Hyper localization of leaks in piping and cabling systems using reinforcement learning

Pranav Parnerkar
Siemens Advanta
Pune, India
pranav.parnerkar@siemens.com

Anindya Chatterjee
Siemens Advanta
Bangalore, India
anindya.chatterjee@siemens.com

Indrajit Kar
Siemens Advanta
Bangalore, India
indrajit.kar@siemens.com

***Abstract*: Leaks have undoubtedly been one of the biggest problems plaguing piping and cabling systems across industries like electricity and power, building and smart cities, oil and gas, etc. Addressing these leaks in time becomes paramount as failure leads to a complete standstill of the transportation chain. Most AI based leak detection systems have failed to reach the deployment state as these systems are prone to output false positives. It is imperative to observe that these leaks don't occur every day or in other words they are rare events. But when they do occur, these leaks more often than not go unnoticed. Due to the insufficient number of identified leak points, it becomes difficult to build an AI based model for the same. In an attempt to aid/replace rule-based and physics-based leak detection systems, this paper proposes a novel AI based leak detection solution using reinforcement learning which not only reduces false positives but also extends itself to multi armed bandit-based leak localization. By using this methodology, we model the latent behavior of any piping or cabling systems and provide a Q-learning based shortest path recommendation in order to help the maintenance team reach the leak node in a short amount of time**.

Keywords: Leak detection, reinforcement learning, multi arm bandit, Q-learning, anomaly detection.

1. Introduction:

In current times, pipeline networks span across the length and breadth of every country for transporting various utilities from its source to numerous destinations. Primarily, they are used to transport fluids and gases. For enhancing the durability of these pipes, they are built using non-corrosive and highly resistant materials like stainless steel, copper, PVC, etc. But like any other material, they have a breaking point. Although these pipes seem rigid from the outside, they are prone to wear and tear over time from the inside. There have been several instances of cracks, slits, and seepages (these disfigurements are often the cause of leaks) that have caused the affected pipeline to burst open when left unchecked. Thus, it becomes an important task to regularly carry out maintenance work to maintain the structural integrity of these pipes, especially the ones carrying high density substances.

In a large-scale piping system, node level leaks or local leaks are the ones which occur because of sudden changes during normal operation. They don't necessarily disrupt the transportation but if these leaks are left unchecked, they may later lead to cause a failure in the entire pipeline. Usually, these leaks are isolated for repair work. Manual detection, for any leak, is impossible nowadays because of the sheer length of these pipelines. Therefore, an IoT based system is usually installed for monitoring certain key parameters like pressure, temperature, etc. These systems generally lack machine intelligence and are programmed to trigger a warning based on a rule, or a physics principle. A system like this has many drawbacks. It is not dynamic enough to recalibrate itself when its contents are changed or if a certain parameter is varied for some reason. It also triggers a number of false positives which may lead to unnecessary halting of the operation and wastage of time. Ultimately, all these problems lead to monetary setbacks for the corporation.

For building an AI based model for determining leaks, one needs to train it on equal number of datapoints from each class (or in other words, data should be fairly distributed amongst the classes present in the dataset). Proper model training is crucial in determining a model's real time performance. Leaks do not occur frequently or in other words are rare events (<1% of the total data). Most leak scenarios in these pipelines are not well documented. As a result of this imbalance in the normal and leak datapoints, classification-based leak detection becomes quite challenging. Conventional anomaly detection algorithms like isolation forest, clustering algorithms, interquartile range-based models, etc. don't help in solving the problem either because of the same reason. To sum up, an AI-based leak detection model needs to be structured in a way wherein it minimizes the cost of misclassification of anomalous datapoints as normal and normal datapoints as anomalous.

2. Related Works and Background:

Anomaly detection has been a buzz word in many sectors because of rapid technological advancements in hardware and software. In the real world, most organizations hence generally deploy physical or virtual anomaly detection systems to curtail failure rates. With the rapid rise of IoT devices, exponential number of sensors are being used to collect different types of data, over a stipulated duration of time, to ascertain efficient operation. Such datasets are referred to as time-series data. Time series data has many shortcomings. It contains varying patterns, different types of parameters, seasonality and in

most cases, it is highly unstructured. Thus, it becomes quite a challenging task to detect and localize anomalies precisely.

Anomaly detection systems, in the past, have been built using supervised [2, 3], semi-supervised and unsupervised [4, 5, 6] learning based methods. Supervised approaches include classification and regression-based models like SVM, ANN, CNN, Random Forest, Linear Regression, Logistic Regression, Isolation Forest, etc. These models are trained on datasets wherein normal datapoints are down sampled or anomaly datapoints are augmented for eliminating bias. But doing so may not be a good idea because the model, then, would not consider an anomaly as rare event thereby altering the very nature of the use case. Judging these models with accuracy being the performance metric may be misleading as accuracy does not take the number of false positives and false negatives into account. A metric like the Cohen Kappa score or MCC score would effectively validate the model's performance in a classification-based or regression-based anomaly detection system. Unsupervised approaches include principle-based modelling, clustering, generative adversarial networks, etc. These models tend to output many false positives and hence are not reliable enough to be deployed in a real-world leak scenario. Although there are a few generic anomaly detection methods [13, 14] which achieve great performance on benchmark datasets, they require a lot of analysis and modifications to the data as part of their implementation which may not be ideal. Such approaches, unfortunately, also require the stakeholder to make strong assumptions which may end up hampering the characteristic of the use case or the data. Despite, numerous shortcomings in these approaches, it is important to acknowledge the evolution of these methods as they culminate to form the bedrock for the current research being done in this area.

3. Reinforcement Learning:

In pursuit of a reliable anomaly detection framework, researchers are now looking at reinforcement learning based approaches as they follow an incremental self-learning process. Formally, reinforcement learning can be defined as a technique which allows an agent to learn and maximize its rewards in an interactive environment by trial and error using feedback from its own actions and experiences. Supervised and unsupervised learning involve predicting next value (a task driven approach) and identifying clusters (a data driven approach) respectively. Whereas reinforcement learning involves reward maximization through experience.

Markov Decision Processes or MDPs are used to formulate most RL problems mathematically. MDP is an extension to a Markov Reward Process where it contains an actual agency which takes decisions/actions. MDP essentially is a tuple <S, A, P, R, γ> where S is a finite set of states, A is a finite set of actions, P is a state transition probability matrix, R is a reward function and γ is a discount factor (γ ∈ [0,1]). Theoretically, a MDP is defined by its state, action sets and by the one-step dynamics of the environment. Given any state and action s and a, the probability of each possible pair of next state and reward, s', r, is denoted

$$p(s', r \mid s, a) = \Pr\{S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a\} \quad (1)$$

These quantities completely specify the dynamics of an MDP. Given the dynamics as specified in (1), one can compute anything that is environment related, such as the expected rewards for state–action pairs (R),

$$r(s, a) = E[R_{t+1} \mid S_t = s, A_t = a] = \sum_{r \in R} r \sum_{s' \in S} p(s', r \mid s, a) \quad (2)$$

the state-transition probabilities (P),

$$p(s' \mid s, a) = \Pr\{S_{t+1} = s' \mid S_t = s, A_t = a\} = \sum_{r \in R} p(s', r \mid s, a) \quad (3)$$

A generic RL model incorporating the above-mentioned fundamentals of MDPs is shown in Fig 1.

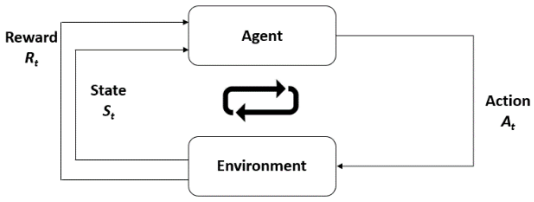

Fig 1: Action-Reward feedback loop of a generic RL model

All reinforcement learning algorithms involve estimating value functions. Value functions are functions of state or of state-action pairs that estimate 'how good' it is for the agent to be in a given state or how good it is to perform a given action in a given state. Value functions are defined with respect to particular policies. A policy, 'π', is a mapping from each state 's' and action 'a' to the probability π (a | s) of taking action 'a' when in state 's'. For MDPs, we define 'v_π' or state-value function for policy 'π' as,

$$v_\pi(s) = E_\pi[G_t \mid S_t = s] = E_\pi[\sum_{k=0 \text{ to } \infty} \gamma^k R_{t+k+1} \mid S_t = s] \quad (4)$$

where $E_\pi[.]$ denotes the expected value of a random variable given that the agent follows policy 'π' and 't' is any time step. A point to note is that the value of the terminal state is always zero. Similarly, one can define the value of taking action 'a' in state 's' under a policy 'π', denoted $q_\pi(s,a)$ or known as the action-value function for policy 'π' as,

$$q_\pi(s, a) = E_\pi[G_t \mid S_t = s, A_t = a] = E_\pi[\sum_{k=0 \text{ to } \infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a] \quad (5)$$

Note that optimal policies for multi-arm bandit problem are mentioned in the methodology section. After taking a brief look at the terms associated with reinforcement learning, it's time to understand the dataset and the use case.

4. Dataset Details and Use Case:

The dataset employed for the said task is taken from a simulation based on a real piping system. The simulation involves a piping system with fifty-one sensors which are placed at important junctions and nodes. The sensors record the pressure metrics at these nodes. Data collected from these sensors span for a period of five months starting from 01/04/2018 to 31/08/2018 taken a minute apart. Thus, in total each sensor has taken 220,320 readings.

The localization problem is to detect anomalous data points in each sensor and to localize the sensor having the highest leakage outflow by mapping the shortest path between the supervision center and the anomalous node. In doing so using multi arm bandit, questions like which sensor to monitor at an instant, how many times should an agent explore each node, etc. are all answered.

5. Methodology:

The framework's modules and flow are shown in Fig 2. The process starts with the local anomaly/leak detection system wherein individual time series of the critical sensors are given as inputs and sudden jolts in the data points are flagged as anomalies. These anomalies are then discretized for implementing the multi-armed bandit-based system which outputs the sensor with the maximum leakage outflow based on its local anomaly distribution. Lastly, a shortest path prediction system maps an optimal path to reach that particular node from the supervision center.

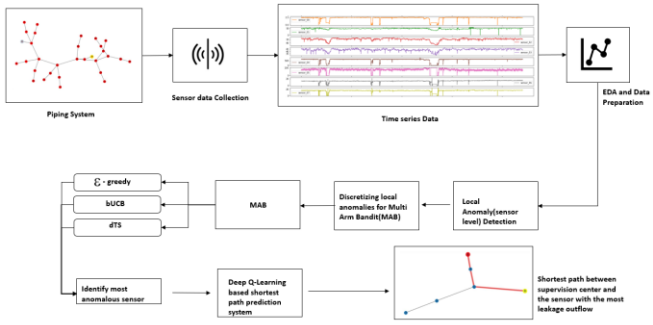

Fig 2: System Diagram

A. Node Level Leak Detection

Successful identification of node level leaks is crucial in avoiding a massive shutdown at a later stage. For this, a LSTM autoencoder-based model was utilized. As part of the hyper localization process, all the sensors are run through the model mentioned below and the top seven sensors with the most anomalies were focused upon for further operations.

An LSTM autoencoder is a type of recurrent neural network wherein the input given is reconstructed back as the output. It comprises of three parts: the encoder, the bottleneck, and the decoder. The encoder is responsible for taking in the input and producing a compressed encoding of the input. The bottleneck is the hidden layer where this compressed encoding is produced. This layer determines the dimension of the encoding. The decoder is responsible for taking the compressed encoding and recreating it back to a form which is similar to the input. The primary objective of such a model is to curtail the reconstruction loss.

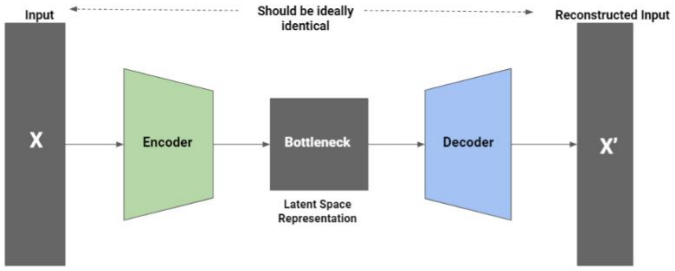

Fig 3: Structure of an autoencoder

The reason for implementing an autoencoder for detecting node level leaks is the fact that these leaks are essentially sudden changes in normal operation and thus will have a higher reconstruction error than that of the datapoints falling under the normal operating window. Hence, it becomes a good fit for solving an unsupervised anomaly detection problem. This solution is implemented on individual time series of the sensors as part of the localization process.

As part of the data preprocessing for implementing this solution, the data is split into a ratio of 75:15:10 for the training, testing and validation sets. After splitting, the data is normalized. A LSTM network requires the data to be fed in be in a specific format, which is, (number of samples, time steps, number of features). As a result, the input data is reshaped in the mentioned format. A time step of 30 was decided. So ultimately, the input data (training set) was reshaped from (198288, 1) to (198258, 30, 1).

A number of variations of the autoencoder architecture were tested before employing the below-mentioned architecture. It was finalized after comparing the loss values attained by individual models after training. The model with the least loss value was chosen, that is, the eleven layered LSTM autoencoder in this case (refer Table 1).

Table 1: Architectures and their associated loss value

Architecture of AE	Loss Value
5-layer	0.071
7-layer	0.062
9-layer	0.053
11-layer	0.042
13-layer	0.048

The reshaped data is then pushed into an eleven layered sequential autoencoder model (refer Fig 4). The encoder comprises of stacked LSTM layers, with the input layer comprising of 64 units and the subsequent layers comprising of 32, 16, 8, 4 and 2 units. The last layer of an encoder is the bottleneck layer. Here, a bottleneck layer is of 2 units which is also the dimension of the compressed encoding. This compressed encoding is then fed into the decoder which also comprises of stacked LSTM layers with 4, 8, 16, 32 and 64 units. A

repeat vector layer is placed after the encoder to bring back the original dimensions and a time distributed layer is placed at the end in order to get the output. Adam is used as the optimizer and the learning rate is set to 8e-5. Mean absolute error is utilized to estimate the reconstruction error.

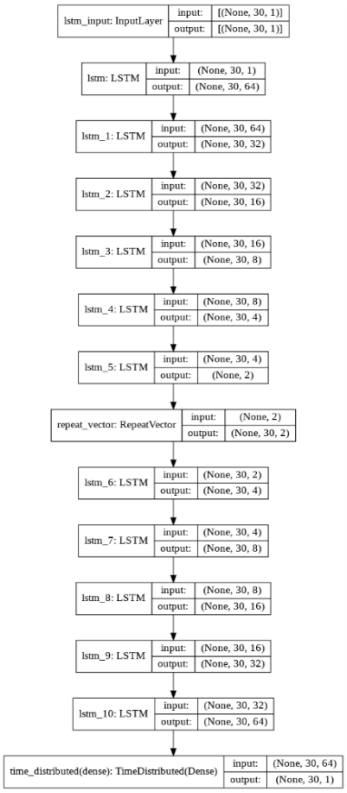

Fig 4: Model Architecture

The loss function curves (refer Fig 5) plotted for both the training and testing process suggest that the autoencoder model is a good fit.

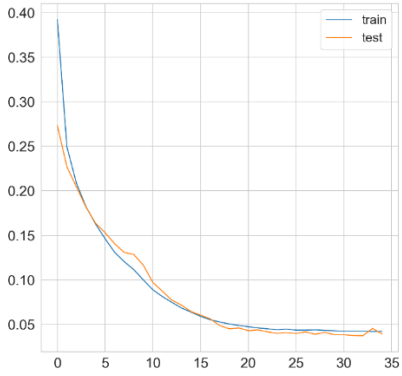

Fig 5: Loss Function

The model is then tested on the training set and the reconstruction error is computed for the included datapoints. A density plot is then utilized to determine the threshold for detecting anomalous datapoints. This threshold is decided as per the business requirement and is unique for each sensor.

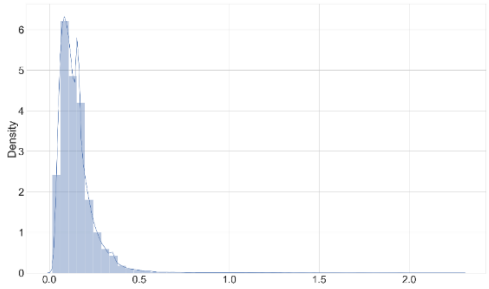

Fig 6: Density plot for sensor 11.

Table 2: Threshold Table for critical sensors

Sensor Number	Threshold Value
02	0.06
05	0.5
07	0.09
11	0.6
40	2.0
41	0.5
47	3.0

After setting up the threshold, the model is tested on the validation set and the reconstruction error for those datapoints is also computed. If a datapoint's reconstruction error shoots up above the set threshold then it is flagged as an anomaly. Table 3 contains the number of anomalous datapoints of the top seven sensors.

Table 3: Number of anomalies

Sensor Number	Number of Anomalies
02	151
05	93
07	426
11	132
40	120
41	142
47	184

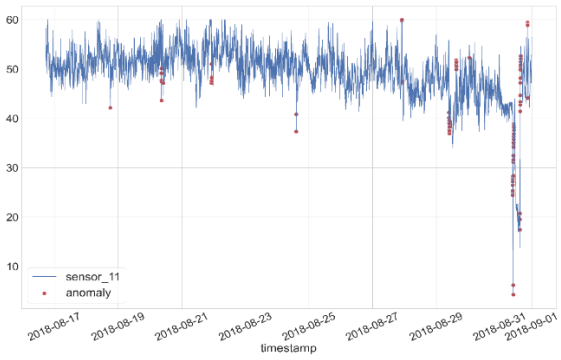

Fig 7: Anomalies in datapoints of sensor 11

This method of determining anomalies is amongst the best as it doesn't need the stakeholder to set up preconceived assumptions beforehand and works purely on a mathematical level.

B. Implementing Multi armed bandit policies for detecting the sensor with maximum leakage outflow

After extrapolating anomalous datapoints from the list of critical sensors, the next task is to find the most anomalous sensor, that is, the sensor which has the maximum leakage outflow. Sensor with maximum leakage outflow does not necessarily imply choosing the one with the most anomalies. It is choosing a sensor which maximizes reward on the basis of its reward distribution. In order to generate this distribution, the detected node level anomalies, are discretized to 0 and 1 using recursive discretization using gain ratio for binary values and corresponding reward probabilities are calculated for setting up the binomial bandits. This discretization methods representatively selects minimum samples from the classes and uses the gain ratio metric that maximizes the reward likelihood.

Table 4 holds the reward probabilities for all the critical sensors.

Table 4: Reward Probabilities

Sensor Number	Reward Probability
02	0.0068
05	0.0042
07	0.0194
11	0.0059
40	0.0054
41	0.0064
47	0.0084

As one can infer from the table, anomalies, in most sensors, are less than one percent of the sample size. Hence, this technique still acknowledges the fact that anomalies are rare events in any real-world scenario.

Here, the critical sensors are the bandits. Each one of them has a unique reward probability and a distinct reward distribution. The objective is to decide which bandit must be picked to play at each timestep in order to maximize the cumulative reward in the end (the agent is rewarded upon choosing an anomalous datapoint). Ultimately, the bandit with the highest cumulative reward is found to be the most anomalous sensor. To tackle the exploration-exploitation dilemma in multi arm bandits, three popular policies are compared in the process. The policies compared are: ϵ-greedy, Deep UCB and Deep Thompson sampling.

ϵ-greedy policy is a simple policy. The concept is to select a bandit based on the best greedy action with a probability of 'ϵ' and others with a probability of '1-ϵ', that is, to select a bandit based on its maximum reward prospects (in this case the agent ultimately goes for the sensor having the highest empirical reward probability). To demonstrate the greediness of this method, the epsilon value is set at 0.1.

Fig 8 below shows how the agent picks a bandit based on the ϵ-greedy policy.

Fig 9 shows the beta distribution after these draws.

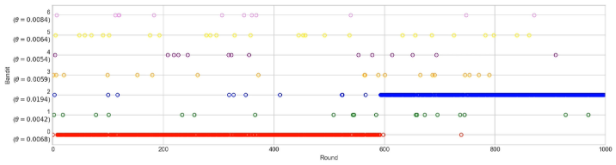

Fig 8: ϵ-greedy policy across random draws

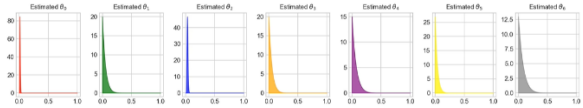

Fig 9: beta distribution after random draws (ϵ-greedy policy)

After analyzing these plots, one can conclude that this policy has its shortcomings. Firstly, tuning the epsilon value is difficult and in most cases is not something trivial. Secondly, exploration is always constant (ineffective) and hence it does not necessarily give us the highest possible reward. Lastly, the risk of suboptimal decision making is high because of the two reasons mentioned above.

The UCB policy prefers selecting bandits with the highest payoff both exploration wise (contact) and exploitation wise (reward). The policy is fairly straight forward. It takes the average reward and the number of times a bandit is picked for every action and tries to maximize the cumulative sum of the two quantities. It picks the maximum upper confidence bound value thereby balancing both exploration and exploitation and prefers arms which look promising even if they are played less.

The traditional UCB

$$A_t = argmax(Q_t(a) + c\sqrt{\frac{\ln(t)}{N_t(a)}})$$

But the Bayesian UCB (bUCB) assumes of each arm are normally distributed, and alter the UCB term with the standard deviation of arm's rewards and tuning the adjustable hyperparameter for determining the size of the confidence interval we are adding to an arm's mean observed reward

Fig 10 below shows how the agent picks a bandit based on bUCB policy across random draws.

Fig 11 shows the beta distribution after these draws.

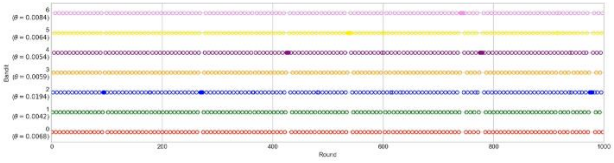

Fig 10: bUCB policy across random draws

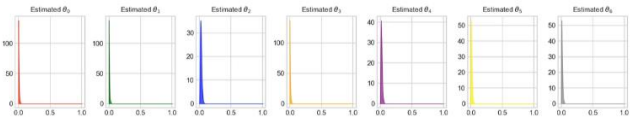

Fig 11: beta distribution after random draws (bUCB policy)

After analyzing these plots, one can conclude that bUCB policy explores far more than the epsilon greedy policy. The random draws also show that it prefers getting rewarded at a later stage but does not give up on sensors immediately (does not pick the sensor with the highest empirical probability) and explores all of them. Another advantage that is evident from these plots is that it does not explore at a constant rate thereby assuring reward maximization.

Thompson sampling works on the principle of probability matching. At every round, a bandit is chosen with a probability of it being an optimal option. This is done by computing the posterior distribution of reward probabilities for each bandit. A singular sample with the maximum value is drawn out of the computed distribution at every round. This approach allows optimal exploration and eliminates unpromising arms (not as greedily as epsilon greedy policy) by giving them a high uncertainty (it does not rule out unpromising bandits immediately and waits until sufficient information is verified from a bandit's beta distribution). But when the best arm's distribution sticks out (by considering uncertainty), it exploits it aggressively.

Fig 12 below shows how the agent picks a bandit based on Deep Thompson sampling(dTS) across random draws.

Fig 13 shows the beta distribution after these draws.

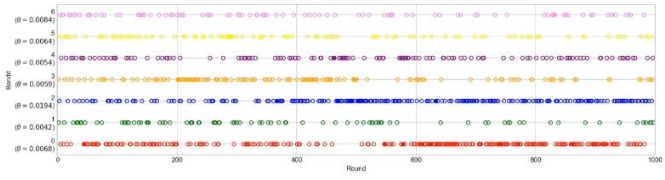

Fig 12: Deep Thompson sampling across random draws

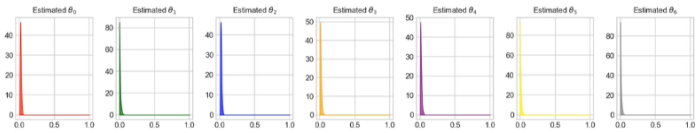

Fig 13: beta distribution after random draws (Deep Thompson sampling)

The algorithm for Deep Thompson Sampling shown below

Algorithm 1: Deep Thompson Sampling

Inputs: Number of rounds, exploration variance, network width, regularization parameter

1. Initialize the collection of parameters of the neural network
2. for t=1…T(no of rounds)
3. for k=1…K(no of arms in bandit)
4. Select an arm based on the reward of each arm from the reward's posterior distribution and then pulling the greedy arm
5. end for
6. Observe reward
7. Update the posterior
8. end for

After analyzing these plots, one can conclude that dTS is the most efficient when compared with both the bUCB policy and epsilon greedy policy as it handles the exploration-exploitation dilemma far more intuitively. A point to note here is that all the distributions for the three policies are on the left side as the agent does not find rewards or anomalies as often since anomalies are rare events.

To check the performance of the three MAB policies on the discretized node level anomalies, the regret curves (refer Fig 14) for the three policies are plotted. Regret is the difference between the reward from the best possible action and the one which is actually taken. We have only considered the positive regret.

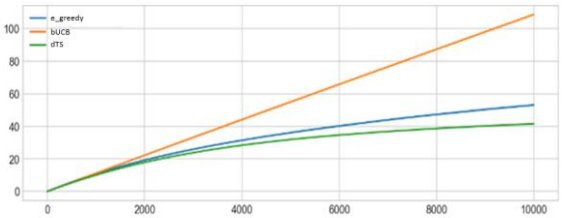

Fig 14: Regret curves for the three MAB policies

The result obtained from the regret curves show that dTS outperforms the epsilon greedy policy and the bUCB policy. It stands more accurate as it does not work on hyperparameters or any dependencies. An

interesting observation is that the epsilon greedy policy outperforms the bUCB policy. This is because the bUCB policy spent a lot more rounds exploring whereas the epsilon greedy policy went ahead by exploiting the cumulative reward. The sensor pointed as most anomalous by dTS was sensor 07 or bandit 2. This was communicated to the shortest path prediction system. A point to note here is that all the three policies correctly identified the node with the highest leakage outflow. The dTS result was taken because the arm selection curve for bandit 2 peaked early as compared to the other policies.

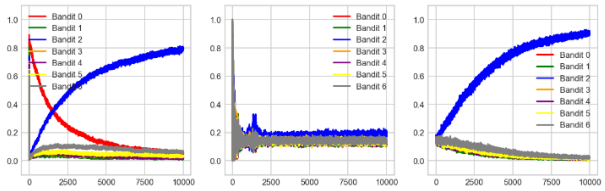

Fig 15: Most anomalous sensor pointed by ϵ-greedy policy, bUCB and Deep Thompson sampling (left to right)

C. Deep Q-learning based shortest path prediction system

After finding the sensor with maximum leakage outflow, the final task of localization is to find the shortest path between the supervision center and the anomalous node. The purpose of building such a system is to aid the maintenance team in isolating the node and to give them a quick pathway to reach there. A deep Q-learning based approach is employed here to keep the system dynamic.

Before getting into the workings of the algorithm, it is imperative to understand the setting or environment in which it is implemented. The pipe network (where the critical sensors are placed) is laid out as shown in Fig 16. The sensors are mounted on the inner side of the pipes. Node 0 of the graph is the supervision center (highlighted in yellow) and nodes 1 to 7 represent the sensors in the pipeline. The critical node or the one with the maximum leakage outflow is node 7 (highlighted in red). The task at hand is to find an optimal path between node 0 and 7. Visually, one can point out that the optimal path of traversal in this case is (0, 1, 4, 6, 7) but one must imagine this problem at a larger scale to fully understand and appreciate the efficiency of this system.

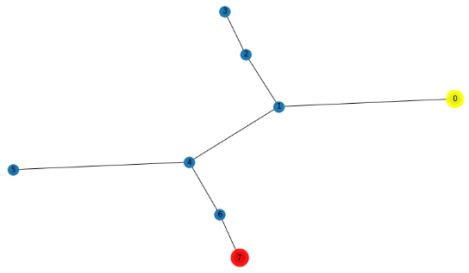

Fig 16: Pipe Network

Q Learning works on an intelligent reward-feedback mechanism. This mechanism involves setting up of the rewards table, Q table, discount factor and the iterations for training. The rewards table is a matrix which holds scores of all the paths the model can take. This matrix is a square matrix of size 'n' where 'n' is the total number of nodes in the graph (in this case n=8). Firstly, the entries of this matrix are initialized to -1 and then entries corresponding to all the nodes on different paths are changed to 0. The entries corresponding to the nodes on the goal paths are changed to 100 (this value can be any large value which will allow propagation during Q learning). The Q table is set up when the model starts training. This table keeps the scores of all the different paths the model takes while training. The dimensions of the Q table are similar to that of the rewards table. Unlike the rewards table, all entries in the Q table are initialized to 0. Each move is recorded using the below mentioned formula,

$$Q\,[State, Action] = R\,[State, Action] + \gamma \times \max\,(Q\,[next\ state]) \quad (6)$$

Thus, with the help of the Q table, the model not only monitors the current score but also is on the lookout for the previous scores to optimize the entries going forward (rewards are taken from the rewards table). The 'γ' parameter in this formula is called the discount factor. This factor is a tunable parameter and can hold any value between 0 and 1. A value nearer to 0 will make the model go for immediate rewards and a value nearer to 1 will make the model go for trying alternative paths and take the reward at a later stage. The discount factor is set to 0.8 in this implementation. Finally, this process is looped over, using an optimum number of iterations.

We have implemented the Deep variation of the Q-learning algorithm which uses a neural network to approximate the Q value function.

A summarized algorithm for deep Q learning is mentioned below.

We have implemented a neural network for Q-training by generating sufficient dataset for of values for state and correct q values. We will store all possible movements as experience and that will be used to predict the action to be taken to the next state

---

Algorithm 2 : Algorithm for deep Q learning

1.Initialize Q for all pairs of state and actions

2. s=initial state = start node i.e. supervision center

3. while (convergence not achieved)
3. simulate action to reach state s_a
4. if(s_a is not terminal node)
5. receive reward
6. receive new node
7. if(s_a is terminal node)
8. receive reward
9. return optimal path

After a substantial number of epochs, the model stabilizes the Q values following which the score is compared to the scores in the rewards table to check for the optimal path. It is trained using early stopping to avoid resource wastage and stops automatically after the Q scores are

confirmed (for the optimal path) for a minimum of 10 iterations. The results of the model are shown below.

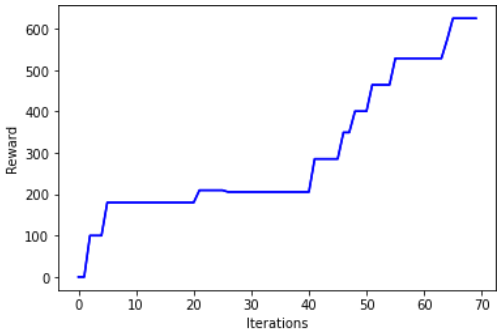

Fig 17: Q learning search score

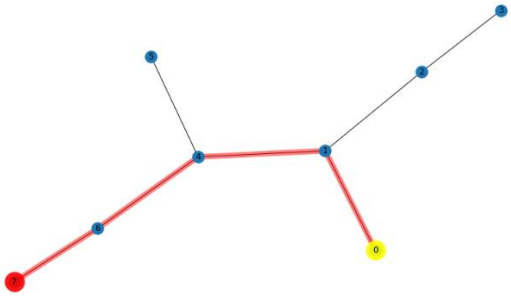

Fig 18: Optimal Path (0, 1, 4, 6, 7)

Fig 17 shows the search score with successive iterations. Inference from the plot is that the model converges at about the 70th iteration to find the optimal path which is shown in Fig 18.

6. Conclusion:

The ultimate objective of this research endeavor was to incorporate reliable machine intelligence in leak detection systems. The presented framework stands validated as it intuitively manages to answer all the questions that were thrown at it. The novelty of this approach is the consideration of anomalies as rare events which allows the framework simulate results which are real world like. This also allows it to be a viable candidate for replacing or aiding rule/physics-based anomaly/leak detection systems in the near future. The incorporation of reinforcement learning in the form of MAB policies showcases its immense potential in the relatively unexplored sector of industrial grade piping and cabling. The inclusion of the Q-learning based shortest path prediction system is an added benefit for a leak detection system because of its ease at mapping dynamic paths for the maintenance team to reach the target node. This, ultimately saves a lot of valuable time, in case of a catastrophic emergency. Reinforcement learning based approaches are not only quick at coming up with the required output but are also accurate. This blend, thus, makes quite a compelling case for it when one decides to incorporate such a system in a real-life situation.

References:

[1] Yu, Mengran & Sun, Shiliang. (2020). *"Policy-Based Reinforcement Learning for Time Series Anomaly Detection"*. 10.13140/RG.2.2.29517.79844.

[2] S. Chauhan, L. Vig, "*Anomaly detection in ECG time signals via deep long short-term memory networks*", in IEEE International Conference on Data Science and Advanced Analytics, 2015, pp. 1–7.

[3] A. R. Tuor, R. Baerwolf, N. Knowles, B. Hutchinson, N. Nichols, R. Jasper, "*Recurrent neural network language models for open vocabulary event-level cyber anomaly detection*", in: Workshops at the AAAI Conference on Artificial Intelligence, 2018, pp. 285–293.

[4] S. Ahmad, A. Lavin, S. Purdy, Z. Agha, *"Unsupervised real-time anomaly detection for streaming data*", Neurocomputing 262 (2017) 134–147.

[5] O. Gorokhov, M. Petrovskiy, I. Mashechkin, "*Convolutional neural networks for unsupervised anomaly detection in text data*", in: International Conference on Intelligent Data Engineering and Automated Learning, 2017, pp. 500–507.

[6] K. Ghasedi Dizaji, X. Wang, H. Huang, "*Semi-supervised generative adversarial network for gene expression inference*", in: International Conference on Knowledge Discovery and Data Mining, 2018, pp. 1435–1444.

[7] R. S. Sutton, A. G. Barto, "*Reinforcement learning: An introduction*", MIT Press, 2018.

[8] Oliveira, Eduardo & Fonseca, Mário & Kappes, Daniele & Medeiros, Arthur & Stefanini, Ihm & Brazil, (2018). "*Leak Detection System using Machine Learning Techniques.*"

[9] Fuentes V.C., Pedrasa J.R.I. (2020) "*Leak Detection in Water Distribution Networks via Pressure Analysis Using a Machine Learning Ensemble.*" In: Pereira P., Ribeiro R., Oliveira I., Novais P. (eds) Society with Future: Smart and Livable Cities. SC4Life 2019. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 318. Springer, Cham.

[10] Liu, Y., Ma, X., Li, Y., Tie, Y., Zhang, Y., & Gao, J. (2019). "*Water Pipeline Leakage Detection Based on Machine Learning and Wireless Sensor Networks"*. Sensors (Basel, Switzerland), 19(23), 5086.

[11] Sidra Rashid, Usman Akram, Shoab A. Khan, "*WML: Wireless Sensor Network based Machine Learning for Leakage Detection and Size Estimation*", Procedia Computer Science, Volume 63, 2015, Pages 171-176, ISSN 1877-0509.

[12] Zhao, H., Wang, Y., Duan, J., Huang, C., Cao, D., Tong, Y., Xu, B., Bai, J., Tong, J., & Zhang, Q. (2020). "*Multivariate Time-series Anomaly Detection via Graph Attention Network*". *2020 IEEE International Conference on Data Mining (ICDM)*, 841-850.

[13] N. Laptev, S. Amizadeh, I. Flint, "*Generic and scalable framework for automated time-series anomaly detection*", in: International Conference on Knowledge Discovery and Data Mining, ACM, 2015, pp. 1939–1947.

[14] S. Venkataraman, J. Caballero, D. Song, A. Blum, J. Yates, "*Black box anomaly detection: is it utopian?*" HotNets (2006) 127.

[15] Vasanth Sena P., Porika S., Venu Gopalachari M. (2021) "*A Mining Framework for Efficient Leakage Detection and Diagnosis in Water Supply System*." In: Kumar A., Mozar S. (eds) ICCCE 2020. Lecture Notes in Electrical Engineering, vol 698. Springer, Singapore.S

[16] Coelho, João & Glória, André & Sebastião, Pedro. (2020). "*Precise Water Leak Detection Using Machine Learning and Real-Time Sensor Data*." IoT. 1. 474-493. 10.3390/iot1020026.

[17] – Weitong Zhang and Dongruo Zhou and Lihong Li and Quanquan Gu (2020) "*Neural Thompson Sampling*"