Proceedings of the SMART–2022, IEEE Conference ID: 55829
11th International Conference on System Modeling & Advancement in Research Trends, 16th–17th, December, 2022
College of Computing Sciences & Information Technology, Teerthanker Mahaveer University, Moradabad, India

2022 11th International Conference on System Modeling & Advancement in Research Trends (SMART) | 978-1-6654-8734-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/SMART55829.2022.10047250

A Graph Database-Based Method for Network Log File Analysis

Ms. Khushboo Sharma[1] and Arvind Kumar[2]
[1]Assistant Professor, Department of Electrical Engineering, Vivekananda Global University, Jaipur, India
[2]Associate Professor, Department of Mechanical Engineering, Chandigarh Engineering College, Jhanjeri
E-mail: [1]khushboo.sharma@vgu.ac.in, [2]arvindkumar.j1753@cgc.ac.in

Abstract-It is frequently required to evaluate data acquired from numerous network logs in order to offer a more accurate judgement of the severity of a cyber threat. This may be accomplished by reviewing the data. One example would be the examination of log records produced by command line tools in solitary confinement. In spite of the fact that employing a log management system makes it simpler to identify certain log documents, it might be time-consuming to examine all of them in order to determine the connection(s) between the different types of network activity. To construct these bridges, we may make use of relational databases by, for example, linking the tables by using complicated database queries that make use of a large number of join processes. Recent years have seen a rise in interest in the use of graph databases as a method for the storage and organization of data in preparation for semantic searches.

Keywords-Network Log, Graphical approach, Parsing, Importing, Nodes, Python.

I. INTRODUCTION

One of the various tools used by data security operation centers (SOCs) is network safety monitoring (NSM). As a technique, NSM involves monitoring, logging, and analyzing network information in real time to spot malicious activity as it occurs. An NSM environment is suitable for the application of a variety of network intrusion detection strategies, such as reputation-based detection structures. Signature-based intrusion detection systems (IDS), such as Laugh or Suricata [1,] are used to investigate network packets in search of potentially malicious positive patterns.

II. PROPOSED METHODOLOGY-

Before explaining our method for modeling network log files using graph databases, we will describe how log files are created in this section.

Preparation of Log Files

A nice case study may be found in Zeek's connection logs (conn.log), DNS logs (dns.log), and HTTP logs (http.log). Nevertheless, the approach that we have proposed is applicable to any log record. As was discussed before [2,] the process of handling new information entries in log records has to take place.

Examining the Logs

The log files must be monitored to verify that no log data input goes unreported and that log rotation is handled appropriately. One method for monitoring log files is to use the built-in linux program with the tail option [3].

Log Parsing and Importing

The recently discovered log data is imported into a Python database so that additional analysis may be performed. To create a data graph that can be integrated into a Neo4j database table, a function will take this thesaurus of log entries as an input [4]. A Python tool developed by a third party and referred to as Py2neo2 may be used to assist with the construction of the graph's nodes and institutions. The data from the graph will be put into Neo4j at that point.

Conn.log Log File Modeling

The conn.log file is used to track and log generic TCP, UDP, and ICMP traffic information. Flow semantics may be used to analyze the recorded data, which includes information on the host machines participating as well as additional metadata [5]. an example of a conn.log record is shown in Figure 1.

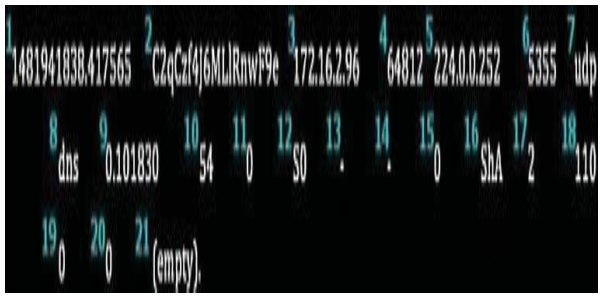

Fig.1 shows an example of a conn.log record [18]

The following suggested graph model is presented as a result of an investigation of the information included in Zeek's conn.log for the purpose of identifying entities and relationships. In operation, as seen in Figure 2, is the conn.log graph data model.

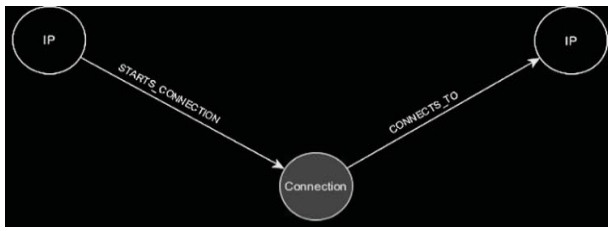

Fig. 2. An overview of the conn.log graph data structure [18]

As its name implies, the "Connection" node is a representation of the connection and stores the vast bulk of its information [6]. Both the sending and receiving host computers are represented by a single "IP" node, which has the Internet protocol address as a solo property shown in Table 1.

Table 1 - Details about conn.log graph nodes and its

Name of node	Name of Property
Link	aid, service, progabides , proto, presbyters, resp bytes, tunnel parents, missed bytes, low-calorie, locales, ringspots,epopts, constant, history , duration,
Internet protocol	Internet protocol 4.0

The relationships between the nodes are distinct from one another and tie them all together. The "IP" value, which identifies the host from which the connection is launched, is linked to the "Connection" via the "STARTS CONNECTION" property, as shown in [7]. This connection is established between the "IP" value and the "Connection." The originating port of this connection is used as an identifier in the dating request that is sent from the IP node to the relationship node. This request is sent when a dating request is submitted from the IP node. It is decided that a connection with the name "CONNECTS TO" shall be established between the node known as "Connection" and the node known as "IP," which acts as a stand-in for the target host. The connection node is linked to the IP node, and the IP node is connected to the port on the target host. There is a link among the joining node & the Internet protocol node shown in Table 2.

Table 2. Relations of conn.log chart information model

Relations	Characteristic	Relations from -> to
START_MAC	RJ45	Internet protocol -> HTTP
CONNECT_TO MAC	RJ45	Connections -> *HTTP*

dns.log Log File Modeling

Zeek sends a copy of every DNS web client it detects to the dns.log file. A DNS query's time, Servers, port, & protocols settings, the asked domains, the query's resolution, and various enquiry information are all recorded. The data from the dns.log file is shown below. Figure 3 illustrates a dns.log entry sample.

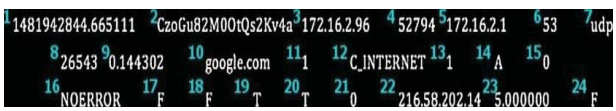

Fig. 3. Example of a record of dns.log [18]

The proposed network is the outcome of modeling an information graph utilizing various components. Figure 4 displays a graphical database structure for the DNS.log.

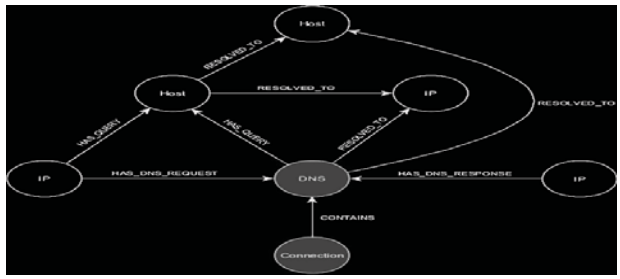

Fig. 4. Dns.log graph data model [18]

Due to its reliance on preexisting information, the clustering algorithm produces outputs that significantly deviate from reality. That's why we went with a method for mining associations, called "association rule mining," in this work. As a data mining technique, association rule mining is quite popular. Essentially, it's a strategy for mining the raw data for actionable patterns. The log of a firewall or IDS is a valuable resource for network forensics. Attribute information in the data includes things like flow size, record type, and source and destination IP addresses and ports. Input network records are broken down into groups of characteristics, and each repeated item represents a group of those qualities. By mining for frequently used items, we learn that the system is vulnerable to a port attack because of the presence of set source IP, destination IP, and stream size. Port scanning attack flows may be represented by compiling records of IP addresses that appear together in groups of frequently used items. The Apriori method and the FPTREE algorithm are examples of classical frequent patterns mining algorithms, and they do not need much training data (Tables 3-4). It has been shown that big log data processing and storage systems that rely on text files or conventional relational databases to store log data have poor storage space utilization and low data dependability.

That's not a good approach to keep track of or store all those logs. Also, the storage devices themselves may be spread. These decentralized storage systems make use of multi-copy technology to guarantee the integrity of their data, however this safeguarding method leads to inefficient use of available storage. Additionally, the log data, whether the raw data or the analyzed findings, may only be kept in files on the distributed storage device. The file is the sole means of accessing these records.

The Dgraph cluster, which is composed of two kinds of compute nodes, is the central component of data management.

Dgraph Zero is the central component that manages the cluster and coordinates the database and the analysis. Dgraph Alpha nodes with indexed data do the processing. At least one Zero and Alpha node are needed \sto handle stored data. The database's documentation contains more information on the system's data analysis features. Components of an indexing system and a graph database that communicate directly with a running instance of Dgraph make up the Data handling module. The indexing feature takes RDF triples that have been uploaded and indexes them, saving them in a local database. The core of the module is the MapReduce-based Dgraph Bulk Loader. It makes efficient use of the available computing resources. In addition, we may tell the component how many Alpha nodes to use in the next graph database step. As a result, the cluster can store and process massive amounts of data without slowing down research. The indexing process

produces binary files that include the indexed data and the indexed data itself. One benefit of this method is a decrease in processing time required whenever data is refreshed. Also, the created index may be used by a different Dgraph installation, perhaps one running on a more potent compute node.

When the file steadily rises, access to the file will likewise be more time-consuming. Graphs may be used to model a wide variety of intricate connections between data elements.

Due to the constraint of scalability, conventional relational database has been unable to satisfy the requirements of large-scale data.

The graph (graph) model is used to store information in a specialized No SQL database called a graph database. You may think of the graph model as a subset of the key-value one. The main differences are its ability to construct a graph structure and its support for algorithms based on that structure. When it comes to relational administration and relational query logic, a graph database is well suited due to its scalability and adaptability. Graph databases often come with a query language well-suited to describing both graph structure and graph query. This study utilizes the graph database Neo4j store the log data.

You may simply do a query on the graph, for instance, to find out how much time has elapsed since a certain IP address was allocated to a particular geographical location.

Table 3. DNS nodes & their characteristics. data-log diagram strategy

Nodes	Property
Domain Name System	Every piece of hardware that is linked to the Internet is assigned a unique IP address, which other computers may use to locate the hardware.
IP	Internet Protocol (IP).

Table 4. Dns.log static data architecture connections

Relation	Area	Connection from -<= to
CONTAIN	TTL	Connection Domain Name System
HAS_DNS_REQUEST	TTL	IP
HAS_DNS_RESPONSE	TTL	IP

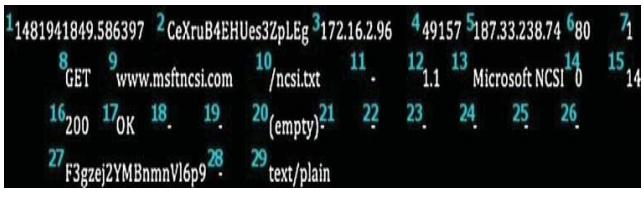

Fig. 5. An exemplar of a http.log file [18]

The data in http.log was analyzed, and a data graph with at least eight nodes representing the various entity kinds was proposed as a consequence. Figure 5 is an example of the information that may be found in the http.log file.

Fig. 6. http.log's graph based architecture [18]

The graph data model for http log is shown in Figure 6. There is a table shown below that shows the relationship between the various contains and reality form that can be seen from the table below. The number of the table can be given as the Table 5 as seen below.

Table 5. http.log graph based architecture connections

	Association from -<= to
CONTAIN	Link -<= HTTPS
REQUEST	IP -<= HTTPS
RESPOND	IP -<= HTTPS
NEXT	HTTPS -<= HTTPS

III. INDEXES AND RESTRICTIONS

In Neo4j, adjacency connections may be established without the requirement for an index to be present beforehand. On the other hand, it is categorized in such a way that makes it easier to discover good jump off sites for traversing graphs. In addition, the data integrity enforcement capabilities of Neo4j are enhanced by the software's ability to impose constraints not just on nodes but also on the links between them. If this condition is satisfied, all of the nodes that have a certain label will have the same value for each attribute. When constraints are applied, these results in the creation of an index that is specific to the node and attribute that are provided [11]. Each node in Neo4j may have its own labels, relationships, properties, and other attributes. Commonly, data pertaining to an entity is kept in the node. It is possible to assign numerous labels to a node for the purposes of indexing and placing minimal restrictions on the model. The nodes are linked together by the connection, and between every pair of nodes, there might be numerous relationships pointing in various directions. Properties, represented as key-value pairs, may be assigned to a node or a relationship. To this end, we suggest the following modelling guidelines, which are informed by the hierarchical multi-domain NSSA model and the Neo4j data structure:

Node: A node is any internal entity or external item with external interactions. To simplify processing and data analysis, we give any entity or object that has a relationship with other nodes the same name and treat them as a single node.

Label: There may be several labels assigned to a node, and when a node is part of a certain domain, that domain's label is applied to it. The node's labels, which are all capitalised, may be added to the query management system as well.

Property: Basically, a node's property is any data about the

node that does not affect how the node interacts with other nodes. The node's name and any relevant information are considered properties. There may be several key-value pairs for a single property.

Relationship: Various ties, including subordination, communication, and connection, all contribute to the network's overall structure, which is known as the relationship. The nodes may have zero or more directed connections, with the exception of the start and end nodes. While a relationship may have several characteristics, it may only have one type, and that type must be named entirely in capital letters.

Because of this, the majority of the nodes in the graph will be subject to additional constraints as a consequence of utilising the suggested graph data model. These limits will be imposed on the nodes. The existence of the singularity ensures that there will always be one and only one node that serves as a representation of a certain notion. The table that follows contains a list of the constraints that are used to ensure separate nodes. These constraints are different for each linked feature.
Evaluation Setup and Results

Simulating community site visitors that can be discovered.
The checks were performed the usage of Oracle Virtual Box, model five.zero.forty, on a virtual machine. the subsequent settings have been used to configure the virtual machine [10].
- An inner network adapter, Zeek version 2.five-seventy six, Neo4j model three.2, and Ubuntu Server 16.04.2 LTS with two CPUs and 10 GB of RAM. A ThinkPad X1 T530 with an I7 CPU and 8 GB ram served as the virtual machine's server Computer.

Import Log data

several packet capture files had been used to mimic community interest at the same time as testing the Python programmes. The 1999 DARPA Intrusion Detection evaluation records Set's week one pcap files. A couple of the capture files included the year 2012 MACCDC [12] as well as the maccdc 2012 00000.pcap [13].The output became in comparison to the findings of Neise's in advance paintings the use of the MACCDC capture [14]. The DARPA files contain 24-hour worth of network site visitors. consequently, replaying these facts additionally required 24 hours. Replaying the DARPA packet capture files found out no difficulties with processing pace, suggesting the prospect of (close to) real-time analysis. Four replays of every file have been additionally executed. despite the fact that a short put off (kind of up to 2-3 min) ought to occasionally be seen among information logging and eating it into Neo4j, the put off became usually speedy made up for as quickly as community traffic reduced. The limits of the way we imported information into the graph database were made clear through the replay of the MACCDC file. in this document, 8,635,943 packets totaling 935,501,635 bytes were transferred across the network in just below 71 minutes. Best more or less one hundred thirty out of just about 04.000.000 graphs were analyzed and kept in Neo4j after this record turned into replayed. We processed almost all 15,000 DNS graphs. Most effective little greater than 97,000 out of over 3,900,000 HTTP graphs and around 18,000 of the nearly 76,000 (about 42%) have been processed [15]. There were produced connection graphs (around 2. 5%). With these final results, it turned into not viable to analyses the HTTP and Connection graphs in close to actual time. Handiest the DNS graphs will be nearly immediately studied. Following many trials replaying with It became showed that the records import on this instance become excessively sluggish through the use of several speed options. The last end result becomes that uploading the information to Neo4j and analyzing the log files from the MACCDC report took near 40 hours. In order to assess the speed of sending files into to the graph dataset, the following tests were taken. Prior to the tests, Zeek was employed to evaluate DARPA's "outside.tcpdump" from the Weekend of week 1 in order to guarantee a steady flow of log file. The created log papers served as the starting point for the study. Table 7 displays the diverse range of logins in accordance to file system [16]. Table 6 displays the total amount of logins in the network

Table 6. The number of entries in the log

Log file	Number of entries
com	49831
dnss	7343
https	55397

Database Searching-

Now that the graph database has been absolutely populated, the graphs can be analyzed. Neo4j's graphical illustration of the facts graph's shape is visible in discern 8. The examine turned into conducted while tracking and replaying Week 1 Thursday facts from DARPA at a 4-fold pace multiplier. The database protected approximately 1,0.5,000 nodes, 5,439,000 relationships, and 8,435,000 assets items when the subsequent queries were carried out [17].

The community hosts which have reported the most suggested outbound connections thus far are returned by way of the following query:

i:IP suit - [sc:STARTS CONNECTION] (c:Connection) go back with remember(sc) as overall connections and that i.ip as host. by using overall connections DESC

In 1,847.00 milliseconds, the preceding request discovered 45 particular servers as well as the range of outbound connections associated with them. The three top findings from your query are listed below, condensed for simplicity. Table 7 displays the findings of the search for the overall number of contacts.

Table 7. Results of query for total number of connections

SERVER	LINKS
“171.26.112.99”	39088
“171.26.117.53”	22024
“171.26.115.86”	98864

The number of outgoing links is the simple outcome of the aforementioned search. Change the test in reality to: "" to see if the domains that have the most interconnections also have the greatest outbound traffic.

Organizational network topology is used to build the overall network domain at the foundational security level. Nodes in

the database represent individual elements of the network infrastructure, including nodes for communication devices, security devices, servers, and user terminals. Nodes in the graph database include features like name, model, and other data about the device that do not allow it to connect with other nodes. Likewise, the device's open port, IP address, system software, and so on is all considered nodes because of their role in facilitating communication with other nodes. For instance, "port 5" on a switch might be abbreviated as SW5, where SW stands for "device" and "5" would be the port number. Using the connection in the graph database, these abstracted nodes are linked to device nodes. Furthermore, we add "topology" as a common term for these networks to demarcate the region in which the constrained nodes are placed, and the "connections" between the different device nodes are translated into "partnerships" in the database table.

IV. CONCLUSION

On this research, we show the feasibility of a near-actual-time log record evaluation in an NSM context making use of a graph database. But, there are some places where our recommended strategy might be bolstered. for example, our created device worked well enough to screen a network with forty five members, however its applicability to a larger, greater practical community hasn't been tested. as an example, the generated code must be stepped forward that allows you to display bigger networks. Inspecting numerous strategies for maintaining an eye on and analyzing log documents, in addition to putting records into the graph database almost right away.

From the angle of log report analysis, using a graph database allows any log records to be effortlessly So, for instance, similar records from many log files can be observed the usage of truthful searches, and a single, straightforward query may be used to discover the referrer order. This makes it simple to reconstruct fraudulent on line behavior, such as how a bunch computer become directed to a website that downloaded malware without the user's understanding (in a drive-through down load). Once this course changed into known, it changed into cleans to question the graph database to locate other hosts that have been related to the entire direction or as a minimum a portion of it. Moreover, this shortens the time needed to examine those eventualities.

While utilizing graph databases to examine log files gives up new possibilities to study new things approximately how log records correlates, there's a drawback to this method. We need to increase our efforts so that it will upload extra log files to the graph database. as an instance, Zeek generates a large wide variety of extra log files that are definitely of forensic and investigative relevance. Zeek's ssh.log, ssl.log, and documents.log are different examples (the latter log shops the

The overall performance of clustering several instances of Neo4j is probably in addition investigated as this work best used one instance of the database. This might be investigated directly in terms of the inquiry of the way to integrate a graph database that contains the output of several Zeek sensors which might be tracking numerous networks.

We conduct our experiment on query efficiency using data from Neo4j's official sandbox, a networking and IT administration environment. This data, on the one hand, is well-structured and hence easy to handle. However, because the amount of data is very little, it may be simple for us to make the request method and result in many databases identical. Because of this, we use this information in our query analysis comparison tests. It has a maximum of 68,122 vertices, 98,610 identifiers, 150,732 connections, and 121,098 characteristics. Each of the four sections of the data center has its own dedicated network and 10 storage cabinets. Various server configurations are housed in separate racks and linked to the switches through interfaces.

We use neo4j-shell-tools to export our personal Neo4j graph database's data to a CSV file after we've imported the networking and IT reporting systems from the Neo4j playground of GitHub. Each data set is exported as a "node-relationship-node" structure. Each node has its own unique identifier, label, and set of attributes. Connections between the two conjoined nodes reflect the relational type that underlies them. Using the LOAD DATA INFILE function, the CSV file is downloaded and then loaded into the MySQL database. As an added bonus, the Mongo DB's mongo import method brings it into the database.

REFERENCES

[1] Bejtlich, R.: The practice of network security monitoring: understanding incident detection and response. No Starch Press (2013).
[2] MIT Lincoln Laboratory. DARPA Intrusion Detection Evaluation. http://www.ll.mit.edu/ ideval/data/1999data.html. Accessed 4 June 2017
[3] National CyberWatch Center. MACCDC—Home of National CyberWatch Mid Atlantic CCDC (2017). https://www.maccdc.org. Accessed 27 July 2017
[4] Neise, P.: Intrusion Detection Through Relationship Analysis. SANS Institute InfoSec Reading Room (2016). https://www.sans.org/reading-room/whitepapers/detection/intrusion- detection-relationship-analysis-37352. Accessed 18 March 2017
[5] Neo4j. Neo4j, the world's leading graph database. https://neo4j.com/. Accessed 21 Aug 2017
[6] Netresec. PCAP files from the US National CyberWatch Mid-Atlantic Collegiate Cyber Defense Competition (MACCDC) (2017). https://www.netresec.com/?page=MACCDC. Accessed 20 Apr 2017
[7] Paxson, V.: Bro: a system for detecting network intruders in real-time. Comput. Netw. 31(23), 2435–2463 (1999)
[8] Py2neo. The py2neo v3 Handbook. http://py2neo.org/v3/. Accessed 11 Mar 2017
[9] Robinson, I., Webber, J., Eirfrem, E.: Graph Databases - New Opportunities for Connected Data, 2nd edn. O'Reilly Media Inc., Sebastpol (2015)
[10] Sanders, C., Smith, J.: Applied Network Security Monitoring: Collection, Detection, and Analysis. Elsevier (2013)
[11] Roesch, M.: Snort: lightweight intrusion detection for networks. In: Lisa, vol. 99, no. 1, pp. 229–238, November 1999
[12] Snort - Network Intrusion Detection & Prevention System. http://www.snort.org/. Accessed 21 Aug 2017
[13] Suricata. Suricata—Open Source IDS/IPS/NSM engine. https://suricata-ids.org/. Accessed 21 Aug 2017
[14] Zeek.org. The Zeek Network Security Monitor. https://www.bro.org. Accessed 15 Jan 2019
[15] Schindler, T.: Anomaly detection in log data using graph databases and machine learning to defend advanced persistent threats. In: Gesellschaft für Informatik e.V. (Hrsg.) Informatik 2017. Lecture Notes in Informatics (LNI). Gesellschaft für Informatik, Bonn (2017)
[16] Uetz, R., Benthin, L., Hemminghaus, C., Krebs, S., Yilmaz, T.: BREACH: a framework for the simulation of cyber attacks on company's networks. In: Digital Forensics Research Conference Europe (Poster) (2017)
[17] Djanali, S., et al.: Coro: graph-based automatic intrusion detection system signature generator for evoting protection. J. Theor. Appl. Inf. Technol. 81(3), 535–546 (2015)
[18] Diederichsen, L., Choo, K.K.R. and Le-Khac, N.A., 2019, December. A graph database-based approach to analyze network log files. In *International Conference on Network and System Security* (pp. 53-73). Springer, Cham.
[19] "Ansari, M. F., Sharma, P. K., & Dash, B. (2022). Prevention of phishing attacks using AI-based Cybersecurity Awareness

Training. International Journal of Smart Sensor and Adhoc Network., 61–72. https://doi.org/10.47893/ijssan.2022.1221

[20] Dash, B., Ansari, M. F., Sharma, P., & Swayamsiddha, S. (2022). FUTURE READY BANKING WITH SMART CONTRACTS - CBDC AND IMPACT ON THE INDIAN ECONOMY. International Journal of Network Security and Its Applications, 14(5). https://doi.org/10.5121/ijnsa.2022.14504

[21] Nagaraju, R.; C, V.; J, K.; G, M.; Goyal, S.B.; Verma, C.; Safirescu, C.O.; Mihaltan, T.C. Secure Routing-Based Energy Optimization for IoT Application with Heterogeneous Wireless Sensor Networks. Energies 2022, 15, 4777. https://doi.org/10.3390/en15134777

[22] Shankar, A., Sivakumar, N. R., Sivaram, M., Ambikapathy, A., Nguyen, T. K., & Dhasarathan, V. (2021). Increasing fault tolerance ability and network lifetime with clustered pollination in wireless sensor networks. Journal of Ambient Intelligence and Humanized Computing, 12(2), 2285-2298.

[23] Singh, A. V. (2018). Fundamental Concepts of Neural Networks and Deep Learning of Different Techniques to Classify the Handwritten Digits. In Advances in Systems, Control and Automation (pp. 287-297). Springer, Singapore.

[24] Jaiswal, P., Gupta, N. K., & Ambikapathy, A. (2018, October). Comparative study of various training algorithms of artificial neural network. In 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN) (pp. 1097-1101). IEEE.

[25] Fakih A.H., Venkatesh A.N., Vani, Naved M., Kshirsagar P.R., and Vijayakumar P., ""An efficient prediction of diabetes using artificial neural networks,"" in AIP Conference Proceedings, vol. 2393, pp. 20071, 2022. doi: 10.1063/5.0087948