Received 4 July 2023, accepted 6 August 2023, date of publication 15 August 2023, date of current version 24 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3305506

RESEARCH ARTICLE

Confimizer: A Novel Algorithm to Optimize Cloud Resource by Confidentiality-Cost Trade-Off Using BiLSTM Network

SANDESH ACHAR[1], NURUZZAMAN FARUQUI[2], ANUSHA BODEPUDI[3], AND MANJUNATH REDDY[4]

[1]Department of Software Engineering, Walmart Global Tech, Sunnyvale, CA 94086, USA
[2]Department of Software Engineering, Daffodil International University, Daffodil Smart City, Birulia, Dhaka 1216, Bangladesh
[3]Workday Inc., Pleasanton, TX 94588, USA
[4]Biometric Ultrasound Fingerprint Technology, Qualcomm Inc., San Diego, CA 92121, USA

Corresponding author: Sandesh Achar (sandeshachar26@gmail.com)

ABSTRACT The world is expiring a 23% annual data growth rate and is projected to have a total surplus volume of 175 Zettabytes by 2025. It imposes significant challenges for small to medium-sized businesses to allocate funds for large-size data storage. The initial large upfront and maintenance costs have made cloud storage services popular. It comes with confidentiality concerns. Encrypting data before storing it in cloud storage is the most effective solution to this challenge. Encrypting and decrypting large volumes of data allocate massive amounts of expensive resources. Storing in plain text reduces system load and expenditure but introduces confidentiality concerns. This paper proposed a Confimizer, a novel algorithm, to optimize cloud resources and reduce costs by balancing the trade-off between confidentiality and cost. It reduces the system overload by 13.75%, saving 9.20% expenditure. It saves 12.33% storage and reduces API calls by 52.99%. The Confimizer uses an optimized BiLSTM network that classifies data according to the confidentiality level by 84.00% accuracy, 76.92% precision, 74.47% recall, and 75.01 F1 score. The innovative approach, optimized BiLSTM network architecture, and outstanding performance of the Confimizer make it a unique and effective cloud resource optimization algorithm.

INDEX TERMS Cloud computing, resource optimization, data confidentiality, optimization, deep learning.

I. INTRODUCTION

The vision of having computing as a utility has come to fruition through the power of cloud computing [1]. This technology has made a significant impact on various facets of the digital services that we utilize today. Small and medium-sized enterprises (SMEs), in particular, stand among the top business-level consumers of cloud services. This technology enables them to conduct their operations without necessitating a substantial investment in large-scale Information Technology (IT) infrastructure [2]. Maintaining and operating hardware demands a considerable initial investment, continued maintenance, and a team of dedicated experts [3]. The fundamental service models of cloud computing–Software-as-a-Service (SaaS), Infrastructure-as-a-Service (IaaS), and Platform-as-a-Service (PaaS) [4]–allow users to select the model that best suits their needs. However, there are inherent limitations in the business models based on cloud services. These limitations include concerns related to data confidentiality and the potential for escalating costs due to intensive usage [5]. This paper outlines the potential for overcoming these challenges within the existing scope.

Encryption is an effective and easier solution to data confidentiality concerns. This concern is also known as data privacy or data security [6]. Popular cloud service providers, including Google App Engine and Amazon, have on-demand data encryption services [7]. They use Security-as-Service architecture. Provisioning these services to ensure confidentiality has become simpler through API integration [8]. These services are advanced enough to maintain data integrity and

The associate editor coordinating the review of this manuscript and approving it for publication was Michael Lyu.

confidentiality. However, the challenge is discovering the scope of cost minimization by encrypting confidential data only. The pay-as-you-go payment method of cloud service is efficient because of its scalability [9]. The expenditure remains minimum when the usage is minimum. Encrypting every data the users create increases the expenditure by requiring higher storage, making more API calls, and using more cloud resources. Storing data without encryption leads to data confidentiality concerns [10]. This paper proposes a novel solution that minimizes the Request for Encryption Service (RES) without raising any risk to confidentiality.

A BiLSTM network has been designed, implemented, and optimized in this paper to classify the level of confidentiality required for the data over the cloud applications [11]. The proposed network classifies the user data into four categories labeled (i) High, (ii) Average, (iii) Below Average, and (iv) Low. The classification received from the network is used in the Confimizer algorithm to encrypt the data, which requires confidentiality only. This simple yet innovative idea, a novel algorithm, and effective implementation ensures optimal use of cloud resource and minimize the cost. The core contributions of this research have been listed below:

- Development of the novel Confimizer algorithm to minimize cost by maintaining a balance between confidentiality and cost, which is a unique approach to optimization.
- Design, implementation, and optimization of a BiLSTM network to predict from a dataset of variable length of string with 84.00% accuracy, 76.92% precision, and 74.57% recall.
- Reducing the annual cost for cloud resources by 9.20% and using 12.33% less storage than usual.
- Achieving an average of 52.99% fewer API calls contributes to cost and cloud resource usage reduction.
- Reducing the computational load on a cloud server by 13.75% on average.

The rest of the paper has been organized into five sections. The literature review has been presented in section two. The proposed methodology has been developed and the third section. The fourth highlight the experimental results and presents an in-depth analysis of the findings. The limitation and future scope of the proposed algorithm are in section four. Finally, the paper has been concluded in section five.

II. LITERATURE REVIEW

Confidentiality-based data classification for cloud computing models is a potential research field with the potential of numerous applications, including resource optimization [12]. Ali et al. [13] has developed a confidentiality-based cloud computing model called Classification-as-a-Service (C2aaS) to tackle the security concern related to cloud storage and preventing system overload. Another study conducted by Zardari et al. [14] developed a KNN-based classifier to classify data into confidential and non-confidential before storing them on cloud storage. Their approach used encrypts the confidential data only to develop an efficient and memory space-friendly cloud computing scheme. The proposed Confimizer is a cloud resource optimization algorithm that agrees with the hypotheses of Ali et al. [13] and Zardari et al. [14]. However, the Confimizer uses the BiLSTM network to classify the data into four classes, whereas Zardari et al. [14] used a KNN classifier limited to binary classification only. The proposed methodology uses 128-bit and 256-bit encryption depending on the confidentiality level, making it robust and more efficient.

A survey by Butt et al. [15] highlights the state-of-the-art cloud security solutions. Most of the research has been observed to focus on data confidentiality in the cloud, ignoring the additional cloud resources and associated costs with it [16], [17], [18]. Encrypting data requires additional computational resources, including more storage, higher bandwidth for transmission, and more processing time than plane text [19]. That is why encrypting everything is not an efficient solution for cloud confidentiality. The proposed Confimizer has been developed from this observation. It ensures confidentiality by encrypting confidential data while saving significant cloud resources by leaving non-confidential data as plain text. The information security and associated technical issues for cloud storage have been studied by Hui et al. [20]. The proposed Confimizer addresses the problems of optimizing the cloud resources while ensuring essential confidentiality.

A hybrid AES-ECC cryptography method proposed by Kumar et al. [21], a combination of orthogonal knowledge swarm optimization with oblique cryptography proposed by Reddy et al. [22], enhanced encryption scheme applicable for lightweight data intended for multi-cloud developed by Raj et al. [23], and single pre-shared key-based homomorphic encryption scheme developed by Kara et al. [24] are some effective methodologies to ensure confidentiality in cloud storage. While these approaches deserve appreciation for methodological innovation and effective performance, it comes with an optimizable computational cost. The proposed Confimizer studies cloud storage data confidentiality from a resource optimization perspective, which is absent in the state-of-the-art research reviewed in this section. The proposed methodology maximizes the probability of achieving an optimal confidentiality-cost trade-off using the proposed novel Confimizer algorithm and BiLSTM network.

III. METHODOLOGY

The proposed Confimizer is a Natural Language Processing (NLP)-based approach. It starts with preparing and processing the dataset consisting of varying text lengths. A vector is formed from the textual dataset. The vectorized words are used in an exclusively designed BiLSTM network. This network predicts the level of confidentiality. According to the level of confidentiality, the Confimizer algorithm decides on the applicability and type of encryption.

A. DATASET & PRE-PROCESSING

The proposed Confimizer is a unique concept, and a related dataset has not been found. Several research projects published in recent literature focused on the same domain. However, the dataset used in those papers has not been made public [25], [26], [27]. Because of having confidential data, it goes against engineering ethics to disclose the dataset [28].

Dataset Background: This paper uses a dataset obtained from a medium-scale marketing agency with 154 employees and more than 92 active clients. This company handles both digital and traditional marketing for clients. The clients share both regular and confidential information with the agency. According to the terms and conditions of the agency, it is bound to ensure the integrity and confidentiality of clients' data. They use local storage devices and encrypt everything the clients share. It has exceeded the agency's financial capability to add more storage units to tackle the high volume of data.

Moving to cloud storage provisioned through a pay-as-you-go payment scheme is the instant and primary solution to the agency's problem. However, it goes against the confidentiality and integrity agreement. Encrypting all data before storing it on the cloud storage is a solution to this problem. However, it costs additional encryption and decryption service. Moreover, the volume of the data increases after encryption which continuously accrues additional costs. It has been observed that the agency receives sensitive and non-sensitive data. Encrypting only the sensitive data and storing non-sensitive data as they are is an effective solution to reduce the annual cloud service subscription cost. A dataset has been created by removing the affiliation of the clients to research to reduce the storage cost while maintaining confidentiality. This dataset has been used in this paper with the agreement of research purpose application only with the consent of non-disclosure.

Dataset Description: The dataset is a collection of strings with varying lengths defined by equation 1 where D_t is the dataset, $S_i(l)$ is the i^{th} string of length l. A subset of this dataset has been presented in table 1. There are 6450 instances in this dataset. Each instance has a unique ID generated from the individual client ID. This string contains different types of information, including confidential and non-confidential information. Depending on the level of confidentiality, the strings have been labeled with four classes defined by equation 2. These four classes are (1) Low (L), (2) Below Average (B), (3) Average (A), and (4) High (H).

$$D_t = \sum_{i=1}^{m} S_i(l) where l = \{x | x \in N\} \tag{1}$$

$$C_i = \{L, B, A, H\} where l = \{y | y \in N, x \leq 4\} \tag{2}$$

1) DATASET PROCESSING

a: TEXT CLEANING

The dataset D_t is a collection of characters defined by equation 3, including irrelevant characters (i_c), punctuation (p_c), and symbols that create emojis (e_s) which is a set expressed by V. Removing elements defined by equation 4 is text cleaning. The text cleaning function follows the working principle expressed in equation 5 where n represents the cleaned character.

TABLE 1. A subset of the dataset that shows one instance from every label.

ID	Data	Label
1	Hi, thank you for taking the time to meet with us today.	Low
2	We've been facing some issues with our product's market positioning. We believe it's not reaching the right audience.	Below Average
3	We appreciate that. Here's the issue - our main product is innovative and targets young professionals, but we've been getting a lot of traction from an older demographic. While this isn't a bad thing, it's just not our target market, and we believe our product will do better with the intended demographic	Average
4	Yes, but remember this is confidential - our sales from the 45+ demographic have been twice as much as those from the 25-35 demographic. But our research shows the younger demographic would benefit more from our product and has a bigger market size. We think our marketing message might be off.	High

$$c_i = \{c_1, c_2, c_3, \ldots, c_n\} \tag{3}$$

$$V = \{i_c, p_c, e_s\} \tag{4}$$

$$F(c_i) = \begin{cases} '' & \text{if } c_i \in V \\ c_i & \text{otherwise} \end{cases} \tag{5}$$

This paper considers the words containing less meaningful information as Stop-Words (SWs). Removing the SWs is a part of the text cleaning. The SWs have not been defined in this paper. Instead, the existing stop-words listed in Natural Language Toolkit (NLTK), a popular NLP library, have been used [29].

b: TEXT NORMALIZATION

The text normalization involves converting every character into lowercase letters designed using the function defined in equation 6. Here, $L()$ is the function, and c_i is the i^{th} input character. This function replaces the capital letters with corresponding small letters.

$$L(c_i) = \begin{cases} \text{lowercase of } c_i & \text{if } c_i \text{ is an uppercase letter} \\ c_i & \text{otherwise} \end{cases} \tag{6}$$

c: TOKENIZATION

The strings are split into valid words to use for Natural Language Processing (NLP). The dataset used in this paper is a collection of strings. A set of strings is defined as a collection of characters. Training a system to predict the level of confidentiality requires feature extraction from words instead of characters. The string expressed by s is converted into a set of tokens that represent tokens using the tokenization function defined in equation 7

$$T(S) = (t_1, t_2, \ldots, t_m) \tag{7}$$

In equation 7, T is the tokenization function, S is the sentence to be tokenized, and $(t_1, t_2, \ldots, t_m)$ is the sequence of tokens resulting from the tokenization of S.

d: Word2Vec FORMATION

After performing Stemming and Part of Speech (POS) Tagging [30], the words are converted into vectors. The process uses a neural network with a vocabulary size of V, the size of hidden layer N, the input word is w, and the context word is c. The input layer and hidden layer weights are W and w', respectively. The vectorization method expressed in equation 8 maximizes the average log probability of a context word for a given the word w.

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0}\log p(w_{t+j}|w_t) \tag{8}$$

here, T is the length of the text, m represents the size of the context window, and $p(w_{t+j})|w_t$ defines the probability of the context word. The probability is calculated using equation 9.

$$p(w_O|w_I) = \frac{\exp({v'w_O}^T vw_I)}{\sum_{j=1}^{V}\exp({v'j}^T vw_I)} \tag{9}$$

The vectorization process keeps the semantically similar words closer and expands the distance among dissimilar words. Every word is identified with a vector value and direction with is exclusive to that particular word.

2) DATASET SPLITTING

The dataset used in this experiment has been split into training, testing, and validation sets by maintaining a ratio of 70:15:15. At this ratio, 968 instances for testing, 968 instances for validation, and the remaining 4514 instances have been used for training. The training dataset has been used to train the BiLSTM network. The validation dataset has been used during the training progress to validate the performance improvement of the network through k-fold cross-validation at $k = 8$. The testing dataset has been kept untouched during training and validation and used after training the network.

B. BiLSTM NETWORK ARCHITECTURE

A simple BiLSTM network architecture, illustrated in figure 1, has been designed and implemented in this paper. The length of the input text is not constant. That is why the BiLSTM network has been used to handle up to 1200 input tokens at a time. The signals from the input layer are passed to the Embedding layer, which generates a 3D Tensor vector of shape vector. This vector contains information on the batch size (B), maximum length (L), and embedding dimension (V). This vector is transmitted to both forward and backward

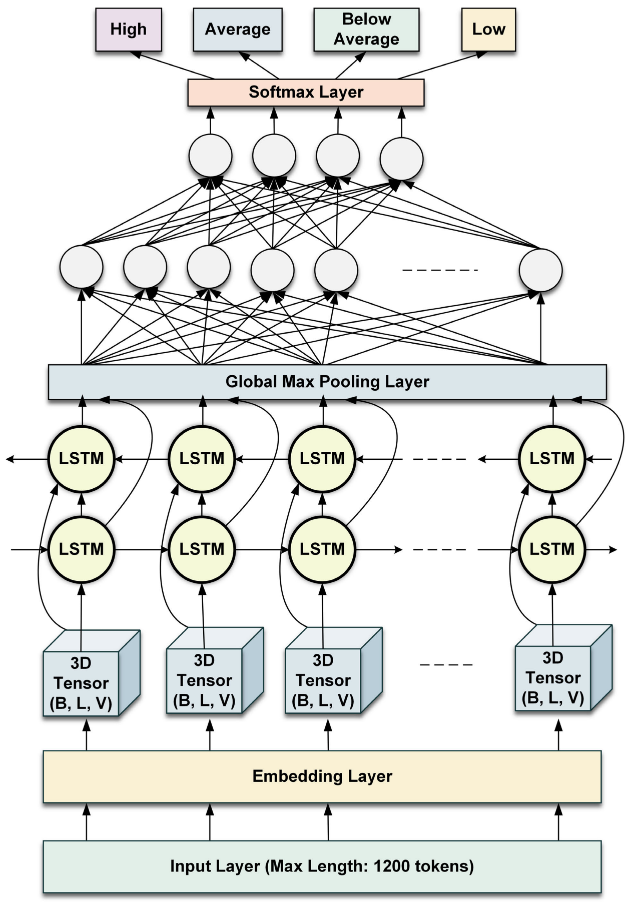

FIGURE 1. The BiLSTM network architecture.

LSTM layers and generates the average of both layers. There is a global max-pooling layer after the BiLSTM layer. This layer condenses the sequences generated by the BiLSTM layer into a single vector for each batch instance. These condensed features are used in the dense layer to learn the features. Finally, the output layer classifies the signals received from the dense layer.

1) CONCEPTUAL DESIGN OF THE NETWORK

The proposed BiLSTM networks consist of three gates. They are - forget gate, input gate, and output gate, which are defined by equations 10, 11, and 12, respectively.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{10}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{11}$$
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{12}$$

In equations 10, 11, and 12, f_t, i_t, and o_t represent activation time step at t of forget, input, output gate, respectively. The σ is the sigmoid activation function. The weights of these three input gates are the W_f, W_i, and o_t. In these three equations, h_{t-1} represents the previous hidden state, and x_t is the current input. The b_f, b_i, and o_t are the bias terms of equations 10, 11, and 12, respectively.

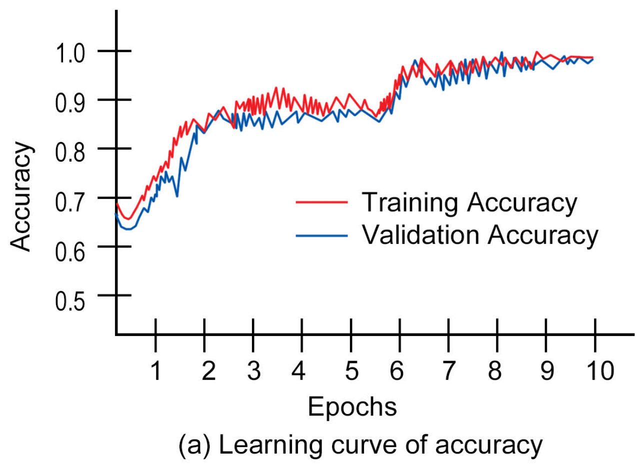

FIGURE 2. **Learning progress of the proposed BiLSTM network.**

Each cell in the BiLSTM network maintains a cell state and a hidden state for both forward and backward layers, which are defined by equation 13 where C_t is the cell state and h_t is the hidden state at time t.

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{13}$$

The cells and hidden states are updated using equation 14, and 15 where i_t is the input gate's activation function, and $\tanh(C_t)$ ensures mapping between -1 to +1.

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{14}$$
$$h_t = o_t \cdot \tanh(C_t) \tag{15}$$

The BiLSTM network propagates forward using equation 16 and backward by equation 17 where h_t^f is the forward propagation hidden state and h_t^b is the backward propagation hidden state.

$$h_t^f = LSTM_{forward}(h_{t-1}^f, x_t) \tag{16}$$
$$h_t^b = LSTM_{backward}(h_{t+1}^b, x_t) \tag{17}$$

Both forward and backward layers contribute to the output. There are three ways of generating output from BiLSTM layers. And they are concatenated, summed, or averaged output. The averaged output is governed by equation 18 has been used in this paper.

$$h_t = \frac{h_t^f + h_t^b}{2} \tag{18}$$

The outputs from the BiLSTM layer enter the global max pooling layer 19. The sequence of vectors, $S = s_1, s_2, \ldots, s_n$, produced by the BiLSTM layer where s_i refers to a vector of i^{th} features. The max pooling layer produces a single vector P. Each element of the vector, P_j, is calculated using equation 19.

$$P_j = \max_{i=1}^{n}(s_{i,j}) \tag{19}$$

In equation 19, n is the length of the input sequence, $s_{i,j}$ represents the j-th feature of the i-th vector. The max-pooling operation is defined by max. The outputs from the global max-pooling layer enter into the dense layer, which is defined by equation 20. The

$$Y = f\left(\sum_{i=1}^{d}(x_i \cdot w_i) + b\right) \tag{20}$$

In equation 20, Y is the output of a neuron, d is the number of dimensions in the input, x_i is the i-th input, w_i is the weight associated with the i-th input, b is the bias, and f is the ReLU activation function expressed by equation 21.

$$f(x) = \max(0, x) \tag{21}$$

Finally, the outputs of the dense layer enter the output layer, which has four nodes representing the four classes of the dataset. Each node uses the Sigmoid activation function given in equation 22. This function maps the signals from the dense layer into output and generates probabilistic output. The node with the highest probability is considered the predicted class.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{22}$$

2) LEARNING & OPTIMIZATION ALGORITHM

There are 4514 instances in the dataset. Instead of using all these instances simultaneously, this paper uses a mini-batch method by dividing the training dataset into smaller batches. For ease of operation, 4500 instances have been used, and it has been divided into 90 mini-batches with 50 instances in each batch. Batch normalization has been used in this experiment to accelerate and stabilize the learning process. The Adaptive Moment Estimation (ADAM) optimization algorithm has been used to update the weights of the BiLSTM network. The learning progress of the proposed BiLSTM network has been illustrated in figure 2. Figure 2(a) illustrates the training and validation accuracy. The training and validation loss curves are in figure 2(b).

a: BATCH NORMALIZATION

Batch normalization in this paper aims to have zero mean and unit variance in each mini-batch when possible. The mini-batches are expressed by equation 23. The mean (μ_B) and variance (σ_B^2) are calculated using equation 24 and 25, respectively.

$$B = \{x_1, x_2, \ldots, x_{90}\} \tag{23}$$

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{24}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \tag{25}$$

The μ_B and σ_B^2 have been used in equation 26 to normalize the activation of the mini-batches where ϵ has been used to prevent zero division error.

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{26}$$

Once the normalized activations are calculated, the shifting and scaling are applied with learnable parameters γ and β, respectively. The shifting and scaling are done using equation 27.

$$y_i = \gamma \hat{x}_i + \beta \tag{27}$$

b: OPTIMIZATION ALGORITHM

The ADAM optimization algorithm has been used to optimize the learning process because it is a combination of Gradient Descent with Momentum (GDM) and RMS Prop optimizer. In this paper, the first moment and second moment have been calculated using equations 28 and 29, respectively.

$$m_i^{t+1} = \beta_1 m_i^t + (1 - \beta_1)\nabla_{w_i} J(w) \tag{28}$$

$$m_i^{t+1} = \beta_1 m_i^t + (1 - \beta_1)\nabla_{w_i} J(w) \tag{29}$$

However, these moments have not been directly applied. The biases of these moments have been corrected using equations 30 and 31. After this correction, the ADAM generates optimal results.

$$\hat{m}_i^{t+1} = \frac{m_i^{t+1}}{1 - \beta_1^{t+1}} \tag{30}$$

$$\hat{v}_i^{t+1} = \frac{v_i^{t+1}}{1 - \beta_2^{t+1}} \tag{31}$$

After correcting the biases, the next phases have been implemented. It involves updating the weights using adaptive learning rates and moments with correct biases. The weight updating process is expressed by equation 32.

$$w_i^{t+1} = w_i^t - \frac{\eta}{\sqrt{\hat{v}_i^{t+1}} + \epsilon}\hat{m}_i^{t+1} \tag{32}$$

The overall optimization algorithm used in this paper is governed by equations 28, 29, 30, 31, and 32. In these equations, the weight is represented using w_i^t. The gradient of the cost function is $\nabla_{w_i} J(w)$. The first and second moments are expressed by m_i^t and v_i^t, respectively. The exponential decay is represented by β_1 and β_2. Although the process uses an adaptive learning rate, there is a global learning rateη is a global learning rate and it is expressed by ϵ, which is a small constant. The corrected biases of first and second moments are $\hat{m}i^{t+1}$ and $\hat{v}i^{t+1}$, respectively.

C. CONFIMIZER WORKING PRINCIPLE

The workflow diagram illustrated in figure 3 reflects the proposed Confimizer's working principle. The data from the clients are the input text for Confimizer. This text is processed to transfer to a BiLSTM network marked as figure 3(a). The BiLSTM network marked as figure 3(b) classifies the input texts into one of the four classes. Depending on the level of confidentiality, the encryption is done, which is marked as figure 3(c).

1) CONFIMIZER ALGORITHM

The proposed Confimizer algorithm, presented in algorithm 1, uses the BiLSTM network to classify the text data received from the clients. The BiLSTM network requires vectorized text which is done in text processing. It is a function named 'Text Processing()' in the algorithm. The processed texts are passed to the BiLSTM network, which classifies the text according to the level of confidentiality. If the text has low or below-average confidentiality, then the Confimizer algorithm calls the storage API and sends the data to the cloud server for storage. When the input text has average confidentiality, the algorithm calls AES encryption API and encrypts the text with a 128-bit key. After that, it is sent to cloud storage through the storage API. For text containing highly confidential information, the Confimizer algorithm encrypts the text with a 256-bit key and sends it to the cloud server for storing it.

The Confimizer algorithm encrypts only confidential data and leaves the basic data as plain text. As a result, it reduces the encryption API call. The plain text takes less storage than the encrypted text with the same length. That means it reduces the bandwidth required to transmit the data, reduces the cloud storage required, and optimizes the cloud resource by lowering the number of API calls.

IV. EXPERIMENTAL RESULTS AND EVALUATION

A. EXPERIMENTAL ENVIRONMENT

The Confimizer algorithm was deployed in an experimental environment for study and evaluation. This test employed a 64-bit version of Ubuntu Server (OS). Two data centers host ten VMs, simulating 150 to 2000 mock tasks. Each server in the test data centers is a Dell PowerEdge R940 Rack Server equipped with 8 SSDs. A pair of Intel Xeon Gold 6252 processors powers the computation of these devices. A total of 48 threads can run simultaneously across the processor's 24 cores. The highest possible frequency is 10.4 GT/s. The cache memory size is 35.75MB. This machine has 4 slots for

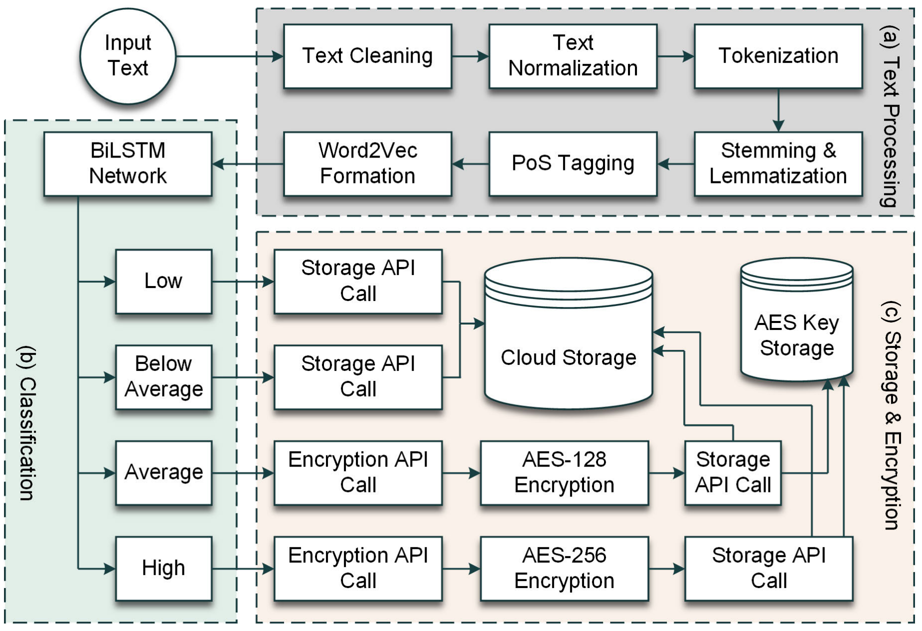

FIGURE 3. The workflow diagram of confimizer.

Algorithm 1 The Confimizer Algorithm

```
1:  Input: Input Text:$I_T$, User Request:$U_r$;
2:  Output: Storage API, $API_s$; Encrypt API, $API_e$;
3:  Start
4:  if $U_r == True$ then
5:      $confidentiality \leftarrow 0$
6:      $Word2Vec \leftarrow$ Text Processing(Input Text)
7:      $[L, B, A, H] \leftarrow BiLSTM(Word2Vec)$
8:      if $confidentiality ==$ 'L' then
9:          $API_s(I_T)$
10:     else if $confidentiality ==$ 'B' then
11:         $API_e(I_T)$
12:     else if $confidentiality ==$ 'A' then
13:         $API_s(API_e(I_T, 128))$
14:     else if $confidentiality ==$ 'H' then
15:         $API_s(API_e(I_T, 256))$
16:     else
17:         $API_s(I_T)$
18:     end if
19: end if
20: end
```

main memory and can accommodate a maximum of 32GB. Their main memory can transfer data at a rate of 3200 MT/s. The PERC H330 Adapter FH storage controller has been utilized in these server machines. There are eight SSD connections for a total capacity of 15.36 terabytes. The highest transfer rate for data from these storage devices is 6Gbps.

B. EVALUATION METRICS

The performance of the proposed Confimizer has been evaluated from two perspectives. It is a Deep Learning (DL) dependent solution that uses predictions from a BiLSTM network. From this context, the evaluation metrics are machine learning evaluation metrics. On the other hand, it is a practical solution developed to balance between confidentiality and cloud resource usage. From this point of view, the evaluation metrics are performance comparisons in terms of various measurements.

1) DL PERFORMANCE EVALUATION METRICS

After reviewing the relevant research, we concluded that the most common machine learning assessment measures are F1-score, accuracy, precision, and recall [31], [32], [33]. The equations 33, 34, 35, and 36, have been used to define these metrics in their respective order. Measurements for these evaluation metrics include True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). These values were taken from examining the confusion matrix, which has been presented in section IV-C.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{33}$$

$$Precision = \frac{TP}{TP + FP} \tag{34}$$

$$Recall = \frac{TP}{TP + FN} \tag{35}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{36}$$

2) EVALUATION METRICS ON APPLICATION

The concept of the Confimizer has been developed to solve a problem that emerged from maintaining confidentiality while outsourcing the data storage to a cloud server. The challenge is to optimize cloud resource usage to reduce the annual cost. The evaluation metrics from the application viewpoint

TABLE 2. Performance analysis on the evaluation metrics.

K	Accuracy	Precision	Recall	F1
1	83.41	79.14	76.41	76.84
2	84.75	78.64	75.49	75.1
3	83.79	76.34	73.64	75.01
4	83.04	76.98	72.45	74.61
5	85.01	73.48	74.84	73.47
Average	84.00	76.92	74.57	75.01

compare cost, storage, memory consumption, CPU usage, and API calls before and after using the Confimizer.

C. CONFUSION MATRIX ANALYSIS

The confusion matrix illustrated in figure 4 has been obtained from the experiments conducted using the proposed BiLSTM network. The confusion matrix summarizes the classifier's accurate and incorrect predictions and helps evaluate classification models. It can uncover trends and nuances in misclassification and determine accuracy, recall, F1 score, and specificity [34]. A confusion matrix is a table that defines the performance of a machine learning classifier with True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) [35]. These numbers are used to calculate state-of-the-art machine learning evaluation metrics, including accuracy, precision, recall, and F1-score.

D. CLASSIFICATION PERFORMANCE ANALYSIS

The performance of the Confimizer algorithm depends on the classification accuracy of the proposed Deep Neural Network (DNN) listed in table 2. We analyzed the performance of a deep neural network using K-fold cross-validation, comparing metrics such as accuracy, precision, recall, and F1 scores. The average accuracy, precision, recall, and F1-score across the five folds are 84.00%, 76.92%, 74.57%, and 75.01%, respectively.

In the first fold, the model achieved an accuracy of 83.41%, precision of 79.14%, recall of 76.41%, and an F1 score of 76.84%. The highest accuracy (85.01%) was observed in the fifth fold, while the lowest accuracy (83.04%) occurred in the fourth fold. Precision scores ranged from 73.48% (fifth fold) to 79.14% (first fold). The highest recall (76.41%) and F1 score (76.84%) were obtained in the first fold.

The results indicate that the deep neural network model demonstrates consistent performance across the 5-fold cross-validation, with average accuracy, precision, recall, and F1 scores of 84.00%, 76.92%, 74.57%, and 75.01%, respectively. However, variations in the performance metrics across folds highlight the importance of using cross-validation to obtain a more reliable estimate of the model's performance. Despite the fifth fold having the highest accuracy, it exhibited the lowest precision among all folds. This suggests that, although the model correctly classified a large proportion of instances, it may have produced more false positives compared to the other folds. Conversely, the first fold achieved the highest precision, recall, and F1 scores, indicating a better balance between identifying true positives and avoiding false positives. The deep neural network classifier analysis using 5-fold cross-validation revealed consistent performance across different folds.

TABLE 3. Annual cost comparison before and after using the Confimizer.

Month	Cost Before Confimizer ($)	Cost After Confimizer ($)	% Reduction
January	767.76	692.14	9.85
February	857.69	777.31	9.37
March	919.72	841.41	8.51
April	843.55	743.10	11.91
May	820.45	723.97	11.76
June	877.38	791.00	9.85
July	749.38	664.69	11.30
August	964.66	896.14	7.10
September	799.76	746.41	6.67
October	908.21	828.52	8.77
November	870.48	819.21	5.89
December	939.31	850.79	9.42
Average	859.86	781.22	9.20

E. ANNUAL COST COMPARISON

The experimental performance of the proposed Confimizer has been presented in this paper. The annual cost comparison presented in this section is the estimation calculated based on the monthly expenditure of the last 12 months. The data exchanged over the previous 12 months have been fetched from the cloud storage. The expenditure has been obtained from the monthly billing report of the service provider. Using the same cost calculation criteria, the annual cost after applying the Confimizer has been calculated in this section. The table 3 lists the annual cost before and after using the proposed Confimizer.

The innovative Confimizer algorithm has been a boon for companies seeking to cut their cloud service costs. An annual comparison showcases the significant impact of Confimizer in controlling expenses. Initially, the cost per month ranged from $749.38 in July to a high of $964.66 in August. However, after applying Confimizer, the monthly expenditures dropped noticeably, with costs fluctuating between $664.69 and $896.14. The table outlines the consistent reduction in cost, varying from 5.89% in November to an impressive 11.91% cutback in April. On average, Confimizer managed to shave off a significant 9.20% from the monthly cloud service costs. This continual reduction leads to substantial yearly savings, translating to an average decrease from $859.86 to $781.22 per month. These findings underscore Confimizer's potential to significantly transform the financial dynamics of cloud services, making it a game-changing tool in the sector. The difference between before and after using the Confimizer is illustrated in figure 5.

F. STORAGE COMPARISON

An experiment has been done on ten randomly selected instances to identify the effect of the Confimizer on storage demand reduction. The findings of the experiment have been

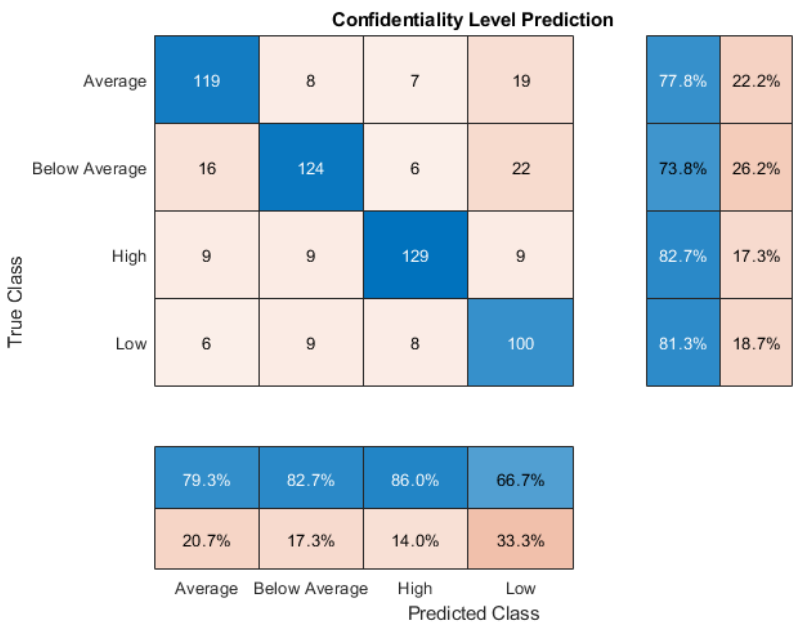

FIGURE 4. Confusion matrix analysis.

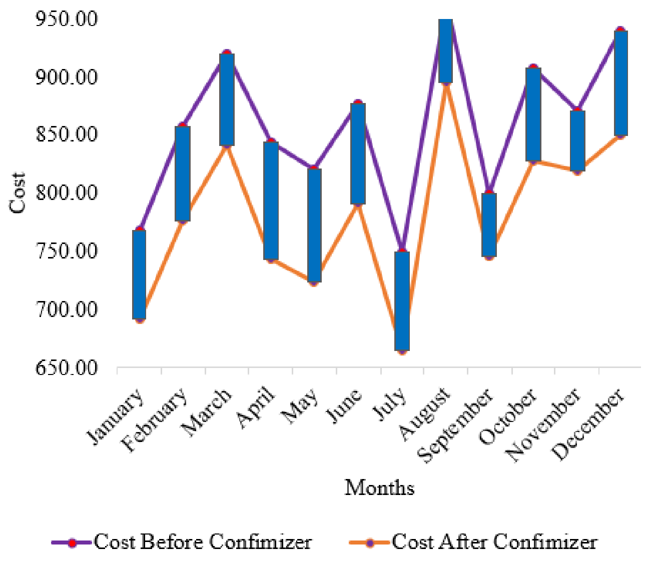

FIGURE 5. The annual cost comparison before and after using Confimizer.

listed in table 4. With an average data reduction rate of 12.33%, the Confimizer effectively optimizes storage utilization, offering substantial savings. The algorithm showcases its efficacy in several instances, with storage reduction percentages ranging from 12.39% to as high as 27.51%. Notably, in two cases, Confimizer managed to curtail storage usage from 1794 KB to 1300.5 KB and 1279 KB to 1001.22 KB, respectively, resulting in around 20% storage savings. However, it is essential to note that the algorithm showed no improvement in three instances, leaving the data storage unchanged. These three instances were non-confidential data.

TABLE 4. Storage usage comparison before and after using the Confimizer.

Serial	Before Confimizer (KB)	After Confimizer (KB)	% Savings
1	1639	1436	12.39
2	661	661	0.00
3	1794	1300.5	27.51
4	1279	1001.22	21.72
5	1021	864.47	15.33
6	1171	986.14	15.79
7	1485	1485	0.00
8	644	644	0.00
9	1032	864.47	16.23
10	1703	1458.47	14.36
Average	1242.9	1070.13	12.33

As a result, there is no effect of before and after using the proposed approach.

Figure 6 shows the storage required before and after using the Confimizer. It also illustrates the percentage savings. Confimizer has demonstrated its value overall in enhancing storage efficiency. This novel approach, therefore, proves vital for businesses seeking to minimize their cloud storage costs while maintaining their data storage needs.

G. API CALLS COMPARISON

The Confimizer reduces the number of API calls to encrypt and decrypt the data. An observation made over ten days shows 79,668 API calls daily on average. After using the Confimizer, the API calls reduce to 37,217 within the same time duration. It reflects an average 52.99% API call reduction. The observational data has been listed in table 5.

Figure 7 illustrates the effect on the API calls. It shows a significant API call reduction over ten days. Throughout

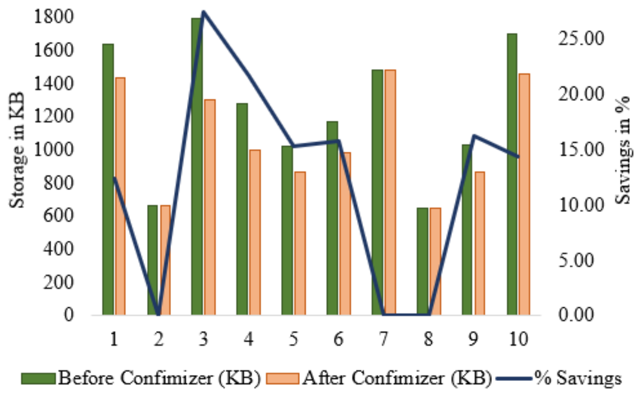

FIGURE 6. Storage usage comparison before and after using the Confimizer.

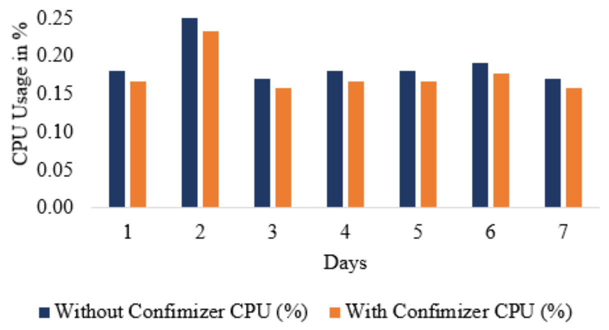

FIGURE 8. The CPU usage comparison before and after Confimizer.

TABLE 5. API calls comparison before and after using the Confimizer.

Day	Before Confimizer	After Confimizer	% Reduction
1	85447	40916	52.115
2	76993	33305.5	56.742
3	93145	35236	62.171
4	77309	32848	57.511
5	77009	44626.5	42.050
6	83014	29281.5	64.727
7	77190	38373.5	50.287
8	75754	38542.5	49.121
9	75187	37207.5	50.513
10	75628	41830	44.690
Average	79667.60	37216.70	52.99

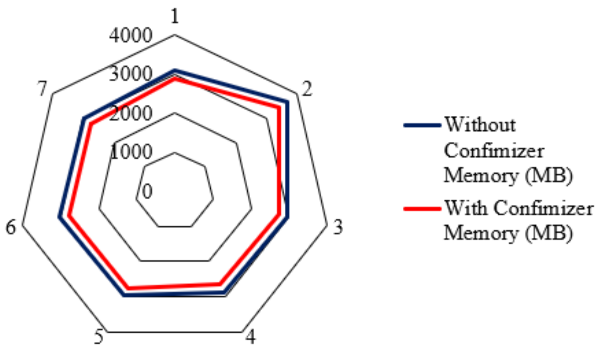

FIGURE 9. The memory consumption comparison before and after Confimizer.

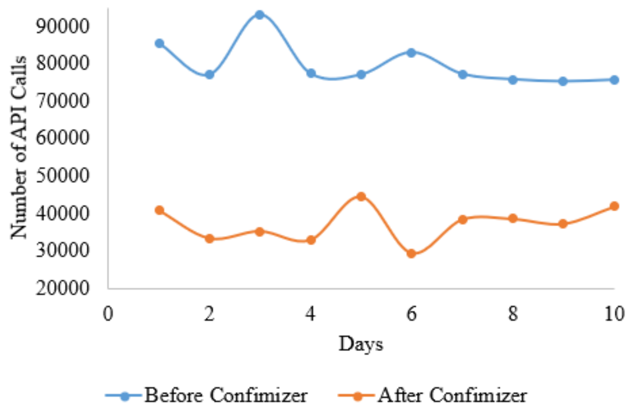

FIGURE 7. Comparison between API calls before and after using the Confimizer.

the experiment, the algorithm's performance was consistently robust, with the highlight being a dramatic 64.727% decrease on the sixth day. The impressive reduction achieved by the Confimizer algorithm signifies a potential for enhanced system performance and reduced operational costs, and a more sustainable approach to managing resources within cloud services.

H. PERFORMANCE ON COMPUTATIONAL LOAD

We analyzed the effect of Confimizer, a system optimization tool, on CPU, memory, and disk usage over a week-long period. Table 6 compares the system resource utilization of a computer with and without Confimizer. For each day, the following metrics are recorded CPU usage (in percentage), Memory usage (in megabytes, MB), and Disk usage (in megabytes per second, MB/s).

Analyzing the table 6, we observe that using Confimizer led to reduced resource utilization. The average CPU usage without Confimizer was 0.19%, while with Confimizer, it decreased to 0.18%. Our comparative analysis of CPU usage data, illustrated in figure 8 indicates that Confimizer can contribute to reduced resource consumption, although the observed improvements are modest. While these results provide some evidence of Confimizer's effectiveness, it is essential to consider the potential variations in long-term effects based on usage patterns and system configurations. More extensive studies and continuous monitoring of CPU usage with Confimizer will help determine its effectiveness in a wide range of scenarios and environments.

The average memory usage without Confimizer was 3,224 MB, whereas, with Confimizer, it dropped to 2,998 MB, a reduction of approximately 7%. The performance illustrated in figure 9 indicates that Confimizer can effectively reduce memory usage, with reductions ranging from approximately 208 to 268 MB per day. Lower memory consumption can lead to improved system performance, especially in multitasking scenarios or when working with memory-intensive applications.

TABLE 6. Performance comparison on computational load.

	Without Confimizer			With Confimizer		
Day	CPU (%)	Memory (MB)	Disk (MB/s)	CPU (%)	Memory (MB)	Disk (MB/s)
1	0.18	3105	0.11	0.17	2879.181818	0.10
2	0.25	3680	0.09	0.23	3412.363636	0.08
3	0.17	2946	0.14	0.16	2731.745455	0.13
4	0.18	2870	0.17	0.17	2661.272727	0.16
5	0.18	2978	0.07	0.17	2761.418182	0.06
6	0.19	3014	0.08	0.18	2794.8	0.07
7	0.17	2976	0.1	0.16	2759.563636	0.09

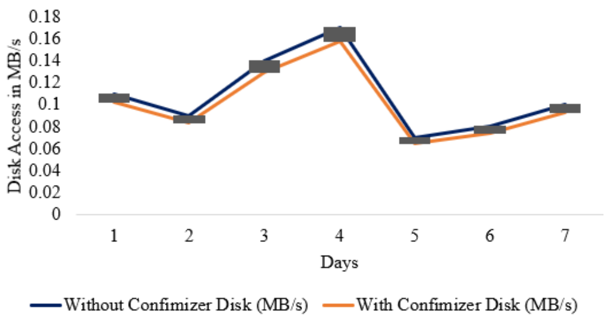

FIGURE 10. The disk access comparison before and after Confimizer.

And the average disk usage without Confimizer was 0.11 MB/s, while with Confimizer, it declined to 0.1 MB/s. The comparison of disk usage presented in figure 10 data between systems without Confimizer and with Confimizer reveals a consistent decrease in disk usage (MB/s) when utilizing the Confimizer optimization tool. In all cases, there was a reduction of 0.01 MB/s to 0.02 MB/s in disk usage when Confimizer was implemented. This indicates that Confimizer can provide modest improvements in optimizing disk resource consumption, potentially leading to faster data access, reduced disk wear, and improved overall system performance.

The results indicate that Confimizer positively impacts cloud system resource utilization by lowering CPU, memory, and disk usage. This reduction in resource consumption can lead to improved system performance, especially when resources are constrained. The most significant improvement was observed in memory usage, with an average reduction of 7%, and the overall system overload reduction is 13.75% considering all parameters.

V. LIMITATION AND FUTURE SCOPE

The proposed Confimizer is an effective algorithm in reducing cloud computational load by ranking data based on confidentiality level. Like any computing system in the world, this algorithm has some limitations.

A. WEB APPLICATION INTEGRATION ONLY

The proposed algorithm has experimented with web applications only. It has not been implemented for Android or iOS mobile applications. The proposed algorithm runs on the device of the users. It intelligently decides to make encryption and, subsequently, decryption requests. The algorithm has been implemented for web applications using the Django framework, which is for web applications. The implementation pattern for Android or iOS applications will differ, which has not been done in this experiment. It is a significant limitation of this research. However, it paves to conduct two more subsequent research to implement the Confimizer in Android and iOS applications.

B. LACK OF GENERALIZATION

The conceptual idea of the Confimizer algorithm has been implemented in this research. It is a Deep Learning (DL)-based approach that requires a relevant dataset. The dataset used in this experiment is not publicly available. It is also beyond the scope of experimenting with the trained model with types of datasets. As a result, it is evident that the BiLSTM network trained for one organization may not be suitable for another organization. As a result, the Confimizer algorithm suffers from generalizability.

C. OBSERVATION DURATION

The performance analysis was done through the data collected through an observational period of seven days. Observing the proposed solution's complete behavior is a short duration. However, because of the time constraint, the analysis was done with the data obtained from this time frame. It gives the scope to perform more research on Confimizer in data analytics and data mining domains.

This paper considers the aforementioned limitations as opportunities to conduct subsequent research projects to overcome them and keep developing the proposed Confimizer and coming up with better version as Confimizer 2.0, Confimizer 3.0, and so on.

VI. CONCLUSION

Digital data storage has become a challenge. At a 23% annual growth rate, the total volume of data will cross 175 Zettabytes by 2025. Almost every organization uses digital computers and data storage for their daily operations. Especially organizations that offer technical services are more dependent on data. Data theft, data loss, or any damage to data may lead to devastating effects. That is why safe, secure, and reliable storage is essential. However, the rapid growth rate of data and ever-rising storage demand have made cloud storage more beneficial than deploying on-premise storage. However, cloud storage imposes the challenge of data confidentiality. Assuring it causes additional expenditure. The Confimizer algorithm ensures confidentiality in cloud computing and reduces cloud resource consumption. As a result, the cost reduces without compromising confidentiality.

The core concept of the Confimizer algorithm is simple yet innovative and effective. Securing data involves encryption and decryption, which consume computational resources. The additional expenditure for encryption and decryption is not significant for small volumes of data. However, it becomes a major issue for massive volumes of data. The pay-as-you-go payment method of cloud storage service becomes expensive for voluminous data. The methodology of the Confimizer labels the data according to their confidentiality level. It encrypts the data which are confidential only, and leaves the other data as plain text. A BiLSTM network has been designed and optimized in this research to predict the input data's confidentiality level. The algorithm uses the prediction from the BiLSTM network and applies AES-256 for the most confidential data. For average confidentiality, it applies AES-128. And the algorithm leaves the below-average and low confidential data as plain text.

This simple yet effective algorithm reduces the computational resource consumption by 13.75%. As a result, it significantly reduces the expenditure. On average, it saves 9.20% cost of cloud storage without compromising confidentiality. After using the Confimizer, the average API calls reduce by 52.99% in the experimental settings. Another outstanding effect of the Confimizer is the storage demand reduction. It requires 12.33% storage when the Confimizer is used. The well-optimized and properly designed BiLSTM network plays the most significant role in the Confimizer algorithm. The algorithm performs remarkably well because of the accurate prediction from the network. The overall accuracy, precision, recall, and F1-score of the BiLSTM network are 84.00%, 76.92%, 74.7%, and 75.01%.

Despite the outstanding performance and potential of the Confimizer, it suffers from multiple limitations. The experiment presented in this paper is replicable. However, it requires the dataset used in this experiment which is confidential. A BiLSTM network trained for a particular organization will be effective in that organization. Moreover, the performance analyses presented in this paper have been observed briefly. Observation over several years is suitable to generate potentially more acceptable results. However, these limitations don't undermine the outstanding performance of the Confimizer algorithm. These are more opportunities for further development and strengthening of the algorithm through subsequent experiments.

REFERENCES

[1] J. Surbiryala and C. Rong, "Cloud computing: History and overview," in *Proc. IEEE Cloud Summit*, Aug. 2019, pp. 1–7.

[2] J. Zhao, L. Zhang, and Y. Zhao, "Informatization of accounting systems in small- and medium-sized enterprises based on artificial intelligence-enabled cloud computing," *Comput. Intell. Neurosci.*, vol. 2022, pp. 1–9, Aug. 2022.

[3] H. Saini, A. Upadhyaya, and M. K. Khandelwal, "Benefits of cloud computing for business enterprises: A review," in *Proc. Int. Conf. Adv. Comput. Manage. (ICACM)*, 2019, p. 5.

[4] D. Rani and R. K. Ranjan, "A comparative study of SaaS, PaaS and IaaS in cloud computing," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 4, no. 6, pp. 458–461, 2014.

[5] M. Rady, T. Abdelkader, and R. Ismail, "Integrity and confidentiality in cloud outsourced data," *Ain Shams Eng. J.*, vol. 10, no. 2, pp. 275–285, Jun. 2019.

[6] N. Khan and S. Anandaraj, "A survey on preserving data confidentiality in cloud computing using different schemes," in *Proc. 3rd Soft Comput. Signal Process. (ICSCSP)*, vol. 2. Germany: Springer-Verlag, 2022, pp. 211–219.

[7] A. R. Patel, R. V. Tiwari, and R. A. Khureshi, "Comparative study of top cloud providers on basis of service availability and cost," *Int. J. Multidisciplinary Res.*, vol. 4, no. 6, pp. 1–8, Dec. 2022.

[8] V. Veeresh and L. R. Parvathy, "Data privacy in cloud computing, an implementation by Django, A Python-based free and open-source web framework," *Int. J. Intell. Syst. Appl. Eng.*, vol. 10, no. 3s, pp. 56–66, 2022.

[9] S. Zhao, J. Miao, J. Zhao, and N. Naghshbandi, "A comprehensive and systematic review of the banking systems based on pay-as-you-go payment fashion and cloud computing in the pandemic era," *Inf. Syst. e-Business Manage.*, vol. 21, pp. 1–29, Jan. 2023.

[10] J. Huang, W. Susilo, F. Guo, G. Wu, Z. Zhao, and Q. Huang, "An anonymous authentication system for pay-as-you-go cloud computing*," *IEEE Trans. Depend. Sec. Comput.*, vol. 19, no. 2, pp. 1280–1291, Mar./Apr. 2022.

[11] Y. Gao, W. Liu, and F. Lombardi, "Design and implementation of an approximate softmax layer for deep neural networks," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Oct. 2020, pp. 1–5.

[12] P. B. Prince and S. P. J. Lovesum, "Privacy enforced access control model for secured data handling in cloud-based pervasive health care system," *Social Netw. Comput. Sci.*, vol. 1, no. 5, p. 239, Sep. 2020.

[13] M. Ali, L. T. Jung, A. H. Sodhro, A. A. Laghari, S. B. Belhaouari, and Z. Gillani, "A confidentiality-based data classification-as-a-service (C2aaS) for cloud security," *Alexandria Eng. J.*, vol. 64, pp. 749–760, Feb. 2023.

[14] M. A. Zardari and L. T. Jung, "Classification of file data based on confidentiality in cloud computing using K-NN classifier," *Int. J. Bus. Anal.*, vol. 3, no. 2, pp. 61–78, Apr. 2016.

[15] U. A. Butt, R. Amin, M. Mehmood, H. Aldabbas, M. T. Alharbi, and N. Albaqami, "Cloud security threats and solutions: A survey," *Wireless Pers. Commun.*, vol. 128, no. 1, pp. 387–413, Jan. 2023.

[16] B. Abd-El-Atty, M. ElAffendi, and A. A. A. El-Latif, "A novel image cryptosystem using gray code, quantum walks, and Henon map for cloud applications," *Complex Intell. Syst.*, vol. 9, no. 1, pp. 609–624, Feb. 2023.

[17] S. A. Sheik and A. P. Muniyandi, "Secure authentication schemes in cloud computing with glimpse of artificial neural networks: A review," *Cyber Secur. Appl.*, vol. 1, Dec. 2023, Art. no. 100002.

[18] M. K. Abdul-Hussein and H. T. S. ALRikabi, "Secured transfer and storage image data for cloud communications," *Int. J. Online Biomed. Eng.*, vol. 19, no. 6, pp. 4–17, May 2023.

[19] K. Vinitha, P. Thirumoorthy, and S. Hemalatha, "Data storage, data forwarding, data retrieval with big data deepfakes in secure cloud storage," in *Handbook of Research on Advanced Practical Approaches to Deepfake Detection and Applications*. Hershey, PA, USA: IGI Global, 2023, pp. 106–119.

[20] S. C. Hui, M. Y. Kwok, E. W. S. Kong, and D. K. W. Chiu, "Information security and technical issues of cloud storage services: A qualitative study on university students in Hong Kong," *Library Hi Tech*, vol. 41, no. 3, Mar. 2023.

[21] S. Kumar and D. Kumar, "Securing of cloud storage data using hybrid AES-ECC cryptographic approach," *J. Mobile Multimedia*, vol. 19, no. 2, pp. 363–388, Nov. 2022.

[22] N. M. Reddy, G. Ramesh, S. B. Kasturi, D. Sharmila, G. Gopichand, and L. T. Robinson, "Secure data storage and retrieval system using hybridization of orthogonal knowledge swarm optimization and oblique cryptography algorithm in cloud," *Appl. Nanosci.*, vol. 13, no. 3, pp. 2449–2461, 2023.

[23] S. Raj and B. Arunkumar, "Enhanced encryption for light weight data in a multi-cloud system," *Distrib. Parallel Databases*, vol. 41, nos. 1–2, pp. 65–74, 2023.

[24] M. Kara, A. Laouid, A. Bounceur, M. Hammoudeh, and M. Alshaikh, "Perfect confidentiality through unconditionally secure homomorphic encryption using OTP with a single pre-shared key," *J. Inf. Sci. Eng.*, vol. 39, no. 1, pp. 183–195, 2023.
[25] X. Zhang, C. Chen, Y. Xie, X. Chen, J. Zhang, and Y. Xiang, "A survey on privacy inference attacks and defenses in cloud-based deep neural network," *Comput. Standards Interfaces*, vol. 83, Jan. 2023, Art. no. 103672.
[26] Y. S. Abdulsalam and M. Hedabou, "Security and privacy in cloud computing: Technical review," *Future Internet*, vol. 14, no. 1, p. 11, Dec. 2021.
[27] R. K. Nema, A. K. Saxena, and R. Srivastava, "Survey of the security algorithms over cloud environment to protect information," in *Proc. 10th Int. Conf. Emerg. Trends Eng. Technol.-Signal Inf. Process. (ICETET-SIP)*, Apr. 2022, pp. 1–6.
[28] K. Shilton, D. Heidenblad, A. Porter, S. Winter, and M. Kendig, "Role-playing computer ethics: Designing and evaluating the privacy by design (PbD) simulation," *Sci. Eng. Ethics*, vol. 26, no. 6, pp. 2911–2926, Dec. 2020.
[29] S. Bird, E. Klein, and E. Loper, *Natural Language Processing With Python: Analyzing Text With the Natural Language Toolkit*. Sebastopol, CA, USA: O'Reilly Media, 2009.
[30] S. Dutta and B. Arora, "Parts of speech (POS) tagging for Dogri language," in *Proc. 2nd Int. Conf. Comput., Commun., Cyber-Secur. (IC4S)*. Singapore: Springer, 2021, pp. 529–540.
[31] N. Faruqui, M. A. Yousuf, M. Whaiduzzaman, A. K. M. Azad, A. Barros, and M. A. Moni, "LungNet: A hybrid deep-CNN model for lung cancer diagnosis using CT and wearable sensor-based medical IoT data," *Comput. Biol. Med.*, vol. 139, Dec. 2021, Art. no. 104961.
[32] L. P. O. Paula, N. Faruqui, I. Mahmud, M. Whaiduzzaman, E. C. Hawkinson, and S. Trivedi, "A novel front door security (FDS) algorithm using GoogleNet-BiLSTM hybridization," *IEEE Access*, vol. 11, pp. 19122–19134, 2023.
[33] S. Trivedi, N. Patel, and N. Faruqui, "Bacterial strain classification using convolutional neural network for automatic bacterial disease diagnosis," in *Proc. 13th Int. Conf. Cloud Comput., Data Sci. Eng. (Confluence)*, Jan. 2023, pp. 325–332.
[34] R. Yacouby and D. Axman, "Probabilistic extension of precision, recall, and F1 score for more thorough evaluation of classification models," in *Proc. 1st Workshop Eval. Comparison NLP Syst.*, 2020, pp. 79–91.
[35] E. Beauxis-Aussalet and L. Hardman, "Visualization of confusion matrix for non-expert users," in *Proc. IEEE Conf. Vis. Anal. Sci. Technol. (VAST)*, 2014, pp. 1–2.

SANDESH ACHAR is currently a cloud distributed computing architect and the engineering leader with experience building, scaling, and managing a "world-class" engineering team for leading fortune 1000 organizations. He is also with Walmart Global Tech, as the Senior Manager of Engineering. He was the Director of Cloud Engineering, leading site reliability engineering, program management, database engineering teams, spread globally, for the fastest-growing, and multi-million-dollar product line with Workday Inc. In his experience with Intuit Inc., he has successfully led the migration of more than 100 enterprise systems to a multi-cloud environment. His most recent scholarly articles have been published in *International Journal of Engineering* journals on forensics, greener cloud, cloud security, artificial intelligence, machine learning, and observability.

NURUZZAMAN FARUQUI received the Bachelor of Science degree (Hons.) in electrical and electronics engineering (EEE) from North South University, Bangladesh, in 2016, and the master's degree in information technology from the Institute of Information Technology (IIT), Jahangirnagar University (JU), Bangladesh, in 2018.

He is currently a Senior Lecturer with the Department of Software Engineering (SWE), Daffodil International University, Bangladesh. He is also a youtuber and the author. He is also globally recognized for his educational video content on neural networks using MATLAB. He is also a Research Coordinator with the Department of SWE. He has authored three books so far. His research interests include artificial intelligence, machine learning, deep learning, cloud computing, and image processing.

Mr. Faruqui is also a member of The Institution of Engineers, Bangladesh (IEB). He is also a member of Bangladesh Society for Private University Academics (BSPUA).

ANUSHA BODEPUDI is currently a Principal Site Reliability Engineer with Workday Inc., for one of their fastest-growing, multi-million-dollar product line. Prior to Workday, she was with Intuit Inc., where she designed architectural solutions for the migration of applications from traditional data centers to multi-cloud platforms. Her most recent scholarly works have appeared in *Asian Journal of Applied Science and Engineering*, *Engineering International*, and *ABC Journal of Advanced Research*.

MANJUNATH REDDY is currently an experienced engineering technology leader with a proven track record in the industry. Highly influential and methodical engineering technology leader encompassing over more than 22 years of leadership experience with a proven track record of building and leading large-scale teams in North America and Asia to deliver complex technologies, services, IP for mobile, compute, and automotive products. He is also the lead of the embedded software and hardware engineering teams globally to deliver software, services, and applications for products shipping in millions of units. He is also the innovative leader with deep expertise in embedded software and platforms and a demonstrated history of building products, partnerships, and ecosystems to enable billions of dollars in revenue. A customer-focused leader with a strong track record of building and sustaining executive relationships.