Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

CDiCENet: A Structure-Aware Lightweight Deep Learning Model for SQL Injection Attack Detection on Resource-Constrained Devices

Dr. Amit Hariyani
Smt. Chandaben Mohanbhai Patel Institute of Computer Applications, CHARUSAT
Changa, Anand, India

Abstract **- SQL Injection Attacks (SQLIAs) remain a critical security threat to database-driven applications, particularly in resource-constrained mobile and embedded environments. Existing detection approaches often rely on predefined signatures or computationally intensive deep learning architectures, limiting their ability to efficiently detect structurally diverse attacks. This paper proposes CDiCENet (Customized Dimension-wise Convolution for Efficient Networks), a lightweight structure-aware deep learning model for SQLIA detection. The proposed approach transforms SQL queries into parse trees and represents each query using a two-channel tensor containing node-depth and node-weight information. Customized depth-wise convolution is employed to extract hierarchical features, followed by global max pooling and point-wise convolution for feature reduction and computational efficiency. The model is evaluated using benchmark datasets containing legitimate and injected queries, including simple, complex, nested, union-based, error-based, time-based, and authentication-bypass queries. CDiCENet achieves 97.4% accuracy, 97.5% precision, 98.2% recall, and 96.3% F1-score, with 234,267 trainable parameters and a reported inference time of 0.09230 s. The results demonstrate that CDiCENet provides competitive detection performance with substantially lower computational complexity, supporting its potential for SQLIA detection in resource-constrained environments.**

Keywords - SQL Injection Attack, Deep Learning, Parse Tree, Depth-wise Convolution, Lightweight Neural Network, Mobile Security, CDiCENet.

I. INTRODUCTION

The increasing adoption of database-driven web and mobile applications has made the protection of database interfaces an important cybersecurity requirement. Applications routinely process user-supplied information through SQL queries, and inadequate handling of such input can allow an attacker to manipulate the intended query structure. SQL Injection Attacks (SQLIAs) exploit this vulnerability by introducing malicious SQL expressions into application inputs, potentially resulting in unauthorized database access, information disclosure, authentication bypass, or modification of stored data. The problem becomes more challenging when malicious queries employ different syntactic forms, nested subqueries, union operations, or other variations that are not easily represented by fixed attack signatures.

A considerable body of research has investigated SQLIA detection using signature-based methods, machine learning, and deep learning. Traditional approaches commonly rely on predefined signatures, keyword lists, regular expressions, or manually selected lexical features. Although these techniques can effectively identify previously observed attack patterns, their dependence on predefined representations can reduce adaptability when attackers modify the structure of an injection. Existing SQL injection detection approaches face several challenges, including dependence on fixed signatures, limited adaptability to evolving attack patterns, reliance on predefined keyword lists, false-positive detection, inadequate handling of complex SQL queries, and the computational overhead of sophisticated detection models [1].

Machine learning and deep learning approaches have been increasingly explored to reduce reliance on manually constructed attack libraries. Existing studies have investigated models such as Support Vector Machines (SVMs), Naïve Bayes, Long Short-Term Memory (LSTM) networks, and Convolutional Neural Networks (CNNs) for SQLIA detection. For example, SVM-based detection has reported an accuracy of 96.47%, while Naïve Bayes-based detection has reported an accuracy of approximately 93.3% [2, 3]. Other studies have applied LSTM-based models and CNN-based approaches to improve SQLIA detection performance [4, 5].

A key observation from these approaches is that the representation of an SQL query plays an important role in detection performance. A flat sequence of tokens may not adequately capture hierarchical relationships among SQL operators, clauses, operands, and nested subqueries. Consequently, recent research has explored structural representations based on parse trees. Parse-tree-based methods preserve relationships among query components, while node-weighted parse-tree representations combined with depth-wise CNN and Tree-LSTM have been investigated for complex and nested SQL queries [19, 20].

The investigation of complex SQL queries has demonstrated the potential of hierarchical representations for SQLIA detection. Recent work has introduced node-weighted parse trees together with depth-wise convolutional neural networks and Tree-LSTM models to address nested SQL subqueries. Such approaches combine structural representations with deep learning to capture relationships within complex queries and have reported accuracy of 98.9% for simple queries and 98.2% for complex queries [20]. However, the use of multiple processing stages and recurrent components can increase the computational complexity of the

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

overall architecture, which may restrict applicability in resource-constrained deployment environments.

This limitation is particularly relevant to mobile and embedded systems, where processing capability, memory, and energy resources are generally constrained. Therefore, an SQLIA detection model intended for such environments should provide reliable classification while minimizing computational complexity and model parameters. The existing research landscape indicates that relatively little attention has been paid to lightweight SQLIA detection in resource-constrained and embedded environments. This highlights the need for efficient detection architectures that preserve effective SQL query analysis while reducing computational and memory requirements.

To address this gap, this paper proposes Customized Dimension-wise Convolution for Efficient Networks (CDiCENet), a lightweight structure-aware deep learning architecture for SQLIA detection [5]. The proposed method transforms an SQL query into a parse tree and represents each node using its hierarchical depth and number of child nodes. These characteristics are encoded into a two-channel distance tensor, where one channel represents depth information, and the other represents node-weight information. Instead of applying conventional convolution directly to the complete representation, CDiCENet employs depth-wise convolution to process individual channels, customized global max pooling to retain significant structural features, and point-wise convolution to reduce channel dimensionality. The resulting representation is flattened and fed into a dense classifier with a sigmoid activation for binary classification [6, 7].

The proposed architecture is designed to preserve the hierarchical characteristics of SQL queries while reducing the computational requirements of the detection model. For a representative 48-token SQL query, the resulting input tensor has dimensions of 2 x 48 x 48. Following depth-wise convolution and global max pooling, point-wise convolution reduces the channel representation before classification. This design provides a direct mechanism for combining structural SQL-query analysis with lightweight convolutional processing.

The effectiveness of CDiCENet is evaluated using two benchmark datasets containing legitimate and malicious SQL queries, including simple, complex, nested, union-based, error-based, time-based, and authentication-bypass queries [8]. The evaluation considers not only classification performance but also model parameters and inference time, since computational efficiency is a central requirement of resource-constrained deployment.

The main contributions of this study are summarized as follows:

1. **Structure-aware SQL representation:** A two-channel representation based on parse-tree depth and node weight is employed to preserve hierarchical SQL-query information [9].
2. **Lightweight convolutional architecture:** CDiCENet integrates customized depth-wise convolution, global max pooling, and point-wise convolution to reduce computational and parameter requirements.
3. **Efficient SQLIA classification:** The proposed architecture provides binary classification of legitimate and injected SQL queries while retaining structural information from complex queries [10, 11].
4. **Comprehensive evaluation:** CDiCENet is evaluated using benchmark datasets containing diverse SQL query patterns and compared with conventional machine learning and deep learning approaches using accuracy, precision, recall, F1-score, inference time, and trainable parameters.
5. **Resource-constrained deployment perspective:** The study investigates the trade-off between detection performance and computational complexity, with particular emphasis on potential deployment in mobile and embedded environments.

The remainder of this paper is organized as follows. Section II presents the related work on SQLIA detection and lightweight deep learning. Section III describes the proposed CDiCENet architecture and the SQL query representation. Section IV presents the experimental datasets, implementation environment, and evaluation metrics. Section V discusses the experimental results and computational comparison. Section VI discusses the findings and limitations, while Section VII concludes the paper and presents directions for future research.

II. RELATED WORK

SQL Injection Attack (SQLIA) detection has been investigated using signature matching, statistical analysis, conventional machine learning, and deep learning techniques. Existing studies demonstrate that the choice of query representation substantially influences detection performance. However, challenges remain in handling previously unseen attack patterns, structurally complex queries, and deployment on resource-constrained platforms.

A. Signature- and Pattern-Based Approaches

Early SQLIA detection systems predominantly relied on predefined signatures, keywords, and pattern-matching techniques. Rawat and Shrivastav employed Support Vector Machine (SVM) classification for SQL injection attack detection [3, 12]. Similarly, regular-expression-based approaches have been investigated as an intermediate mechanism for identifying potential injection patterns. Although such approaches can effectively detect predefined attack signatures, their dependence on fixed patterns may limit their ability to represent the structural diversity of SQL injection attacks.

Uwagbole et al. investigated machine-learning-based predictive analytics for SQL injection attack detection and prevention using SQL-related features [13, 14]. Regular-expression-based approaches have also been investigated for identifying predefined SQL injection patterns. Although pattern-based approaches can provide effective detection of known attack characteristics, their reliance on predefined patterns can limit their adaptability to structurally different attacks [15].

The principal limitation of signature-based detection is its dependence on previously identified attack characteristics. Variations in token arrangement, query structure, or injection representation may reduce the effectiveness of static signatures. Consequently, limited adaptability to evolving attack patterns

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

and reliance on predefined keyword lists remain important challenges for signature-based SQLIA detection [16].

B. *Machine Learning-Based Approaches*

Machine learning methods have been introduced to learn statistical and lexical characteristics of SQL queries. Joshi and Geetha investigated SQL injection detection using machine learning, while Uwagbole et al. applied machine-learning-based predictive analytics to SQL injection attack detection and prevention [2, 14]. These studies demonstrate the applicability of conventional machine learning to SQLIA detection, although many such approaches rely on lexical or manually constructed features.

Other machine-learning approaches have explored classifiers such as SVM, Naïve Bayes, and related supervised learning methods for SQLIA detection [2, 3, 16, 17]. However, several of these methods rely primarily on lexical, statistical, or manually constructed features, which may not fully preserve the hierarchical relationships within complex SQL statements.

These observations suggest that increasing classification accuracy alone is insufficient. A useful SQLIA detector should also provide a representation that captures structural relationships within the query while maintaining acceptable computational requirements.

C. *Neural Network-Based Approaches*

Deep learning has subsequently been applied to reduce dependence on manually designed classification rules. Zhang et al. proposed SQLNN, which transformed SQL query information into word-vector and sparse-matrix representations before classification with a multi-hidden-layer neural network [18]. The reported accuracy of the SQLNN model was maintained above 96%.

CNN-based approaches have also been investigated for SQLIA detection. Falor et al. studied the use of convolutional neural networks for SQL injection attack detection and compared CNN performance with conventional machine learning algorithms using accuracy, precision, recall, and ROC-based measures [5].

More recent neural architectures have combined convolutional and recurrent processing to capture structural and contextual characteristics of SQL queries. Begum et al. proposed a node-weighted parse-tree representation combined with depth-wise CNN and Tree-LSTM for detecting nested query-based SQL injection attacks. Their approach reported 98.9% accuracy for simple queries and 98.2% for complex queries [20]. However, the approach relies on predefined SQL tags for semantic encoding, which may limit its generalizability across different datasets and diverse attack scenarios.

D. *Parse-Tree-Based SQLIA Detection*

Another research direction is to represent SQL queries by their syntactic structure [19]. Instead of treating an SQL statement solely as a sequence of tokens, parse-tree representations preserve relationships among different query components.

The research underlying this paper first introduced a modified parse-tree approach for simple SQL queries. Weighted sibling-node pairs were extracted from the parse tree to represent patterns within the WHERE clause, and an MLP classifier was used to distinguish legitimate and injected queries. The approach achieved 97.8% accuracy in the reported evaluation.

The research was subsequently extended to attack-type classification using a stacked neural-network ensemble. Four major SQLIA categories—error-based, time-based, union-based, and authentication-bypass attacks—were considered. The ensemble architecture combined multiple level-0 neural learners with a level-1 meta-learner and achieved 97.98% accuracy.

For more complex queries, including nested subqueries, the research introduced a node-weighted parse tree, a depth-wise CNN, a class-based TF-IDF representation, and a Tree-LSTM. The DWCNN was used to reduce the structural representation and emphasize important nodes, while Tree-LSTM captured hierarchical and contextual relationships [20]. The resulting approach reported 97.37% accuracy for simple queries and 98.5% for complex queries.

E. *Research Gap*

The reviewed approaches reveal three major gaps. First, signature- and keyword-based techniques can be constrained by predefined attack patterns and may not adapt effectively to evolving query structures. Second, although deep learning and hierarchical models improve the ability to capture complex SQLIA patterns, architectures incorporating multiple computational stages or recurrent components can increase computational requirements. Third, comparatively little attention has been given to lightweight SQLIA detection specifically designed for resource-constrained environments, such as mobile and embedded systems. These limitations are explicitly identified in the underlying research as challenges involving adaptability, complex SQLIA detection, computational resource requirements, and embedded-system deployment.

The progression of the underlying research consequently leads from pattern-based detection → structural parse-tree representation → complex-query analysis → lightweight structural deep learning. The final stage introduces CDiCENet, which retains the structural information of the parse-tree representation while replacing computationally heavier processing with customized dimension-wise convolution, global max pooling, and point-wise convolution.

III. PROPOSED METHODOLOGY

This section presents the proposed Customized Dimension-wise Convolution for Efficient Networks (CDiCENet) model for SQL Injection Attack (SQLIA) detection. The methodology is designed to preserve the hierarchical structure of SQL queries while reducing the computational and parameter requirements of the detection model. The complete processing pipeline consists of SQL query parsing, node-weight assignment, two-channel tensor construction, depth-wise convolution, customized global max pooling, point-wise convolution, feature flattening, and binary classification. The overall architecture consists of SQL query parsing, node-depth and node-weight extraction, two-channel tensor construction, depth-wise convolution, customized global max pooling, point-wise convolution, feature flattening, and binary classification.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

A. *Overview of CDiCENet*

Unlike approaches that represent an SQL query solely as a sequence of lexical tokens, CDiCENet first transforms the query into a hierarchical parse tree. Each node in the tree represents an SQL-query element and is associated with structural information derived from its position and connectivity within the tree.

The proposed architecture uses two structural attributes:

1) Node depth, *representing the hierarchical level of a node in the parse tree.*

2) Node weight*, representing the number of child nodes associated with the node.*

These attributes are encoded into a two-channel three-dimensional distance matrix. The first channel represents depth information, while the second channel represents node-weight information.

The resulting representation is processed using three principal operations:

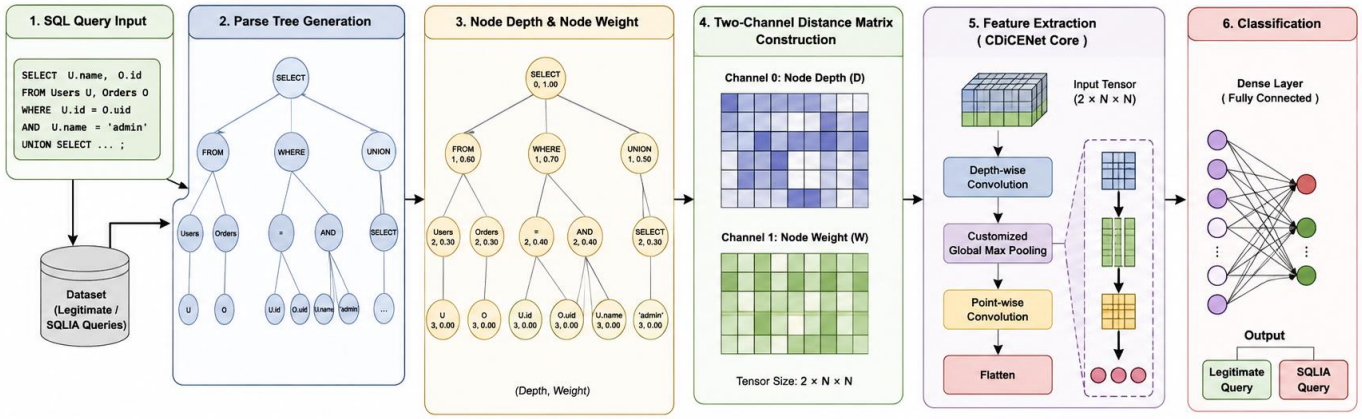

Fig. 1. Overall architecture of the proposed CDiCENet model

Depthwise Convolution→Customized Global Max Pooling →Point-wise Convolution

The resulting compact representation is subsequently flattened and supplied to a dense classification layer.

B. *SQL Query Parsing*

The first stage of CDiCENet is the conversion of an input SQL query into a parse tree. The parsing procedure analyzes the principal components of the query, including the SELECT, FROM, and WHERE clauses. Nested subqueries are handled recursively so that the resulting tree preserves the hierarchical structure of complex SQL statements.

A simplified representation of the process is:

$$Q \rightarrow P(Q)$$

where Q denotes the input SQL query and $P(Q)$ denotes its corresponding parse tree.

For example, a query containing a nested subquery is represented using parent-child relationships rather than as a flat sequence. Consequently, nodes occurring at different hierarchical levels can be distinguished according to their depth.

This representation is particularly important for complex SQLIA because malicious operations may occur inside nested expressions or subqueries where purely lexical representations may not adequately capture the relationships among query components.

C. *Node Depth and Weight Assignment*

After generating the parse tree, structural information is assigned to each node.

1) Node depth

The depth of a node indicates its hierarchical position within the parse tree. The root node is assigned the initial depth, and the depth increases as the traversal moves toward lower levels of the tree.

For a node v, its depth can be represented as:

$$D(v) = D(parent(v)) + 1$$

with the root node assigned the initial depth value.

This information allows CDiCENet to distinguish nodes according to their hierarchical positions.

2) Node Weight

The second structural characteristic is the node weight, which represents the contribution of a node based on its children. The research describes the weight-assignment process such that leaf nodes initially receive zero weight, while internal-node weights are determined from their child nodes. The resulting node representation can therefore be expressed as:

$$Vi = (Di, Wi)$$

where:

Di= depth of node i
Wi= weight associated with node i

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

The combination of these two attributes provides a compact structural description of the SQL query.

D. Two-Channel Distance Matrix Construction

After depth and weight information have been assigned, the parse-tree representation is converted into a three-dimensional tensor.

Unlike conventional image-based CNN input, which commonly uses three channels, CDiCENet uses two channels specifically designed for SQL parse-tree information. Channel 0 stores node-depth information, whereas Channel 1 stores node-weight information.

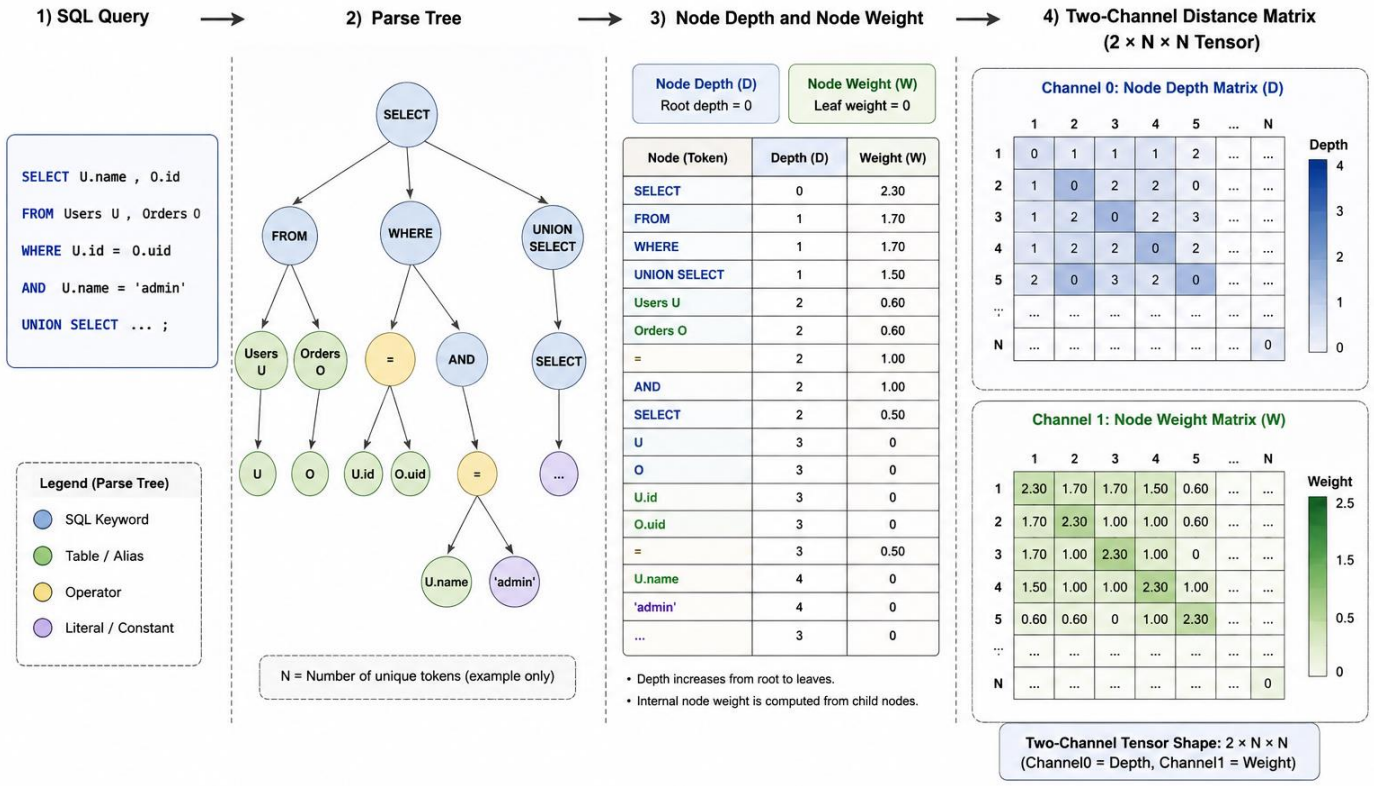

Fig. 2. Transformation of an SQL query into a two-channel structural representation.

For a parse tree containing N nodes, the input tensor can be represented as:

$$X \in \mathbb{R}^{2\times N\times N}$$

where the two channels correspond to:

$$X_0 = \text{Depth information}$$
$$X_1 = \text{Weight information}$$

The distance matrix is constructed through a traversal-based process in which the depth and weight values of the parse-tree nodes are populated into their corresponding matrix positions.

This representation allows the subsequent convolutional layers to operate on structural information rather than directly on raw SQL text.

E. Depth-Wise Convolution

The first neural-processing stage is depth-wise convolution. Conventional convolution combines information across spatial and channel dimensions simultaneously, which increases the number of parameters as the number of input and output channels increases.

In contrast, depth-wise convolution processes each input channel independently using a separate filter. This significantly reduces the computational requirements while retaining channel-specific structural information.

The research describes the depth-wise convolution operation using an input feature space I, kernel K, and output feature map O'. The operation can be represented as:

$$O'_{i',j',k'} = \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{2} I_{i,j,k} * K_{i,j,k}$$

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

where N represents the number of parse-tree nodes and K represents the two input channels. The convolutional kernel is customized for the proposed representation.

The purpose of this stage is to identify structural relationships and hierarchical patterns within the SQL parse tree while avoiding the parameter overhead associated with conventional convolution.

F. *Customized Global Max Pooling*

Following depth-wise convolution, CDiCENet applies a customized global max-pooling operation.

The purpose of this stage is to retain the most significant responses generated by the convolutional operation while suppressing less informative structural components. The proposed pooling mechanism is applied specifically to the structural channels generated by the convolution stage.

Conceptually, for a feature map F, global maximum pooling can be expressed as:

$$G_k = \max_{i,j} F_{i,j,k}$$

where G_k represents the maximum activation associated with channel K.

In CDiCENet, the customized pooling mechanism is intended to identify prominent nodes and structural patterns rather than simply reducing the spatial dimensions using a conventional pooling configuration.

G. *Point-Wise Convolution*

The third major processing stage is point-wise convolution using a 1 X 1 convolution operation.

After the depth-wise convolution and customized pooling stages, point-wise convolution combines the resulting channel information and reduces the dimensionality of the feature representation. The research identifies dimension reduction through point-wise convolution as one of the principal components of CDiCENet.

For an input feature vector X, the point-wise operation can be represented as:

$$Y_{i,j,c'} = \sum_{c=1}^{C} W_{c,c'} X_{i,j,c} + b_{c'}$$

where:

C = number of input channels,

c' = output channel,

W = point-wise convolution weights,

b = bias.

This operation produces a compact feature representation while preserving the information extracted from the preceding depth-wise convolution stage.

H. *Feature Flattening and Classification*

The output of the point-wise convolution layer is converted into a one-dimensional feature vector using a flattening operation. This vector is then provided to the dense classification layer.

The final classification layer uses a sigmoid activation function to perform binary classification:

$$P(y = 1 \mid X) = \frac{1}{1 + e^{-z}}$$

where Zis the weighted input to the output neuron.

The output represents the predicted class of the SQL query:

$$y = \begin{cases} 0, & \text{Legitimate Query} \\ 1, & \text{SQL Injection Attack} \end{cases}$$

Thus, the complete CDiCENet processing pipeline can be summarized as:

$$Q \rightarrow P(Q) \rightarrow (D,W) \rightarrow X2 \times N \times N \rightarrow$$

$$DWConv \rightarrow GMP \rightarrow PWConv \rightarrow Flatten \rightarrow Dense$$

The architecture described in the research follows this sequence from parse-tree construction through tensor generation, depth-wise convolution, global max pooling, point-wise convolution, flattening, and dense classification.

I. *Computational Efficiency of CDiCENet*

The principal motivation for using dimension-wise convolution is to reduce the computational cost of convolutional processing. For a conventional convolution operation, the number of multiplications depends on the number of filters, spatial dimensions, kernel dimensions, and input channels:

$$N \times K \times K \times M \times M \times C$$

where N is the number of filters, K X K represents the output spatial dimensions, M X M represents the kernel dimensions, and C represents the number of input channels.

Depth-wise convolution separates channel-wise processing from channel combination, thereby reducing the computational burden. CDiCENet further reduces the representation through customized pooling and point-wise convolution. Therefore, the proposed architecture is designed around three complementary objectives:

- **Preserve:** hierarchical information from SQL parse trees.
- **Extract:** important structural features using customized depth-wise filters.
- **Compress:** the learned representation using pooling and point-wise convolution.

This combination forms the basis of the lightweight architecture intended for resource-constrained environments. The research specifically identifies customized filters,

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

customized pooling, and point-wise dimension reduction as the principal architectural contributions

J. *Overall Algorithm*

The complete methodology can be summarized in the following algorithmic sequence.

Algorithm: CDiCENet-Based SQLIA Detection

Input: SQL query Q

Output: Legitimate or SQLIA

1. Parse the SQL query Q.
2. Construct the hierarchical parse tree P(Q).
3. Determine the depth of every tree node.
4. Assign a weight to each node according to its child-node structure.
5. Construct the two-channel distance tensor containing depth and weight information.
6. Apply customized depth-wise convolution independently to the input channels.
7. Apply customized global max pooling to retain prominent structural responses.
8. Apply point-wise convolution to combine and reduce channel dimensions.
9. Flatten the resulting feature representation.
10. Pass the flattened representation to the dense classification layer.
11. Apply sigmoid activation.
12. Classify the query as legitimate or SQL injection attack.

The research identifies the corresponding implementation as BUILDCDiCENetMODEL, with the input represented by the number of nodes, number of channels, input feature maps, and convolutional kernel.

IV. EXPERIMENTAL SETUP

A. *Experimental Environment*

The experiments were conducted on a system equipped with an Intel Core i7-3770 processor operating at 3.40 GHz, with four physical cores, eight logical processors, and 8 GB DDR3 memory. The same environment was used for evaluating the proposed model and the comparative CNN-based approaches.

TABLE I. EXPERIMENTAL ENVIRONMENT

Component	Specification
Processor	Intel Core i7-3770
CPU Frequency	3.40 GHz
Physical Cores	4
Logical Processors	8
Memory	8 GB DDR3

The use of a conventional CPU-based experimental environment is relevant to the objective of evaluating a lightweight architecture, as the proposed model is intended to reduce computational and memory requirements rather than depend on high-end GPU resources.

B. *Dataset Description*

Two benchmark datasets were used for experimental evaluation:

1) SQL-injection-payload-list dataset, obtained from GitHub.
2) SQL-injection-payload dataset, obtained from Kaggle.

The datasets contain both legitimate and injected SQL queries and include different query structures and SQLIA categories. The evaluated query types include simple queries, complex queries, nested queries, union-based attacks, error-based attacks, time-based attacks, and authentication-bypass attacks. The datasets were further organized into subsets to evaluate the model on different levels of query complexity.

C. *Dataset Composition*

Table II. Dataset Distribution

Dataset	Error-based SQLIA	Time-based SQLIA	Union	Authentication Bypass	Subqueries	Injected	Legitimate	Total
Dataset S	–	–	–	–	–	11,352	18,578	29,930
KComplex	–	–	5,040	–	4,020	9,060	10,120	19,180
GComplex	–	–	240	–	140	380	456	836
KSimple	1,589	–	–	2,838	–	4,427	6,323	10,750
GSimple	154	95	–	77	–	326	363	689

The dataset distribution reported in the research is reproduced in Table II.

The Dataset S subset contains 29,930 queries, including 11,352 injected queries and 18,578 legitimate queries. The complex-query subsets contain union-based and nested-subquery samples, whereas the simple-query subsets include error-based, time-based, and authentication-bypass attacks.

D. *SQL Query Representation*

Each input SQL query is first transformed into a hierarchical parse tree. The parsing process considers the SELECT, FROM, and WHERE components and recursively processes nested subqueries. Each resulting tree node is characterized using structural information associated with its depth and child-node structure.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

For CDiCENet, these node attributes are encoded into a two-channel tensor. The first channel contains node-depth information, while the second contains node-weight information. This representation enables the convolutional network to process the structural characteristics of the SQL query rather than relying exclusively on its textual sequence.

For example, the research demonstrates the processing of a 48-token complex SQL query. The corresponding input tensor has dimensions:

$$2 \times 48 \times 48$$

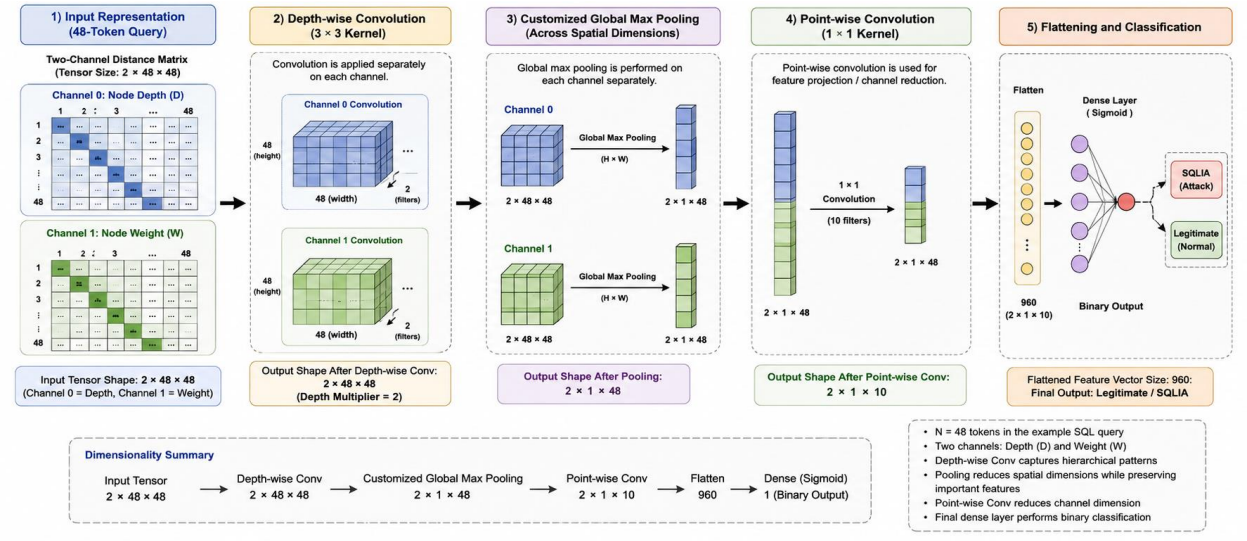

Fig. 3. Feature transformation of a 48-token SQL query through the CDiCENet architecture.

After depth-wise processing and point-wise convolution, the resulting representation has dimensions 2 X 48 X 10 which is subsequently flattened into a vector of 960 features before classification.

E. Evaluation Metrics

The performance of CDiCENet is evaluated using four standard classification metrics:

1) Accuracy

Accuracy measures the proportion of correctly classified queries among all evaluated queries:

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

where TP, TN, FP and FN denote true positives, true negatives, false positives, and false negatives, respectively.

2) Precision

Precision measures the proportion of queries classified as SQLIA that are actually malicious:

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

3) Recall

Recall measures the proportion of actual SQLIA samples correctly detected by the model:

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

4) F1-Score

The F1-score provides the harmonic mean of precision and recall:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

These metrics provide complementary information about the model. Accuracy measures overall classification correctness, whereas precision and recall are particularly important for security applications because false alarms and missed attacks have different operational consequences.

F. Comparative Models

CDiCENet was evaluated against conventional machine learning and deep learning approaches. The comparison includes:

- Support Vector Machine (SVM)
- Random Forest
- K-Nearest Neighbour (KNN)
- Decision Tree
- Naïve Bayes
- Multilayer Perceptron (MLP)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

- Standard CNN
- CNN+LSTM
- Depth-wise CNN
- Proposed CDiCENet

The research reports that these models were compared using classification metrics, while the CNN-oriented models were additionally evaluated according to inference time and parameter count.

This comparison is important because CDiCENet is not proposed solely to maximize classification accuracy. Its principal objective is to achieve competitive detection performance while reducing the computational footprint.

G. *Computational-Efficiency Evaluation*

In addition to classification performance, CDiCENet is evaluated on model parameters and inference time, which are important for deployment in resource-constrained environments. The proposed model contains 234,267 trainable parameters, compared with 700,721 for the depth-wise CNN. Its reported inference time is 0.09230 s, which is slightly lower than the depth-wise CNN's 0.09670 s and considerably lower than the standard CNN (0.49380 s) and CNN+LSTM (0.74930 s).

These measurements demonstrate that CDiCENet achieves a favorable balance between SQLIA detection performance and computational efficiency, making the model potentially suitable for resource-constrained deployment.

V. RESULTS AND DISCUSSION

The performance of CDiCENet was evaluated against conventional machine learning and deep learning models using Dataset S. The proposed model achieved 97.4% accuracy, 97.5% precision, 98.2% recall, and 96.3% F1-score. These results demonstrate that the structural representation based on parse-tree depth and node weight provides effective information for distinguishing legitimate queries from SQL injection attacks.

Although the CNN+LSTM model achieved the highest accuracy of 98.6%, its computational requirements were considerably higher. It contained 4,359,265 parameters and required 0.74930 s for inference. In comparison, CDiCENet required only 234,267 parameters and reported an inference time of 0.09230 s.

The proposed model also demonstrated competitive performance across different query complexities. CDiCENet achieved 96.8% accuracy for simple queries and 97.2% for complex queries, indicating that the structural representation remains effective when the query contains more complicated patterns.

The parameter reduction is particularly significant for resource-constrained environments. Compared with the depth-wise CNN, which contains 700,721 parameters, CDiCENet uses 234,267 parameters, representing a reduction of approximately 66.57%. At the same time, its inference time is slightly lower than that of the depth-wise CNN (0.09230 s versus 0.09670 s).

Overall, the results indicate that CDiCENet provides a favorable trade-off between detection performance and computational complexity. While a more computationally intensive model can obtain marginally higher accuracy, the substantially smaller parameter count and lower inference time of CDiCENet make it a promising candidate for resource-constrained SQLIA detection.

VI. CONCLUSION

This paper presented CDiCENet, a lightweight structure-aware deep learning model for SQL Injection Attack detection. The proposed approach represents SQL queries using parse-tree information and encodes node depth and node weight into a two-channel tensor. Depth-wise convolution is then used to extract structural features, followed by customized global max pooling and pointwise convolution to reduce the feature representation prior to binary classification.

Experimental results demonstrate that CDiCENet achieves 97.4% accuracy, 97.5% precision, 98.2% recall, and 96.3% F1-score on Dataset S. More importantly, the model contains only 234,267 parameters and reports an inference time of 0.09230 s, demonstrating a substantially lower computational footprint than the compared CNN and CNN+LSTM architectures.

The findings indicate that combining a hierarchical SQL query representation with lightweight convolutional processing can enable effective SQLIA detection without requiring a computationally intensive architecture. CDiCENet therefore offers a promising approach for security applications with limited computational resources. Future work should evaluate the model directly on mobile and embedded platforms and investigate its robustness against previously unseen and obfuscated SQL injection attacks.

ACKNOWLEDGMENT

I would like to sincerely thank my institution, faculty members, and colleagues for their valuable support, guidance, and encouragement throughout this research work. I also acknowledge the publicly available datasets used for evaluating the proposed approach.

REFERENCES

[1] P. N. Yeboah, A. S. M. Kayes, W. Rahayu, E. Pardede, and S. Mahbub, “SQL injection detection using self-supervised pre-training and multimodal techniques,” *Intelligent Systems with Applications*, vol. 31, Art. no. 200702, 2026, doi: 10.1016/j.iswa.2026.200702.

[2] B. Ladjal, M. Nadour, M. Bechouat, N. Hadroug, M. Sedraoui, A. Rabehi, M. Guermoui, and T. F. Agajie, “Hybrid deep learning CNN-LSTM model for forecasting direct normal irradiance: A study on solar potential in Ghardaia, Algeria,” *Scientific Reports*, vol. 15, no. 1, Art. no. 15404, May 2025, doi: 10.1038/s41598-025-94239-z.

[3] W. P. Chow, L. C. Y. Lau, S.-K. Teh, and C. L. Siow, “SVM-LSTM with SHAP for dynamic API security whitelisting in AWS,” in *Proc. 2025 Multimedia University Engineering Conference (MECON)*, July 2025, doi: 10.1109/MECON67253.2025.11277044.

[4] K. S. Fathi, S. Barakat, and A. Rezk, “An effective SQL injection detection model using LSTM for imbalanced datasets,” *Computers & Security*, vol. 153, Art. no. 104391, 2025, doi: 10.1016/j.cose.2025.104391.

[5] A. Falor, M. Hirani, H. Vedant, P. Mehta, and D. Krishnan, “A deep learning approach for detection of SQL injection attacks using convolutional neural networks,” in *Proc. Data Analytics and Management: ICDAM* 2021, vol. 2, Singapore: Springer, 2022, pp. 293–304, doi: 10.1007/978-981-16-6285-0_24.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

[6] A. Hariyani and P. Dolia, "An innovative method for detecting SQLi attacks by altering SQL query attribute values," *International Journal of Advanced Computer Research*, vol. 14, no. 68, pp. 89–96, 2024, doi: 10.19101/IJACR.2024.1466005.

[7] N. S. Dasari, A. Badii, A. Moin, and A. Ashlam, "Enhancing SQL injection detection and prevention using generative models," arXiv preprint arXiv:2502.04786, 2025, doi: 10.48550/arXiv.2502.04786.

[8] A. Hariyani and P. Dolia, "Comprehensive review of advanced techniques for mitigating SQL injection vulnerabilities in modern applications," *International Journal of Innovative Science and Research Technology*, vol. 10, no. 3, pp. 3063–3070, Mar. 2025, doi: 10.38124/ijisrt/25mar1982.

[9] Q. Qin, Y. Li, Y. Mi, J. Shen, K. Wu, and Z. Wang, "SQL injection detection algorithm based on Bi-LSTM and integrated feature selection," *The Journal of Supercomputing*, vol. 81, no. 4, p. 608, Mar. 2025, doi: 10.1007/s11227-025-07109-w.

[10] A. Hariyani and P. Dolia, "CryptoSQLShield: A comprehensive study on cryptography-assisted methods for SQL injection defense," *International Journal of Engineering Research & Technology (IJERT)*, vol. 15, no. 1, Jan. 2026, doi: 10.17577/IJERTV15IS010090.

[11] A. Hariyani and P. Dolia, "A Cryptography-Enforced SQL Query Integrity Framework for Complete SQL Injection Prevention," *International Journal of Scientific Research in Engineering and Management (IJSREM)*, vol. 10, no. 03, Mar. 2026, doi: 10.55041/IJSREM58457.

[12] S. Y. Hilgurt, A. M. Davydenko, T. V. Matovka, and M. P. Prygara, "Tools for Analyzing Signature-Based Hardware Solutions for Cyber Security Systems," *Journal of Cyber Security and Mobility*, vol. 12, no. 3, pp. 339–366, May 2023, doi: 10.13052/jcsm2245-1439.123.5.

[13] A. Hariyani, "A Structured Framework for Detection and Mitigation of SQL Injection Vulnerabilities in Web Applications," *International Journal of Creative Research Thoughts (IJCRT)*, vol. 14, no. 8, pp. e332–e343, Aug. 2026.

[14] S. O. Uwagbole, W. J. Buchanan, and L. Fan, "Applied machine learning predictive analytics to SQL injection attack detection and prevention," in *Proc. 2017 IFIP/IEEE Symp. Integr. Netw. Service Manag. (IM)*, Lisbon, Portugal, 2017, pp. 1087–1090, doi: 10.23919/INM.2017.7987433.

[15] P. Y. Jane and R. K. Mukadam, "A survey on hybrid SQL injection detection: Feature-selection, classical machine learning, and deep learning approaches to obfuscated, blind, and time-based SQLi," *Iconic Research and Engineering Journals*, vol. 9, no. 6, Dec. 2025, doi: 10.64388/IREV9I6-1712995.

[16] F. O. Okello, "A study of machine learning-based approaches for SQL injection detection and prevention," *International Journal of Advanced Research (IJAR)*, vol. 13, no. 2, pp. 1035–1044, 2025, doi: 10.21474/IJAR01/20461.

[17] P. Panadiya and M. K. Singhal, "Advanced detection and prevention of SQL injection attacks using machine learning techniques for enhanced web security," *Int. J. Sci. Res. Sci. Technol.*, vol. 11, no. 6, pp. 554–564, Dec. 2024, doi: 10.32628/IJSRST241161101.

[18] W. Zhang, Y. Li, X. Li, M. Shao, Y. Mi, H. Zhang, and G. Zhi, "Deep neural network-based SQL injection detection method," *Security and Communication Networks*, vol. 2022, Art. no. 4836289, 2022, doi: 10.1155/2022/4836289.

[19] Manikandan, "SQLIA detection and prevention using dynamic parse tree with query tokenization," *International Research Journal of Engineering and Technology (IRJET)*, vol. 7, no. 5, pp. 6194–6200, May 2020.

[20] A. M. Begum, M. Arock, and U. S. Reddy, "Channel minimised depth-wise CNN with node weighted tree-LSTM model to detect nested query-based SQL injection attacks," *International Journal of Intelligent Engineering Informatics*, vol. 13, no. 1, pp. 78–113, 2025, doi: 10.1504/IJIEI.2025.144267.