TPM Vol. 32, No. S8, 2025
ISSN: 1972-6325
https://www.tpmap.org/

CNN-BASED CODE READABILITY CLASSIFICATION AND BUG LOCALIZATION IN PROGRAMMING FRAMEWORK

ROLANDO B. BARRAMEDA[1], MELVIN A. BALLERA[2]
[1]TECHNOLOGICAL INSTITUTE OF THE PHILIPPINES
[2]TECHNOLOGICAL INSTITUTE OF THE PHILIPPINES
EMAIL: mrbarrameda@tip.edu.ph[1], melvin.ballera@tip.edu.ph[2]

Abstract— Learning to program poses several challenges for students, particularly when it comes to writing readable code and effectively localizing bugs. Readability is a critical software quality attribute, aiding students in understanding their programs, identifying mistakes, and collaborating with others. However, the large volume and complexity of programming tasks often make it difficult for instructors to thoroughly assess the readability of every student's code. Similarly, bug localization remains a time-consuming and complex task, especially for students in the early stages of their coding education. Traditional methods for assessing code readability and localizing bugs tend to rely on simplistic metrics, which fail to fully capture the complexities of these phenomena. This research explores the potential of deep learning, specifically Convolutional Neural Networks (CNNs), to address these challenges. CNNs, originally designed for image processing, have shown promise in recognizing complex patterns within structured data, making them suitable for analyzing source code. With this, the researcher aims to create a framework on how the integration of a hybrid activation function model that combines ReLU (Rectified Linear Unit) and Leaky ReLU to enhance the performance of CNNs in these tasks. This will to contribute valuable insights into how deep learning models, particularly CNNs, can complement the learning experience of students, making coding education more efficient and effective.

Keywords— Convolutional Neural Networks, code readability, bug localization, deep learning, hybrid activation functions, ReLU, Leaky ReLU

INTRODUCTION

Learning programming hurdles mostly — Writing Readable Code Readability helps students better understand how their programs work, where they went wrong, and how to work with others. The sheer amount of tasks and how complicated some of them are makes it impossible for the instructor to go through all of her students' code variants and make sure every single one of them is readable. Invoking a similar pattern, bug localization is still a time-consuming and challenging task for students, especially those who are still building their coding skills.
Code readability has been well-established as an important software quality attribute. A number of different methods are being pursued to quantify code readability (Fakhoury et al., 2018; Xu et al., 2019). However, these approaches are often based on simplified metrics that do not do justice with such a complex phenomenon like readable code. While writing any development code fast is one aspect, another very important thing is writing code that will be easily understandable, modifiable, and maintainable. Researchers needed more sophisticated and holistic models of code readability assessment (Jaffe et al., 2017).

To these challenges, promising deep learning models such as Convolutional Neural Networks (CNNs) in software maintenance tasks like code readability classification and bug localization have shown their promise using machine learning techniques. CNNs, which were initially designed for image processing, can recognize complex patterns in structured data and thus can be applied to the analysis of source code (Author, 2021). The image recognition tasks can be adapted to classify and analyze structured data, such as programming code. By converting the source code into a structured format (like matrices or vectors), CNNs can distinguish between readable and unreadable code and identify bugs that could be present in it.

This study will use open-source datasets to apply CNNs in the context of code readability and bug localization for first year and second-year students in Computer Science and Information Technology at De La Salle University - Dasmarinas. By using secondary data, this research aims to analyze the performance of CNN-based models in providing automated feedback on code readability and bug localization, which can complement the learning experience for students at DLSU.

A. Research Questions

The following are the research questions that the researcher aims to solve

1. How can the performance of Convolutional Neural Networks (CNNs) be improved in code readability classification and bug localization by integrating a hybrid activation function model that combines ReLU and Leaky ReLU?
2. What are the key features and techniques necessary to design an automated system that classifies source code as readable or unreadable and localizes bugs using deep learning models, particularly CNNs?
3. How does the effectiveness of CNN-based bug localization compare to traditional bug localization methods in terms of scalability, accuracy, and efficiency when handling large codebases?
4. What impact does the use of hybrid activation functions (ReLU and Leaky ReLU) have on the training process of CNNs, and how can these functions mitigate issues like dying ReLU and improve the flow in deep networks?

B. Signficance of the Study

This research will to be of great benefits to the following:

Students

This research offers a novel approach to enhancing the programming skills of first-year and second-year students at De La Salle University-Dasmarinas. By implementing automated systems that classify code readability and localize bugs, students can receive real-time feedback on their code. This early feedback enables them to identify and correct issues early in the learning process, improving both the clarity and quality of their code. Ultimately, this can significantly benefit their programming careers by developing essential coding skills from the outset.

Teachers

The application of Convolutional Neural Networks (CNNs) for automatic code analysis presents educators with an effective tool for assessing students' code. By utilizing CNNs for the classification of code readability and bug localization, teachers can more easily identify common mistakes among students and provide targeted interventions. Furthermore, incorporating CNNs into the automated grading process will allow educators to focus more on other critical aspects of instruction, enhancing overall teaching efficiency and effectiveness.

De La Salle University – Dasmarinas

The findings of this study could have a significant impact on curriculum development at De La Salle University-Dasmarinas by advocating for the integration of machine learning models into programming courses. An automated feedback system could help students improve their coding practices early on, ensuring that they acquire the necessary skills to excel in more advanced stages of their education. This approach could ultimately enhance the quality of the university's computer science and information technology programs, aligning with the evolving needs of the tech industry.

C. Conceptual Framework

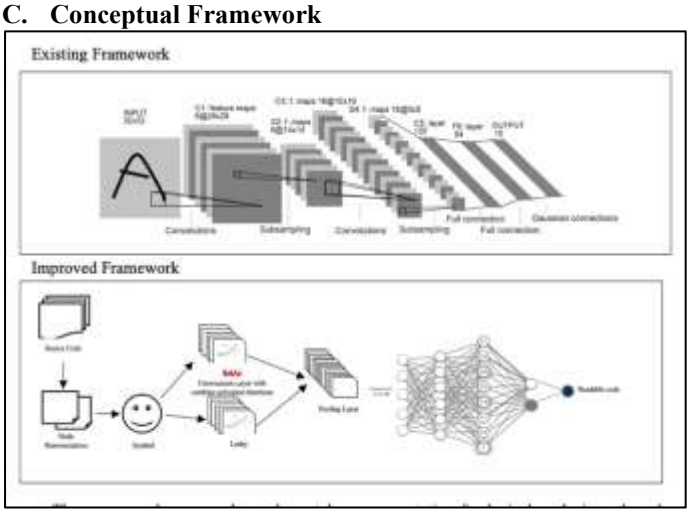

Figure 1.0 Conceptual Framework

The source code process by node or token representation for lexical analysis such as keywords, Operators, Sepator etc. and Node level representation for Abstract syntax such as declaration, definition, statement, condition etc. then mapping using vertex symbols where the stage trained by ConvNets and perform classification. The researcher aims to develop a framework that integrates a hybrid activation function model, combining ReLU (Rectified Linear Unit) and Leaky ReLU, to enhance the performance of Convolutional Neural Networks (CNNs) in specific tasks. This framework seeks to provide valuable insights into how deep learning models, particularly CNNs, can improve the learning experience for students, thereby making coding education more efficient and effective. By optimizing the performance of CNNs through this hybrid activation model, the research intends to contribute to the advancement of educational tools and techniques in coding, offering a more impactful and practical approach to teaching and learning.

LiTERATURE REVIEW

Deep learning techniques have been increasingly applied across various fields to address complex challenges, leading to the development of more sophisticated models. One such model, the Convolutional Neural Network

(CNN), has shown substantial success in tasks related to image recognition, natural language processing (NLP), and code analysis (LeCun, Bengio, & Hinton, 2015)

However, CNNs have their downside, especially when it comes to training and testing, which presents a form of time complexity problem.

The computational intensity is effected by the number of filters or kernels, the depth of the network, and the stride length (He et al., 2016).

The increased parameters lead to higher memory usage and longer training times, which is a major problem in large-scale applications (Chen et al., 2017).

Recent research has focused on optimizing CNN architectures to reduce computational overhead while maintaining performance (Szegedy et al., 2017).

He et al. (2016) discuss the time complexity of CNNs, focusing on the impact of the width (number of filters), depth (number of layers), and parameter counts on the computational cost. It discusses the trade-offs between filter size and network depth, replacing larger filters with smaller cascades, such as using two 2x2 filters instead of a 3x3 filter, which may reduce the computational burden without a loss in accuracy.

These techniques optimize the performance of CNN within constrained environments, especially when the memory and processing power are constrained.

Nwankpa et al. (2018) discuss the role of activation functions in deep learning models in great detail. ReLU has becomemost popular activationfunction since it is highly effective in solving the vanishing gradient problem that prevents the training of deeper networks.

ReLUworks as it sets every negative value at zero, resulting in a computationally faster setup and deeper network Although helpful,ReLU has the major drawback of allowing overfittingespecially within networks that include a large amount of parameters.

Dropout techniques as well as the use of several regularization methods to prevent overfitting and also improve generalization are common tools used (Srivastava et al., 2014).

Traditional application of CNNs is limited to image processing tasks, and very recently, this area of application has been expanded to other fields like software engineering tasks such as bug localization and code readability classification. Bug localization in the field of software testing is one of the primary activities that contribute towards efficient software maintenance and improvement. NLP was applied to improve the understanding of bug reports and source code comments. According to Badreesh (2018), NLP is the subfield of artificial intelligence that makes machines understand and generate human language. With NLP techniques used in the analysis of bug reports, there will be proper categorization that would allow the developers to first tackle the most important issues (Gonzalez et al., 2017).

METHODOLOGY

D. Reseearh Design

The research design for this study is primarily applied and quantitative in nature, as it aims to explore the effectiveness of Convolutional Neural Networks (CNNs) for automating code readability classification and bug localization in programming tasks for first-year and second-year students at De La Salle University-Dasmarinas (DLSU-D). The design focuses on using machine learning models to assess the quality of source code and identify bugs, offering an innovative approach to enhance the learning experience of students.

E. Methods and Techniques

The researcher will utilize Python 3.8+ because it provides a wide, comprehensive ecosystem with extensive support for libraries and is compatible with the deep learning frameworks like TensorFlow and Keras. Its expressive syntax makes it suitable for manipulating data, building machine learning models, and for doing computational work. Moreover, an active community with extensive documentation facilitates efficient debugging and optimization. Its flexibility allows for smooth integration with all types of data processing and machine learning libraries, making this code a candidate for implementing convolutional neural network (CNN) architectures that focus on readability and bug localization

To improve the development process, Jupyter Notebook was used as an interactive coding environment to test different configurations of models, preprocess data, and train visualization. This cell-based implementation supported the module development that, step by step, allows one to test and debug each part of a CNN. Coupling with libraries implemented in Python simplified model testing and parameters' calibration. Also, the capabilities for visual presentation implemented in Jupyter Notebook facilitated drawing feature maps, loss functions, and metrics in graphs to polish up the model's architecture as well as optimal parameters of the hyperplane.

TensorFlow is a completely free, open-source deep learning framework that has been optimized for implementing CNNs across the CPU and GPU to effect acceleration. The extended libraries and APIs of TensorFlow ensure that machine learning models can be built and deployed for scalability and performance. On the other hand, for better designing, training, and fine-tuning deep neural networks, Keras, a high-level API built on top of TensorFlow, was employed. With the user-friendly interface, it will enable rapid prototyping and simplified model implementation at the same time maintaining the freedom needed for intricate customizations. The combination of TensorFlow and Keras made development and deployment in an efficient model that enhanced performance overall in CNN-based approach.

Deep learning frameworks were not singular; rather a few Python libraries were used as support for the processing, visualizing, and feature extraction needed. Numerical computations and array manipulations were carried out with NumPy. Mathematical operations required to train the model were thus done efficiently. Data preprocessing, handling structured datasets, and simplifying data preparation for CNN input was facilitated with Pandas. Data visualization and analysis of the model's performance was done with Matplotlib and Seaborn to graphically depict training progress, loss functions, and accuracy metrics. Furthermore, OpenCV was used to pre-process source code representations to a format understandable by the CNN-based analysis. Together, all these libraries improved the management of data and preprocessing; this made the deep learning workflow more solid and efficient,

To support further in the development process, Jupyter Notebook served as the interactive coding environment for testing various configurations of models, preprocessing data, and visualizing training results. Its cell-based execution allowed for modular development, allowing step-by-step testing and debugging of CNN components. Integration with Python libraries helped in efficient model evaluation and parameter tuning. Moreover, Jupyter Notebook's visualization capabilities supported the graphical representation of feature maps, loss functions, and performance metrics, contributing to the refinement of the model's architecture and the optimization of hyperparameters.

In addition to that, Integrated Development Environments played an important role in code development and execution. PyCharm was used for writing and debugging the Python scripts, including intelligent code completion, error detection, and integrated version control. Google Colab was used for the training of models for taking advantage of cloud-based execution with GPU acceleration, allowing the efficient training of deep learning models with hardware freedom. These IDEs made the code development workflow to be better based on organization improvements, enhanced debug efficiency, and proper management of computational resources.

F. Model Architecutre

The proposed Convolutional Neural Network (CNN) model follows a structured pipeline designed specifically for code readability and bug localization analysis. To transform the input representations of source code, graph-based representations are used to convert the code into visual formats that are amenable to CNN processing. In addition, code symbolization is applied to convert hard functions into meaningful vector representations, ensuring that key patterns within the code are captured effectively.

The CNN model consists of multiple layers where hierarchical features are extracted from the source code representations. The convolutional layers capture spatial and structural dependencies within the code, while activation functions such as ReLU and Leaky ReLU ensure optimized gradient flow during the training process. Pooling layers are utilized to reduce dimensionality and computational overhead. Finally, fully connected layers are employed to transform the extracted features into meaningful predictions.

G. Model and Training

The model is trained using supervised learning on labeled datasets to achieve optimal performance with well-defined training strategies. **Categorical Cross-Entropy** is utilized as the loss function to facilitate multi-class classification, enabling the model to distinguish between various levels of code readability and different types of bugs. The **Adam optimizer**, with its adaptive learning rate properties, is chosen to ensure effective convergence throughout the training process. A batch size of 32 is selected to strike a balance between computational efficiency and model performance, facilitating effective learning while minimizing memory overhead.

To evaluate the model's effectiveness, several performance metrics are used. **Accuracy** is measured to assess the overall correctness of the model's predictions. **Precision** and **Recall** are calculated to evaluate the reliability of bug localization, ensuring that the model effectively identifies and categorizes coding errors. The **F1-score** is computed to provide a comprehensive assessment of the model’s classification performance, balancing both precision and recall.

H. Integrated Development Environments (IDEs)

Integrated Development Environments (IDEs) play a crucial role in promoting efficient code writing and execution. **PyCharm** was used for writing and debugging Python scripts, offering intelligent code completion, error detection, and integrated version control, which streamlined the debugging process and ensured well-organized code structure. Additionally, **Google Colab** was employed for model training, utilizing cloud-based execution with GPU acceleration to optimize computation. This setup mitigated hardware constraints, enabling large-scale training of deep learning models without requiring high-end local resources. The integration of these IDEs into the development workflow enhanced code management, debugging capabilities, and made efficient use of available computational resources

CONCLUSION AND FUTURE WORKS

The proposed model uses advanced machine learning techniques to automatically assess the readability of students' code and localize potential bugs, offering real-time feedback that can significantly improve learning outcomes. By integrating hybrid activation functions such as ReLU and Leaky ReLU, the model ensures more efficient training and better performance, mitigating issues like dying ReLU and enhancing the accuracy of predictions. Additionally, the research will evaluate the effectiveness of CNNs in comparison to traditional bug localization methods, offering insights into how deep learning models can address the limitations of conventional techniques.

In conclusion, this research offers a promising direction for improving programming education through the use of machine learning and deep learning technologies. The findings could lead to the development of automated systems that complement the traditional learning process, creating a more efficient, scalable, and impactful educational environment. This approach has the potential not only to enhance the learning experience at DLSU-D but also to provide a model for integrating advanced technologies into academic curricula worldwide

Future Works

Building on the findings of this study, future research could explore several directions to further enhance and refine the application of Convolutional Neural Networks (CNNs) for code readability and bug localization. Below are potential areas for future work, aligned with the research questions:

Improving CNN Performance with Advanced Hybrid Activation Functions

Future research could explore the integration of additional advanced activation functions, beyond the hybrid ReLU and Leaky ReLU used in this study, to further optimize the performance of CNNs in code readability classification and bug localization. For instance, newer activation functions like Swish or GELU could be tested to evaluate whether they provide better gradient flow and help in mitigating issues like dying ReLU. Additionally, comparing the performance of these new functions with existing models could provide deeper insights into the effectiveness of hybrid activation functions in deep learning models applied to software maintenance tasks.

Expanding the Automated System for Broader Code Analysis

Building on the design of an automated system for code readability and bug localization, future work could focus on expanding the system to handle a broader range of programming languages and frameworks. The current model is tailored to Python, but by extending it to other languages such as Java, C++, or JavaScript, the system could be made more universally applicable. This would involve adapting the model architecture to account for syntax and structure differences between languages and ensuring that the model can generalize across diverse programming environments.

Enhancing Comparison with Traditional Methods for Bug Localization

While this study compares CNN-based bug localization with traditional methods, future research could perform a more in-depth analysis of specific scenarios where CNNs outperform traditional methods, and where they might fall short. Investigating factors such as code complexity, size of the codebase, and the nature of bugs could provide insights into the scalability of CNN-based models. Furthermore, introducing hybrid models that combine CNNs with rule-based or heuristic-based methods could be explored to achieve higher accuracy and efficiency in bug localization tasks.

Exploring the Impact of Hybrid Activation Functions on Deep Network Training

Future studies could focus on understanding the nuances of how hybrid activation functions affect the training dynamics of deep CNN architectures, particularly in the context of large-scale software codebases. This includes investigating how such functions influence convergence rates, training stability, and generalization capabilities

REFERENCES

[1] Robosa, J., Paras, N., Perante, L., Alvez, T., & Tus, J. (2021). The experiences and challenges faced of the public school teachers amidst the COVID-19 pandemic: A phenomenological study in the Philippines. International Journal Of Advance Research And Innovative Ideas In Education, 7(1), 10-6084.

[2] McKinsey & Company. (2020, October 5). How COVID-19 has pushed companies over the technology tipping point-and transformed business forever. McKinsey & Company.

[3] https://www.mckinsey.com/capabilities/strategy-and-corporate-finance/our-insights/how-covid-19-has-pushed-companies-over-the-technology-tipping-point-and-transformed-business-forever

[4] Hara, M. (2023). Educational reform for middle-income trap under digitalization: Culprits, challenges, and strategies in the Philippines.

[5] Moreno, F., & Sulasula, J. (2023). The Governance of State Universities and Colleges. Available at SSRN 4513423.

[6] Hanisch, M., Goldsby, C. M., Fabian, N. E., & Oehmichen, J. (2023). Digital governance: A conceptual framework and research agenda. Journal of Business Research, 162, 113777.

[7] Manoharan, A. P., Melitski, J., & Holzer, M. (2023). Digital governance: An assessment of performance and best practices. Public Organization Review, 23(1), 265-283.

[8] Palma, J. P. B., Avila, L. S., Mag-iba, M. A. J., Buman-eg, L. D., Nacpil Jr, E. E., Dayrit, D. J. A., & Rodelas, N. C. (2023). E-governance: A critical review of e-government systems features and frameworks for success. International Journal of Computing Sciences Research, 7, 2004-2017.

[9] Chua, C. (2014). Digital governance implementation and institutional performance of state universities and colleges (SUCs) in the Philippines. Computer Engineering and Intelligent Systems, 5(2), 53-61.

[10] Esteban, A. P. (2023). Web engineering and e-commerce: Bridging technology and business in the Philippines. Nueva Ecija University of Science and Technology.

[11] Chao, R. Y. (2021). Higher education in the Philippines. International Handbook on Education in South East Asia, 1-28.

[12] Estdale, J., & Georgiadou, E. (2018). Applying the ISO/IEC 25010 quality models to software product. In Systems, Software and Services Process Improvement: 25th European Conference, EuroSPI 2018, Bilbao, Spain, September 5-7, 2018, Proceedings 25 (pp. 492-503). Springer International Publishing.

[13] Ivankova, N. V., Creswell, J. W., & Stick, S. L. (2006). Using mixed-methods sequential explanatory design: From theory to practice. Field methods, 18(1), 3-20.
[14] Ahmed, A., Ahmad, S., Ehsan, N., Mirza, E., & Sarwar, S. Z. (2010, June). Agile software development: Impact on productivity and quality. In 2010 IEEE International Conference on Management of Innovation & Technology (pp. 287-291). IEEE.
[15] Dotong, C. I., & Laguador, J. M. (2015). Philippine quality assurance mechanisms in higher education towards internationalization. Studies in Social sciences and Humanities, 3(3), 156-167.
[16] Berchin, I. I., de Aguiar Dutra, A. R., & Guerra, J. B. S. O. D. A. (2021). How do higher education institutions promote sustainable development? A literature review. Sustainable Development, 29(6), 1204-1222.
[17] Schöpfel, J., & Azeroual, O. (2021). Current research information systems and institutional repositories: From data ingestion to convergence and merger. In Future directions in digital information (pp. 19-37). Chandos Publishing.
[18] The rule-based matching process. The Rule-Based Matching Process. (2023, May 24). https://community.reltio.com/blogs/suchen-chodankar/2021/10/31/the-rule-based-matching-process
[19] Esteban, A. P. L. (2023). Anomaly Recognition in Wireless Ad-hoc Network by using Ant Colony Optimization and Deep Learning. International Journal on Recent and Innovation Trends in Computing and Communication, 11(5), 395-403.