International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-1S3, December 2019

Smart Helmet: Thresh-Learner–Online Machine Learning on Data Streams

Dhruv Garg, Aditya Chitlangia, Vetrivelan P

Today, with an enormous generation and availability of time series data and streaming data, there is an increasing need for an automatic analyzing architecture to get fast interpretations and results. One of the significant potentiality of streaming analytics is to train and model each stream with unsupervised Machine Learning (ML) algorithms to detect anomalous behaviors, fuzzy patterns, and accidents in real-time. If executed reliably, each anomaly detection can be highly valuable for the application. In this paper, we propose a dynamic threshold setting system denoted as Thresh-Learner, mainly for the Internet of Things (IoT) applications that require anomaly detection. The proposed model enables a wide range of real-life applications where there is a necessity to set up a dynamic threshold over the streaming data to avoid anomalies, accidents or sending alerts to distant monitoring stations. We took the major problem of anomalies and accidents in coal mines due to coal fires and explosions. This results in loss of life due to the lack of automated alarming systems. We propose Thresh-Learner, a general purpose implementation for setting dynamic thresholds. We illustrate it through the Smart Helmet for coal mine workers which seamlessly integrates monitoring, analyzing and dynamic thresholds using IoT and analysis on the cloud.

***Keywords*: Anomaly Detection, Internet of Things, Online Machine Learning, Stream Processing**

I. INTRODUCTION

With sensors pervasive our everyday lives [13], we are observing an exponential increase in the availability of time series data and streaming data from sensors [14] along with data from social networks [11], and smart cities [15]. Largely complemented by the advancement in Internet of Things (IoT) and real-time sources of data, we now have a wide range of applications with the time changing data produced by these sensors. Effectively analyzing and finding patterns in these streams can provide invaluable insights and results for many applications and use cases.

The detection of accidents and anomalies in real-time streaming data has become a significant need in the industrial field. Use cases such as environmental condition detection, fault detection, and monitoring can be found

beneficial in a wide range of industries such as IT, security, medical, construction industries, and mining industries. For anomaly detection setting manual thresholds is like drawing a line in the sand that doesn't always correlate to real-life needs and scenarios. In spite of manually setting the threshold to best at current knowledge, these static parameters don't take into account the future needs and scenarios. Streaming technology along with dynamic thresholds is a gold standard to detect anomalies providing real-time perishable and actionable insights. Instead of batch-oriented processing [16], [17], [18], the use of stream-oriented analytics [19], [20] has become highly beneficial because, in contrast to batch processing, the full dataset is not available. The system observes each data record sequentially as it arrives and all processing and learning must be carried out in an online fashion.

In this paper, we ask the question: how can we combine powerful online machine learning algorithms with the streaming processing for dynamic thresholds to prevent accidents and blasts in Coal mines? We propose the stream processing based Thresh-Learner that decouples expertise of online Machine Learning for anomaly detection and accident prevention.

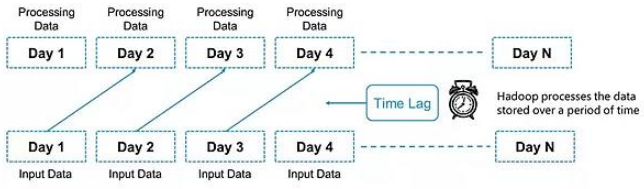

Fig. 1. Computation illustration in a batch processing scenario

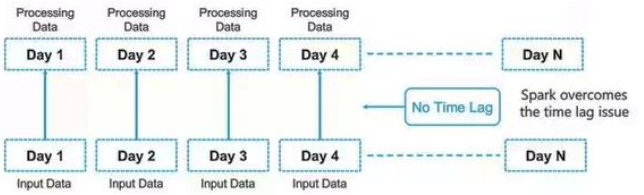

Fig. 2. Computation illustration in a stream processing scenario

II. CHALLENGES AND GOALS

Machine Learning models usually train a model using batch training. In the case of streaming data (e.g. from sensors) being available, models need to be dynamically adapted. That means the new data should be taken into account for training the model while the old data becomes irrelevant. This could be visualized as a sliding window over the incoming data streams.

Manuscript published on 30 December 2019.
***** Correspondence Author (s)
Dhruv Garg*, Computer Science, Vellore Institute of Technology, Chennai, India. Email: dhruvshekhar.garg2016@vitstudent.ac.in
Aditya Chitlangia*, Computer Science, Vellore Institute of Technology, Chennai, India. Email: Aditya.chitlangia@vitstudent.ac.in
Vetrivelan P, Electronics and Communications, Vellore Institute of Technology, Chennai, India. Email: vetrivelan.p@vit.ac.in

Machine Learning on sliding windows is also called non-stationary machine learning [13]. It keeps the model updated as the underlying streaming data changes. Thus, in streaming learning or non-stationary machine learning, the model will not be re-built from scratch for new data, but incrementally updated with new data, while old becomes irrelevant. Another problem with ML in streaming data is that the data from different streams leads to independent models. This might not be appropriate for many applications. For example, a model suitable for one machine might not be suitable for another independent machine. Thus, the challenge is to determine which models have been built based on which event streams. When these questions have been solved, the output must be driven at low latency to the end device. Given that IoT applications mostly use wireless communication, the technology must be chosen wisely, so as to satisfy the application's constraints.

The paper is organized as follows. Section II describes the challenges and goals of an IoT based threshold setting system using online machine learning. Section III describes the design of Thresh-Learner- the programming model, supporting hardware architecture and the workflow. Section IV presents an IoT case study for dynamically setting a gas alert threshold for coal mine workers. Section V evaluates the case study implementation and Section VI concludes the paper.

III. THRESH-LEARNER

In this section, we give an overview of the Thresh-Learner architecture, followed by a description of the implementation for online machine learning and situation inference models.

A. Programming model

The architecture for the hardware supporting Thresh-Learner (which is implemented in the cloud) is shown in Figure 3. To parallelize the Machine Learning computation, the split-process-merge architecture of the older event-based systems [10], [11], [12]. The splitter receives time-series data via the input stream and forwards them for processing. The processing involves implementing the ML algorithm and forwarding it to the output queue. Due to the independent processing of events, the architecture will be able to support both scale-up operations and scale-out operations, as required by the application.

Each processing step occurs in three phases: shaping, training, and inference. In the shaping phase, we perform processing to transform the input into appropriate data values v to be supplied to the online machine learning model. In the training phase, the trainer module incrementally updates the parameters of the model M (e.g. a linear regression model) according to the user-specified model update function. In the last phase- inference phase, the updated model and processed event serve as an input for predicting the output. There are two key points of the Thresh-Learner. First, the application user can provide any input format, but they must apply the requisite transformation(s) to supply values to the ML model. Second, the Thresh-Learner does not mandate that the data for training and inference be different. Based on the application use case, the same data could be used for training as well as predicting the output. Although it is a common practice in Machine Learning to separate data that is used for training and testing, we provide the flexibility to the user. This allows the use-cases that might require the same event to be used to incorporate change in pattern in the ML model and initiating an inference event using the predictor. However, Thresh-Learner assumes that training will happen before inference. In the next subsection, we describe the hardware implementation which serves as the sensing circuit. It provides the raw sensed data to be supplied to the programming model described above.

B. Hardware architecture

There are numerous use-cases where Thresh-Learner could be applicable. Some examples include alerting systems where a concerned authority could be notified if the sensed value exceeds a dynamically set threshold. For example, Alerts could be sent out to the local civic and administrative bodies if an area has a high concentration of water impurity (water pollution), highly harmful gases (air pollution) or constant sound emissions over long periods of time (noise pollution). This model can also be applied to certain elements of Smart City projects. For example, sensors attached below street lights could detect light intensity and alert the civic body if the intensity is too low, i.e. for preventive maintenance. Thresh-Learner can also be applied to sense environments for natural calamities, like forest fires (record temperature and gases in forest areas) and floods (keep monitoring water level). An advance alert in cases of natural calamities can expedite evacuation and save many lives. Thus, Thresh-Learner with hardware implementation for IoT could find great avenues for application. For this to be possible, the architecture has been made such that it behaves like a plug-and-play model, and fits many applications.

The skeleton of the circuit has 4 key elements that act as the core components for any application that employs the Thresh-learner programming model. They are the Arduino Uno board, sensors required by the application, ESP8266 Wi-Fi module, and the ThingSpeak cloud service. The following paragraph describes the core components and other associated tools to implement an application.

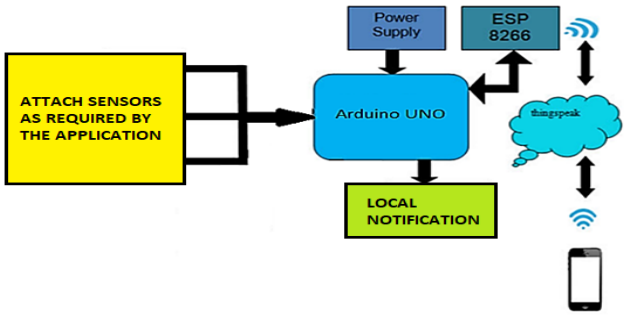

Fig. 3. Architecture diagram for the general application which implements Thresh-Learner

Arduino Uno

At the center of our hardware implementation is the Arduino Uno. The Arduino board is an open-source microcontroller board based on Microchip ATmega328P. Based on the application requirements, one or more sensors can be connected to the board's digital or analog pins.

Arduino programming language is a simplified form of C/C++ programming language which uses basic programming structures, variables and functions. The Arduino and connected sensors can be powered up via the type B USB cable or by the external 9-volt battery, though it accepts voltages between 7 and 20 volts. A major requirement of Internet of Things applications is the portability of the sensing circuit. With the flexibility of power input, we are able to make the sensing circuit portable using batteries or power-banks as power sources. With adequate power supply for the sensors attached to the board, sufficient up-time can be provided on a single charge. This is based on the assumption of the optimized power consumption of sensors designed for Internet of Things applications.

Sensors

The sensors required vary from application to application. Some of the use-cases where Thresh-Learner would be applicable have been introduced earlier. Keeping re-usability in mind, the hardware architecture has been designed such that it is applicable to multiple use cases. We can connect sensors such as Heart rate pulse sensor, Ultrasonic Distance sensor, Temperature-humidity sensor, Alcohol Gas Detector Sensor, Color Sensor, CH4 Gas Sensor, Flame sensor, PIR sensor, Line tracking sensor, Optical fingerprint sensor, Knock sensor, the Sound sensor to the Arduino Uno board. Given that sensing circuits are mostly power constrained, it is advisable to use good quality sensors. Standard sensors designed for Internet of Things applications are designed for optimal power consumption. This allows the use of batteries and power-banks, which can keep the circuit powered up for a sufficiently long time.

Wi-Fi module (ESP 8266)

The Wi-Fi module is used to transfer the data from the working environment to the ThingSpeak cloud. For this purpose, the Wi-Fi communication network provides a wider area range and better connectivity when compared to ZigBee. Although ZigBee has low power consumption and is a mesh networking standard, i.e. each node in the network is connected to each other and it doesn't have to solely rely on the router and the endpoint, it has interoperability problems. This means that ZigBee profiles can interfere with one another. The range of ZigBee is restricted to wireless personal area networks (WPAN) of 10 to 30 meters, and its data transfer speed is lower than Wi-Fi too. For these reasons- a wider range of 30-100m and significantly faster data speeds, Wi-Fi is preferred.

Protocol stack

Table I describes the entire protocol stack for the hardware implementation, with respect to the 5 layers of IoT deployment.

Table I. Protocol stack for IoT layers

IOT LAYER	PROTOCOL
Physical layer (sensor to board)	**I2C™ Communication Protocol** I2C is a serial communication protocol, so data is transferred bit by bit along a single wire (the SDA line).
Link layer	**WIFI 802.11** Data Link Layer consists of MAC and LLC sublayer. The Wifi system consist of Wifi client, access point and LAN.
Network layer	**IPv4**
Transport layer	**TCP** TCP makes sure that all packets are received, that the packets are in order, and that corrupted packets are re-sent.
Application layer (ESP8266 to Cloud)	**HTTP** Since the application uses the **Exclusive pair communication model**, the application layer protocol used is HTTP. The system on the helmet communicates with the cloud based incident response and notification system via a RESTful architecture over HTTP

ThingSpeak

ThingSpeak is an open source IoT application and API to store and retrieve data from things using the HTTP protocol over the Internet. For each application, we can create a dedicated channel. Each channel can be either public or private, as desired by the user. Also, the channels are given Read and Write API keys on ThingSpeak. To push sensor data to the cloud, we use the Write API key. To retrieve data from the cloud, we use the Read API key. Based on the count of sensors used, we can also add a number of fields to that channel. Data sensed from the environment using the sensors, was sent to the cloud by the running AT commands (also called attention commands) on the ESP8266 Wi-Fi module. The Wi-Fi module first establishes a TCP connection with thingspeak.com, and then it sends the data to the cloud using the channel's Write API key supplied in the Arduino code.

Webhooks Applet and IFTTT

IFTTT (If This Then That) helps us to connect all of our different applications and devices. There are thousands of Applets to explore and they can be easily connected to the application. For example, one can use the Webhooks Applet from IFTTT. A WebHook is an HTTP call-back: an HTTP POST that occurs when certain things happen. It provides various web applets that can be invoked on the occurrence of an event by using the URL. Based on the application use-case, one can determine the best utility IFTTT Applet. In our application, we used the Webhooks Mail Applet to send an alert email when a certain condition occurred.

MATLAB Analysis and Time Control

ThingSpeak allows users to analyse and visualize uploaded data through the MATLAB Analysis app. The analysis can be done using MATLAB without requiring the purchase of a license from Mathworks. Moreover, ThingSpeak offers TimeControl which enables us to run the MATLAB Analysis code at a particular frequency (e.g. once every 10 minutes).

C. Workflow

This subsection talks about the workflow of an application that is used for real-time monitoring and utilizing a dynamically set threshold using Thresh-Learner.

The implementation begins at the sensing-circuit consisting of the Arduino board, connected sensors, and the ESP8266 Wi-Fi module. Based on the user-defined rate of sensing, the sensors take their analog/digital readings from the environment and send the data to the Arduino board. Since the data needs to be transmitted to the cloud, basic pre-processing is done to get the data in the required format. Next, the Arduino board uses AT commands or attention commands to communicate with the ESP8266 Wi-Fi module, over a given pre-defined baud rate. The Wi-Fi module establishes a TCP connection with thingspeak.com URL and uses the POST method to send the data. While sending the data, it uses the Write API key of the application channel.

Once the data is sent to the cloud, the Thresh-Learner programming model begins its execution. The code for the Thresh-Learner model is written in the MATLAB Analysis app of ThingSpeak. The model changes incrementally in the training phase. Finally, the current event and updated model serve to predict the output. Since Thresh-Learner evaluates to a dynamic threshold value, there is likely to be an alerting system. For the alerts, numerous IFTTT Applets can be utilized, based on the application's requirement. The Time Control app in ThingSpeak will keep executing the MATLAB Analysis code at fixed intervals and the model will be updated incrementally.

IV. CASE-STUDY: SMART HELMET FOR COAL MINE WORKERS

A. Problem description

A major resource for power generation has been fossil fuel, namely coal. Underground coal mining is hazardous and thousands of miners lose their lives due to accidents like coal mine fires, gas leakages, suffocation, and explosions. In a tragic accident in November 2009, 104 miners were killed in Heilongjiang, China. There have been numerous such incidents reported in around the world. Coal catches fire when exposed to air for long, and the areas exposed near the surface catch fire very often. The resulting impact is extensive. It leads to economic loss, land subsidence, water pollution, air quality deterioration, and most importantly, loss of life. Working environment hazards for the coal mine workers include suffocation, gas poisoning, and gas explosion. Hence air quality and hazardous event detection is a very important factor in the coal mining industry. In order to achieve those safety measures, the proposed system based on Thresh-Learner provides a wireless sensor network for monitoring the situation of the working environment from the base station, located at a distance. It provides real-time monitoring of combustible/harmful gases like CO, CH4, and LPG and also the temperature at which the workers are working.

Safety for mining workers is one of the main aspects of coal mining Industry. The system provides real-time monitoring of mines from the monitoring station. This aim for a monitoring and protection device for tracking and protection of underground employees primarily based on Wi-Fi wireless network.

B. Formulating the problem in Thresh-Learner

The helmet section of the system comprises of the following IoT sensors:

Air quality sensor (MQ 2)

Air Quality Sensor is used to sense and detect the emission of combustible/hazardous gases found in the coal mine areas like LPG, methane and carbon monoxide. It causes several health hazards to the workers of the industries. At the worker end, a static threshold value for gas concentration in this system is 350 ppm (parts per million).

Temperature and humidity sensor (LM35)

The temperature and humidity sensor is used to sense the temperature and humidity of the working environment. Since humidity readings are not relevant for this use case, there is no threshold set for humidity. At the worker end, a static threshold value for temperature used in this system is 35 degrees Celsius.

Buzzer

Buzzer which is connected to the Arduino emits a beep sound of set frequencies whenever the sensor readings cross the threshold conditions. Since we have two thresholds, two different alerts from the buzzer are issued.

- For Gas > 350 ppm: One 1000 Hz beep for 2 seconds
- For temperature > 35 Celsius: Two 500 Hz beeps with a gap of 2 seconds

There have been some assumptions made in the Smart Helmet prototype: It is assumed that sufficient supporting infrastructure (routers, repeaters) would be deployed in the coal mine to capture data packets sent over Wi-Fi. It is also assumed that the circuit made using standard sensors and components is intrinsically safe and no sparking would occur. The architecture diagram and the prototype of the Smart Helmet for coal mine workers are shown below:

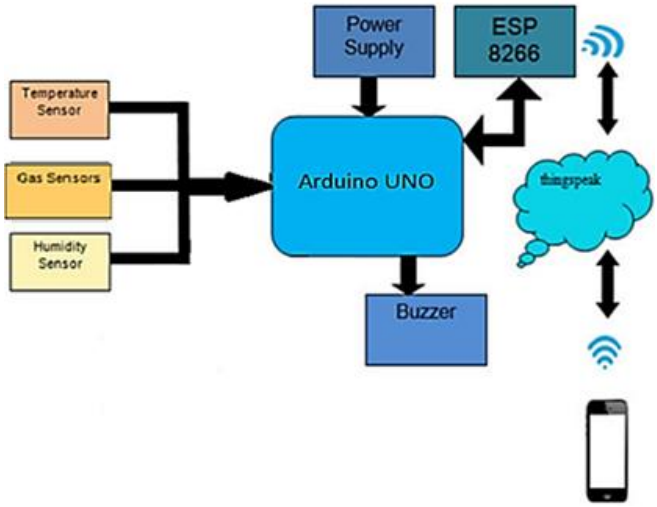

Fig. 4. Architecture diagram for the Smart Helmet for coal mine workers incorporating Thresh-Learner

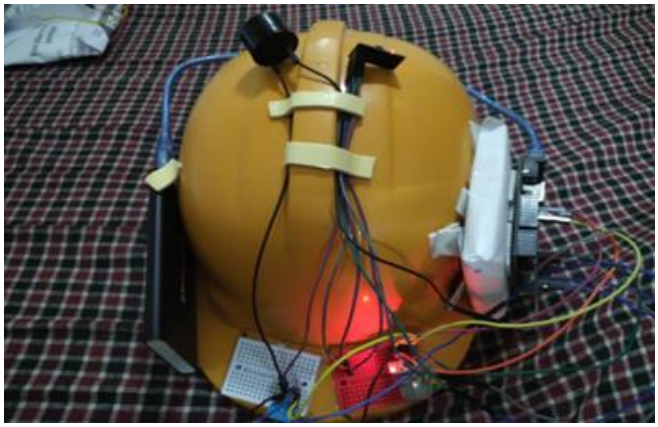

Fig. 5. Smart helmet prototype

. **Thresh-Learner formulation for the given scenario**

This scenario fits nicely into Thresh-Learner: for each smart helmet, an independent ML model is subject to incremental training and inference steps. Thus each channel in the Thresh-Learner on ThingSpeak is responsible for all observations of a single sensor enabling Thresh-Learner to monitor multiple workers in parallel.

Incremental Linear Regression

In this application of setting a dynamic threshold for coal mine workers, ordinary linear regression is not applicable. This is because it is a batch learning technique which creates a predictor by training on the entire dataset at once. It is not appropriate in this scenario as the sensed data from the environment (where the coal mine worker works) is being provided sequentially. Thus, a hard threshold set after training on old data will not be efficient. Online machine learning implemented through Thresh-Learner is an improvisation method of learning step, where the model is updated as per the new data arrived, rather than re-training the whole model. It is basically a step-wise learning method.

First, the sensors will sense the concentration of combustible gases in the environment. Training data is last n days of sensor data, obtained from the Gas concentration and Temperature channel using its Read API Key. The attribute n is set by the user. This means that the most recent n days will be used as training data. After getting training data from the Gas concentration and Temperature channel, the Thresh-Learner model is trained. It takes into account the minimum concentration and maximum concentration of gases to arrive at a new threshold, which will change as the newer data arrives.

C. Smart helmet: Workflow

The system provides real-time monitoring of mines from the monitoring station. There is a transmitter unit placed on the helmet of each worker and the monitoring station can view the sensor data from the cloud. The transmitter unit consists of an air quality sensor, temperature sensor, buzzer, and the Wi-Fi module- all connected to the Arduino Uno board. The air quality sensor monitors the level of harmful gases like LPG, methane and carbon monoxide. Wi-Fi wireless communication is used for data transmission from the working environment to the cloud. Wi-Fi communication network provides a higher range and better connectivity when compared to Zigbee. The manager at the monitoring station can view the sensor data via the ThingSpeak platform. The MQ2 Air Quality Sensor is used to monitor the level of combustible and hazardous gases like methane and LPG. The LM35 Temperature and Humidity sensor are used for monitoring the temperature. Both of them are semiconductor type sensor.

The sensors sense the concentration of gases and transmit real-time data to the microcontroller in the Arduino. The controller receives the data, process it and transmit it to the monitoring station through the AT commands of the ESP8266 Wi-Fi module. In this application, we created a channel called Gas concentration and Temperature and the channel identified by its write API key. The sensing of the environment is set to occur every 30 seconds and it takes approximately 10 seconds for the data to be uploaded to the channel after that. The gas concentration and temperature data are uploaded to the two fields (Gas concentration and temperature) of the Gas concentration and Temperature channel. Figures 8 and 9 show the readings of the data uploaded on ThingSpeak Channel. In case the concentration of gases is too high, or the temperature increases rapidly, two alerts need to be raised. First, at the worker's end (static threshold is used)- the buzzer is activated. Second, at the monitoring station (dynamic threshold using online machine learning), to rescue the workers based on the severity.

The data from the gathered from sensors is transmitted to the ThingSpeak channel app using Wi-Fi. At a frequency of 20 seconds, new sensor data is pushed to the cloud, and this can be viewed at the corresponding ThingSpeak channel in the monitoring station. All real-time data are received from helmet to the monitoring station can be visualized on ThingSpeak in the form of graphical charts. At the monitoring side, online machine learning is used for abnormal concentration detection. If an abnormal condition is detected, a Webhook applet is activated and an email is sent to the manager of the monitoring station.

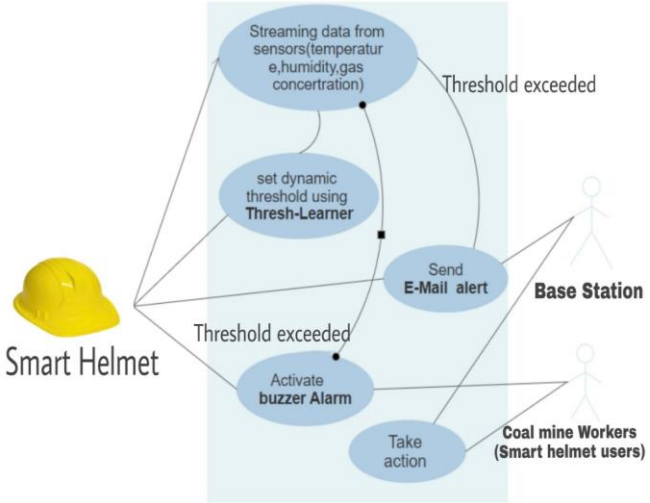

Fig. 6. Use-case diagram of the workflow of Smart Helmet with Thresh-Learner

Sending E-Mail alerts to the base station

The Thresh-Learner model is implemented in MATLAB Analysis of ThingSpeak. We have set a TimeControl to run the MATLAB Analysis code every 5 minutes.

We want to issue an emergency alert every time the combustible gas concentration crosses 110% of previous n days. If new gas concentration reading is greater than 110% of previous n days, a Webhook Email Applet HighGasConcentration created using IFTTT is executed. This Applet sends an alert email to the base station as soon as it is triggered.

MODES OF OPERATION

The Smart Helmet designed above is programmed to perform the following two modes of operation described below.

Fig. 7. IFTTT Webhooks Mail Applet

MODE 1: Local alert mode

In this mode the user who wears the helmet is alerted as to when the helmet finds any abnormalities in the surrounding gas, temperature or humidity the buzzer starts beeping thereby alerting the user and the people around him.

MODE 2: Management alert mode

In this mode, in case any abnormalities are detected, the software interface automatically alerts the management regarding the emergency and so helps to mitigate the loss of life. Figure 7 shows the IFTTT Webhooks mail applet and Figure 10 shows the alert email received.

V. EVALUATION

The Smart Helmet prototype worked efficiently when connected to a power bank, and provided correct readings over long durations of time. Thresh-Learner programming model for machine learning on streaming data was also prompt and provided local (buzzer) and management (e-mail) alerts when the combustible gas concentration exceeded 110% of normal. There were no false positives detected and the latency incurred was entirely dependent on the Wi-Fi connectivity. This reminds us to be cautious of the hard constraints of poor wireless connectivity in the underground mines. As stated in the previous section on Thresh-Learner, the model is scalable and is capable of handling many parallel nodes at a given time. Some noted challenges and limitations of the current model are: security- a significant challenge for IoT devices and good Wi-Fi infrastructure inside the coal mine must be established for successful implementation.

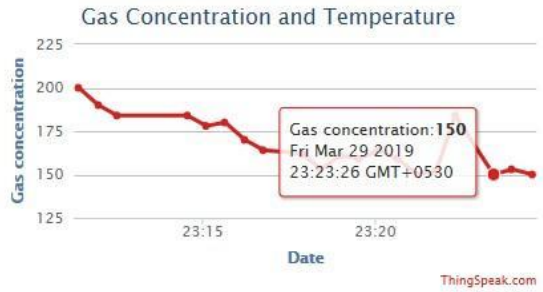

Fig. 8. Gas concentration data uploaded on ThingSpeak

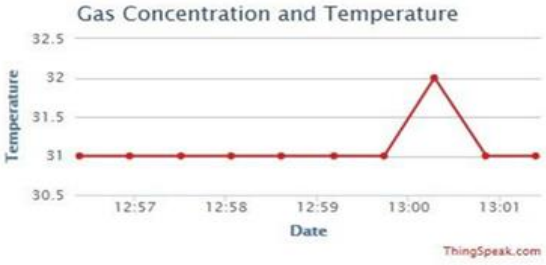

Fig. 9. Temperature data uploaded on ThingSpeak

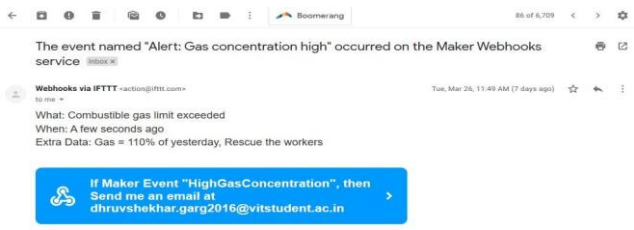

Fig. 10. Mail received when combustible gas concentration exceeds 110% of past 10 day's concentration.

VI. CONCLUSION AND FUTURE WORK

Because Thresh-Learner is a distributed system tailored to scalable event detection using Online Machine Learning on streaming data. Although the model is a general purpose, Thresh-Learner is especially suited for applications pertaining to dynamic threshold setting and anomaly detection. It can also handle multiple end devices, i.e. it is applicable to data parallel problems. For these reasons, Thresh-Learner and hardware implementation is a powerful functionality while scaling due to incremental learning and independent models. A prototype of an intelligent mine safety helmet is developed that is competent to recognize various kinds of unsafe situations in the mining industries such as carbon monoxide gas accumulation and various inflammable gases. The Smart Helmet for coal miners was successfully implemented, and emergency notifications were being sent to the monitoring station when gas concentration exceeded 110% of the past few days' concentration. A couple of things can be enhanced in the hardware system for the future: implementation of the circuit using a single Printed Circuit Board (PCB) as per the application specific requirements; testing in real-world conditions is required; the performance metrics in implementation (such as latency) can be improved.

REFERENCES

1. T. Cucinotta, A. Mancina, G. F. Anastasi et al., "A real-time service-oriented architecture for industrial automation," IEEE Transactions on Industrial Informatics, vol. 5, no. 3, pp. 267
2. R. S. Nutter, ―Hazard evaluation methodology for computer-controlled mine monitoring/control systems, IEEE Trans. on Industry Applications, vol. IA-19, no. 3, pp. 445-449, May/June 1983
3. D. Kock and J. W. Oberholzer, "The development and application of electronic technology to increase health, safety, and productivity in the South African coal mining industry," IEEE Trans. on Industry Applications, vol. 33, no. 1, pp. 100-105, Jan/Feb. 1997.
4. Cheng Qiang, Sun Ji-ping, Zhang Zhe, Zhang Fan "ZigBee Based Intelligent Helmet for Coal Miners" World Congress on Computer Science and Information Engineering 2009.
5. Shirish Gaidhane, Mahendra Dhame and Prof. Rizwana Qureshi "Smart Helmet for Coal Miners using ZigBee Technology" Imperial Journal of Interdisciplinary Research (IJIR) Vol-2, Issue-6, 2016 ISSN: 2454-1362
6. Kumar and G. P. Hancke, "Energy efficient environment monitoring system based on the IEEE 802.15.4 standard for low cost requirements", IEEE Sensors Journal, vol. 14, no. 8, pp. 2557-2566, Aug. 2014.
7. H. Hongjiang and W. Shuangyou, "The application of ARM and ZigBee technology wireless networks in monitoring mine safety system," IEEE International Colloquium on Computing, Communication, Control, and Management (ISECS 2008), 3-4 Aug. 2008, Guangzhou, pp. 430–433, 2008.
8. "Head and neck injury criteria a consensus workshop" Research information and publications center. University of Michigan transportation research institute.
9. M. Li and Y. Liu, Underground coal mine monitoring with wireless sensor network, available at: https: //www.ntu.edu.sg/home/limo/papers/TOSN-official-prints.pdf
10. Ruben Mayer, Boris Koldehofe, and Kurt Rothermel. 2015. Predictable Low Latency Event Detection with Parallel Complex Event Processing. Internet of Things Journal, IEEE 2,4(Aug 2015).
11. Ruben Mayer, Christian Mayer, Muhammad Adnan Tariq, and Kurt Rothermel. 2016. Graph CEP: Real-time Data Analytics Using Parallel Complex Event and Graph Processing. In Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems (DEBS '16). ACM, New York, NY, USA.
12. Ruben Mayer, Muhammad Adnan Tariq, and Kurt Rothermel. 2017. Minimizing Communication Overhead in Window-Based Parallel Complex Event Processing. In Proceedings of the 11th ACM International Conference on Distributed and Event based Systems (DEBS '17).ACM,New York, NY, USA, 12.
13. Christian Mayer, Ruben Mayer, and Majd Abdo. 2017. StreamLearner: Distributed Incremental Machine Learning on Event Streams: Grand Challenge. In Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems (DEBS '17), 298-303.
14. Narayanan C Krishnan and Diane J Cook. 2014.Activity recognition on streaming sensor data. Pervasive and mobile computing 10 (2014), 138–154.
15. Michael Batty. 2013. Big data, smart cities and city planning. Dialogues in Human Geography 3, 3 (2013), 274–279.
16. Jerey Dean and Sanjay Ghemawat. 2008. MapReduce: Simplified Data Processing on Large Clusters. Commun. ACM 51, 1 (Jan. 2008), 107–113.
17. Grzegorz Malewicz, Matthew H Austern, Aart JC Bik, James C Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. 2010. Pregel: a system for large scale graph processing. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. ACM, 135–146.
18. Christian Mayer, Muhammad Adnan Tariq, Chen Li, and Kurt Rothermel. 2016. GrapH: Heterogeneity-Aware Graph Computation with Adaptive Partitioning. In Proc. of IEEE ICDCS.
19. Gianpaolo Cugola and Alessandro Margara. 2012. Processing ows of information: From data stream to complex event processing. ACM Computing Surveys (CSUR) 44, 3 (2012), 15.
20. Ahmad, S., Lavin, A., Purdy, S. and Agha, Z. Unsupervised real-time anomaly detection for streaming data. (2019)

AUTHORS PROFILE

Dhruv Garg, is a final year Bachelor of Technology undergraduate from the Vellore Institute of Technology (VIT), Chennai. He has gained research experience through research projects and assistantships in college. As his capstone project, he is involved in a semester-long research internship at Ericsson Research, Bangalore, and the project is in collaboration with a professor from the Indian Institute of Science (IISc), Bangalore. His key areas of interest for research are Distributed Computing and Machine Learning. In these two areas, he has made two paper presentations at two international conferences, both of which are being published in Scopus indexed journals. In Distributed Computing, his work is around data processing, data management, and query processing at the fog and the edge. This also encompasses the interesting use-cases in the Internet of Things where query processing and decision making must occur in a few seconds, if not in a sub-second interval. In the domain of Machine Learning, he works on models learning from massive data and thus require a distributed setup for learning. A publication of his in this area talks about training models on multiple compute nodes. Not just that, it talks about training models over private data and yet keeping the model oblivious to the given data to meet the user's privacy in mind.

Aditya Chitlangia, is a final year Bachelor of Technology undergraduate from the Vellore Institute of Technology (VIT), Chennai. He has gained research experience through research projects in college and research internship at Indian Space Research Organization (I.S.R.O). As his capstone project, he is involved in a semester-long internship at Cerner Corporation, Bangalore. His key areas of interest for research are Machine Learning and Image processing. In these areas, he has made two paper presentations at two international conferences, both of which are being published in Scopus indexed journals. In Image Processing, his work is around extracting handwritten text from images and further training a model to gain information of the writer from its handwriting. He has done research internship in I.S.R.O in the domain of Machine Learning in healthcare sector, where he came up with a time-sensitive automated framework to analyze disease progression from initial diagnosis. He used multivariate logistic regression to identify the correct intervention points at which patients need to be approached in order to avoid emergency hospital visits. Further, developed Convolutional Neural Networks to predict whether patients with certain chronic illness will need surgery in near future**.**

Vetrivelan Pandu, PhD, is an Associate Professor and Head of the Department for the Bachelor of Technology (Electronics and Communication Engineering) in School of Electronics Engineering at Vellore Institute of Technology (VIT), Chennai, India. He has completed his Bachelor of Engineering from the University of Madras,
Chennai. Further, he completed his Master of Engineering in Embedded System Technologies and Doctor of Philosophy in Information and Communication Engineering from Anna University, Chennai. He has 14 years of teaching experience altogether in CSE and ECE departments in both private Engineering Colleges in Chennai (affiliated to Anna University, Chennai) and Private Engineering University in Chennai respectively. He has authored 3 book chapters and one proceeding in lecture notes published by reputed Springer publisher, and has authored 25 Scopus indexed Journal papers published in reputed international conferences. He has served as member in Board of Studies, doctoral committee, doctoral thesis Examiner, doctoral oral Examiner in both private and government Universities. He also serves as a reviewer for reputed International Journals and International Conferences. His research interests include Wireless Networks, Adhoc and Sensor Networks, VANETs, Embedded Systems and Internet of Things (IoT).