Enhancing Machine Learning Based SQL Injection Detection Using Contextualized Word Embedding

Janet Zulu
Texas A&M University-San Antonio
San Antonio, Texas, USA
jzulu01@jaguar.tamu.edu

Bonian Han
Hangzhou Dianzi University
Hangzhou, Zhejiang, China
bonian985@gmail.com

Izzat Alsmadi
Texas A&M University-San Antonio
San Antonio, Texas, USA
ialsmadi@tamusa.edu

Gongbo Liang
Texas A&M University-San Antonio
San Antonio, Texas, USA
gliang@tamusa.edu

ABSTRACT

SQL injection (SQLi) attacks continue to severely threaten application security, allowing malicious actors to exploit web input and manipulate an application's database with malicious SQL code. This work explores the possibility of building effective SQLi detectors through machine learning. Specifically, we investigate the impact of contextualized and non-contextualized embedding methods for converting SQL queries into vector space. Our results demonstrate the superiority of the contextualized embedding method, achieving consistent accuracy above 99% across various classification algorithms and reducing model training time by 31 times. In addition, the analysis of reliability diagrams indicates that contextualized embeddings provide better model calibrations. These findings underscore the significance of contextualized word embeddings in enhancing the performance and reliability of SQLi detection models.

CCS CONCEPTS

• **Security and privacy**; • **Information systems**; • **Computing methodologies** → **Natural language processing**; **Machine learning**;

KEYWORDS

Cyber security, artificial intelligence, text embeddings.

ACM Reference Format:
Janet Zulu, Bonian Han, Izzat Alsmadi, and Gongbo Liang. 2024. Enhancing Machine Learning Based SQL Injection Detection Using Contextualized Word Embedding. In *2024 ACM Southeast Conference (ACMSE 2024), April 18–20, 2024, Marietta, GA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3603287.3651187

1 INTRODUCTION

The prevalence of SQL injection (SQLi) attacks continues to pose a significant threat to application security. These attacks allow malicious actors to manipulate an application's database by inserting malicious SQL code through web input, leading to potentially devastating consequences. From unauthorized data access and modification to compromising backend infrastructure, the impacts of successful SQLi attacks are far-reaching [8].

Current SQLi detection methods predominantly rely on manually defined features, but their ability to handle the ever-evolving range of real-world attacks remains a concern [14, 25, 31]. Machine learning (ML) approaches, such as neural networks (NNs), learn task specific features directly from extensive training data offering potential advantages over conventional hand-crafted features in terms of robustness and precision [12, 20, 34, 39]. As a result, ML models show promise as effective tools for SQLi detection, such as [21, 28].

The conceptual framework of employing ML models to combat SQLi attacks involves passing user input through a pre-trained ML model to detect the presence of malicious code, and only inputs deemed safe are subsequently forwarded to the web API server (Figure 1). However, developing such a model involves carefully choosing building blocks in two folds. First, since SQL queries are textual data, which are not understandable by ML models as-is, they need to be converted to numerical vectors through word embedding. Then, the vectors are fed into a detection model comprising feature-learning and decision-making components. There are many possible choices for each step. The multitude of choices within each of these decisions can be overwhelming.

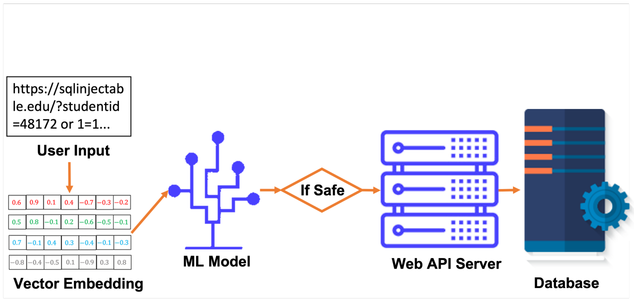

Figure 1: Illustration of Using Machine Learning Model to Prevent SQLi Attacks

In this paper, we focus on examining the influence of contextual and non-contextual word embedding methods on the performance of various ML-based SQLi detection models, including neural networks, k-nearest neighbor, random forest, and logistic regression. Our experimental results reveal that the use of contextual embedding methods not only enhances the model training efficiency (achieving 31 times or more faster training) but also leads to a notable improvement in neural network calibration, a critical aspect often ignored in evaluation [7, 13, 16].

The subsequent sections of this paper introduces the common building blocks of ML-based SQLi detector and the ones used in this study (Section 2), a comprehensive evaluation is presented (Section 3), and conclude with discussions and insights (Section 4).

2 MACHINE LEARNING-BASED SQLI DETECTOR

Since the SQLi attackers need to inject malicious code into textual SQL queries, SQLi detection may be modeled as a natural language binary classification problem, using supervised training approach. Specifically, each training sample is represented as a two-tuple $\{x, y\}$, with x is the textual input and y is the label, indicating the presents of SQL injection. The detection process is represented as:

$$(\hat{y}, \hat{p}) = h(x_v), \tag{1}$$

where $h(\cdot)$ is a ML model, such as those in Section 2.2, x_v is the vector representation of x that is generated using a non-contextualized (Section 2.1.1) or contextualized embedding method (Section 2.1.2) , $\hat{y}$ is the predicted label, and $\hat{p}$ is the predicted probability. Ideally, $\hat{y}$ is equal to y and $\hat{p}$ is the true data distribution (p). A perfectly calibrated classification model should present the following property:

$$\mathbb{P}\left(\hat{y} = y | \hat{p} = p\right) = p, \forall p \in [0, 1], \tag{2}$$

where $\mathbb{P}(\cdot|\cdot)$ indicate the conditional probability.

2.1 Word Embedding Methods

The initial step of building a classification model for textual input is to convert the raw text into vector representations using word embedding. These methods can be broadly categorized into two groups: non-contextualized embedding and contextualized embedding.

2.1.1 Non-contextualized Embedding. Non-contextualized word embedding generates fixed vector representations for each word in the vocabulary without considering the context in which the word appears in a sentence or document. Popular non-contextualized embedding methods include Bag-of-Words (BoW), Word2Vec [23], and more. These methods typically use co-occurrence statistics or predict words based on their neighboring words in a large corpus to create word embeddings. As a result, the embeddings remain the same regardless of the sentence or document they appear in, making them computationally efficient and straightforward to implement.

In our work, we adopt the Bag-of-Words model as our non-contextualized method since it is a fundamental technique, which is widely used in natural language processing (NLP) [11, 33]. In this approach, text is treated as an unordered collection of words or n-grams, and the frequency of each word is used to represent the text. While the BoW model does not capture word order or contextual information, it remains a popular choice for scenarios where word frequency and occurrence information suffice for the task at hand, offering a balance between efficiency and simplicity.

2.1.2 Contextualized Embedding. Contextualized word embedding methods generate word representations that adapt to the context in which words appear, resulting in dynamic embeddings sensitive to surrounding words [4]. This contextualization enables the embeddings to capture subtle nuances in word meaning and resolve ambiguities in polysemous words based on their context.

RoBERTa [41] is one of the prominent and influential models for NLP contextualized embedding. RoBERTa, short for "A Robustly Optimized BERT Pretraining Approach," is a variant of BERT [4] that enhances bidirectional context modeling during pretraining, utilizing the transformer [32] architecture. A pretrained RoBERTa model may be finetuned for various tasks without requiring too much effort [6, 41, 42]. In this study, we use RoBERTa as our contextualized embedding method for SQLi detection.

2.2 Machine Learning Models

Given the natural of SQLi being modeled as a binary classification task, it can be trained using a conventional machine learning model (Section 2.2.1) or neural network model (Section 2.2.2).

2.2.1 Conventional Machine Learning Model. In this work, we uses k-nearest neighbor (KNN), logistic regression, and random forest as conventional machine learning models.

KNN is a simple and effective algorithm used for classification tasks, based on the principle that similar data points are more likely to belong to the same class [24]. In our case, for each new textual input x, KNN identifies the K nearest training samples and classifies x based on the majority class among its neighbors. KNN is easy to implement and interpret, but its performance might be sensitive to the choice of K and the distance metric.

Logistic Regression is a widely used statistical method for binary classification [36]. It models the relationship between the input features and the probability of the positive class (SQL injection present) using a logistic function. The model can be trained using optimization techniques to find the best parameters that minimize the logistic loss. Logistic Regression is computationally efficient, interpretable, and can handle both linear and non-linear relationships between features and the target variable [9].

Random Forest [2] is an ensemble learning method that combines multiple decision trees [37] to make predictions. Each tree is trained on a random subset of the data with random feature subsets, and the final prediction is determined by aggregating the predictions of individual trees. Random Forest is robust against overfitting, works well with high-dimensional data, and can capture complex relationships in the data. It is less sensitive to the choice of hyperparameters compared to single decision trees [1].

2.2.2 Neural Network Model. In recent years, the remarkable success of neural networks (NNs) has captured considerable attention across diverse domains, ranging from medical imaging [19, 22, 38, 43] to astrophysics [15, 18, 45, 46], and roadway safety [17, 29, 40]. Inspired by their impressive capabilities, we also incorporated neural networks as one of the classification models in our SQL injection detection work.

Table 1: Details of the Neural Network Architecture

Layer	CountVectorizer	RoBERTa
Input	40,933	768
FC1	256 ReLU Batch Norm	128 ReLU Batch Norm
FC2	256 Dropout (p=0.2) ReLU Batch Norm	128 Dropout (p=0.2) ReLU Batch Norm
FC3	2	2
Parameter Count	10,546,434	115,714

In this study, we designed and trained neural networks for feature learning and decision making. This process involves passing the data through multiple layers of interconnected neurons, where each layer extracts increasingly abstract representations of the input. For our specific problem, we employed a deep feedforward neural network, commonly known as a multi-layer perceptron (MLP) model. The MLP consists of an input layer that takes the textual input x and processes it through hidden layers, ultimately leading to the output layer that predicts the probability of SQL injection ($\hat{y}$). The choice of the number of hidden layers and the number of neurons in each layer was guided by experimentation to strike the right balance between model complexity and generalization. The specific structures of two NN models used in this study are shown in Table 1.

To train the neural network, we utilized a supervised learning approach with labeled training samples $\{x, y\}$. During training, the network learns the optimal weights and biases by minimizing the binary cross-entropy loss (BCE) [5]:

$$BCE = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(\hat{p}_i) + (1-y) \cdot (log(1-\hat{p}_i). \qquad (3)$$

Through rigorous experimentation and performance evaluation, we compared the neural network's performance with the other classification algorithms (K-Nearest Neighbors, Logistic Regression, and Random Forest). This comprehensive analysis allowed us to determine which model and embedding method achieved the most robust and accurate SQL injection detection results.

3 EXPERIMENTS AND RESULTS

3.1 Experiment Setup

This study was conducted using Google Colaboratory with 12GB of RAM and a Nvidia Tesla T4 GPU with 16GB of memory. The `CountVectorizer` method from the `scikit-learn` library [3] served as the BoW embedding method. The `RobertaModel`, Specifically `roberta-base`, from HuggingFace [35] was utilized for contextualized embedding.

The SQL queries were directly fed into `CountVectorizer` or `roberta-base` for the embedding generation. No data prepossessing was performed on the raw SQL queries.

Table 2: The Embedding Space of Different Methods

CountVectorizer	RoBERTa
40,933	768

Table 3: Details of the Dataset

Dataset Size	64,335
Positive/Negative Samples	22,763 / 41,572
Vocabulary Size	40,933
Train/Val/Test Split	46,482 / 8,203 / 9,650

The BoW embedding was fitted on the training and validation sets, resulting in an embedding space of 40,933. For the contextualized embedding, we directly used the HuggingFace pre-trained weights of RoBERTa, resulting in a feature space of 768 for each sample (Table 2).

To compare the performance of the two embedding methods, we trained eight machine learning models using four classification algorithms: logistic regression, random forest, k-nearest neighbor, and multi-layer perceptron (neural network). The logistic regression, random forest, and KNN models were directly loaded from the `scikit-learn` library and trained on the CPU. In contrast, the multi-layer perceptron models were implemented using the `PyTorch` library [26] and trained on the GPU. We use the default parameters for the logistic regression model; we set the $max\_depth = 5$ and $leaf\_size = 15$ for random forest; and we set $n\_neighbors = 25$ for the KNN model. The setup for the multi-layer perceptron models is detailed in Table 1.

3.2 Dataset

A SQLi dataset from Kaggle[1] is used in this project. The dataset consists of three subsets, totaling $64,645$ samples. After combining the subsets, we identified and removed 310 samples that lacked proper labels, resulting a dataset of 64,335 labeled instances with $41,572$ being negative cases and $22,763$ being positive cases (Table 3). We randomly divided the dataset into training, validation, and test sets, approximately in a $72 : 13 : 15$ ratio.

The dataset consists of 40,933 unique tokens (words). The samples in the dataset exhibit a wide range of lengths, varying from 1 token to 5,370 tokens, with the majority of samples being less than 100 tokens in length. Figure 2 depicts the histograms of the sample length distribution, displaying both the probability density function (PDF) and cumulative distribution function (CDF).

3.3 Evaluation Metrics

Given the binary classification nature of the task, we evaluate model performance using accuracy (Acc), F1 score (F1), Precision, and Recall. The accuracy is calculated with the following:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \qquad (4)$$

[1]https://www.kaggle.com/datasets/syedsaqlainhussain/sql-injection-dataset

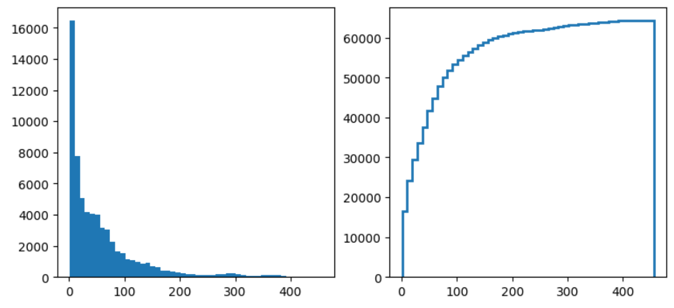

Figure 2: Histograms of the Sample Length Distribution

where TP is the True Positive, TN is the True Negative, FP is the False Positive, and FN is the False Negative.

The precision is calculated by the following:

$$precision = \frac{TP}{TP + FP}. \tag{5}$$

The recall is calculated by the following:

$$recall = \frac{TP}{TP + FN}. \tag{6}$$

The F1-score is calculated by the following:

$$F1 = 2 \times \frac{precision \cdot recall}{precision + recall}. \tag{7}$$

In addition, we also assess the model's training efficiency in seconds and model calibration using reliability diagrams and mean calibration error.

Reliability diagrams are graphical representations of the calibration quality of a classification model. These diagrams divide the predicted probabilities into bins and plot the average predicted probability against the empirical probability (the actual proportion of positive outcomes) for each bin. An ideal calibration is reflected by points close to the diagonal line on the diagram [7]. Mean calibration error is calculated as the difference between the model accuracy and the predicted mean probability based on Equation 2, i.e., the mean differences between the bins and the diagonal line for a diagram.

3.4 SQLi Detection Performance

Table 4 presents the performance of the eight SQLi detection models. The models using contextualized embedding (RoBERTa) exhibit consistent performance, achieving accuracy above 99%, with the highest accuracy of 99.61% attained by logistic regression. It is possible to still pushing the performance higher by tuning the hyperparameters or increasing the model complexity of the multi-layer perceptron and random forest models. Since all models already achieve accuracy above 99%, we decided not to focus on pushing the accuracy further, as it would not yield substantial practical gains.

On the other hand, the table reveals significant disparities in the performance of the BoW embedding (CountVectorizer) models. While the multi-layer perceptron model achieves above 99% accuracy, other models show considerable variation. For instance, the random forest model achieves only 67.62% accuracy, barely surpassing random guessing. Precision and recall scores highlight that the random forest model is heavily biased towards negative classes. We attribute this performance drop to the large BoW embedding space size, which drastically increases computation requirements and necessitates more complex models for effective classification.

Another drawback of the BoW embedding is the large embedding space increases the training time complexity dramatically. For instance, it only takes 6 seconds to train a multilayer perceptron model using contextualized embedding for 10 epochs. However, the time increases over 31 times when training a multilayer perceptron model to achieve a similar performance using the BoW embedding. In addition, the large embedding space also requests a unanimous amount of computational power and high memory consumption. In fact, the training of logistic regression and KNN cannot be completed due to the high memory consumption

3.5 Neural Network Calibration

Recent advances in deep neural networks have had a profound impact on numerous research domains, such as [30, 44]. While researchers continuously strive to achieve higher classification model performance, uncertainty quantification is often overlooked. Nevertheless, quantifying uncertainty in neural networks is crucial since they are often overconfidence in their prediction [7, 16, 27], which is known as *miscalibration* that may lead to significant consequences in the real-world [10, 13].

Figure 3 shows the reliability diagrams of the CountVectorizer trained model (left), the RoBERTa trained model (middle), and the direct comparison of the two models (right). Ideally, all the bins in a reliability diagram should align on the diagonal line, indicating perfect calibration. While none of the models achieve perfect calibration, the model trained with CountVectorizer performs worse than the RoBERTa model. Specifically, it severely underestimates the probabilities of samples falling within the median probability bins (i.e., bins of 0.5 and 0.6).

To quantify the calibration performance, we computed the mean calibration error, which is the mean differences between the bins and the diagonal line for each diagram. The results show that the RoBERTa trained model has a mean calibration error of 0.1664, while the CountVectorizer trained model has a mean calibration error of 0.1925, which is about 16% higher than its counterpart. Both the reliability diagrams and mean calibration error indicate that the RoBERTa model achieves better calibration compared to the CountVectorizer model.

4 DISCUSSION AND CONCLUSION

In this study, we compared the performance of SQL injection (SQLi) detection models using both contextualized and non-contextualized word embeddings. Specifically, we utilized the RoBERTa model as our contextualized embedding method and the Bag-of-Words as our non-contextualized method, training eight machine learning detection models. The results demonstrated the superiority of RoBERTa, achieving consistent performance with accuracy above 99% across various classification algorithms, while significantly reducing model training time by 31 times.

The analysis of the reliability diagrams further revealed that RoBERTa's contextualized embeddings provided better calibration compared to CountVectorizer's embeddings. The RoBERTa trained

Table 4: Model Performance Across Different Classification Algorithms and Embedding Methods

Embedding Method	ML Model	Accuracy	F1 Score	Precision	Recall	Training Time
CountVectorizer	**Multilayer Perceptron**	0.9907	0.9870	0.9891	0.9850	194
	Random Forest	0.6762	0.1510	1.0000	0.0817	211
	Logistic Regression	Models are not trainable due to high memory consumption.				
	K-Nearest Neighbor					
RoBERTa	**Multilayer Perceptron**	0.9907	0.9870	0.9901	0.9840	6
	Random Forest	0.9801	0.9712	0.9923	0.9509	59
	Logistic Regression	0.9961	0.9944	0.9979	0.9909	8
	K-Nearest Neighbor	0.9965	0.9950	0.9982	0.9918	40

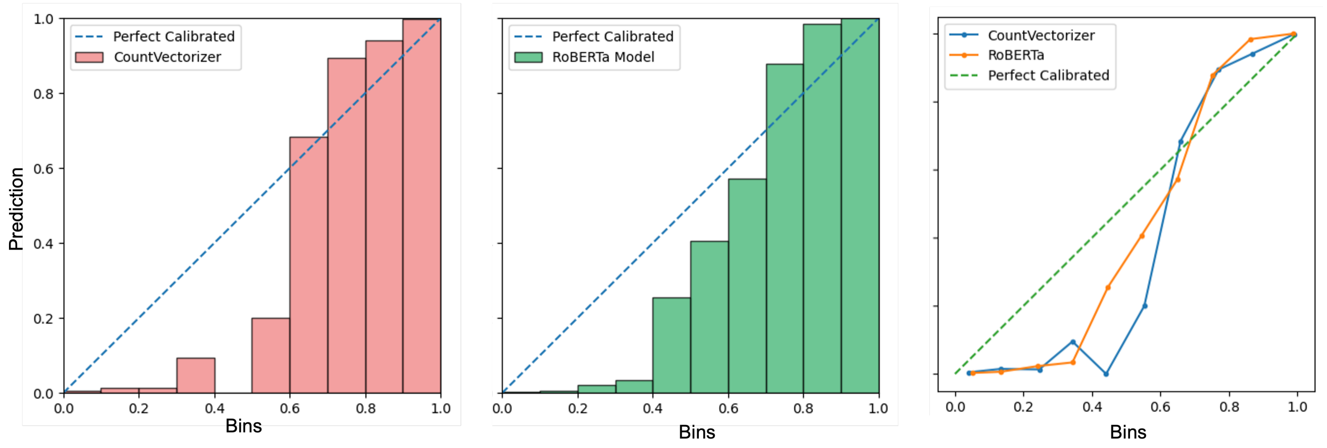

Figure 3: Reliability Diagrams of the Multilayer Perception Models with CountVectorizer Embedding (Left), RoBERTa Embedding (Middle), and a Direct Comparison (Right)

model exhibited closer alignment of the reliability diagram bins to the diagonal line, indicating more accurate probability predictions.

Our findings emphasize the importance of contextualized word embeddings, such as RoBERTa, in enhancing the performance and reliability of SQLi detection models. RoBERTa's ability to capture contextual information allows it to effectively handle nuances in word meaning and disambiguate polysemous words, leading to higher accuracy and better calibration compared to the non-contextualized CountVectorizer.

However, we acknowledge that the time complexity evaluated in this study focuses solely on model training and does not consider the pre-training process of the contextualized embedding method. Additionally, due to RoBERTa's larger model size, its embedding process may require more computational resources than Bag-of-Words.

In future work, it would be beneficial to explore methods that mitigate the computational overhead of contextualized embeddings, making them more feasible for practical applications. Furthermore, investigating other state-of-the-art contextualized embedding techniques and their impact on the performance and calibration of SQLi detection models could provide valuable insights.

Overall, this study contributes to our understanding of the role of word embeddings in SQLi detection and underscores the significance of considering both accuracy and uncertainty quantification when designing automated decision-making systems.

REFERENCES

[1] Gérard Biau. 2012. Analysis of a Random Forests Model. *The Journal of Machine Learning Research* 13 (2012), 1063–1095.
[2] Leo Breiman. 2001. Random Forests. *Machine learning* 45 (2001), 5–32.
[3] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, et al. 2013. API Design for Machine Learning Software: Experiences from the Scikit-Learn Project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. Prague, Czech Republic, 108–122.
[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186.
[5] Irving John Good. 1952. Rational Decisions. *Journal of the Royal Statistical Society: Series B (Methodological)* 14, 1 (1952), 107–114.
[6] Jesus Guerrero, Gongbo Liang, and Izzat Alsmadi. 2023. Adversarial Text Perturbation Generation and Analysis. In *2023 3rd Intelligent Cybersecurity Conference (ICSC)*. IEEE, San Antonio, USA, 67–73.
[7] Chuan Guo, Geoff Pleiss, Yu Scn, and Kilian Q Weinberger. 2017. On Calibration of Modern Neural Networks. In *The Thirty-fourth International Conference on Machine Learning*. Sydney, Australia, 1321–1330.
[8] William G Halfond, Jeremy Viegas, Alessandro Orso, et al. 2006. A Classification of SQL-Injection Attacks and Countermeasures. In *Proceedings of the IEEE international symposium on secure software engineering*, Vol. 1. IEEE, Washington DC,

USA, 13–15.

[9] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. 2013. *Applied Logistic Regression*. Vol. 398. John Wiley & Sons.

[10] Xiaoqian Jiang, Melanie Osl, Jihoon Kim, and Lucila Ohno-Machado. 2011. Calibrating Predictive Model Estimates to Support Personalized Medicine. *Journal of the American Medical Informatics Association* 19, 2 (2011), 263–274.

[11] Krishna Juluru, Hao-Hsin Shih, Krishna Nand Keshava Murthy, and Pierre Elnajjar. 2021. Bag-of-Words Technique in Natural Language Processing: A Rrimer for Radiologists. *RadioGraphics* 41, 5 (2021), 1420–1426.

[12] Kazuma Kobayashi, Mototaka Miyake, Masamichi Takahashi, and Ryuji Hamamoto. 2021. Observing Deep Radiomics for the Classification of Glioma Grades. *Scientific Reports* 11, 1 (2021), 1–13.

[13] Aviral Kumar, Sunita Sarawagi, and Ujjwal Jain. 2018. Trainable Calibration Measures for Neural Networks from Kernel Mean Embeddings. In *The Thirty-fourth International Conference on Machine Learning*. Stockholm, Sweden, 2810–2819.

[14] Qi Li, Fang Wang, Junfeng Wang, and Weishi Li. 2019. LSTM-Based SQL Injection Detection Method for Intelligent Transportation System. *IEEE Transactions on Vehicular Technology* 68, 5 (2019), 4182–4191.

[15] Gongbo Liang, Yuanyuan Su, Sheng-Chieh Lin, Yu Zhang, Yuanyuan Zhang, and Nathan Jacobs. 2020. Optical Wavelength Guided Self-Supervised Feature Learning for Galaxy Cluster Richness Estimate. In *Neural Information Processing Systems (NeurIPS) Workshop on Machine Learning and the Physical Sciences*. Virtual.

[16] Gongbo Liang, Yu Zhang, Xiaoqin Wang, and Nathan Jacobs. 2020. Improved Trainable Calibration Method for Neural Networks on Medical Imaging Classification. In *British Machine Vision Conference (BMVC)*. Manchester, England.

[17] Gongbo Liang, Janet Zulu, Xin Xing, and Nathan Jacobs. 2023. Unveiling Roadway Hazards: Enhancing Fatal Crash Risk Estimation Through Multiscale Satellite Imagery and Self-Supervised Cross-Matching. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 17 (2023), 535–546.

[18] Sheng-Chieh Lin, Yuanyuan Su, Gongbo Liang, Yuanyuan Zhang, Nathan Jacobs, and Yu Zhang. 2022. Estimating Cluster Masses from SDSS Multiband Images with Transfer Learning. *Monthly Notices of the Royal Astronomical Society* 512, 3 (2022), 3885–3894.

[19] Liangliang Liu, Ying Wang, Jing Chang, Pei Zhang, Gongbo Liang, and Hui Zhang. 2022. LLRHNet: Multiple Lesions Segmentation Using Local-Long Range Features. *Frontiers in Neuroinformatics* 16 (2022), 859973.

[20] Liangliang Liu, Pei Zhang, Gongbo Liang, Shufeng Xiong, Jianxin Wang, and Guang Zheng. 2023. A Spatiotemporal Correlation Deep Learning Network for Brain Penumbra Disease. *Neurocomputing* 520 (2023), 274–283.

[21] Srishti Lodha and Atharva Gundawar. 2022. SQL Injection and Its Detection Using Machine Learning Algorithms and BERT. In *International Conference on Cognitive Computing and Cyber Physical Systems*. Springer, 3–16.

[22] Radu Paul Mihail, Gongbo Liang, and Nathan Jacobs. 2019. Automatic Hand Skeletal Shape Estimation from Radiographs. *IEEE Transactions on Nanobioscience* 18, 3 (2019), 296–305.

[23] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. *Advances in Neural Information Processing Systems* 26 (2013).

[24] Antonio Mucherino, Petraq J Papajorgji, Panos M Pardalos, Antonio Mucherino, Petraq J Papajorgji, and Panos M Pardalos. 2009. K-Nearest Neighbor Classification. *Data Mining in Agriculture* (2009), 83–106.

[25] Mohammed Nasereddin, Ashaar Alkhamaiseh, Malik Qasaimeh, and Raad Al-Qassas. 2021. A Systematic Review of Detection and Prevention Techniques of SQL Injection Attacks. *Information Security Journal: A Global Perspective* (2021), 1–14.

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, et al. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., Vancouver, Canada, 8024–8035.

[27] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing Neural Networks by Penalizing Confident Output Distributions. *arXiv:1701.06548* (2017).

[28] S Pooja, CB Chandrakala, and Laiju K Raju. 2022. Developer's Roadmap to Design Software Vulnerability Detection Model Using Different AI Approaches. *IEEE Access* 10 (2022), 75637–75656.

[29] Weilian Song, Scott Workman, Armin Hadzic, Xu Zhang, Eric Green, Mei Chen, Reginald Souleyrette, and Nathan Jacobs. 2018. Farsa: Fully Automated Roadway Safety Assessment. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Lake Tahoe, USA, 521–529.

[30] Jiawei Tang, Fengbo Zheng, Gongbo Liang, and Lifen Jiang. 2023. Utilize Multichannel Attention Amplification Fusion for Skin Disease Diagnosis. In *Fifth International Conference on Computer Information Science and Artificial Intelligence (CISAI 2022)*, Vol. 12566. SPIE, Chongqing, China, 853–858.

[31] Peng Tang, Weidong Qiu, Zheng Huang, Huijuan Lian, and Guozhen Liu. 2020. Detection of SQL Injection Based on Artificial Neural Network. *Knowledge-Based Systems* 190 (2020), 105528.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. 2017. Attention is All You Need. In *Advances in Neural Information Processing Systems*, Vol. 30. Long Beach, USA.

[33] Tomasz Walkowiak, Szymon Datko, and Henryk Maciejewski. 2019. Bag-of-Words, Bag-of-Topics and Word-to-Vec Based Subject Classification of Text Documents in Polish – A Comparative Study. In *Contemporary Complex Systems and Their Dependability: Proceedings of the Thirteenth International Conference on Dependability and Complex Systems DepCoS-RELCOMEX, July 2-6, 2018, Brunów, Poland 13*. Springer, 526–535.

[34] Xiaoqin Wang, Gongbo Liang, Yu Zhang, Hunter Blanton, Zachary Bessinger, and Nathan Jacobs. 2020. Inconsistent Performance of Deep Learning Models on Mammogram Classification. *Journal of the American College of Radiology* 17, 6 (2020), 796–803.

[35] Thomas Wolf et al. 2019. HuggingFace's Transformers: State-of-the-Art Natural Language Processing. *arXiv:1910.03771* (2019).

[36] Raymond E Wright. 1995. Logistic Regression. (1995).

[37] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, Philip S Yu, et al. 2008. Top 10 Algorithms in Data Mining. *Knowledge and Information Systems* 14 (2008), 1–37.

[38] Xin Xing, Gongbo Liang, Yu Zhang, Subash Khanal, Ai-Ling Lin, and Nathan Jacobs. 2022. Advit: Vision Transformer on Multi-Modality PET Images for Alzheimer Disease Diagnosis. In *2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI)*. IEEE, Kolkata, India, 1–4.

[39] Xin Xing, Muhammad Usman Rafique, Gongbo Liang, Hunter Blanton, Yu Zhang, Chris Wang, Nathan Jacobs, and Ai-Ling Lin. 2023. Efficient Training on Alzheimer's Disease Diagnosis with Learnable Weighted Pooling for 3D PET Brain Image Classification. *Electronics* 12, 2 (2023), 467.

[40] Zhexiao Xiong, Feng Qiao, Yu Zhang, and Nathan Jacobs. 2023. StereoFlowGAN: Co-training for Stereo and Flow with Unsupervised Domain Adaptation. In *British Machine Vision Conference (BMVC)*. Aberdeen, Scotland.

[41] Zhuo Xu. 2021. RoBERTa-wwm-ext Fine-Tuning for Chinese Text Classification. *arXiv preprint arXiv:2103.00492* (2021).

[42] Guang Yang, Yanlin Zhou, Chi Yu, and Xiang Chen. 2021. DeepSCC: Source Code Classification Based on Fine-Tuned RoBERTa. *arXiv preprint arXiv:2110.00914* (2021).

[43] Qi Ying, Xin Xing, Liangliang Liu, Ai-Ling Lin, Nathan Jacobs, and Gongbo Liang. 2021. Multi-Modal Data Analysis for Alzheimer's Disease Diagnosis: An Ensemble Model Using Imagery and Genetic Features. In *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. IEEE, Guadalajara, Mexico, 3586–3591.

[44] Yu Zhang, Gongbo Liang, and Nathan Jacobs. 2022. Dynamic Feature Alignment for Semi-Supervised Domain Adaptation. In *British Machine Vision Conference (BMVC)*. London, England.

[45] Yu Zhang, Gongbo Liang, Yuanyuan Su, and Nathan Jacobs. 2021. Multi-Branch Attention Networks for Classifying Galaxy Clusters. In *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, Milan, Ital, 9643–9649.

[46] JA ZuHone, DJ Barnes, NB Jacobs, WR Forman, PEJ Nulsen, RP Kraft, et al. 2020. A Deep Learning View of the Census of Galaxy Clusters in Illustristng. *Monthly Notices of the Royal Astronomical Society* 498, 4 (2020), 5620–5628.