International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988Volume 15 (2023) pp. 166-175

Received: 15 Dec.2022; Accepted: 19 April, 2022; Published: 9 June, 2023

An Intelligent Pattern Matching approach with Deep Hypersphere Model for Secure Big Data Storage in Cloud Environment

K R Remesh Babu[1], Saritha S[2], Preetha K G[3], Sangeetha U[4] and Sminu Izudheen[5]

[1]Government Engineering College, Idukki, Kerala, India,
remeshbabu@yahoo.com

[2,3,5]Rajagiri School of Engineering & Technology, Kochi, Kerala, India,
[2]*saritha_s@rajagiritech.edu.in*, [3]*preetha_kg@rajagiritech.edu.in*, [5]*sminu_i@rajagiritech.edu.in*

[4]Government Engineering College, Sreekrishnapuram, Palakkad,
Kerala, India
sangeethau2013@gmail.com

***Abstract*: Due to the rapid growth of volume, velocity, and diversity of data termed as big data, the traditional data storage systems are inadequate nowadays. There are lots of advancement in the area of storage, processing, and analysis of data. The data are mostly stored in cloud environment and distributed computing frameworks are used to access those data. The distributed data environment in the big data processing and cloud computing imposes many security and privacy challenges. Security measures applied to data storage, analytics and processing. Firewalls are not sufficient to identify content based at-tack. A more efficient algorithm, called as PRS is proposed in the paper, which uses pattern matching technology to identify the intrusion. The working of the algorithm is based on the input data evaluated by a centralized controller based on the previous attack history and attack feature data. The performance analysis of PRS algorithm is compared with the existing pattern matching algorithms such as KMP (Knuth-Morris-Pratt Algorithm) and BM (Boyer-Moore). The simulation result shows the proposed algorithm gives a better and quick attack pattern detection compared to the existing ones. After the PRS (Pattern Recognition for Security) algorithm has identified a matching pattern, the suspected pattern is fed into a deep hypersphere model, which produces a more accurate result for identifying attack patterns.**

***Keywords*:** Big data, Security, Cloud computing, Patten Recognition, Deep learning, Classification**.**

I. Introduction

A massive amount of data is produced every second in today's world, because of the advancement of technology. Of the huge amount of data produced, more than 60% of the same has been generated within last two years [1]. The term "Big Data" refers to the large volume of versatile heterogeneous information, along with its management and analysis. The traditional data processing technologies are inadequate to process Big Data due to it's massive size and heterogeneity. Big Data is differentiated from traditional data by 3Vs – volume (the amount of data), velocity (the rate of data generation and transmission), and variety (the types of structured and unstructured data) [2]. It is also characterized by Variability and Veracity, thus known as 5Vs in Big Data [2]. Checking embedded virus patterns in big data is a time-consuming operation due to the 5Vs of large data. The situation is again worsen by cybercriminals by packing and encrypting the at-tacking malware patterns into the data stream to cloud, escalating the arms race once again. The embedded and encrypted patterns known as attack payloads forced the anti-malware researches to develop more enhanced methods in order to discover malware patterns or behavior from this massive data.

The primary goal of big data processing is to make accurate, strategic decisions by examining the ever-increasing amount of data available, which involves data acquisition, extraction, integration, review, interpretation, and decision-making [3]. Data acquisition is the first and most important phase in big data processing, and it necessitates good data collection, filtering, and metadata generation strategies. Data transformation, normalization, aggregation, and error handling are all part of the data extraction process. During the integration process, data standardization, dispute resolution, and other tasks are completed. Data interpretation is a component of data analysis, which necessitates extensive domain awareness as well as a variety of analysis techniques. The final and most critical phase in big data processing is decision making, and perfect decision making requires excel-lent managerial skills. Because it impacts previously saved uninfected data, security is the primary concern. Affected data poses a risk in processing, storing, and making decisions based on it. As a result, an effective strategy is required to avoid data that has been damaged or data that has attack patterns incorporated in it.

Security and privacy are two of the most significant issues in big data. The majority of computing resources are housed

in the cloud, and services are delivered via the Internet, posing challenges such as data protection and access control for sensitive data. Information theft, DDoS attacks, ransomware, and other malicious actions area examples of big data attacks that can cause a system to crash [4]. When firms keep sensitive or secret information like credit card numbers or customer information, the implications of information theft can be significantly worse. Cybercriminals can tamper with data on endpoint devices and send the erroneous information to cloud storage. This highlights the importance of an intrusion detection system (IDS) in a cloud environment. The structure and format of the network data is different, hence gathering network data from many sources makes it challenging to analyse using traditional methods. An IDS should continuously tracks behavior in a system to decide whether or not they are part of an attack.

There are various IDS that exists for network monitoring and detecting attacks such as network IDS, host IDS, signature-based IDS, anomaly-based IDS, etc. [5]. In network IDS data packets sent over the network were captured and analyzed by comparing them to existing attack patterns (signature) in the database. If the patterns differ from the normal, these IDS will be useless. In host IDS, detection modules are installed in host IDS which is used to analyze the log files to detect intrusion. The main drawback of host IDS is it consumes large amount of re-sources, which may lead to host's performance; and attacks will not be detected until they have already reached the host. In anomalous IDS, intrusions are detected by monitoring the network for new attacks. The disadvantage of anomaly detection is that if traffic or activity deviates from the defined normal traffic pat-terns or activity, an alarm is issued. Signature IDS based on signatures which describe a collection of rules or signatures that are used to determine whether or not a given pattern is that of an intruder. However, in the era of the internet, mal-ware attacks can spread quickly before security experts can release the most up-to-date signatures to block them. Another issue is that as cyber-attacks got more regular, signature databases grew in size, possibly slowing system performance. So it is not advisable to new attacks. Due to its capacity to prevent attacks by malicious network users, a pattern matching intrusion detection system (IDS) has become a hot research topics.

The proposed paper evaluates various available IDS schemes Pattern matching is a technique for locating a pattern within a document. Brute force search algorithm, Rabin Karp algorithm, Knuth Morris Pratt (KMP) algorithm, Boyer Moore algorithm are some classic examples of pattern matching algorithms [6]. The biggest drawback of brute force strikes is that they take a long time to complete and it is only effective for a limited number of patterns. The Rabin Karp algorithm calculates the hash value for the pattern and the time complexity increases due to the spurious hit. In KMP algorithm when the alphabets get bigger, it doesn't work as well. As a result, more and more errors arise. The fundamental disadvantage of Boyer-Moore algorithms is the time and space required for pre-processing, which is dependent on the alphabet size and pattern size. Thus, it is evident that the existing algorithms have a gap in terms of time and space requirement, as well as, accurate classification, for using it in IDS. In this work, this particular research gap is addressed and a new pattern matching algorithm is demonstrated to identify intrusion with more accuracy and less latency, which improves the detection phase and thus enhances data security in cloud environment to a greater extent.

In this work, the problem addressed is to create an effective IDS for cloud environment using pattern matching technique. The work proposes a pattern matching algorithm called Pattern Recognition for Security (PRS). The algorithm is based on two stages, first, in which a centralized controller analyzes attack history patterns for a matching pattern and second, in which attack function data from previous attacks are matched for detection of the same using deep learning techniques. The highlighted novelty of the method is that the proposed model attempts single pattern matching algorithm for faster pattern recognition. In the second stage, the matched pattern is classified into an attack or a non-attack pattern using deep learning methodologies, in which significant functionalities as mentioned below are being added for effectiveness of the detection system. The IDS gains added performance and accuracy from the ANN combined method. The significant contributions of the work described are three-fold as mentioned.

- The proposed methodology uses a deep hypersphere model which aids in the learning of the relative relevance of attack pattern recognition, thus allowing the right attack pattern to be identified.
- A combination of "center" loss and "softmax" loss technique is used to enhance the discriminative power of the attack feature pattern for more accurate classification of the same.
- It is established through experimental evaluations, that the proposed PRS algorithm can be used in real time applications as it has low latency and high recall, which confirms its effectiveness.

The rest of this paper is organized as follows. The Section II briefs the related research reported in the relevant area. The proposed design and PRS algorithm is described in Section III. Section IV presents the experimental setup and validation of results and a conclusion is given in Section V.

II. Related Works

Advances in data diversity have ushered in a revolution in the field of big data and cloud computing over the last decade. Big data protection is extremely important and has always attracted the attention of researchers all over the world. The author in paper [7] gives an elaborated study on security issues and challenges related to big data. Detailed comparisons and the shortcomings of traditional security domains and technologies are also presented. The authors provide a practical solution for different security infrastructure components based on access control and identity management. Scientific Data Infrastructure (SDI) and Scientific Data Lifecycle Management (SDLM) models are discussed in [8]. Using current technology and best practices, these models can be used to create an interoperable data or project-centric SDI.

In [9] the authors discuss the firewalls and all types of intrusion detection systems that can be used in cloud environment. Paper [10] present a study of various cloud

storage systems for big data applications. A conceptual hybrid technique is proposed to improve the reliability and storage efficiency. In paper [11] the run time and size of network captured files were compared for brute-force, Rabin Karp, Boyer-Moore, and Knuth-Morris-Pratt. The string matching problems are analyzed in paper [12] and prove that the Rabin-Karp algorithm is more efficient than other pattern matching algorithms. A new linear search algorithm is pro-posed in [13]. In this algorithm, matching is a two-step process that is used to avoid unnecessary backtracking. The useless shift of patterns during matching is avoided by using a prefix function. The output of the prefix function is fed to the matching function and the function returns the location of the pattern within the input string. A matching algorithm is proposed in [14] which performs the same from the last character of the string instead of the first character. The proposed algorithm called sublinear algorithm depends on two tables, namely, delta1 and delta 2. Table delta1 contain an entry for each character in the alphabet. The rightmost possible reoccurrence of a terminal substring of a pattern is described in table delta2.

A multiple pattern matching algorithm is used to locate all occurrences of a finite number of keywords in the paper [15]. A finite state pattern matching machine is created for all keywords. Using this machine the patterns are searched in the given text in a single pass. The authors in the paper [16] discuss the need for an IDS in a distributed environment and also propose a distributed IDS model for intrusion detection. The proposed model in [16] captures and analyzes large flow of data, generate a report based on knowledge and also provides an analysis of legitimate and non-legitimate users' behaviour. A multi-threaded cloud IDS model is defined in paper [17], which is managed by a third party monitoring service for improved productivity and accountability for cloud users.

A two stage intrusion detection method is proposed using a combined behaviour-based and knowledge-based approach in [18]. If the first method failed to detect the attack, the second method captures and analyzes the data by comparing it to database signatures. To detect intrusion, the authors use the system's usual actions as well as various attack signatures and they also claim that the false positive rate is reduced appreciably. A comprehensive approach to monitor security events from many heterogeneous sources is described in [19]. The method in [19] also evaluates the issues existing in heterogeneous intrusion detection architectures, and Security Information and Event Management (SIEM) systems. In paper [20], a hierarchical IDS for Power Systems was proposed, which combines misuse detection and irregular detection. They use data mining algorithms to create detection rules from historical data processing and mining. This proposed model reliably detects cyber-attacks with a low rate of false positives and negatives. In [21], the authors focused on two important features of distributed storage: the capacity of distributed storage and information security in cloud computing. They also introduced Brewer's CAP Theorem and summarize current implementations of the popular NoSQL database. In [22], a real-time intrusion response systems to mitigate attacks that affect integrity, confidentiality, and availability in cloud computing platforms are analyzed. It also suggested self-awareness, self-optimization, and self-healing autonomic intrusion response strategy. Non-relational data is normally stored in NoSQL databases, but NoSQL databases do not provide any support to enhance security. Due to the tremendous advancements in the size of big data, an auto tiring solutions are used, which also do not keep track of the storage location. It allows you to search through a continuous data stream for exact matches to previously saved data sequences. The method in [23] use a neural network with input and output layers, as well as variable connections between them, at its core. Each possible character or number in the data stream is represented by one neuron in the input layer, and each stored pattern is represented by one neuron in the output layer. The network's unique feature is that the delays between the input and output layers are tailored to match the temporal occurrence of an input character within a stored sequence. In paper [24] a thorough examination of deep learning models used in cyber security tasks, such as intrusion detection is carried out by authors. In the intrusion detection domain, restricted Boltzmann machines, autoencoders, and recurrent neural networks are particularly popular, according to the authors. They also point out that evaluating the performance of different models is challenging be-cause academics utilize different datasets and measures to evaluate models. On two intrusion datasets (NSL-KDD and UNSW-NB15) [25], compare the performance of four deep learning models such as multi-layer perceptron (MLP), restricted Boltzmann machine (RBM), Sparse autoencoder (SAE), and MLP with feature embeddings. However, several prominent models, such as recurrent neural networks, are not included in their trials, and no evaluation of newer incursion datasets is done. The authors compare the performance based on accuracy, precision, and recall, and the study does not provide insight into the performance parameters such as latency, time, and memory requirement.

The review of literature section brings out clearly the requirement for a safe, reliable, and intelligent method to store and retrieve data in a cloud environment. The following session describes the features of the proposed method.

III. Proposed Methodology

While reviewing the literature, it was observed that several methods were put forward for improving the efficiency and security of big data storage mechanism but, none of them were optimal, thus necessitating the need of further research and development. Traditional security strategies targeted to private computing infrastructures, such as firewalls and demilitarized zones (DMZs), are no longer successful as Big Data expands with the aid of public clouds. As a result, an efficient security mechanism that operates through a heterogeneous set of hardware, operating systems, and network domains is required. More security issues arise as a result of the diversity of data, vast networks, large infrastructure and high-speed inter-cloud communication.

Data centres are the primary target area of attackers wherein SQL injection, scripting attacks etc., are the normal attacking methods. It is greatly appreciated that an effective IDS/IPS, as well as a firewall, be required to provide more protection to data in data centres. In this work, the attack patterns are identified in two stages. In the first stage, a fast pattern matching algorithm is used to detect a pattern. There are

various pattern matching techniques available in literature which uses different methods for pattern recognition. There are single as well as multiple pattern matching algorithms in the literature. Compared to single pattern matching algorithms, multiple pattern matching algorithms are more efficient in terms of pattern recognition, but it requires more memory space and time for prediction. Hence the proposed model attempts single pattern matching algorithm for faster pattern recognition. In the second stage, the matched pattern is classified into an attack or a non-attack pattern using deep learning methodologies.

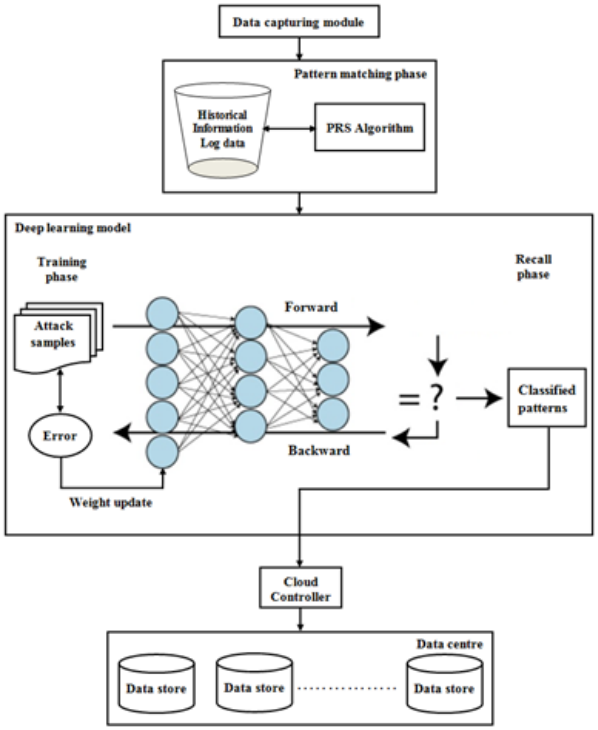

Figure 1. DeepPRS -Proposed Methodology

Figure 1 shows the architecture of the proposed system. The main components of the architecture includes data capturing module, pattern matching phase, deep learning hypersphere modelling phase and the cloud controller. When a user uploads a file, the data capturing module collects it and analyses the source address, user id, and other information before passing it on to the pattern matching phase. The pattern matching phase the most important component named as the PRS algorithm. The proposed new algorithm, PRS, along with the support of historical data, performs the pattern matching process in very less time. To prevent needless comparisons, the PRS algorithm starts by comparing the first four characters of each pattern with the uploaded file. If any match occurs, the hash value of text up to the duration of patterns of the matched characters is computed. Then compare the hash values of all patterns to the string hash value in the uploaded file. Patterns are stored as distinct files. KMP and Boyer Moore algorithms are used for pattern matching. Once the patterns are matched, they are passed to the deep hypersphere modelling phase, wherein it is fed as input to the Convolutional Neural Network. The matched patterns are also logged in the historical data file for future prospective matching to be done. The Convolutional Neural Networks has two stages (i) training and (ii) recalling. The output from the convolutional neural network classifies the pattern into either of the two kinds (a) attack or (b) non-attack. Depending upon the classified pattern, the cloud controller takes decisions regarding data storage to the cloud.

A. PRS Algorithm

The proposed two phase PRS algorithm is given in figure 2. Suppose T = {t_1, t_2, ... t_n} is the input string and P = {p_1, p_2, ... p_j}, is the pattern to be identified, subject to the conditions $j \leqslant n$, $n \geqslant 1$. Here both T and P are set of characters from a finite set of alphabets. The first four alphabets of each pattern are extracted and compared to the contents of the uploaded document in the first stage. The hash value H(T)of the text content upto the length of the pattern is computed.

If any match occurs, then, it reads characters from the substring to the length of the pattern. In the next step, it compares this value with the hash value of the patterns present in the cached historical data, H(P). If any similarity is found be-tween these patterns i.e., if H(T) is equal to H(P), a pattern matching is claimed and the algorithm proceeds to the next phase. The cached historical data is also updated with the occurrence count of the pattern, count_P. If the similarity is not detected, then the pattern is further matched with the logged historical data and compares H(T) with the hash value of the patterns present in the logged historical data, H(Q). After a match being detected, the attack logs is updated in historical data, and the occurrence count, count_Q is incremented. The count_Q is compared with Threshold_Count, and if it exceeds the same, the attack log is transferred to the cached historical data. If no matching hash values are detected, then the algorithm reports the same. In this scenario, the cloud controller has to make a decision of storing/not storing the data in the cloud, with the consideration of risk factors.

Correctness of the Algorithm

Let the pattern P with length 'h', be aligned with the text starting at position a_i of the String S. Assume that for a given i, the value of i is such that $0 \leqslant i \leqslant m$, where m is the maximum length of the string S, then minimum number of comparisons required is

$$a + h \leqslant m. \quad (1)$$

Since m is the maximum length of the string and the start position of the pattern is a, then a pattern of length h can be done maximum up to the length m of the string S. i.e., the sum of the start position of the pattern in the string and length of the pattern should always be less than or equal to the maximum of string length. E.g. if the string length is 10, pattern length is 4, and the start position of the pattern in the string is 5, then comparisons possible are 5 + 4 = 9, which is legal. Next comparison is 6 + 4 = 10, which is also possible. But the next comparison is 7^{th} position to next 4 positions, i.e., 7 + 4 = 11, which is impossible since the maximum string length is 10. So by adjusting the 'h' value we can speed up the comparison procedure of the algorithm. Henceforth, it is substantiated that the algorithm performs the pattern matching within an optimal time period.

Table 1. Proposed PRS Algorithm

Input: Array of input string T[1..n],historical data Q, array of Patterns P[1..m], Threshold_count
Output: Hashed value of matched pattern, H[S] and execution time

1. Read pattern_files P_1, P_2,…,P_m and input string T[1…n]
2. Compute hash value for each input pattern, H(P_1), H(P_2),… ,H(P_m)
3. while (i<m && j<n) do
   3.1 S ← First four characters of pattern
   3.2 if S is equal to T[j]..T[j+8], then compute hash
   value of matched text, H[S]
4. If H[T] is equal to H[P]
   4.1 Pass the matched pattern to *Deep_Model*
   4.2 Update cached historical data
   4.3 *Increment count_P*
   4.3 Break from the loop
   4.4 Set Flag
5. Else if H[T] is equal to H[Q]
   5.1 Pass the matched pattern to *Deep_Model*
   5.2 Update logged historical data
   5.3 *Increment count_Q*
   5.4 Set Flag
   5.5 If *count*_Q>*Threshold_Count*, then copy pattern Q to cache
   5.6 Break from the loop
6. Repeat steps 3 -6 for all patterns, P_1, P_2, …P_m
7. If (!Flag)
   7.1 No Matching Pattern Detected
   7.2Print msg
8. Stop

B. Deep Hypersphere for Identifying Attack/Non-Attack Patterns

In the first stage, the proposed model generates matching patterns.. The second stage identifies the attack patterns, and generates the appropriate intrusion response depending on the pattern type. The success of deep learning methodologies has motivated to bring in the same to identify the attack patterns. The general architecture of the Convolutional Neural Networks (CNN) is presented in Figure 2.

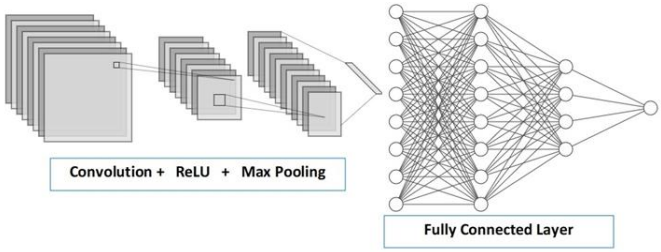

Figure 2. General Architecture of Convolutional NeuralNetwork

The network has an input layer, hidden layers, and an output layer. Every deep learning model has a training phase. In our proposed method, the training samples fed into the CNN are the attack definitions, stored in the logs. Attack definitions are stored in the form of a multidimensional feature model, thus accounting for the use of a deep learning model in the problem. The hidden layers in the network help in extracting the information from the attack definitions. The working of a CNN is briefed in this context from the perspective of the defined problem. A CNN generally consists of a convolution layer, ReLU layer, pooling layer, and a fully connected layer. The convolution layer houses many filters to perform the convolution operation by sliding through the extensive attack features to obtain the convolved feature matrix. The ReLU layer produces a rectified feature map of the attack definitions. Multiple convolutions and ReLU operations are needed for obtaining the feature map of the attack definitions with maximum information. A down sampling operation to reduce the dimension of the rectified feature map is done in the pooling layer with the aid of appropriate filtering techniques. The feature map is further flattened before being fed to the fully connected layer to get the output as either an attacking pattern or a non-attacking pattern. The cloud controller analyses the classified/labelled pattern and then takes the appropriate decision of transferring the data to the cloud, thus ensuring the security of the environment.

Training Phase

The training phase of the deep learning model is given in Figure 3. The attack patterns that had been encountered by the cloud environment are logged and saved as attack patterns. These attack patterns serve as the training samples in this phase. The training samples are fed into the CNN and undergo several convolutions, rectification, and max-pooling to obtain rectified feature vector, which is being fed into the fully connected layer. In the fully connected layer, the feature is propagated in the forward direction, the error in classification is back propagated and the weights are adjusted to minimize the loss.

Pioneering work in deep learning methodologies uses “softmax” loss functions [26] for discriminating classes. It is seen that the features learned by softmax loss functions are not discriminative enough to ensure accurate classification in the aforementioned problem. Hence, in the proposed model, a combination of “center” loss and “softmax” loss is used to enhance the discriminative power of the attack feature pattern for more accurate classification. This innovative part of our work, is substantiated with necessary proof, towards the end of this section. Once the loss is minimized, the network is considered to be stable and can be utilized for real time attack detection in the next phase, viz., recall phase.

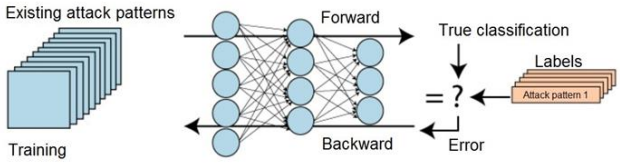

Figure 3. Training Phase of Deep Learning Model

Recall Phase

In the recall phase, the matched pattern is fed into the deep hypersphere model for classification purpose. The recall phase of the deep hypersphere model is given in Figure 4. The matched pattern from the PRS algorithm is fed into the hypersphere model, which is trained using the attack definitions. The trained model is more accurate as we are using a combination of “softmax” and “center” loss functions

in the model. The matched pattern from the PRS algorithm is the input pattern in the recall phase. The trained hypersphere model produces labelled patterns, viz., attack or non-attack kind.

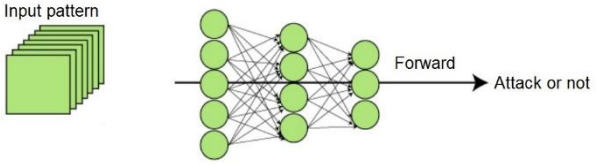

Figure. 4. Recall Phase of Deep Learning Model

Correctness of the deep hypersphere learning model

Consider the problem space as hypersphere decision boundary where Euclidean distance between the unknown pattern and a fixed point c (i.e., centre of the hypersphere) is considered. The aim is to classify whether the input data contains any kind of attacking pattern or not. So the proposed problem can be designed into a binary classification that contains m number of attack patterns as input.

Let f_p be the feature set of attack pattern, say, i and the hypersphere circumscribing data points having a radius *R*, then the objective functions is to minimize the error in classification, so that the attack patterns can be easily identified. The objective function Z can be represented as,

$$\text{Minimize } Z = R^2 + \frac{1}{e_n}\sum s_i \text{ where } R \subseteq f_p \text{ and R} > 0. \quad (2)$$

subject to:

$$||\psi_k(x_i) - c_k||^2 f_p \leq R^2 \text{ for all } i = 1, \ldots, n \quad (3)$$

where function ψ_k is the mapping function of data to a higher dimensional feature space f_p, such that a minimal volume hypersphere can be found.

But, there will be possibility of error in the prediction, and therefore, the equation (3) can be modified as

$$||\psi_k(x_i) - c_k||^2 f_p \leq R^2 + s_i \text{ where } s_i \geq 0, \forall_i \quad (4)$$

where s_i is the error correction factor for the hypersphere border line conditions and the e [0 - 1] is a control factor to regulate the error correction factor s_i and boundary of the sphere. Thus the data points that lie outside the hypersphere are outliers or non-attacking patterns. It is evident that the error correction factor controls the decision boundary of the hypersphere.

Thus this neural network maps input data stream $X \subseteq R^d$ to feature space $F \subseteq R^p$.

Let $W = \{w^1, \ldots, w^l\}$ is weight of the layers in the network, where $l = \{1, 2, \ldots\ldots, L\}$. Then the objective function can be modified for the deep learning network as

$$\text{Minimize } Z = R^2 + \frac{1}{e_n}\sum_{i=1}^{n} \max\{0, ||\psi(x_i; W) - c||^2 - R^2\} + \frac{1}{2}\sum_{l=1}^{l=L} ||W^l||_F^2 \quad (5)$$

As in the equation (2), the first term try to minimize *R* and the second term determines the boundary of the hypersphere. This error correction factor provides more accuracy for the classification. Finally, the last term is to control the weight W in the network layers. Thus by minimizing *R* in each layer the method tries to minimize the error in the classification of the input patterns into attacking or non-attacking kind. The experimental results presented in the next section will prove the efficiency of the proposed model.

IV. Results and Discussions

The proposed algorithm is implemented using Adobe Dreamweaver CS6 and Amazon database. The collected pattern is stored in MySQL database. The efficiency of the proposed PRS algorithm is compared against two well-known algorithms such as Boyer-Moore and KMP. PRS algorithm blocks the attack files to store in the database and the attack details are recorded for further analysis. The proposed pattern matching algorithm is evaluated using files containing 25000 records. The attack patterns are forcefully injected with different lengths 2, 4, 6, 8, 10, 20, 40, 80, 100, 200, 400, and 800 to generate test cases as shown in table 1. Later the attack patterns are randomly added to these record files and generated three more test cases, which are shown in the last three rows of Table 2.

It is evident from the results that the PRS algorithm outperforms the KMP and BM in terms of its quick response time. Consider the attack pattern of 2 bytes, compared to KMP, PRS gives 0.22461ms reduction in the inspection time. When the attack pattern increases to 800 bytes the time reduction is 0.43464ms. The performance analysis with randomly generated attack patterns in text files 1, 2, and 3 are 0.25355ms, 0.34266ms, and 0.2421ms respectively. These results show that PRS produces huge performance improvement compared to the traditional KMP algorithms.

While comparing with the BM, the performance improvement with 2 bytes, 800 bytes pattern attacks are 0.0507ms and 0.0814ms respectively. Similarly, comparison with random pattern attacks are 0.0481ms, 0.0869ms and 0.0518ms related to text files 1, 2 and 3. PRS algorithm thus proves its time efficiency to detect attack patterns compared to KMP and BM algorithms.

For the evaluation of the efficiency of the PRS algorithm, the latency is inspected for different data sizes. Comparing the performance of PRS in terms of latency against the KMP and BM for 5000, 10000, 20000, and 25000 size data files, it is interesting to note that the inspection delay is less than 0.3, 0.6, 0.9, and, 2.1 milliseconds respectively than KMP. The PRS takes less than 0.60ms to inspect data than BM because the length of hash value as mentioned in the pro-posed design is a significant factor in reducing the inspection delay. The latency is noticeably low in the case of PRS as against the other two algorithms, thereby confirming the increased credence of the proposed PRS as given in Figure 5.

A. Deep Hypersphere Model

One of the hyperparameters used to control the model's performance is learning rate. It determines how quickly the network updates the information it has gathered. The outcomes of experimenting with various learning rate settings are summarized in Table 3. The learning rate of 0.001, which is also the default rate for the Adam optimizer employed by the model, yields the best results. Figure 6 shows the model's training and validation loss when the learning rate is set to 0.001.

Table 2. Comparison of proposed algorithm PRS with KMP and BM algorithms in terms of execution time with varying file size which contains attack pattern.

Pattern Size (Bytes)	Average Pattern Inspection Time in milliseconds											
	5000			10000			20000			25000		
	KMP	BM	PRS	KMP	BM	PRS	KMP	BM	PRS	KMP	BM	PRS
2	0.22861	0.0547	0.004	0.61275	0.3848	0.008	0.92009	0.5635	0.0125	2.13249	1.2773	0.0279
4	0.24477	0.064	0.0042	0.75125	0.4602	0.008	0.87875	0.5365	0.0122	2.16297	1.3151	0.0271
6	0.24673	0.0621	0.004	0.63548	0.4165	0.007	0.81142	0.5854	0.0123	2.09309	1.3004	0.0319
8	0.42209	0.0817	0.0057	0.72216	0.5144	0.0078	0.97336	0.5047	0.0122	2.19378	1.3338	0.0354
10	0.41526	0.0842	0.007	0.62497	0.379	0.0082	0.98676	0.4976	0.0148	2.13336	1.261	0.0245
20	0.26945	0.053	0.0039	0.63258	0.5357	0.0091	0.97001	0.5037	0.0138	2.17317	1.3159	0.0307
40	0.3028	0.0637	0.0045	0.64584	0.452	0.0105	1.00227	0.6701	0.0125	2.15637	1.2498	0.032
80	0.22507	0.0639	0.0039	0.65453	0.4848	0.0113	0.87653	0.5109	0.0122	1.98845	1.2708	0.0329
100	0.38114	0.0862	0.0053	0.60076	0.4451	0.0065	0.85145	0.5016	0.0124	2.06707	1.2742	0.0319
200	0.24565	0.0611	0.0071	0.56595	0.3723	0.0066	0.91966	0.4969	0.0122	2.04767	1.2687	0.0317
400	0.24853	0.067	0.004	0.63258	0.5117	0.0083	0.9166	0.5349	0.0122	2.0807	1.2683	0.0311
800	0.44094	0.0877	0.0063	0.60917	0.4945	0.0116	0.80299	0.5717	0.0124	2.10773	1.2345	0.0305
Text file 1	0.26125	0.0558	0.0077	0.56275	0.3372	0.0059	0.92113	0.5519	0.0142	2.03222	1.3154	0.032
Text file 2	0.34986	0.0941	0.0072	0.63136	0.4506	0.0079	0.92948	0.6157	0.0158	2.16953	1.193	0.0326
Text file 3	0.2462	0.0559	0.0041	0.54309	0.3704	0.0063	0.97326	0.6305	0.0146	2.07895	1.2524	0.028

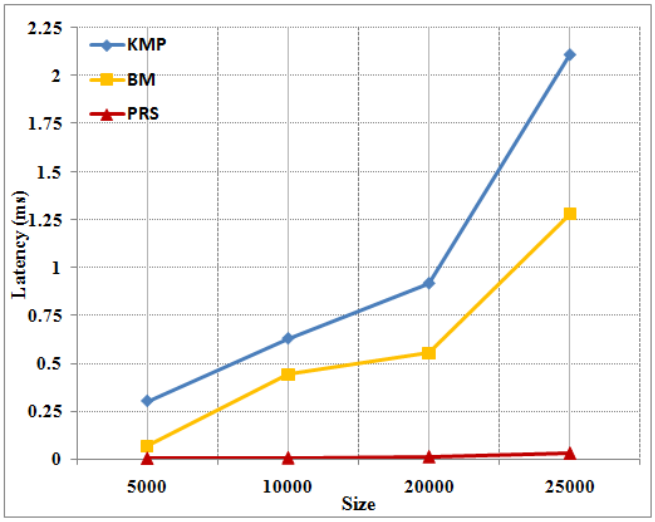

Figure 5. Comparison of PRS algorithm with KMP and BM for proving latency time

Table 3. Effect of learning rate in model accuracy

Dropout Rate	RMSE
0.0000001	0.098543520
0.00001	0.097820940
0.0001	0.096456226
0.001	0.094187275
0.01	0.099730260
0.1	0.123614386

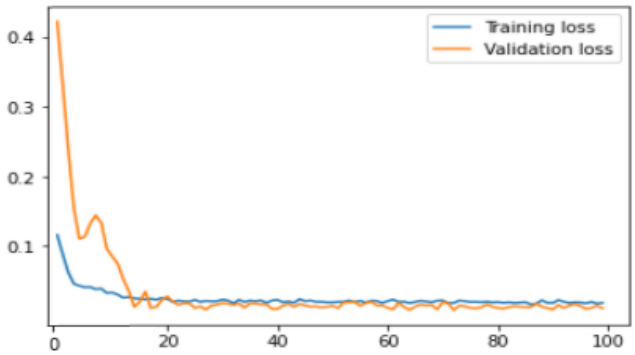

Figure 6. Training and validation loss with learning rate = 0.001

In order to counter over fitting and improve regularization error, dropout nodes are introduced. Dropout nodes are used to counteract over fitting and improve regularization error. This is an extra hyperparameter that specifies the chance of the layer's outputs being dropped out. Table 4 summarizes the model's impact for various values of this hyerparameter. It is noticed that a dropout rate of 0.1 results in higher performance. Figure 7 shows the training and validation losses for a model with a dropout rate of 0.1.

Table 4. Effect of dropout rate in model accuracy

Dropout Rate	RMSE
0.5	0.10334229
0.3	0.09148774
0.2	0.10313607
0.1	0.08498888
0.05	0.10534138
0.01	0.09218325

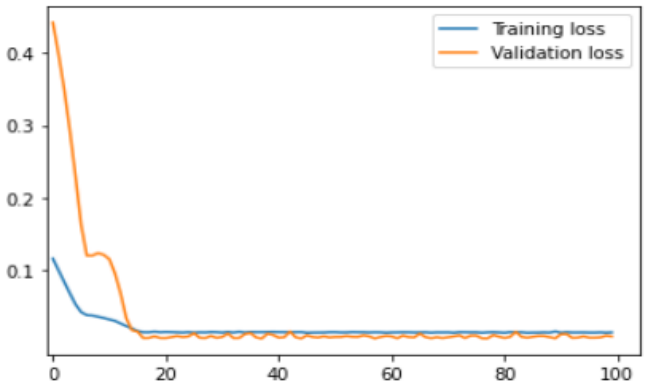

Figure 7. Training and validation loss with dropout rate = 0.001

The output represented in Figure 8 clearly shows that the proposed deep hypersphere model outperforms the existing feed forward [27], Jordan's [28], Elman's [29], and fully connected [30] recurrent neural networks. The error rates for the training set for feed forward, Jordan's, Elman's, and fully connected networks are 0.85, 0.65, 0.625, and 0.575, respectively, whereas the suggested technique has just 0.472.During the testing phase, the above approach had error rates of 3.7, 3.1, 2.9, and 2.7. However, the proposed model only reveals a 1.3% error rate. After tuning the hyperparameters the error rate convergence of the proposed deep hypersphere model is represented in Figure 9. The error rate is steadily falling, indicating that the network has converged and that the weights have been trained to their ideal value. As a result, the network's output bias is minimal.

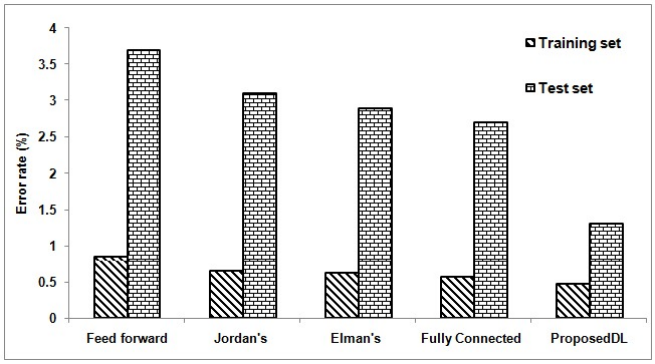

Figure 8. Comparison of proposed Hypersphere model with existing Neural Network models in respect of error rate during training and testing

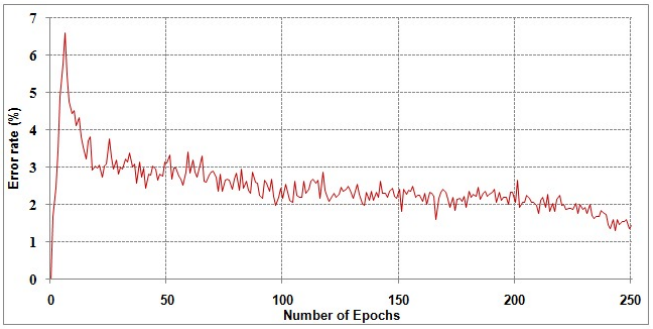

Figure 9. Error rate Convergence in proposed Deep Hypersphere Model

V. Conclusions

The paper proposed an algorithm PRS for improving the efficiency and security of data storage in the cloud environment. The PRS algorithm detects attacks by matching patterns, and if an attack is discovered, the suspected pattern is put into the newly built Deep hypersphere model, which confirms the attack. When an attack is detected, the model immediately ceases storing data in the cloud storage area, thereby reducing the risk. The performance of the PRS algorithm is evaluated in respect of execution time for attack pattern recognition. The results are compared against the existing Knuth-Morris-Pratt Algorithm (KMP) and Boyer-Moore (BM) Algorithms. The experiment is carried out for varying file sizes and the average execution time is taken for comparison. The result shows the pronounced improvement in the performance parameter of the proposed algorithm PRS.

The proposed deep hypersphere model recognizes attack patterns and generates the appropriate intrusion response based on the type of pattern. The new approach has precision values ranging from 96.21% to 99.98%. Similarly, recall ranges from 99.59% to 99.91% percent, and the approach achieved F1 score of 0.9974 during the test phase. The proposed model for security-based intelligent storage proved to be a successful way for quickly and efficiently identifying intrusions in the big data processing.

References

[1] Dias, Luis Filipe, and Miguel Correia. "Big data analytics for intrusion detection: an overview." Handbook of Research on Machine and Deep Learning Applications for Cyber Security (2020): 292-316.

[2] Kumari, Aparna, Sudeep Tanwar, Sudhanshu Tyagi, and Neeraj Kumar. "Verification and validation techniques for streaming big data analytics in internet of things environment." IET Networks 8, no. 3 (2018): 155-163.

[3] Du, Miao, Kun Wang, Yuanfang Chen, Xiaoyan Wang, and Yanfei Sun. "Big data privacy preserving in multi-access edge computing for heterogeneous Internet of Things." IEEE Communications Magazine 56, no. 8 (2018): 62-67.

[4] Smys, S., Abul Basar, and Haoxiang Wang. "Hybrid intrusion detection system for in-ternet of Things (IoT)." Journal of ISMAC 2, no. 04 (2020): 190-199.

[5] Du, Miao, Kun Wang, Yuanfang Chen, Xiaoyan Wang, and Yanfei Sun. "Big data privacy preserving in multi-access edge computing for heterogeneous Internet of Things." IEEE Communications Magazine 56, no. 8 (2018): 62-67.

[6] Xian-feng, Hou, Yan Yu-bao, and Xia Lu. "Hybrid pattern-matching algorithm based on BM-KMP algorithm." In 2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), vol. 5, pp. V5-310. IEEE, 2010.

[7] Demchenko, Yuri, Canh Ngo, Cees de Laat, Peter Membrey, and Daniil Gordijenko. "Big security for big data: Addressing security challenges for the big data infrastructure." In Workshop on secure data management, pp. 76-94. Springer, Cham, 2013.

[8] Demchenko, Yuri, Zhiming Zhao, Paola Grosso, Adianto Wibisono, and Cees De Laat. "Addressing big data challenges for scientific data infrastructure." In 4th IEEE

International Conference on Cloud Computing Technology and Science Proceedings, pp. 614-617. IEEE, 2012.
[9] Archana D Wankhade, P N Chatur, Comparison of Firewall and Intrusion Detection System, International Journal of Computer Science and Information Technologie(IJCSIT), Vol. 5, 2014, pp. 674-678.
[10] Nachiappan, Rekha, Bahman Javadi, Rodrigo N. Calheiros, and Kenan M. Matawie. "Cloud storage reliability for big data applications: A state of the art survey." Journal of Network and Computer Applications 97 (2017): 35-47.
[11] Gupta, Vibha, Maninder Singh, and Vinod K. Bhalla. "Pattern matching algorithms for intrusion detection and prevention system: A comparative analysis." In 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 50-54. IEEE, 2014.
[12] KARP, RM. "Efficient randomized pattern-matching algorithms, The IBM Journal of Research and Development." http://www. research. ibm. com/journal/rd/312/ibmrd3102P. pdf 31 (1987).
[13] Hakak, Saqib Iqbal, Amirrudin Kamsin, Palaiahnakote Shivakumara, Gulshan Amin Gilkar, Wazir Zada Khan, and Muhammad Imran. "Exact string matching algorithms: Survey, issues, and future research directions." IEEE Access 7 (2019): 69614-69637.
[14] Namjoshi, Kedar, and Girija Narlikar. "Robust and fast pattern matching for intrusion detection." In 2010 Proceedings IEEE INFOCOM, pp. 1-9. IEEE, 2010.
[15] Yang, Zhenglu, Jianjun Yu, and Masaru Kitsuregawa. "Fast algorithms for top-k approximate string matching." In Proceedings of the AAAI Conference on Artificial Intel-ligence, vol. 24, no. 1. 2010.
[16] Shelke, Ms Parag K., Ms Sneha Sontakke, and A. D. Gawande. "Intrusion detection sys-tem for cloud computing." International Journal of Scientific & Technology Re-search 1, no. 4 (2012): 67-71.
[17] M.Madhavi, An Approach For Intrusion Detection System In Cloud Computing, International Journal of Computer Science and Information Technologies (IJCSIT), Vol. 3 (5) , 2012.
[18] S.V. Narwane, S. L. Vaikol, Intrusion Detection System in Cloud Computing Environment, International Conference on Advances in Communication and Computing Technologies (ICACACT), Proceedings published by International Journal of Computer Applications (IJCA), 2012 .
[19] Richard Zuech, Taghi M Khoshgoftaar, Randall Wald, Intrusion detection and Big Heterogeneous Data: a Survey, Journal of Big Data, a Springer open Journal, V.3, pp. 2-41, 2015
[20] Zeng, An Intrusion Detection System Based on Big Data for Power System, International Symposium on Advances in Electrical, Electronics and Computer Engineering (ISAEECE), pp. 322-328, 2016.
[21] C.W. Hsu, C.W. Wang, Shiuhpyng Shieh, Reliability and Security of Large Scale Data Storage in Cloud Computing, part of the Reliability Society Annual Technical Report 2010.
[22] Kleber Vieira, Fernando Schubert, Guilherme Arthur Geronimo, Carlos Becker Westphall, Autonomic Intrusion Detection System in Cloud Computing with Big Data, International Conference on Security and Management(SAM),pp.173-178,2014.
[23] Hoffmann, Heiko, Michael D. Howard, and Michael J. Daily. "Fast pattern matching with time-delay neural networks." In The 2011 International Joint Conference on Neural Networks, pp. 2424-2429. IEEE, 2011.
[24] Berman, Daniel S., Anna L. Buczak, Jeffrey S. Chavis, and Cherita L. Corbett. "A sur-vey of deep learning methods for cyber security." Information 10, no. 4 (2019): 122.
[25] Yan, Jiaqi, Dong Jin, Cheol Won Lee, and Ping Liu. "A comparative study of off-line deep learning based network intrusion detection." In 2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN), pp. 299-304. IEEE, 2018.
[26] Liu, Weiyang, Yandong Wen, Zhiding Yu, and Meng Yang. "Large-margin softmax loss for convolutional neural networks." In ICML, vol. 2, no. 3, p. 7. 2016.
[27] Victor H. Benitez, Pattern Classification and Its Applications to Control of Biomechatronic Systems, Artificial Neural Networks for Engineering Applications, Academic Press, 2019, pp. 139-154.
[28] Michael I. Jordan. Serial order: A parallel distributed processing approach. Technical Report 8604, Institute for Cognitive Science, University of California, San Diego, 1986.
[29] Hee-Heon Song, Sun-Mee Kang and Seong-Whan Lee, "A new recurrent neural net-work architecture for pattern recognition," Proceedings of 13th International Conference on Pattern Recognition, 1996, pp. 718-722 vol.4, doi: 10.1109/ICPR.1996.547658.
[30] Hongzhi Li, Joseph G. Ellis, Lei Zhang and Shih-Fu Chang, PatternNet: Visual Pattern Mining with Deep Neural Network. In ICMR '18: 2018 International Conference on Multimedia Retrieval, June 11–14, 2018, Yokohama, Japan.
[31] Remesh Babu, K.R., Saritha, S., Preetha, K.G., Unnikrishnan, S., Izudheen, S. (2023). DeepPRS: A Deep Learning Integrated Pattern Recognition Methodology for Secure Data in Cloud Environment. In: Abraham, A., Bajaj, A., Gandhi, N., Madureira, A.M., Kahraman, C. (eds) Innovations in Bio-Inspired Computing and Applications. IBICA 2022. Lecture Notes in Networks and Systems, vol 649. Springer, Cham. https://doi.org/10.1007/978-3-031-27499-2_20.

Author Biographies

K R Remesh Babu received PhD Degree from Cochin University of Science and Technology (CUSAT) in 2019. He also holds bachelor degree in Mathematics, Information Technology and master's degree in Computer Science. Currently he is working as Professor in the department of Information Technology, Government Engineering College, Idukki, India. He is author of several publications in the leading international journals and conferences. He is interested in Distributed and Cloud Computing, Internet of Things, Wireless Sensor Networks, Machine Learning, and Big Data Analytics.

Saritha S is a highly accomplished Computer Science engineer with extensive experience in research and academia. She earned her PhD in Computer Science from Cochin University of Science and Technology, India and has made significant contributions to the field through her research, publications, and patents. She is currently working as an Associate Professor in the Department of

Computer Science in Rajagiri School of Engineering & Technology (Autonomous), Kochi, Kerala, India. Her expertise lies in areas like Spatiotemporal Data Mining, Cloud Computing and Internet of Things where she has published numerous research papers in top-tier academic journals and conferences.

Preetha K G has completed her PhD in Mobile Ad hoc Networks from Cochin University of Science and Technology in 2018. She has completed her M Tech and B Tech Degree in Computer Science and Engineering. She has been associated with Rajagiri School of Engineering & Technology since 2004 and is now working as Professor and Head in the Department of Computer Science & Engineering. She has around 20 years of academic experience. Her research interests include Mobile Computing, Wireless Networks, Ad-hoc Networks, Data Analytics etc.

Sangeetha U received B Tech degree in Information Technology from the Cochin University of Science and Technology (CUSAT), Kochi, India and MTech degree in Computer Science and Engineering from NIT Trichy. She received PhD in from NIT Kozhikode in 2022. Currently she is working as Associate Professor and Head in the department of information technology at Government Engineering College, Palakkad, India. Her research interest includes internet of things, wireless communication, computer networks and cloud computing..

Sminu Izudheen is a Professor in the Department of Computer Science & Engineering at Rajagiri School of Engineering & Technology, Kakkanad, Kochi. She was awarded Ph.D. in Data Mining in Bioinformatics from Cochin University of Science and Technology. She has around twenty years of academic experience in Rajagiri School of Engineering & Technology. Her research interests include Bioinformatics, Data Analytics and Automata Theory. She has numerous publications in various journals and conferences. She also has an Indian National Patent to her credit.