# INTRUSION DETECTION IN IOT SYSTEMS


Akshat Khandelwal
Computer Science and Engineering
AIET Moodbidri, India
akshatalr.ak@gmail.com

Jagath Haren
Computer Science and Engineering
AIET Moodbidri, India
jagathharen15@gmail.com

Jayalakshmi. M
Computer Science and Engineering
AIET Moodbidri, India
jayanaik9090@gmail.com

Jayraj Chiragkumar Shah
Computer Science and Engineering
AIET Moodbidri, India
jayrajcshah@gmail.com

Jhon Alsten Tauro
Computer Science and Engineering
AIET Moodbidri, India
alstonro45@gmail.com

K. Thrishul
Computer Science and Engineering
AIET Moodbidri, India
acharyathrishul@gmail.com



***Abstract*:**

**The Internet of Things (IoT) connects, interacts, and exchanges data between a vast number of devices. The IoT network opens up a slew of new attack vectors that can jeopardise the integrity of data and systems. Over the last two decades, intrusion detection systems have been investigated extensively, with the majority of methods relying on classical machine learning and data mining approaches that need an offline period for model training on massive volumes of data. This study describes three data stream novelty detection approaches that have been used to the intrusion detection issue, as well as IDSA-IoT, an unique implementation architecture that integrates the usage of network edge resources with a public cloud.**



***Keywords*: Internet of Things, Sensors, Aggregator, Communication Channel, eUtility, Decision Trigger.**


## I. INTRODUCTION

The Internet of Tings (iot) is considered as a technology and economic wave in the global information industry after the Internet. The Internet of Things (iot) is a smart network that links everything to the Internet with the goal of sharing data and interacting via data sensing devices that follow agreed-upon protocols. By permitting comparisons between nots, it achieves the objective of intelligent identifying, finding, tracking, monitoring, and controlling objects with the use of primitives. The word primitives refers to the tiny components that will be used to construct larger blocks or systems. As a result, primitives provide a common vocabulary that allows for network composition and data interchange.

According to a Gartner report issued in 2017, by 2020, there will be around 20 billion gadgets connected to the Internet, many of them via the Internet of Things.According to a Gartner report issued in 2017, by 2020, there will be around 20 billion gadgets connected to the Internet, many of them via the Internet of Things. According to a 2018 poll by the same business, approximately

20% of firms had faced at least one iot-based assault in the previous three years. According to the findings, most firms have no control over the origin and type of connected device software and hardware. To combat these risks, iot security investment will rise from $1.5 billion in 2018 to $3.1 billion in 2021 [2], including tools and services to improve asset detection and management, software security review, and network security.

The Internet of Things (iot) connects a wide range of devices, including mobile phones, wearables, consumer electronics, automobiles, and different sensors. These gadgets link to other devices, systems, and apps on the backend through the Internet. They can be used to attack other devices and systems, steal information, do instant physical harm, or perform a variety of other malicious operations after they've been hacked. The majority of them will most likely have a lengthy lifespan and require fewer software changes. The growing mesh of gadgets from various technologies increases the attack surface significantly. In this context, cybersecurity specialists and front-line workers are now faced with combating new forms of threats that are becoming more common.

Intrusion Detection Systems (IDS) are critical defence tools for business networks. Intrusion detection has long been a research topic, and the majority of related work relies on Data Mining (DM) or Machine Learning (ML) approaches to identify assaults based on existing patterns or to uncover new patterns. The data collection is static and can be scanned repeatedly with classic data mining methods, and detecting new attack patterns necessitates a fresh cycle of training, testing, and distribution of new models. Unlike traditional approaches, stream mining algorithms have various benefits when it comes to intrusion detection, such as I working with little memory, allowing for implementation in tiny devices (for example, on the edge of the Web); (ii) processing traffic data with a single read; (iii) delivering real-time reaction; and (iv) identifying novelty and changes in previously acquired ideas.

Depending on the number of devices and their physical location, detecting online intrusions might be difficult. Due to high latency, transporting traffic data from the devices where it is gathered to be processed on a standard cloud or datacenter can be costly or prohibitive when there are hundreds or thousands of iot devices and objects distributed throughout corporate networks or smart cities. Moving massive amounts of traffic data from a large number of devices to a centralized infrastructure to be scanned is also not scalable. The existing cloud computing architecture will struggle to achieve the low latency and scalability criteria required for intrusion detection. To meet this problem, tiny devices and objects can be used to approximate the intrusion detection function processing, taking use of the resources available on small devices that occupy the network's edge.

We propose a distributed intrusion detection system for iot situations involving hundreds or thousands of items and devices in the current paper. There are two significant contributions. To begin, we suggest an intrusion detection architecture that takes advantage of cloud edge capabilities in order to reduce latency and improve security.

Scalability is being improved. In addition, three Novelty Detection (ND) approaches are used to understand developing network traffic patterns, which we analyse. Our concept uses edge network resources to gather and analyse data streams, as well as the public cloud to process offline data for more accurate results.

## II. NOVELTY DETECTION

The observation that an example varies in some manner from prior instances is known as novelty detection (ND). It permits new patterns to be recognized, which might lead to the formation of a new notion, the development of an existing notion, or the existence of noise. When it comes to intrusion detection, ND is a useful tool in a variety of ways. It may, for example, be advantageous to recognize when certain cases are

part of a developing pattern early on rather than misclassifying them in a recognized class. Traditional machine learning algorithms require a new offline phase to retrain the model on fresh data and incorporate knowledge of developing attack patterns into the decision model. This process slows down the application of new knowledge and uses up computer resources. Furthermore, supervised learning approaches are costly since they need the labelling of a large number of samples for model training by a professional. ND, on the other hand, tries to find new and relevant behaviors in the network and swiftly change the decision model to recognize them as they appear.

There are generally two steps to ND methods for data streams. The first is a static, offline phase in which a dataset is tagged and used to learn the previously learned concepts. In the offline period, three things might change: (i) the number of classes associated with known concepts: just normal and attack classes, or discriminating several attack classes; (ii) the number of classifiers: single or multiple classifiers; and (iii) the learning task, which specifies which algorithms will be used to build the decision model, such as decision trees and KNN, which need the presence of labels for training or clustering.

During the second, online phase, new data that is constantly arriving is examined. Three objectives are completed in this phase: categorization of new cases, detection of novelty patterns, and decision model adaptation. The categorization process entails examining each new case to see if it can be explained by the present decision model. If it can't, it'll be classified as unknown and evaluated again in the future to look for new patterns.

Finally, in the third step the decision model must be updated, with or without specialist feedback. If specialist feedback is used, then a human expert should label the data, otherwise the model will only add the information of a new found pattern, without knowing if this pattern corresponds to normal or malicious traffic.

While obtaining personalized feedback for all cases is impossible to obtain in real-world scenarios, this suggests that the methodology utilized has a good ceiling for performance and accuracy. The scenario defined by the absence of specialized feedback, on the other hand, reflects the worst case scenario of this procedure. When only a little quantity of input is offered and just a subset of cases is made available to the expert, this is a situation that is closer to what happens in real life. Active learning or semi-supervised approaches can be used to accomplish this.

Ecsminer is an example of a complete feedback approach that employs a series of decision trees to represent known ideas. This methodology may be defined as a classifier capable of identifying just one new class in the online phase, which employs the unknown label, has a forgetting mechanism, and expects entire expert feedback at a certain time period, according to the taxonomy. MINAS and anynovel, on the other hand, are novelty detection algorithms that describe the decision model with a collection of clusters. The fundamental benefit of these methods is that they allow for decision model updates with or without external inputs. Both systems employ a single classifier and a short-time memory to hold the so-called unknown cases, which are examples that are not explained by the present decision model. The decision model is updated using these examples. The primary distinction between both techniques is that anynovel does not change stable existing notions.

## III. PROPOSED ARCHITECTURE

Stream processing methods are used to detect intrusions at the network's edge, close to iot devices and sensors. Below is a diagram of an architecture that uses stream processing techniques to implement intrusion detection. Local resources, such as the Single Board Computer (SBC) or routers with suitable computing resources, are utilised, as indicated in

Figure 1. As a result, traffic processing takes place at the network's edge, avoiding data transfer to the network's core.

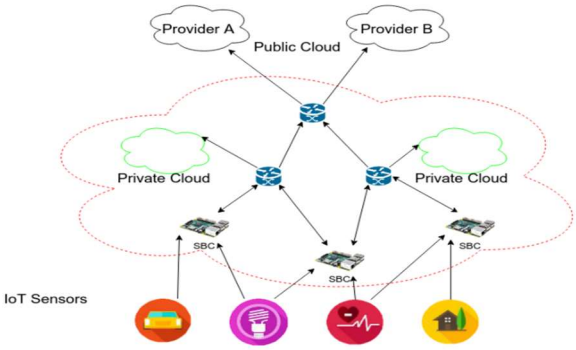


Fig. 1. Physical architecture

Fig. 2 shows the steps and where they are executed. Fig. 3 presents a more abstract view of the architecture, highlighting the offline and online processing steps.

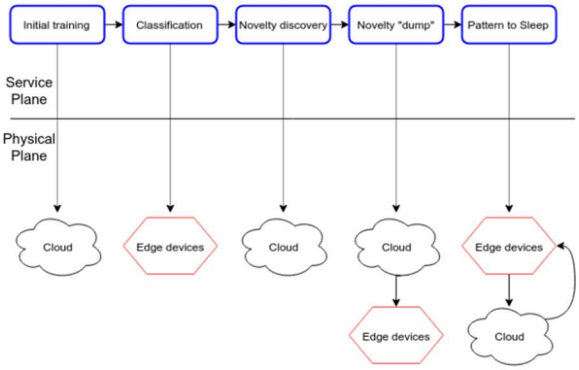


Fig. 2. Service distribution

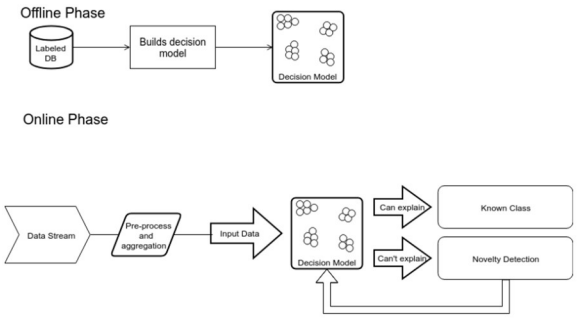


Fig. 3. Logical architecture.

This allows for low-latency intrusion detection and eliminates the requirement for every traffic to be examined in the network's core, such as in large data centres or the cloud, boosting system scalability. Only data that the model does not explain and that can be utilised to improve the models must be sent to the public cloud. Data from diverse network locations can be stored in the cloud until it is examined for training new models or making improvements to current ones.

The process begins with an offline step in which decision models are created from labelled data sets. Because this stage may demand a lot of computing power, it's done in the cloud.

Models are then sent to devices at the network's edge, where data streams are processed. Preprocessing of data streams, aggregation, and categorization of data are the first steps in the online phase, which correspond to the classification service in Figure 2. These tasks are dispersed amongst several devices at the network's edge.

If the model is unable to explain the example, it will be sent to the novelty discovery phase. The novelty detection service kicks in once the cloud has received a sufficient number of unexplained cases from various devices in numerous network locations. If valid groups (patterns) are discovered, new models must be distributed to the ND edge devices. Alternatively, before the novelty patterns are propagated throughout the system, a specialist can examine them. The knowledge sharing service corresponds to this step.

Because different network locations function differently, some patterns may become obsolete in some areas. Local models can then relocate these patterns to the sleep memory in this situation. The edge device must alert the cloud whenever a pattern is submitted to the sleep buffer. If a sufficient number of edge devices (specified by a threshold) shift the same pattern to the sleep buffer, the cloud sends a message to all edge devices directing them to transfer the same pattern to the sleep buffer, resulting in a more consistent pattern. The Obsolete Mechanism service represents this step.

# IV. EXPERIMENTS

We used the Kyoto 2006+ dataset for the tests, which covers data collected from 2006 to December 2015 [20] 1. Data was collected continually from 348 Honeypots of various types, including Windows (many releases), Linux / Unix (Solaris 8, macos X), network printers, and home appliances (e.g., TV set, HDD Recorder), to name a few.

Three of the attributes were generated by commercial IDS with malware and shell script detection capabilities, and the dataset comprises 24 attributes plus four label attributes. The fourth label attribute separates typical assaults from known attacks and unknown assaults.

The majority of the attributes are numeric fields that needed to be normalised. However, it's difficult to predict ahead of time what the highest values for parameters like duration, bytes received, and bytes transmitted will be in the stream. We set maximum values in advance for such properties. Nominal qualities have been eliminated. We chose instances from a single month in 2015, December. Only known attack types and known IDS warning code instances with at least 10,000 occurrences (for significance) were evaluated. The holdout strategy was used to do offline training with 72,000 examples (about 10% of the dataset).

## A. Assessing the Quality of Detection

The first set of experiments assesses the ND's quality. Fnew, as defined in [17], is the percentage of normal class cases that are improperly classified as belonging to a novelty or extension, whereas Mnew is the proportion of novelty examples that are improperly categorised as normal class. The quotient of the sum F N + F P and the number of classified cases determines the inaccuracy. Another metric considered was the number of expert-labeled examples required for each methodology. The effectiveness of three ND approaches that use total or partial feedback were tested.We ran tests to see how well the ecsminer methodology worked with increasing amounts of labels given while keeping the latency constant. Figure 4 depicts the technique's standard behavior, in which every label is available to the model after 50,000 examples. Figures 5 and 6 depict a modified version that attempts to match the characteristics of a real-world scenario in which the number of labels provided is 20% and 1%, respectively.

The number of vertical lines is maybe the most noticeable difference between the figures. More lines appear as the model receives fewer labels. The lines in Fig. 4 are between timestamps 0 and 50,000, indicating that while the model cannot interpret the examples, it cannot update itself since it lacks labels, resulting in an alarm. When the labels start to arrive, the model tries to include this new class, which causes Fnew to rise. Because this is a set of binary data, the type of attack is irrelevant, hence dos and Scan attacks are classified as the same thing despite their differences. Because of this aspect of the dataset, the model misclassified samples of a known class as novelties. On Figs. 5 and 6, notice how the lines extend beyond timestamp 50,000. Because ecsminer uses labelled instances to train a new tree for the ensemble, the process will take longer if the model is receiving less labels.

Based on these experiments, it is possible to say that tree ensemble is a good technique to classify security datasets. Given that the Fnew measure was less than 3% in all experiments and the Error was less than 5%, Even with less labels supplied, the results were constant; nevertheless, the amount of input required remained substantial (7,200 labelled examples on Fig. 6 and 144,000 on Fig. 5). Another disadvantage of this method is that it takes a long time to process data when compared to other methods (more information on Subsection V-B). As a result, we looked for alternate ways that could be processed more quickly in a limited device and required less expert comment.

We tested MINAS with a variety of additional parameters and found that it maintained the same Fnew rate for the majority of them, implying that this technique is robust in terms of ND (as opposed to anynovel, which had a wide range of measurements), but has to be improved in terms of error rate. The anynovel result, on the other hand, reveals that the Fnew is growing most of the time. This suggests that the approach is having trouble identifying innovations appropriately. Another example with anynovel is where Fnew is less than 2% for the entire execution, but Mnew is greater than 90%, showing a tradeoff between detection accuracy and detection volume.

There's also a tradeoff between the number of labelled instances required and the type of model utilised. As expected, a limited number of labelled instances results in a greater error and Fnew, whereas a large number of labelled examples results in a smaller error and Fnew. The ensemble of trees produced superior results, although it requires more labels than the two clustering methods. Finally, regardless of the labels provided, clustering techniques can react more quickly to flow fluctuations.

Anynovel is not suited for intrusion detection, as evidenced by the large error in Fig. 8, although it does give useful information. We selected to fix the four labels from anynovel's ten parameters, leaving only six. We chose two values for five of the six parameters and three for the sixth. Segmentsize 200, 500, and 2000 were chosen as the values. 0.1 and 0.9 for slacks, and 0.1 and 1.0 for movement Sizes 20 and 50 are considered stable. Novelslack 0.1 and 0.6 and awayt hreshold 0.1 and 0.6 There were 96 tests total in these configurations, from which we selected the most intriguing ones to present here.

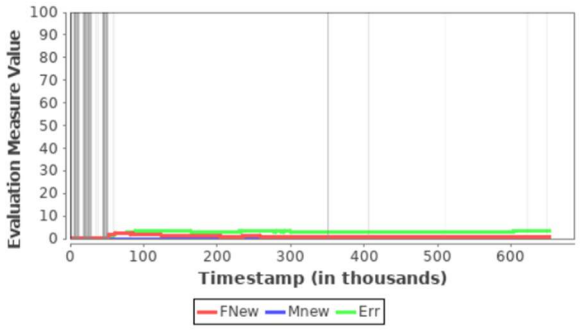


Fig. 4. Binary classification with a label delivery delay of 50,000 examples and total feedback (ECSMiner).

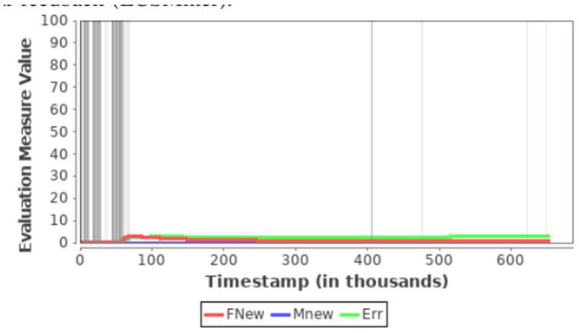


Fig. 5. Binary classification with a label delivery delay of 50,000 examples and partial feedback of 20% (ECSMiner).

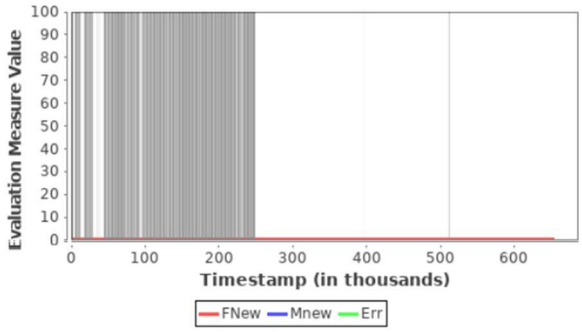


Fig. 6. Binary classification with label delivery delay of 50,000 examples and partial feedback of 1% (ECSMiner).

MINAS and anynovel, two cluster-based partial feedback systems, were tested. Figures 7 and 8 illustrate the outcomes that were chosen.

Fig. 7 shows a Fnew of approximately 5%, We tested MINAS with a variety of additional parameters and found that it maintained the same Fnew rate for the majority of them, implying that this technique is robust in terms of ND (as opposed to anynovel, which had a wide range of measurements), but has to be improved in terms of error rate. The anynovel result, on the other hand, reveals that the Fnew is growing most of the time. This suggests that the approach is having trouble identifying innovations appropriately. Another example with anynovel is where Fnew is less than 2% for the entire execution, but Mnew is greater than 90%, showing a tradeoff between detection accuracy and detection volume.

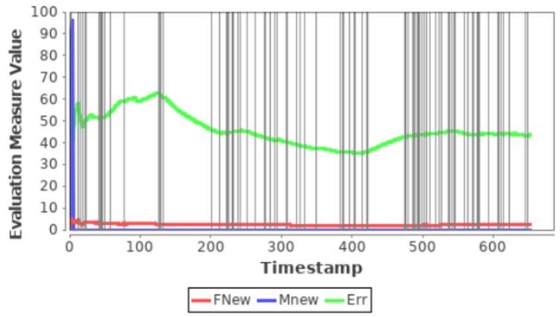


Fig. 7. Binary classification, clustering algorithm with partial feedback of 0.023% (MINAS).

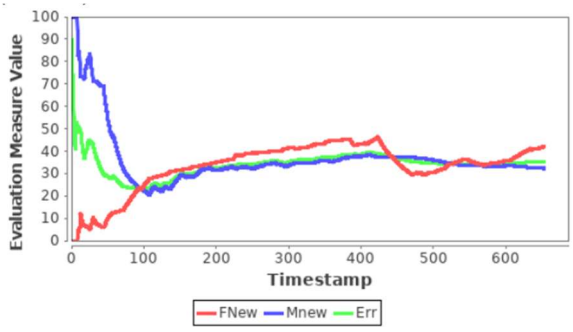


Fig. 8. Binary classification, clustering algorithm with partial feedback of 0,041% (AnyNovel).

In addition, there is a tradeoff between the number of labelled instances necessary and the model type used. A small number of marked instances leads to a higher error and Fnew, as expected, but a large number of labelled examples leads to a lower error and Fnew. Although it requires more labels than the other two clustering algorithms, the ensemble of trees generated superior results. Finally, clustering algorithms can react more quickly to flow changes, independent of the labels provided.

Anynovel isn't designed for intrusion detection, as seen by the huge inaccuracy in Fig. 8, but it does provide relevant data. We choose to fix four labels out of the 10 parameters in anynovel, leaving only six. For five of the six parameters, we chose two values, and for the sixth, we chose three. The choices for Segment Size were 200, 500, and 2000. Slacks should be between 0.1 and 0.9, and movement should be between 0.1 and 1.0. The sizes 20 and 50 are regarded as stable. Awayt hreshold 0.1 and 0.6 and novelslack 0.1 and 0.6 There were a total of 96 tests in these combinations, and we chose the most interesting ones to show here.

Table II summarizes the Fnew, Mnew, Error, and Error and Error and Error and Error and Error and Error and Error time of execution (on the notebook described in Subsection) (V-B) When you see an x in front of a parameter, it signifies that this parameter is not available. Had no effect on the outcome There's an inverse relationship between the two. Fnew, as well as Mnew and Error. 200-0.1-0.1-0.1-0.1-0.1-0.1-0.1-0.1-0.1-0.1-0.1-0.1 However, the smallest achieved, 50-0.6 Fnew, is less than 2%. Mnew is greater than 95%, and error is greater than 60%. On the other hand, Experiment 500-0.9-x-x-x (shown in Fig. 8) demonstrates this. Fnew is at 42%, whereas Mnew and Error are at 32% and 35%, respectively. As a result of the diametrically opposed parameters in these trials, It is simpler to study each parameter separately for better results. Comprehending.

The size of a Slack boundary around the concept clusters has the greatest influence. Higher values for this parameter allow the model to classify examples in an existing concept even if they are further away, ensuring that data dispersion is not

a problem. Bigger Slacks value provides superior results. As a result of the high dispersion in the Kyoto dataset, a model capable of accommodating greater distances produces better results. With a greater value, the parameter Away Threshold has likewise produced better results. This parameter indicates how far a cluster must be from the model to be recognized as a novelty; thus, smaller values will result in a higher frequency of novelties.

Only the Segment Size, which determines the size of each window, had noteworthy results among the other factors. A distinction in performance In general, a 200-foot window is recommended. There were two drawbacks to using examples: I the processing time was longer; and (ii) the processing time was longer, higher than on larger windows; (ii) the detection quality In all cases where only the window changed, was smaller. This study suggests that the approach is highly volatile, implying that it should be avoided. It gets progressively difficult to distinguish between behaviors when there is a lack of evidence. This volatility is corroborated by experiment 200-0.1-0.1-50-0.6, in which the algorithm is able to recognize the more evident innovations but misses the majority of the innovations that are more typical of regular behavior. When additional data is used, the algorithm is capable of recognizing more patterns, lowering Mnew and Error but increasing false positives (Fnew). Based on these findings, it's reasonable to believe that attribute selection strategies could improve this technique's performance on this dataset.

TABLE I

Summary of anynovel results

| SegmentSize-Slacks -Movement-StableSize -AwayThreshold | Fnew | Mnew | Error | Time(s) |
|---|---|---|---|---|
| 2000-0.1-x-x-x | 17.75% | 65.15% | 50.18% | 129.670 |
| 2000-0.9-x-x-x | 25.32% | 52.78% | 44.10% | 143.612 |
| 500-0.1-x-20-x | 18.35% | 66.40% | 51.10% | 152.639 |
| 500-0.1-x-50-x | 20.83% | 60.01% | 47.53% | 163.291 |
| 500-0.9-x-x-x | 41.95% | 32.23% | 35.30% | 337.103 |
| 200-0.1-0.1-50-0.6 | 1.93% | 95.58% | 64.19% | 532.388 |
| 200-0.1-0.1-50-0.1 | 34.42% | 49.42% | 44.69% | 422.720 |
| 200-0.1-1.0-50-0.1 | 11.66% | 76.69% | 56.09% | 228.172 |
| 200-0.9-1.0-20-x | 23.05% | 59.33% | 47.86% | 735.232 |
| 200-0.9-0.1-20-x | 23.60% | 59.25% | 47.97% | 702.040 |

Smaller values of the Slacks parameter result in faster execution times, indicating a compromise between speed and quality of output. Segment Size is another factor that has a significant impact on execution time; the smaller the segment, the slower the execution. Another notable difference with the next paragraph is that anynovel's memory use in solitary testing reached nearly 1 GB.

## B. Performance in Constrained Devices

The goal of this experiment is to see if the evaluated techniques can execute on devices with limited resources. We set up an isolated network using a 3Com Baseline 2824 switch with 24 10/100/1000 ports, a Raspberry Pi, and a router. Laptop with a 2.5ghz Core i5-7200U CPU and DDR4 RAM 8GB of 2133mhz RAM.

The contents of the dataset were reproduced as data streams with controlled parameterized rates by a producer process running on the notebook. The data was stored on the same notebook's Kafka server. A Raspberry Pi 3 card ingested Kafka data and conducted categorization and non-destructive testing. The producer has three parameters (R, S, and I), with R (Base rate) being the initial rate of examples published per second in a Kafka topic. The rate will grow by S units every I seconds until a total of 80MB of data is created. In the table III, for each approach, we provide the Base Rate that resulted.The quickest execution time and the most memory It was devoured. S = 10 was used in every experiment in this table. I equals ten The base rate was raised by a factor of 100 until the When a producer's rate exceeds a certain threshold, the execution time becomes stagnant.

The speed samples provided by customers will begin to collect on the website. Subject, but it has little bearing on customer performance. Ps auxiliary libraries were employed in the memory collecting methodology. And took the necessary procedures to ensure that the only JVM was used the classifier of choice was a process running on the Raspberry Pi. Each individual The settings

that resulted in the technique being setup in the shortest possible time.

TABLE II

Summary of experiments

| Technique | Max Rate | Time (s) | Memory (Mb) |
|---|---|---|---|
| MINAS | 2300 | 60.579 | 65.1 |
| AnyNovel | 5100 | 121.029 | 268.6 |
| ECSMiner | 500 | 764.426 | 146.1 |

MINAS uses the least amount of memory and takes the shortest amount of time to execute since it uses the simplest model and categorization. Even while anynovel has a higher rate than MINAS, its execution time is longer. Because anynovel analyses data in windows, an idle time occurs while data is accumulated for each window, increasing memory use. These findings also point to a trade-off between classification quality and classification speed. Ecsminer has the greatest accuracy metrics in the prior studies, but its execution time is more than 10 times slower than MINAS, despite the rate employed being less than 5 times larger.

### C. Summary of Experiments

In conclusion, these tests revealed information on the methodologies and dataset. Ecsminer is the ideal solution in a situation when processing power and expert feedback are plentiful. In iot applications, however, none of these requirements is easily met. If the classification performance might be improved, the second least memory usage offers a promising option. The MINAS approach has a significant error rate. However, MINAS is suited for iot applications due to the modest and constant Fnew, as well as the shortest execution time and memory usage. Also, by modifying the settings, the error rate may be improved.

Finally, anynovel has a fast execution time, and memory usage in Kafka experiments is lower than in standalone experiments. However, the obtained findings were too disparate, and better parametrization or feature selection may enhance them. On our tests, however, there was no combination that produced an excellent metric in one area while being acceptable in others.

## V. CONCLUSION

An intrusion detection system architecture for iot contexts was proposed in this paper. The design of the building has an impact. With the use of a high-capacity computing infrastructure, such as the public cloud and edge-based distributed resources the network's The classification is done towards the edge. Devices and sensors are being used to minimize processing delay, while The cloud is used to improve and disseminate models. Such that the parts on the periphery are constantly current 'The' The physical schema of the system allows it to be used for a variety of purposes.

The logical approach given is general and adaptable to various ND approaches. Experiments show that the ND approach may be used to detect intrusions in the context of intrusion detection. With regards to the sort of model utilised and the quality of the outcomes, certain trade-offs have been established for the quantity of feedback necessary. It is well recognised that in a real-world setting, the number of labels accessible will be limited, hence it is intriguing that the approaches employ as little as possible to provide effective outcomes. Experiments have demonstrated that a device with limited resources is capable of executing novelty categorization and detection at a pace of 450KB per second.

We want to test this architecture with a broader range of approaches in the future, as well as enhance and/or implement some of the approaches now in use. We also mention the idea of a comparison examination of performance utilising a bigger quantity of data as a possible future project.

## REFERENCES

[1] "Gartner says 8.4 billion connected "things" will be in use in 2017, up 31 percent from 2016," 2017. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billion-connected-things-will-be-in-use-in-2017-up-31-percent-from-2016.

[2] "Gartner says worldwide iot security spending will reach $1.5 billion in 2018," Mar 2018. [Online]. Available: https://www.gartner.com/newsroom/id/3869181

[3] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," IEEE Communications Surveys & Tutorials, vol. 18, no. 2, pp. 1153–1176, 2016.

[4] R. Mitchell and I.-R. Chen, "A survey of intrusion detection techniques for cyber-physical systems," ACM Computing Surveys (CSUR), vol. 46, no. 4, p. 55, 2014.

[5] A. V. Dastjerdi and R. Buyya, "Fog computing: Helping the internet of things realize its potential," Computer, vol. 49, no. 8, pp. 112–116, 2016.

[6] P. Kasinathan, C. Pastrone, M. A. Spirito, and M. Vinkovits, "Denial of-service detection in 6lowpan based internet of things," in IEEE 9th Intl. Conf. on Wireless and Mobile Computing, Networking and Communications (WiMob), Oct 2013, pp. 600–607.

[7] M. Sheikhan and H. Bostani, "A hybrid intrusion detection architecture for internet of things," in 8th Intl. Symp. on Telecommunications (IST), Sept 2016, pp. 601–606.

[8] S. Raza, L. Wallgren, and T. Voigt, "Svelte: Real-time intrusion detection in the internet of things," Ad Hoc Networks, vol. 11, no. 8, pp. 2661 – 2674, 2013.

[9] G. Furquim, G. P. R. Filho, R. Jalali, G. Pessin, R. W. Pazzi, and J. Ueyama, "How to improve fault tolerance in disaster predictions: A case study about flash floods using iot, ml and real data," Sensors, vol. 18, no. 3, 2018.

[10] D. Midi, A. Rullo, A. Mudgerikar, and E. Bertino, "Kalis - a system for knowledge-driven adaptable intrusion detection for the internet of things," in IEEE Intl. Conf. on Distributed Computing Systems (ICDCS), June 2017, pp. 656–666.

[11] J. Lloret, J. Tomas, A. Canovas, and L. Parra, "An integrated iot archi tecture for smart metering," IEEE Communications Magazine, vol. 54, no. 12, pp. 50–57, 2016.

[12] D. E. Difallah, P. Cudré-Mauroux, and S. A. McKenna, "Scalable anomaly detection for smart city infrastructure networks," IEEE Internet Computing, vol. 17, no. 6, pp. 39–47, Nov 2013.

[13] M. A. Faisal, Z. Aung, J. R. Williams, and A. Sanchez, "Data-stream based intrusion detection system for advanced metering infrastructure in smart grid: A feasibility study," IEEE Systems Journal, vol. 9, no. 1, pp. 31–44, March 2015.

[14] J. Gama, Knowledge Discovery from Data Streams. CRC Press Chapman Hall, 2010.

[15] P. Perner, "Concepts for novelty detection and handling based on a case-based reasoning process scheme," in Advances in Data Mining. Theoretical Aspects and Applications. Springer, 2007, pp. 21–33