this thesis.

- STORM [8]: A Stream Manager and a Query Manager have been proposed. The former is an indexed stream buffer whose job is to store the number of successive neighbors and the identifiers of the most recent preceding neighbors. The latter is a procedure that efficiently answers queries about whether a certain data point is an inlier or an outlier.
- HS-Tree [179]: This method constructs complete binary trees where each tree has at most $2^{h+1} - 1$ node. Each subtree is constructed by randomly selecting a feature and breaking it in half. A random perturbation of each subtree is performed to create diverse subtrees. Each point has to traverse each binary tree to capture the mass profile. Finally, a scoring function is computed using the mass function of a query data point passing through each data node.
- iForestASD [43]: This algorithm has its roots in the Isolation Forest method. The method uses a window for streaming data and is sent to the Isolation Forest. An abnormality score is calculated using the average depth of the point on each tree in the forest. If the point is normal, it is joined to the Isolation Forest.
- RS-Hash [160]: This method uses the Isolation Forest method at its roots. A tree is constructed as a sequence of randomly selected features. Anomalous points are those that are easily separated from the normal data points. With the scoring function given by the Isolation Forest, a score of anomalous sliding windows is computed to detect concept drift.
- RCF [71]: The algorithm constructs a robust randomized cutting tree (rrct) using a random selection of the dimension weighted by its range. A uniform distribution is used to cut the selected dimensions. This process is repeated n times, with n being the maximum depth. A forest is constructed using several robust randomized cutting tree. A new point is classified as an anomaly using the comparison of its insertion and deletion complexity.
- LODA [140]: The algorithm uses a set of histograms for each dimension. Each histogram is mapped to a projection space using $\boldsymbol{w}$ parameters that capture the importance of the feature. The log likelihood in the projection space is then calculated. An anomalous data point is expected to have a lower value of the log likelihood.
- Kitsune [124]: Kitsune is an algorithm that uses an ensemble of autoencoders to provide an anomaly score. Features are sent to l autoencoders of 3 layers each. The reconstruction error calculated as root mean square error (RMSE) is sent to a final 3-layer autoencoder. Finally, the RMSE of the reconstruction is calculated and used as the anomaly score.
- DILOF [128]: The method uses the k nearest neighbor information as in the Local Outlier Factor (LOF) but improves it in the case of streaming data. The algorithm has two phases: a detection phase and a summary phase. In the first phase it decides

whether a point is anomalous or not. In the second, it uses an approximation algorithm with a sampling strategy to update the memory of the k nearest neighbors.

- xStream [116]: The method uses a hash structure to reduce the dimensionality of the data points, which allows the evolution of features in the stream. After reduction, the method partitions the space into so-called half-space chains. These partitions capture the density estimate at a different granularity. The outlier score is calculated based on the density estimate score. For streaming, a previous window is used to score the point outlier.
- MStream [15]: This algorithm uses two locality-sensitive hash functions: feature hashing and record hashing. The former computes a hash for each feature in the data point. The latter computes a hash for all features simultaneously. The anomaly score is calculated using a chi-square density function. The algorithm is combined with an autoencoder to reduce the dimensionality of the data.
- Ex. IF [72]: This algorithm uses a random slope to cut the hyperplane and a random intercept. This differs from the isolation forest because the latter chooses random features and generates a random cut on that feature. The method shows better results when compared to the isolation forest for anomaly detection.
- MemStream [16]: The method proposes a nearest neighbor memory-based algorithm. Each data point passes through a shallow autoencoder to reduce the dimensionality of the original space. A memory is built using n-normal data points as initialization. Then, when a new point arrives, it is forwarded to the autoencoder and compared to the memory. An anomalous point will have a high mean square error compared to its neighbors. The memory is updated only with normal data points.

2.3. Kernel Approximation Techniques - Random Fourier Features

Kernel density estimation can be seen as a kind of kernel method. Several methods are based on kernel methods, for example, support vector machines [164] and Gaussian processes [149]. Kernel methods rely on the kernel trick, i.e., the feature space is not computed explicitly [36]. Instead, the feature space is computed intrinsically by the kernel function. Due to this kernel function, kernel methods increase their complexity with at least the square of the number of data points [162].

Increasing the speed of kernel methods is one of the challenges that researchers have studied during the last two decades [30, 185]. Several approaches are proposed in the literature: divide and conquer methods in which the original problem is broken into small subproblems that are independent and efficient to solve [10]; compute a low-rank approximation of

the kernel matrix, e.g., greedy techniques and the Nyström method [195, 192]; and random Fourier features (RFFs) [144]. The RFFs are data-independent, unlike data-dependent techniques, such as the greedy and Nyström methods. The RFFs uses Bochner's theorem, which is defined as follows:

Theorem 2.3.1. *(Bochner's Theorem) Let f be a bounded continous function on $\mathbb{R}^d$. Then, f is positive definite if and only if it is the Fourier transform of a nonnegative and finite Borel measure $p(w)$. The Fourier transform of a nonnegative Borel measure, call it $p(w)$, is*

$$f(\boldsymbol{x}) = \int_{\mathbb{R}^d} p(\boldsymbol{w}) exp(i\langle \boldsymbol{w}, \boldsymbol{x}\rangle) d\boldsymbol{w} \tag{2-3}$$

Using the Bochner's theorem, note that $f(\boldsymbol{x}) = \mathbb{E}_w[exp(i\langle \boldsymbol{w}, \boldsymbol{x}\rangle)]$; therefore, given an isotropic kernel and the Bochner's theorem, the following equality hold

$$k(x - y) = \mathbb{E}_w[\psi_w(x)\psi_w(y)^*] \tag{2-4}$$

where $\psi_w(x) = \exp(jw^*x)$ and $\psi_w(y)^* = \exp(-jw^*y)$ is the complex conjugate. Rahimi and Recht [144] used the Equation 2-4 to show that if we sample N $i.i.d.$ realizations from $\{w_n\}_{n=1}^N$, the following equation hold:

$$k(\boldsymbol{x} - \boldsymbol{y}) = \mathbb{E}_w[Z_w(\boldsymbol{x}) Z_w(\boldsymbol{y})] \tag{2-5}$$

$$\approx \frac{1}{N} \sum_{n=1}^{N} exp\left(i\boldsymbol{w}_n^T(\boldsymbol{x} - \boldsymbol{y})\right) \tag{2-6}$$

where $Z_w(x) = \sqrt{2}cos(w^*x + b)$, with $b \sim Uniform[0, 2\pi]$. Rahimi et al. [144] showed that the Equation 2-6 uniformly converges to $k(x, y)$ as the following theorem shown:

Theorem 2.3.2. *Let $\mathcal{M}$ be a compact subset of $\mathbb{R}^d$ with a diameter $diam(\mathcal{M})$. Then for the mapping ϕ_{rff} defined above, we have*

$$\Pr\left[\sup_{x,y\in\mathcal{M}} |\langle \phi_{\text{rff}}^*(x), \phi_{\text{rff}}(y)\rangle - k(x, y) \geq \epsilon|\right] \leq 2^8 \left(\frac{\sigma_p \text{diam}(\mathcal{M})}{\epsilon}\right)^2 \exp\left(-\frac{D\epsilon^2}{4(d+2)}\right) \tag{2-7}$$

where, σ_p^2 is the second momentum of the Fourier transform of k. In particular, for the Gaussian kernel $\sigma_p^2 = 2d\gamma$.

The idea of random Fourier features was extended with other characteristics. Figure **2-7** shows different approaches based on random Fourier features. Different approaches have been proposed to compute the random features for the kernel approximation based on data-independent strategies: Monte Carlo sampling [97, 197], quasi-Monte-Carlo sampling [9, 170], and quadrature rules [37]. Other approaches using a data-dependent strategy are proposed: leverage score sampling [105, 110], reweighted random features [174, 9], and kernel learning [105, 21]. Although the RFFs has demonstrated incredible power in machine learning tasks, there are three main open questions being actively investigated in state-of-the-art work: sampling schemes [97], learning procedures [106], and variance reduction [106].

2.4. Brief Summary and Perspectives

This chapter offers a thorough and in-depth review of the literature related to density estimation, covering various topics such as kernel density estimation, neural density estimation, and efficient kernel density estimation techniques. In addition, the chapter delves into kernel approximation techniques, specifically random Fourier features, and provides a detailed explanation of density matrices and their characteristics. The chapter also explores the applications of density estimation and concludes with a discussion on anomaly detection, outlier detection, and streaming anomaly detection.

The knowledge and insights gained from this literature review form the basis for the novel density matrix kernel density estimation method that we will introduce in the following chapter. This method is the foundation of our present thesis and builds upon the existing literature to provide a new and innovative approach to density estimation.

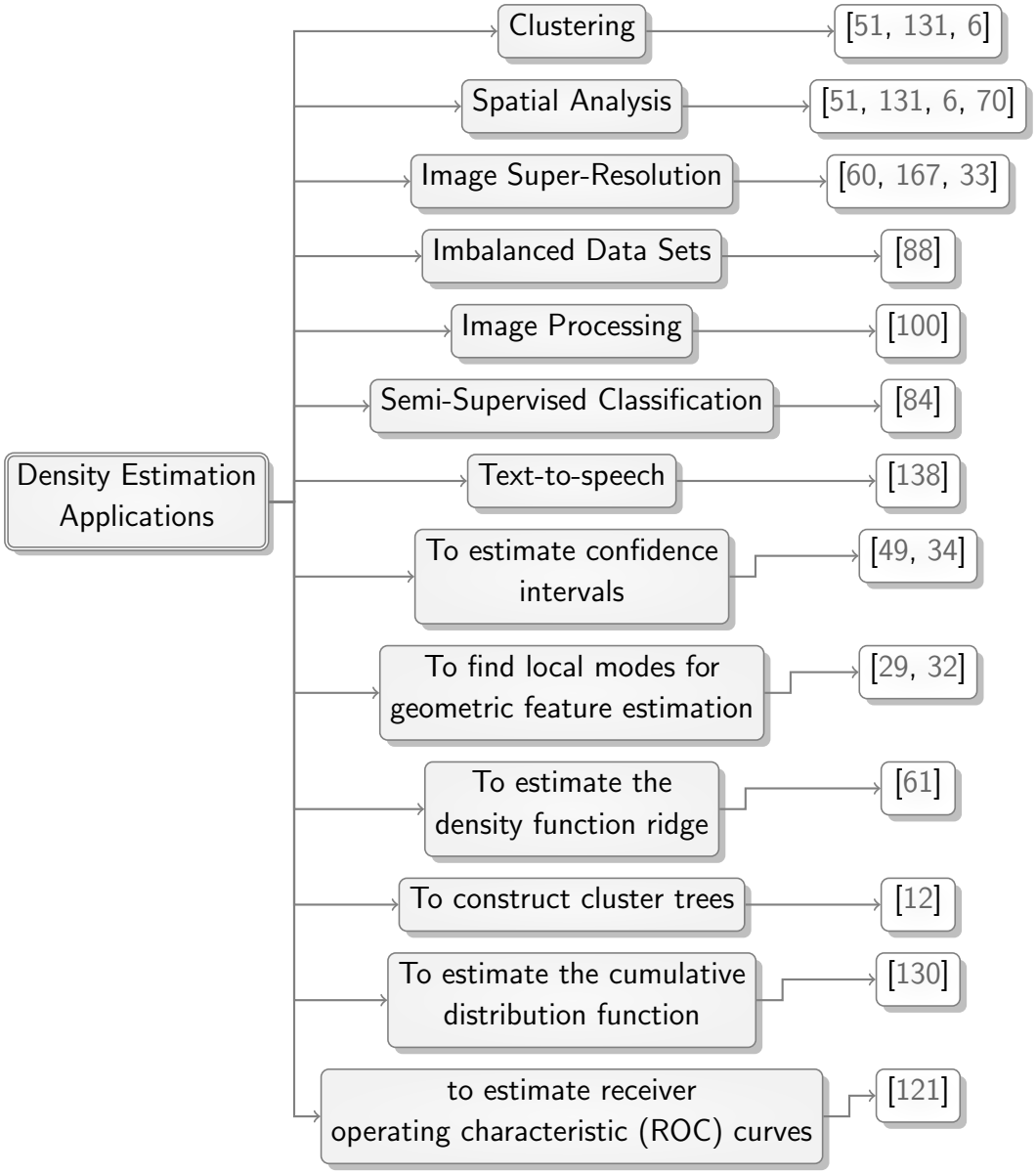

Figure 2-3.: Taxonomy map for density estimation applications.

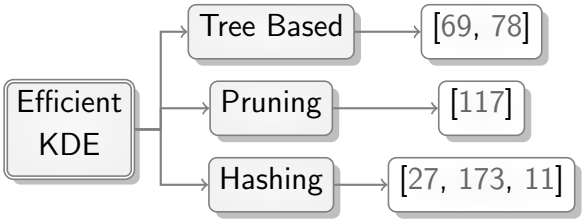

Figure 2-4.: Taxonomy map for efficient kernel density estimation literature.

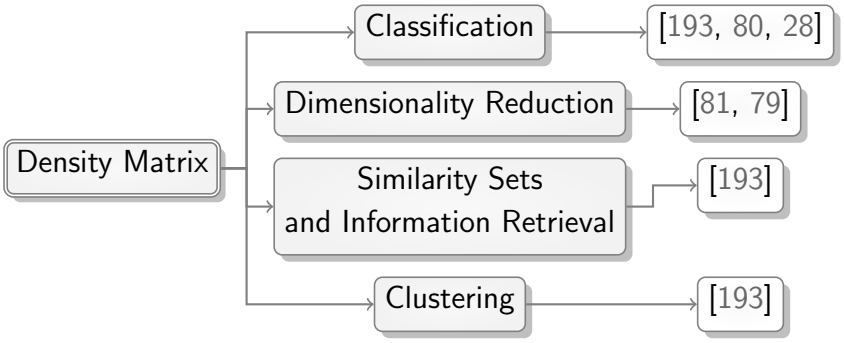

Figure 2-5.: Taxonomy map for density matrices used in machine learning literature.

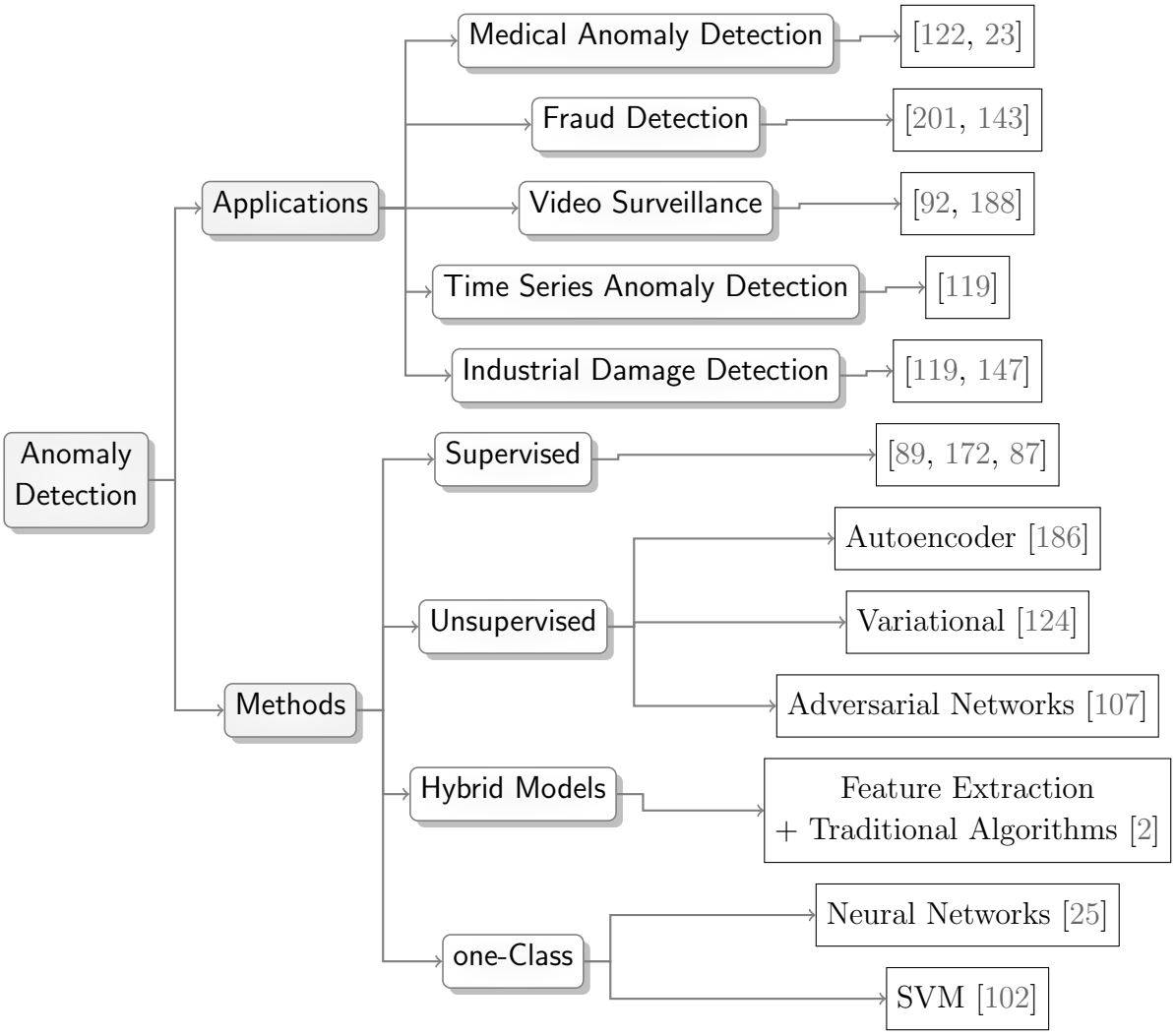

Figure 2-6.: Taxonomy map for anomaly detection. Based on the ideas exposed in [24].

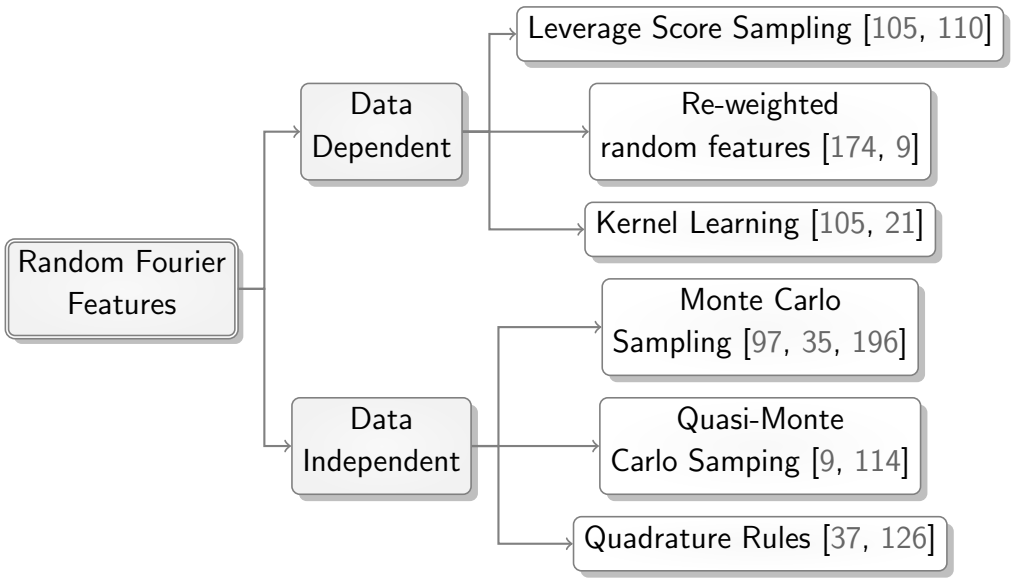

Figure 2-7.: Taxonomy map for random Fourier features approximation literature.

Part I.

Neural Density Estimation

3. DEMANDE: Density Matrix Neural Density Estimation

Density estimation is a fundamental task in statistics and machine learning that aims to estimate, from a set of samples, the probability density function of the distribution that generated them. There are different methods for addressing this problem but recently deep-neural density estimation methods have emerged as a powerful alternative. This chapter presents a novel method for neural density estimation based on density matrices and adaptive Fourier features. Density matrices are commonly used in quantum mechanics to represent the quantum state of a physical system. In this work, they are used to estimate probability densities using an operation called quantum measurement. The proposed method can be trained without optimization using an averaging operation over the samples of the training dataset. It can also be integrated with deep learning architectures and trained using gradient descent. The performance of the proposed method was evaluated on a range of synthetic and real datasets and compared with fast kernel density estimation and state-of-the-art neural density estimation methods. The results demonstrate that the proposed method achieves competitive performance while being faster and more efficient than existing methods.

The work presented in this chapter corresponds to:

- **Gallego, J.A.**, & González, F.A. (2023). DEMANDE: Density Matrix Neural Density Estimation. IEEE Access(accepted). doi: https://10.1109/ACCESS.2023.3279123. [55**?**]

3.1. Introduction

The estimation of the joint distribution, $p(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n)$, of a set of random variables is a generally important task in machine learning. This estimation of the underlying distribution has a variety of applications, for instance: density estimation, anomaly detection, non-supervised, and supervised learning. Parametric models, such as Gaussian mixture models, suffer from several problems, such as model misspecification, when the assumed parametric model does not capture the underlying probability density function, resulting in a biased or inaccurate model [18, 73]; overfitting, if the selected parametric model is too complex compared to the original data [127, 99]; computational complexity, when the model has

to be optimized using maximum likelihood estimation and there are several parameters to be adjusted [39, 123]; limited flexibility, when parametric estimation is constrained by the selected parametric family [86, 59]; data preprocessing, a parametric model is often biased or inaccurate given normalization, scaling or anomaly removal [159]. Kernel density estimation (KDE) as a non-parametric density estimation partially solved these problems. The method can approximate arbitrary density functions and its performance increases when more data points are available [155, 136]. The drawback of this method is that it requires all the training data points to make a prediction, which makes it a memory-based method [31]. A different approach to density estimation is based on neural networks with deep architecture.

This approach is called neural density estimation and one of its main advantages is that it can be integrated with other deep-learning architectures. The most representative approaches to neural density estimation are autoregressive neural models, normalizing flows, and generative adversarial models. One of the early attempts at creating autoregressive models was to stack Restricted Boltzmann Machines together, forming a deep belief network [176, 75]. However, this method is based on the calculation of the partition function. In general, this calculation for more than 30 hidden variables is forbidden [95]. In [95, 52, 14], the authors proposed other autoregressive models. They use the conditional probability rule to obtain an estimate of the distribution. While these methods have shown powerful results, they can be slow to optimize and consume significant computational resources. Furthermore, they have to compute the partition function that is not always available, which can further slowdown the model and make it more difficult to train. Normalizing Flow models were proposed in the last decades as an improvement of autoregressive flow models, whose strength is based on the change of variables [44, 152]. This change of variable can be composed in a series of differentiable and invertible transformations of a known density function, for instance, the normal distribution. The change of variables needs to preserve the volume, which imposes a constraint on the availability of basis density functions. However, these normalization flow algorithms are really difficult to tune, thus their convergence is not always guaranteed [112]. We have evaluated several of these methods in the present manuscript and their implementation software is publicly available in the manuscript repository.

In this chapter, we present a novel combination of adaptive Fourier features and density matrices that can be used to perform density estimation. Density matrices are a formalism used in quantum mechanics to represent the state of a quantum system. Its application in machine learning, and in particular in density estimation, has been limited, but initial exploration suggested that this approach could be a competitive alternative [64, 55]. In this chapter, we present a neural density estimation method that surprisingly has a deep relationship with KDE and can be seen as an approximation of it. We systematically evaluate it in different benchmark tasks. One of the main advantages of this approach is that it provides an efficient prediction method whose complexity does not depend on the number of training samples compared to KDE. Besides, its implementation as a computational graph

produces a differentiable and integrable deep learning architecture that can be optimized by gradient descent. An important characteristic of the model is that, for some applications, it is also possible to train it without using optimization, which alleviates the computational burden associated with gradient-based optimization methods. The method is systematically evaluated on different density estimation tasks and compared against state-of-the-art neural density estimation methods.

In summary, the contributions of the present work are:

- **A novel neural density estimation method** that integrates deep learning, adaptive Fourier features, and density matrices, offering an end-to-end framework for density estimation.
- **Systematic evaluations of the proposed method** in four distinct scenarios: a fast kernel density estimation comparison, an evaluation of unconditional density estimation in both low and high dimensions, and a conditional density estimation evaluation.
- **A novel adaptive Fourier features approach**, which utilizes a siamese network to enhance the random Fourier feature method.
- **A comprehensive convergence and complexity analysis** of the proposed algorithm.

Collectively, these contributions represent a significant advancement in the field of neural density estimation, with practical applications in various fields, including data analysis and modeling. The rest of the chapter is organized as follows. Section 3.2 presents the novel neural density estimation model based on adaptive Fourier features and density matrices. Section 5.3 presents the evaluation of the method in different datasets and its comparison against fast kernel density estimation and state-of-the-art neural density estimation methods. Finally, Section 5.4 presents the conclusions and ideas for future work.

All the software used to generate the results of the experiments on the present chapter are publicly available on Github: https://github.com/Joaggi/demande and Zenodo [85] with the DOI: 10.5281/zenodo.7709633. Besides, the datasets generated are also publicly available on Zenodo [57] with the DOI: 10.5281/zenodo.7822851.

3.2. Density Matrix Neural Density Estimation (DEMANDE)

In this section, we present a method for neural density estimation that can be coupled with other deep neural density estimation methods, such as convolutional neural networks, transformers, and generative adversarial networks, among others. It is derived from the density matrix which is a matrix for estimating the quantum state of a physical system. Therefore, it can be used in quantum computers and trained on them. Surprisingly, it

Table 3-1.: Notation

Random variables:	
$\boldsymbol{x}_1 \cdots \boldsymbol{x}_n$	set of identical and independent distributed variables
$\boldsymbol{x}_*$	an identical and independent distributed variables
$\boldsymbol{x}_{a:b}$	slice of the vector $\boldsymbol{x}$ from the variable a to b
Mathematical symbols:	
$O(\cdot)$	complexity in Bachmann Landau notation
$Pr[]$	probability density function
$k(\cdot,\cdot)$	kernel function
γ	parameter of the Gaussian kernel
$\odot$	Hadamard product
$\langle\cdot,\cdot\rangle$	inner product
$\mathcal{R}^d$	d-dimensional real space
$cos(\cdot)$	cosine function
$\mathcal{N}(\cdot)$	normal distribution
$Uniform$	uniform distribution
ρ	density matrix
Λ	eigen values of density matrix
V	eigenvectors of density matrix
$\phi_{\text{aff}}(\cdot)$	adaptive Fourier feature explicit projection
$\lvert\cdot\rvert$	absolute value
$diam(\mathcal{M})$	$\sup d(x,y) : x,y \in \mathcal{M}$

can also be derived as an approximation of the kernel density estimation allowing a non-memory-based method that is more efficient in prediction and can also be trained using gradient descent. A graphical explanation of Density Matrix Neural Density Estimation (DEMANDE) is shown in Fig. **3-1**. In this method, there are three different spaces: the data $\mathcal{X}$ space, the deep learning representation $\mathcal{X}'$ space, and the induced $\mathcal{F}$ space. Each training sample $x_0, \cdots, x_n$ is passed through a deep neural network generating a $\mathcal{X}'$ space. From the new space $\mathcal{X}'$, using adaptive Fourier features explained in Subsection 3.2.2, we explicitly map to the inner space ϕ induced by a Gaussian kernel $\mathcal{F}$. We can use either a summation or an optimization process to compute the ρ-density matrix using the induced training points. When a new sample arrives x_*, we can compute its density estimate. First, the point is forwarded through a deep neural network that generates a $x_*^{'}$. Second, it is explicitly induced into the embedded space induced by the Gaussian kernel. This $\mathcal{F}$ space can be generated explicitly due to adaptive Fourier features (AFF). The ρ density matrix can be computed as the sum of the outer product between the AFF representation of each $x_*^{'}$ or using a gradient descent optimization. Using the ρ density matrix calculated before, we can compute the density estimate $\hat{f}(x_*)$ of x_* by calculating a quantum measurement.

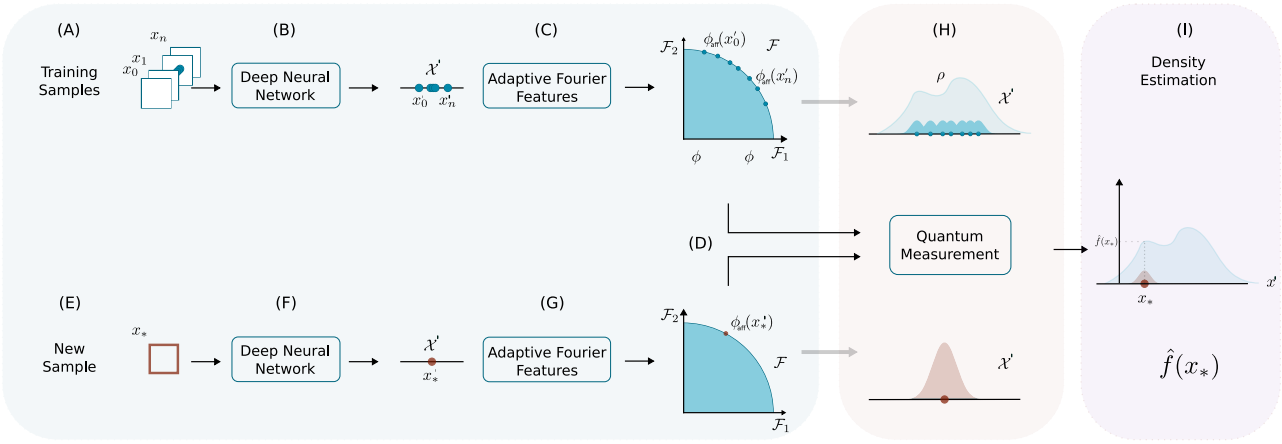

Figure 3-1.: Illustration of Density Matrix Neural Density Estimation (DEMANDE) training and prediction flow. Training samples (A) are represented (e.g., using a deep neural network) in a problem space $\mathcal{X}'$ (B). The training samples are then mapped to a feature space $\mathcal{F}$ using an adaptive Fourier-feature transformation (C). The dot product of points in $\mathcal{F}$ approximate a Gaussian kernel in the $\mathcal{X}'$ (D). The set of Gaussian functions associated to the training points collectively approximate the PDF of the data distribution on $\mathcal{X}'$ (H). A new sample, x_* (E), is represented in the space $\mathcal{X}'$ (F) and mapped to the space $\mathcal{F}$ (G). The quantum measurement module calculates the density of x'_* (I) with the help of a density matrix built from the images of the training points (H).

Each component of the proposed method is further explained in the following subsections. It is worth pointing out that while deep neural networks are often used in the more general model to map from $\mathcal{X}$-space to $\mathcal{X}'$-space, they may not always be necessary depending on the problem and data complexity. For example, in the experimental setup outlined in Subsection 3.3.1, where the problem space is relatively simple, a simpler model without a deep learning representation step may be more appropriate, in contrast, in Subsection 3.3.4 a deep representation on an image dataset is used. However, the performance of a deep or shallow model should be rigorously evaluated using appropriate metrics, to ensure that it can adequately capture the underlying patterns in the data.

3.2.1. Density Matrix Neural Density Estimation (DEMANDE)

If we start from 2-2, with k_γ representing the Gaussian kernel, we can do the following derivation [66]:

$$
\begin{aligned}
\hat{f}(\boldsymbol{x}) &= \frac{1}{M_\gamma N}\sum_{i=1}^{N} k_\gamma(\boldsymbol{x}, \boldsymbol{x_i}) \\
&= \frac{1}{M_\gamma N}\sum_{i=1}^{N} k^2_{\gamma/2}(\boldsymbol{x}, \boldsymbol{x_i}) \\
&\simeq \frac{1}{M_\gamma N}\sum_{i=1}^{N} \langle \phi_{\text{aff}}(\boldsymbol{x}), \phi_{\text{aff}}(\boldsymbol{x_i})\rangle^2 \\
&= \frac{1}{M_\gamma N}\sum_{i=1}^{N} \phi^T_{\text{aff}}(\boldsymbol{x})\phi_{\text{aff}}(\boldsymbol{x_i})\phi^T_{\text{aff}}(\boldsymbol{x_i})\phi_{\text{aff}}(\boldsymbol{x}) \\
&= \frac{1}{M_\gamma}\phi^T_{\text{aff}}(\boldsymbol{x}) \left(\frac{1}{N}\sum_{i=1}^{N} \phi_{\text{aff}}(\boldsymbol{x_i})\phi^T_{\text{aff}}(\boldsymbol{x_i})\right) \phi_{\text{aff}}(\boldsymbol{x}) \\
&= \frac{1}{M_\gamma}\phi(\boldsymbol{x})^T_{\text{aff}}\, \rho\, \phi(\boldsymbol{x})_{\text{aff}}
\end{aligned}
\tag{3-1}
$$

The matrix ρ is called a density matrix [66] and it can be seen as a summary of the training data set that can be used to estimate the density of a new sample. Density matrices are a formalism used in quantum mechanics to represent the state of a quantum system. The last line in 5-3 can be seen as an instance of the Born rule that calculates the probability of obtaining a particular state $\phi(\boldsymbol{x})_{\text{aff}}$ when measuring a quantum system whose state is described by the density matrix ρ [66]. In DEMANDE the ρ matrix is defined as:

$$
\rho = \frac{1}{N}\sum_{i=1}^{N} \bar{\phi}_{\text{aff}}(\boldsymbol{x_i})\bar{\phi}^T_{\text{aff}}(\boldsymbol{x_i}) \tag{3-2}
$$

where $\bar{\phi}_{\text{aff}}(\boldsymbol{x}) = \frac{\phi_{\text{aff}}(\boldsymbol{x})}{||\phi_{\text{aff}}(\boldsymbol{x})||}$. This guarantees that $\text{Tr}(\rho) = 1$.

In quantum mechanics, a measurement on a quantum system is represented by a Hermitian operator, also known as an observable, that corresponds to the physical quantity being measured. When a measurement is performed on a quantum system, the system is projected into an eigenstate of the observable, and the outcome of the measurement corresponds to the eigenvalue of that eigenstate. The *quantum measurement* step in Figure **3-1**, corresponds to projecting the density matrix ρ into the state $\bar{\phi}_{\text{aff}}(\boldsymbol{x})$, where $\boldsymbol{x}$ is a new sample. The magnitude of the projection is determined by the Born rule: $\text{Tr}(\rho|\bar{\phi}_{\text{aff}}(\boldsymbol{x}^*)\rangle\langle\bar{\phi}_{\text{aff}}(\boldsymbol{x}^*)|) = \text{Tr}(\langle\bar{\phi}_{\text{aff}}(\boldsymbol{x}^*)|\rho|\bar{\phi}_{\text{aff}}(\boldsymbol{x}^*)\rangle)$. This is used to calculate the density of $\boldsymbol{x}$ as:

$$
\hat{f}_\gamma(\boldsymbol{x}) \simeq \frac{1}{M_\gamma}\bar{\phi}(\boldsymbol{x})^T_{\text{aff}}\rho\bar{\phi}(\boldsymbol{x})_{\text{aff}} \tag{3-3}
$$

where M_γ is a normalizing constant. Using directly (3-3) to do an estimation of the probability of a new sample could be very inefficient since the size of the training density matrix is $O(D^2)$, where D is the dimension of the adaptive Fourier features explained in Subsection 3.2.2. To alleviate this, we can perform a low-rank factorization of ρ as follows:

$$\rho \approx V^T \Lambda V \tag{3-4}$$

where $V \in \mathbb{R}^{r \times D}$ contains as rows the eigenvectors of ρ corresponding to the r largest eigenvalues, and $\Lambda \in \mathbb{R}^{r \times r}$ is a diagonal matrix with the r largest eigenvalues. In this way, the estimation in 5-3 can be calculated as:

$$\hat{f}_\gamma(\boldsymbol{x}) \simeq \frac{1}{M_\gamma} ||\Lambda^{1/2} V \bar{\phi}(\boldsymbol{x})_{\text{aff}}||^2 \tag{3-5}$$

Equation 4-5 is the basis for the neural density estimation model. The parameters of this model are the weights W and b of the AFF mapping and the V and Λ parameters of the density estimation step. Note that this decomposition incurs an addition to the training cost of $O(D^3)$. However, during the testing phase, the quantum measurement reduces from $O(D^2)$ to $O(Dr)$. The eigenvalues obtained from a spectral decomposition of the density matrix can be interpreted as probabilities; therefore, we can discard those eigenvalues with lower probabilities. It is essential to discuss the parameter r, as it significantly influences the performance of the model. This value can be chosen as a hyperparameter using a cross-validation approach. In our experiments, we observed that selecting the minimum r value that preserves more than 90% of the likelihood is usually sufficient. However, in some cases, we obtained good results even with less than 90% probability preservation. Therefore, it is crucial to carefully analyze the trade-off between the amount of probability preserved and the computational cost associated with higher r values. After performing the decomposition, the trace property of the density matrix may no longer hold, which means that $Tr(\rho) \neq 1$. To address this issue, we can re-normalize the density matrix by dividing it by the trace of the resulting matrix. This will ensure that the trace of the density matrix equals 1 and that the probabilities of all possible outcomes add up to 1. Algorithm 1 presents the steps to produce a prediction estimate of a point x_* using 4-5 presented above. The AFF parameters are learned independently by training the neural architecture shown in Fig. **3-2** using Algorithm 4. The parameters V and Λ can be learned using two different approaches detailed in Algorithm 2 and 3.

The first approach, Algorithm 2, estimates the density matrix ρ from training data and calculates the factorization components V and Λ using spectral decomposition. An important

Algorithm 1: Density Matrix Neural Density Estimation prediction (DEMANDE)

Input: Testing data point $\{\boldsymbol{x}_*\}, \{\boldsymbol{w}, \boldsymbol{b}\}$ AFF parameters, γ-parameter, ρ or $\{\Lambda, V\}$
Output: $\hat{f}_\gamma(\boldsymbol{x}_*)$

Compute $\phi_{\text{aff}}(\boldsymbol{x}_*) = \sqrt{2}cos(\boldsymbol{w}^T\boldsymbol{x}_* + \boldsymbol{b})$
  and $\bar{\phi}_{\text{aff}}(\boldsymbol{x}_*) = \frac{\phi_{\text{aff}}(\boldsymbol{x}_*)}{||\phi_{\text{aff}}(\boldsymbol{x}_*)||}$
Estimate $\hat{f}_\gamma(\boldsymbol{x}_*) \simeq \frac{1}{M_\gamma}\bar{\phi}(\boldsymbol{x}_*)^T_{\text{aff}}\rho\bar{\phi}(\boldsymbol{x}_*)_{\text{aff}}$
  or $\hat{f}_\gamma(\boldsymbol{x}_*) \simeq \frac{1}{M_\gamma}||\Lambda^{1/2}V\bar{\phi}(\boldsymbol{x}_*)_{\text{aff}}||^2$

Algorithm 2: Density Matrix Neural Density Estimation training (DEMANDE)

Input: Training dataset $D = \{\boldsymbol{x}_i\}_{i=1,\cdots,N}, \boldsymbol{w}, \boldsymbol{b}$ AFF parameters
Output: V, Λ

$\forall i \in \{1, \cdots, N\}$ Compute
  $\phi_{\text{aff}}(\boldsymbol{x}_i) = \sqrt{2}cos(\boldsymbol{w}^T\boldsymbol{x}_i + \boldsymbol{b})$
  and $\bar{\phi}_{\text{aff}}(\boldsymbol{x}_i) = \frac{\phi_{\text{aff}}(\boldsymbol{x}_i)}{||\phi_{\text{aff}}(\boldsymbol{x}_i)||}$
Calculate $\rho = \frac{1}{N}\sum_{i=1}^{N}\bar{\phi}_{\text{aff}}(\boldsymbol{x}_i)\bar{\phi}^T_{\text{aff}}(\boldsymbol{x}_i)$
Perform a spectral decomposition of ρ
  $\rho \approx V^T\Lambda V$
return V, Λ

feature of this approach is that it does not require any optimization, just averaging the density matrices representing the training samples.

The second approach, Algorithm 3, exploits the fact that the prediction model (Algorithm 1) is a differentiable neural network that can be trained by backpropagation and gradient descent, as it is the widespread practice for neural models. This process is in general more computationally demanding than the optimization-less approach of Algorithm 2. Its main advantage is that it can be integrated with other deep architectures and trained jointly. This approach is explored in Section 3.3.4 of the experimental evaluation.

In [66], it is shown that $\hat{f}$, as defined in 5-3, uniformly converges to the Gaussian kernel Parzen's estimator $\hat{f}_\gamma$ (2-2).

Let $X = \{x_i\}_{i=1\ldots N} \subset \mathcal{M}$ a set of *iid* samples, where $\mathcal{M}$ is a compact subset of $\mathbb{R}^d$ with a diameter diam($\mathcal{M}$), then $\hat{f}$ (5-3) and $\hat{f}_\gamma$ satisfy:

Algorithm 3: Density Matrix Neural Density Estimation training using gradient descent (DEMANDE-SGD)

Input: Training dataset $D = \{\boldsymbol{x}_i\}_{i=1,\cdots,N}, \boldsymbol{w}, \boldsymbol{b}$ AFF parameters
Output: V^*, Λ^*
Apply gradient descent to find:
  $V^*, \Lambda^* = \arg\min_{V,\Lambda} \sum_{i=1}^{N} \log \hat{f}(\boldsymbol{x}_i)$
  where $\hat{f}(\boldsymbol{x}_i) = \frac{1}{M_\gamma} ||\Lambda^{1/2} V \bar{\phi}_{\text{aff}}(\boldsymbol{x})||^2$,
  $\phi_{\text{aff}}(\boldsymbol{x}_i) = \sqrt{2} cos(\boldsymbol{w}^T \boldsymbol{x}_i + \boldsymbol{b})$,
  and $\bar{\phi}_{\text{aff}}(\boldsymbol{x}_i) = \frac{\phi_{\text{aff}}(\boldsymbol{x}_i)}{||\phi_{\text{aff}}(\boldsymbol{x}_i)||}$
return V^*, Λ^*

$$\Pr\left[\sup_{x\in\mathcal{M}} |\hat{f}(x) - \hat{f}_\gamma(x)| \geq \epsilon\right] \leq 2^8 \left(\frac{\sqrt{2d\gamma}\text{diam}(\mathcal{M})}{3M_\gamma N\epsilon}\right)^2 \exp\left(-\frac{D(3M_\gamma N\epsilon)^2}{4(d+2)}\right) \tag{3-6}$$

Similarly, the optimization of the random Fourier features using backpropagation does not modify the last proposition. Parzen's estimator of the Gaussian kernel is an unbiased estimator of the true underlying density function. However, its value does not sum to one given the approximation of the random Fourier features. In arbitrary settings, it could cause problems; however, in many density applications, the exact value of the density is not used, but a comparison reference to other values.

3.2.2. Adaptive Fourier feature learning

Kernel methods are the backbone of several machine learning algorithms, such as support vector machines [74], Gaussian processes [148], kernel density estimation [136, 155], kernel principal component analysis [163], among others. A kernel calculates the dot product in an implicit feature space. This feature space is usually high-dimensional or even of infinite dimension, as it is the case for the Gaussian kernel [162]. Random Fourier features (RFF) [145] is a method that given a shift-invariant kernel, $k : X \times X \to \mathbb{R}$, calculates an explicit feature map $\phi_{\text{rff}} : X \to F$ such that $k(\boldsymbol{x}, \boldsymbol{y}) \approx \langle \phi_{\text{rff}}(\boldsymbol{x}), \phi_{\text{rff}}(\boldsymbol{y}) \rangle$. RFF are based on Bochner's theorem [145] and approximates the kernel by estimating an expected value $k(\boldsymbol{x}, \boldsymbol{y}) \simeq \mathbb{E}_{\boldsymbol{w}}[Z_{\boldsymbol{w}}(\boldsymbol{x}) Z_{\boldsymbol{w}}(\boldsymbol{y})]$ where $Z_{\boldsymbol{w}}(\boldsymbol{x}) = \sqrt{2} cos(\boldsymbol{w}^* \boldsymbol{x} + \boldsymbol{b})$, with $\boldsymbol{w} \sim \mathcal{N}(0, 1)$ and $\boldsymbol{b} \sim Uniform[0, 2\pi]$ for the Gaussian kernel. The features correspond to a set $\{\phi_{\text{rff},i}\}_{i=1\ldots D}$ with $\phi_{\text{rff},i}(\boldsymbol{x}) = \sqrt{2} cos(\boldsymbol{w}_i^* \boldsymbol{x} + \boldsymbol{b}_i)$ where $\boldsymbol{w}_i$ and $\boldsymbol{b}_i$ are sampled from the aforementioned distributions. The higher the number of features, the better the approximation.

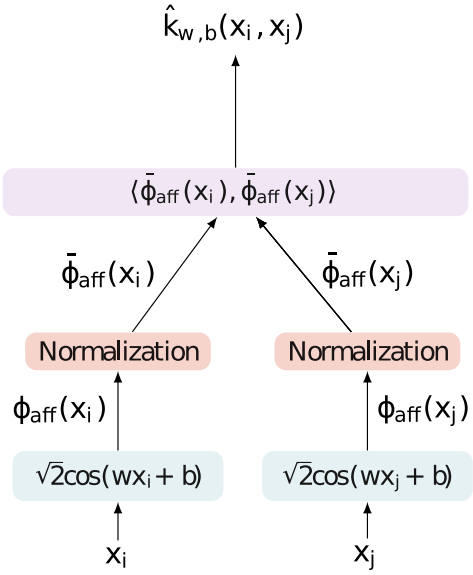

Figure 3-2.: This figure describes the learning process of adaptive Fourier features (AFF) through a siamese neural network. This involves selecting random samples, denoted as x_i and x_j, and processing them through a dot product with a non-linear cosine function used by random Fourier features. To ensure that the density matrices have a norm equal to one, a normalization step is implemented. The inner product of the normalized samples is then computed and compared with the original kernel. The loss function is defined as the discrepancy between the dot product of the projected samples and the Gaussian kernel computed between them. The optimization process involves tuning the parameters of the AFF through gradient descent to minimize the discrepancy. By using a siamese neural network, the learning process of the RFF is enhanced, allowing for efficient computation and optimization of the features.

An interesting characteristic of RFF is that they are data independent. However, it is possible to achieve a better approximation of the kernel with the same number of features if we use data to learn the features instead of the data-agnostic sampling procedure of the original RFF method. Some works have proposed data-dependent strategies to obtain better features: leverage score sampling [105, 110], reweighted random features [174, 9], and kernel learning [105, 21].

In this work, we propose a new method to learn the $\boldsymbol{w}$ and $\boldsymbol{b}$ vectors using gradient descent. We called this approach adaptive Fourier features (AFF). The method uses a siamese neural network which is shown in Fig. **3-2**. The neural network is trained by sampling pairs of samples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ from the data set and minimizing a square error loss function $L = \left(k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \hat{k}_{\boldsymbol{w},\boldsymbol{b}}(\boldsymbol{x}_i, \boldsymbol{x}_j)\right)^2$, as shown in Algorithm 4.

Algorithm 4: Adaptive Fourier Feature learning

Input: Training data set $D = \{\boldsymbol{x}_i\}_{i=1,\cdots,N}$, γ kernel bandwidth

Output: $\boldsymbol{w}, \boldsymbol{b}$ AFF parameters

Build a set $s = \{(\boldsymbol{x}'_i, \boldsymbol{x}''_i)\}_{i=1,\cdots,m}$ where $\boldsymbol{x}'_i$ and $\boldsymbol{x}''_i$ are randomly sampled from D

Apply gradient descent to find

$\boldsymbol{w}^*, \boldsymbol{b}^* = \arg\min_{\boldsymbol{w},\boldsymbol{b}} \frac{1}{m} \sum_{\boldsymbol{x}_i,\boldsymbol{x}_j \in s} (k_\gamma(\boldsymbol{x}_i, \boldsymbol{x}_j) - \hat{k}_{\boldsymbol{w},\boldsymbol{b}}(\boldsymbol{x}_i, \boldsymbol{x}_j))^2$

return $\boldsymbol{w}^*, \boldsymbol{b}^*$

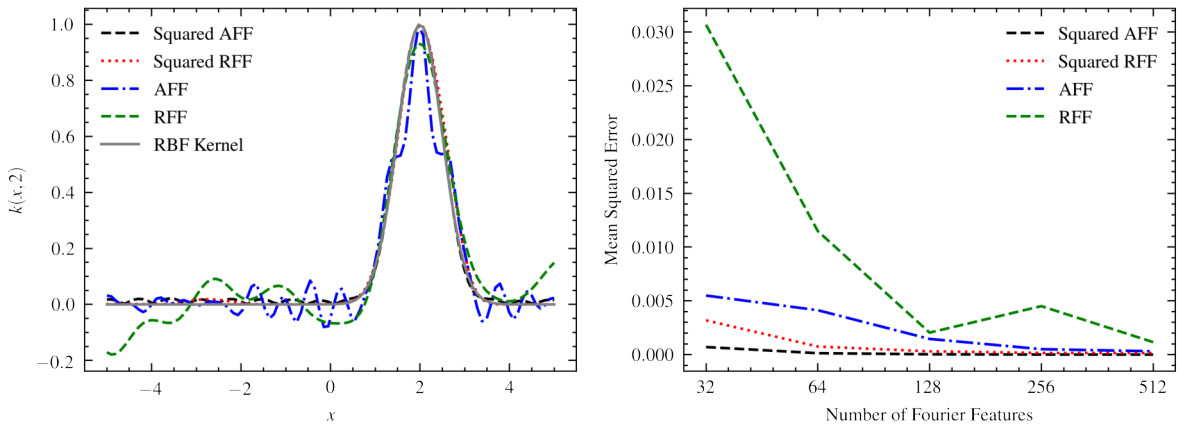

Figure 3-3.: (left) Comparison between the real Gaussian kernel centered on 2 as $k(x, y) = \exp -\gamma||x - y||^2$, the random Fourier feature, the random Fourier feature squared, the adaptive Fourier feature, and the adaptive Fourier feature squared. (Right) The mean squared error between approximation Fourier features and the real Gaussian kernel.

González et al. [66] propose to use the square of the dot product of samples represented using RFF as a better approximation of the Gaussian kernel. We follow the same approach here. Fig. **3-3** shows the comparison between the real Gaussian kernel, its approximation using RFF, squared RFF, adaptive Fourier features, and squared adaptive Fourier features. The best approximation is obtained by squared AFF, followed by AFF, squared RFF, and RFF. As shown by the right plot in Fig. **3-3**, squared AFF can reach a good approximation even with a small number of features, this has a positive impact on the efficiency of the density estimation algorithms presented in the next sections.

Table 3-2.: Complexity analysis of DEMANDE with and without spectral decomposition.

	Training	Testing
Without	$O(nDd^2 + nD^2)$	$O(mDd^2 + mD^2)$
With	$O(nDd^2 + nD^2 + D^3)$	$O(mDd^2 + mDr)$

3.2.3. Complexity Analysis

Table 3-2 shows the complexity analysis with and without spectral decomposition. In this subsection, we use n and m to denote the number of training and testing data points, respectively, while D represents the number of adaptive Fourier features, and d represents the number of features. During the training phase of the DEMANDE algorithm without decomposition, the time complexity is dominated by the dot product calculation between each data point and the adaptive Fourier features. Specifically, the time complexity is proportional to $O(nDd^2)$. Additionally, a dot product is calculated between the resulting vector and the density matrix, with a time complexity proportional to $O(nD^2)$. This gives an overall training time complexity of $O(nDd^2 + nD^2)$. During the testing phase, the time complexity is determined by the dot product calculation between each data point and the adaptive Fourier features, as well as the time complexity of the quantum measurement. Therefore, the overall testing time complexity is proportional to $O(mDd^2 + mD^2)$. For the DEMANDE with decomposition variant, the training and testing time complexity is $O(nDd^2 + nD^2 + D^3)$ and $O(mDd^2 + mDr)$, respectively. The memory footprint of DEMANDE is determined by the need to maintain a density matrix, whose size is proportional to $O(D^2)$. Additionally, it stores the weights of the adaptive Fourier features, whose size is proportional to $O(dD)$.

3.3. Experimental Evaluation

Four different experiments are presented in this section. The first one is a comparison of the novels DEMANDE and DEMANDE-SGD methods against fast kernel density estimation methods. We compare their efficiency according to the mean average error and their efficacy in terms of milliseconds spent making the inference. The second and third experiments compare the novel methods against state-of-the-art neural density estimation methods in low and high dimensions respectively. The Spearman correlation and mean average error between the underlying density function and the density estimation obtained by each method are reported. Finally, the last experiment is a comparison between the proposed methods against neural density estimation in a conditional density estimation setting using synthetic image datasets. The accuracy is reported to both DEMANDE and DEMANDE-SGD methods. All the experiments were running using a computer with 8 cores Intel 12 2.2 GHz, 64 GB of ram, and a GPU NVIDIA 3080 RTX.

1 : $p(x_1, x_2) = \mathcal{N}(\boldsymbol{x}\|\mu_1, \Sigma_1)$,	$\boldsymbol{\mu}_1 = (1, -1), \quad \text{diag}(\boldsymbol{\Sigma}_1) = (1, 2)$
2 : $p(x_1, x_2) = \mathcal{N}(x_2\|0, 4)\mathcal{N}(x_1\|\frac{1}{4}x_2^2, 1)$	
3 : $p(x_1, x_2) = 0.5 \cdot \mathcal{N}(\boldsymbol{x}\|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) + 0.5 \cdot \mathcal{N}(\boldsymbol{x}\|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$,	$\boldsymbol{\mu}_1 = (1, -1), \quad \boldsymbol{\mu}_2 = (-2, 2),$ $\text{diag}(\boldsymbol{\Sigma}_1) = (1, 2), \quad \text{diag}(\boldsymbol{\Sigma}_2) = (2, 1)$
4 : $p(x_1, x_2) = (1/8)\sum_{i=1}^{8} \mathcal{N}(\boldsymbol{\mu}_{(4,i)}, \boldsymbol{\Sigma}_4)$,	$\boldsymbol{\mu}_{(4,i)} = (3\cos(\pi \cdot i/4), 3\sin(\pi \cdot i/4)),$ $\sum_{(4)} = \begin{pmatrix} \cos^2\frac{\pi}{4} + 0.16^2\sin^2\frac{\pi}{4} & (1-0.16^2)\sin\frac{\pi}{4}\cos\left(\frac{\pi}{4}\right) \\ (1-0.16^2)\sin\frac{\pi}{4}\cos\left(\frac{\pi}{4}\right) & \sin^2\frac{\pi}{4} + 0.16^2\cos^2\frac{\pi}{4} \end{pmatrix}$
5 : $p(x_1) = \mathcal{N}(r\sin(2r), 0.4^2)$, $p(x_2) = \mathcal{N}(r\cos(2r), 0.4^2)$,	$r \sim \mathcal{U}(0, 2\pi)$
6 : $\frac{1}{2}(\frac{\|z-2\|}{0.4})^2 - \ln\left(e^{-\frac{1}{2}\left[\frac{z_1-2}{0.6}\right]^2} + e^{-\frac{1}{2}\left[\frac{z_1+2}{0.6}\right]^2}\right)$	
7 : $\frac{1}{2}\left[\frac{z_2 - w_1(z)}{0.6}\right]^2$	
8 : $-\ln\left(e^{-\frac{1}{2}\left[\frac{z_2-w_1(z)}{0.35}\right]^2} + e^{-\frac{1}{2}\left[\frac{z_2-w_1(z)+w_2(z)}{0.35}\right]^2}\right)$,	$w_1(z) = \sin(\frac{2\pi z}{4}), \quad w_2(z) = 3e^{-\frac{1}{2}\left[\frac{z-1}{0.6}\right]^2}$
9 : $-\ln\left(e^{-\frac{1}{2}\left[\frac{z_2-w_1(z)}{0.4}\right]^2} + e^{-\frac{1}{2}\left[\frac{z_2-w_1(z)+w_3(z)}{0.35}\right]^2}\right)$,	$w_3(z) = 3\sigma\left(\frac{z-1}{0.3}\right), \quad \sigma(x) = 1/(1+e^{-x})$

Figure 3-4.: We analyze nine synthetic datasets, each of which is labeled based on its probability and energy function. These datasets include: (1) Multivariate normal distribution, (2) Arc distribution, (3) Mixture Gaussian distribution, (4) Star, (5) Swiss Roll, (6) Potential 1, (7) Potential 2, (8) Potential 3, and (9) Potential 4.

3.3.1. Fast Kernel Density Estimation

In this section, we systematically assess the performance of DEMANDE-SGD on various synthetic data sets and compare it with kernel density estimation approximation methods.

Data sets and experimental setup

Table 3-4 shows nine datasets that were used in this experiment and Subsection 3.3.2. The density estimation function of each synthetic where each number represents the following distributions: (1). Multivariate normal distribution, (2). Arc distribution [135], (3). Mixture Gaussian distribution, (4). Star [112], (5). Swiss Roll [112], (6). Potential 1, (7). Potential 2, (8). Potential 3, and (9). Potential 4 [152]. For each dataset, we generated 100 000 training data points and 50 000 testing data points. In this experiment, we use six datasets: Arc (2), Potential 1 to 4 (6-9), and a mixture of Gaussian distribution (3).

We tested DEMANDE against several approximate kernel density estimation methods in

this experiment. For this comparison, we use: (1) a raw implementation of kernel density estimation using NumPy (RAWKDKE, (2) a naive implementation of kernel density estimation using the KDE.py library (NAIVEKDE) [175], (3) tree-based kernel density estimation (TREEKDE), (4) kernel density estimation using a k-dimensional tree (KDEKDT), and (5) kernel density using a ball tree (KDEBT). For more information on NAIVEKDE, KDEKDT, and KDEBT see 2.1.4. Due to the limited reproducibility of the hashing-based algorithm implementations, we did not compare our method with theirs. The Gaussian kernel was used for all experiments. Two kinds of experiments have been carried out. The first was an evaluation of the accuracy of each algorithm on each data set. In the second, we evaluated the time it took to make a new density prediction on the training set. Each run was performed with different training set sizes, using a scale of 10^i where $i \in \{1, 2, 3, 4, 5\}$, and we set the test set to 10^4 test examples. The spread parameter was found using a 5-fold cross-validation for each data set and each training size in a logarithmic scale $\gamma \in \{2^{-20}, \cdots, 2^{20}\}$. The optimal number of random Fourier features used by DEMANDE-SGD was searched in the set $\{50, 100, 500, 1000\}$. After finding the best hyperparameter, several attempts were performed for each algorithm in each data set.

We measure the efficacy of each algorithm on each data set using the L_1-error also known as *average error*. Some advantages over L_2-error are outlined in [41]. The L_1-error is defined over n samples by the following equation: $MAE = \frac{1}{n}\sum_{i=1}^{n} |\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x})|$. In [41], the author shows that the loss L_1 loss $\int |\hat{f} - f|$ is invariant under monotone transformations of the coordinate axes and points out that it is related to the maximum error made if we were estimating the probabilities of all Borel sets of $\hat{f}$ and f respectively. The efficiency has been evaluated using the CPU time in milliseconds (ms), which defines the amount of time it takes for the central processing unit (CPU) to execute its processing instructions to compute the evaluation request. We used the built-in **time** package in the Python programming language to measure the elapsed time in prediction time for each algorithm.

Results and discussion

Fig. **3-5** shows the comparison of the efficacy measure with the mean average error (MAE) of each algorithm on six synthetic data sets. The MAE of **DEMANDE** and **DEMANDE-SGD** is close to other KDE approximation methods in ARC, Potential 1, Potential3, Potential 4, and 2d mixture. In Arc, however, their performance does not improve after 10^3 training data points. In Potential 2, both **DEMANDE** and **DEMANDE-SGD** are better than the KDE approximation methods. On 2D Mixture, **DEMANDE** and **DEMANDE-SGD** outperform other KDE approximation methods. Fig. **3-6** shows the comparison of the efficiency measure in time taken of the central processing unit (CPU) of approximation methods of KDE. All approximation methods, except **DEMANDE** and **DEMANDE-SGD**, increase their prediction time when the number of points increases. However, it is observed that **DEMANDE** does not increase linearly like the other methods. **DEMANDE-SGD** has a

larger initial footprint, but it remains constant as the number of data points increases. If we evaluate it with more than 10^5 points, we would expect all methods to exceed the time consumed by **DEMANDE-SGD**.

3.3.2. Unconditional Density Estimation

In this subsection, we proposed an experiment based on synthetically generated data to evaluate the performance of various neural density estimation methods and compare them with DEMANDE and DEMANDE-SGD. It should be noted that the underlying density function is known for each data set used in the current experiment.

Data sets and experimental setup

The datasets were defined above in Subsection 3.3.1. Besides, we assessed four baseline neural flow algorithms: (1). Masked Autoregressive Flow for Density Estimation [135], (2). Inverse Autoregressive Flow [91] and (3). Planar Flow [152]. The novel DEMANDE and state-of-the-art methods were tunning using Adam Optimizer with polynomial decay. To select the hyperparameters, a cross-validation method was used together with a random search. For each data set and algorithm, we tested 30 different combinations of hyperparameters. For the validation phase, we used 40 000 training data points and 1 000 validation data points. The batch size was set to 64. The initial learning rate for the polynomial decay was searched in the interval $[10^{-5}, 10^{-1}]$; the final learning rate was set to 10^{-5}. For each neural flow algorithm, the following setup was used: the hidden shape was searched between 20 and 1000; the number of layers was searched in the list $2, \cdots, 24$. For Neural Splines: the number of bins was searched between three and eleven; the b-interval was searched between three and seven for each dimension. For DEMANDE: AFF dimension was explored between 250 and 2 000, sigma was explored between 2^{-20} and 2^{20}, and the training of the AFF was performed using 10000 pairs of different points. In the case of DEMANDE-SGD, we selected the number of eigen-components in the list 0, 0.1, 0.5, 1 where each number represents a percentage of the number of AFF. Moreover, we assessed random initialization and DEMANDE initialization for the density matrix of DEMANDE-SGD.

We used two metrics to evaluate the performance of each algorithm. Those metrics are the mean average error (MAE) and Spearman's rank coefficient. For each metric, we compute the metrics using the estimate of each algorithm and the real probability. MAE is computed as $1/n \cdot \sum_i^n |p(\boldsymbol{x_i}) - \hat{p}(\boldsymbol{x_i})|$, where n is the number of data points, $p(\boldsymbol{x_i})$ is the real probability density, $\hat{p}(\boldsymbol{x_i})$ is the estimated probability density given by the method, and $|\cdot|$ is the absolute value function.

Table 3-3.: Spearman correlation for Unconditional Density Estimation Experiment.

Dataset	DEMANDE	DEMANDE-SGD	MADE	Inverse Maf	Planar Flow	Neural Splines
arc	0.9943	0.9773	0.9993	**0.9978**	0.7395	0.9650
bimodal_l	**0.9954**	0.9941	0.9924	0.9918	0.9443	0.9892
binomial	**0.9991**	0.9986	0.9991	**0.9991**	0.9866	0.9988
potential_1	**0.9902**	0.9839	0.9516	0.9667	0.8637	0.9723
potential_2	**0.9072**	0.8112	0.8478	0.8374	0.5540	0.8175
potential_3	**0.8665**	0.8558	0.8059	0.8113	0.6280	0.8125
potential_4	0.8928	**0.9094**	0.8768	0.8791	0.6034	0.8816
star_eight	**0.9871**	0.9348	0.7710	0.7349	0.5940	0.9042
swiss_roll	**0.9936**	0.9686	0.9713	0.9475	0.7753	0.9451

Table 3-4.: Mean average error for Unconditional Density Estimation Experiment.

Dataset	DEMANDE	DEMANDE-SGD	MADE	Inverse Maf	Planar Flow	Neural Splines
arc	0.0052	0.0179	**0.0004**	0.0007	0.0197	0.0057
bimodal_l	0.0014	0.0152	**0.0009**	0.0010	0.0029	0.0012
binomial	0.0286	0.0266	**0.0008**	**0.0008**	0.0032	0.0010
potential_1	0.0089	0.0735	0.0104	0.0097	0.0235	**0.0069**
potential_2	**0.0273**	0.0548	0.0475	0.0514	0.0619	0.0520
potential_3	0.0321	0.0231	0.0225	**0.0218**	0.0340	0.0219
potential_4	0.0481	0.0377	0.0312	0.0331	0.0451	**0.0308**
star_eight	**0.0102**	0.0161	0.0196	0.0232	0.0388	0.0172
swiss_roll	**0.0028**	0.0344	0.0040	0.0045	0.0223	0.0051

Results and discussion

Table 3-3 presents the Spearman correlation results obtained from six neural density estimation algorithms on nine synthetic data sets. The algorithms' performance on the ARC, Bimodal, and Binomial data sets is comparable, except for **Planar Flow**, which performs inferiorly. **DEMANDE** and **DEMANDE-SGD** outperform the other methods in Potentials 1 to 4, with **DEMANDE-SGD** showing better results than **DEMANDE** at Potential 4, a difficult distribution function. **DEMANDE** has superior performance in Star Eight and Swiss Roll compared to the other algorithms, while **Planar Flow** is the worst-performing algorithm. **Neural Splines** consistently produces satisfactory results.

Table 3-4 reports the mean average error (MAE) between the actual and estimated density functions for the nine synthetic data sets. **MADE** exhibits the best MAE performance in the ARC, Bimodal, and Binomial data sets. **DEMANDE** performs best in Potential 2, Star Eight, and Swiss Roll. Neural Splines is the best-performing algorithm in Potentials 1 and 4. **DEMANDE-SGD** has comparable MAE results to **DEMANDE**, with its performance

being similar to the best-performing algorithms in Potentials 1 to 4, Star Eight, and Swiss Roll.

Fig. 3-8 depicts the density estimate results of the six algorithms on the nine two-dimensional synthetic data sets. **DEMANDE** exhibits the best results in all data sets, including the most challenging Potentials 3 and 4. **DEMANDE-SGD** produces comparable results to **DEMANDE** in seven out of nine data sets, with different results in Potentials 4 and Swiss Roll. Fig. 3-9 presents the comparison between the real density and the density estimates obtained by the six algorithms on the nine synthetic data sets. **DEMANDE** shows good results with lower scatter in all data sets, albeit with slightly more dispersion than MADE in the first and second data sets. In the Swiss Roll, **DEMANDE** exhibits the best performance among all algorithms. **DEMANDE-SGD** shows a tendency to overestimate low-density points due to its log-likelihood optimization approach.

DEMANDE is not the best one in terms of MAE. However, it is worth noting that for the majority of real applications such as classification, anomaly detection, and regression, the real value of the density function is not important. The most important property is being able to differentiate between low-density areas and high-density areas. This property is intrinsically captured by the Spearman correlation.

3.3.3. Unconditional Random Density Estimation in Higher Dimensions

Classical methods for density estimation, such as Gaussian Mixture Models [101] or Kernel Density Estimation [136, 155], suffer from higher dimensions due to the curse of dimensionality. Several papers claim that normalizing flow methods can solve this problem. Related to DEMANDE, it potentially can inherit the problems from kernel density estimation; however, as shown in the following experiment, it can deal with higher dimensions and obtain good density estimates. In this subsection, we propose an experiment to show the robustness of DEMANDE methods when the number of dimensions varies and increased.

Data sets and experimental setup

In this experiment, we generate data points from a mixture of random independent Gaussian distributions. For n dimensions, we generated $10 \cdot n$ Gaussian distributions for $n \in [1, \cdots, 10]$. The μ_i parameters of the Gaussians were generated from a uniform distribution $\mu_i \in (0, 1)^n$. The covariance matrices, Σ_i, were generated with a vector of uniform values as eigenvalues. With this vector, we compute the algorithm proposed by Davies et al. [38] to generate a random correlation matrix. Using both the centroid and the covariance matrix, we sampled an equal number of points from each Gaussian distribution to obtain 40 000 training data points and 10 000 testing data points. Fig. 3-7 shows an example in two dimensions of the randomly generated samples.

Results and discussion

Fig. 3-7 shows the results obtained by each algorithm in the random Gaussian mixture model synthetic data set. **Planar flow** has the worst performance among the six algorithms. **Made** and **Inverse Maf** have a similar behavior. They start at nearly 0.92 of Spearman's correlation and decrease with higher dimensions. A Spearman correlation of almost 0.76 was obtained for ten dimensions. **Neural Splines** starts better than **Made** and **Inverse Maf**; however, its performance decreases with more dimension. It shows the worst performance in ten dimensions compared to **Made** and **Inverse Maf** obtaining 0.61 of Sperman's correlation. **DEMANDE** start with nearly 1.0 Spearman's correlation and drop slightly with more dimensions. However, its performance is the best among all the neural density estimation methods. **DEMANDE-SGD** has similar behavior to **DEMANDE** but with lower Spearman's correlation.

3.3.4. Conditional Density Estimation

Density estimation can be used as a conditional density algorithm (conditioned on class values). In this subsection, we present a systematic evaluation of the conditional density estimation obtained by DEMANDE and compared it against state-of-the-art neural flow methods in two frequently used benchmark image data sets MNIST and CIFAR.

Data sets and experimental setup

Table 3-5.: Data sets used for conditional density estimation.

Data set	Attributes	Classes	Train-Test
Mnist	784	10	60000-10000
Cifar	3072	10	60000-10000

Two benchmark image datasets were utilized in this study. The specific characteristics of these datasets are presented in Table 3-5. The DEMANDE algorithm was trained using a conditional Bayesian density estimation approach and the ADAM algorithm was utilized as the stochastic optimization gradient algorithm. To establish a baseline for comparison, we compared DEMANDE against three existing state-of-the-art normalizing flow algorithms (MADE [62], RealNVP [44], MAF [135]), as well as RoundTrip [112], a normalizing flow generative adversarial algorithm. For further details, refer to Subsection 2.1.2. Each algorithm was assessed by computing the conditional density over the test image dataset and by maximizing the posterior probability, conditioned on the class label. Additionally, we designed a LeNet architecture [98] as a feature extraction method and coupled it with

DEMANDE-SGD as the density estimation method. The LeNet architecture comprised two sequential convolutional layers, with a kernel size of 5, the same padding, and a ReLU activation function. The first and second convolutional layers consisted of 20 and 50 filters, respectively. The third layer was a fully connected layer, comprising 84 neurons. This fully connected layer was integrated with the DEMANDE layer. The AFF layer was trained using 1000 adaptive Fourier features. Additionally, the AFF layer was solely optimized using the algorithm described in Section 3.2, and not in the **DEMANDE-SGD** optimization steps, to ensure a fair comparison. Thus, all AFF weights were set as untrainable in the **DEMANDE-SGD** optimization step of the neural network.

For each dataset, we performed hyperparameter optimization using a cross-validation methodology with 30 randomly generated settings. To tune the γ parameter of DEMANDE, we computed the mean distance between pairs of points in each dataset and selected an appropriate value for $\gamma = \frac{1}{2\sigma^2}$. The number of adaptive Fourier features was set to 1000 for each DEMANDE algorithm. The learning rate was selected from the interval (0, 0.001]. The number of eigen-components was chosen from the list 0, 0.1, 0.5, 1, where each number represented a percentage of the total number of AFF. The mean accuracy from ten experiments was reported.

Results and discussion

Table 3-6.: Accuracy results in conditional density estimation experiment using neural density estimation methods.

Algorithm	MNIST	CIFAR-10
MADE	0.911	0.358
RealNVP	0.744	0.309
MAF	0.926	0.295
RoundTrip (CNN)	0.983	0.427
DEMANDE	0.811	0.271
DEMANDE-SGD	0.952	0.484
LENET DEMANDE-SGD	**0.989**	**0.628**

Table 3-6 shows the results obtained by each algorithm in the density estimation task for MNIST and CIFAR-10 image datasets. It can be seen that without convolutional neural networks **DEMANDE**-SGD is better than all other neural flow methods except RoundTrip on Mnist, and it is the best on CIFAR-10. When we use convolutional layers with DEMANDE-SGD its performance outperforms the other methods on both image datasets. The method can be used not only as a density estimation algorithm but also as a conditional density estimation method. Future experiments on state-of-the-art image datasets, such as ImageNet

or COCO, may be conducted to further explore the capabilities of the proposed method. However, the current experiment aimed to demonstrate the potential of the method in a similar context, such as density estimation.

3.3.5. DEMANDE versus DEMANDE-SGD discussion

It is worth noting that DEMANDE exhibits superior performance in pure density estimation tasks, as demonstrated in Subsection 3.3.1 and Subsection 3.3.2. However, in proxy tasks, such as the one presented in Subsection 3.3.4, DEMANDE-SGD outperforms DEMANDE. Conventionally, the optimization process aims to derive overall better results, and we expected to see the same in the proposed method. Nevertheless, our experiments revealed that the DEMANDE-SGD model tends to overestimate dense regions and underestimate sparse regions where data resides, leading to inaccurate probability density function estimation. However, we also found that this property of the DEMANDE-SGD model can be exploited to obtain better results in scenarios such as conditional density estimation.

3.4. Conclusion

In this chapter, we have introduced a novel approach to neural density estimation based on the combination of density matrices and adaptive Fourier features, which can be seamlessly integrated with other deep learning architectures. The proposed method represents a significant improvement over kernel density estimation, as it overcomes the limitations associated with high-dimensional data and provides an efficient prediction method whose computational complexity is independent of the number of training samples. Our approach has been rigorously evaluated on a variety of synthetic and real-world datasets and compared against fast kernel density estimation and state-of-the-art neural density estimation methods. Our results demonstrate the competitive performance of our proposed method, highlighting its potential to offer a powerful alternative for density estimation tasks and to be applied in a wide range of statistical and machine learning applications. Future work could involve exploring the extension of our proposed approach to different types of data or developing hybrid models that combine our approach with other density estimation techniques, to further improve the accuracy and scalability of the proposed method.

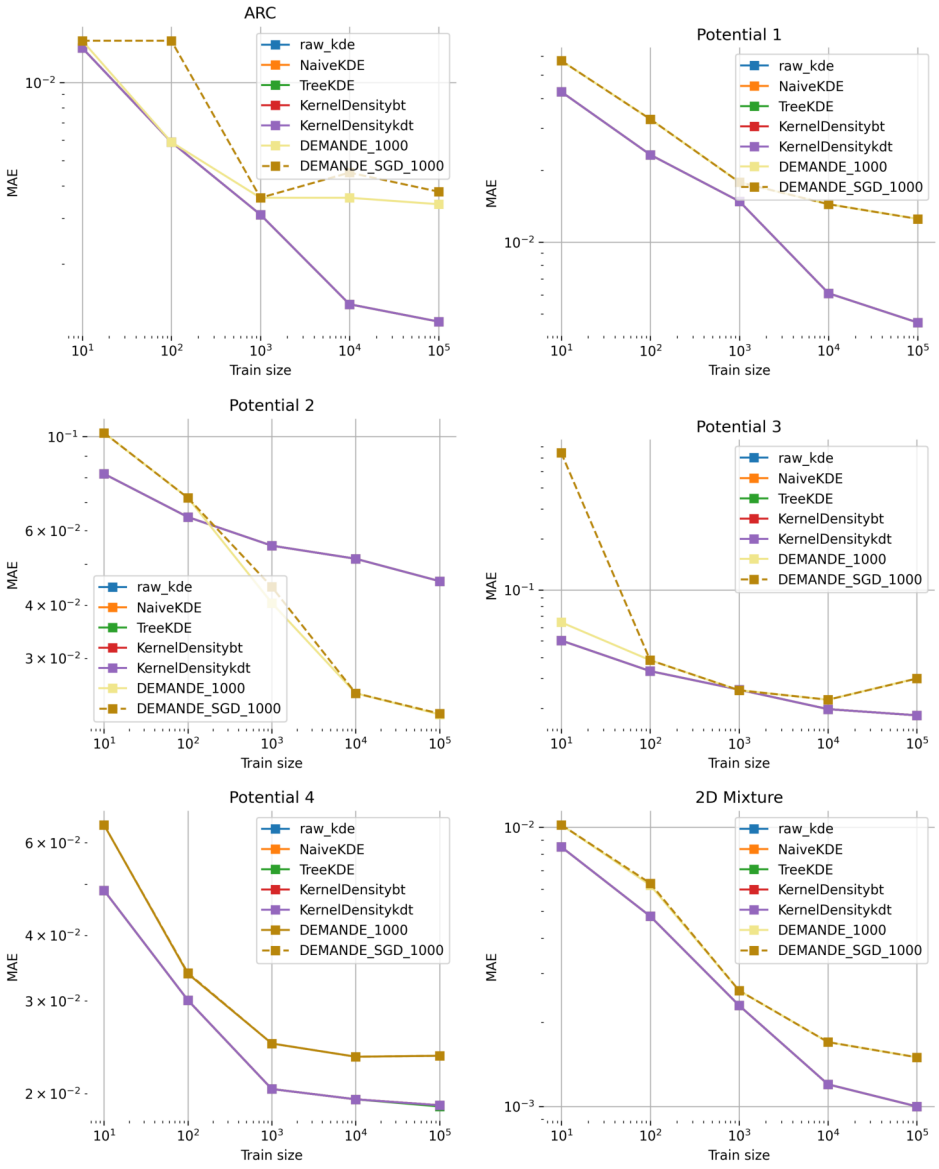

Figure 3-5.: Comparison of the efficacy of each algorithm on six synthetic data sets. The x-axis is a logarithmic scale of 10^i where $i \in \{1, \cdots, 5\}$. The y-axis represents the mean average error between the prediction of the algorithm and the true density.

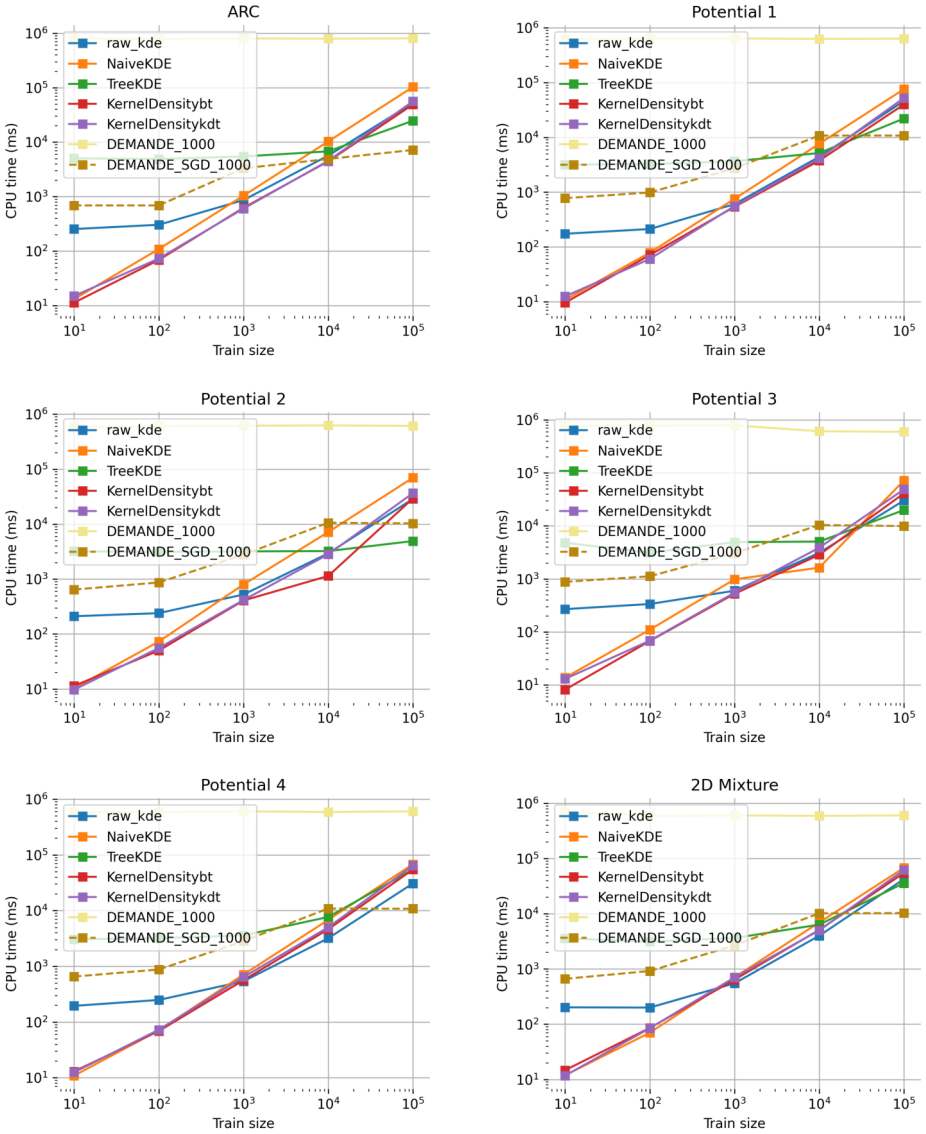

Figure 3-6.: Comparison of the efficiency of each algorithm on six synthetic data sets. The x-axis is a logarithmic scale of 10^i where $i \in \{1, \cdots, 5\}$. The y-axis represents the time consumed by each algorithm in milliseconds used by the central processing unit (CPU).

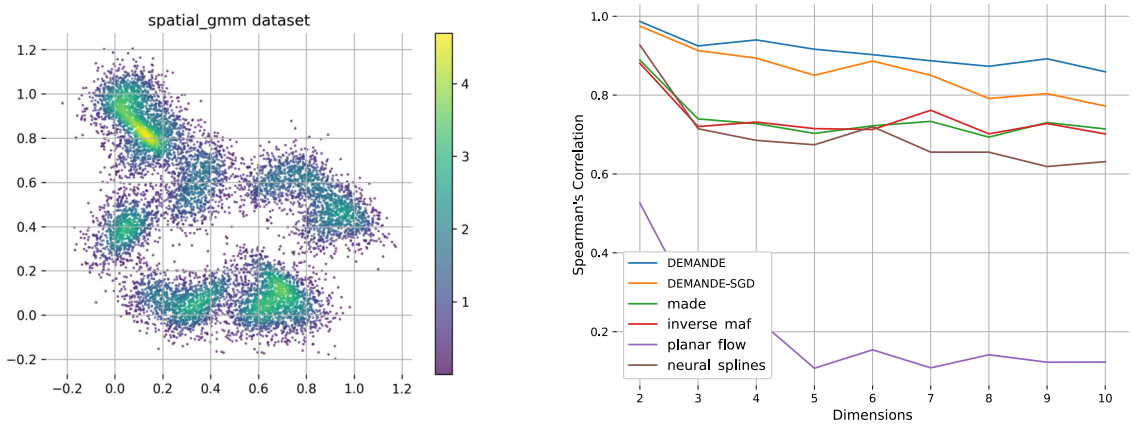

Figure 3-7.: (Top) Randomly generated samples, (bottom) experimental results of unconditional random density estimation in higher dimensions. The x-axis is the dimensions, and the y-axis is the Spearman's correlation.

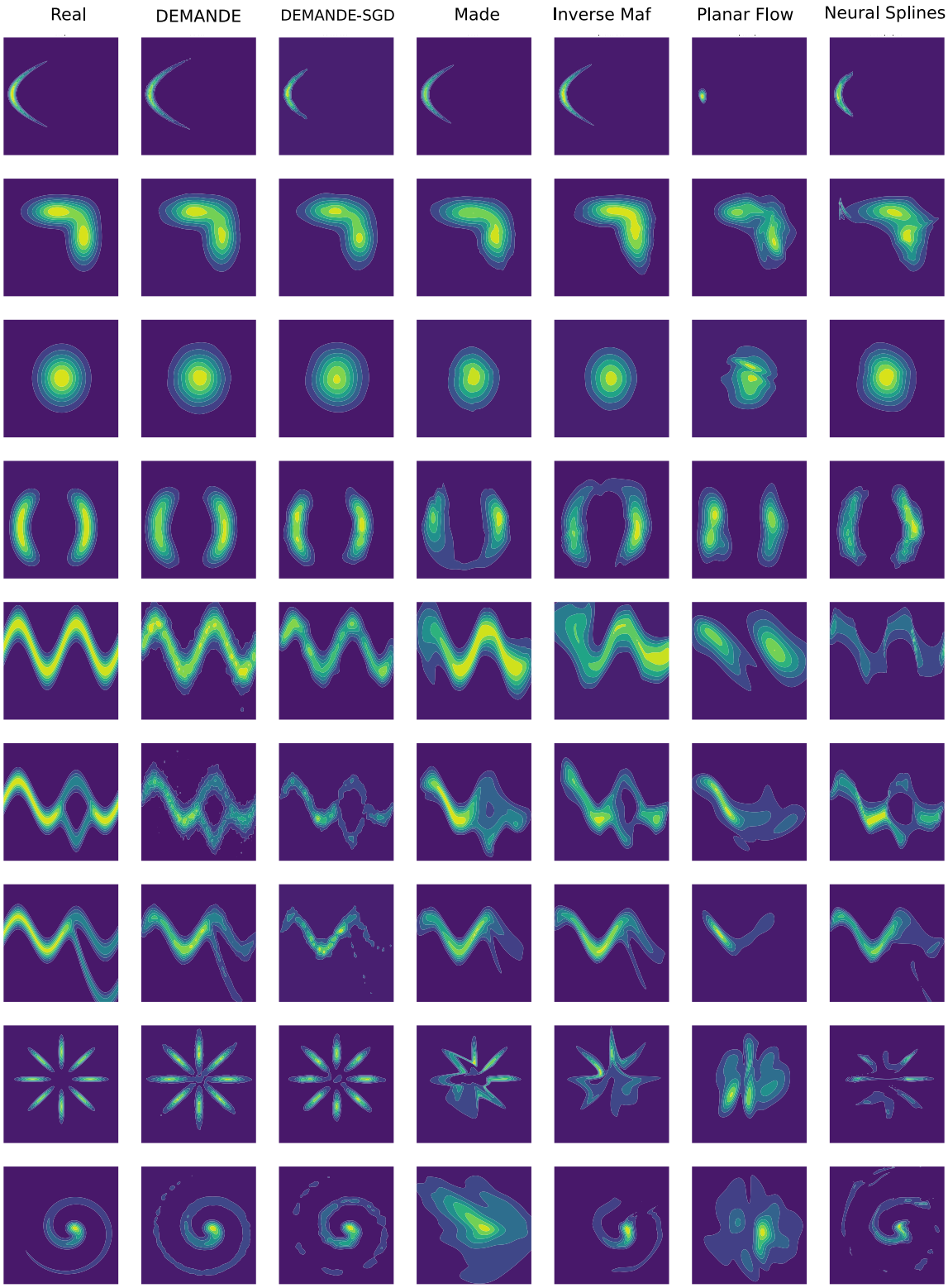

Figure 3-8.: Each graph represents the density estimate obtained by a specific algorithm on a two-dimensional data set. From left to right the algorithms are Real Density, DEMANDE, DEMANDE-SGD, Made, Inverse Maf, Planar Flow, and Neural Splines. From top to bottom, the data sets are Arc, Bimodal Gaussian, two-dimensional Gaussian distribution, Potential 1-4, Star, and Swizz Roll.

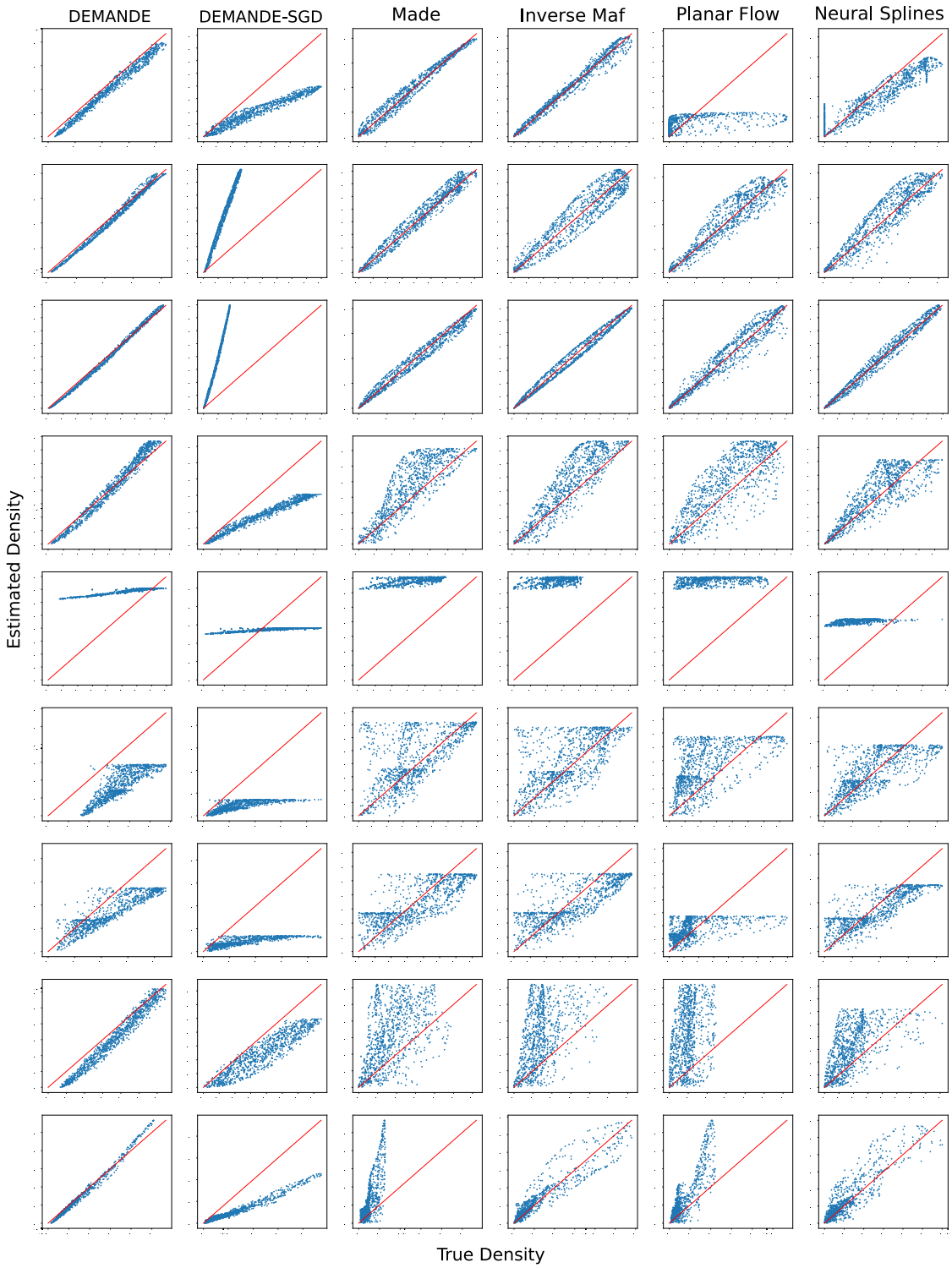

Figure 3-9.: Each graph represents the comparison of Spearman's correlation between the real density for each data set and the density estimate obtained by each algorithm. From left to right the algorithms are DEMANDE, DEMANDE-SGD, Made, Inverse Maf, Planar Flow, and Neural Splines. From top to bottom, the data sets are Arc, Bimodal Gaussian, 2-dimensional Gaussian distribution, Moons, Potential 1-4, Star, and Swizz Roll.

Part II.

Anomaly Detection

4. Quantum Anomaly Detection through Density Matrices, Adaptive Fourier Features and Autoencoders

This chapter presents an anomaly detection model that combines the strong statistical foundation of density-estimation-based anomaly detection methods with the representation-learning ability of deep-learning models. The method combines an autoencoder, for learning a low-dimensional representation of the data, with a density-estimation model based on random Fourier features and density matrices in an end-to-end architecture that can be trained using gradient-based optimization techniques. The method predicts a degree of normality for new samples based on the estimated density. A systematic experimental evaluation was performed on different benchmark datasets. The experimental results show that the method performs on par with or outperforms other state-of-the-art methods. The work presented in this chapter corresponds to the following articles:

Gallego-Mejia, J., Bustos-Brinez, O., & González, F. A. (2022). LANDM and ANDM: Quantum Inspired Density Matrices for Anomaly Detection. (To Be Submitted on Data Mining and Knowledge Discovery) [22, 56]
arXiv:2211.08525.

4.1. Introduction

Anomaly detection is a critical task in several applications, such as fraud detection [153], video surveillance [189], industrial defects [40], and medical image analysis [183], among others. In addition, it can be used as a preprocessing step in a machine learning system. In these scenarios, the idea is to detect whether a new sample is anomalous or not and leverage it to make actionable decisions [158]. Anomalies can change their name depending on the application domain, such as anomalies, outliers, novelties, exceptions, peculiarities, contaminants, among other terms. Anomaly detection refers to the task of finding patterns within the data that deviate from expected behavior [3].

Anomaly detection is a well-studied topic in machine learning. The idea is to detect unexpected behavior which deffer from the normal, expected behavior. Given r as the proportion of anomalous data points, a proportion greater than 10% is typically solved using a classification-based approach such as One Class Support Vector Machines [165]. Nonetheless, in general, anomalous data points are scarce and datasets are unbalanced, with a proportion of anomalous data points commonly less than 2%. The two main assumptions here are that the anomalous points are scarce, and the anomalous points deviate greatly from the normal points.

Classical methods for dealing with anomaly detection fall into three main categories: classification-based such as one-class SVM [165], distance-based such as local anomaly factor (LOF) [20] and isolation forest [111], and statistically-based such as kernel density estimation [129]. These methods have been used in several applications; however, they lack the representation learning ability of deep learning models that allow them to deal with complex, high-dimensional data such as images [109]. Recent shallow methods, such as Stochastic Outlier Selection (SOS) [83] and COPOD [104], are based on affinity and copulas, respectively. However, SOS constructs a matrix that is quadratic in terms of data points and COPOD does not directly compute joint distributions, which reduces its power. Recent deep learning methods, such as variational autoencoders (VAE) [142], do not directly solve the anomaly detection problem, but solve an indirect task with the minimization of the reconstruction error. The VAE assumes that anomalous points have a higher value of the reconstruction error compared to normal points, however, the algorithm has been shown to have sufficient power to reconstruct even anomalous points. The deep support vector data description [157] surrounds the normal points in a hypersphere but has an absence of degree of abnormality. Finally, the LAKE method [113] is based on the union of two parts: variational autoencoder and kernel density estimation (KDE). One problem with this method is that it optimizes the variational autoencoder and feeds its results into the memory-based KDE algorithm. Each latent space in the training data set must be saved to feed the KDE, with its corresponding memory footprint. In this chapter, we propose a new method, named LEAN-DMKDE, that uses the deep representation obtained by the autoencoders and the density captured by the density matrices, a formalism of quantum mechanics. The new method can be trained end-to-end, thus solving the shortcomings of LAKE.

The contributions of the present work are:

- Two new algorithms to anomaly detection that can be trained end-to-end with current deep learning tools.
- A framework for producing a degree of normality in high dimensional and big datasets.
- A methodology for combining autoencoders with kernel methods to produce normality likelihood estimation.
- A systematic comparison on multiple datasets of the novel method against state-of-

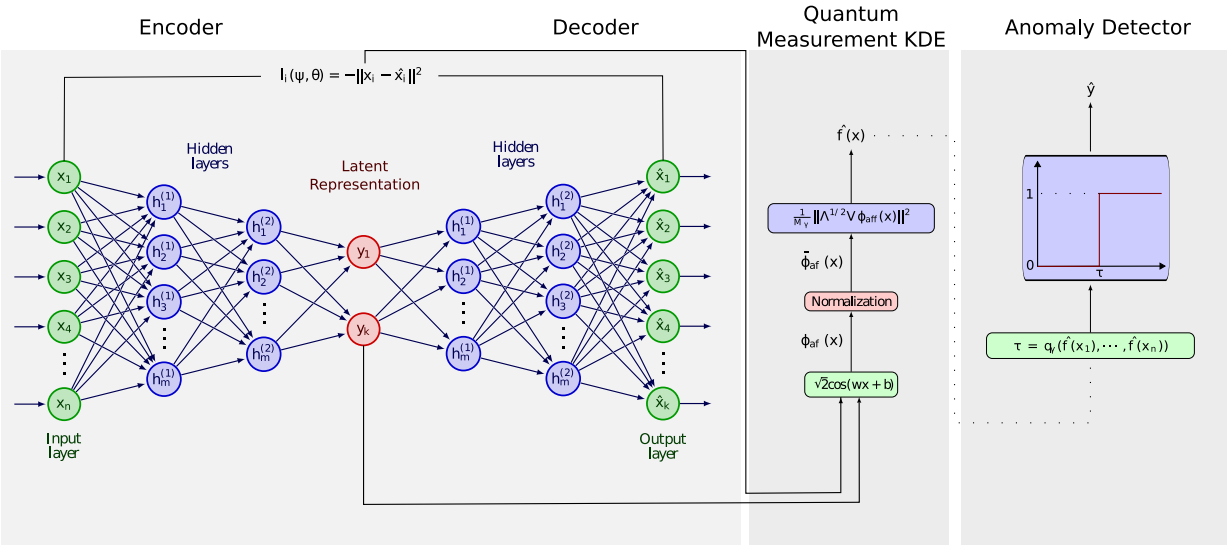

Figure 4-1.: Quantum Anomaly Detection through Density Matrices, Adaptive Fourier Features and Autoencoders (LEAN-DMKDE) method. Step 1, autoencoder representation learning. Step 2, training of the model using both the reconstruction error and the maximum likelihood estimation of the density matrix ρ. Step 3, estimation of the density of a new sample using the learn density matrix $\rho = V^T \Lambda V$. Step 4, anomaly detector using the proportion of the anomalies.

the-art anomaly detection methods.

Reproducibility: all the code is available at the Github repo http:// and a release in Zenodo.

4.2. Quantum Latent Density Estimation for Anomaly Detections (LEAN-DMKDE)

The proposed novel Quantum Latent Density Estimation for Anomaly Detection (LEAN-DMKDE) is shown in Figure 5-1. This model is constructed using an autoencoder, an adaptive Fourier feature layer, a quantum measurement layer and finally a threshold anomaly detection layer. The autoencoder is responsible for performing a dimensionality reduction phase and giving the reconstruction error. The adaptive Fourier feature layer maps the reduced event space to an approximate Hilbert space. The quantum measurement layer produces density estimates of the given data points. The last step of the algorithm is an anomaly detector that uses the density estimate and classifies whether a point is anomalous or normal.

4.2.1. Autoencoder

The first step of the algorithm is an autoencoder. Given an input space $\mathcal{X} \subseteq \mathbb{R}^d$, and a latent space $\mathcal{F} \subseteq \mathcal{R}^p$, where typically $p \ll d$, a point in the input space is sent through an ψ-encoder and a θ-decoder step, where $\psi : \mathcal{X} \to \mathcal{F}$ and $\theta : \mathcal{F} \to \mathcal{X}$, such that:

$$\boldsymbol{z}_i = \psi(\boldsymbol{x}_i, \boldsymbol{w}_\psi)$$
$$\hat{\boldsymbol{x}}_i = \theta(\boldsymbol{z}_i, \boldsymbol{w}_\theta)$$

where $\boldsymbol{w}_\psi$ and $\boldsymbol{w}_\theta$ are the network parameters of the encoder and the decoder respectively, and $\hat{x}$ is the reconstruction of the original data. The reconstruction error of the autoencoder is defined as the Euclidean distance between the original data point and its reconstruction:

$$l_i(\psi, \theta) = -||\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i||^2 \tag{4-1}$$

To capture further information computed by the autoencoder, the Euclidean distance error ($l_{\text{Euc\_dist},i}$) were combined with the cosine similarity ($l_{\text{cos\_sim},i}$), defined as the cosine of the angle between $\boldsymbol{x}_i$ and $\hat{\boldsymbol{x}}_i$, into one vector with the latent space ($\boldsymbol{z}_i$) as:

$$\text{rec\_measure}_i = [l_{\text{Euc\_dist},i}, l_{\text{cos\_sim},i}]$$

and the latent space output of the autoencoder to the next layer is $\boldsymbol{o}_i = [\boldsymbol{z}_i, \text{rec\_measure}_i]$.

4.2.2. Quantum Measurement Kernel Density Estimation

Density estimation is one of the most studied topics in statistics. Given a random variable X and its associated measurable space $\mathcal{A}$, from which we capture some events, observations or samples $x_1, \cdots, x_n$, the probability density function f in a measurable set $A \in \mathcal{A}$ is defined as a function that satisfies the following property:

$$Pr[X \in A] = \int_A f_X(\boldsymbol{x}) d\boldsymbol{x}$$

In general, the underlying density function of an arbitrary system is unknown. The most popular method for density estimation is kernel density estimation (KDE), also known as Parzen window. KDE has two principal drawbacks: its memory-based approach, i.e., it stores each data point $x_1, \cdots, x_n$ in the training phase and uses them to produce inferences in the testing phase, and it cannot be combined with deep learning architectures. Given a set of training points $\{\boldsymbol{x}_i\}_{i=1,\cdots,N}$, and a point $\boldsymbol{x}$ whose density is to be calculated, the Kernel Density Estimation calculates it as:

$$\hat{f}(\boldsymbol{x}) = \frac{1}{NM_\gamma} \sum_{i=1}^{N} k_\gamma(\boldsymbol{x}_i, \boldsymbol{x}) \tag{4-2}$$

where γ is the bandwidth parameter, $M\gamma$ is a normalization parameter, and k_γ is a kernel with specific properties [136].

In González et al. [66], Gallego and González [53], the authors showed that using a derivation of the kernel density estimation method using the Gaussian kernel, we can derive a density estimation method that captures quantum information as follows:

$$\hat{f}(\boldsymbol{x}) = \frac{1}{M_\gamma} \phi_{\text{aff}}(\boldsymbol{x})^T \rho \, \phi_{\text{aff}}(\boldsymbol{x}) \tag{4-3}$$

where ρ is a density matrix defined as

$$\rho = \frac{1}{N} \sum_{i=1}^{N} \phi_{\text{aff}}(\boldsymbol{x}_i) \phi_{\text{aff}}^T(\boldsymbol{x}_i)$$

and M_γ is a normalization constant. Note that ρ is a D-dimensional square matrix. The density matrix captures the classical and quantum probabilities of a quantum system, and it can be viewed as an aggregation of each data point in the feature space. The ρ matrix is required to be a Hermitian matrix and its trace must be equal to one, $tr(\rho) = 1$.

$\hat{\phi}_{\text{aff}}$ indicates the adaptive Fourier features, which are explained right next. Random Fourier features were first proposed by [144]. Its main idea is to build an explicit approximate mapping of the reproducing Hilbert space induced by a given kernel. In particular, they showed that if a Gaussian kernel is used, we can approximate

$$k(\boldsymbol{x}, \boldsymbol{y}) \simeq \mathbb{E}_{\boldsymbol{w}}(\langle \hat{\phi}_{\text{aff},\boldsymbol{w}}(\boldsymbol{x}), \hat{\phi}_{\text{aff},\boldsymbol{w}}(\boldsymbol{y}) \rangle)$$

where $\hat{\phi}_{\text{aff},\boldsymbol{w}} = \sqrt{2}\cos(\boldsymbol{w}^T\boldsymbol{x} + b)$, $\boldsymbol{w} \sim N(\boldsymbol{0}, \mathbb{I}^d)$ and $\boldsymbol{b} \sim \text{Uniform}(0, 2\pi)$. This expectation approximation is possible due to Bochner's theorem. The adaptive Fourier features are neural network-based fine-tuning of the random Fourier features. The explanation of this network and its usage is presented in the Training Strategy subsection. 4.2.4

The dimensions of the ρ matrix can be problematic, given its size D^2, but this can be alleviated by using the matrix eigen-decomposition:

$$\rho \approx V^T \Lambda V \tag{4-4}$$

where $V \in \mathcal{R}^{r \times D}$ is a matrix, whose rows contain the r vectors with the largest eigenvalues, and $\Lambda \in \mathcal{R}^{r \times r}$ is a diagonal matrix whose values correspond to the r largest eigenvalues of ρ. This matrix eigen-decomposition modify the Equation 5-3 to:

$$\hat{f}_\gamma(\boldsymbol{x}) \simeq \frac{1}{M_\gamma}||\Lambda^{1/2} V \phi_{\text{aff}}(\boldsymbol{x})||^2 \tag{4-5}$$

Equation 4-5 defines the algorithm to perform new density estimates based on density matrices. The parameters Λ and V are obtained by gradient descent optimization. Note that the adaptive Fourier features are trained independently using only the autoencoder and adaptive Fourier feature layer. The output of the adaptive Fourier feature layer is the input of the density matrices using the Equation 4-5. The density estimation error of the optimization process is as follows:

$$V^*, \Lambda^* = \arg\min_{V,\Lambda} \beta \sum_{i=1}^{N} \log \hat{f}(\boldsymbol{x}_i)$$

where $\hat{f}(\boldsymbol{x}_i) = \frac{1}{M_\gamma}||\Lambda^{1/2} V \phi_{\text{aff}}(\boldsymbol{x})||^2$ and $\phi_{\text{aff}}(\boldsymbol{x}_i) = \sqrt{2} cos(\boldsymbol{w}^T \boldsymbol{x}_i + b)$. The algorithm is trained minimizing both reconstruction and estimation losses using gradient descent.

4.2.3. Anomaly Detection Step

The last step of the anomaly detection algorithm is to detect whether a given data point is a normal or anomalous point. The algorithm uses the proportion of anomalies (γ) within the data set to obtain a threshold value τ as:

$$\tau = q_\gamma(\hat{f}(\boldsymbol{x}_1), \cdots, \hat{f}(\boldsymbol{x}_n))$$

where q is the percentile function. In the detection phase, the algorithm uses the τ value to classify anomalous points as:

$$\hat{y}(x_i) = \begin{cases} \text{'normal'} & \textit{if } \hat{f}(x_i) \geq \tau \\ \text{'anomaly'} & \textit{otherwise} \end{cases}$$

4.2.4. Training Strategy

The random Fourier feature approximation can be further tuned using gradient descent, in a process called "Adaptive Fourier features". This algorithm was first proposed in [53]. It selects random pairs of data points $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and applies gradient descent to find:

$$\boldsymbol{w}^*, b^* = \arg\min_{\boldsymbol{w},b} \frac{1}{m} \sum_{\boldsymbol{x}_i, \boldsymbol{x}_j \in s} (k_\gamma(\boldsymbol{x}_i, \boldsymbol{x}_j) - \hat{k}_{\boldsymbol{w},b}(\boldsymbol{x}_i, \boldsymbol{x}_j))^2$$

where $k_\gamma(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is the Gaussian kernel with a γ-bandwidth parameter, and $\hat{k}_{\boldsymbol{w},b}(\boldsymbol{x}_i, \boldsymbol{x}_j))^2$ is the Gaussian kernel approximation. As an initialization to the Gaussian kernel approximation, we can sample $\boldsymbol{w}$ and b using $N(\mathbf{0}, \mathbb{I}^d)$, and Uniform$(0, 2\pi)$, respectively, the same as the random Fourier features.

It should be noted that each $\boldsymbol{x}_i$ is a vector with m dimensions. For the anomaly detection algorithm, m is the latent space of the autoencoder defined above. Therefore, we propose an intermediate step to approximate the kernel in the latent space. First, the autoencoder is trained without any additional layer. Second, we use forward propagation for several $\boldsymbol{x}_i$ data points to reduce the original space to the encoded space. Finally, the adaptive Fourier feature algorithm is optimized as proposed above.

The training process is shown in the Algorithm 5. The hyperparameter α controls the trade-off between the minimization of the reconstruction error and the maximization of the log-likelihood. Using forward-propagation, each $\boldsymbol{x}_i$ is sent through the encoder $\psi(\boldsymbol{x}_i, \boldsymbol{w}_\psi)$ and the decoder $\theta(\boldsymbol{z}_i, \boldsymbol{w}_\theta)$. Then, reconstruction errors are computed and combined with the encoded $\boldsymbol{z}_i$ as $\boldsymbol{o}_i$. Using the parameters of the adaptive Fourier features, the vector $\boldsymbol{o}_i$ is mapped to a Hilbert space in which dot product approximates the Gaussian kernel. With the mapping $\phi_{\text{aff},\boldsymbol{w}_{\text{aff}}}$, we compute the likelihood of each sample using the density matrix decomposition obtaining $\hat{f}(\boldsymbol{x}_i)$. A minimization optimization is performed using backpropagation and stochastic gradient descent. Finally, τ is computed as the percentile associated to a given γ proportion of anomalies.

4.3. Experimental Evaluation

In this section, we present an experimental evaluation of the novel LEAN-DMKDE method. Our objective was to compare its performance against fourteen classic and deep anomaly detection methods across twenty-four diverse datasets. To assess the effectiveness of the methods, we employed two primary metrics: AUC-PR and AUC-ROC. Furthermore, we performed statistical analysis to demonstrate the superior performance of the novel method.

4.3.1. Experimental Setup

The experimental framework used in this chapter intends to compare LEAN-DMKDE with all the baseline algorithms listed in previous sections. To run One Class SVM, Minimum Covariance Determinant, Local Outlier Factor and Isolation Forest, the implementation used is the one provided by Scikit-Learn Python library. KNN, SOS, COPOD, LODA, VAE and DeepSVDD were run using the implementation provided by PyOD Python library [200]. LAKE algorithm was implemented based on the Github repository of its authors, although we had to correct the way the test dataset was split to include both normal and anomalous

Algorithm 5: LEAN-DMKDE training process

Input: Training dataset $D = \{\boldsymbol{x}_i\}_{i=1,\cdots,N}$
Parameters: α: trade-off between reconstruction error and log-likelihood,
$\boldsymbol{w}_{\text{rff}}, b$: adaptive Fourier features parameters
γ: proportion of anomalies
Output: $\boldsymbol{w}_\psi$: encoder parameters,
$\boldsymbol{w}_\theta$: decoder parameters,
V^*, Λ^* : quantum measurement KDE parameters
for $\boldsymbol{x}_i \in D$ ***until convergence*** **do**
  $\boldsymbol{z}_i = \psi(\boldsymbol{x}_i, \boldsymbol{w}_\psi)$
  $\hat{\boldsymbol{x}}_i = \theta(\boldsymbol{z}_i, \boldsymbol{w}_\theta)$
end
$\boldsymbol{w}^*_{\text{AFF}}, b^*_{\text{AFF}} = \arg\min_{\boldsymbol{w},b} \frac{1}{m} \sum_{\boldsymbol{x}_i,\boldsymbol{x}_j \in D} (k_\gamma(\psi_{\boldsymbol{w}_\psi}(\boldsymbol{x}_i), \psi_{\boldsymbol{w}_\psi}(\boldsymbol{x}_j) - \hat{k}_{\boldsymbol{w},b}(\psi_{\boldsymbol{w}_\psi}(\boldsymbol{x}_i), \psi_{\boldsymbol{w}_\psi}(\boldsymbol{x}_j)))^2$
for $\boldsymbol{x}_i \in D$ ***until convergence*** **do**
  $\boldsymbol{z}_i = \psi(\boldsymbol{x}_i, \boldsymbol{w}_\psi)$
  $\hat{\boldsymbol{x}}_i = \theta(\boldsymbol{z}_i, \boldsymbol{w}_\theta)$
  $l_{\text{Euc\_dist},i} = \text{Euclidean\_distance}(\boldsymbol{x}_i, \hat{\boldsymbol{x}}_i)$
  $l_{\text{cos\_sim},i} = \text{cosine\_similarity}(\boldsymbol{x}_i, \hat{\boldsymbol{x}}_i)$
  $\boldsymbol{o}_i = [\boldsymbol{z}_i, l_{\text{Euc\_dist},i}, l_{\text{cos\_sim},i}]$
  $\hat{f}(\boldsymbol{o}_i) = \frac{1}{M_\gamma} ||\Lambda^{1/2} V \phi_{\text{aff},\boldsymbol{w}}(\boldsymbol{o_i})||^2$
  $\mathbf{L}_{\boldsymbol{w}_\psi,\boldsymbol{w}_\theta,V^*,\Lambda^*} = ||\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i||^2 - \alpha \sum_{i=1}^N \log \hat{f}(\boldsymbol{o}_i)$
  $\boldsymbol{w}_\psi, \boldsymbol{w}_\theta, V^*, \Lambda^* \leftarrow$ update parameters minimizing $\mathbf{L}$
end
$\tau = q_\gamma(\hat{f}(\boldsymbol{x}_1), \cdots, \hat{f}(\boldsymbol{x}_n))$
return $\boldsymbol{w}_\psi, \boldsymbol{w}_\theta, V^*, \Lambda^*, \tau$

samples in it.

In order to handle the inherent randomness found in some of the algorithms, it was decided to fix in advance (into a single, invariant number) all the random seeds that could affect the different stages of each algorithm (particularly dataset splitting and initialization steps). All experiments were carried out on a machine with a 2.1GHz Intel Xeon 64-Core processor with 128GB RAM and two RTX A5000 graphic processing units, that run Ubuntu 20.04.2 operating system.

Datasets

To evaluate the performance of LEAN-DMKDE for anomaly detection tasks, twenty public datasets were selected. These datasets came from two main sources: the Github repository

Dataset	Instances	Dimensions	Outlier Rate
Arrhythmia	452	274	0,146
Cardio	2060	22	0,2
Glass	214	9	0,042
KDDCUP	5000	118	0,1934
Lympho	148	18	0,04
Ionosphere	351	33	0,359
Letter	1600	32	0,0625
MNIST	7603	100	0,092
Musk	3062	166	0,0317
OptDigits	5216	64	0,0288
PenDigits	6870	16	0,0227
Pima	768	8	0,349
Satellite	6435	36	0,3164
SatImage	5803	36	0,0122
Shuttle	5000	9	0,0715
SpamBase	3485	58	0,2
Thyroid	3772	36	0,0247
Vertebral	240	6	0,125
Vowels	1456	12	0,03434
WBC	378	30	0,0556

Table 4-1.: Main features of the datasets.

associated with LAKE[1], and the ODDS virtual library of Stony Brook University[2]. The main characteristics of the selected datasets, including their sizes, dimensions and outlier rates can be seen in Table **5-1**. These datasets were chosen because of the wide variety of features they represent, with which the performance of the algorithms can be tested in multiple scenarios, its extensive use in anomaly detection literature, and because of their accessibility, since the files associated with each dataset can be easily accessed in their respective sources.

Metrics

To evaluate the performance of the proposed algorithm in comparison to baseline anomaly detection methods, we employed AUC-PR and AUC-ROC as the primary metrics. These metrics are extensively utilized in machine learning frameworks due to their effectiveness. Although additional metrics such as accuracy and F1-Score on the anomaly class were also calculated, they were not reported in this study. The calculations for AUC-PR, AUC-ROC, and the aforementioned additional metrics were performed using the implementation

[1]https://github.com/1246170471/LAKE
[2]http://odds.cs.stonybrook.edu/about-odds/

provided by the Scikit-Learn Python library [137].

For each algorithm, a set of parameters of interest was selected in order to perform a series of searches for the combinations of parameters that gave the best results for each data set under study. The list of algorithms and their parameters under study can be found in Table 4-2.

Algorithm	Parameters of Interest
One Class SVM	gamma (Gaussian kernel), outlier percentage
Isolation Forest	number of estimators, samples per estimator, outlier percentage
Covariance Estimator	outlier percentage
LOF (Local Outlier Factor)	number of neighbors, outlier percentage
K-nearest Neighbors	number of neighbors, outlier percentage
SOS	perplexity (matrix parameter), outlier percentage
COPOD	outlier percentage
LODA	outlier percentage
VAE-Bayes	outlier percentage
DeepSVDD	outlier percentage
LAKE	normality ratio (related to outlier percentage)
LEAN-DMKDE	sigma (Gaussian kernel), architecture of the autoencoder, size of Fourier features mapping, size of density matrix eigen-decomposition, alpha (trade-off parameter)

Table 4-2.: Parameters of all the algorithms selected for grid search.

The selection of the best parameters was made by using a grid search strategy, ranking all the possible combinations of parameters (up to a limit of 100 experiments) in terms of the Auc-PR and AUC-ROC, and choosing the combination that showed the highest value for the metrics. This selection of parameters was performed for each algorithm and each dataset.

4.3.2. Results and Discussions

The Auc-ROC and Auc-PR obtained when using the winning combinations for all experiments (algorithm and dataset pairs) are reported in Table 4-5 and Table 4-6. At a first glance, there is a noticeable difference between the performance of most of the baseline methods and LEAN-DMKDE, with AD-DMKDE being a notable exception; other methods that show better-than-average results include an autoencoder and isolation forest, that had a slightly better performance than deep learning alternatives like LAKE, VAE-Bayes or DeepSVDD.

Upon analyzing the influence of dataset features on performance, several patterns emerge. LEAN-DMKDE demonstrates strong performance on datasets with outlier rates below 10% but achieves its best performance with datasets ranging from 10% to 30%. Notably, LEAN-DMKDE outperforms AD-DMKDE on datasets with higher dimensionality (more than 20 dimensions), which supports our hypothesis.

It is important to note that while the AUC-ROC of AD-DMKDE has a higher average value compared to LEAN-DMKDE, this metric tends to be biased towards highly imbalanced datasets. It gives higher scores to methods that predominantly predict the majority class, which, in this case, is the normal class. Therefore, we relied on the AUC-PR of the anomaly class as a more appropriate measure. This metric specifically evaluates the performance of the methods on the anomaly class. It is worth mentioning that a method with a higher AUC-ROC may undermine the importance of detecting anomalies.

To further analyze these results, a comprehensive statistical analysis was conducted. Specifically, the Friedman test was utilized to assess the Auc-PR and Auc-ROC values, revealing a statistically significant difference among the methods. Subsequently, the Friedman-Nemenyi test was performed to delve deeper into the findings. The outcomes of this test are visually presented in Figure 4-2. In the figure, significant differences between dataset pairs are denoted by black squares, while white squares indicate no significant difference. Upon analyzing the figure, it becomes evident that the methods can be grouped into three distinct categories. The first group comprises LEAN-DMKDE, AD-DMKDE, Autoencoder, and iForest. The subsequent group consists of Covariance, KNN, VAE-Bayes, and COPOD. Lastly, the third group encompasses LODA, ALAD, OCSVM, LOF, SOS, LAKE, and DSVDD.

4.3.3. Ablation Study

To establish that the LEAN-DKMDE architecture can achieve a better performance than each of its separate components, the following experiments were performed:

- AN-DMKDE: using AN-DMKDE that is the final part of LEAN without the autoencoder. This method uses the density matrices to produce anomaly score.

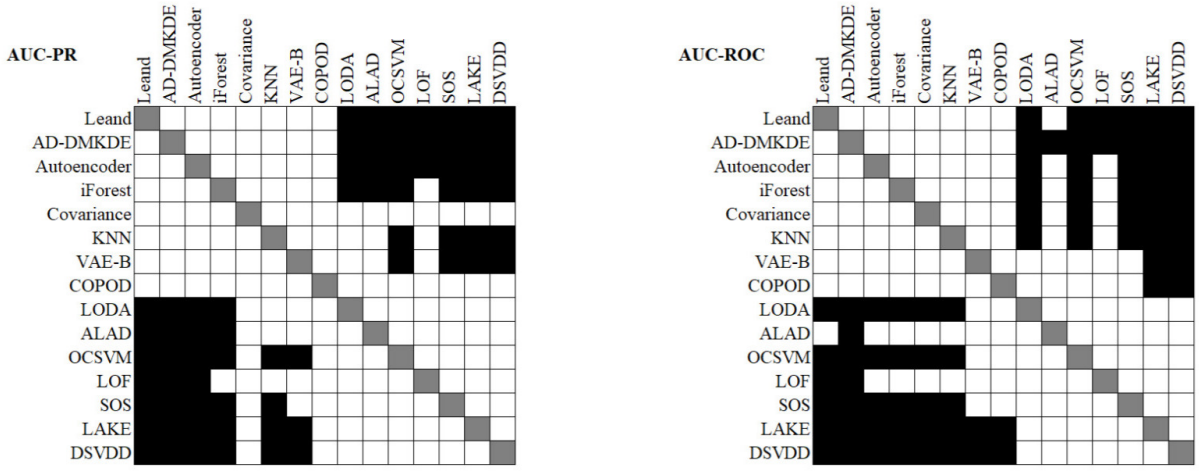

Figure 4-2.: Results for Friedman-Nemenyi test over the metrics of Auc-PR and Auc-ROC.

- AutoEncoder only: since the proposed method uses an autoencoder as a mechanism to create a latent representation of the data, the neural network alone was applied over the datasets, in such a way that the reconstruction error was used as a measure of anomaly.
- LAKE: is a method that leverages an autoencoder and Kernel Density Estimation (KDE) without end-to-end optimization.

The performance of these different experiments over all datasets are compared with LEAN-DKMDE using Auc-PR as before. A parameter grid search was also performed over the parameters that applied in each case and with the same value ranges used in LEAN-DKMDE. Results can be seen in Table 4.3.3.

Based on the results, it is evident that LEAN-DKMDE outperforms both AN-DMKDE and the Autoencoder stages, particularly the latter. AN-DMKDE demonstrates favorable performance in certain datasets, potentially due to its effectiveness in datasets with fewer features. However, when compared to the combined approach of LEAN-DKMDE, relying solely on reconstruction error, as in the case of the Autoencoder alone, proves to be a notable limitation. LAKE exhibits the poorest performance among the methods evaluated. These findings highlight the significance of fine-tuning the density matrix in accordance with the autoencoder to achieve optimal results.

As established in the Experimental Setup section, a parameter grid search was performed for every algorithm and dataset, in order to find the combination of parameters that fitted the best for each dataset. For every parameter, the search was conducted over a series of values that were the same in all experiments (when possible). In Table 4-4, a list of the ranges for all the parameters is presented. Some of these parameters are associated with the outlier rate of each dataset: the outlier percentage parameter range was defined as the

Table 4-3.: Ablation study: area under the Precision-Recall curve (AUC-PR) for all algorithms applied on all data sets. The highest values have been highlighted, the first in bold and the second in underline.

name	Leand	AD-DMKDE	Autoencoder	LAKE
Annthyroid	**0,248**	0,187	0,232	0,174
Arrhythmia	0,454	**0,534**	0,515	0,116
Breastw	**0,988**	0,987	0,98	0,864
Cardio	0,47	**0,627**	0,604	0,116
ForestCover	0,465	**0,494**	0,033	0,095
Glass	**0,643**	0,167	0,077	0,036
Ionosphere	0,797	**0,984**	0,966	0,228
Letter	0,156	**0,489**	0,29	0,036
Mammography	0,253	0,187	**0,356**	0,072
MNIST	**0,71**	0,549	0,551	0,333
Musk	0,747	**1**	**1**	0,853
OptDigits	0,407	**0,526**	**0,555**	0,021
PenDigits	**0,672**	0,614	0,173	0,06
Pima	0,572	**0,596**	0,487	0,319
Satellite	0,809	**0,882**	0,795	0,23
SatImage	0,918	0,764	**0,948**	0,175
Shuttle	**0,9855**	0,957	0,96	0,163
SpamBase	**0,966**	0,818	0,487	0,576
Speech	**0,043**	0,016	0,019	0,026
Thyroid	0,364	0,374	**0,499**	0,013
Vertebral	**0,787**	0,26	0,197	0,164
Vowels	0,276	**0,757**	0,574	0,267
WBC	**0,758**	0,703	0,489	0,237
Wine	**1**	0,467	0,756	0,274
Mean	**0,6036**	0,5807	0,5226	0,227

true value plus five near values over it and five near values under it. The same applies to normality ratio, defined as 1.0 minus the true outlier rate, plus five values over it and five values under it.

4.3.4. Dataset Description

The data sets used in the different experiments come from various scientific fields and have a wide variety of characteristics, which allowed testing the performance of the algorithm in different scenarios. Next, we present a list of the datasets used, including their source and a brief description of their contents.

- Annthyroid: dataset from the UCI Machine Learning Repository is a publicly available dataset that is commonly used for classification tasks in machine learning. It is derived from a thyroid disease study and contains various attributes related to thyroid function tests. The dataset is named "annthyroid" because it is often used to train and evaluate artificial neural networks (ANNs) for thyroid disease diagnosis. Patients with thyroid disease are labeled as anomalous points.
- Arrhythmia: originally a multiclass dataset, it was modified for anomaly detection, labeling the eight smallest classes as outliers, and the remaining classes as normal data. The data file used in the experiments came from LAKE Github repository.
- Breastw: The Breast Cancer Wisconsin (Diagnostic) Dataset contains various features computed from digitized images of fine needle aspirates (FNA) of breast masses. These

Parameter	Range of Values
Outlier percentage	dataset dependant
Number of neighbors	[2, 4, 6, ..., 46, 48, 50] (LOF) [10, 20, ..., 90, 100] (K-nearest neighbors)
Gamma (Gaussian kernel)	$[2^{-10}, 2^{-9}, ..., 2^{4}, 2^{5}]$ (One-class SVM)
Normality ratio	dataset dependant
Number of estimators	[20, 40, 60, 80, 100] (Isolation-Forest)
Samples per estimator	[20, 40, 60, 80, 100] (Isolation-Forest)
Perplexity	[10, 20, ..., 90, 100] (SOS)
Sigma (Gaussian kernel)	$[2^{-5}, 2^{-4}, ..., 2^{8}, 2^{9}]$ (LEAN-DKMDE)
Autoencoder architecture (layer sizes)	[(64,32,16,32,64), (128,64,32,8,32,64,128), (128,32,2,32,128), (64,20,10,4,10,20,64)] (LEAN-DKMDE)
Size of Fourier features mapping	[250, 500, 1000, 2000] (LEAN-DKMDE)
Size of density matrix eigen-decomposition	[12, 25, 50, 100, 125, 200, 250, 400, 500, 1000, 2000] (LEAN-DKMDE)
Alpha (loss function parameter)	[0, 0.01, 0.1, 0.5, 0.9, 0.99, 1] (LEAN-DKMDE)

Table 4-4.: Range of all the parameters used for grid search.

features are used to predict whether a given breast mass is benign (non-cancerous) or malignant (cancerous).

- Cardio: This dataset is related to fetal heart measurements. Originally a 3-class dataset, one of the classes was discarded and the pathological samples are considered

outliers. The data file used in the experiments came from LAKE Github repository.

- ForestCover:
- Glass: This data set contains information on six different types of glass in terms of iron oxide content. The sixth type is considered an outlier and the rest are normal data. The data file used in the experiments came from ODDS virtual library.
- Ionosphere: it contains data from radar measurements in high-altitude atmospheric layers, The data file used in the experiments came from ODDS virtual library.
- Letter: this dataset originally contained rectangular displays (4x4 size) of three English letters, labeling one of them as 'anomaly'. The version we used came from the ODDS virtual library, where the data were combined in pairs to obtain a dimension of 32; outliers correspond to pairs where one letter belongs to the 'anomaly' class.
- Mammography: dataset is a publicly available dataset that is often used for breast cancer detection and classification tasks in machine learning. It contains features extracted from mammographic images, along with corresponding class labels indicating the presence or absence of breast cancer. The dataset is designed to aid in the development of predictive models for early detection of breast cancer.
- MNIST: one of the best-known datasets for automatic classification, the dataset we use (a subset of the original with only digits zero and six) contains images of size 10x10. The zero-digit class is considered normal data, and the six-digit class is the outlier data. The data file used in the experiments came from ODDS virtual library.
- Musk: This dataset contains multiple configurations of molecules, in order to determine whether each configuration is a musk or not (i.e., has a strong odor or not). The musk molecules are fewer, so they are labeled as outliers. The data file used in the experiments came from ODDS virtual library.
- OptDigits: the original data set had ten classes of handwritten digits of size 8x8, so for our experiments, cases of digits 1-9 are normal data, and cases of zero digits are outliers. The data file used in the experiments came from ODDS virtual library.
- PenDigits: This data set also refers to handwritten digits but is represented as eight different pairs of (x,y) coordinates through which each stroke passes. Cases of null digits are labeled as outliers. The data file used in the experiments came from ODDS virtual library.
- Pima: this dataset contains medical data on Indian women and was intended to determine whether or not patients have diabetes. The data file used in the experiments came from ODDS virtual library.
- Satellite: This dataset is composed of 3x3 slices taken from a Landsat satellite image, and there are six classes representing different soil types. The three smallest classes ('2', '4' and '5') are labeled as outliers. The data file used in the experiments came

from ODDS virtual library.

- SatImage: coming from the previous Satellite dataset, here class '2' has been downsampled and considered as outlier data, while all the other classes are labeled as normal data. The data file used in the experiments came from ODDS virtual library.

- SpamBase: this dataset contains email information, in which spam (unsolicited commercial emails) are labeled as outliers. The data file used in the experiments came from LAKE Github repository.

- Shuttle: this dataset has 9 numerical values and 7 classes. Class '4' was discarded, class '1' is taken as normal data, and the remaining classes make up the outliers. The data file used in the experiments came from ODDS virtual library.

- Speech: The dataset comprises a collection of vectors that represent segments of word recordings in English, featuring a diverse range of accents. Each accent is assigned a distinct class label. In this particular experiment, the American accent is considered the normal data, while the remaining accents are treated as anomalous data.

- Thyroid: This dataset includes 3372 patient instances for diagnosing hypothyroidism. It has three classes, but only sick patients are treated as outliers, because it is a minority class. The data file used in the experiments came from LAKE Github repository.

- Vertebral: this dataset contains biomechanical information from 240 patients, referring to attributes of the pelvis and lumbar spine. The normal patients are fewer, so they are taken as outliers. The data file used in the experiments came from ODDS virtual library.

- Vowels: this dataset contains 12 discrete time series sampled from Japanese vowel recordings. Originally, each class represented a different speaker, and one of the speakers is considered as outlier data and the rest as normal data. The data file used in the experiments came from ODDS virtual library.

- WBC: the data here is the result of analyzing breast cancer images and contain attributes of tumor cell nuclei. Benign instances are labeled as normal data, and malignant instances are labeled as outliers. The data file used in the experiments came from ODDS virtual library.

- Wine: The dataset contains chemical analysis results of wines produced in three distinct Italian vineyards. Each class represents a specific vineyard. In the ODDS version of the dataset, a small subset of samples from Class '1' is designated as anomalous, while Classes '2' and '3' are treated as normal data.

4.3.5. Statistical Analysis

To establish whether LEAN-DKMDE is statistically different (in terms of performance) with respect to the baseline methods, the Friedman test, a well-known statistical method to compare different populations or groups, was performed. This test uses the following formula:

$$Q = \left[\frac{12}{Nk(k+1)} \sum_{j=1}^{k} R_j^2 \right] - 3N(k+1)$$

where N is the number of datasets, k is the number of algorithms, and R_j^2 is the squared sum of the observations for each particular algorithm.

Given a confidence value $\alpha = 0.05$ and the p-value given by $\mathbf{P}[\chi_{n-1}^2 \geq Q]$, we determine that if the p-value is lower than α, there is statistically significant evidence supporting that the methods are different. When applying this test to the results, it indicates that for both F1 Score and Accuracy there is a statistically significant difference between the methods. But Friedman test does not answer which of the methods are responsible for the difference if present. To compare them two by two, we use the Friedman-Nemenyi test, a variation of the former that applies over all pairs of methods. This statistical test was performed for the two considered metrics (F1-Score and Accuracy).

4.4. Conclusions

This chapter presented Quantum Latent Density Estimation for Anomaly Detection (LEAN-DKMDE), a novel method for anomaly detection based on the combination of autoencoders, adaptive Fourier features and density estimation through quantum measurements. This new method was compared against fourteen different anomaly detection algorithms, using a framework that included twenty-four labeled anomaly detection datasets. For each dataset and algorithm, a grid search of the best parameters was performed, and the performance of the winning algorithms was compared using Auc-PR and Auc-ROC. LEAN-DKMDE showed state-of-the-art performance, being superior to most classic algorithms and comparable to deep learning-based methods. The reliability of the proposed method does not seem to be affected by the features of the dataset, although LEAN-DKMDE highlights in high-dimensional datasets. Also, LEAN-DKMDE performs better than its separate parts (KDE and autoencoder) and there is a noticeable difference when using only reconstruction measures, all of which shows that the combination of density estimation and reconstruction from autoencoders can perform better than these two elements separately.

Table 4-5.: Area under the Precision-Recall curve (AUC-PR) for all algorithms applied on all data sets. The highest values have been highlighted, the first in bold and the second in underline.

name	Leand	AD-DMKDE	Autoencoder	iForest	Covariance	KNN	VAE-Bayes	COPOD	LODA	ALAD	OCSVM	LOF	SOS	LAKE	DSVDD
Annthyroid	0,248	0,187	0,232	0,34	**0,504**	0,212	0,209	0,197	0,147	0,192	0,127	0,186	0,142	0,174	0,083
Arrhythmia	0,454	0,534	0,515	0,558	0,459	0,514	0,532	**0,566**	0,365	0,393	0,425	0,433	0,472	0,116	0,32
Breastw	0,988	0,987	0,98	**0,994**	0,969	0,988	0,968	0,987	0,963	0,958	0,825	0,335	0,886	0,864	0,904
Cardio	0,47	0,627	0,604	**0,693**	0,469	0,555	0,595	0,581	0,198	0,507	0,465	0,185	0,254	0,116	0,267
ForestCover	0,465	**0,494**	0,033	0,097	0,015	0,06	0,07	0,06	0,065	0,043	0,116	0,015	0	0,095	0,01
Glass	**0,643**	0,167	0,077	0,072	0,097	0,121	0,077	0,076	0,058	0,056	0,035	0,035	0,072	0,036	0,143
Ionosphere	0,797	**0,984**	0,966	0,711	0,92	0,951	0,778	0,77	0,449	0,848	0,754	0,837	0,843	0,228	0,958
Letter	0,156	**0,489**	0,29	0,111	0,241	0,228	0,087	0,072	0,063	0,081	0,075	0,375	0,288	0,036	0,271
Mammography	0,253	0,187	0,356	0,307	0,145	0,193	0,196	**0,416**	0,154	0,19	0,042	0,113	0,072	0,072	0,078
MNIST	**0,71**	0,549	0,551	0,326	0,494	0,468	0,431	0,246	0,185	0,264	0,216	0,309	0,228	0,333	0,148
Musk	0,747	**1**	**1**	0,891	0,982	0,992	**1**	0,449	0,545	0,142	0,214	0,032	0,251	0,853	0,026
OptDigits	0,407	0,526	**0,555**	0,034	0,021	0,022	0,028	0,044	0,017	0,021	0,024	0,032	0,032	0,021	0,026
PenDigits	**0,672**	0,614	0,173	0,339	0,092	0,208	0,208	0,162	0,289	0,121	0,118	0,018	0,063	0,06	0,048
Pima	0,572	**0,596**	0,487	0,463	0,49	0,497	0,45	0,479	0,339	0,424	0,393	0,463	0,415	0,319	0,453
Satellite	0,809	**0,882**	0,795	0,66	0,76	0,61	0,631	0,601	0,596	0,561	0,54	0,36	0,354	0,23	0,32
SatImage	0,918	0,764	0,948	0,912	0,635	**0,955**	0,822	0,751	0,923	0,575	0,394	0,06	0,056	0,175	0,014
Shuttle	**0,9855**	0,957	0,96	0,968	0,842	0,19	0,922	0,954	0,911	0,878	0,568	0,1	0,111	0,163	0,072
SpamBase	**0,966**	0,818	0,487	0,372	0,307	0,345	0,348	0,44	0,302	0,264	0,273	0,155	0,26	0,576	0,231
Speech	0,043	0,016	0,019	0,023	0,022	0,019	0,02	0,02	0,029	0,02	0,019	**0,044**	0,024	0,026	0,028
Thyroid	0,364	0,374	0,499	0,675	**0,688**	0,324	0,442	0,223	0,131	0,402	0,286	0,293	0,082	0,013	0,125
Vertebral	**0,787**	0,26	0,197	0,133	0,138	0,125	0,165	0,119	0,106	0,123	0,218	0,153	0,121	0,164	0,077
Vowels	0,276	**0,757**	0,574	0,074	0,05	0,641	0,148	0,049	0,05	0,075	0,083	0,374	0,199	0,267	0,217
WBC	0,758	0,703	0,489	0,587	0,597	0,484	0,506	0,7	0,626	0,161	0,229	**0,76**	0,476	0,237	0,192
Wine	**1**	0,467	0,756	0,61	0,756	0,444	0,297	0,533	0,494	0,421	0,242	0,639	0,091	0,274	0,137
Mean	**0,6036**	0,5807	0,5226	0,4562	0,4455	0,4227	0,4137	0,3956	0,3335	0,3216	0,2783	0,2627	0,2413	0,227	0,2145

Table 4-6.: Area under the ROC curve (AUC-ROC) for all algorithms applied on all data sets. The highest values have been highlighted, the first in bold and the second in underline.

Algorithm	LEAN	ADDMKDE	Autoencoder	iForest	Covariance	KNN	VAE-Bayes	COPOD	LODA	ALAD	OCSVM	LOF	SOS	LAKE	DSVDD
Annthyroid	0,733	0,788	0,648	0,833	**0,918**	0,704	0,704	0,799	0,66	0,687	0,617	0,71	0,642	0,7	0,51
Arrhythmia	0,753	0,773	0,778	**0,863**	0,817	0,776	0,769	0,787	0,676	0,724	0,807	0,832	0,738	0,31	0,707
Breastw	0,994	0,994	0,989	**0,997**	0,985	0,993	0,961	0,993	0,962	0,969	0,899	0,518	0,927	0,93	0,946
Cardio	0,761	0,813	0,951	0,951	0,863	0,937	**0,953**	0,919	0,677	0,924	0,893	0,645	0,829	0,474	0,655
ForestCover	0,972	**0,987**	0,837	0,939	0,686	0,891	0,933	0,877	0,905	0,859	0,877	0,558	0	0,878	0,5
Glass	0,769	**0,885**	0,663	0,635	0,74	0,788	0,663	0,654	0,49	0,529	0,212	0,221	0,558	0,25	0,76
Ionosphere	0,806	**0,988**	0,978	0,772	0,916	0,964	0,833	0,853	0,456	0,874	0,743	0,842	0,84	0,069	0,965
Letter	0,637	**0,879**	0,837	0,647	0,83	0,822	0,499	0,532	0,464	0,509	0,462	0,837	0,816	0,138	0,554
Mammography	0,907	**0,914**	0,885	0,88	0,747	0,845	0,883	0,908	0,773	0,84	0,557	0,786	0,626	0,703	0,531
MNIST	**0,948**	0,925	0,893	0,8	0,908	0,876	0,856	0,784	0,658	0,734	0,693	0,767	0,739	0,842	0,543
Musk	0,984	**1**	**1**	0,993	0,999	**1**	**1**	0,958	0,938	0,717	0,813	0,442	0,856	0,881	0,408
OptDigits	0,945	**0,991**	0,975	0,568	0,355	0,394	0,522	0,694	0,215	0,368	0,424	0,417	0,548	0,33	0,433
PenDigits	0,977	**0,996**	0,91	0,941	0,862	0,96	0,939	0,907	0,925	0,896	0,865	0,374	0,683	0,65	0,516
Pima	**0,744**	0,705	0,655	0,666	0,685	0,677	0,608	0,613	0,378	0,604	0,533	0,638	0,589	0,465	0,574
Satellite	0,839	**0,875**	0,827	0,674	0,79	0,758	0,625	0,658	0,598	0,609	0,582	0,542	0,563	0,294	0,498
SatImage	0,964	0,99	0,997	0,997	0,994	**0,998**	0,99	0,979	0,995	0,939	0,924	0,69	0,816	0,837	0,54
Shuttle	0,988	0,987	0,991	**0,994**	0,989	0,716	0,988	0,993	0,973	0,988	0,927	0,541	0,505	0,634	0,501
SpamBase	**0,969**	0,884	0,861	0,738	0,669	0,722	0,692	0,766	0,637	0,63	0,667	0,384	0,581	0,851	0,536
Speech	**0,709**	0,513	0,504	0,483	0,516	0,507	0,505	0,534	0,562	0,582	0,495	0,597	0,568	0,531	0,651
Thyroid	0,855	0,951	0,97	0,99	**0,993**	0,945	0,958	0,937	0,585	0,972	0,929	0,951	0,741	0,075	0,605
Vertebral	**0,965**	0,528	0,593	0,483	0,498	0,477	0,566	0,477	0,383	0,501	0,69	0,524	0,45	0,523	0,143
Vowels	0,732	**0,982**	0,916	0,668	0,653	0,961	0,613	0,539	0,54	0,733	0,58	0,912	0,825	0,298	0,555
WBC	0,96	0,922	0,913	0,94	0,94	0,909	0,898	0,94	0,909	0,762	0,464	**0,976**	0,918	0,371	0,633
Wine	**1**	0,967	0,967	0,922	0,967	0,9	0,833	0,944	0,767	0,9	0,8	0,967	0,5	0,7	0,556
Mean	0,87129	**0,8848**	0,8557	0,8072	0,805	0,8133	0,7829	0,7935	0,6719	0,7437	0,6855	0,6529	0,6607	0,5305	0,5758

5. Continuous and Incremental Quantum Anomaly Detection

In this chapter, we introduces InQMAD. InQMAD: Incremental Quantum Measurement Anomaly Detection is a novel method for detecting anomalies in streaming data. Traditional methods for anomaly detection are not suitable for streaming data due to the need for continuous model updates and high computational costs. InQMAD uses adaptive Fourier features to map the data to a Hilbert space, where a density matrix captures the density of the dataset. The method processes data in a single pass, using a nearly constant memory and inference processing time, making it suitable for streaming data. InQMAD provides a score for each data point, which is used to detect anomalous data. The method is compared with other state-of-the-art methods on various datasets, and the results demonstrate its effectiveness in detecting anomalies in streaming data.

The work presented in this chapter corresponds to the following article:

Gallego-Mejia, J., Bustos-Brinez, O., & González, F. "InQMAD: Incremental Quantum Measurement Anomaly Detection," 2022 IEEE International Conference on Data Mining Workshops (ICDMW), Orlando, FL, USA, 2022, pp. 787-796,
doi: 10.1109/ICDMW58026.2022.00107 [58].

Gallego-Mejia, J., Bustos-Brinez, O., & González, F. "Streaming and Incremental Anomaly Detection through Density Matrices" 2023 Springer Neural Computing and Applications (To Be Submitted).

5.1. Introduction

Anomaly detection is a well-studied problem [26, 158, 134]. The main idea is to detect data points or a group of data points that deviate from a 'normality' in a specific context (note that normality is not related to the Gaussian normal distribution) . This problem arises in several domains, such as network security [124], telecommunications [50], retail industry [132], network traffic [190], financial transactions [4], and wired and wireless sensors [194]. In recent years, particular interest has been given to methods that can deal with problems

where data is continuously generated as a stream rather than as a batch of data points. This behavior is natural in credit card fraud detection [143], network system intrusion detection [124], camera surveillance [177], Internet of Things (IOT) device problems [120], among others. Data in this streaming environment is rapidly generated, potentially infinite, has tremendous volume, and can exhibit concept drift.

An anomaly can be broadly defined as an observation or data that deviates significantly from some kind of normality. Several types of anomalies can occur in real datasets, such as point anomalies, group anomalies, contextual anomalies, low-level texture anomalies, and high-level semantic anomalies. Classical methods have been used to solve this problem, but they suffer with high-dimensional datasets [94]. Most of the recent deep learning methods capture a lot of attention for their good properties, such as automatic feature extraction. However, their training time and streamwise inference is prohibitively long [16].

Classical anomaly detection algorithms in batch environments are based on density estimation, such as kernel density estimation [96], classification, such as one-class support vector machines [115], and distance, such as the isolation forest [111]. Another type of recent solutions is based on deep neural networks that have shown good properties in the anomaly detection task. They are mainly based on variational autoencoders [5], deep belief networks [94], one-class deep networks [25] and adversarial autoencoders [42]. These methods assume that all training data points are available in the training phase. However, in a streaming context the data arrives continuously. Some of these methods have poor adaptability and extensibility, or inability to detect new anomalies continuously, where they have high model update cost and/or slow update speed.

Stream anomaly detection can be viewed as a generalization of the typical anomaly detection problem where data grows infinitely. Therefore, it is impractical, impossible or unnecessary to store every data point that arrives as a stream. In this context, the method has to distinguish between normal and anomalous data points, where concept drift can occur, and the number of anomalies is scarce compared to normal data [43]. Several types of concept drift can arise in streaming datasets, such as sudden, gradual, incremental or recurrent drift. Concept drift occurs in data streams, where usually old data is less important than new data. This trend in data has an evolutionary pattern, where recent behavior should be of greater importance than older patterns [140]. In order to solve this problem, the method must use a constant memory and a nearly constant inference processing time. Therefore, it will process the data in a single pass. In the Subsection 2.2.2, 12 methods for streaming anomaly detection are presented.

To pave the way, several methods have been developed in the last decade. Methods such as iForestASD [43], RCF [71], xStream [116] and Ex. IF [72] present a modification of the Isolation Forest batch anomaly detection method. One of the problems of these methods is how to improve their inference complexity which is related to the depth of the trees which typically will be $log(n)$ where n is the number of data points. To solve this problem, HS-

Tree [179], RS-Hash [160] and MStream [15] use a hash structure to avoid traversing each tree. Other state-of-the-art methods use a modification of the well-studied local outlier factor (LOF) [20]. Methods such as LODA [140], DILOF [128] and MemStream [15] use a modification of the k-nearest neighbor algorithm, the roots of LOF, to score the outlierness of data points. However, some of them are based on a memory that stores m-data points which, viewed as a moving average, may not be able to detect high frequency points or possible outliers.

In this chapter we present the novel method incremental quantum measurement anomaly detection (InQMAD). This method uses adaptive Fourier features to map the original space to a Hilbert space. In this space, a density matrix is used to capture the density of the dataset. A new point passes through each stage and provides a score that is used as an anomaly score in the final stage. An important feature of the method is that it is able to build a density estimation model that is continuously updated with the incoming data, and it has the ability to give more importance to recent samples similar to an exponential moving average. The method works in streaming, is an unsupervised algorithm and retraining can be performed continuously. The proposed method requires constant memory to process new data points, can process data in a single pass, and process potentially endless streaming data or even massive datasets. It can update the model in constant time, and its complexity is $O(1)$.

In summary, the contributions of the present work are:

- **A novel streaming anomaly detection method**: the new method works in a streaming, potentially infinite and with potentially concept drift environment.
- **A systematic evaluation of the proposed method**: the algorithm is evaluated in 12 streaming datasets and compared it against 12 state-of-the-art streaming anomaly detection methods.
- **An ablation study analyzing the new method**: a systematic evaluation of each component of the method is performed.

Reproducibility: the code used in this chapter is released as open source and can be found in https://doi.org/10.5281/zenodo.7183564

The outline of the chapter is as follows: in Section 2, we present the new method, explaining all the stages of the algorithm. In Section 3, we systematically evaluate the proposed method against state-of-the-art streaming anomaly detection algorithms. In Section 4, we state conclusions and outline future lines of research.

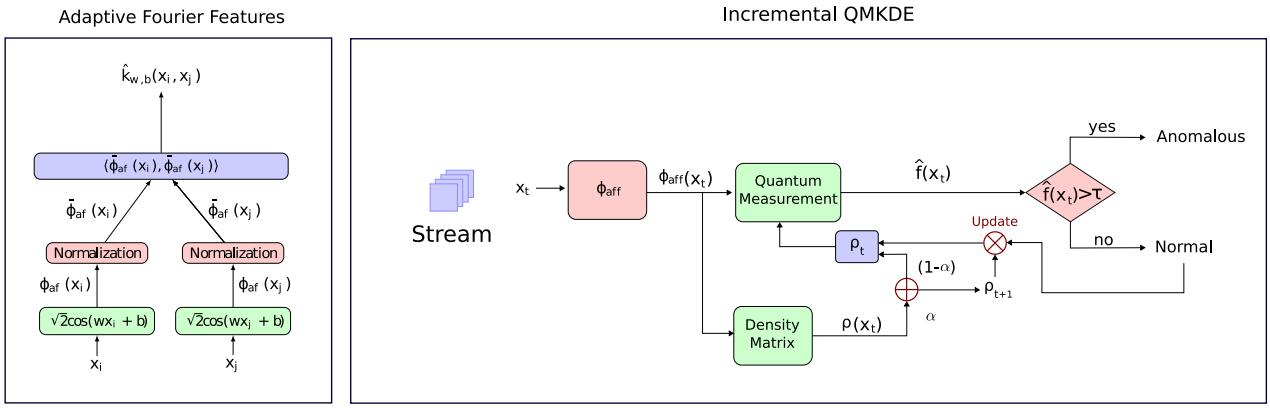

Figure 5-1.: Incremental Quantum Measurement Anomaly Detection (InQMAD) method. The method consists of five steps: (1) an adaptive Fourier features (left subfigure), (2) a density matrix initialization, (3) a quantum measurement of the new streaming data points, (4) a decision anomaly detection threshold for the new data, and (5) an update of the density matrix ρ in case of new normal data points.

5.2. Incremental Quantum Measurement Anomaly Detection (InQMAD)

In this section, we present the new method new Incremental Quantum Measurement Anomaly Detection (InQMAD). Figure 5-1 shows each of the steps of the Algorithms 6 and 7. The method consists of five steps: (1) an adaptive Fourier features, (2) a density matrix initialization, (3) a quantum measurement of the new streaming data points, (4) a decision anomaly detection threshold for the new data, and (5) an update of the density matrix ρ in case of new normal data points. Each stage is explained in detail in this section.

5.2.1. Adaptive Fourier Features

All streaming data points are mapped into a Hilbert space using adaptive Fourier features, first proposed as random Fourier features by [144]. They showed that a Gaussian kernel can be approximated as:

$$k(\boldsymbol{x}, \boldsymbol{y}) \simeq \mathbb{E}_{\boldsymbol{w}}(\langle \hat{\phi}_{\text{rff},\boldsymbol{w}}(\boldsymbol{x}), \hat{\phi}_{\text{rff},\boldsymbol{w}}(\boldsymbol{y}) \rangle) \tag{5-1}$$

where $\hat{\phi}_{\text{rff},\boldsymbol{w}} = \sqrt{2}\cos(\boldsymbol{w}^T\boldsymbol{x} + b)$, $\boldsymbol{w} \sim N(\boldsymbol{0}, \mathbb{I}^d)$ and $\boldsymbol{b} \sim \text{Uniform}(0, 2\pi)$. The expectation is possible to Bochner's Theorem and the fact that using Equation 5-1 that defines a randomized map converge in probability to the Gaussian kernel. However, the randomized map can

be further refined using an optimization step, in particular using gradient descent, as shown in [103, 53]. Given two pairs of random data points ($\boldsymbol{x}_i$, $\boldsymbol{x}_j$), the parameters $\boldsymbol{w}, b$ can be optimized using the following equation:

$$\boldsymbol{w}^*, b^* = \arg\min_{\boldsymbol{w},b} \frac{1}{m} \sum_{\boldsymbol{x}_i,\boldsymbol{x}_j \in s} (k_\sigma(\boldsymbol{x}_i, \boldsymbol{x}_j) - \hat{k}_{\boldsymbol{w},b}(\boldsymbol{x}_i, \boldsymbol{x}_j))^2$$

where $\hat{k}_{\boldsymbol{w},b}(\boldsymbol{x}_i, \boldsymbol{x}_j))^2$ is the Gaussian kernel of Fourier feature approximation and $k_\sigma(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is the Gaussian kernel with a bandwidth parameter σ. Adaptive Fourier features allow us to compute the feature space avoiding the explicit kernel computation and will be used as input to the density matrix in the next step. We performed an empirical evaluation of this improvement algorithm and propose an intermediate enhancement step. Assume that all points are normalized between $[0, 1]^d$, where d is the number of features, generate random samples from Uniform$(-0.5, 1.5)^d$. The increase in the range of the sampling space is to account for future points outside the range. The next question that arises is how many points we should generate. We use an empirical approach depending on how large the initial training dataset is. If the training set is small (< 1000), we will sample until it consists of 10,000 data points. Otherwise, we will sample twice the size of the initial training dataset. The intuition here is that if we have few data points, the algorithm will be prone to overfitting in a local space near initial training dataset. The algorithm 6 shows the step in 6.

5.2.2. Density Matrix Initialization

The second step of the algorithm consists of calculating the density matrix using the mapping obtained by passing each $\boldsymbol{x}_t$ to the adaptive Fourier feature step explained above. The density matrix is a formalism of quantum mechanics that was used as a base tool by [66, 53] to create a density estimation method. The authors derive a new algorithm that uses random Fourier features to store the density matrix ρ. The following equation is a slight modification of the density matrix in terms of stream data:

$$\rho_t = \frac{1}{n} \sum_{i=1}^{N} q_i \cdot \phi_{\text{aff}}(\boldsymbol{x_i}) \phi_{\text{aff}}^t(\boldsymbol{x_i}) \tag{5-2}$$

where $\sum_{i=1}^{t} q_i = 1$. To initially compute the density matrix, an initial portion of the stream $\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\}$ is selected and sent to the equation 5-2 to calculate the density matrix ρ. The initial training dataset size can degrade the final performance of the algorithm; therefore, it is necessary to find it using a cross-validation approach. It should be noted that the density matrix is computed using adaptive Fourier features avoiding explicit kernel computation as in kernel density estimation; however, other feature mappings can be used as shown in [65]. The density matrix is initialized in Algorithm 6 in Step 7.

5.2.3. Quantum Measurement

The third step of the algorithm is quantum measurement. A quantum measurement can be obtained by using the density matrix ρ and mapping a data point $\boldsymbol{x}_{t+1}$ at time $t+1$ as:

$$\hat{f}(\boldsymbol{x}_{t+1}) = \frac{1}{M_\sigma} \phi_{\text{aff}}(\boldsymbol{x}_{t+1})^T \rho_t \, \phi_{\text{aff}}(\boldsymbol{x}) \tag{5-3}$$

where ρ is the density matrix defined in the equation 5-2 and M_σ is a normalization constant. This step gives us an estimate of the density of the given data point that will be used in the next step. The initialized Algorithm 6 uses the quantum measurement in Step 8 and the inference Algorithm 7 uses it in Step 8.

5.2.4. Anomaly Detection Classification

The fourth step of the algorithm is the anomaly detection classification stage. The threshold is defined as τ, where a new point $\boldsymbol{x}_i$ is defined as anomalous according to the following equation:

$$\hat{y}(\boldsymbol{x}_i) = \begin{cases} \text{'normal'} & \textit{if } \hat{f}(\boldsymbol{x}_i) \geq \tau \\ \text{'anomaly'} & \textit{otherwise} \end{cases}$$

After initialization of the density matrix using the initial training dataset, the threshold is found using two different approaches. The first is that the threshold can be found using prior knowledge of the proportion of anomalies (β). The second approach is to use an optimization metric computed over $\{\hat{f}(\boldsymbol{x_1}), \cdots, \hat{f}(\boldsymbol{x_n})\}$, for instance using the best threshold with respect to AUC-ROC in initial memory. The τ is obtained in Step 10 of the Algorithm 6 and the anomaly detection classification occurs in Step 9 of the Algorithm 7.

5.2.5. Density Matrix Update

The fifth and final step of the algorithm is to update the density matrix. If a new point $\boldsymbol{x_t}$ at time t of the flow is classified as a normal data point, the density matrix ρ_{t+1} will be calculated as follows:

$$\rho_{t+1} = (1-\alpha) \cdot \rho_t + \alpha \cdot \phi_{\text{aff}}(\boldsymbol{x}_{t+1}) \phi_{\text{aff}}^T(\boldsymbol{x}_{t+1}) \tag{5-4}$$

Proposition 1. *The resulting matrix ρ_{t+1} in Equation 5-4 is a valid density matrix of the form $\rho_{t+1} = \sum_{i=1}^{t+1} q_i \phi(\boldsymbol{x}_i) \phi^T(\boldsymbol{x}_i)$ with $q_1 = (1-\alpha)^{t+1}$, $q_i = (1-\alpha)^{t-i+1} \cdot \alpha \; \forall i \in \{1, \cdots, t\}$ and $\alpha \in [0, 1]$.*

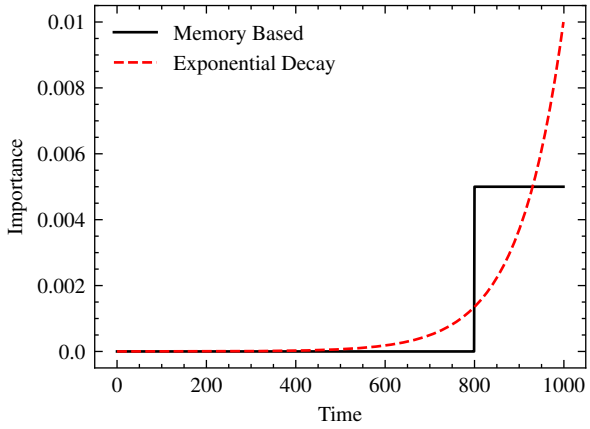

Figure 5-2.: Importance comparison between memory based against exponential decay.

Proposition 1 shows that an exponential decay defines a valid density matrix. This matrix consists of all data points that have reached the method up to time t, where older examples are less important and have a lower weight compared to more recent ones similar to an exponential moving average.

Figure 5-2 shows the difference between a constant memory and an exponential decay method. The figure shows a memory-based algorithm using a window width of 200 and compares it against an exponential decay memory using a $\alpha = \frac{2}{\text{memory window width}}$. The black memory-based line shows a constant memory whose points are of equal importance to the method. The red exponential decay line shows an exponentially decaying method whose points become less and less important as they age. The density matrix update only occurs if the new point is classified as 'normal' and is performed in Step 10 of Algorithm 7.

5.2.6. Complexity Analysis

Proposition 2. *The resulting matrix* $\rho_{t+1} = (1-\alpha)\cdot\rho_t + \alpha\cdot\phi(\boldsymbol{x}_{t+1})\phi^T(\boldsymbol{x}_{t+1})$ *in Equation 5-4 is a valid density matrix of the form* $\rho_{t+1} = \sum_{i=1}^{t+1} q_i\phi(\boldsymbol{x}_i)\phi^T(\boldsymbol{x}_i)$ *with* $q_1 = (1-\alpha)^{t+1}$, $q_i = (1-\alpha)^{t-i+1}\cdot\alpha\ \forall i \in \{1,\cdots,t\}$ *and* $\alpha \in [0,1]$.

Proof. ρ_{t+1} is a valid density matrix if $(1-\alpha)^{t+1} + \sum_{i=1}^{t}(1-\alpha)^{t-i+1}\cdot\alpha = 1$. We give a proof by induction on t.

Base Case: for $t = 1$, define $\rho_t = \phi(\boldsymbol{x}_1)\phi^T(\boldsymbol{x}_1)$. for $t = 2$, define $q_1 = (1-\alpha)$ and $q_2 = \alpha$ then $q_1 + q_2 = 1$

Induction Step: Show that for every $t \geq 2$, if ρ_t holds, then ρ_{t+1} also holds.

Define

$$p_t = (1-\alpha)^t + \alpha \cdot \sum_{i=1}^{t-1} (1-\alpha)^{t-i+1} \tag{5-5}$$

Using $\sum_{k=1}^{n} x^k = (x - x^{n+1})/(1-x)$, then

$$p_t = (1-\alpha)^t + \alpha \cdot (\frac{(1-\alpha) - (1-\alpha)^t}{1-(1-\alpha)}) = 1 \tag{5-6}$$

Now, for t+1 :

$$q_{t+1} = q_1 \cdot \left((1-\alpha)^t + \alpha \cdot \sum_{i=1}^{t-1} (1-\alpha)^{t-i+1} \right) + q_2 \tag{5-7}$$

$$= (1-\alpha) \cdot \left((1-\alpha)^t + \alpha \cdot \sum_{i=1}^{t-1} (1-\alpha)^{t-i+1} \right) + \alpha \tag{5-8}$$

$$= (1-\alpha)^{t+1} + \alpha \cdot \sum_{i=1}^{t} (1-\alpha)^{t-i+1} + \alpha \tag{5-9}$$

$$= (1-\alpha)^{t+1} + \alpha \cdot \sum_{i=0}^{t+1} (1-\alpha)^{t-i+1} \tag{5-10}$$

$$= 1 \tag{5-11}$$

□

In this subsection, d will be the number of features and D will be the number of adaptive Fourier features. For time complexity, in the anomaly detection phase, a dot product is computed between each data point and the adaptive Fourier feature whose time complexity is proportional to $O(dD)$. In addition, a dot product is computed from the above result and the density matrix whose time complexity is proportional to $O(D^2)$. In terms of memory, InQMAD needs to maintain a density matrix whose size is proportional to $O(D^2)$ and stores the weights of the adaptive Fourier features whose size is proportional to $O(dD)$.

5.3. Experimental Evaluation

We designed and tested an experimental setup in order to answer the following questions:

- **Q1. Streaming Method Comparison.** Does our method perform accurately in anomaly detection over streams of data, when compared it against to state-of-the-art baseline methods?

Algorithm 6: InQMAD initialization process

Input: Training dataset $D = \{\boldsymbol{x}_t\}_{t=1,\cdots,n}, \boldsymbol{x}_t \in \mathbb{R}^d$
α: forgetting trade-off,
σ: bandwith parameter
β: proportion of anomalies
Output: $\boldsymbol{w}_{\text{AFF}}, b_{\text{AFF}}, \rho, \tau$
$\boldsymbol{w}^*_{\text{AFF}}, b^*_{\text{AFF}} = \arg\min_{\boldsymbol{w},b} \frac{1}{m} \sum_{\boldsymbol{x}_i,\boldsymbol{x}_j \in D} (k_\sigma(\boldsymbol{x}_i, \boldsymbol{x}_j) - \hat{k}_{\boldsymbol{w},b}(\boldsymbol{x}_i, \boldsymbol{x}_j))^2$ **for** $\boldsymbol{x}_t \in D$ **do**
  $\rho_t = (1-\alpha) \cdot \rho_{t-1} + \alpha \cdot \phi_{\text{aff}}(\boldsymbol{x}_t)\phi^T_{\text{aff}}(\boldsymbol{x}_t)$
  $\hat{f}(\boldsymbol{x}_t) = \frac{1}{M_\sigma}\phi_{\text{aff}}(\boldsymbol{x}_t)^T \rho_t \, \phi_{\text{aff}}(\boldsymbol{x}_t)$
end
$\tau = q_\beta(\hat{f}(\boldsymbol{x}_1), \cdots, \hat{f}(\boldsymbol{x}_n))$
return $\boldsymbol{w}_{\text{AFF}}, b_{\text{AFF}}, \rho_n, \tau$

- **Q2. Adaptability.** How well does our method adapt to "concept drift", that is, sudden changes in the data stream inner structure?
- **Q3. Ablation Study.** What are the effects of removing some stages of the algorithm (in particular, the Adaptive Fourier features embedding) on the overall performance of our method?

5.3.1. Comparison to Streaming Methods

Experimental Setup

The experimental setup presented in this chapter took inspiration from the setup proposed in [16]. There, it was performed a comparison of a dozen algorithms based on area under the ROC curve (commomly known as AUC or AUROC), by applying them over a series of streaming and anomaly detection datasets. Our method was implemented using the JAX framework, a Python library designed for high-performance machine learning. All the experiments were carried out on a machine with a 2.1GHz Intel Xeon 64-Core processor with 128GB RAM and two NVIDIA RTX A5000 graphic processing units, that run Ubuntu 20.04.2 operating system.

To handle the inherent randomness that the proposed method can present in some of its stages (particularly in dataset splitting and neural network training), we selected a unique, invariant value for all the random seeds that affect the behavior of our method.

Algorithm 7: InQMAD inference and density matrix update

Input: $\boldsymbol{x}_{t+1}$
ρ_t: density matrix
α: forgetting trade-off,
$\boldsymbol{w}_{\text{aff}}, b_{\text{aff}}$: adaptive Fourier features parameters
τ: threshold anomaly detector
Output: ρ_{t+1}-quantum measurement KDE parameter,
$\hat{y}_{t+1}$ classification of the given data point
$\hat{f}(\boldsymbol{x}_{t+1}) = \frac{1}{M_\sigma}\phi_{\text{aff}}(\boldsymbol{x}_{t+1})^T \rho_t \, \phi_{\text{aff}}(\boldsymbol{x}_{t+1})$
if $\hat{f}(\boldsymbol{x}_{t+1}) \geq \tau$ **then**
    $\rho_{t+1} = (1-\alpha) \cdot \rho_{t-1} + \alpha \cdot \phi_{\text{aff}}(\boldsymbol{x}_{t+1}) \phi_{\text{aff}}^T(\boldsymbol{x}_{t+1})$
    $\hat{y}_{t+1} = `normal'$
else
    $\rho_{t+1} = \rho_t$
    $\hat{y}_{t+1} = `anomaly'$
end
return $\rho_{t+1}, \hat{y}_{t+1}$

Datasets

In this framework, we selected twelve different datasets, that can be divided into two groups: seven datasets (Cardio, Ionosphere, Mammography, Pima, Satellite, Satimage and Synthetic) with relatively low dimensions and low number of records, mainly used to perform proof-of-concept in anomaly detection, and five datasets (KDD99, NSL, DoS, UNSW and Cover) with hundreds of thousands of registers and a high number of dimensions, that require an intensive use of resources to be processed. A summary of the main features of all the datasets can be seen in Table **5-1**, and a brief description of each dataset is presented in the Supplemental Material.

We present a summarized description of the datasets used in the experimental setup to apply the algorithms on.

- Cardio [150]: consists of measurements of fetal heart rate, where the original classes are normal, suspect, and pathologic; to adapt it to outlier detection, the normal class formed the inliers and the pathologic class is down sampled and labeled as outliers, while the suspect class is discarded.
- Ionosphere [150]: originally a binary classification dataset from UCI ML repository, the 'bad' class is considered as outliers and the 'good' class as inliers.
- Mammography [150]: an open dataset about breast calcification, for outlier detection tasks the minority class of 'calcification' is considered as outliers and the 'non-

calcification' class as inliers.

- Pima [150]: also from UCI ML repository, it includes data of female Indian patients with the objective of classifying them as diabetic or healthy.
- Satellite [150]: derived from the Statlog dataset from UCI ML repository, it is a multi-class dataset. For anomaly detection, the smallest three classes (2, 4, 5) are combined to form the outliers, while all the other classes are combined to form the inliers.
- Satimage [150]: coming from the previous Satellite dataset, here class 2 has been down sampled and considered as outliers, while the other classes are labeled as normal data. This dataset and the latter came from ODDS virtual library.
- Synthetic: this dataset was created in [16] to analyze if their method could afford sudden changes in data distribution. For this, the authors of the dataset combined two sinusoidal waves, and contaminated 10% of the samples by adding Gaussian noise to simulate anomalous data.
- KDD99 [1]: one of the best-known datasets in anomaly detection, the original dataset contains 34 numerical dimensions and 7 categorical dimensions, that were transformed using one-hot encoding to obtain a dataset of 121 dimensions. We treat normal data as outliers in this experiment, given the fact that only 20% of all records are labeled as normal.
- NSL [181]: coming from the previous KDD dataset, it adds some dimensions and solves redundant and duplicate records.
- Cover [150]: originally a multiclass dataset from UCI ML repository, it is used to predict forest cover type from wilderness areas in Colorado. Instances from class 2 are considered as normal and instances from class 4 are labeled as anomalies. Instances from other classes are omitted.
- DoS [76]: this dataset was created by the Canadian Institute of Cybersecurity. Each record corresponds to a network package, and they were captured from simulations of normal network traffic and synthetic attacks.
- UNSW [125]: created by ACCS (an Australian institute of computer science and cybersecurity), it contains both real network normal activities and synthetic attacks. Originally it included nine types of attacks. It has 13% of anomalies.

Parameter Search

The behavior of InQMAD depends on a series of parameters that regulate the Adaptive Fourier features embedding, the density estimation stage and the size of the initialization dataset of the method. Particularly, we tried to find which of these parameters had a larger impact over the performance of the algorithm, and we selected four to carry out a parameter

Table 5-1.: Main features of the datasets

Dataset	Records	Dimensions
Cardio	1831	21
Ionosphere	351	33
Mammography	11183	6
Pima	768	8
Satellite	6435	36
Satimage	5803	36
Synthetic	10000	1
KDD99	494021	121
NSL	125973	126
Cover	286048	10
DoS	1048575	95
UNSW	2540044	122

grid search over them in order to find the combination of parameters that showed the best performance. The selected parameters are the following:

- n: this corresponds to the size of the initial training dataset D, that is used in the initialization stage of the method. The n samples allow the construction of the first density matrix ρ_n, and thus they indirectly influence the subsequent behavior of the method. We searched this value in the set $\{64, 128, 256, 512, 1000, 2000, 2048, 5000\}$, taking into account that it cannot be bigger than the total of samples in the dataset.
- lr_{base}: since the construction of the adaptive Fourier features parameters $\boldsymbol{w}_{\text{aff}}$ and b_{aff} involves the training of a neural network, we searched the best possible values for the learning rate of this process. We designed the learning rate to follow a polynomial decay, so we needed a start point and an end point for it. With the end point (lr_{end}) fixed on the value 10^{-7}, we searched the start point (lr_{base}) over the values $\{10^{-2}, 10^{-3}, 10^{-4}\}$.
- σ (of kernel): we used a Gaussian kernel at the core of the KDE stage of the method. The shape of this kernel depends on the variance value, commonly notated as σ^2, and it has a notable influence over the quality of the density estimation. Since the variance is related to the structure of the data, we chose a different set of values for each dataset.
- α: this parameter controls the tradeoff between the samples that are stored into the density matrix of the method and the incoming samples that can substitute them. This parameter can only take values between zero and one, and bigger values correspond to bigger substitution rates on the memory. We searched this parameter over the range $[0.001, 1)$.

The winning combinations of parameters for each dataset can be found in the Supplemental Material.

Evaluation Metrics

Following the framework established in [16], we selected the area under the ROC curve (also known as AUC or AUC-ROC) as the main metric to establish the performance of our algorithm and compare it with the baseline methods. This metric was chosen because it represents the overall capability of the model to distinguish between the classes, regardless of the thresholds used in specific situations.

Results and discussion

The AUCROC score obtained for every pair of dataset and algorithm is presented in Table **5-2**. The metrics corresponding to our method can be found in the column labeled InQMAD. The absent values correspond to cases where the algorithms were not able to handle that particular dataset. For each row, the best value is marked in bold, the second is underlined and the third is marked in italics.

When looking at the average performance, InQMAD is the best method, being slightly better than MemStream, and notably better than all other methods, thus showing a noticeable improvement over the state-of-the-art algorithms in the area. When considering the size of the datasets, there is a clear advantage of InQMAD over the smaller datasets (the ones in the top rows of Table I), due to the fact that it shows the best performance for all of these datasets. For bigger datasets (the ones in the bottom rows of Table I), it is still competitive, but some algorithms do better, mainly in the biggest datasets like DoS or UNSW.

On the other hand, the number of dimensions does not seem to be as strongly related to the performance of our method as the size, considering that it had a mixed behavior for datasets with low dimensionality (high for Pima or Satellite, lower for Cover) and for datasets with high dimensions (high for KDD or NSL, lower for UNSW). In general, memory-based methods, such as MemStream, underperform InQMAD. Our intuition for this behavior is that memory-based algorithms are similar to moving averages that may be prone to a high frequency point or outlier. However, InQMAD can be considered an exponential moving average that dampens the weight of high frequency points.

Table 5-2.: Area under the ROC curve (AUCROC) for all algorithms over all datasets. The first, second and third positions are respectively highlighted in bold, underlined and italics.

Dataset	STORM	HS-Tree	iForestASD	RS-Hash	RCF	LODA	Kitsune	DILOF	xStream	Mstream	Ex.IF	MemStream	InQMAD
Cardio	0,507	0,673	0,515	0,532	0,617	0,501	*0,966*	0,570	0,918	0,986	0,921	0,884	**0,989**
Cover	0,778	0,731	0,603	0,640	0,586	0,500	0,888	0,688	0,894	0,874	*0,902*	**0,952**	0,923
DoS	0,511	0,707	0,529	0,527	0,514	0,500	*0,907*	0,613	0,800	0,930	0,734	**0,938**	0,788
Ionosphere	0,637	0,764	0,694	0,772	0,675	0,503	0,514	**0,928**	0,847	0,670	*0,872*	0,821	**0,928**
KDD	0,914	0,912	0,575	0,859	0,791	0,500	0,525	0,535	*0,957*	0,844	0,874	0,980	**0,995**
Mammo	0,650	0,832	0,574	0,773	0,755	0,500	0,592	0,733	0,856	0,567	*0,867*	0,894	**0,914**
NSL	0,504	*0,845*	0,500	0,701	0,745	0,500	0,659	0,821	0,552	0,544	0,767	0,978	**0,982**
Pima	0,528	0,667	0,525	0,562	0,571	0,502	0,511	0,543	0,663	0,529	*0,672*	0,742	**0,750**
Satellite	0,662	0,519	0,504	0,675	0,552	0,500	0,665	0,561	0,677	0,563	*0,716*	0,727	**0,764**
Satimage	0,514	0,929	0,554	0,685	0,738	0,500	0,973	0,563	**0,996**	0,958	*0,995*	0,991	**0,996**
Syn	0,910	0,800	0,501	*0,921*	0,774	0,506	-	0,703	0,539	0,505	-	0,955	**0,982**
UNSW	0,810	0,769	0,557	0,778	0,512	-	0,794	0,737	0,804	*0,860*	0,541	**0,972**	0,873

5.3.2. Ablation Study

Being one of the most important stages of our algorithm, we wanted to determine the degree of influence of the Adaptive Fourier Features embedding over the algorithm performance. This comparison was made by building alternate versions of our method: InQM-NoAdp, that instead of using adaptive Fourier features as mapping, uses the random Fourier features approach stated in previous works like [66], and InQMAD-200, where the number refers to the size of the encoding given by adaptive Fourier features. Since the size of the encoding was chosen to be 2000 in all the experiments of InQMAD, we expect to determine if the size of the adaptive features encoding enhances or diminishes the overall performance of the method.

Using these different versions of our method over the datasets in the experimental setup, we calculated the metrics that we present in Table **5-3**. In order to make a fair comparison, in every dataset we used the same parameters for all the versions. The best result of each row is marked in bold.

From these results, there is a clear advantage in using the adaptive Fourier Features in the majority of datasets, regardless of their size or dimensionality. Only in one of the datasets (DoS) there is a small decrease in performance when using many Adaptive features, in favor of a lower embedding size. Although, the difference in this case is way smaller than the differences in favor of the use of Adaptive features on other datasets, particularly Ionosphere, Syn and Satellite.

Table 5-3.: Results for Ablation Study on InQMAD, including Adaptive and NoAdaptive versions

DATASET	AUCROC		
	InQM-NoAdp	InQMAD-200	InQMAD
Cardio	0,976	0,974	**0,989**
Cover	0,847	0,899	**0,923**
DoS	0,767	**0,808**	0,788
Ionosphere	0,825	0,852	**0,928**
KDD	0,937	0,943	**0,995**
Mammo	0,907	0,912	**0,914**
NSL	0,947	0,96	**0,982**
Pima	0,728	0,737	**0,75**
Satellite	0,67	0,694	**0,764**
Satimage	0,965	0,977	**0,996**
Syn	0,928	0,929	**0,982**
UNSW	0,798	0,836	**0,873**

5.3.3. Parameters

The winning combinations of parameters for our method are presented in Table 5-4. For each dataset, their respective parameters are presented as an array.

The parameters selected to run the baseline methods were the same for all datasets and follow the recommended values in the original proposal of each method. Their values are presented in Table 5-5.

Table 5-4.: Best combinations of InQMAD parameters for each dataset

Dataset	Winning Parameters
	[n, lr_{base}, sigma σ, alpha α]
Cardio	[256 0,001 3,0 0,99]
Cover	[1000 0,01 2,0 0,05]
DoS	[2048 0,001 0,11 0,40]
Ionosphere	[100 0,001 0,9 0,40]
KDD	[5000 0,001 2,0 0,5]
Mammo	[512 0,001 4,0 0,80]
NSL	[2000 0,01 1,25 0,25]
Pima	[64 0,001 0,5 0,95]
Satellite	[128 0,001 0,7 0,04]
Satimage	[256 0,0001 0,8 0,005]
Syn	[64 0,01 0,1 0,04]
UNSW	[2000 0,001 2,5 0,6]

Table 5-5.: Best parameters for baseline methods

Method	Winning Parameters
STORM	$window\_size = 10000, max\_radius = 0.1$
HS-Tree	$window\_size = 100, num\_trees = 25,$ $max\_depth = 15, initial\_window\_X = None$
iForestASD	$window\_size = 100, n\_estimators = 25,$ $anomaly\_threshold = 0.5, drift\_threshold = 0.5$
RS-Hash	$sampling\_points = 1000, decay = 0.015,$ $num\_components = 100, num\_hash\_fns = 1$
RCF	$num\_trees = 4, shingle\_size = 4,$ $tree\_size = 256$
LODA	$num\_bins = 10, num\_random\_cuts = 100$
Kitsune	$max\_size\_ae = 10, learning\_rate = 0.1,$ $hidden\_ratio = 0.75, grace\_feature\_mapping = 0.1$
DILOF	$window\_size = 400, K = 8,$ $thresholds = [0.1, 1.0, 1.1, 1.15, 1.2, 1.3, 1.4, 1.6, 2.0, 3.0]$
xStream	$projection\_size = 50, number\_chains = 50,$ $depth = 10, rowstream = 0, nwindows = 0,$ $scoring\_batch\_size = 100000$
Mstream	$alpha = 0.85$
Ex. IF	$n\_trees = 200, sample\_size = 256,$ $limit = None, Extension\_Level = 1$
MemStream	$memory\_length = [4, 16, 32, 64, 128, 256, 1024, 2048],$ $update\_threshold = [10, 1, 0.1, 0.01, 0.001, 0.0001]$

5.3.4. Statistical Tests

Using the Table 5-2 as starting point, the Friedman test, a well-known statistical method to compare different populations or groups, was applied in order to determine whether there are statistically significant differences between the different methods. The Friedman test uses the following formula:

$$Q = \left[\frac{12}{Nk(k+1)} \sum_{j=1}^{k} R_j^2 \right] - 3N(k+1)$$

where N is the number of datasets, k is the number of algorithms, and R_j^2 is the squared sum of the observations for each particular algorithm. Given a confidence value of 0.05 and the p-value given by $\mathbf{P}[\chi_{n-1}^2 \geq Q]$, if the p-value is lower than the confidence value, there is statistically significant evidence supporting that the methods are different. Applying the test returns a p-value of 3.06×10^{-14}, clearly showing that such difference does exist.

The Friedman test does not indicate which of the methods are responsible for this difference, so to compare them two by two, we use the Friedman-Nemenyi test, applying it over all pairs of methods. This test generates coefficients for each pair of datasets that indicate if they are different (near to 0) or not (near to 1).

Figure 5-3 shows the results of Friedman-Nemenyi test, where black squares correspond to pairs of datasets that differ significantly, and white squares correspond to pairs that do not. The most different methods are LODA (due to their poor results) and InQMAD (the best overall method). Other methods that differ notably from others include MemStream (the second-best method) and iForestASD (the second worst method). The remaining methods differ with others in only one or two cases.

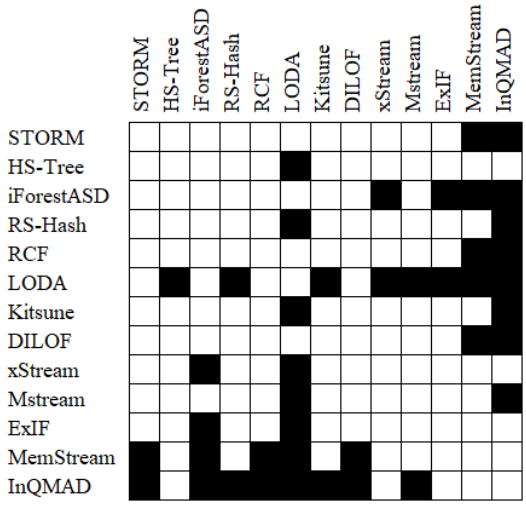

Figure 5-3.: Results of Friedman-Nemenyi test, performed on AUCROC performances of all methods over all datasets.

5.4. Conclusion

In this chapter, we present a new method for data stream anomaly detection called Incremental Quantum Measurement Anomaly Detection (InQMAD). The new method uses the adaptive Fourier feature to map the data to a higher space and density matrices to capture the density estimate. The new method has an initial linear training complexity in terms of the number of training data points. In addition, it has a linear update complexity, which makes it suitable for streaming problems. A systematic evaluation has been performed against 12 state-of-the-art methods on 12 streaming datasets, showing similar performance. We give a theoretical guarantee that the update stage of the algorithm generates a valid density matrix. In addition, an ablation study shows the importance of the new empirical random sampling applied to the adaptive Fourier features and the importance of the update memory parameter α. For future research, we will use dimensionality reduction steps such as principal component analysis or autoencoders to help with the curse of dimensionality in massive datasets.

6. Conclusions and Perspectives

In conclusion, this PhD thesis has presented several innovative methods for density estimation and anomaly detection in machine learning. The proposed methods combine classical statistical foundations with the representation-learning abilities of deep-learning models, and leverage techniques such as density matrices, random Fourier features, and adaptive Fourier features. These methods have been systematically evaluated on various benchmark datasets and have demonstrated competitive performance compared to state-of-the-art methods. Moreover, the thesis has introduced a quantum-inspired approach to density estimation that can be coupled with deep neural networks, which has shown promise for future applications in machine learning and statistical applications.

The proposed method DEMANDE density matrices and adaptive Fourier features offers a quantum-inspired approach to density estimation that can be coupled with deep neural networks. The method has demonstrated competitive performance when evaluated on various synthetic and real datasets, making it a promising alternative for machine learning and statistical applications. In addition, the method has several advantages over traditional density estimation methods. For example, the method does not require any prior knowledge of the data distribution or the choice of an appropriate kernel function, making it a more flexible and adaptable method. Furthermore, the method can efficiently capture the complex relationships between data points, even in high dimensions, due to the use of adaptive Fourier features.

Another advantage of the proposed method is that it can be trained without optimization using only explicit summation. This makes the method computationally efficient and allows for easy integration with deep learning architectures, which have become increasingly popular in recent years. The method can also be trained using gradient descent, which is a widely used optimization technique in machine learning. This allows for the method to be fine-tuned and optimized for specific applications, further improving its performance. Overall, the proposed method offers a new approach to density estimation that combines the strengths of quantum-inspired density matrices with the flexibility and adaptability of deep neural networks. This approach has demonstrated competitive performance on a range of synthetic and real datasets, making it a promising alternative for a variety of machine learning and statistical applications. In addition, the method has several advantages over traditional density estimation methods, including its flexibility, efficiency, and ability to capture complex relationships between data points.

Besides, we introduced a powerful new approach to anomaly detection that combines the strengths of traditional density-estimation-based methods with the representation-learning abilities of deep learning models. By using an autoencoder to learn a low-dimensional representation of the data and combining it with a density-estimation model based on random Fourier features and density matrices, the method is able to predict the degree of normality for new samples based on the estimated density. What's more, this approach addresses the challenges posed by streaming anomaly detection by providing a new incremental anomaly detection method that performs continuous density estimation using quantum measurements and density matrices. This method can process potentially endless data, and its update complexity is constant O(1). The systematic evaluation of this approach shows that this method is not only effective but also outperforms other state-of-the-art methods. This research has significant implications for a range of industries and applications that require real-time detection of anomalous events or data points in a continuous stream, from healthcare to finance, to cybersecurity and beyond.

One possible future research direction could be exploring the effectiveness of different methods for anomaly detection using advanced techniques such as Kernel Quantum Measurement (KQM) or a few adaptive Fourier features. The first approach could involve using KQM with robust kernels to enhance the detection of anomalies in complex datasets with noisy or corrupted samples. Additionally, the use of incremental quantum latent measurement through denoising autoencoder could be explored as a way to optimize the detection of anomalies in real-time scenarios where data is continuously evolving. Another potential direction could be focused on window concept drift incremental quantum anomaly detection, which could improve the detection of changes in data distributions over time. Furthermore, KQM density estimation with robust kernels and adaptive positive Fourier kernel density estimation could be used to estimate the probability density function of the data, which could be useful in many anomaly detection applications. Finally, the use of robust density estimation methods such as kenel density estimation (KDE), density matrix kernel density estimation (DEMANDE), or contaminated KDE with robust kernels could help to improve the accuracy and reliability of anomaly detection algorithms in the presence of outliers or other types of contamination in the data.

Overall, the contributions presented in this thesis represent significant advances in the fields of density estimation and anomaly detection and provide new avenues for further research in the field.

Bibliography

[1] Kdd cup dataset, 1999. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[2] Andrews Jerone T. A., Edward J Morton, and Lewis D Griffin. Detecting anomalous data using auto-encoders. *International Journal of Machine Learning and Computing*, 2016.

[3] Charu C Aggarwal. Outlier analysis second edition, 2016.

[4] Mohiuddin Ahmed, Abdun Naser Mahmood, and Md Rafiqul Islam. A survey of anomaly detection techniques in financial domain. *Future Generation Computer Systems*, 55:278–288, 2016.

[5] Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2(1):1–18, 2015.

[6] Tessa K. Anderson. Kernel density estimation and K-means clustering to profile road accident hotspots. *Accident Analysis and Prevention*, 41(3):359–364, may 2009. ISSN 00014575. doi: 10.1016/j.aap.2008.12.014.

[7] Jerone T A Andrews, Thomas Tanay, Edward J Morton, and Lewis D Griffin. Transfer representation-learning for anomaly detection, 2016. URL http://www.vlfeat.org/matconvnet.

[8] Fabrizio Angiulli and Fabio Fassetti. Detecting distance-based outliers in streams of data. pages 811–820, 2007. ISBN 9781595938039. doi: 10.1145/1321440.1321552.

[9] Haim Avron, Vikas Sindhwani, Jiyan Yang, and Michael W. Mahoney. Quasi-monte carlo feature maps for shift-invariant kernels. *Journal of Machine Learning Research*, 17(120):1–38, 2016. URL http://jmlr.org/papers/v17/14-538.html.

[10] Francis R Bach and Michael I Jordan. Predictive low-rank decomposition for kernel methods. In *ICML 2005 - Proceedings of the 22nd International Conference on Machine Learning*, pages 33–40, 2005. ISBN 1595931805. doi: 10.1145/1102351.1102356.

[11] Arturs Backurs, Piotr Indyk, and Tal Wagner. Space and time efficient kernel density estimation in high dimensions. volume 32, 2019.

[12] Sivaraman Balakrishnan, Srivatsan Narayanan, Alessandro Rinaldo, Aarti Singh, and Larry Wasserman. Cluster trees on manifolds. *arXiv preprint arXiv:1307.6515*, 2013.

[13] David M Bashtannyk and Rob J Hyndman. Bandwidth selection for kernel conditional density estimation, 2001. URL www.elsevier.com/locate/csda.

[14] Yoshua Bengio and Samy Bengio. Modeling high-dimensional discrete data with multi-layer neural networks. *Advances in Neural Information Processing Systems*, 12, 1999.

[15] Siddharth Bhatia, Arjit Jain, Pan Li, Ritesh Kumar, and Bryan Hooi. Mstream: Fast anomaly detection in multi-aspect streams. pages 3371–3382. Association for Computing Machinery, Inc, 4 2021. ISBN 9781450383127. doi: 10.1145/3442381.3450023.

[16] Siddharth Bhatia, Arjit Jain, Shivin Srivastava, Kenji Kawaguchi, and Bryan Hooi. Memstream: Memory-based streaming anomaly detection. *WWW 2022 - Proceedings of the ACM Web Conference 2022*, pages 610–621, 2022. doi: 10.1145/3485447.3512221.

[17] Peter J Bickel and Kjell A Doksum. *Mathematical statistics: basic ideas and selected topics, volumes I-II package*. CRC Press, 2015.

[18] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[19] Giuseppe Borruso. Network Density Estimation: A GIS Approach for Analysing Point Patterns in a Network Space. *Transactions in GIS*, 12(3):377–402, 2008. ISSN 1361-1682.

[20] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

[21] Brian Bullins, Cyril Zhang, and Yi Zhang. Not-so-random features. 10 2017. URL http://arxiv.org/abs/1710.10230.

[22] Oscar Bustos-Brinez, Joseph Gallego-Mejia, and Fabio A González. Ad-dmkde: Anomaly detection through density matrices and fourier features. *arXiv preprint arXiv:2210.14796*, 2022.

[23] Chensi Cao, Feng Liu, Hai Tan, Deshou Song, Wenjie Shu, Weizhong Li, Yiming Zhou, Xiaochen Bo, and Zhi Xie. Deep learning and its applications in biomedicine. *Genomics, Proteomics and Bioinformatics*, 16:17–32, 2 2018. ISSN 22103244. doi: 10.1016/j.gpb.2017.07.003.

[24] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. 1 2019. URL http://arxiv.org/abs/1901.03407.

[25] Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla. Anomaly detection using one-class neural networks. *arXiv preprint arXiv:1802.06360*, 2018.

[26] Varun Chandola. Anomaly detection : A survey, 2009.

[27] Moses Charikar and Paris Siminelakis. Hashing-based-estimators for kernel density in high dimensions. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1032–1043. IEEE, 2017.

[28] Sotirios P. Chatzis, Dimitrios Korkinof, and Yiannis Demiris. A quantum-statistical approach toward robot learning by demonstration. *IEEE Transactions on Robotics*, 28:1371–1381, 2012. ISSN 15523098. doi: 10.1109/TRO.2012.2203055.

[29] Frédéric Chazal, Brittany Fasy, Fabrizio Lecci, Bertrand Michel, Alessandro Rinaldo, Alessandro Rinaldo, and Larry Wasserman. Robust topological inference: Distance to a measure and kernel distance. *The Journal of Machine Learning Research*, 18(1): 5845–5884, 2017.

[30] Gal Chechik, Varun Sharma, Uri Shalit, Samy Bengio, and Samy Bengio CHECHIK. Large Scale Online Learning of Image Similarity Through Ranking. Technical report, 2010.

[31] Yen Chi Chen. A tutorial on kernel density estimation and recent advances. *Biostatistics and Epidemiology*, 1:161–187, 1 2017. ISSN 24709379. doi: 10.1080/24709360.2017.1396742.

[32] Yen-Chi Chen, Christopher R Genovese, Larry Wasserman, et al. A comprehensive approach to mode clustering. *Electronic Journal of Statistics*, 10(1):210–241, 2016.

[33] Peitao Cheng, Yuanying Qiu, Xiumei Wang, and Ke Zhao. A New Single Image Super-Resolution Method Based on the Infinite Mixture Model. *IEEE Access*, 5:2228–2240, 2017. ISSN 21693536. doi: 10.1109/ACCESS.2017.2664103. URL http://arxiv.org/abs/2006.05218.

[34] Victor Chernozhukov, Denis Chetverikov, Kengo Kato, et al. Gaussian approximation of suprema of empirical processes. *Annals of Statistics*, 42(4):1564–1597, 2014.

[35] Krzysztof Choromanski and Vikas Sindhwani. Recycling randomness with structure for sublinear time kernel expansions. 5 2016. URL http://arxiv.org/abs/1605.09049.

[36] N Cristianini. *Kernel Methods for Pattern Analysis*, volume 1. 2004. doi: 10.1198/tech.2005.s264.

[37] Tri Dao, Christopher De Sa, and Christopher Ré. Gaussian quadrature for kernel features. *Advances in neural information processing systems*, 30:6109, 2017.

[38] Philip I Davies and Nicholas J Higham. Numerically stable generation of correlation matrices and their factors. *BIT Numerical Mathematics*, 40(4):640–651, 2000.

[39] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1):1–22, 1977.

[40] B Denkena, M-A Dittrich, H Noske, and M Witt. Statistical approaches for semi-supervised anomaly detection in machining. *Production Engineering*, 14(3):385–393, 2020.

[41] Luc Devroye. Nonparametric density estimation. *The L_1 View*, 1985.

[42] Asimenia Dimokranitou. *Adversarial autoencoders for anomalous event detection in images*. PhD thesis, Purdue University, 2017.

[43] Zhiguo Ding and Minrui Fei. An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window. volume 3, pages 12–17. IFAC Secretariat, 2013. ISBN 9783902823458. doi: 10.3182/20130902-3-CN-3020.00044.

[44] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[45] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. 5 2016. URL http://arxiv.org/abs/1605.08803.

[46] Joni A Downs. Time-geographic density estimation for moving point objects, 2010.

[47] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[48] Vincent Dutordoir, Hugh Salimbeni, Marc Peter Deisenroth, and James Hensman. Gaussian process conditional density estimation. volume 2018-Decem, pages 2385–2395, 2018.

[49] Bradley Efron. Bootstrap methods: another look at the jackknife. In *Breakthroughs in statistics*, pages 569–593. Springer, 1992.

[50] Gilberto Fernandes, Joel JPC Rodrigues, Luiz Fernando Carvalho, Jalal F Al-Muhtadi, and Mario Lemes Proença. A comprehensive survey on network anomaly detection. *Telecommunication Systems*, 70(3):447–489, 2019.

[51] Chris Fraley and Adrian E. Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97:611–631, 2002. ISSN 01621459. doi: 10.1198/016214502760047131.

[52] Brendan J Frey, J Frey Brendan, and Brendan J Frey. *Graphical models for machine learning and digital communication*. MIT press, 1998.

[53] Joseph A. Gallego and Fabio A. González. Quantum adaptive fourier features for neural density estimation, 2022. URL https://arxiv.org/abs/2208.00564.

[54] Joseph A Gallego, Fabio A González, and Olfa Nasraoui. Robust kernels for robust location estimation. *Neurocomputing*, 429:174–186, 2021.

[55] Joseph A Gallego, Juan F Osorio, and Fabio A Gonzalez. Fast kernel density estimation with density matrices and random fourier features. In *Advances in Artificial Intelligence–IBERAMIA 2022: 17th Ibero-American Conference on AI, Cartagena de Indias, Colombia, November 23–25, 2022, Proceedings*, pages 160–172. Springer, 2023.

[56] Joseph Gallego-Mejia, Oscar Bustos-Brinez, and Fabio A González. Leandmkde: Quantum latent density estimation for anomaly detection. *arXiv preprint arXiv:2211.08525*, 2022.

[57] Joseph A. Gallego-Mejia and Fabio A Gonzalez. Demande dataset, April 2023. URL https://doi.org/10.5281/zenodo.7822851.

[58] Joseph A Gallego-Mejia, Oscar A Bustos-Brinez, and Fabio A González. Inqmad: Incremental quantum measurement anomaly detection. In *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 787–796. IEEE, 2022.

[59] Weihao Gao, Sreeram Kannan, Sewoong Oh, and Pramod Viswanath. Estimating mutual information for discrete-continuous mixtures. *Advances in neural information processing systems*, 30, 2017.

[60] Ioannis Gatopoulos, Maarten Stol, and Jakub M. Tomczak. Super-resolution Variational Auto-Encoders. jun 2020. URL http://arxiv.org/abs/2006.05218.

[61] Christopher R Genovese, Marco Perone-Pacifico, Isabella Verdinelli, Larry Wasserman, et al. Nonparametric ridge estimation. *Annals of Statistics*, 42(4):1511–1545, 2014.

[62] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation, 2015.

[63] Kaan Gokcesu and Suleyman S. Kozat. Online density estimation of nonstationary sources using exponential family of distributions. *IEEE Transactions on Neural Networks and Learning Systems*, 29:4473–4478, 9 2018. ISSN 21622388. doi: 10.1109/TNNLS.2017.2740003.

[64] Fabio A. González, Vladimir Vargas-Calderón, and Herbert Vinck-Posada. Supervised Learning with Quantum Measurements. 2020. URL http://arxiv.org/abs/2004.01227.

[65] Fabio A González, Vladimir Vargas-Calderón, and Herbert Vinck-Posada. Classification with quantum measurements. *Journal of the Physical Society of Japan*, 90(4): 044002, 2021.

[66] Fabio A. González, Alejandro Gallego, Santiago Toledo-Cortés, and Vladimir Vargas-Calderón. Learning with density matrices and random features, 2021.

[67] Artur Gramacki. Nonparametric kernel density estimation and its computational aspects, 2018.

[68] Artur Gramacki. *Nonparametric kernel density estimation and its computational aspects*. Springer, 2018.

[69] Alexander G Gray and Andrew W Moore. Nonparametric density estimation: Toward computational tractability. In *Proceedings of the 2003 SIAM International Conference on Data Mining*, pages 203–211. SIAM, 2003.

[70] Claudio Guardnaccia, Michele Grimaldi, Gabriella Graziuso, and Simona Mancini. CROWDSOURCING NOISE MAPS ANALYSIS BY MEANS OF KERNEL DENSITY ESTIMATION. pages 1691–1697, 2021. doi: 10.48465/fa.2020.0505. URL https://hal.archives-ouvertes.fr/hal-03233732.

[71] Sudipto Guha, Nina Mishra, Gourav Roy, and Okke Schrijvers. Robust random cut forest based anomaly detection on streams, 2016.

[72] Sahand Hariri, Matias Carrasco Kind, and Robert J. Brunner. Extended isolation forest. 11 2018. doi: 10.1109/TKDE.2019.2947676. URL http://arxiv.org/abs/1811.02141http://dx.doi.org/10.1109/TKDE.2019.2947676.

[73] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[74] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[75] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[76] A. H. Lashkari I. Sharafaldin and A. A Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization, 2018.

[77] Félix Iglesias and Tanja Zseby. Analysis of network traffic features for anomaly detection. *Machine Learning*, 101:59–84, 10 2015. ISSN 15730565. doi: 10.1007/s10994-014-5473-9.

[78] Piotr Indyk and Rajeev Motwd. Approximate nearest neighbors: Towards removing the curse of dimensionality, 1998.

[79] Marko V Jankovic. Probabilistic Approach to Neural Networks Computation Based on Quantum Probability Model Probabilistic Principal Subspace Analysis Example. 2010. URL http://arxiv.org/abs/1001.4301.

[80] Marko V. Jankovic. Quantum Low Entropy based Associative Reasoning – QLEAR Learning, 2017. ISSN 23318422.

[81] Marko V. Jankovic and Masashi Sugiyama. Probabilistic principal component analysis based on joystick probability selector. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1414–1421. IEEE, 2009. ISBN 9781424435531. doi: 10.1109/IJCNN.2009.5178696.

[82] Marko V. Janković, Tomislav Gajić, and Branimir D. Reljin. Applications of probabilistic model based on main quantum mechanics concepts. In *12th Symposium on Neural Network Applications in Electrical Engineering, NEUREL 2014 - Proceedings*, pages 33–36, 2014. ISBN 9781479958887. doi: 10.1109/NEUREL.2014.7011453.

[83] J.H.M. Janssens, F. Huszar, E.O. Postma, and H.J. van den Herik. Stochastic outlier selection, 2012.

[84] Ping Ji, Na Zhao, Shijie Hao, and Jianguo Jiang. Automatic image annotation by semi-supervised manifold kernel density estimation. *Information Sciences*, 281:648–660, 10 2014. ISSN 00200255. doi: 10.1016/j.ins.2013.09.016.

[85] Joagg. Joaggi/demande: v1.0, March 2023. URL https://doi.org/10.5281/zenodo.7709634.

[86] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[87] Vilen Jumutc and Johan A.K. Suykens. Multi-class supervised novelty detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36:2510–2523, 12 2014. ISSN 01628828. doi: 10.1109/TPAMI.2014.2327984.

[88] Firuz Kamalov. Kernel density estimation based sampling for imbalanced class distribution. *Information Sciences*, 512:1192–1201, feb 2020. ISSN 00200255. doi: 10.1016/j.ins.2019.10.017.

[89] Sangwook Kim, Yonghwa Choi, and Minho Lee. Deep learning with support vector data description. *Neurocomputing*, 165:111–117, 10 2015. ISSN 18728286. doi: 10.1016/j.neucom.2014.09.086.

[90] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014.

[91] Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751, 2016.

[92] B. Ravi Kiran, Dilip Mathew Thomas, and Ranjith Parakkal. An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos, 2018. ISSN 2313433X.

[93] Matej Kristan, Aleš Leonardis, and Danijel Skočaj. Multivariate online kernel density estimation with gaussian kernels. *Pattern Recognition*, 44:2630–2642, 10 2011. ISSN 00313203. doi: 10.1016/j.patcog.2011.03.019. URL https://linkinghub.elsevier.com/retrieve/pii/S0031320311001233.

[94] Donghwoon Kwon, Hyunjoo Kim, Jinoh Kim, Sang C Suh, Ikkyun Kim, and Kuinam J Kim. A survey of deep learning-based network anomaly detection. *Cluster Computing*, 22(1):949–961, 2019.

[95] Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 29–37. JMLR Workshop and Conference Proceedings, 2011.

[96] Longin Jan Latecki, Aleksandar Lazarevic, and Dragoljub Pokrajac. Outlier detection with kernel density functions. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 61–75. Springer, 2007.

[97] Quoc Le, Tamas Sarlos, and Alex Smola. Fastfood - approximating kernel expansions in loglinear time. In *30th International Conference on Machine Learning (ICML)*, 2013. URL http://jmlr.org/proceedings/papers/v28/le13.html.

[98] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[99] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[100] Jeisung Lee and Mignon Park. An adaptive background subtraction method based on kernel density estimation. *Sensors (Switzerland)*, 12:12279–12300, 9 2012. ISSN 14248220. doi: 10.3390/s120912279.

[101] Jonathan Li and Andrew Barron. Mixture density estimation. *Advances in neural information processing systems*, 12, 1999.

[102] Kun-Lun Li, Hou-Kuan Huang, Sheng-Feng Tian, and Wei Xu. Improving one-class svm for anomaly detection. In *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 03EX693)*, volume 5, pages 3077–3081. IEEE, 2003.

[103] Yanjun Li, Kai Zhang, Jun Wang, and Sanjiv Kumar. Learning adaptive random features. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4229–4236, 2019.

[104] Zheng Li, Yue Zhao, Nicola Botta, Cezar Ionescu, and Xiyang Hu. Copod: Copula-based outlier detection. pages 1118–1123, 11 2020.

[105] Zhu Li, Jean François Ton, Dino Oglic, and Dino Sejdinovic. Towards a unified analysis of random fourier features. In *36th International Conference on Machine Learning, ICML 2019*, volume 2019-June, pages 6916–6936, 2019. ISBN 9781510886988.

[106] Zhu Li, Jean-Francois Ton, Dino Oglic, and Dino Sejdinovic. Towards a unified analysis of random fourier features. In *International Conference on Machine Learning*, pages 3905–3914. PMLR, 2019.

[107] Zilong Lin, Yong Shi, and Zhi Xue. Idsgan: Generative adversarial networks for attack generation against intrusion detection. 9 2018. URL http://arxiv.org/abs/1809.02077.

[108] Fang Liu, Yanwei Yu, Peng Song, Yangyang Fan, and Xiangrong Tong. Scalable KDE-based top-n local outlier detection over large-scale data streams. *Knowledge-Based Systems*, 204:106186, 2020. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2020.106186. URL https://www.sciencedirect.com/science/article/pii/S0950705120304159.

[109] Fanghui Liu, Xiaolin Huang, Yudong Chen, and Johan A. K. Suykens. Random Features for Kernel Approximation: A Survey on Algorithms, Theory, and Beyond. apr 2020.

[110] Fanghui Liu, Xiaolin Huang, Yudong Chen, Jie Yang, and Johan Suykens. Random fourier features via fast surrogate leverage weighted sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4844–4851, 2020.

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

[112] Qiao Liu, Jiaze Xu, Rui Jiang, and Wing Hung Wong. Density estimation using deep generative neural networks. *Proceedings of the National Academy of Sciences*, 118(15), 2021.

[113] Peng Lv, Yanwei Yu, Yangyang Fan, Xianfeng Tang, and Xiangrong Tong. Layer-constrained variational autoencoding kernel density estimation model for anomaly detection. *Knowledge-Based Systems*, 196, 5 2020. ISSN 09507051. doi: 10.1016/j.knosys.2020.105753.

[114] Yueming Lyu. Spherical structured feature maps for kernel approximation, 2017.

[115] Larry M Manevitz and Malik Yousef. One-class svms for document classification. *Journal of machine Learning research*, 2(Dec):139–154, 2001.

[116] Emaad Manzoor, Hemank Lamba, and Leman Akoglu. Xstream: Outlier dete'x'ion in feature-evolving data streams. pages 1963–1972. Association for Computing Machinery, 7 2018. ISBN 9781450355520. doi: 10.1145/3219819.3220107.

[117] William B. March, Bo Xiao, and George Biros. Askit: Approximate skeletonization kernel-independent treecode in high dimensions. 10 2014. URL http://arxiv.org/abs/1410.0260.

[118] William B March, Bo Xiao, and George Biros. Askit: Approximate skeletonization kernel-independent treecode in high dimensions. *SIAM Journal on Scientific Computing*, 37(2):A1089–A1110, 2015.

[119] Luis Martí, Nayat Sanchez-Pi, José Manuel Molina, and Ana Cristina Bicharra Garcia. Anomaly detection based on sensor data in petroleum industry applications. *Sensors (Switzerland)*, 15:2774–2797, 1 2015. ISSN 14248220. doi: 10.3390/s150202774.

[120] Luis Martí, Nayat Sanchez-Pi, José Manuel Molina, and Ana Cristina Bicharra Garcia. Anomaly detection based on sensor data in petroleum industry applications. *Sensors (Switzerland)*, 15:2774–2797, 1 2015. ISSN 14248220. doi: 10.3390/s150202774.

[121] Barbara J McNeil and James A Hanley. Statistical approaches to the analysis of receiver operating characteristic (roc) curves. *Medical decision making*, 4(2):137–150, 1984.

[122] Seonwoo Min, Byunghan Lee, and Sungroh Yoon. Deep learning in bioinformatics. 18 (5):851–869, 2017. ISSN 14774054. doi: 10.1093/bib/bbw068.

[123] Tom Minka et al. Divergence measures and message passing. Technical report, Citeseer, 2005.

[124] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. Kitsune: An ensemble of autoencoders for online network intrusion detection, 2 2018. ISSN 23318422.

[125] Nour Moustafa and Jill Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set), 2015.

[126] Marina Munkhoeva, Yermek Kapushev, Evgeny Burnaev, and Ivan Oseledets. Quadrature-based features for kernel approximation. 2 2018. URL http://arxiv.org/abs/1802.03832.

[127] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[128] Gyoung S. Na, Donghyun Kim, and Hwanjo Yu. Dilof: Effective and memory efficient local outlier detection in data streams. pages 1993–2002. Association for Computing Machinery, 7 2018. ISBN 9781450355520. doi: 10.1145/3219819.3220022.

[129] Benjamin Nachman and David Shih. Anomaly detection with density estimation. *Physical Review D*, 101(7):075042, 2020.

[130] Elizbar A Nadaraya. Some new estimates for distribution functions. *Theory of Probability & Its Applications*, 9(3):497–500, 1964.

[131] Tomoki Nakaya and Keiji Yano. Visualising crime clusters in a space-time cube: An exploratory data-analysis approach using space-time kernel density estimation and scan statistics. *Transactions in GIS*, 14:223–239, 6 2010. ISSN 13611682. doi: 10.1111/j.1467-9671.2010.01194.x.

[132] HD Nguyen, Kim Phuc Tran, Sébastien Thomassey, and Moez Hamad. Forecasting and anomaly detection approaches using lstm and lstm autoencoder techniques with the applications in supply chain management. *International Journal of Information Management*, 57:102282, 2021.

[133] Hien D. Nguyen, Dianhui Wang, and Geoffrey J. McLachlan. Randomized mixture models for probability density approximation and estimation. *Information Sciences*, 467:135–148, 10 2018. ISSN 00200255. doi: 10.1016/j.ins.2018.07.056.

[134] Guansong Pang, Charu Aggarwal, Chunhua Shen, and Nicu Sebe. Editorial deep learning for anomaly detection. *IEEE Transactions on Neural Networks and Learning Systems*, 33:2282–2286, 6 2022. ISSN 2162-237X. doi: 10.1109/TNNLS.2022.3162123. URL https://ieeexplore.ieee.org/document/9786561/.

[135] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. 5 2017. URL http://arxiv.org/abs/1705.07057.

[136] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

[137] 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. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[138] Kainan Peng, Wei Ping, Zhao Song, and Kexin Zhao. Non-autoregressive neural text-to-speech. 5 2019. URL http://arxiv.org/abs/1905.08459.

[139] Tomáš Pevný. Loda: Lightweight on-line detector of anomalies. *Machine Learning*, 102:275–304, 2016. ISSN 1573-0565.

[140] Tomáš Pevný. Loda: Lightweight on-line detector of anomalies. *Machine Learning*, 102:275–304, 2 2016. ISSN 15730565. doi: 10.1007/s10994-015-5521-0.

[141] Marco A.F. Pimentel, David A. Clifton, Lei Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal Processing*, 99:215–249, 6 2014. ISSN 01651684. doi: 10.1016/j.sigpro.2013.12.026.

[142] Adrian Alan Pol, Victor Berger, Cecile Germain, Gianluca Cerminara, and Maurizio Pierini. Anomaly detection with conditional variational autoencoders. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 1651–1657, 2019. doi: 10.1109/ICMLA.2019.00270.

[143] Rimpal Popat and Jayesh Chaudhary. A survey on credit card fraud detection using machine learning. 2018.

[144] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, page 1177–1184, Red Hook, NY, USA, 2007. Curran Associates Inc. ISBN 9781605603520.

[145] Ali Rahimi, Benjamin Recht, et al. Random features for large-scale kernel machines. In *NIPS*, volume 3, page 5. Citeseer, 2007.

[146] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. page 427438. Association for Computing Machinery, 2000. ISBN 1581132174.

[147] Daniel Ramotsoela, Adnan Abu-Mahfouz, and Gerhard Hancke. A survey of anomaly detection in industrial wireless sensor networks with critical water system infrastructure as a case study. *Sensors (Switzerland)*, 18, 8 2018. ISSN 14248220. doi: 10.3390/s18082491.

[148] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.

[149] Carl Edward. Rasmussen. Gaussian Processes in Machine Learning. *Springer-Verlag Berlin Heidelberg 2004*, 19(1):63–71, 2004. ISSN 00219509.

[150] Shebuti Rayana. ODDS library, 2016. URL http://odds.cs.stonybrook.edu.

[151] S. Reed, Y. Chen, T. Paine, A. van den Oord, S. M.A. Eslami, D. Rezende, O. Vinyals, and N. de Freitas. Few-shot autoregressive density estimation: Towards learning to learn distributions, 2017.

[152] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. volume 2, pages 1530–1538, 2015. ISBN 9781510810587.

[153] Baqar Rizvi, Ammar Belatreche, Ahmed Bouridane, and Ian Watson. Detection of stock price manipulation using kernel based principal component analysis and multivariate density estimation. *IEEE Access*, 8:135989–136003, 2020.

[154] Vijay K Rohatgi and AK Md Ehsanes Saleh. *An introduction to probability and statistics*. John Wiley & Sons, 2015.

[155] Murray Rosenblatt. Remarks on some nonparametric estimates of a density function. *Ann. Math. Statist.*, 27(3):832–837, 09 1956. doi: 10.1214/aoms/1177728190. URL https://doi.org/10.1214/aoms/1177728190.

[156] Peter J Rousseeuw and Katrien Van Driessen. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41(3):212–223, 1999.

[157] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. volume 80. PMLR, 2018.

[158] Lukas Ruff, Jacob R. Kauffmann, Robert A. Vandermeulen, Gregoire Montavon, Wojciech Samek, Marius Kloft, Thomas G. Dietterich, and Klaus Robert Muller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 109: 756–795, 5 2021. ISSN 15582256. doi: 10.1109/JPROC.2021.3052449.

[159] G Rupert Jr et al. Simultaneous statistical inference. *Springer Series in Statistics*, 2012.

[160] Saket Sathe and Charu C Aggarwal. Subspace outlier detection in linear time with randomized hashing. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 459–468. IEEE, 2016.

[161] Issei Sato, Kenichi Kurihara, Shu Tanaka, Hiroshi Nakagawa, and Seiji Miyashita. Quantum annealing for variational Bayes inference. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence, UAI 2009*, pages 479–486, 2009.

[162] B. Schölkopf. Learning with kernels. *Journal of the Electrochemical Society*, 129 (November):2865, 2002. ISSN 0162-1459. doi: 10.1198/jasa.2003.s269. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.167.5140&rep=rep1&type=pdf.

[163] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *International conference on artificial neural networks*, pages 583–588. Springer, 1997.

[164] Bernhard Scholkopf, Robert C Williamson, and Peter L Bartlett. New Support Vector Algorithms ¤. *Neural Computation*, 1245:1207–1245, 2000.

[165] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 2001.

[166] David W Scott. Multivariate density estimation and visualization. In *Handbook of computational statistics*, pages 549–569. Springer, 2012.

[167] Younghyun Jo Sejong, Yang Seon, and Joo Kim. SRFlow-DA: Super-Resolution Using Normalizing Flow with Deep Convolutional Block. Technical report, 2021. URL https://github.com/yhjo09/SRFlow-DA.

[168] Razieh Sheikhpour, Mehdi Agha Sarram, and Robab Sheikhpour. Particle swarm optimization for bandwidth determination and feature selection of kernel density estimation based classifiers in diagnosis of breast cancer. *Applied Soft Computing Journal*, 40:113–131, 3 2016. ISSN 15684946. doi: 10.1016/j.asoc.2015.10.005.

[169] Boxi Shen, Xiang Xu, Jun Li, Antonio Plaza, and Qunying Huang. Unfolding spatial-temporal patterns of taxi trip based on an improved network kernel density estimation. *ISPRS International Journal of Geo-Information*, 9:683, 11 2020. ISSN 2220-9964. doi: 10.3390/ijgi9110683.

[170] Weiwei Shen, Zhihui Yang, and Jun Wang. Random features for shift-invariant kernels with moment matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[171] Xun Shi. Selection of bandwidth type and adjustment side in kernel density estimation over inhomogeneous backgrounds. *International Journal of Geographical Information Science*, 24:643–660, 5 2010. ISSN 13658816. doi: 10.1080/13658810902950625.

[172] Alistair Shilton, Sutharshan Rajasegarar, and Marimuthu Palaniswami. Combined multiclass classification and anomaly detection for large-scale wireless sensor networks. In *2013 IEEE eighth international conference on intelligent sensors, sensor networks and information processing*, pages 491–496. IEEE, 2013.

[173] Paris Siminelakis, Kexin Rong, Peter Bailis, Moses Charikar, and Philip Levis. Rehashing kernel evaluation in high dimensions. volume 2019-June, pages 10153–10173, 2019. ISBN 9781510886988.

[174] Aman Sinha and John Duchi. Learning kernels with random features. In *Advances in Neural Information Processing Systems*, pages 1306–1314, 2016.

[175] Daniel B. Smith. Kde-for-scipy (python script). https://github.com/Daniel-B-Smith/KDE-for-SciPy/blob/master/kde.py, 2021. Accessed on March 6, 2023.

[176] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, Colorado Univ at Boulder Dept of Computer Science, 1986.

[177] Angela A Sodemann, Matthew P Ross, and Brett J Borghetti. A review of anomaly detection in automated surveillance. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1257–1272, 2012.

[178] R. F. Streater. Classical and quantum probability. *Journal of Mathematical Physics*, 41:3556–3603, 2000. ISSN 00222488. doi: 10.1063/1.533322.

[179] Swee Chuan Tan, Kai Ming Ting, and Tony Fei Liu. Fast anomaly detection for streaming data. In *Twenty-second international joint conference on artificial intelligence*, 2011.

[180] Xiaofeng Tang and Aiqiang Xu. Multi-class classification using kernel density estimation on K-nearest neighbours. *Electronics Letters*, 52(8):600–602, apr 2016. ISSN 00135194. doi: 10.1049/el.2015.4437.

[181] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A detailed analysis of the kdd cup 99 data set, 2009. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[182] P. Tiwari and M. Melucci. Towards a quantum-inspired binary classifier. *IEEE Access*, 7:42354–42372, 2019. doi: 10.1109/ACCESS.2019.2904624.

[183] Maximilian E. Tschuchnig and Michael Gadermayr. Anomaly detection in medical imaging - a mini review. In Peter Haber, Thomas J. Lampoltshammer, Helmut Leopold, and Manfred Mayr, editors, *Data Science – Analytics and Applications*, pages 33–38, Wiesbaden, 2022. Springer Fachmedien Wiesbaden. ISBN 978-3-658-36295-9.

[184] Berwin A Turlach. Bandwidth selection in kernel density estimation: A review. In *CORE and Institut de Statistique*. Citeseer, 1993.

[185] Andrea Vedaldi and Andrew Zisserman. Efficient additive kernels via explicit feature maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):480–492, 2012. ISSN 01628828. doi: 10.1109/TPAMI.2011.153.

[186] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders, 2009.

[187] John Von Neumann. Wahrscheinlichkeitstheoretischer aufbau der quantenmechanik. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, 1927:245–272, 1927.

[188] Lin Wang, Fuqiang Zhou, Zuoxin Li, Wangxia Zuo, and Haishu Tan. Abnormal Event Detection in Videos Using Hybrid Spatio-Temporal Autoencoder. In *Proceedings - International Conference on Image Processing, ICIP*, pages 2276–2280, 2018. ISBN 9781479970612. doi: 10.1109/ICIP.2018.8451070.

[189] Xuanzhao Wang, Zhengping Che, Bo Jiang, Ning Xiao, Ke Yang, Jian Tang, Jieping Ye, Jingyu Wang, and Qi Qi. Robust unsupervised video anomaly detection by multipath frame prediction. *IEEE Transactions on Neural Networks and Learning Systems*, 33(6):2301–2312, 2022. doi: 10.1109/TNNLS.2021.3083152.

[190] Yong Wang, Xinbin Luo, Lu Ding, Shan Fu, and Xian Wei. Detection based visual tracking with convolutional neural network. *Knowledge-Based Systems*, 175:62–71, 2019.

[191] Zhipeng Wang and David W. Scott. Nonparametric density estimation for high-dimensional data—algorithms and applications, 7 2019. ISSN 19390068.

[192] Christopher K. I. Williams. Using the nystrom method to speed up kernel machines. 2001.

[193] Lior Wolf. Learning using the born rule. Technical report, Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory, 2006.

[194] Miao Xie, Song Han, Biming Tian, and Sazia Parvin. Anomaly detection in wireless sensor networks: A survey. *Journal of Network and computer Applications*, 34(4): 1302–1325, 2011.

[195] Tianbao Yang, Yu Feng Li, Mehrdad Mahdavi, Rong Jin, and Zhi Hua Zhou. Nyström method vs random Fourier features: A theoretical and empirical comparison. In *Advances in Neural Information Processing Systems*, volume 1, pages 476–484, 2012. ISBN 9781627480031.

[196] Felix Xinnan Yu, Ananda Theertha Suresh, Krzysztof Choromanski, Daniel Holtmann-Rice, and Sanjiv Kumar. Orthogonal random features. pages 1983–1991, 10 2016. URL http://arxiv.org/abs/1610.09072.

[197] Felix Xinnan X Yu, Ananda Theertha Suresh, Krzysztof M Choromanski, Daniel N Holtmann-Rice, and Sanjiv Kumar. Orthogonal random features. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29, pages 1975–1983. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/53adaf494dc89ef7196d73636eb2451b-Paper.pdf.

[198] Houssam Zenati, Manon Romain, Chuan-Sheng Foo, Bruno Lecouat, and Vijay Chandrasekhar. Adversarially learned anomaly detection. In *2018 IEEE International conference on data mining (ICDM)*, pages 727–736. IEEE, 2018.

[199] Liangwei Zhang, Jing Lin, and Ramin Karim. Adaptive kernel density-based anomaly detection for nonlinear systems. *Knowledge-Based Systems*, 139:50–63, 2018. ISSN 09507051. doi: 10.1016/j.knosys.2017.10.009.

[200] Yue Zhao, Zain Nasrullah, and Zheng Li. Pyod: A python toolbox for scalable outlier detection. *Journal of Machine Learning Research*, 20, 2019. ISSN 15337928.

[201] Xun Zhou, Sicong Cheng, Meng Zhu, Chengkun Guo, Sida Zhou, Peng Xu, Zhenghua Xue, and Weishi Zhang. A state of the art survey of data mining-based fraud detection and credit scoring. volume 189. EDP Sciences, 8 2018. doi: 10.1051/matecconf/201818903002.