NULLDect: A Dynamic Adaptive Learning Framework for Robust NULL Pointer Dereference Detection

Tasmin Karim[1][a], Md. Shazzad Hossain Shaon[1][b], Md. Fahim Sultan[1][c], Alfredo Cuzzocrea[2,3,*][d] and Mst Shapna Akter[1][e]

[1]*Department of Computer Science and Engineering, Oakland University, Rochester, MI 48309, U.S.A.*

[2]*iDEA Lab, University of Calabria, Rende, Italy*

[3]*Department of Computer Science, University of Paris City, Paris, France*

Keywords: GloVe Embeddings, Long Short-Term Memory, NULL Pointer Dereference, Adaptive Learning.

Abstract: The identification of null pointer dereference vulnerabilities has implications for software security and reliability, as well as satisfying market needs for user data protection. This study introduces NULLDect, an adaptive learning-based approach that addresses this issue using the CWE-476 (NULL Pointer Dereference) dataset. Such detection becomes essential for averting software failures and unforeseen events that could compromise system stability and security. The proposed approach combines the uses of Long-Short-Term Memory (LSTM) networks, attention mechanisms, and adaptive learning with callback techniques to produce a phenomenal accuracy rate of 0.806 by extracting features utilizing the CodeT5 paradigm. Furthermore, the work incorporates and evaluates advanced computational models, including CodeT5, BERT, UniXcoder, and NLP-based GloVe embeddings, to discover the most successful strategy for null pointer detection across many evaluation metrics. This adaptability improves model accuracy, robustness, and longevity. NULLDect's synergistic combination of approaches defines it as a comprehensive and effective solution for detecting and mitigating NULL pointer dereference problems.

1 INTRODUCTION

The fundamental objective of software development is to produce software of the highest standard in a manner that is efficient and affordable. Reuse of code is an essential and widely recognized method of achieving this objective Abdalkareem et al. (2017). Therefore, preserving software systems has become essential for preventing violations that might result in substantial financial damage, a negative reputation, and litigation Security Importance (2024). Model validation is a standard verification approach that uses quantitative and logical confirmation to determine if a system satisfies particular criteria Byun et al. (2020a).

[a] https://orcid.org/0009-0008-2042-2610
[b] https://orcid.org/0009-0008-2491-1622
[c] https://orcid.org/0009-0009-2550-257X
[d] https://orcid.org/0000-0002-7104-6415
[e] https://orcid.org/0000-0002-9859-6265

*This research has been made in the context of the Excellence Chair in Big Data Management and Analytics at University of Paris City, Paris, France.

The checking of models might prove highly beneficial when dealing with NULL Pointer Dereference (NPD), including in CWE-476. Since 2019, NPD has been listed in the "CWE Top 25 Most Dangerous Software Weaknesses" presented on the CWE portal CWE Ranking (2020). In some circumstances, attackers can use this vulnerability to run arbitrary code or circumvent authentication procedures. The prediction thoroughly evaluates all program states to verify that pointers are never dereferenced while they are NULL, hence avoiding potential vulnerabilities. This strategy supports identifying and eliminating safety concerns early in the development process, lowering the probability of run-time mistakes that might result in system failures or security breaches. .

This present study represents some innovative contributions to detecting the NPD with the CWE-476 dataset. In this work, an adaptive computational approach was used that included LSTM, attention mechanisms, and adaptive learning with callback techniques called as NULLDect.The NULLDect model provides various benefits for identifying Null pointer dereference difficulties. First,

Karim, T., Shaon, M. S. H., Sultan, M. F., Cuzzocrea, A. and Akter, M. S.
NULLDect: A Dynamic Adaptive Learning Framework for Robust NULL Pointer Dereference Detection.
DOI: 10.5220/0013494400003979
In *Proceedings of the 22nd International Conference on Security and Cryptography (SECRYPT 2025)*, pages 571-576
ISBN: 978-989-758-760-3; ISSN: 2184-7711

LSTM improves the model's ability to grasp long-term dependencies in data, which increases detection accuracy. The attention mechanism enables the model to focus on the most important features, hence improving performance. Adaptive learning with callback approaches ensures that NULLDect develops to handle new patterns and threats while remaining effective over time. This combination yields a highly precise, adaptive, and robust model that ensures the reliable detection of potential software vulnerabilities.

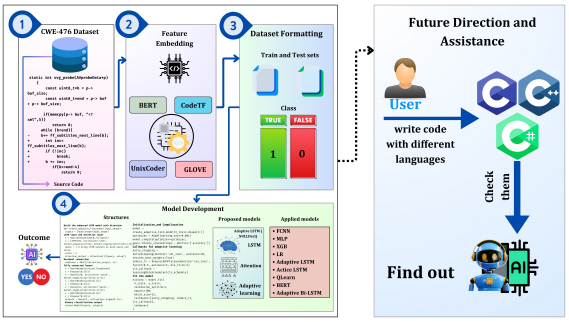

Figure 1: General methodology of the proposed strategy for operation: The whole procedure of our recommended approach for identifying Null Pointer Dereferences (NPD).

2 RELATED WORK

According to the previous study, Lu et al. developed GRACE, a tool that enhances large language models (LLMs) that utilize graph structure information to comprehend source code context better Lu et al. (2024). In another paper, Wang et al. generated the DefectHunter method, which uses convolutional networks and self-attention methods to discover vulnerabilities in software with great precision Wang et al. (2023). Akuthota et al. proposed a pretrained GPT-3.5-Turbo model that utilizes enhanced language-generating capabilities for various applications Akuthota et al. (2023). Jin et al. proposed the Npdhunter framework for NPD detection in binary Jin et al. (2021). Gotovchits et al. introduced the Saluki technique, which uses static property checking to detect taint-style vulnerabilities in software systems Gotovchits et al. (2018). In another study, Byun et al. recommended a method for identifying vulnerabilities in software using CBMC (C Bounded Model Checker) and the CWE framework Byun et al. (2020b). This method includes using CBMC to systematically check for vulnerabilities by mapping them to known flaws cataloged in CWE. Lipp et al. conducted empirical research to assess the usefulness of static C code analyzers in discovering vulnerabilities. Their research compares multiple static analysis techniques to C codebases to measure their accuracy, coverage, and efficiency in detecting security issues. The work sheds light on the strengths and limits of various analyzers, guiding future enhancements and selection criteria for C vulnerability identification.

Bojanova et al. demonstrated a methodology for identifying memory issues based on the issues methodology approach. This approach entails identifying and assessing memory-related issues in software systems using a systematic framework that assigns defects based on their features and impacts Bojanova and Galhardo (2021). Alqaradaghi et al. presented an experiment to evaluate which static analysis tool is most successful at identifying null pointer dereferences in Java code Alqaradaghi and Kozsik (2022). Sandoval et al. introduced the "Lost at C" technique, which analyzes the security consequences of utilizing LLM code assistants. Their research looks into how LLMs, when applied to C programming, might cause or fail to discover security issues Sandoval et al. (2023). In another study, Choi et al. proposed Graph Neural Network (GNN) model for the detection of error in null pointers Choi and Kwon (2022). Furthermore, multiple studies Alon et al. (2018, 2019) explored various strategies to improve the efficiency of source code representation in model analysis based on machine learning. These research aims to reduce information loss during the representation learning process by employing Abstract Syntax Tree (AST)-based encoding methods.

Building on prior findings, this study delved deeper into the CWE-476 dataset to address issues in detecting NULL Pointer Dereference (NPD) vulnerabilities. We offer an alternate technique that uses CWE information to improve detection reliability and efficiency. Our solution makes use of a model that is optimized for computational speed and resource utilization. By incorporating CWE samples, which provide a complete repository of known vulnerabilities, the model increases its capacity to discover NPD concerns with better precision and efficiency. The contributions of the NULLDect framework are as follows:

1. Extensive evaluations were performed utilizing many advanced computational models, including CodeT5, BERT, UniXcoder, and GloVe-based NLP embeddings, to determine the best effective strategies for NPD identification.

2. Combining LSTM, attention mechanisms, and callback-based adaptive learning to detect NULL pointer dereference (NPD) vulnerabilities efficiently.

3 MATERIALS AND METHODS

3.1 Dataset Descriptions

As we delve into NPD identification, we begin utilizing publicly accessible datasets that have been used in previous studies. To help with our study and increase detection accuracy, we used the CWE-476 dataset, which focuses on vulnerabilities related to NPD. We have collected 24,492 samples from the "VDISC Dataset", where the dataset is designed for vulnerability detection in source code VDISC Dataset (2024). The dataset includes two labels: "True" and "False," where "True" is represented by 1, indicating the presence of a vulnerability, and "False" is represented by 0, indicating no vulnerability. These textual and category inputs were converted into numerical representations using various embedding approaches, which allowed the machine learning model to handle the data efficiently for vulnerability identification Guo et al. (2022); Wang et al. (2022); Pennington et al. (2014). In Table. 1 overall dataset description are depicted for better understanding.

3.2 Overview of the Proposed Methodology

The proposed approach for detecting NPD vulnerabilities is both easy to follow, with great room for further development. While past research have been commendably accurate, many attempt to present routes for furthering their conclusions. In contrast, our approach creates a meta-model framework that will be a useful resource for future experts. Figure 1 illustrates every step of the methods, including model building and dataset preparation. Using a large dataset, we used many feature embedding approaches, including CodeT5, UniXcoder, BERT, and GloVe. These embeddings were tested using nine computational algorithms: Logistic Regression (LR), Multi-Layer Perceptron (MLP), Extreme Gradient Boosting (XGB), Fully Connected Neural Networks (FCNN), Active LSTM, Q-Learning, Bidirectional Encoder Representations from Transformers (BERT), Adaptive Bi-LSTM, and our proposed method, Adaptive LSTM, that is known as NULLDect LaValley (2008); Chen (2015); Pratt et al. (2017); Greff et al. (2016); Koroteev (2021a); Jang et al. (2019); Koroteev (2021b).

3.3 Feature Embedding

Feature embedding approaches have become essential for translating raw data into meaningful numerical representations that machine learning models can process effectively. Therefore, this study used CodeT5, UniXcoder, BERT and GloVe Bilgin et al. (2020) methods. These embedding methods offer a variety of ways of encoding features, each with a particular set of advantages. CodeT5 and UniXcoder excel in capturing domain-specific nuances of programming languages, whereas BERT relies on deep bidirectional context. GloVe, on the other hand, provides a simpler, more computationally efficient method for producing embeddings.

Table 1: Distribution of total, true, and false sets in the train and test datasets.

Dataset	Total Set	True Set	False Set
Train Set	18880	9440	9440
Test Set	5612	2810	2802

3.4 Applied Models

A variety of machine learning and deep learning models were used, each with their own set of skills to detect vulnerabilities in NPD. LR is a statistical model designed for binary categorization. Predicts the likelihood of a target variable using a logistic function on a linear combination of input information. In another, MLP is a feedforward neural network with input, hidden, and output layers. It employs backpropagation for training, allowing it to detect complex nonlinear correlations in data. This study used an ensemble-based model as XGB, which is a treebased ensemble learning method that enhances prediction accuracy via iterative boosting. Unlike the first, FCNNs are dense neural networks that connect all nodes in each layer to the next. They excel at capturing complicated feature interactions in datasets. In another, Active LSTM improves traditional LSTM networks by using active learning methodologies. LSTMs are intended to capture sequential dependencies and long-term relationships in data, making them ideal for time-series or ordered datasets. For adaptive learning, Q-Learning is a reinforcement learning algorithm that optimizes decision-making by learning policies that maximize cumulative rewards. It is useful for challenges that need sequential decision-making or adaptive techniques. BERT is a deep bidirectional transformer architecture that captures context from both directions in sequential data. This makes it especially effective for understanding subtle relationships in textual or tokenized information.

3.5 Proposed Approach: NULLDect Framework

The proposed model is an adaptive LSTM-based architecture with an The proposed model is an adaptive LSTM-based architecture with an integrated attention mechanism. To improve performance, the design uses a variety of advanced techniques such as spatial dropout, attention layers, residual connections, and adaptive learning procedures. The architecture starts with a 256-unit LSTM layer that captures sequential dependencies, followed by a spatial dropout to prevent overfitting. An attention layer concentrates on the most important sequence elements, enhancing interpretability and performance. Attention output is mixed with LSTM output through a residual link to stabilize learning. The model consists of fully linked layers with swish activation and dropout for regularization, culminating in a sigmoid output layer for prediction. It employs the Adam optimizer, binary cross-entropy loss, and callbacks such as early halting and learning rate adaption to facilitate training. This method provides reliable detection with higher accuracy and versatility.

The model accepts input sequences with a shape corresponding to the dataset features. Then, LSTM layer of 256 units processes sequential data. We uses the *tanh* activation function and *L2* regularization to avoid overfitting. The outputs of the LSTM layer pass through Spatial dropout and Attention layer, where a spatial dropout layer reduces overfitting by randomly dropping entire feature maps rather than individual elements and the attention mechanism calculates the relevance of each time-step. Additionally the residual connection adds attention outputs to the original LSTM outputs and the attention-enhanced sequence representation is passed through three dense layers. A 128-unit layer with *swish* activation:

$$\text{swish}(x) = x \cdot \sigma(x), \tag{1}$$

where $\sigma(x)$ is the sigmoid function. A 64-unit layer with *swish* and *sigmoid activation* for binary classification.where $\hat{y}$ is the predicted probability:

$$\hat{y} = \sigma(W \cdot x + b), \tag{2}$$

Then we used batch normalization and Dropout layers with a 0.5 rate reduce over fitting by randomly deactivating neurons during training. To enhance training phase we use callbacks which halts training when the validation loss does not improve for 10 consecutive epochs. Then, reduces the learning rate by a factor of 0.5 if validation loss stagnates for 5 epochs. All the workflows of the model development has been illustrate in Fig. 2

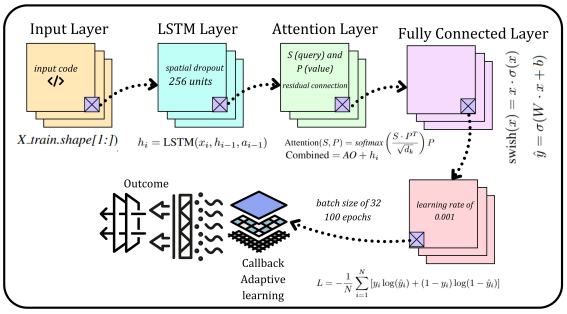

Figure 2: Working Procedure of the NULLDect model.

4 EXPERIMENTAL ANALYSIS

This study utilizes a variety of assessment criteria. These criteria were used to assess the efficacy and dependability of the proposed framework. Table 2 compares the performance of several models among extractors, including BERT, CodeT5, UnixCoder, and Glove, utilizing important metrics. CodeT5 emerges as the best extractor, with top scores for Adaptive LSTM (accuracy: 0.806, F1: 0.814), demonstrating its capacity to capture patterns effectively. BERT performs well, particularly with Adaptive LSTM (accuracy: 0.789), but simpler models, such as FCNN, perform poorly. UnixCoder produces competitive results, particularly with BERT (accuracy: 0.799, F1: 0.802), although its recall is significantly worse than CodeT5. The glove underperforms overall, with a top accuracy of only 0.727. CodeT5 with Adaptive LSTM is the best-performing combination.

4.1 Ablation Study

This work used an ablation strategy to verify the Enhanced LSTM model with Attention, as demonstrated in Table 3. Removing the Attention layer dramatically decreased performance, emphasizing its importance in concentrating on crucial characteristics. Reducing LSTM units decreased accuracy, highlighting the requirement for enough capacity to record complicated patterns. Omitting residual connections resulted in slower convergence and decreased performance, highlighting their necessity for training stability. The Base Model frequently outperformed others, demonstrating the usefulness of combining these components for precise and efficient detection. The Base Model (NULLDect) has the highest accuracy (0.818) and F1 score (0.817). Removing the attention mechanism or residual connections and lowering LSTM units reduced performance, with accuracy stabilizing at 0.799. This underlines the importance of these components.

Table 2: Performance comparison of different models and extractors in NPD dataset.

Extractor	Model	Acc.	Pre.	Rec.	F1	Sn	Sp	AUC
BERT	LR	0.762	0.772	0.764	0.772	0.762	0.753	0.700
	XGB	0.744	0.752	0.744	0.743	0.745	0.732	0.765
	MLP	0.745	0.745	0.775	0.761	0.639	0.678	0.751
	BERT	0.747	0.754	0.767	0.754	0.744	0.752	0.658
	FCNN	0.657	0.635	0.639	0.627	0.635	0.639	0.752
	Active LSTM	0.738	0.767	0.678	0.720	0.761	0.639	0.752
	Adaptive LSTM	**0.789**	**0.782**	**0.779**	**0.751**	**0.751**	**0.744**	**0.743**
	Adaptive BiLSTM	0.775	0.762	0.770	0.759	0.744	0.743	0.752
	Q-learning	0.758	0.715	0.742	0.742	0.755	0.654	0.743
CodeT5	LR	0.780	0.774	0.757	0.775	0.767	0.732	0.750
	XGB	0.792	0.808	0.780	0.794	0.768	0.740	0.770
	MLP	0.800	0.797	0.819	0.808	0.772	0.750	0.780
	FCNN	0.756	0.710	0.718	0.718	0.739	0.700	0.760
	BERT	0.802	0.796	0.827	0.814	0.755	0.732	0.785
	Active LSTM	0.805	0.814	0.814	0.806	0.760	0.720	0.800
	Adaptive LSTM	**0.806**	**0.814**	**0.804**	**0.814**	**0.778**	**0.750**	**0.790**
	Adaptive BiLSTM	0.797	0.801	0.795	0.795	0.762	0.740	0.770
	Q-learning	0.780	0.774	0.784	0.774	0.745	0.732	0.750
UnixCoder	LR	0.754	0.765	0.742	0.754	0.739	0.710	0.750
	XGB	0.791	0.812	0.782	0.792	0.744	0.700	0.765
	MLP	0.792	0.814	0.778	0.791	0.772	0.732	0.780
	FCNN	0.760	0.732	0.731	0.732	0.750	0.678	0.760
	BERT	0.799	0.778	0.827	0.802	0.755	0.710	0.775
	Active LSTM	0.776	0.758	0.807	0.782	0.761	0.720	0.765
	Adaptive LSTM	**0.788**	**0.781**	**0.772**	**0.767**	**0.750**	**0.710**	**0.770**
	Adaptive BiLSTM	0.782	0.774	0.765	0.779	0.744	0.700	0.750
	Q-learning	0.783	0.774	0.784	0.782	0.750	0.710	0.760
Glove	LR	0.703	0.721	0.689	0.705	0.744	0.654	0.700
	XGB	0.691	0.709	0.677	0.693	0.732	0.620	0.685
	MLP	0.727	0.733	0.737	0.735	0.744	0.678	0.710
	FCNN	0.573	0.578	0.588	0.579	0.639	0.552	0.600
	BERT	0.622	0.652	0.568	0.607	0.635	0.520	0.590
	Active LSTM	0.608	0.734	0.330	0.455	0.755	0.420	0.550
	Adaptive LSTM	0.676	0.688	0.679	0.678	0.744	0.610	0.665
	Adaptive BiLSTM	**0.721**	**0.722**	**0.725**	**0.721**	**0.761**	**0.654**	**0.735**
	Q-learning	0.698	0.658	0.682	0.682	0.744	0.654	0.690

Table 3: Performance metrics for ablation study models.

Model	Acc.	Pre.	Rec.	F1
Base Model	0.818	0.818	0.818	0.817
No Attention	0.806	0.806	0.806	0.805
Reduced LSTM Units	0.799	0.799	0.799	0.798
No Residual Connections	0.799	0.799	0.799	0.798

4.2 Discussion

The NULLDect model uses LSTM, attention processes, and adaptive learning to successfully detect NPD vulnerabilities. LSTM detects long-term interdependence, attentiveness improves concentration, and residual connections boost stability. Adaptive learning responds to different coding patterns, ensuring generalization. Regularization, dropout, and binary classification improve efficiency while reducing overfitting. NULLDect adapts to changing software paradigms by using embeddings such as CodeT5, BERT, GloVe, and UnixCoder. This framework provides a robust and scalable method for NPD detection, as demonstrated in Table 4, whereas other research focus on various vulnerabilities. From the table, NULLDect performs best across major assessment metrics, making it the most effective model in this comparison. With an accuracy of 0.806, it outperforms models like Devign (0.790) and RealVul (0.798). Furthermore, NULLDect earns the greatest precision score of 0.814, much outperforming models such as RealVul (0.469). It also shares the highest F1-score with Devign, at 0.814, indicating a good combination of precision and recall. Thus, based on these findings, NULLDect is the best model for discovering vulnerabilities in the given dataset. It should be noted here that this innovation can have surprising applications, even in the emerging machine learning context (e.g., Lugosch (2018); Howlader et al. (2018); Camara et al. (2018); Leung et al. (2019); Li and Sun (2025)).

Table 4: Comparison NULLDect with outer studies.

Model	Acc.	Pre.	F1
Devign	0.790	-	0.841
AIBugHunter	0.741	-	-
VulDeePecker	-	-	0.808
RealVul	0.798	0.469	0.598
RefBilgin et al. (2020)	-	0.701	0.598
RefTanwar et al. (2020)	0.700	0.740	-
RefZagane et al. (2020)	-	0.734	0.729
NULLDect	0.806	0.814	0.814

5 CONCLUSIONS AND FUTURE WORK

This study aimed to develop another model for future vulnerability identification, focusing on overcoming difficulties identified in prior studies. The suggested model, NULLDect, employs a meta-classifier method, processing text and code input and the second handling categorization and adapting to new data for enhanced recognition. NULLDect obtained an accuracy of 80.06%, indicating its ability to identify CWE-476 vulnerabilities. Key benefits include its capacity to adapt to new datasets and improve detection accuracy over time, making it a viable tool for future vulnerability identification.

ACKNOWLEDGEMENTS

This research is supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU and by the National Aeronautics and Space Administration (NASA), under award number 80NSSC20M0124, Michigan Space Grant Consortium (MSGC).

REFERENCES

Abdalkareem, R., Shihab, E., and Rilling, J. (2017). On code reuse from stackoverflow: An exploratory study on android apps. *Information and Software Technology*, 88:148–158.

Akuthota, V., Kasula, R., Sumona, S., Mohiuddin, M., Reza, M., and Rahman, M. (2023). Vulnerability detection and monitoring using llm. In *Proceedings of WIE and WIECON-ECE*, pages 309–314. IEEE.

Alon, U., Brody, S., Levy, O., and Yahav, E. (2018). code2seq: Generating sequences from structured representations of code. *arXiv preprint arXiv:1808.01400*.

Alon, U., Zilberstein, M., Levy, O., and Yahav, E. (2019). code2vec: Learning distributed representations of code. *Proceedings of PACMPL*, 3(POPL):1–29.

Alqaradaghi, M. and Kozsik, T. (2022). Inferring the best static analysis tool for null pointer dereference in java source code.

Bilgin, Z., Ersoy, M., Soykan, E., Tomur, E., Çomak, P., and Karaçay, L. (2020). Vulnerability prediction from source code using machine learning. In *IEEE Access*, pages 150672–150684. IEEE.

Bojanova, I. and Galhardo, C. (2021). Classifying memory bugs using bugs framework approach. In *Proceedings of COMPSAC*, pages 1157–1164. IEEE.

Byun, M., Lee, Y., and Choi, J. (2020a). Analysis of software weakness detection of cbmc based on cwe. In *Proceedings of ICACT*, pages 171–175. IEEE.

Byun, M., Lee, Y., and Choi, J. (2020b). Analysis of software weakness detection of cbmc based on cwe. In *Proceedings of ICACT*, pages 171–175. IEEE.

Camara, R. C., Cuzzocrea, A., Grasso, G. M., Leung, C. K., Powell, S. B., Souza, J., and Tang, B. (2018). Fuzzy logic-based data analytics on predicting the effect of hurricanes on the stock market. In *Proceedings of FUZZ-IEEE*, pages 1–8. IEEE.

Chen, T. (2015). Xgboost: extreme gradient boosting. In *R package version 0.4-2*, pages 1–4. R Foundation for Statistical Computing.

Choi, Y. and Kwon, Y. (2022). An assessment of graph neural networks for detecting pointer and type errors. In *Proceedings of ICTC*, pages 1167–1171. IEEE.

CWE Ranking (2020). https://cwe.mitre.org/top25/archive/2020/2020_cwe_top25.html. Accessed 10 April 2024.

Gotovchits, I., Van Tonder, R., and Brumley, D. (2018). Saluki: finding taint-style vulnerabilities with static property checking. In *Proceedings of BAR*.

Greff, K., Srivastava, R., Koutník, J., Steunebrink, B., and Schmidhuber, J. (2016). Lstm: A search space odyssey. In *IEEE Transactions on Neural Networks and Learning Systems*, pages 2222–2232. IEEE.

Guo, D., Lu, S., Duan, N., Wang, Y., Zhou, M., and Yin, J. (2022). Unixcoder: Unified cross-modal pre-training for code representation. *arXiv preprint arXiv:2203.03850*.

Howlader, P., Pal, K. K., Cuzzocrea, A., and Kumar, S. D. M. (2018). Predicting facebook-users' personality based on status and linguistic features via flexible regression analysis techniques. In *Proceedings of SAC*, pages 339–345. ACM.

Jang, B., Kim, M., Harerimana, G., and Kim, J. (2019). Q-learning algorithms: A comprehensive classification and applications. In *IEEE Access*, pages 133653–133667. IEEE.

Jin, W., Ullah, S., Yoo, D., and Oh, H. (2021). Npdhunter: Efficient null pointer dereference vulnerability detection in binary. *IEEE Access*, 9:90153–90169.

Koroteev, M. (2021a). Bert: a review of applications in natural language processing and understanding. In *arXiv preprint arXiv:2103.11943*, pages 1–14. arXiv.

Koroteev, M. (2021b). Bert: a review of applications in natural language processing and understanding. In *arXiv preprint arXiv:2103.11943*, pages 1–14. arXiv.

LaValley, M. (2008). Logistic regression. In *Circulation*, pages 2395–2399. Lippincott Williams & Wilkins.

Leung, C. K., Braun, P., and Cuzzocrea, A. (2019). Ai-based sensor information fusion for supporting deep supervised learning. *Sensors*, 19(6):1345.

Li, K. and Sun, D. (2025). A global-features and local-features-jointly fused deep semantic learning framework for error detection of machine translation. *J. Circuits Syst. Comput.*, 34(1):2550025:1–2550025:22.

Lu, G., Ju, X., Chen, X., Pei, W., and Cai, Z. (2024). Grace: Empowering llm-based software vulnerability detection with graph structure and in-context learning. *Journal of Systems and Software*, 212:112031.

Lugosch, L. P. (2018). *Learning algorithms for error correction*. McGill University (Canada).

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543. Association for Computational Linguistics.

Pratt, H., Williams, B., Coenen, F., and Zheng, Y. (2017). Fcnn: Fourier convolutional neural networks. In *Proceedings of ECML PKDD*, pages 786–798. Springer International Publishing.

Sandoval, G., Pearce, H., Nys, T., Karri, R., Garg, S., and Dolan-Gavitt, B. (2023). Lost at c: A user study on the security implications of large language model code assistants. In *Proceedings of USENIX SS*, pages 2205–2222.

Security Importance (2024). https://moldstud.com. Accessed 5 April 2024.

Tanwar, A. et al. (2020). Predicting vulnerability in large codebases with deep code representation. In *arXiv preprint arXiv:2004.12783*, pages 1–20. arXiv.

VDISC Dataset (2024). https://osf.io/d45bw/. Accessed 10 April 2024.

Wang, J., Huang, Z., Liu, H., Yang, N., and Xiao, Y. (2023). Defecthunter: A novel llm-driven boosted-conformer-based code vulnerability detection mechanism. *arXiv preprint arXiv:2309.15324*.

Wang, Y., Wang, W., Joty, S., and Hoi, S. (2022). Codet5: Identifier-aware unified pre-trained model for code understanding and generation. *Proceedings of ACL*.

Zagane, M., Abdi, M., and Alenezi, M. (2020). Deep learning for software vulnerabilities detection using code metrics. In *IEEE Access*, pages 74562–74570. IEEE.