InQMAD: Incremental Quantum Measurement Anomaly Detection

Joseph A. Gallego-Mejia
MindLab Research Group,
Universidad Nacional de Colombia
Bogotá, Colombia
jagallegom@unal.edu.co

Oscar A. Bustos-Brinez
MindLab Research Group,
Universidad Nacional de Colombia
Bogotá, Colombia
oabustosb@unal.edu.co

Fabio A. González
MindLab Research Group,
Universidad Nacional de Colombia
Bogotá, Colombia
fagonzalezo@unal.edu.co

Abstract—Streaming anomaly detection refers to the problem of detecting anomalous data samples in streams of data. This problem poses challenges that classical and deep anomaly detection methods are not designed to cope with, such as conceptual drift and continuous learning. State-of-the-art flow anomaly detection methods rely on fixed memory using hash functions or nearest neighbors that may not be able to constrain high frequency values or remove seamless outliers and cannot be trained in an end-to-end deep learning architecture. We present a new incremental anomaly detection method that performs continuous density estimation based on random Fourier features and the mechanism of quantum measurements and density matrices. The method continuously updates the estimation of the density of normal samples, giving more importance to new samples and decreasing the importance of older samples. It can process potentially endless data and its update complexity is constant $O(1)$. A systematic evaluation against 12 state-of-the-art streaming anomaly detection algorithms and using 12 streaming datasets is presented.

Index Terms—incremental learning, anomaly detection, density matrix, random features, kernel density estimation, quantum machine learning

I. INTRODUCTION

Anomaly detection is a well-studied problem [9], [30], [35]. 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 [26], telecommunications [12], retail industry [29], network traffic [40], financial transactions [2], and wired and wireless sensors [41]. 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 [32], network system intrusion detection [26], camera surveillance [37], Internet of Things (IOT) device problems [25], among others. Data in these streaming environments is rapidly generated, potentially infinite, has tremendous volume, and can exhibit concept drift.

Classical anomaly detection algorithms in batch environments are based on density estimation, such as kernel density estimation [20], classification, such as one-class support vector machines [23], and distance, such as the isolation forest [22]. Another type of recent solutions are based on deep neural networks that have shown good properties in the anomaly detection task. They are mainly based on variational autoencoders [3], deep belief networks [19], one-class deep networks [8] and adversarial autoencoders [10]. 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.

Several methods for stream anomaly detection have been developed in the last decade. Methods such as iForestASD [11], RCF [16], xStream [24] and Ex. IF [17] 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 [38], RS-Hash [36] and MStream [5] 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) [7]. Methods such as LODA [31], DILOF [28] and MemStream [5] 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 changes on the data stream.

In this paper 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 whose dot product approximates a Gaussian kernel. In this space, a density matrix is used to capture the density of the data stream. The model estimates the density of new points 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 with data streams, is an unsupervised algorithm and retraining can be performed continuously. The proposed method requires
A novel stream anomaly detection method: the new method works in a potentially infinite data stream that could exhibit concept drift.

A systematic evaluation of the proposed method: the algorithm is evaluated in 12 streaming datasets and compared 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 paper is released as open source and can be found in https://github.com/Joaggi/Incremental-Anomaly-Detection-using-Quantum-Measurements and in Zenodo https://zenodo.org/record/7183564

The outline of the paper is as follows: in Section 2, we describe anomaly detection in streaming and present the benchmark methods with which we will compare our algorithm. In Section 3, we present the new method, explaining all the stages of the algorithm. In Section 4, we systematically evaluate the proposed method against state-of-the-art streaming anomaly detection algorithms. In Section 5, we state conclusions and outline future lines of research.

II. BACKGROUND AND RELATED WORK

A. Streaming Anomaly Detection

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 [19]. 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 [6].

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 [11]. 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 [31]. 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 following subsection II-C, 12 methods for streaming anomaly detection are presented.

B. Streaming Anomaly Detection Problem

In streaming anomaly detection, the data points $X = \{x_1, \cdots, x_t\}$ 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 in 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.

C. Streaming Anomaly Detection Baseline Methods

- STORM [4]: 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 [38]: 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 [11]: This algorithm has its roots in the Isolation Forest method. The method uses a window of the streaming data and is sent to the Isolation Forest. An abnormality score is calculated using the average of the depth of the point on each tree in the forest. If the point is normal, it is joined to the Isolation Forest.
- RS-Hash [36]: 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 [16]: The algorithm constructs a robust randomized cutting tree (rcrt) using a random selection of the dimension weighted by its range. A uniform distribution is used to cut the selected dimension. This process is repeated $n$ times, with $n$ being the maximum depth. A forest is constructed using several rcrt. A new point is classified as an anomaly using the comparison of its insertion and deletion complexity.
MemStream [6]: The method proposes a nearest neighbor
Ex. IF [17]: This algorithm uses a random slope to cut the
MStream [5]: This algorithm uses two locality-sensitive
xStream [24]: The method uses a hash structure to reduce
DILOF [28]: The method uses the
Kitsune [26]: Kitsune is an algorithm that uses an
LODA [31]: The algorithm uses a set of histograms for
The anomaly score is calculated using a chi-square density
The latter computes a hash for all features simultaneously.
DILOF [28]: 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 [24]: 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 [5]: 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 [17]: 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 [6]: 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.

III. INCREMENTAL QUANTUM MEASUREMENT ANOMALY DETECTION (INQMAD)

In this section, we present the new method new Incremental Quantum Measurement Anomaly Detection (InQMAD). Figure 1 shows each of the steps of the Algorithms 1 and 2. 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 \( \rho \) in case of new normal data points. Each stage is explained in detail in this section.

A. Adaptive Fourier Features

All streaming data points are mapped into a Hilbert space using adaptive Fourier features, first proposed as random Fourier features by [33]. They showed that a Gaussian kernel can be approximated as:

\[
k(x, y) \simeq \mathbb{E}_{w}(\langle \hat{\phi}_{w}(x), \hat{\phi}_{w}(y) \rangle)
\]

where \( \hat{\phi}_{w} = \sqrt{2} \cos(w^{T}x + b), w \sim N(0, I^{d}) \) and \( b \sim \text{Uniform}(0, 2\pi) \). The expectation is possible to Bochner’s Theorem and the fact that using Equation 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 [13], [21]. Given two pairs of random data points \((x_{i}, x_{j})\), the parameters \( w, b \) can be optimized using the following equation:

\[
w^{*}, b^{*} = \arg \min_{w, b} \frac{1}{m} \sum_{x_{i}, x_{j} \in S} (k_{\sigma}(x_{i}, x_{j}) - \hat{k}_{w,b}(x_{i}, x_{j}))^{2}
\]

where \( \hat{k}_{w,b}(x_{i}, x_{j})^{2} \) is the Gaussian kernel of Fourier feature approximation and \( k_{\sigma}(x_{i}, x_{j}) \) is the Gaussian kernel with a bandwidth parameter \( \sigma \). 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 1 shows the step in 1.
where A quantum measurement can be obtained by using the density matrix \( \rho \). The density matrix is initialized in Algorithm 1 in Step 1.

C. Quantum Measurement

The density matrix is a formalism of quantum mechanics that was used as a base tool by [13], [14] to create a density estimation method. The authors derive a new algorithm that uses random Fourier features to store the density matrix \( \rho \). The following equation is a slight modification of the density matrix in terms of stream data:

\[
\rho_t = \frac{1}{N} \sum_{i=1}^{N} q_i \cdot \phi_{\text{aff}}(x_i) \phi_{\text{aff}}^T(x_i)
\]  

(2)

where \( \sum_{i=1}^{t} q_i = 1 \). To initially compute the density matrix, an initial portion of the stream \( \{x_1, \cdots, x_n\} \) is selected and sent to the equation 2 to calculate the density matrix \( \rho \). 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 [15]. The density matrix is initialized in Algorithm 1 in Step 1.

D. Anomaly Detection Classification

The fourth step of the algorithm is the anomaly detection classification stage. The threshold is defined as \( \tau \), where a new point \( x_t \) is defined as anomalous according to the following equation:

\[
\hat{y}(x_t) = \begin{cases} 
\text{‘normal’} & \text{if } \hat{f}(x_t) \leq \tau \\
\text{‘anomaly’} & \text{otherwise}
\end{cases}
\]

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 (\( \beta \)). The second approach is to use an optimization metric computed over \( \{\hat{f}(x_1), \cdots, \hat{f}(x_n)\} \), for instance using the best threshold with respect to AUC-ROC in initial memory. The \( \tau \) is obtained in Step 1 of the Algorithm 1 and the anomaly detection classification occurs in Step 2 of the Algorithm 2.

E. Density Matrix Update

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

\[
\rho_{t+1} = (1 - \alpha) \cdot \rho_t + \alpha \cdot \phi_{\text{aff}}(x_{t+1}) \phi_{\text{aff}}^T(x_{t+1})
\]

(4)

Proposition 1. The resulting matrix \( \rho_{t+1} \) in Equation 4 is a valid density matrix of the form \( \rho_{t+1} = \sum_{i=1}^{t+1} q_i \phi(x_i) \phi^T(x_i) \) with \( q_i = (1 - \alpha)^{t+1-i}, \quad q_i = (1 - \alpha)^{t+1-i}, \alpha \forall i \in \{1, \cdots, t\} \) and \( \alpha \in [0,1] \).

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 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 \( \alpha = \frac{\text{memory window width}}{2} \). 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 2 of Algorithm 2.

### F. Complexity Analysis

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 at 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 \( O(dD) \).

### IV. EXPERIMENTAL EVALUATION

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

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

- **Q2. Adaptability.** How well our method does 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?

#### A. Comparison to Streaming Methods

**Experimental Setup:** The experimental setup presented in this paper took inspiration from the setup proposed in [6]. There, it was performed a comparison of a dozen of algorithms, based on area under the ROC curve (commonly 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.

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 I, and a brief description of each dataset is presented in the Supplemental Material.

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 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, that is used in the initialization stage of the method. The $n$ samples allow the construction of the first density matrix $\rho_0$, 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 $w_{\text{aff}}$ and $b_{\text{aff}}$ 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 $\{10^{-2}, 10^{-3}, 10^{-4}\}$.
- $\sigma$: (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 $\sigma^2$, 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.
- $\alpha$: 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 [6], 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 each pair of dataset and algorithm is presented in Table II. 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.

B. Ablation Study

Being one of the most important stages of our algorithm, we wanted to determine the degree of influence of the Adaptive

| 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        |

The selected parameters are the following:

- $n$: this corresponds to the size of the initial training dataset $D_0$ that is used in the initialization stage of the method. The $n$ samples allow the construction of the first density matrix $\rho_0$, 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 $w_{\text{aff}}$ and $b_{\text{aff}}$ 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 $\{10^{-2}, 10^{-3}, 10^{-4}\}$.
- $\sigma$: (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 $\sigma^2$, 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.
- $\alpha$: 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 $[0.001, 1]$.
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 [14], 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 III. 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.

![Table III](https://example.com/table.png)

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.

### V. Conclusion

In this paper 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 of the method was performed against 12 state-of-the-art methods on 12 streaming datasets, showing improved performance. 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 $\alpha$. For future research, we will use a dimensionality reduction step such as principal component analysis or autoencoders to help with the curse of dimensionality in high-dimensional datasets.

### References

[1] Kdd cup dataset (1999), http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[2] Ahmed, M., Mahmood, A.N., Islam, M.R.: A survey of anomaly detection techniques in financial domain. Future Generation Computer Systems 55, 278–288 (2016)
[3] An, J., Cho, S.: Variational autoencoder based anomaly detection using reconstruction probability. Special Lecture on IE 2(1), 1–18 (2015)
[4] Angiulli, F., Pazzetti, F.: Detecting distance-based outliers in streams of data. pp. 811–820 (2007). https://doi.org/10.1145/1321440.1321552
[5] Bhatia, S., Jain, A., Li, P., Kumar, R., Hooi, B.: Mstream: Memory-based streaming anomaly detection. pp. 3371–3382. Association for Computing Machinery, Inc (4 2021). https://doi.org/10.1145/3442381.3450023
[6] Bhatia, S., Jain, A., Srivastava, S., Kawaguchi, K., Hooi, B.: Memstream: Memory-based streaming anomaly detection. WWW 2022 - Proceedings of the ACM Web Conference 2022 pp. 610–621 (2022). https://doi.org/10.1145/3485447.3512221
[7] Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: LoF: identifying density-based local outliers. In: Proceedings of the 2000 ACM SIGMOD international conference on Management of data. pp. 93–104 (2000)
[8] Chalapathy, R., Menon, A.K., Chawla, S.: Anomaly detection using one-class neural networks. arXiv preprint arXiv:1802.06360 (2018)
[9] Chandola, V.: Anomaly detection: A survey (2009)
[10] Demokrantiou, A.: Adversarial autoencoders for anomalous event detection in images. Ph.D. thesis, Purdue University (2017)
[11] Ding, Z., Fei, M.: An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window, vol. 3, pp. 12–17. IFAC Secretariat (2013). https://doi.org/10.3182/20130902-3-CN-3020.00044
VI. SUPPLEMENTAL MATERIAL

Proposition 2. The resulting matrix $\rho_{t+1} = (1-\alpha) \cdot \rho_t + \alpha \cdot \phi(\mathbf{x}_{t+1}) \phi^T(\mathbf{x}_{t+1})$ in Equation 4 is a valid density matrix of the form

$$\rho_{t+1} = \sum_{i=1}^{t+1} q_i \phi(\mathbf{x}_i) \phi^T(\mathbf{x}_i)$$

with $q_1 = (1-\alpha)^{t+1}$, $q_i = (1-\alpha)^{t-i} \cdot \alpha$, for $i \in \{1, \ldots, t\}$ and $\alpha \in [0, 1]$. 

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

Base Case: for $t = 1$, define $p_1 = \phi(\mathbf{x}_1) \phi^T(\mathbf{x}_1)$, 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 $p_t$ holds, then $\rho_{t+1}$ also holds.

Define

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

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

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

Now, for $t+1$:

$$q_{t+1} = q_1 \cdot (1-\alpha)^t + \alpha \cdot \sum_{i=1}^{t-1} (1-\alpha)^{t-i+1} + q_2$$

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

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

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

$$= 1$$

A. Dataset Description

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

- Cardio [34]: 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

[37] Sodemann, A.A., Ross, M.P., Borghetti, B.J.: 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)

[38] Tan, S.C., Ting, K.M., Liu, T.F.: Fast anomaly detection for streaming data. In: Twenty-second international joint conference on artificial intelligence (2011)

[39] Tavallaee, M., Bagheri, E., Lu, W., Ghorgani, A.A.: A detailed analysis of the kdd cup 99 data set (2009), http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[40] Wang, Y., Luo, X., Ding, L., Fu, S., Wei, X.: Detection based visual tracking with convolutional neural network. Knowledge-Based Systems 175, 62–71 (2019)

[41] Xie, M., Han, S., Tian, B., Parvin, S.: Anomaly detection in wireless sensor networks: A survey. Journal of Network and computer Applications 34(4), 1302–1325 (2011)
formed the inliers and the pathologic class is downsampled and labeled as outliers, while the suspect class is discarded.

- Ionosphere [34]: originally a binary classification dataset from UCI ML repository, the ‘bad’ class is considered as outliers and the ‘good’ class as inliers.
- Mammography [34]: 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 [34]: also from UCI ML repository, it includes data of female Indian patients with the objective of classifying them as diabetic or healthy.
- Satellite [34]: derived from the Statlog dataset from UCI ML repository, it is a multiclass 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 [34]: coming from the previous Satellite dataset, here class 2 has been downsampled 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 [6] 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 [39]: coming from the previous KDD dataset, it adds some dimensions and solves redundant and duplicate records.
- Cover [34]: 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 [18]: 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 [27]: 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.

### B. Parameters

The winning combinations of parameters for our method are presented in Table IV. 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 V.

#### TABLE IV

| Dataset | Winning Parameters |
|---------|-------------------|
| 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 | [3000 0.001 2.0 0.5] |
| Mammography | [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 V

| Method | Winning Parameters |
|--------|-------------------|
| STORM | window_size = 10000, max_radius = 0.1 |
| HS-Tree | window_size = 100, num_trees = 25, max_depth = 15, init_window = 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 |
| LOIDA | num_bins = 10, num_random_cats = 100 |
| Kitsune | max_size = 10, learning_rate = 0.1, hidden_ratio = 0.75, grace_feature_mapping = 0.1 |
| DILOF | window_size = 500, K = 8, thresholds = [0.1, 0.1, 1.1, 1.15, 1.2, 1.2, 1.3, 1.4, 1.6, 2.0, 3.0] |
| xStream | projection_size = 50, number_chains = 50, depth = 10, rowstream = 0, rwindows = 0, scoring_batch_size = 10000 |
| 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] |

### C. Statistical Tests

Using the Table II 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. Friedman test uses the following formula:

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

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 $P[\chi^2_{n-1} \geq Q]$, 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 $3.06 \times 10^{-14}$, clearly showing that such difference does exist.

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 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.