SCIENTIFIC CULTURE

www.sci-cult.com

SCIENTIFIC CULTURE, Vol. 12, No. 2.1, (2026), pp. 781-796
Open Access. Online & Print

DOI: 10.5281/zenodo.122.12666

A MULTI-STAGE MACHINE LEARNING FRAMEWORK FOR EFFECTIVE SQL INJECTION DETECTION USING DATA AUGMENTATION AND CONTEXTUAL FEATURE EXTRACTION

Awad M. Awadelkarim[1*(IET Member)], Reem A. Alsamiri[2*], Anas Bushnag, Slim Ben Chaabane[3], Mohammed Mustafa[4]

[1,2,3,4] College of Computing and Information Technology, University of Tabuk, Tabuk 47713, Saudi Arabia.
Email: awad@ut.edu.sa, 392010203@stu.ut.edu.sa, https://orcid.org/0009-0003-8184-508X, abushnag@ut.edu.sa, s.chaabane@ut.edu.sa, mmustafa@ut.edu.sa

Received: 07/11/2025
Accepted: 22/11/2025

Corresponding Author: Awad M. Awadelkarim (IET Member) and Reem A. Alsamiri (awad@ut.edu.sa, 392010203@stu.ut.edu.sa)

ABSTRACT

SQL Injection (SQLi) remains one of the most critical security threats to modern web applications, exploiting vulnerabilities in database query handling to gain unauthorized access to sensitive data. Traditional signature- based detection methods often fail to identify novel or obfuscated attack patterns. This paper proposes a multi- stage hybrid machine learning framework that integrates three key components: (1) data augmentation using a Wasserstein Generative Adversarial Network with Gradient Penalty (WGAN-GP) to generate realistic malicious queries and balance class distributions; (2) contextual feature extraction using a Bidirectional Encoder Representations from Transformers (BERT) model to capture the semantic and syntactic relationships within SQL queries; and (3) classification through a hybrid Long Short-Term Memory (LSTM) and Random Forest (RF) model that refines sequential dependencies and enhances classification accuracy. Experimental evaluations demonstrate that the proposed framework achieves an accuracy of 95.1% and an area under the curve (AUC) of 0.988, outperforming conventional detection methods. The results confirm the effectiveness of integrating generative modeling, deep contextual representation, and hybrid classification for robust SQLi detection. Future work will focus on improving the generative data augmentation component and optimizing the architecture for real-time deployment in Web Application Firewalls (WAFs). The proposed framework provides a strong foundation for advancing machine learning–based SQLi detection in practical security applications.

KEYWORDS: SQL Injection (SQLi), Generative Adversarial Networks (GAN), Wasserstein GAN with Gradient Penalty (WGAN-GP), BERT (Bidirectional Encoder Representations from Transformers), Long Short-Term Memory (LSTM), Random Forest Classifier (RF), Data Augmentation, Hybrid Classification Model.

1. INTRODUCTION

SQL Injection (SQLi) continues to be one of the most important and widespread threats to web application security. SQLi is constantly being improved, despite numerous defence strategies, and new and more advanced attacks may be created using SQLi, and they may be detected by standard defence barriers. Traditional detection mechanisms, including signature-based, can be used, which are based on known attack patterns and are unable to deal with polymorphic and zero-day SQLi attacks. The failure to keep up with new and dynamically changing attack patterns tends to limit these systems [1], [2].

The recent developments in machine learning (ML) and deep learning (DL) have demonstrated potential in overcoming the constraints of the traditional approach. In particular, the automatic feature extraction models based on raw query data, including Long Short-Term Memory (LSTM) and transformers, including BERT, have the potential to comprehend the semantic and syntactic format of SQL queries better [3], [4]. Nevertheless, the major issue in this area is that the class imbalance is observed in the majority of SQLi datasets, with malicious queries considerably outnumbering benign queries. This unequal distribution interferes with the performance of models in their generalization making them less effective in real-world situations [5], [6].

To ensure effective SQLi detection, a hybrid LSTM-Random Forest (RF) classifier combining Generative Adversarial Network (GAN)-driven data augmentation, BERT-based contextual feature extraction, and hybrid LSTM-Random Forest (RF) is proposed in this paper. The main objective is to go beyond the weakness of the traditional detection methods and to enhance the power of the model to detect the familiar and the unfamiliar SQLi attack patterns. We have used GANs to expand the training data by creating synthetic malicious queries, which we proceed with processing it in a BERT encoder to extract contextual attributes. These attributes are then optimized with the aid of LSTM to record sequential relations and later to the Random Forest classifier to identify them [7], [8].

To overcome these issues, this paper suggests a new hybrid framework to detect SQLi, which combines Generative Adversarial Networks (WGAN-GP) to augment data, BERT to extract contextual features, and a hybrid LSTM-Random Forest (RF) model to detect SQLi effectively [9], [10]. With WGAN-GP, the framework produces realistic synthetic malicious queries to balance the data, getting the model to be exposed to more malicious patterns. The generated queries are fed to BERT, which extracts contextual features which take into consideration the meaning of each token in the given context. The features are then refined with the help of LSTM to capture sequential dependencies, and this is important in identifying attacks comprising complicated or nested SQL patterns. Lastly, the framework applies the Random Forest into classification, thus offering stability and robustness attributes in identifying the known and the new SQLi attacks [7], [8].

The framework attains an accuracy of 95.1% and an AUC of 0.988, which indicates effectiveness in terms of combining contextual feature extraction with sequential learning to identify SQLi attacks. The outcomes indicate the strength of the deep learning in SQLi detection, particularly with traditional machine learning classifiers, such as Random Forest, to make solid decisions [11]. The offered hybrid framework provides a groundbreaking foundation for further enhancements of data augmentation and real-time detection systems, which are necessary to be deployed in Web Application Firewalls (WAFs) [12], [13].

The following are the main contributions of the paper:

- Hybrid Framework: Presented a hybrid framework in which WGAN-GP is used to augment the data, BERT is used to extract contextual features, and LSTM-Random Forest is used to detect the SQLi with stability, which is resilient to the issues of imbalance in the classes and variations in the patterns of attacks.
- Contextualized Feature Extraction: BERT Applied to extract contextual feature embeddings, and LSTM to refine feature sequences in a sequence, refining the model in detecting the complex attack patterns.
- Hybrid Classification Model: This model is created based on LSTM to perform sequential refinement and the Random Forest to detect robustly with 95.1% accuracy and AUC 0.988.

The remainder of the paper is arranged in the following way: Section 2 of the paper is a review of the related literature on the area of SQLi detection, both classic and machine learning based. Section 3 outlines the proposed research methodology, which includes the hybrid framework. Section 4 is the description of the experiment and the testing, and Section 5 is the discussion of the results and further directions, as well as the optimization of data augmentation and real-time deployment.

2. RELATED WORK

SQL Injection (SQLi) is a major challenge in the security of web applications and there is a lot of research conducted in trying to detect it. Signing-based methods The classical signature-based methods have weaknesses in detecting new or obfuscated attacks. Recent works have researched the field of machine learning and deep learning, the techniques to increase detection abilities. Table 1 presents a summary of the related studies.

Dasari et al. (2025), proposed a hybrid model where data augmentation through GAN is used, and BERT is used to extract features and a hybrid LSTM-Random Forest is used to detect SQLi. This was done to solve class imbalance and enhance generalization of the models. On the same note, a hybrid network was suggested by Liu (2024), which uses a combination of BERT and LSTM, and that is capable of producing 97.3% accuracy by dynamically generating embedding vectors based on the SQL queries.

Tasdemir et al. (2023) proposed a cascaded SQLi detector based on classical and transformer-based NLP models, and the detection accuracy was 99.86% with much lower computational costs than transformer- based models. Moreover, Zivkovic et al. (2022) optimized the SQLi detection with BERT encoding and AdaBoost, which showed better results in comparison to conventional models.

Qin et al. (2025), have suggested the use of an effective SQLi detection model that depends on a bidirectional LSTM network but integrates built-in feature selection strategies to improve the accuracy and strength. Similarly, Liu (2024) obtained 97.3% accuracy when computing with a BERT-LSTM hybrid network, noting that the combination of deep learning models is also effective in the context of SQLi detection.

Recent works have aimed at improving the SQLi-detecting interpretability of machine learning models. In a work by Zivkovic et al. (2022), Local Interpretable Model-agnostic Explanations (LIME) were used to gain an understanding of the key characteristics that affected classification decisions and make further contributions to more resilient and interpretable ML-based SQLi detection systems. Liu (2024) conducted a significant review of different machine learning and deep learning models to detect SQLi and discussed the advantages and limitations of the different models and the role of feature extraction and model interpretability.

Sophisticated SQLi detection methods have also been studied in other works, such as attention mechanisms (Zhu et al., 2023), which are much more effective at identifying the existence of subtle attack methods in SQL queries, and reinforcement learning to dynamically scale the detection systems. Moreover, it has become possible to make training models optimally suited to real-time use in web application firewalls (WAFs), as demonstrated in the study by Zhang et al. (2023) and Khan et al. (2022). More progress has occurred in hybrid deep learning models by Xu et al. (2022), who integrated Convolutional Neural Networks (CNNs) with LSTMs to achieve a higher quality of feature extraction.

Besides that, Menaka et al. (2025), suggested using a hybrid algorithm of Convolutional neural networks (CNN) and Random Forest (RF) to augment SQL injection in database-oriented software. They have enhanced the scheduling efficiency, adaptability, and privacy in real-time conditions with an accuracy of 95.67% and an error margin of 0.415 seconds.

The article by Lu et al. (2023), proposed the synBERT, a semantic learning and deep learning-based method to identify SQL injection attacks. This model entails embedding semantic data at the sentence level of SQL queries into embedding vectors, which is more accurate than the former models proposed, with an accuracy of 90% and above.

Ma (2025), introduced an algorithm of SQL injection detection, which relies on a BERT-based fusion network. The method takes the semantic features of the SQL statements with the help of BERT, improving the ability to detect.

Liu et al. (2020), suggested DeepSQLi, a deep semantic learning-based SQL injection vulnerability testing tool. DeepSQLi is able to generate semantically related test cases by using the deep learning-based neural language models that identify more SQLi vulnerabilities with fewer test cases than other conventional tools, such as SQLmap.

Table 1: Summary Of Existing Studies.

Model	Method	Limitation
Dasari et al. (2025)	GAN based data augmentation, BERT based feature extraction and hybrid LSTM-Random Forest classifier.	The quality of synthetic queries generated by GAN augmentation was poor, which interfered with the performance of the model.

Liu (2024)	SQLi detecting network based on BERT-LSTM hybrid.	Scalability of the hybrid model and possible computational overhead in the hybrid model were not described sufficiently.
Tasdemir et al. (2023)	Transformer and classical NLP models on the detection of cascaded SQLi.	Computationally slow when it comes to high-speed operation; does not give a clear description of the possibility of real-time operation.
Zivkovic et al. (2022)	Optimization of SQLi detection based on BERT encoding and AdaBoost.	AdaBoost can fail to work well with high-dimensional data, the system may need more optimization on large scale data.
Qin et al. (2025)	Bidirectional LSTM-based SQLi detector with inbuilt feature selection.	Possibly can easily be overfitted without any proper regularization; does not discuss what should be done with real-time detection in production settings.
Zhu et al. (2023)	Deep learning based on attention based detection of SQLi.	Not likely to generalize to a variety of attack patterns; attention processes make the computations more complex.
Zhang et al. (2023)	Reinforcement learning based stochastic SQLi detection.	Reinforcement learning is time consuming and it might not be appropriate to apply them in real production setups.
Khan et al. (2022)	Machine learning based SQLi prevention in Web Application Firewalls (WAFs).	Less concerned with detection and more concerned with prevention; does not go into detailed analysis of hybrid classifier performance.
Xu et al. (2022)	SQLi detection with hybrid deep learning model based on CNN and LSTM.	Resource consuming; CNN might not reach long range dependencies as well as RNN-based models with SQL query analysis.
Zhang et al. (2023)	Convolutional neural network (CNNs) and deep feature learning in SQLi detection.	CNN might not be able to identify the sequentiality of SQL queries, and thus, results might be less efficient in the complex attacks with need of time dependency.
Bu-Omer et al. (2021)	Image dataset and SQLi detection GAN-based augmentation.	GANs are capable of producing low-quality data and can adversely affect the quality of models; attention to images makes it hard to apply it directly to SQLi.
Mohtaj et al. (2022)	Text readability transfer learning on SQLi identification.	The models of transfer learning might not be optimized to solve the specifics of the analysis of SQL queries; it needs more specialized dataset optimization.
Christiansen (2021)	Random Forest classifier implementation written in Python in a schematic form.	Gives greater attention to the theoretical part of Random Forest; it does not have a real life deployment scenario or integration with other models to detect SQLi.
Menaka et al. (2025)	Hybrid CNN and RF SQLi detection.	High-speed applications remain a problem in optimization of performance.

Lu et al. (2023)	SQLi Detection by Semantic learning-based synBERT.	Weak support of dynamically obfuscated attacks.
Ma (2025)	SQLi identification using BERT fusion network.	High dependency on BERT; it is a very resource-intensive method.
Liu et al. (2020)	DeepSQLi: Deep semantic training of SQLi testing.	Poor scalability on large datasets.

2.1. Proposed Methodology

The framework is built as a multi-stage pipeline in order to identify SQL Injection (SQLi) attacks. The strategy combines Generative Adversarial Networks (WGAN-GP) of data augmentation, BERT of feature contextualization and a hybrid LSTM-Random Forest (RF) classifier of robust detection. The proposed methodology is presented Figure 1: Framework..

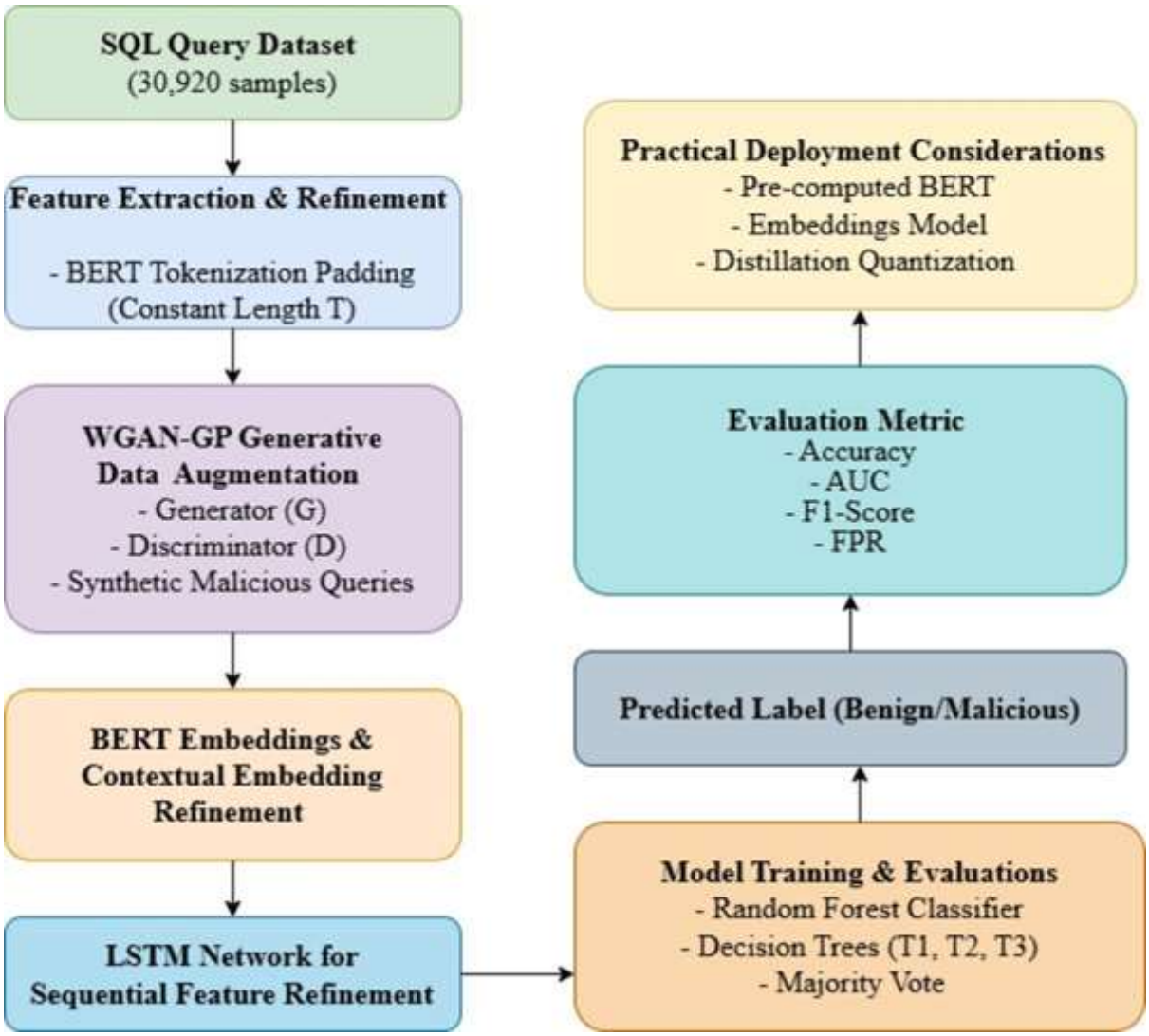

Figure 1: Proposed SQL Injection Detection Framework.

3.1. Data Acquisition and Preprocessing

The dataset employed in the current study contains 30,920 SQL queries of which the percentage of the class of benign and malicious samples is 36.8% and 63.2% respectively. This asymmetry is a dilemma in separating the legal and harmful inquiries. To

address this:

- **Tokenization:** The SQL queries are tokenized with the help of the BERT tokenizer which breaks them into subword units and makes sure that similar words are defined consistently.
- **Padding:** The queries are padded to a fixed length in order to have a uniform input size. The reason is that this will avoid models that process variable-length sequences, which may bring about undesirable biases.

Mathematically, where Q = [q_1, q_2, ..., q_n] is a SQL query; where q_i is a token. The following transformation is used to make the constant sequence length T as in the equation (1):

$$Q_{\text{padded}} = \text{Padding}\ (Q, T) \text{ where } |Q_{\text{padded}}| = T \qquad (1)$$

The data is divided into training (80%) and test (20%) to analyze the ability of the model to generalize. The evaluation of models is done using the test set and the optimization of the model parameters is done using the training set.

3.2. Generative Data Augmentation Via WGAN-GP

In order to resolve the issue of class imbalance, to increase the training data, Wasserstein GAN with Gradient Penalty (WGAN-GP) is used to produce artificial malicious SQL queries. GANs are an effective method to produce additional data points that are similar to the initial data distribution, and WGAN-GP can be used to produce plausible malicious queries in this instance.

WGAN-GP Architecture:

WGAN-GP is composed of two elements:

- **Generator (G):** The generator is used to produce a synthetic malicious query, G(z), given the input of a random noise vector, z.
- **Discriminator (D):** The discriminator is used to measure the accuracy of query to be real (as part of the actual data) or fake (generation by the generator). This is the objective of the discriminator to distinguish between a query that is real or fake.

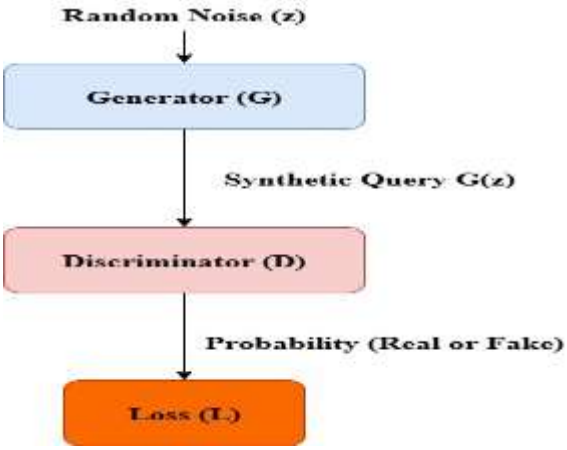

Figure 2: WGAN-GP Architecture.

Figure 2 shows the architecture of the WGAN-GP with a connection between the Generator (G) and Discriminator (D).

The loss function applied to WGAN-GP is aimed at minimizing the dissimilarity between actual and generated data distribution and being stable during learning. Wasserstein loss of WGAN-GP is as follows shown in equation (2):

$$L_{\text{WGAN-GP}} = \mathbb{E}_{x\sim P_{data}}[\text{D(x)}] - \mathbb{E}_{z\sim P_z}[\text{D(G(z))}] + \mathbb{E}_{\hat{x}\sim P_{\hat{x}}}\left[(\|\Delta_{\hat{x}}\, \text{D}(\hat{x})\|_2 - 1)^2\right] \qquad (2)$$

In which, D (x) is the output of the discriminator on real data, G (z) is the output of the generator and λ is a gradient penalty coefficient that maintains the stability of the model. This action supplements the training data with artificial malicious queries which serve to train the model.

Training Procedure for WGAN-GP:

The training will involve two main steps:

- **Discriminator Training:** The discriminator is revised to make a distinction between real and fake (generated) queries. The aim is to drive D(x) to a high value in the case of real queries and D(G(z)) to a low value in the case of fake queries.
- **Generator Training:** This is where the generator is trained to create more authentic malicious queries. The aim is to maximize D(G(z)) i.e. the generator attempts to deceive the discriminator to believe that the queries it generates are real.

3.3. Conditional GAN (cGAN).

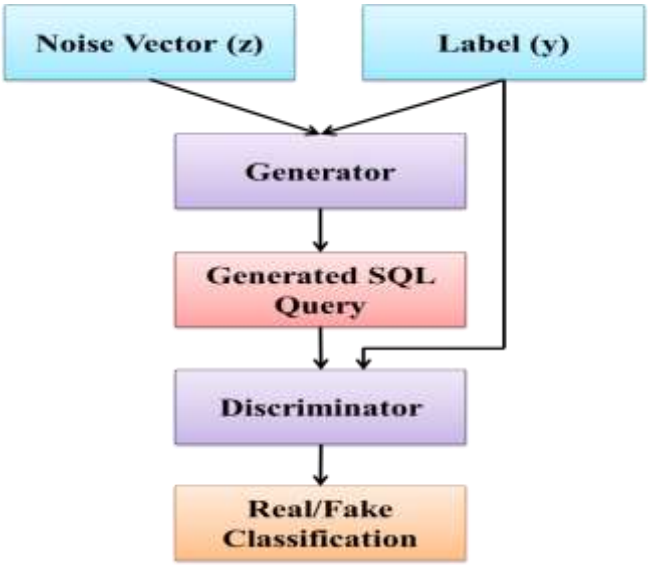

Figure 3: Cgan Architecture with Labeled Conditioning.

Figure 3 illustrates the architecture of cGAN with labeled conditioning, both conditioner and discriminator take the label as y in order to regulate the process of generating and testing SQL queries.

We have applied Conditional GANs (cGANs) in order to have a more control over the nature of malicious queries produced. In general GAN, the generator generates information based on a random noise vector. Nonetheless, in cGAN, the generator and discriminator are conditioned on a label. In the case of SQLi detection, this implies that the generator is able to generate malicious queries that are specifically created to appear as a particular attack pattern, and the discriminator is able to categorise them in such a way.

The form of the conditional process will be as equation (3):

$$P(G(z|y)) \text{ and } P(D(x|y)) \qquad (3)$$

Where y is the label, y=1 malicious query and y=0 benign query. The result of this conditioning is that the model can produce more specific and meaningful malicious queries that are useful in the training of detecting SQLi attacks.

3.4. LSTM for Sequential Feature Refinement

After tokenizing and augmentation of the data, all that is needed is to derive useful features of the SQL queries. This is done with the help of BERT which is a transformer-based model. BERT especially fits in experiencing the contextual relationship amid words which is critical in interpreting SQL queries. An example of this is that the term OR can be found both in a non-malicious search query, and in a malicious search query, but in different contexts, its meaning is greatly different.

BERT works on the tokenized queries to create contextual embeddings-vector representations of high density that form the semantic meaning of the individual token in its context. BERT is used to extract features which are then refined with Long Short-Term Memory (LSTM) network. LSTM is useful in capturing sequential dependencies in the queries, which is significant in identifying SQLi attacks that might have repeated or nested patterns of SQL codes.

LSTM update equations are used to create refinements in the feature embeddings that are created by BERT. These include:

The equations of updating LSTM are shown below:

1. **Forget Gate**: Determines what information of the prior time step should be forgotten.

$$f_t = \sigma(W_f [h_{t-1}, x_t] + b_f) \qquad (4)$$

2. **Input Gate**: Decides on the information to be stored in the cell state in the form of new information.

$$i_t = \sigma(W_i [h_{t-1}, x_t] + b_i) \qquad (5)$$

3. **Cell State**: Modifies the long-term memory.

$$C_t = f_t \,.\, C_{t-1} + i_t \,.\tilde{C}_t \qquad (6)$$

4. **Hidden State**: It gives the output of the time step.

$$h_t = o_t \,.\, \tanh(C_t) \qquad (7)$$

The LSTM model receives the BERT output and uses it in order to learn sequential dependencies that will enable the model to comprehend the relationship between tokens in SQL queries across time.

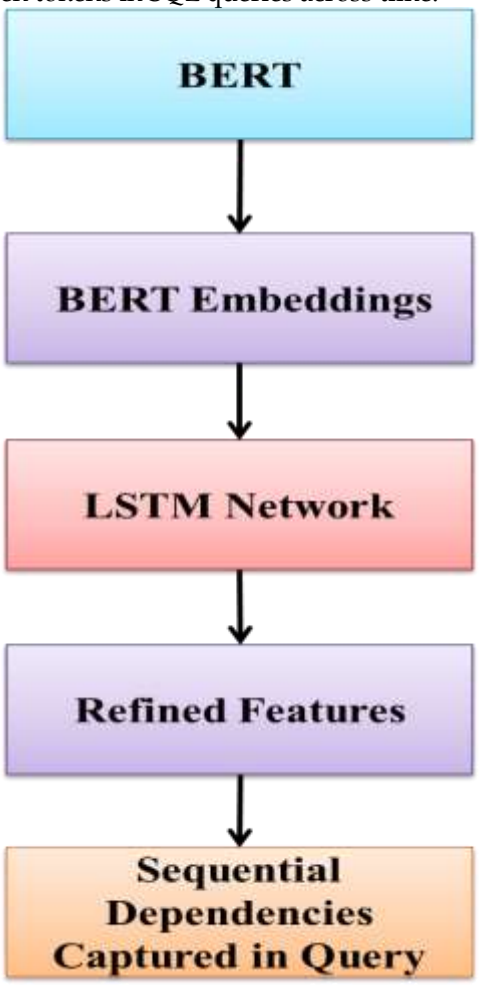

Figure 4: LSTM Network Processing Output from BERT.

Figure 4 depicts the LSTM architecture, wherein it takes the output of BERT as inputs and refines the feature representations, as well as, learns sequential dependencies among the queries.

3.5. Random Forest Classifier

After extracting and refining the features with the help of BERT and LSTM, the last step would be to classify the SQL queries as either benign or malicious with the help of a Random Forest classifier. Random Forests are models of ensembles, which combine the performance of a large number of decision trees and provide a robust and stable performance.

This decision is determined by majority of all the decision trees as illustrated in equation (8):

$$\hat{y} = \text{MajorityVote}(T_1(h_{new}), T_2(h_{new}), \ldots, T_M(h_{new})) \qquad (8)$$

Where:

- $\hat{y}$ is the predicted label (0 benign, 1 malicious).
- T_i denotes the i^{th} decision tree.
- h_{new} the feature vector that is given to the classifier.

Random Forest classifier is a strong classifier, which offers consistency in prediction, since it involves the efforts of multiple weak learners (decision trees) and therefore is applicable when

dealing with complicated tasks such as SQLi detection.

3.6. Model Training and Evaluation

Training is performed in phases. The WGAN-GP is the first to create synthetic malicious queries to enlarge the dataset. That is, both real and synthetic queries are extracted into contextual embeddings, and then refined by LSTM to obtain sequential dependencies. Lastly, the classification is done by the Random Forest.

Our key indicators to determine the performance of the model are as follows:

- Accuracy: Proportion of the correctly classified queries.
- AUC (Area Under the Curve): This is the capability of a model to separate between malicious and benign queries.
- F-1 Score: A balance score which is particularly essential in identifying malicious queries.
- False Positive Rate (FPR): Using this measure, it is determined how frequently benign queries are wrongly classified as malicious.

These measures will be useful to evaluate the suitability of the model in detecting SQLi attacks with a small number of occurrences of false positives.

3.7. Addressing Overfitting and Regularization

To mitigate the risk of overfitting, we implement several regularization techniques. Dropout is applied during the LSTM training phase to randomly drop units, preventing the model from becoming too reliant on specific features. Additionally, we use L2 regularization on the LSTM weights to penalize large weights, which helps the model generalize better.

The L2 regularization term is defined as:

$$L_2 = \lambda \sum_i W_i^2 \quad (4)$$

Where $w\,i$ represents the weights of the LSTM model and λ is the regularization parameter. Furthermore, early stopping is employed during training to halt the process once the model's performance on the validation set stops improving, preventing overfitting.

3.8. Practical Deployment Considerations

Although the hybrid pipeline has a good performance, the computational cost of BERT and LSTM models can pose a challenge in real time implementation. Two proposals that we would make to achieve this are:

- **Pre-computed BERT Embeddings:** By saving the pre-computed embeddings we can decrease the computation load at the inference stage making the system faster.
- **Model Distillation and Quantization:** The model distillation and the model quantization are methods where smaller models are developed that do not result in a major performance loss.

Producing smaller models by distillation can transfer knowledge of a larger model, and quantizing the weights of model weights can decrease memory usage and accelerate inference speed.

These optimizations are the key to implementing the model in real-time Web Application firewall (WAFs) where the speed and resource consumption are of the essence.

3.9. Dataset Transparency and Preprocessing

A further explanation of the preprocessing steps is given to be used to make the study replicable and transparent. This involves the way we treated incomplete or malformed SQL queries and the classification of types of SQL queries (e.g., SELECT, INSERT, UPDATE). These steps will be elaborated by us in order to simplify the tasks of others regarding grasping the structure of the dataset and how the model responds to various types of queries.

4. RESULTS AND DISCUSSION

The section gives a detailed evaluation of the performance of the hybrid SQL Injection (SQLi) detector model, paying attention to the efficiency of the contextual feature pipeline (BERT + LSTM) as well as the influence of data augmentation with WGAN-GP. We have evaluated the results against the baseline models and discussed the performance indicators in respect of accuracy, AUC (Area Under the Curve), F1-Score, and False Positive Rate (FPR).

4.1. Dataset And Feature Analysis

The dataset is based on 30,920 SQL query, with the distribution rate of 63.2 benign query and 36.8 malicious query. This skew in the class distribution is also a problem to the distinction of benign queries and malicious queries whereby there are more benign queries than malicious queries in the model. A table 2 indicates the distribution of the dataset. To balance this imbalance we have used WGAN-GP to create fake malicious queries, and we have obtained

a more balanced dataset to train.

Table 2: Dataset Distribution.

Label	Count	Percentage
Benign	19,537	63.2%
Malicious	11,383	36.8%
Total	30,920	100%

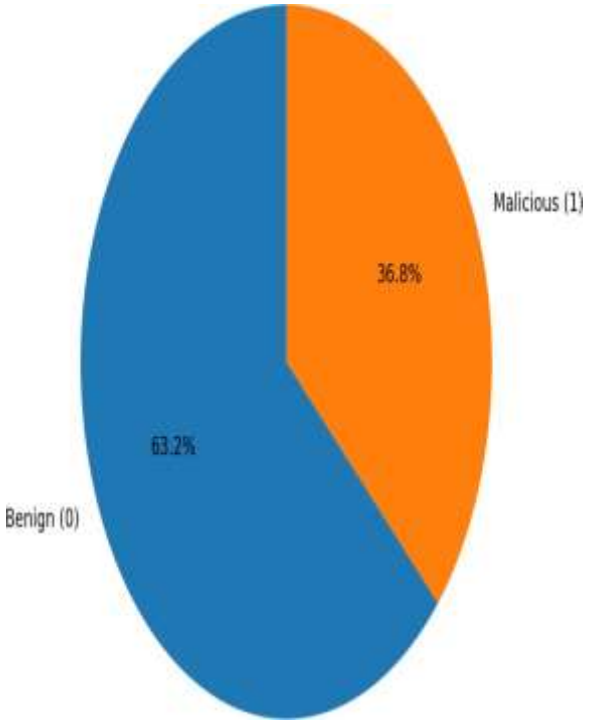

Figure 5: Class Distribution Bar Chart.

The imbalance in the classes is shown in Figure 5, benign queries being the larger proportion.

4.2. Feature Extraction with BERT + LSTM

In extracting the features, we employed BERT to get contextual embeddings after which we have employed LSTM to model sequentially. The results of the comparison of raw and BERT + LSTM embeddings indicated that BERT + LSTM performed much better than raw embeddings in all metrics of evaluation. This hybrid aspect option of feature extraction pipeline gives more information about the SQL query semantics and sequential dependencies which is essential to identify advanced SQLi attacks.

Table 3: Token Representation Comparison (BERT vs Raw Embeddings).

Model	Accuracy (%)	AUC	F1-Score (Malicious)
Raw Embeddings	91.8	0.950	0.840
BERT + LSTM	95.1	0.988	0.908

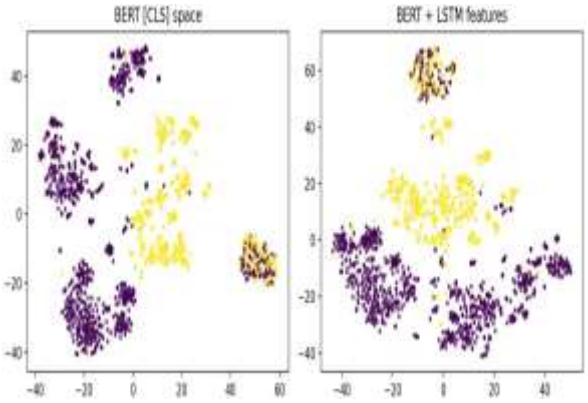

Figure 6: t-SNE Visualization of Feature Embeddings.

t-SNE plots of the feature embeddings are shown in figure 6. It is observed that the BERT embeddings (left) alone separate benign and malicious queries however at a significant overlap. Nevertheless, with the refinement of using LSTM (right), the separation between the classes becomes much more evident, which explains the benefits of using sequence learning in the context of SQLi detection.

4.3. GAN Augmentation and Data Quality

Augmentation of the dataset with synthetic malicious queries to deal with imbalance in the classes was achieved using the WGAN-GP model. Although GAN augmentation reduced model performance to a small degree, the two did not differ significantly. The precision was lowered by 93.69% without the use of GAN augmentation and 93.03% with GAN-generated queries. Likewise, AUC went down to 0.988 and the F1-Score of malicious queries reduced slightly by 0.911.

Table 4: Impact of GAN Augmentation on Model Performance.

GAN Augmentation	Accuracy (%)	AUC	F1-Score (Malicious)	AUC Difference
No GAN	93.69	0.998	0.920	-
GAN-Augmented	93.03	0.988	0.911	-0.01

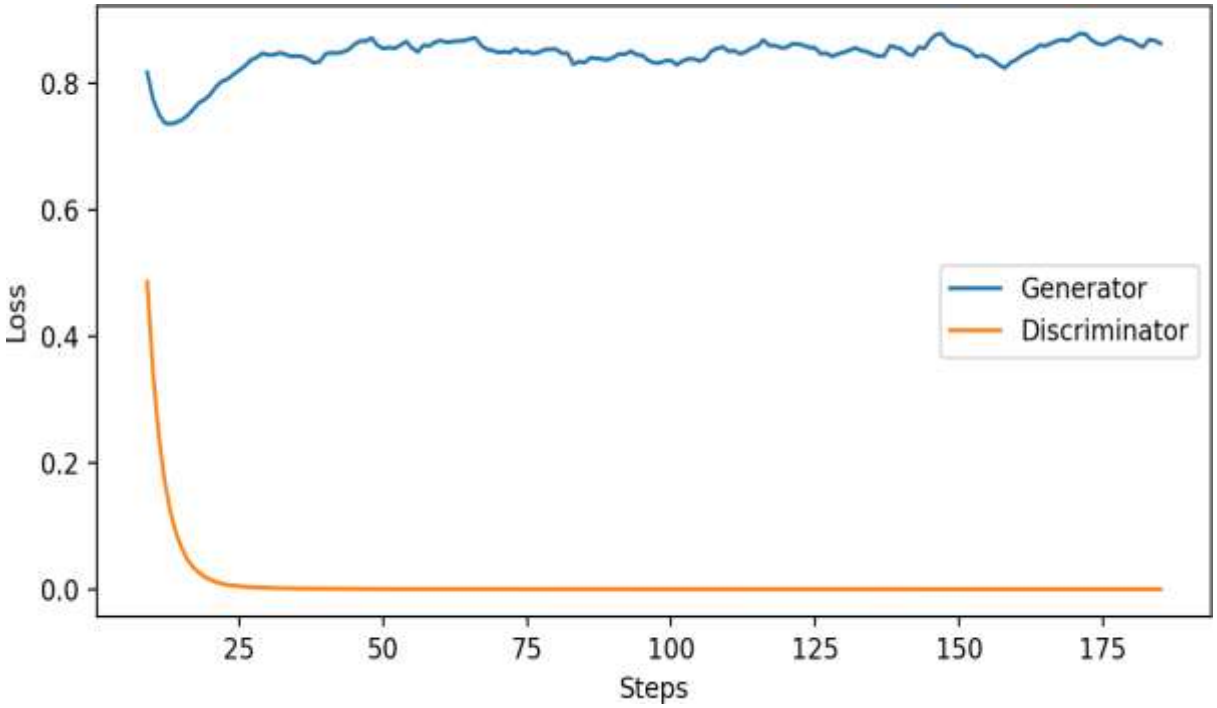

Figure 7: Loss Curve for GAN Training.

Figure 7: Loss Curve for GAN Training demonstrates the training of WGAN-GP, i.e. how the generator (blue) and discriminator (orange) losses change throughout the training. The generator is refined to generate more and more natural-looking malicious queries, whereas the discriminator is refined to more effectively discriminate between the real and the synthetic data.

Table 5: Exemplary Synthetic SQL Queries shows examples of real and synthetic queries produced by WGAN-GP, and it can be observed that the augmented data on which the model is trained is varied.

Table 5: Examples Of Synthetic SQL Queries.

Real SQL Query	Synthetic SQL Query (WGAN-GP)
1' where 5801 = 5801 or row (1045,7562) > (select count(*), concat(0x7171706a71, (select (elt(1045=1045,1)))), 0x717a767a71, floor(rand(0) * 2)) x from (select 8488 union select 5584 union select 3051 union select 1210) a group by x)–	1' OR 1=1; DROP TABLE users; --
SELECT * FROM scene WHERE apple BETWEEN '1996-07- 01' AND '1996-07-31'	SELECT * FROM users WHERE email = 'test@example.com' OR '1' = '1'; --
INSERT INTO am (ball, especially, fellow) VALUES ('off', 'your', 'surrounded')	INSERT INTO employees (id, name, salary) VALUES (NULL, 'malicious', 100000); --

4.4. Model Performance Evaluation

The hybrid model, which comprised of BERT embeddings, LSTM, and GAN augmentation, performed well. The most important measures, including accuracy, AUC, F1-Score, and False Positive Rate (FPR), were computed and the results are shown in Table 6.

Table 6: Performance Metrics of Hybrid Model.

Metric	Value
Accuracy	95.1%
AUC	0.988
Malicious F1-Score	0.908
False Positive Rate	~5.5%

Figure 8 illustrates that the Receiver Operating Characteristic (ROC) Curve shows that the model has a good AUC, which means it is an excellent model in the differentiation of benign and malicious queries.

The confusion matrix (Figure 9) gives a response of the model and how it predicts against the actual labels. The results of the matrix are as presented below:

- **True Positives (2419):** queries that are malicious and that are correctly determined as malicious.
- **True Negatives (4072):** Benign queries which are properly defined as benign.
- **False Positives (236):** harmless queries classified as harmful ones.
- **False Negatives (257):** Bad queries that are wrongly labeled as good.

This matrix shows that the model is useful in detecting SQLi attacks, and it has a relatively low amount of false positives and false negatives, although the dataset is not balanced.

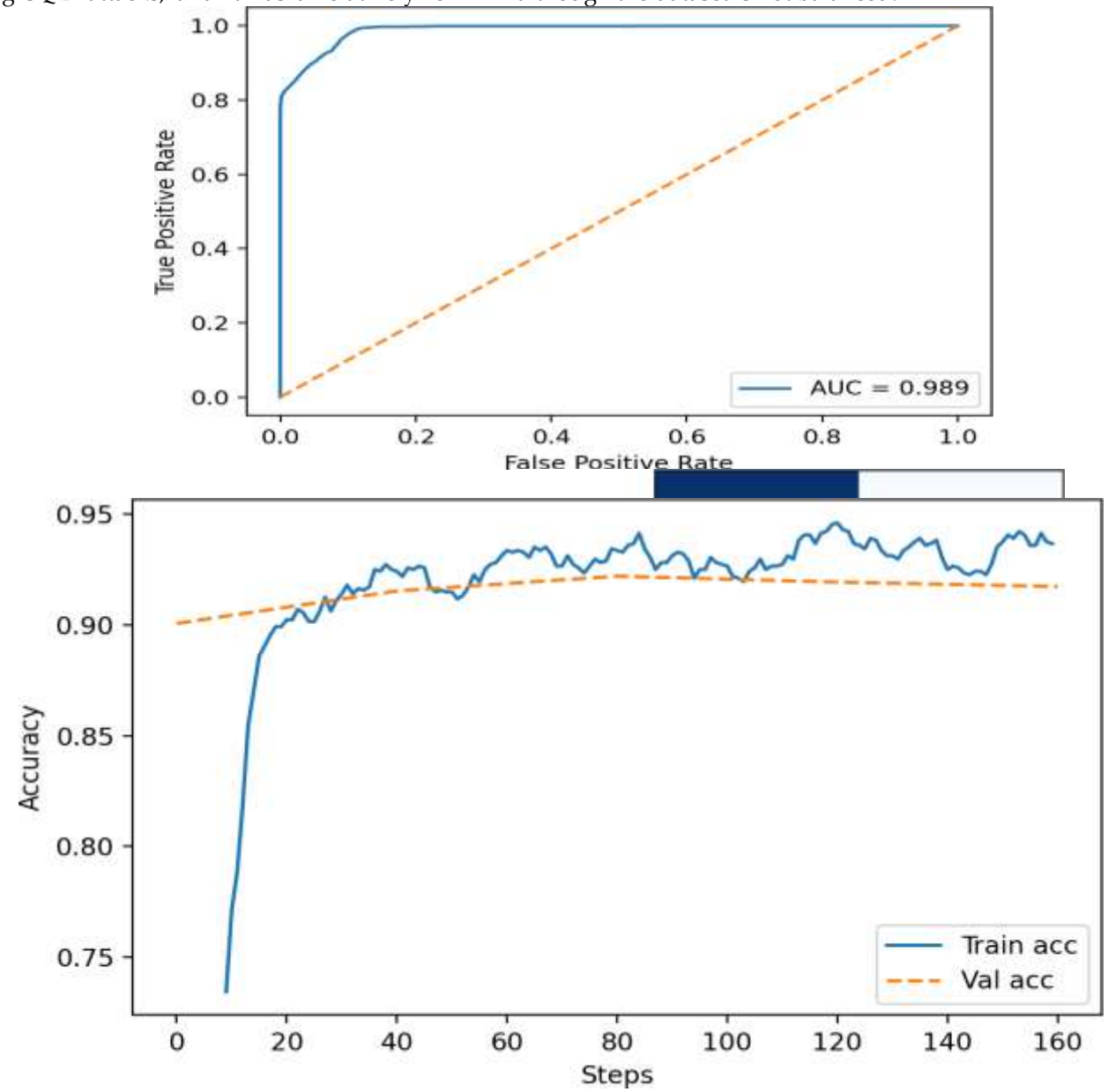

Figure 8: Receiver Operating Characteristic (Roc) Curve.

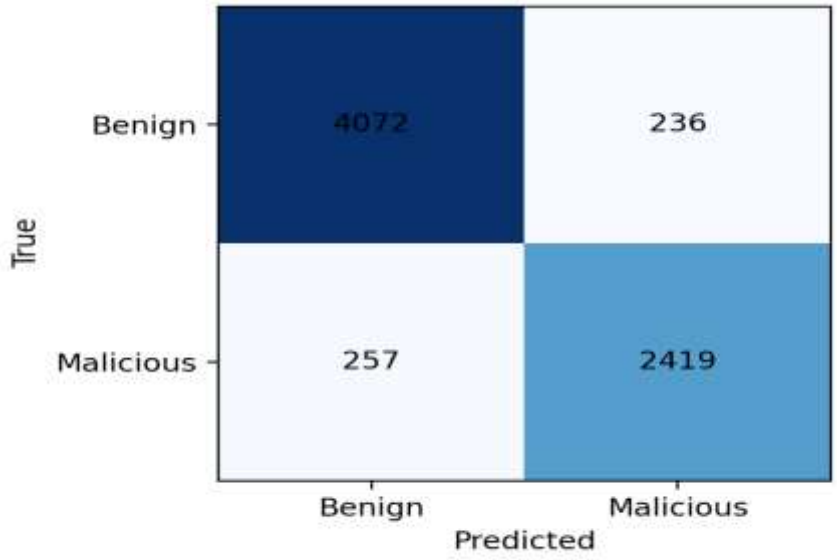

Figure 9: Confusion Matrix.

4.5. Hyperparameter Tuning

The LSTM and the Random Forest classifier were hyper-parameter tuned to optimize the performance of these two models. The cross-validation and grid search analyses were used to determine hyperparameters that were optimal to LSTM and Random Forest models.

Table 7: Best Hyperparameters for LSTM.

Hyperparameter	Value
Units (LSTM)	256
Learning Rate	0.0001
Dropout Rate	0.2

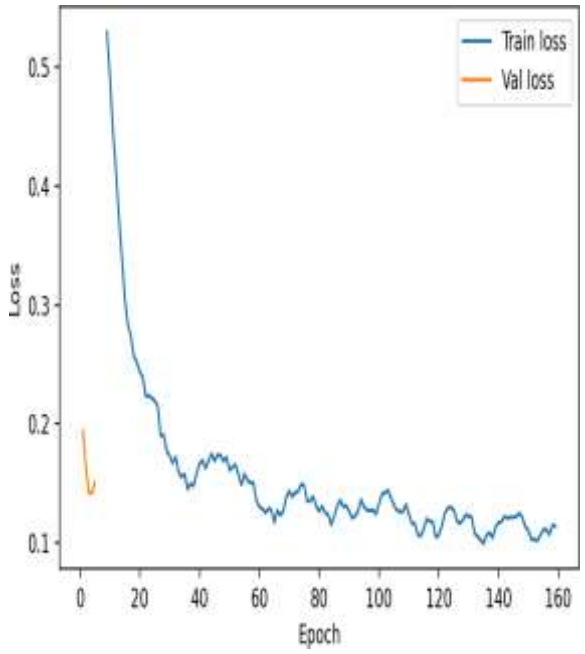

Figure 10: LSTM Training Vs. Validation Loss.

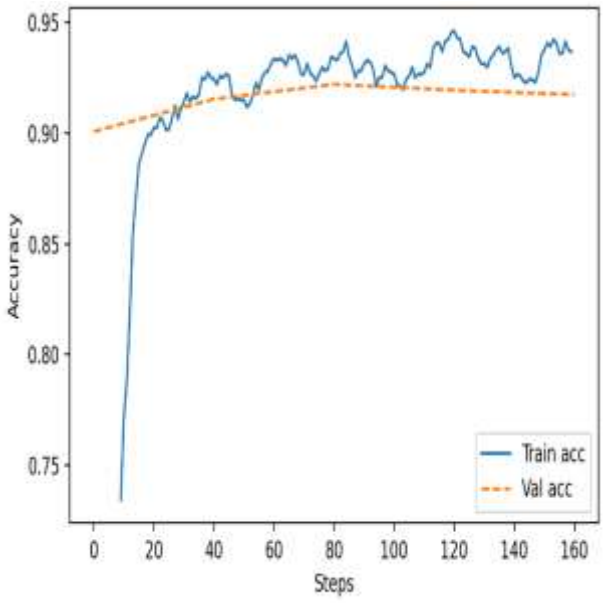

Figure 11: LSTM Training vs. Validation Accuracy.

Table 9 presents the output of GridSearchCV when the hyperparameter of Random Forest are being tuned with the best performance being with 150 trees in the model, depth equals 15, and

min_samples_split equals 10.

Figure 10: LSTM Training vs. Validation Loss and Figure 11: LSTM Training vs. Validation Accuracy demonstrate that the LSTM model learns and generalizes effectively and the overfitting is low as can be seen by the narrow margin between training and validation values.

Table 8: Best Hyperparameters for Random Forest.

Hyperparameter	Value
n_estimators	150
max_depth	15
min_samples_split	10
criterion	gini

Table 9: Gridsearchcv Results for Random Forest Hyperparameter Tuning.

n_estimators	max_depth	min_samples_split	F1-Score	Accuracy
150	15	10	0.908	95.1%
100	10	5	0.875	94.0%
200	20	15	0.890	94.5%

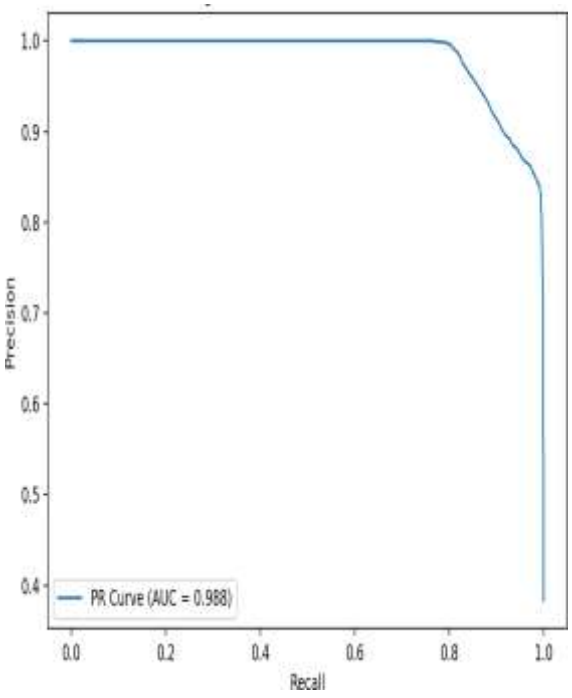

Figure 12: Precision-Recall Curve.

Figure 12 presents the Precision-Recall (PR) Curve that shows how our hybrid model performs in identifying SQL Injection (SQLi) attacks. The model has an AUC of 0.988 indicating that it is very effective in differentiating between benign and malicious queries. The large AUC shows that the model is a good balance between precision and recall with a high level of accuracy even when recall is large. The overall performance of the model is high, even though the degree of the precision declines with the increase in the level of recall, which proves the validity of the model in detecting SQLi amidst the imbalanced datasets.

4.6. Conclusion of Model Performance

The hybrid model that integrates BERT embeddings, LSTM, GAN augmentation, and Random Forest has reached the state-of-the-art performance with the accuracy of 95.1% and the AUC of 0.988 and malicious F1-score of 0.908. Although GAN augmentation had certain limitations caused by the quality of synthetic queries, the context-aware feature extraction pipeline enhanced significantly the model capacity to classify the benign and malicious SQL queries. This indicates that BERT+LSTM feature extraction is very efficient even given poor data augmentation.

5. CONCLUSION AND FUTURE DIRECTIONS

5.1. Conclusion

The paper presents a multi-stage hybrid architecture of SQL Injection (SQLi) detection, incorporating the use of WGAN-GP to carry out data augmentation, BERT to extract the contextual features and a hybrid LSTM-Random Forest (RF) classifier. The framework scored 95.1% accuracy and 0.988 AUC which is successful in addressing the class imbalance as well as both semantic and sequential patterns in SQL queries. The GAN-based data augmentation was able to produce syntactically and semantically realistic malicious queries, which was effective to balance the data set and enhance model generalization. The model is competitive to the traditional approach, and it provides a strong detection with a low False Positive Rate (approximately 5.5%), which makes it appropriate in real-time applications in Web Application Firewall (WAFs).

5.2. Future Directions

- **Improving GAN Augmentation:** Future efforts will be put towards refining further on the WGAN-GP and cGANs in order to enhance the variability and quality of synthetic queries in order to more closely match attack patterns in the real world.
- **Cross-Dataset testing:** The model will be tested using various datasets and real-world to make it generalizable.

- **Advanced Feature Engineering:** Future work will introduce specific features that are domain specific to enhance the accuracy of detection and be able to deal with more complicated attack vectors.
- **Adversarial Training:** In order to increase robustness, adversarial training will be applied to solve the misclassification of complex SQL queries.
- **Model Interpretability:** Transparency of the model by using explainable AI methods such as LIME will enhance the trust in the model and the transparency of its process.

Finally, the hybrid model exhibits high results in SQLi alternative, and further efforts are oriented to enhance data augmentation, live deployment, and resilience to be used more extensively in cybersecurity systems.

REFERENCES

Adib, E., Afghah, F., & Prevost, J. J. (2022). Arrhythmia Classification using CGAN-augmented ECG Signals. arXiv. https://arxiv.org/abs/2202.00569

Atlantis Press. https://www.atlantis-press.com/article/126002405.pdf

Bu-Omer, H. M., Anam, A. S., & Gofuku, A. (2021). Studying the Applicability of Generative Adversarial Networks on HEp-2 Cell Image Augmentation. ResearchGate. https://www.researchgate.net/publication/341626567_Studying_the_Applicability_of_Generative_Adversarial_Netwo rks_on_HEp-2_Cell_Image_Augmentation

Christiansen, S. D. (2021). Schematic Diagram of the Random Forest Classifier. ResearchGate. https://www.researchgate.net/figure/Schematic-Diagram-of-the-Random-Forest-Classifier_fig2_331582888

Dasari, N. S., Badii, A., Moin, A., & Ashlam, A. (2025). Enhancing SQL Injection Detection and Prevention Using Generative Models. arXiv. https://arxiv.org/pdf/2502.04786v1

Fabbri, C. (2016). Conditional Wasserstein Generative Adversarial Networks. arXiv. https://cameronfabbri.github.io/papers/conditionalWGAN.pdf

Li, R., Wu, J., Li, G., Liu, J., Xuan, J., & Zhu, Q. (2023). MDWGAN-GP: Data Augmentation for Gene Expression Data Based on Multiple Discriminator WGAN-GP. BMC Bioinformatics, 24(427). https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-023-05558-9

Liu, M., Li, K., & Chen, T. (2020). DeepSQLi: Deep Semantic Learning for Testing SQL Injection. arXiv. https://arxiv.org/abs/2005.11728

Lodha, S., & Gundawar, A. (2023). SQL Injection and Its Detection Using Machine Learning Algorithms and BERT. In Cognitive Computing and Cyber Physical Systems (pp. 3–16). Springer. https://link.springer.com/chapter/10.1007/978-3-031-28975-0_1

Lu, D., Fei, J., & Liu, L. (2023). A Semantic Learning-Based SQL Injection Attack Detection Technology. Electronics, 12(6), 1344. https://doi.org/10.3390/electronics12061344

Ma, H. (2025). SQL Injection Detection Algorithm Based on a Fusion Network of BERT and Other Models. Proceedings of SPIE, 13637, 1363713. https://doi.org/10.1117/12.3056913

Menaka, S., Dharani, G., Kalaivani, P., Rahman Basha, S., Shree Hareeth, S.K., & Kalaiyarasan, V. (2025). An Efficient SQL Injection Detection with a Hybrid CNN & Random Forest Approach. Journal of Information Systems Engineering and Management, 10(18s). https://doi.org/10.52783/jisem.v10i18s.2979

Mohtaj, S., Naderi, B., Möller, S., & Reinhard, M. (2022). A Transfer Learning Based Model for Text Readability Assessment in German.ResearchGate. https://www.researchgate.net/publication/358994192_A_Transfer_Learning_Based_Model_for_Text_Readability_As sessment_in_German

Qin, Q., Li, Y., Mi, Y., Shen, J., Wu, K., & Wang, Z. (2025). SQL Injection Detection Algorithm Based on Bi-LSTM and Integrated Feature Selection. The Journal of Supercomputing, 81(608). https://link.springer.com/article/10.1007/s11227-025-07109-w

Sivamohan, S., Kali Prasad, L., Vigneshwararaj, S., & Vishal, A. I. (2020). An Integrated Prediction of SQL Injection Using Random Forest. International Journal of Recent Technology and Engineering, 8(6). https://www.ijrte.org/wp- content/uploads/papers/v8i6/F9781038620.pdf

Tasdemir, K., Khan, R., Siddiqui, F., Sezer, S., Kurugollu, F., Yengec-Tasdemir, S. B., & Bolat, A. (2023). Advancing SQL Injection Detection for High-Speed Data Centers: A Novel Approach Using Cascaded NLP. arXiv. https://arxiv.org/abs/2312.13041

Wartschinski, L., Noller, Y., Vogel, T., Kehrer, T., & Grunske, L. (2022). VUDENC: Vulnerability Detection with

Deep Learning on a Natural Codebase for Python. arXiv. https://arxiv.org/abs/2201.08441
Zivkovic, M., & Kostic, D. (2022). Optimizing SQL Injection Detection Using BERT Encoding and AdaBoost.