Quantum Clustering on Streaming Data: A Novel Method for Analyzing Big Data

Rebecca Hofer[1] and Kevin Mallinger[1,2]
[1]*SBA Research gGmbh, Vienna, Austria*
[2]*Technical University Vienna, Vienna, Austria*

Keywords: Quantum Clustering, Data Stream Clustering, Intrusion Detection, CIDDS-001 Dataset, IoTDI20 Dataset.

Abstract: Quantum Clustering is an efficient unsupervised machine learning method that exploits models of quantum mechanics to discover clusters in data points. We applied an adaption of the algorithm on the CIDDS-001 and IoTID20 network intrusion datasets to distinguish malicious from benign network activity. For this purpose, we integrated Quantum Clustering into the framework of DenStream, adjusting it to the streaming data conditions required for analyzing network data. We found that this significantly improved running time and memory requirements compared to the original version of Quantum Clustering, which is known to have high computational complexity. We also found that the accuracy with which the proposed version detected patterns in network activity was comparable to established methods, confirming the algorithm's applicability for intrusion detection.

1 INTRODUCTION

1.1 Background

Novel advances in data collection methods increased data availability tremendously over the past years. Data that is generated and processed at fast rates is also called data stream (C. Aggrawal, 2003). This vast accessibility opens up many areas of application, such as intrusion detection, stock market analysis, social network analysis, or observation of environmental systems. As analyzing big data is a challenge due to limited computational resources, storage, and processing time, there is a growing interest in increasing the efficiency of methods while still producing confident results for processing and managing massive data (J. A. Silva, 2013).

In common data mining techniques, a finite amount of data with a stationary probability distribution is assumed, and the data analysis method is performed over the entire dataset. However, a core issue with data streams is that new objects arrive continuously, and the stream size is potentially unbounded. Formally, a data stream can be described as a sequence of objects x_1, x_2,...,x_k arriving at times T_1, T_2,..., T_k where each of these objects represents a $m \times 1$ - dimensional feature vector $x_i = (x_i^1, x_i^2, ... x_i^m)$. Because of this continuous arrival of new data and non-stationary data generation processes, the underlying probability distribution is not stable but dynamically evolves (J. A. Silva, 2013). In network traffic analysis, for instance, patterns of user connections are not stable but gradually change over time. In environmental monitoring, data can be subject to an external influence such as forest fire, which will subsequently change its data distribution (C. Feng, 2006). Considering the properties mentioned above, methods that provide fast analysis are needed and can have a valuable contribution to the research areas and applications where real-time monitoring and fast response due to time-critical processes are needed.

The Internet-of-Things (IoT) is an ever-growing source of data streams, attracting attention from the industry as well as the academic fields (D.E. Kouicern, 2018). As such, it is vulnerable to security attacks in hardware and network components. The heterogeneity, resource constraints, physical coupling, and complex environment of the devices make it especially hard for traditional network or internet security approaches to ensure safety (M.M. Noor, 2019). Security solutions for networks can be grouped into three main components: prevention, detection, and mitigation. When prevention methods fail, intrusion detection comes into play (I. Butun, 2014). Our research focuses on implementing a new clustering approach to detect malicious be-

Hofer, R. and Mallinger, K.
Quantum Clustering on Streaming Data: A Novel Method for Analyzing Big Data.
DOI: 10.5220/0011764200003482
In *Proceedings of the 8th International Conference on Internet of Things, Big Data and Security (IoTBDS 2023)*, pages 17-28
ISBN: 978-989-758-643-9; ISSN: 2184-4976

havior through scanning network data produced in a simulated business environment and a network of Internet-of-Things devices. Since the types of attacks can vary, our approach solely relies on detecting attacks, assuming that those patterns are distinguishable from malign network behavior but without prior knowledge about the specific attack type.

1.2 State-of-the-Art

Clustering is a widely used technique for processing data streams (A. Zubaroglu, 2021), (C. Aggrawal, 2003), (C. Feng, 2006), (T. Zhang, 1997). It is an unsupervised machine-learning technique that groups data according to a well-defined similarity. Clustering algorithms adapted for streaming data differ from static methods because they have to consider this dynamic evolution of the data probability distribution. Algorithms handling data streams need to discard historical data after a certain period to capture actual patterns when clustering is performed and to ensure memory accessibility (J. A. Silva, 2013). To address this issue, a common approach of data stream clustering methods is separating the clustering process into an online and offline process. In the online component, statistical summary information of the data is stored. At the same time, the offline component takes this summary information as input for the actual clustering (C. Feng, 2006). The most relevant and cited data stream clustering algorithms are BIRCH (T. Zhang, 1997), CluStream (C. Aggrawal, 2003), StreamKM++ (M.R. Ackermann, 2012), which are based on the K-means algorithm, DenStream (F. Cao, 2006), D-Stream (Y. Chen, 2006), that are based on DBSCAN, ClusTree (P. Kranen, 2011) as a hybrid method of k-means and DBSCAN, or Dynamical Gaussian Mixture model, based on Gaussian Mixture model clustering, to name a few (J. Diaz-Rozo, 2018). A commonly known problem with k-means adaptions is the infeasibility of detecting complex and arbitrarily shaped clusters (J. Diaz-Rozo, 2018). As stream data only can be read once and the number and shape of clusters are unknown in advance, methods need to be able to adapt to these data complexities without prior specifications (L. Wan, 2009). Although density-based clustering methods, such as the State-of-the-Art algorithm DBSCAN, often perform very well (A. Zubaroglu, 2021), there is a need to adapt and fine-tune methods to specific use cases. For network intrusion analysis, algorithms need to be able to detect and discriminate attack patterns of varying intensities. It was shown that k-means, as well as DBSCAN, lack sensitivity to detect those patterns in the context of streaming data (P. Casas, 2012). This is why we propose a new method that is tailored to the specific challenges of network intrusion detection. Nevertheless, taking DenStream as an extension of DBSCAN as a comparison algorithm in this study, we compare our method to a well-performing and widely established stream clustering algorithm, highlighting the benefits our new method brings to the field.

We analyze our algorithm for processing data streams in the context of network intrusion detection. Currently, no research was conducted to investigate the application of Quantum Clustering for streaming data or network anomaly detection. As Quantum technology and, thus, new possibilities in computation and analysis are on the rise, quantum-inspired machine learning is bridging the gap where quantum hardware computations are not commercially available. However, the conceptual or mathematical framework can already be used and mapped to traditional computation tasks. Case studies using quantum-inspired machine learning techniques are found in the context of analyzing high-energy physics data (T. Felser, 2020), the analysis of 6G networks (T.Q. Duong, 2022), or recommendation systems (Tang, 2019) to name a few. Some of the studies simulate the behavior of qubits or quantum states, in general, (Y. Mahmoudi, 2020) as processing units and use these approaches in the development of new methods, while others exploit the mathematical framework to solve classical computation tasks in a new way (K.H. Han, 2002). Quantum Clustering is a technique inspired by the description and evolution of states described as wavefunctions under the Schrödinger equation. It works with a Parzen-window estimator as a Gaussian wave packet over data points, viewing those as the Eigenfunctions of the Schrödinger equation (Y. Mahmoudi, 2020). Studies have shown the strength of the method in several applications such as asteroid spectrum taxonomy (Deutsch, 2017), document analysis (D. Liu, 2016), or financial data analysis (Shaked, 2013). Furthermore, (D. Liu, 2016) showed the feasibility of the method for detecting outliers in datasets, which makes it an exciting candidate for the application of network data analysis. They showed that the algorithm could capture subtle changes in the density of the data and outperformed standard methods such as DBSCAN and conventional Parzen-window methods (D. Liu, 2016). A significant advantage of Quantum Clustering is that it does not require a predefined number of clusters as conventional methods like k-means, which makes it particularly practicable for automated analysis of data streams. The data partition and sensitivity can be varied with the length scale of sigma - the width parameter of the Gaussian kernel - where a bigger value

of sigma increases the width of the Gaussian function. As a framework for finding the optimal hyperparameter value, (R.V. Casana-Eslava, 2020) suggested an approach that considers selecting the kernel width as the mean distance between the data points and its k-nearest neighbors. A detailed explanation of the method can be found in (R.V. Casana-Eslava, 2020). As Quantum Clustering comes with the challenge of having a high time and space complexity O(n^2) due to the computation of distances between each data point with all the other data points and the gradient of the potential function (P. Jiménez, 2022), (Shaked, 2013) (see Eq. 3), studies deployed the original Quantum Clustering mostly to small datasets (D. Liu, 2016), (D. Horn, 2001), (R.V. Casana-Eslava, 2020). To tackle the complexity problem, (Shaked, 2013) suggested using Approximate Quantum Clustering to reduce the number of data points for calculating the probability function. In this approach, the dataspace gets divided into an n-dimensional grid, where one unit is called a voxel. The data points falling within a voxel are approximated as one data point. This reduces the problem's dimensionality, but the method is not directly applicable to analyzing data streams, as it does not consider the dynamic changes in the underlying probability distribution of data.

To overcome the mentioned limitations, this paper investigates the implementation of Quantum Clustering into the framework for handling large data streams while combining the benefits of fast and accurate cluster detection and the detection of small groups of irregularities, without prior knowledge of cluster amounts. With this, we present a new method in data stream analysis and propose an approach to address the complexity problem of Quantum Clustering. The suggested approach builds on the work of (C. Feng, 2006) and the DenStream algorithm, as an extension of DBSCAN for handling large data streams. The model is based on a density clustering approach seeking to partition the data space into high and low-density regions.

In the following, some preliminary concepts will be described. We will then introduce the adapted version of Quantum Clustering for streaming data and evaluate it in terms of clustering quality, space, and time complexity.

2 METHODS

2.1 Quantum Clustering

The method of Quantum Clustering was developed by David Horn, and Assaf Gottlieb (D. Horn, 2001). It uses a Gaussian kernel to generate a probability distribution from the data points given by:

$$\Psi(x) = \sum_i e^{\frac{(x-x_i)^2}{2\sigma^2}} \tag{1}$$

The probability distribution results as the sum of Gaussians computed over the data points. Embedded in the Quantum framework, we interpret the probability distribution as a wave function and insert it into the stationary Schrödinger equation:

$$H\psi \equiv \big(-\frac{\sigma^2}{2}\nabla^2 + V(x)\big)\Psi(x) = E\Psi(x) \tag{2}$$

where H is the Hamiltonian, E is the energy eigenvalue, σ is the width parameter of the Gauss function, ∇ denotes the Laplacian operator and $V(x)$ is the potential function. We denote the ground state energy eigenvalue with $E = \frac{d}{2}$ as the lowest possible energy in the system. When solving the Schrödinger equation, we usually look for solutions for the wavefunctions as evolving in the system described by the Hamiltonian, from which the probability of finding a particle in a certain state can be calculated. The Hamiltonian, specifying this evolution, is composed of a kinetic term and the potential V. In the case of Quantum Clustering, we look at the problem from the opposite side - given the probability distribution of data points, we want to calculate the potential function V, which results from the Gaussians over the data distribution. We then interpret the local minima of this function as our cluster centers. Rewriting Eq. (2), we arrive at the expression for the potential function of the form:

$$V(x) = E + \frac{\frac{\sigma^2}{2}\nabla^2\Psi}{\Psi} = E - \frac{d}{2} + \frac{1}{2\sigma^2\Psi}\sum_i (x-x_i)^2 e^{-\frac{(x-x_i)^2}{2\sigma^2}} \tag{3}$$

Setting $E = -min\frac{\frac{\sigma^2}{2}\nabla^2\Psi}{\Psi}$, the potential function V can be determined uniquely. To arrive at the clusters, we apply the Gradient descent algorithm to our potential function, with η as the learning rate. Defining $y_i(0) = x_i$ the data points get clustered by moving them along the gradient from their initial position to the local minima of the potential function, i.e. the cluster centers, in discrete steps $y_i(t + \nabla t)$:

$$y_i(t + \nabla t) = y_i(t) - \eta(t)\nabla V(y_i(t)) \tag{4}$$

The time complexity for calculating the potential function is $O(n)$, where n is the number of data points, whereas the time complexity for each complete step of the gradient descent is $O(n^2)$. The complete calculation of Quantum Clustering, therefore, is of the order $O(m \cdot n^2)$ with m being the number of iterations in the gradient descent (Shaked, 2013). For large

datasets, the complexity of the algorithm makes calculations unfeasible. Therefore, we propose an adaptation of the algorithm integrating it into an existing framework for streaming data.

2.2 DenStream

In this study, we draw on the method of DenStream (C. Feng, 2006). It uses a damped window model, which considers the latest information given by data points by assigning weights, whereas more recent objects have higher weights than older objects. The weight ω of each data point decreases with time, following an exponentially decreasing function $f(x) = 2^{-\lambda t}$, in which λ is the decay constant. A high value of λ results in a faster decrease of old information.

Clustering in the data stream can be divided into two steps: the online component, also known as the data abstraction step, and the offline component at which the actual clustering is performed. In the data abstraction step, summary statistics of the data in the stream are stored in a feature vector. In the case of DenStream, data points get sampled into micro-clusters of outlier-micro-clusters (o-micro-cluster) or potential-micro-clusters (p-micro-cluster). An arriving data point gets merged into a p-micro-cluster if the resulting cluster radius after adding this data point lies within a maximum radius of ε. If this is not the case, the data point is tried to be merged into an existing o-micro-cluster. If the radius of this cluster then succeeds the maximum radius, ε, a new o-micro-cluster is created. The algorithm continuously checks the updated weights of the clusters after a defined time period. When a given threshold $\omega > \beta\mu$ for the weight is reached (the hyper-parameters β and μ have to be set respectively), o-micro-clusters evolve into p-micro-clusters. Due to the decay function, when no new points get clustered into an existing p-micro-cluster, it will become an o-micro-cluster since its weight ω will fall under the given threshold and subsequently be deleted from the outlier buffer. With this, DenStream reduces the required space in memory and takes care of the dynamical evolution of the probability distribution of the data stream. The offline component data get clustered in the original version by the DBSCAN algorithm. When a clustering request arrives, the p-micro-cluster-centers given at that time are treated as virtual data points with an assigned weight.

2.3 Quantum Clustering Implemented in DenStream Framework

Instead of DBSCAN, in this study, we apply Quantum Clustering as an offline clustering method, treating the p-micro-cluster-centers as data points for calculating the probability function. On the one hand, this approach is valuable as we overcome the high computational complexity of Quantum Clustering, finding a way to apply it to the use case of network intrusion detection. On the other hand, Quantum Clustering shows comparable to increased performance over the traditionally used DBSCAN. In the following, we will refer to the new method as $\text{DenStream}_{\text{QC}}$ for the adapted version of Quantum Clustering and $\text{DenStream}_{\text{DBSCAN}}$ for the conventional method. When a clustering request arrives on the data stream, the potential function gets calculated over the p-cluster-centers, and the algorithm clusters the p-micro-cluster-centers. After this clustering step, the data points get assigned the label of their nearest p-cluster center, following the distance-based approach suggested in the original framework. Using Quantum Clustering to cluster the data in the offline phase is a novel approach to fasten its computation time. The number of data points in calculating the potential function and the gradient descent gets reduced since we only consider the p-cluster-centers. Furthermore, through the dynamic calculation of the cluster centers, the clustering result reflects a real-time picture of the underlying data distribution. As with the DBSCAN algorithm DenStream uses, Quantum Clustering does not require a predefined number of clusters, which is important for the clustering data stream, as the number of clusters might change over time (C. Feng, 2006).

2.4 Experimental Design

In the first step, we compare the original $\text{DenStream}_{\text{DBSCAN}}$ with our new approach $\text{DenStream}_{\text{QC}}$ for cluster quality, calculating the metrics Purity and F1-Score over different time-windows for both methods for the CIDDS-001 and the IoTID20 dataset. To simulate a natural streaming data environment, the data is fed into the clustering algorithms in the order of the instances given by the data set. Employing the decay function, which decreases the cluster weights when no new data is assigned to that cluster, we further ensure that old data points get discarded when the clustering request arrives.

To evaluate the feasibility of our method for anomaly detection, we analyzed the results for each attack class respectively and calculated Precision, Recall, and Accuracy. We could do this since both datasets were labeled for the respective attack classes.

We then analyze the method in terms of memory requirement and running time compared with the

original Quantum Clustering to show the increased computational performance. All test runs were conducted on the same hardware, a 64-bit operating system with a CPU with 1.80 GHz and an available RAM of 8.00 GB.

2.5 Datasets

2.5.1 CIDDS-001 Dataset

The first dataset used in this study is the CIDDS-001 dataset (M. Ring, 2017). It is a labeled, flow-based dataset generated for anomaly-based intrusion evaluation. It consists of four weeks of network traffic data in total, generated through OpenStack and an External Server respectively. The data is labeled for five different classes: *normal* data traffic, *attack*, *victim*, *suspicious*, and *unknown*. Each flow instance is assigned to one class. The labeling of the flows happened as follows, according to the authors: flows that come from the OpenStack environment were only benign and therefore labeled as *normal*, flows coming from port 443, and 80 could be normal traffic or intrusion attempts categorized as *unknown*. The attacks came from three controlled servers; those flows were labeled as either *attack* or *victim*, respectively. All remaining traffic was labeled *suspicious*, leaving the class relatively broad and undefined (M. Ring, 2017).

In this study, we used the dataset of the third week, captured at the external server. We chose this set because of the variety of employed attacks: week 1 on the external server had no executed attacks and was therefore not interesting for our analysis, whereas our dataset had the most attacks executed, 12 in total, 5 of which were *PortScan* and 7 *Brute Force*. The set consists of 153026 instances and 8 attributes used for further processing: Source IP Address, Source Port, Destination IP Address, Destination Port, Proto, Duration, TCP-Flags, and Packets. Flows with class labels *attack* or *victim* are further categorized into the types of attacks. We use this distinction for our ground truth labeling for our analysis, resulting in seven class labels. Table (1) shows a summary of the dataset. We can see that the dominant class is the class labeled *suspicious* with more than half of the instances belonging to it. What is apparent is the imbalance in the dataset, with comparatively small classes *Brute Force attack* and *victim*.

2.5.2 IoTID20 Dataset

The second dataset used is the IoTID20 (I. Ullah, 2020), which is more focused on the context of the Internet of Things and contains a combination of IoT devices with interconnections. The data was generated in a smart home environment consisting of SKT NGU devices and an EZVIZ Wifi camera that is connected to a router. Other devices connected to that network include laptops, smartphones, and tablets. The SKT NGU and EZVIZ Wifi camera act as victim devices, whereas all the other devices in the architecture are attacking devices. The dataset consists of 80 network features and is labeled according to the attack classes *DoS*, *MITM*, *Mirai*, *Normal*, and *Scan*. In total, it contains 625783 flow instances, whereas we only took 150000 to test our model. As in the CIDDS-001 dataset, we used a categorical encoding method for the features' Source and Destination IP addresses. The other numerical features were normalized. Features contained in the dataset were Protocol Type, Duration, Packets, and Flag, to name a few. For further details about the data, we point the reader to the original paper (I. Ullah, 2020). For our analysis, we used the mutual information score as a feature selection method, reducing the number of features from 80 to 41. In Table (2), we depict a summary of the labeled instances, as well as the assigned label that we use in the result section to display the cluster results of the respective classes.

Table 1: CIDDS-001 dataset external server week 3.

Class	Total Flows	Label
bruteForce Attack	700	0
portScan Attack	8555	1
Normal	6180	2
Suspicious	97852	3
Unknown	33837	4
bruteForce Victim	700	5
portScan Victim	5202	6

Table 2: IoTID20 dataset; normal and attacked instances.

Class	Total Flows	Label
DoS	14149	0
MITM	8381	1
Mirai	99811	2
Normal	9519	3
Scan	18140	4

2.6 Metrics

Because both datasets contained labeled instances, we were able to use classification metrics for the clustering analysis. To quantify our results and compare it to the given label classes, we labeled the resulting clusters according to their most frequently occurring class. The Purity of the clustering is then calculated by summing the correct assigned data points in each class and dividing it by the number of total data points

N.

$$purity(\Omega, C) = \frac{1}{N}\sum_{k}\max_{j}|\omega_k \cap c_j| \quad (5)$$

here $\Omega = \{\omega_1, \omega_2, ..., \omega_k\}$ is the set of predicted clusters and C= $\{c_1, c_2, ..., c_j\}$ is the set of classes (Stanford, 2008).

Purity alone is not a good indicator of cluster quality since a high number of clusters automatically results in a higher Purity - if each data point is separately clustered in one class, the Purity is 1 (Stanford, 2008). We, therefore, additionally evaluate the clustering result with the Precision, Recall, Accuracy, and F1-Score, as described in Equations (6)-(9). Here TP is the true positive, TN is the true negative, FP is the false positive and FN is the false negative:

$$Precision = \frac{TP}{TP+FP} \quad (6)$$

$$Recall = \frac{TP}{TP+FN} \quad (7)$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (8)$$

$$F1-Score = 2 \cdot \frac{Precision * Recall}{Precision + Recall} \quad (9)$$

3 RESULTS

In this section, we test the performance of the streaming adapted version DenStream$_{QC}$ by evaluating its ability to detect patterns in the datasets, as well as its space and running time requirements in comparison to the original Quantum Clustering.

3.1 Cluster Quality

3.1.1 CIDDS-001 Dataset

We test the Purity and F1-Score for different periods of stream execution for the conventional DenStream algorithm using DBSCAN as its method for the clustering step and our version using Quantum Clustering. As the weights of the data points gradually decrease and are thus erased from memory and no longer available for clustering, we compute the cluster metrics for defined windows starting from the present time. Since lambda is the decisive parameter for how fast this decrease happens, the one for the comparison was chosen to be the same for both algorithms. We have chosen the parameters for both algorithms to achieve the highest possible Purity and F1-Score. Figure (1) and Figure (2) show that the performance for both metrics of DenStream$_{DBSCAN}$ is slightly better than that of DenStream$_{QC}$ across the windows. For both, DenStream$_{QC}$ and DenStream$_{DBSCAN}$, the values of Purity and F1-score are around 0.95 with a minimum F1-Score of 0.87 for DenStream$_{QC}$ and 0.89 for DenStream$_{DBSCAN}$. For DenStream$_{DBSCAN}$, both values remain relatively constant across the windows, with Purity and F1-Score decreasing slightly for DenStream$_{QC}$ for larger periods of stream execution.

As the attack clusters are highly underrepresented, the generalized metrics do not adequately indicate the cluster quality. We, therefore, added Tables (3)-(6) to compare the ability of the algorithms to detect small clusters. Figures (3) and (4) show the cluster results for DenStream$_{QC}$ and DenStream$_{DBSCAN}$, respectively, for the whole dataset. On the horizontal axis, we can see the detected classes. As described in the methods section, these have been labeled based on the most common class found in them. The vertical axis shows the ground truth class to which the data points actually belong. Tables (4) and (5) show that both algorithms detect the majority classes (e.g., suspicious, unknown) well, whereas DenStream$_{DBSCAN}$ has a slightly higher detection rate for this data set. On the other side, underrepresented classes, as its most common in network intrusion sets, could not be detected at all by DBSCAN. Therefore, the classes that stand out in the result of both methods are *bruteForce attacker and victim*. With DenStream$_{DBSCAN}$, none of the flows could be clustered correctly, but nearly entirely fell into the class *suspicious*. In the case of DenStream$_{QC}$, the flows of *bruteForce attacker* were clustered either as *normal* or *suspicious*, and the flows belonging to ground truth label *bruteForce victim* were nearly entirely clustered as *suspicious*.

In the case of DenStream$_{DBSCAN}$, adapting the hyperparameters did not improve the clustering result for the class *bruteForce* - varying the parameters only led to an overall deterioration in cluster quality. For DenStream$_{QC}$, however, we enhanced the clustering result of *bruteForce attacker* and *victim* for smaller values of ε. Figure (5) and Table (6) show the clustering result for a value of the maximum micro-cluster radius $\varepsilon = 0.1$ for DenStream$_{QC}$. This resulted in a significantly better cluster detection for *bruteForce attacker* and *victim* of 47% respectively in comparison to 0% for DenStream$_{DBSCAN}$. The flows which could not be assigned to the correct classes again predominantly fell into the category *suspicious*. We will discuss this result in more detail in the next sec-

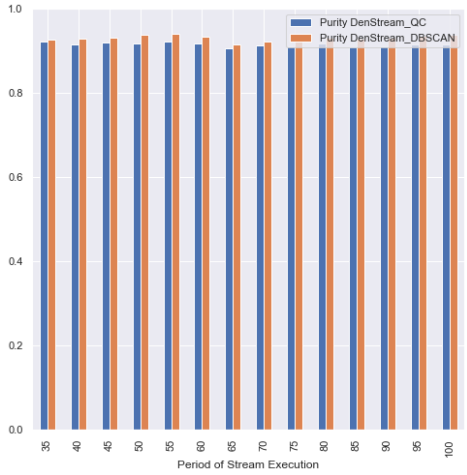

Figure 1: Purity for different window sizes for $\text{DenStream}_{\text{QC}}$ CIDDS-001 dataset.

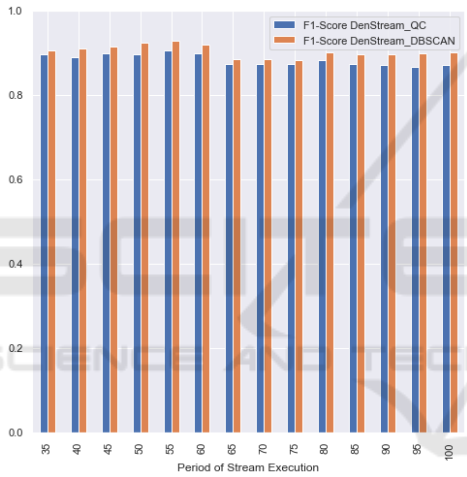

Figure 2: F1-Score for different window sizes for $\text{DenStream}_{\text{QC}}$ and $\text{DenStream}_{\text{DBSCAN}}$ CIDDS-001 dataset.

tion. Table (3) further shows the summary results of $\text{DenStream}_{\text{QC}}$ and $\text{DenStream}_{\text{DBSCAN}}$ of the metrics Precision, Recall, and Accuracy. For the CIDDS-001 dataset, $\text{DenStream}_{\text{QC}}$ achieved an accuracy of 0.91 for the clustering displayed in Figures (3) and (5), and $\text{DenStream}_{\text{DBSCAN}}$ an accuracy of 0.93. The methods for network intrusion detection must be tailored to the reliable detection of attacks. An assignment of instances as False-Positive is often more cost-effective than False-Negative, i.e., a non-detection of attack patterns can have more severe consequences than a slightly too sensitive attack analysis method. Figure (5) shows that after epsilon's fine-tuning, only 104 data flows were incorrectly clustered as *normal* from $\text{DenStream}_{\text{QC}}$. With $\text{DenStream}_{\text{DBSCAN}}$, 237 flows were not detected as attacks or potential attacks but clustered as *normal* network behavior. This observation is also reflected in the values for Recall. For $\text{DenStream}_{\text{QC}}$ this is 90% after fine-tuning; for $\text{DenStream}_{\text{DBSCAN}}$, 88% is the best case.

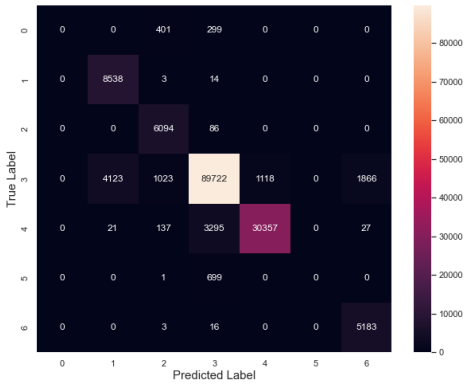

Figure 3: Result $\text{DenStream}_{\text{QC}}$ with $\varepsilon = 0.4$ CIDDS-001 dataset.

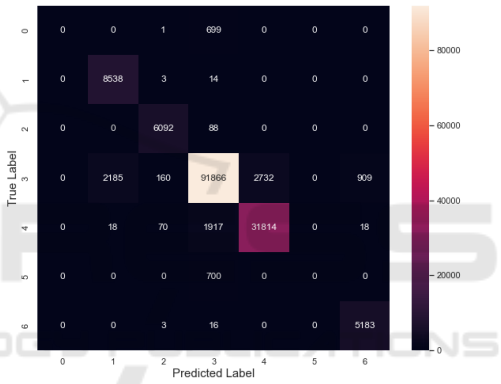

Figure 4: Clustering result $\text{DenStream}_{\text{DBSCAN}}$ CIDDS-001 dataset.

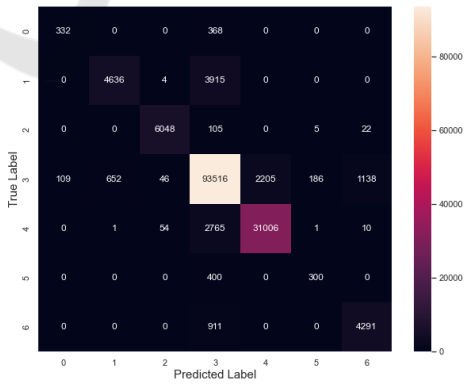

Figure 5: Result $\text{DenStream}_{\text{QC}}$ with $\varepsilon = 0.1$ CIDDS-001 dataset.

3.1.2 IoTID20-Dataset

For the IoTID20 dataset, we see that the Quantum clustering method achieves much better results than

Table 3: Metrics Clustering Results DenStream$_{QC}$.

Data	Precision	Recall	Accuracy
Fig. 3 DenStream$_{QC}$ CIDDS-001/	0.9034	0.8419	0.9142
Fig. 5 DenStream$_{QC}$ CIDDS-001	0.8461	0.9037	0.9157
Fig. 4 DenStream$_{DBSCAN}$ CIDDS-001	0.9176	0.8845	0.9377
Fig. 8 DenStream$_{QC}$ IoTID20	0.5939	0.9521	0.7301
Fig. 9 DenStream$_{DBSCAN}$ IoTID20	0.5747	0.8658	0.7033

Table 4: Summary results DenStream$_{QC}$ with $\varepsilon = 0.4$ CIDDS-001; Figure 3.

Class	correctly clustered instances
bruteForce attacker	$0 \equiv 0\%$
portScan attacker	$8538 \equiv 99\%$
Normal	$6094 \equiv 98\%$
suspicious	$89722 \equiv 92\%$
unknown	$30357 \equiv 90\%$
bruteForce victim	$0 \equiv 0\%$
portScan victim	$5183 \equiv 99\%$

Table 5: Summary results DenStream$_{DBSCAN}$ CIDDS-001; Figure 4.

Class	correctly clustered instances
bruteForce attacker	$0 \equiv 0\%$
portScan attacker	$8538 \equiv 99\%$
Normal	$6092 \equiv 98\%$
suspicious	$91866 \equiv 94\%$
unknown	$31814 \equiv 94\%$
bruteForce victim	$0 \equiv 0\%$
portScan victim	$5183 \equiv 99\%$

Table 6: Summary results DenStream$_{QC}$ with $\varepsilon = 0.1$ CIDDS-001; Figure 5.

Class	correctly clustered instances
bruteForce attacker	$332 \equiv 47\%$
portScan attacker	$4636 \equiv 54\%$
Normal	$6048 \equiv 97\%$
suspicious	$93516 \equiv 95\%$
unknown	$31006 \equiv 91\%$
bruteForce victim	$300 \equiv 47\%$
portScan victim	$4291 \equiv 82\%$

the original DenStream$_{DBSCAN}$ algorithm. The overall performance of the methods for the second dataset is not as good as for CIDDS-001, indicating the complexity of the data produced in an IoT environment. For the Purity and F1-measure, DenStream$_{QC}$ shows a constant performance over window sizes, with an average score of 0.7 in both metrics, whereas DenStream$_{DBSCAN}$ only achieves an average result of around 0.65.

Similar to the CIDDS-001 dataset, some attack classes tend to be overpredicted by the clustering; in the IoTID20 dataset, it is the class *Mirai*. In the case of DenStream$_{DBSCAN}$, the class *DoS* was overpredicted as well - in the end, the method only clustered the instances into these two classes. Again this is probably because of the imbalance in the dataset, where the majority of instances belong to this attack class, and especially DenStream$_{DBSCAN}$ has problems detecting the small clusters in the large dataset. Our method shows very good results for the differentiation of the *DoS* attack class, with a correct prediction of 99% of the instances belonging to this category, and predicted in total only 13 instances False-Positive for this class. Also, DenStream$_{QC}$ detected the *Scan* attack much better with 41% compared to 0% for DenStream$_{DBSCAN}$. The Accuracy of DenStream$_{QC}$ for clustering the data into the different attack classes was 0.73, as shown in Table (3). We see that although DenStream$_{DBSCAN}$ performed very badly for this dataset, its accuracy is still higher than one would expect. That is because it overpredicted the classes with the most instances, which distorts the result. Therefore the analysis and comparison of individual cluster assignments are important for imbalanced data.

Other studies that analyzed the IoTID20 dataset using clustering techniques didn't report on individual results of the respective attack class categories, which is why a comparison in this regard is difficult. In fact, most studies we found deploying IoTID20 used a supervised learning approach (R. Qaddoura, 2021), (A.Y. Hussein, 2021), (H. Alkahtani, 2021), (S. Bajpai, 2021), (T.V. Ramana, 2022). For real-world data, such approaches remain often unfeasible, as attack patterns are often not known beforehand, or no prior attack data is available. Quantum Clustering is able to recognize some classes with high accuracy, which is a promising result and can be used as a benchmark result for other clustering studies in the domain of anomaly detection.

Table 7: Summary results DenStream$_{QC}$ with $\varepsilon = 0.8$ IoTID20; Figure 9.

Class	correctly clustered instances
Normal	$6108 \equiv 64\%$
DoS	$14030 \equiv 99\%$
Mirai	$90104 \equiv 90\%$
MITM	$1393 \equiv 16\%$
Scan	$7366 \equiv 41\%$

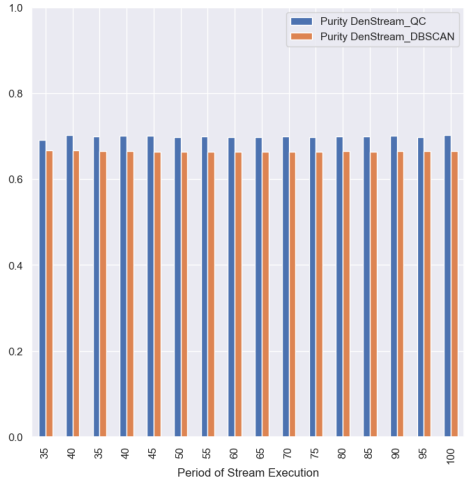

Figure 6: Purity for different window sizes for $\text{DenStream}_{\text{QC}}$ and $\text{DenStream}_{\text{DBSCAN}}$ IoTID20 dataset.

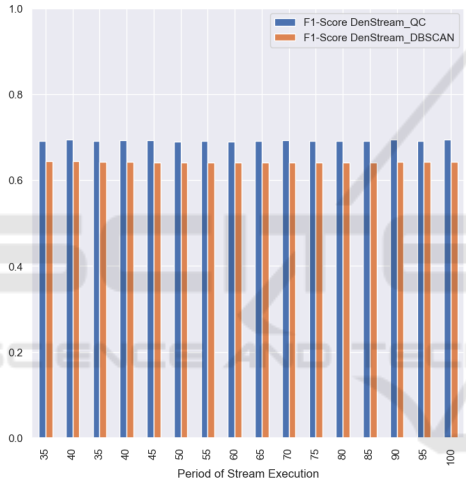

Figure 7: F1-Score for different window sizes for $\text{DenStream}_{\text{QC}}$ and $\text{DenStream}_{\text{DBSCAN}}$ IoTID20 dataset.

Table 8: Summary results $\text{DenStream}_{\text{DBSCAN}}$ IoTID20; Figure 10.

Class	correctly clustered instances
Normal	$0 \equiv 0\%$
DoS	$11536 \equiv 82\%$
Mirai	$93774 \equiv 94\%$
MITM	$0 \equiv 0\%$
Scan	$0 \equiv 0\%$

3.2 Runtime and Memory Analysis

In this section, we compare the runtime and memory requirement of our new method and the original Quantum Clustering. We only do this for the CIDDS-001 dataset, which is sufficient since the size of both datasets is almost the same.

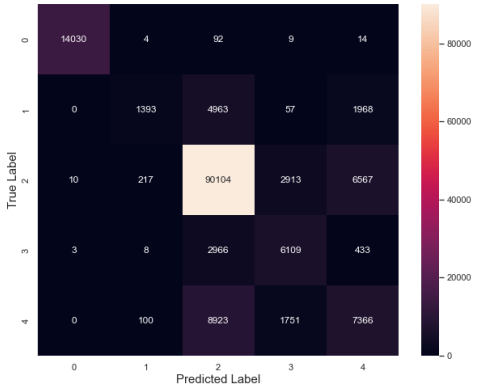

Figure 8: Clustering result $\text{DenStream}_{\text{QC}}$ with $\varepsilon = 0.8$ IoTID20 dataset.

Figure 9: Clustering result $\text{DenStream}_{\text{DBSCAN}}$ IoTID20 dataset.

In the original version of Quantum Clustering, the computational complexity makes the algorithm infeasible for large streaming datasets. Therefore, one of the main foci of our study was to find a way to improve the runtime performance and memory space requirement of the clustering. We could not test the original version of Quantum Clustering for the whole dataset as the computational demand exceeded the available RAM capacities. We only used the first forty percent of the dataset. As expected, there were no memory capacity problems for the version adapted for streaming data, so we used the complete dataset.

In the graphs for the running time (Figures 10 and 11), we see the time measured on the hardware required to run the algorithm for different numbers of instances. In the graphs showing the memory requirements, we executed one run for the complete dataset for $\text{DenStream}_{\text{QC}}$ (Figure 12) and one run using only forty percent of the data for the original Quantum Clustering (Figure 13). The graphs show the RAM required for different time points in the algorithm execution and highlight the high computational demand of the original Quantum Clustering approach in

comparison to DenStream$_{QC}$. In the original Quantum Clustering, we are already close to the maximum RAM capacity available for the test run by only using a fraction of the data.

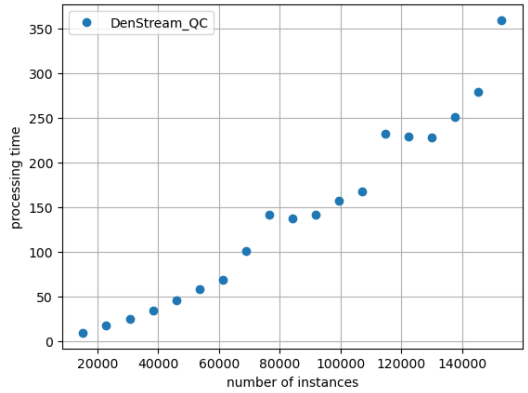

Figure 10: Running time DenStream$_{QC}$ for all instances of the dataset CIDDS-001.

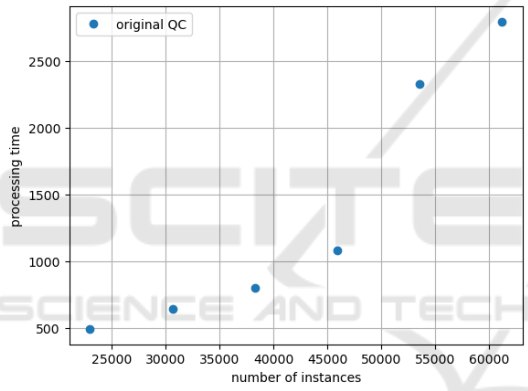

Figure 11: Running time for the original Quantum Clustering for 40% instances of the dataset CIDDS-001.

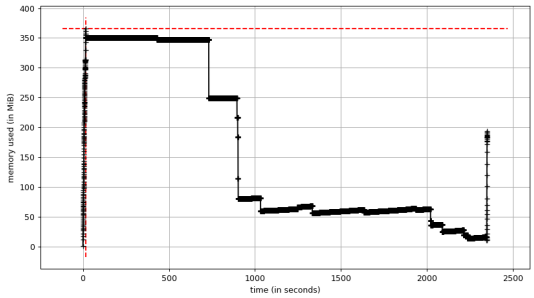

Figure 12: Memory requirement in DenStream$_{QC}$ for all instances of the dataset CIDDS-001.

4 DISCUSSION

The implementation of Quantum Clustering into the framework of DenStream for streaming data leads to a

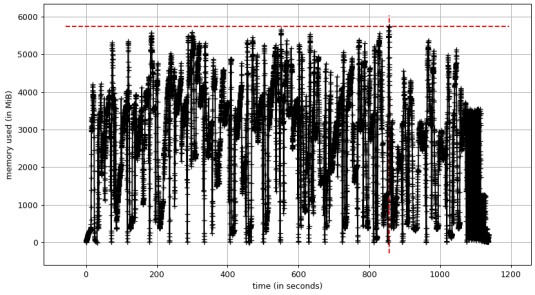

Figure 13: Memory requirement in the original Quantum Clustering for 40% instances of the dataset CIDDS-001.

significant reduction in runtime and memory requirements. Although runtime and memory consumption is not stable for different hardware, we want to illustrate the improved performance by highlighting the clear difference between the streaming adapted version DenStream$_{QC}$ compared to the original method. This reduction is achieved as the potential function, and the subsequent clustering is not performed over all data points of the dataset but only on those who are present in memory at the time when clustering is requested. We could only evaluate forty percent of the dataset for the original version of Quantum Clustering because of exhausted RAM capacity. We have therefore refrained from comparing the cluster results of Quantum Clustering with Denstream$_{QC}$. The small number of data flows to be evaluated and the high computation time excludes the original algorithm from being applied to streaming data. As far as we know, our proposed method DenStream$_{QC}$ is the first use case where it was possible to use Quantum Clustering to analyze data streams and further extend its applicability to intrusion detection.

We are interested in detecting malign network behavior and attack patterns in real time for this use case. From Table (4), we can see that our streaming adopted method could assign all classes except *bruteForce attacks* and *bruteForce victims* very well for the CIDDS-001 dataset, with a 90%-99% match for each of the labels, respectively.

Overall, 91% of the instances were clustered correctly. When false clustering happens, the flows are most often assigned to the class *suspicious* for both methods, DenStream$_{QC}$ and DenStream$_{DBSCAN}$. This tendency of disproportionately frequent assignment to these categories could also be observed after the adaptation of the hyperparameters. From these results, we can conclude that this class seems to be not clearly defined and distinguishable for machine learning models, i.e., flows that fall into this category are not as clearly delineated as those of the other classes. This was also pointed out in (M. Ring, 2017) and (J. Carneiro, 2022) and comes from the labeling

process described above. The class *suspicious* represents a broad container term for flows that are neither attacks, flows from the OpenStack environment, nor flows coming from ports 80 and 443. Another reason for the overabundant assignment to the class *suspicious* is the imbalance in the dataset, where it is by far the largest in the number of flows.

On the other hand, this imbalance makes it hard for the algorithm to identify the classes *bruteForce victim* and *bruteForce attacker*. Both classes represent the smallest number of flows, making it difficult for the algorithm to detect their patterns. Comparing our method with the performance of $\text{DenStream}_{\text{DBSCAN}}$ leads to similar results in purity and F1-Score. However, $\text{DenStream}_{\text{QC}}$ was better at detecting the small-sized clusters of *bruteForce victim* and *bruteForce attacker* in the data. By choosing appropriate parameters, it was possible to obtain a better delimitation of these classes, which was not possible for $\text{DenStream}_{\text{DBSCAN}}$. Further research has to be done to test $\text{DenStream}_{\text{QC}}$'s sensitivity to data patterns and to various hyperparameters settings. Nevertheless, the observations show the possibility of using $\text{DenStream}_{\text{QC}}$ in a combination with variable hyperparameter settings when dealing with imbalanced data to make hardly discernible patterns visible. Furthermore, the better Recall value makes it a promising tool, especially in the context of intrusion detection, where sensitivity to pattern changes and outliers is needed.

Similar conclusions can be drawn from the analysis of the IoTID20 dataset. Also, this dataset has an imbalance in the classes, where the class *Mirai* is the biggest one with 99811 instances. Since also for this dataset, the majority of wrong-clustered instances fall into this category, we can conclude that $\text{DenStream}_{\text{QC}}$ still needs to be fine-tuned for imbalances in data and the detection of outliers. Still, in comparison to the original $\text{DenStream}_{\text{DBSCAN}}$ method, Quantum Clustering shows good performance for both datasets. $\text{DenStream}_{\text{QC}}$ is able to locate clusters that are difficult to delineate for $\text{DenStream}_{\text{DBSCAN}}$. This is especially evident for the IoTID20 dataset. The method of constructing the potential function to estimate the data density distribution allows quantum clustering to pick up dynamic patterns in the data. As studies have previously shown, quantum clustering shows superiority in finding outliers, compared to DBSCAN (D. Liu, 2016). Our results in a direct comparison of the two methods in the framework of stream clustering confirm this trend.

We also point to the need for more data stream clustering studies using this dataset. Hence, a quantitative comparison of our method with other standard streaming methods in terms of cluster quality was out of the scope of this study.

Overall, the results suggest that $\text{DenStream}_{\text{QC}}$ is an exciting new method in the repertoire of streaming algorithms that brings the advantages of the original Quantum Clustering shown in previous studies (D. Liu, 2016), (D. Liu, 2020), (Deutsch, 2017), (R.V. Casana-Eslava, 2020), (Shaked, 2013) to the context of big data. The reduced memory and running time requirements of the method can benefit applications on devices that cope with limited resources but must, at the same time, ensure secure data analysis. Constrained devices can be found in the context of the internet of things a lot, pointing to an important use case area of $\text{DenStream}_{\text{QC}}$ - especially in the context of security and resilience, where continuous monitoring and real-time analysis of systems are required, and standard solutions fail to deliver the desired performance (N. Ntuli, 2016). With this study, we address the question of how security aspects in the IoT can be ensured despite limited capacities, suggesting a method that can provide fast, reliable, and computationally effective results.

5 CONCLUSION

This paper proposed a variation of the original algorithm Quantum Clustering as an implementation into the framework of DenStream. We show that the Quantum Clustering approach achieves solid results compared to State-of-the-Art algorithms like DBSCAN. It was able to detect small clusters in an unbalanced dataset, which was not possible for DBSCAN. This makes the Quantum Clustering approach particularly suitable for data consisting of small clusters, such as the network intrusion domain. It could also be shown, by comparing the runtime and memory costs, that the adapted quantum clustering approach could be significantly enhanced in processing speed, rendering it a suitable application for various network domains. This advancement allows the use of optimization methods of hyperparameters to detect different network anomalies. These results make us confident that $\text{DenStream}_{\text{QC}}$ can be used for further streaming data analysis applications.

REFERENCES

A. Zubaroglu, V. A. (2021). Data Stream Clustering: A review. *In: Artificial Intelligence Review)*, 54.

A.Y. Hussein, P. Falcarin, A. S. (2021). Enhancement performance of random forest algorithm via one hot en-

coding for IoT IDS. *Periodicals of Engineering and Natural Sciences*, 9.

C. Aggrawal, J. Han, J. W. P. Y. (2003). A framework for clustering evolving data streams. *Proceedings of the 29th International Conference on Very Large Data Bases*, 29.

C. Feng, M. Ester, W. Q. A. Z. (2006). Density-Based Clustering over an Evolving Data Stream with Noise. *In: Proceedings of the 2006 SIAM International Conference on Data Mining*, SDM.

D. Horn, A. G. (2001). The method of Quantum Clustering. *In: Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS 01.

D. Liu, H. L. (2020). Outlier Detection Using a Novel method. *Arxiv*.

D. Liu, M. Jiang, X. Y. H. L. (2016). Analyzing documents with Quantum Clustering: A novel pattern recognition algorithm based on quantum mechanics. *Pattern Recognition Letters*, 77.

D.E. Kouicern, A. Bouabdallah, H. L. (2018). Internet of things security: A top-down survey. *Computer Networks*, 141.

Deutsch, L. (2017). Quantum Clustering and its Application to Asteroid Spectral Taxonomy. *Tel Aviv University*.

F. Cao, M. Ester, W. Q. A. Z. (2006). Density-based clustering over an evolving data stream with noise. *Proceedings of the 2006 International Conference on Data Mining*, 6.

H. Alkahtani, T. A. (2021). Intrusion Detection System to Advance Internet of Things Infrastructure-Based Deep Learning Algorithms. *Complexity*, 2021.

I. Butun, S.D. Morgera, R. S. (2014). A Survey of Intrusion Detection Systems in Wireless Sensor Networks. *IEEE Communications Survey & Tutorials*, 16.

I. Ullah, Q. M. (2020). A Scheme for Generating a Dataset for Anomalous Activity Detection in IoT Networks. *Advances in Artificial Intelligence: 33rd Canadian Conference on Artifial Intelligence*.

J. A. Silva, E.R. Faria, R. B. E. H. (2013). Data Stream Clustering: A Survey. *In: ACM Computing Surveys*, 46.

J. Carneiro, N. Oliveira, N. S. E. M. I. P. (2022). Machine Learning for Network-based Intrusion Detection Systems: an Analysis of the CIDDS- 001 Dataset. *In: Distributed Computing and Artificial Intelligence*, 1.

J. Diaz-Rozo, C. Bielza, P. L. (2018). Clustering of Data Streams with Dynamic Gaussian Mixture Models. An IoT Application in Industrial Processes. *Internet of Things Journal*, Special Issue on Real-Time Data Processing for Internet of Things.

K.H. Han, J. K. (2002). Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. *IEEE Transactions on Evolutionary Computation*, 6.

L. Wan, W. N. (2009). Density based Clustering of Data Streams at Multiple Resolutions. *ACM Transactions on Knowledge Discovery from Data*, 3.

M. Ring, S. Wunderlich, D. G. (2017). Technical Report CIDDS-001 data set. *In: Proceedings of the 16th European Conference on Cyber Warfare and Security (ECCWS)*, ACPI.

M.M. Noor, W. H. (2019). Current research on Internet of Things (IoT) security: A survey. *Computer Networks*, 148.

M.R. Ackermann, M. Märtens, C. R. K. S. C. L. C. S. (2012). StreamKM++: A clustering algorithm for data streams. *Journal of Experimental Algorithmics*, 17.

N. Ntuli, A. A.-M. (2016). A Simple Security Architecture for Smart Water Management System. *In: Procedia Computer Science*, 83.

P. Casas, J. Mazel, P. O. (2012). Unsupervised Network Intrusion Detection Systems: Detecting the Unknown without Knowledge. *Computer Communications*, 7.

P. Jiménez, J.C. Roldán, R. C. (2022). A hybrid quantum approach to leveraging data from HTML tables. *In: Knowledge and Information Systems*, 64.

P. Kranen, I. Assent, C. B. T. S. (2011). The ClusTree: indexing micro-clusters for anytime stream mining. *Knowledge and Information Systems*, 29.

R. Qaddoura, A.M. Al-Zoubi, I. A. H. F. (2021). A Multi-Stage Classification Approach for IoT Intrusion Detection Based on Clustering with Oversampling. *Applied Scineces*, 11.

R.V. Casana-Eslava, P. Lisboa, S. O.-M. I. J. J. M.-G. (2020). Probabilistic Quantum Clustering. *Knowledge-Based Systems*, 194.

S. Bajpai, K. S. (2021). A Framework for Intrusion Detection Models for IoT Networks using Deep Learning. *Research Square*, 2021.

Shaked, G. (2013). Quantum Clustering for Large Data Sets. *Tel Aviv University*.

Stanford, N. (2008). Evaluation of clustering. *https://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html*.

T. Felser, M. Trenti, L. S. A. G. D. Z. D. L.-S. M. (2020). Quantum inspired machine learning on high-energy physics data. *npj Quantum Inf*, 7.

T. Zhang, R. Ramakrishan, M. L. (1997). BIRCH: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1.

Tang, E. (2019). A quantum-inspired classical algorithm for recommendation systems. *STOC 2019: Proceedings of the 51th Annual ACM SIGACT Symposium on Theory of Computing*.

T.Q. Duong, J.A. Ansere, B. N. (2022). Quantum-Inspired Machine Learning for 6G: Fundamentals, Security, Resource Allocations, Challenges, and Future Research Directions. *IEEE Open Journal of Vehicular Technology*, 3.

T.V. Ramana, M. Thirunavukkarasan, A. M. G. D. S. N. (2022). Ambient intelligence approach: Internet of Things based decision performance analysis for intrusion detection. *Computer Communications*, 195.

Y. Chen, L. T. (2006). Density-based clustering for real-time stream data. *Proceeding of the 2006 International Conference on Data Mining*, 6.

Y. Mahmoudi, N. Zioui, H. B. (2020). A new quantum-inspired clustering method for reducing energy consumption in IOT networks. *npj Quantum Inf*, 7.