ON THE EFFECT OF EXPLAINABLE AI ON PROGRAMMING LEARNING: A CASE STUDY OF USING GRADIENT INTEGRATION TECHNOLOGY

Feng Hsu Wang

Department of Computer Science and Information Engineering,
Ming Chuan University, Taoyuan, Taiwan

ABSTRACT

AI-based learning technologies, especially deep learning, hold significant promise for enhancing students' learning experiences in educational systems. However, providing accurate predictions or answers to students' learning problems through high-performance deep learning models is not sufficient for students to achieve effective learning. This study explores Explainable Artificial Intelligence (XAI) in reducing students' cognitive load and improving learning outcomes within the realm of object-oriented programming education. Specifically, this study examines the application of Gradient Integration to generate coloured code segments associated with code errors predicted by a Performer-based deep learning classification model for debugging tasks. Thirty-six participants took part in a controlled experiment assessing students' cognitive load and learning performance through the XAI system. They were randomly assigned to a control group (N=18) and an experiment group (N=18). The independent-samples Wilcoxon-Mann-Whitney test results revealed that the coloured code segments reduce students' cognitive load (p=0.006) and improve their exam scores (p=0.006) significantly. This study contributes to an appropriate application of the XAI technique that can reduce students' cognitive load and improve learning outcomes in educational settings.

KEYWORDS

Explainable Artificial Intelligence, Deep Learning Technology, Human-Computer Collaborative Learning, Programming Education

1. INTRODUCTION

Learning technologies, particularly those harnessing deep learning, hold great promise for enriching educational experiences [1]. Nevertheless, the integration of deep learning into education encounters challenges regarding its efficacy in supporting student learning. Simply furnishing precise predictions or solutions to students' learning queries through high-performing deep learning models falls short of ensuring effective learning outcomes. The emergence of Explainable Artificial Intelligence (XAI) addresses this issue by endowing intelligent systems with the capability to elucidate the reasoning behind their decisions. By employing transparent models, educators and students can gain deeper insights into the rationale guiding intelligent agents' choices, facilitating personalized recommendations. This fosters greater trust in the system and enables tailoring of learning experiences to individual needs. From an educational standpoint, the integration of XAI techniques holds substantial promise for enhancing students' learning journeys and mitigating various challenges within the educational landscape.

DOI: 10.5121/csit.2024.141202

One prominent XAI technique encompasses Gradient-based methods, including saliency maps and feature importance scores. These methodologies empower researchers and practitioners to delve into the intricate mechanisms of deep neural networks. Through scrutinizing the gradients of a deep-learning model concerning input features, analysts can discern the significance of each feature in shaping the model's output. This transparency proves pivotal as it not only enhances comprehension of the model's decision-making but also enables the detection of potential biases, thereby fostering a more transparent and equitable utilization of deep-learning models.

Among gradient-based methods, Gradient Integration [2] is distinguished as one of the most robust techniques for interpreting decisions made by complex machine learning models. Providing a harmonious blend of simplicity and effectiveness, Gradient Integration emerges as an attractive option for researchers aiming to enhance interpretability without compromising computational efficiency. Furthermore, its visually intuitive nature makes explanations accessible to a broader audience, including stakeholders with limited technical expertise, thus fostering heightened trust and adoption of AI systems across diverse applications. Consequently, this study adopts Gradient Integration as the selected XAI technology to generate color-coded segments of code. These segments serve as a scaffolding framework to assist learners in swiftly identifying code locations associated with the predictions of deep learning models, thus aiding in comprehension.

On the other hand, object-oriented programming (OOP) stands as a fundamental skill in programming, having evolved into a paradigm widely embraced by the software industry for crafting robust and adaptable software solutions. Among university information-engineering departments, introductory courses in OOP are often perceived as challenging and intricate by most students. Rectifying errors within OOP code demands a substantial cognitive load alongside proficient programming expertise. Hence, the educational focus of this study centres on evaluating how explanations provided through XAI techniques, such as Gradient Integration, may impact students' learning experiences in OOP, considering cognitive load and learning outcomes.
This study aims to explore the effects of employing the scaffolding framework facilitated by deep learning models using the Gradient Integration technique as an XAI tool to alleviate student cognitive load and enhance learning outcomes. Our methodology entails crafting a deep learning model based on the Performer architecture, augmented with an explanation module utilizing gradient integration to elucidate identified code errors. The incorporation of this XAI technology furnishes rationale for the model's predictions by visually pinpointing error locations within the code. Consequently, this study revolves around key research inquiries, as articulated below:

(RQ1) How does the transparency mechanism for code errors, employing Gradient Integration, impact students' cognitive load?

(RQ2) To what extent does the transparency mechanism for code errors, employing Gradient Integration, influence students' learning outcomes?

2. Literature Review

2.1. Deep Learning

Deep learning, a subset of artificial intelligence, harnesses multi-layer neural networks to tackle specific learning tasks through the analysis of vast datasets. It has demonstrated success across various domains including computer vision, speech recognition, and natural language processing. Notably, Transformer-based deep learning models have made remarkable advancements in tasks like image/speech recognition and machine translation [3]. However, the traditional Transformer architecture incurs a computational cost of O(L2) when performing attention operations on a sequence of length L. Amidst various improved iterations of Transformer, Performer stands out as one of the most efficient in terms of both time and space utilization [4]. Leveraging a novel attention mechanism known as Fast Attention Via Positive Orthogonal Random Features (FAVOR+), Performer provides scalable, low-variance, and unbiased estimation of attention values. This is achieved through the computation of kernels represented by sets of random feature map decompositions. Wang [5] introduced an efficient inference algorithm utilizing cached Performer, capable of generating sequences of length L in O(L) time complexity. This breakthrough makes it feasible to implement deep learning models even on resource-limited computing environments. Consequently, the Performer architecture has been selected for constructing the deep learning model in this study, considering the constraints posed by limited hardware resources.

2.2. Explainable AI

The transparency of decision-making in neural models significantly impacts their acceptability in educational applications [5]. Explainable AI (XAI) hence becomes crucial because it not only aids in understanding model judgments but also improves models by facilitating bias detection and providing new insights. Consequently, the transformation of black-box models into transparent and interpretable ones has become essential [6][7][8][9][10][11]. Samek et al. [8] categorized XAI methods into two types for explaining deep learning model predictions. The first type involves developing methods that reveal what the model has learned [10], identifying features learned by each neuron. The second type computes the sensitivity of the model's prediction to changes in input [11], understanding how input changes affect predictions. Montavon et al. [6] provided insights into both methods, emphasizing their contribution to a better understanding of deep neural networks. This study focuses on the second attribution method so that we can investigate the impact of the model's input on classification decisions.
Model attribution falls under the category of the second type of Explainable AI methods, specifically focusing on quantifying how sensitive the model's predictions are to changes in the input. It elucidates the model's behaviour by analysing how alterations in the input correspond to variations in the model's predictions. For example, the Integrated Gradients [2] is an attribution method used widely in deep learning networks. The gradient reflects the model's sensitivity to changes in the dimensions of the input, indicating the importance of each dimensional data. Integrated Gradients, aligning with the study's objective, will be used for analysis in this study.

2.3. AI-Based Approaches to Programming Education

AI-based approaches to programming education hold significant potential due to their ability to effectively handle code diversity. Several studies have utilized neural networks to provide tailored feedback on specific programming skills, such as recurrent programming [13], [14]. While the Automatic Program Repairs (APR) technique has found widespread application in the software engineering industry, its impact on programming education remains limited. This limitation arises from its original design for professional programmers' code, making it less suitable for novice students [15]. Embracing a constructivist and student-cantered learning framework, intelligent agents can perform data analysis tasks and present information in an "intelligent" manner, facilitating students' engagement in critical thinking and uncovering the underlying meaning and value within the data. This approach epitomizes human-machine collaborative learning.

With the advancements in deep learning technology, the capabilities of intelligent agents in terms of automation and autonomy are significantly enhanced [5]. As educational companions, intelligent agents endowed with improved automation and autonomy can effectively bolster human learning experiences [16], [17], [18]. However, there remains a notable gap in research concerning the effectiveness of integrating deep learning technology within educational settings [1]. Recently, considerable attention has been directed towards employing deep learning models, such as ChatGPT, for generating natural language feedback on open-ended coding problems. This novel approach has sparked considerable interest within the education community. Nevertheless, despite these models demonstrating high proficiency in predicting and generating feedback in response to student queries, there's a concern that students might not derive significant benefits from them, as they might miss out on the critical thinking skills necessary for learning through problem-solving. Consequently, these deep learning models should provide intelligent scaffolding that aids students' learning processes rather than merely furnishing them with answers.

3. METHOD

To address the research inquiries, we initially construct a deep learning model utilizing the cached Performer architecture integrated with the Integrated Gradient XAI mechanism. Subsequently, we delineate the Integrated Gradient mechanism and the progression of the deep learning model designed to assist learners in mastering Object-Oriented Programming (OOP) debugging tasks. Following that, we outline the experimental design and the tools employed for data analysis.

3.1. Integrated Gradients

Integrated Gradients [2] is an attribution method used widely in deep learning networks. An input baseline x' (usually a vector of zeros) is required so that the Integrated Gradients algorithm can compute gradients along a linear path from the baseline input to the original input x, according to (1),

$$\mathrm{IG}_i(x) ::= (x_i - x_i') \times \int_{\alpha=0}^{1} \frac{\partial F\left(x' + \alpha \times (x - x')\right)}{\partial_{x_i}} d\alpha \tag{1}$$

where x_i denotes the i^{th} dimensional data of the original input, and x_i' represents the i^{th} dimensional data of the baseline input, F represents a neural network model, and $\frac{\partial F(x)}{\partial_{x_i}}$ is the gradient of $F(x)$ on the i^{th} dimension of the original input *x*. This gradient reflects the model's sensitivity to changes in the i^{th} dimension of the input, indicating the importance of each dimensional data. Integrated Gradients, aligning with the study's objective, is used as the case study for its effectiveness in supporting learning programming in this study.

3.2. The Performer-Based Encoder-Classifier Neural Network

A performer-based deep-learning model has been developed to analyse Java code, aiming to predict error types and support students in learning Object-Oriented Programming (OOP) design principles (see Figure 1). The Encoder is given a Java code and encodes it into a sequence of context features. In this study, we adopted a Performer-based encoder with five attention blocks and four parallel attention layers or heads. The Classifier is a multiple layer perceptron network that predicts multiple error type labels, including 35 error types and the correct one.

The code attributions are generated by the Integrated Gradient method as follows. Since the embedding layer lacks the capability to compute gradients for raw source code, the attribution process is exclusively conducted on the embedded output derived from the embedding layer.

Consequently, attributions of the embedded output are averaged across its embedding dimensions for each input token, providing insight into the relevance of the code to the error predictions.

A website deployed the deep learning model was developed using the Flask package, as shown in Figure 2. In this website, descriptions of debugging tasks are displayed in the left window, and the sample code is displayed in the middle window for editing. Clicking the green Diagnose button will submit the code to the server for analysis, and the predicted error types will be displayed in the right window. Students can accept or refuse the feedback and submit his/her decisions.

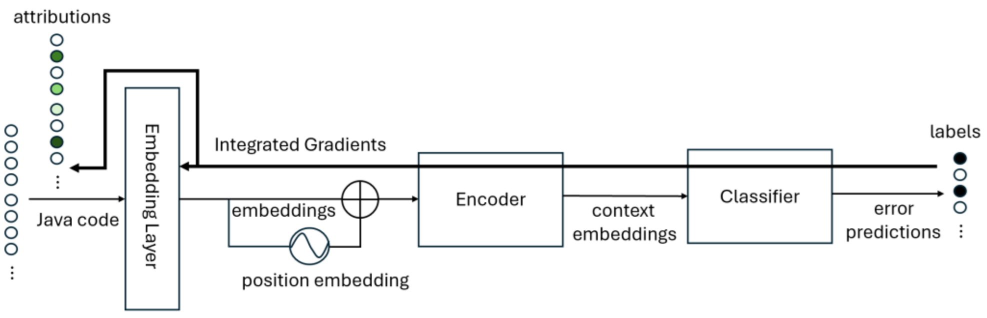
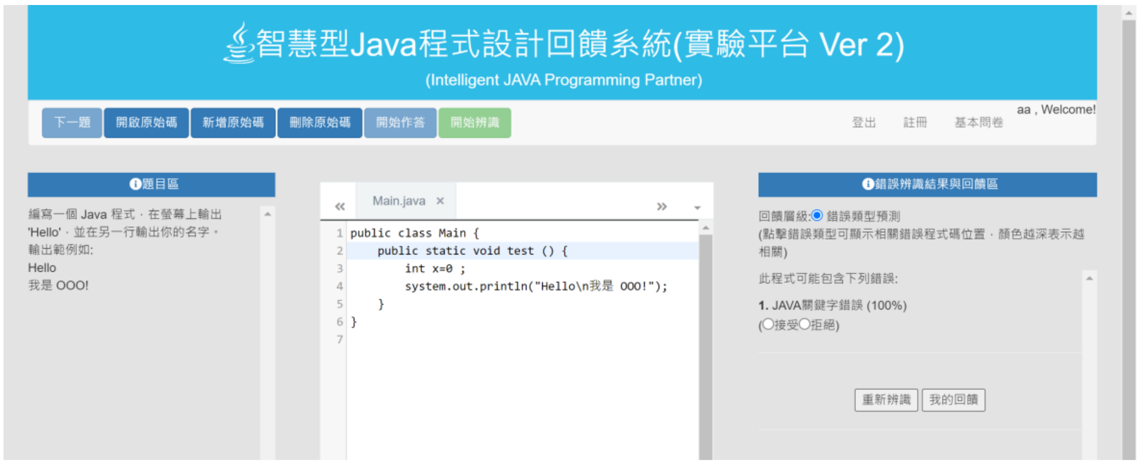


Figure 1. The encoder-classifier network utilizes Integrated Gradients to determine input attributions.



編寫一個 Java 程式，在螢幕上輸出 'Hello'，並在另一行輸出你的名字。
輸出範例如:
Hello
我是 OOO!

```
public class Main {
    public static void test () {
        int x=0 ;
        system.out.println("Hello\n我是 OOO!");
    }
}
```

回饋層級:[✓ 錯誤類型預測]
(點擊錯誤類型可顯示相關錯誤程式碼位置，顏色越深表示越相關)

此程式可能包含下列錯誤:

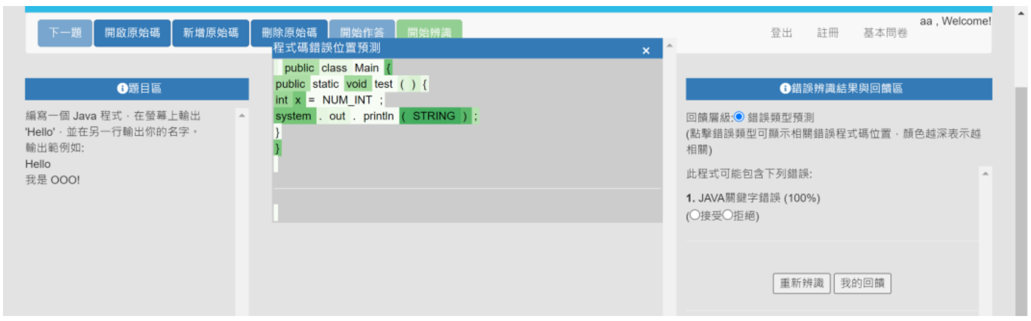
1. JAVA關鍵字錯誤 (100%)
(○接受○拒絕)

Figure 2. A screenshot showing task descriptions, student codes and system feedback.

The AI system facilitates student learning in programming by providing concise error predictions based on the code modifications submitted by students. It empowers students to take charge of correcting their code based on their own insights and assessments of the system's predictions. By clicking on the error labels presented in the right-hand side window, students can investigate the color-coded segments of code associated with each error, with deeper shades of green indicate higher relevance (refer to Figure 3). Utilizing these color-coded segments, students analyse pertinent sections of code and assess the predicted error descriptions to uncover underlying bugs. This process encourages students to autonomously integrate the feedback provided by the system into their learning journey without becoming overwhelmed by it.

3.3. Model Development

A deep-learning model was first developed based on data collected from Java code submitted by students for exercises, quizzes, and exams in an introductory OOP course at a university in Taiwan. A total of 330 codes were collected and manually annotated. The codes were randomly assigned to humans, with one code annotated by two annotators and proofread against each other. Since the GPU hardware device used in this study has limited storage, the code length was restricted to 1000 tokens. As a result, we sorted out a total of 35 error types. However, the small amount of code is far from the sample size required to train a neural model. Therefore, this study adopts the code augmentation method proposed by Wang [[19]] to increase the sample size by changing local variable names and randomly reordering statements without affecting the original code semantics. In the end, the sample size was increased from 330 to 11200.



編寫一個 Java 程式，在螢幕上輸出 'Hello'，並在另一行輸出你的名字。
輸出範例如:
Hello
我是 OOO!

```
public class Main {
public static void test ( ) {
int x = NUM_INT ;
system . out . println ( STRING ) ;
}
}
```

回饋層級: [✓ 錯誤類型預測]
(點擊錯誤類型可顯示相關錯誤程式碼位置，顏色越深表示越相關)

此程式可能包含下列錯誤:

1. JAVA關鍵字錯誤 (100%)
(○接受○拒絕)

Figure 3. A screenshot displays an interactive session featuring error predictions and pinpointed error locations highlighted by color-coded segments of code.

The data is divided into 80% training data and 20% testing data. The model was trained for 1,000 epochs, using 20% of the training data as validation. The training process is conducted on the Window 10 operating system with an Intel(R) Core (TM) i9-11900K 3.50GHz processor, 128GB RAM, and an NVIDIA RTX 3090 with 24GB RAM. We investigate the prediction performance of the Performer-based model. The error type prediction is evaluated by binary accuracy. In addition, feedback is collected from users to assess whether they find the code attributions helpful, accurate, and aligned with their mental models.

3.4. Experiment Design

This study conducts a human evaluation of the system's predictions and explanations represented as coloured code segments. Thirty-six students enrolled in an advanced Java programming course were invited to participate in the evaluation. Students are randomly assigned to the control group and experiment group. Totally, 18 students (5 females and 13 males) are assigned to the control group, while 18 students (7 females and 11 males) are assigned to the experiment group.

Both groups of students underwent a pre-test initially, with scores ranging between 0 and 5 points. Following this, they were assigned three debugging tasks of varying difficulty levels in a randomized order, each with different levels of assistance from the system. Students in the control group were provided solely with error type predictions from the system. Conversely, students in the experimental group received additional assistance in the form of code error explanations represented as coloured code segments. Subsequently, after completing each assigned task, all students took the same post-test, with scores normalized within the range of 0 to 1.

After completing the experiment, participants were asked to complete an eleven-item survey using a 5-point Likert scale, as detailed in Table 1. This survey comprised two sections: one focused on cognitive load, measuring participants perceived cognitive load arising from the system's error predictions and explanations, and the other aimed at gathering their evaluations of the system.

Table 1. Cognitive and system evaluation survey.

Item	Description
IL1	I find the thematic content covered in this activity to be complex.
IL2	I find the code diagnostic content (error descriptions/coloured code segments) generated by this system to be complex.
IL3	I find the object-oriented concepts and definitions covered in this activity to be complex
EL1	The feedback display and arrangement used in this system make the content appear unclear.
EL2	The feedback display and arrangement used in this system are helpful for my learning
EL3	The feedback display and arrangement used in this system make the content difficult to understand
GL1	I feel that the [Error Type Prediction] feature of this system can enhance my knowledge and understanding of object-oriented programming
GL2	In my opinion, the [Error Location Indication] feature of this system can improve my knowledge and understanding of object-oriented programming.
GL3	Overall, this system has enhanced my knowledge and understanding of object-oriented programming."
SO1	How does the [Error Type Prediction] feature provided by this system assist me in learning during debugging tasks? What are its drawbacks? (open question)
SO2	How does the code error location highlighting feature provided by this system assist my learning? What are its drawbacks? (open question)

An analysis was then conducted to gauge the internal consistency and reliability of the cognitive load survey. The findings, presented in Table 2, revealed that the Cronbach's Alpha for Intrinsic Cognitive Load (IL) was 0.79, for Extrinsic Cognitive Load (EL) was 0.56, and for Germane Cognitive Load (GL) was 0.83. These values indicate a moderate level of internal consistency across the dimensions of cognitive load.

Table 2. Reliability analysis results of the survey.

Construct	Items	Cronbach's Alpha
IL	IL1, IL2, IL3	0.790
EL	El1, EL2, El3	0.559
GL	GL1, GL2, GL3	0.826

Due to the limited participant size and the observable departure from a normal distribution, the cognitive load and learning performance of participants underwent analysis using the independent-samples Wilcoxon-Mann-Whitney test to determine whether there is a significant difference between the distributions of assessed scores (cognitive load and exam scores) between the control and experiment groups. The purpose of this analysis was to investigate whether significant differences in medians existed between the two groups of participants.

4. RESULTS

4.1. Prediction Accuracy

The loss and accuracy of the training process for the error classifier is shown in Figure 4. As seen in the figures, the model is well-trained for training and validation data. For error prediction, the error classifier achieves the average binary accuracy of 1.000, 0.999, and 0.999 for the training, validation, and testing data sets, respectively. The high accuracy of error predictions shows that it performs very well in the code analysis tasks, at least for the collected data.

4.2. Data Analysis

Table 3 and Table 4 show the descriptive statistics of the control and experiment groups, respectively. The experiment group exhibited significantly lower first intrinsic cognitive load (IL1) compared to the control group (U = 93.500, z = -2.299, p = 0.021, exact sig. p = 0.029; mean rank control = 22.31, mean rank experiment = 14.69). However, there was no significant difference between the second intrinsic cognitive load (IL2) of the control and experiment groups (U = 135.000, z = -0.901, p = 0.368, exact sig. p = 0.406). Notably, the experiment group showed significantly lower third intrinsic cognitive load (IL3) than the control group (U = 76.500, z = -2.889, p = 0.004, exact sig. p = 0.006; mean rank control = 23.25, mean rank experiment = 13.75).

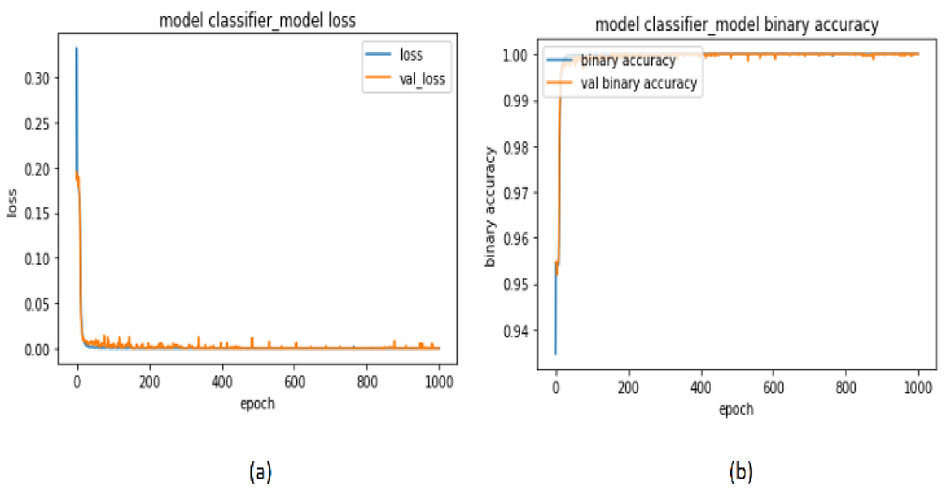


Figure 4. (a) Training loss and (b) training accuracy of the classifier model.

No significant differences in extrinsic cognitive load (EL1, EL2, EL3) were observed between the control and experiment groups. Specifically, for EL1, U = 147.500, z = -0.485, p = .628 (exact sig. p = 0.650); for EL2, U = 136.500, z = -0.856, p = .392 (exact sig. p = 0.424); and for EL3, U = 100.000, z = -2.112, p = .035 (exact sig. p = 0.051).

The study found no significant difference in the first germane cognitive load (GL1) between the control and experimental groups (U = 155.500, z = -0.218, p = .827; exact sig. p = 0.839). Similarly, there was no significant distinction in the germane cognitive load (GL3) between the control and experimental groups (U = 173.000, z = 0.398, p = .691; exact sig. p = 0.743).

The pre-test scores showed no significant difference between the control group and experiment group (U = 185.500, z = 0.768, p = .442, exact sig. p = 0.462). However, the experiment group demonstrated significantly higher post-test scores compared to the control group (U = 247.500, z

= 2.794, p = 0.005, exact sig. p = 0.006). The control group had a mean rank of 13.75, whereas the experiment group had a mean rank of 23.25.

Table 3. Descriptive statistics of the control group (N=18).

Factor	**Minimum**	**Maximum**	**Mean**	**Std. Dev.**
IL1	2	5	3.56	0.784
IL2	2	4	3.17	0.857
IL3	2	5	3.67	0.686
GL1	1	4	3.22	0.878
GL2	-	-	-	-
GL3	1	4	3.17	0.707
EL1	2	5	3.56	0.705
EL2	1	5	3.33	0.840
EL3	3	4	3.44	0.511
Pre-Test	2.00	5.00	3.28	1.074
PostT-est	0.00	0.50	0.35	0.110

Table 4. Descriptive statistics of the experiment group (N=18).

Factor	**Minimum**	**Maximum**	**Mean**	**Std. Dev.**
IL1	1	4	2.72	1.127
IL2	1	4	2.83	1.043
IL3	1	4	2.83	0.857
GL1	1	5	3.11	1.132
GL2	1	4	3.11	0.900
GL3	1	4	3.22	0.808
EL1	1	5	3.33	1.188
EL2	1	5	3.00	1.138
EL3	1	4	2.78	1.003
Pre-Test	2.00	5.00	3.50	0.9075
Post-Test	0.00	1.00	0.49	0.244

Table 5. Independent-samples Wilcoxon-Mann-Whitney test results between control group (N=18) and experiment group (N=18).

Factor	***U***	***z***	**Asymptotic *sig. p***	**Exact sig. p**
IL1	93.500	-2.299	0.021	0.029*
IL2	135.000	-0.901	0.368	0.406
IL3	76.500	-2.889	0.004**	0.006**
EL1	147.500	-0.485	0.628	0.650
EL2	136.500	-0.856	0.392	0.424
EL3	100.000	-2.112	0.035*	0.051
GL1	155.500	-0.218	0.827	0.839
GL2	-	-	-	-
GL3	173.000	0.398	0.691	0.743
Pre-Test	185.500	0.768	0.442	0.462
Post-Test	247.500	2.794	0.005**	0.006**

*: <0.05, ** < 0.01, ***<0.001

5. DISCUSSIONS

The cognitive load survey showed a notable decrease in the first and third intrinsic cognitive loads, indicating effective reduction with coloured code segments. However, there was no significant change in the second intrinsic cognitive load, suggesting minimal impact of coloured code segments on students' intrinsic cognitive load. Likewise, no significant difference was found in the extrinsic cognitive load, suggesting that students could manage the cognitive load of error descriptions and coloured code segments effectively. Additionally, there was no significant distinction in the germane cognitive load, indicating that processing error descriptions and coloured code segments did not distinctly enhance student learning and understanding.

This study discovered that error predictions generated by the deep learning model offer students general guidance for code correction. However, relying solely on these predictions may present challenges for students in understanding vague error descriptions and pinpointing bugs within the code. This can impede debugging efficiency by causing cognitive overload, as noted by survey participants,

> "When I'm unsure of a mistake, error type predictions offer hints, though textual descriptions can be challenging to grasp."
> "Identifying potential errors is faster with error type predictions, but the brief descriptions may only assist with obvious errors."

AI-generated coloured code segments have proven to reduce cognitive load during debugging tasks by directing students' focus to relevant code areas, thus speeding up error comprehension. However, less knowledgeable students may still encounter difficulties with coloured code. Additionally, there is room for improvement in the perceived reliability of error predictions and associated coloured code segments, as noted by participants in the open survey,

> "It's feasible to identify errors more precisely, but occasionally it seems somewhat inaccurate."
> "I can locate the lines where the error occurred, but I'm uncertain about the nature of the mistake."

Furthermore, the AI system provides a robust scaffolding platform for human-machine collaborative learning. Within this framework, students utilize error predictions and color-coded segments to quickly identify and deal with bugs in the code. Nevertheless, meaningful learning occurs when students autonomously evaluate predictions and code segments using their own knowledge and skills, as indicated by survey participants.

> "Predicting errors can expedite root cause identification yet should not be solely relied upon."
> "Error predictions aid bug resolution, but excessive reliance on debugging systems may backfire."
> "When unsure of the error, predictions can offer helpful clues."
> "I can verify if the system's suggestions match my ideas."

In brief, the AI-generated coloured code segments reduce student cognitive load and enhance learning outcomes, as shown by experiments. This demonstrates effective integration of AI technology in education, providing scaffolding while promoting independent learning.

6. CONCLUSION

6.1. Implications for Education, Research and Practice

This study shows the potential of XAI techniques like Integrated Gradient [2] in reducing student's cognitive load and improving learning performance in educational settings. Therefore, integration of Explainable Artificial Intelligence (XAI) in educational systems can enhance students' learning experiences. Educators can consider incorporating XAI techniques, such as Integrated Gradient for generating coloured code segments, to aid in reducing cognitive load and improving learning outcomes, particularly in technical subjects like object-oriented programming.

In addition, further exploration of XAI's role in educational settings, particularly in different subject areas and with diverse learner populations, is warranted. Future research should investigate additional XAI techniques and their effectiveness in reducing cognitive load and enhancing learning outcomes across various educational contexts.

Finally, educators should receive training on how to effectively utilize XAI tools to support students' learning processes. Implementation of XAI systems should be accompanied by thorough evaluation and feedback mechanisms to continuously improve their efficacy in educational practice.

6.2. Limitations and Suggestions for Future Research

The study's sample size is relatively small (N=36), which may limit the generalizability of the findings. Discretion must be exerted to extend the results to other courses, populations, and disciplines. The research focuses specifically on object-oriented programming education, potentially limiting its applicability to other subjects. The duration of the experiment and its long-term effects on learning outcomes are not addressed.

Future research should focus on conducting larger-scale studies with diverse participant groups to validate the effectiveness of XAI techniques in various educational contexts. Additionally, exploring the long-term effects of XAI integration on students' learning trajectories and knowledge retention is essential. Investigating potential interactions between XAI tools and individual differences in learning styles and preferences is also crucial. Moreover, examining educators' and students' perceptions and experiences regarding the usability and effectiveness of XAI systems in real-world educational settings is warranted.

ACKNOWLEDGEMENTS

This work was partly supported by the Ministry of Science and Technology, Taiwan, R.O.C., under Grant NSTC 112-2410-H-130-010 -.

REFERENCES

[1] X. Chen, H. Xie, D. Zou, and G.-J. Hwang, "Application and theory gaps during the rise of Artificial Intelligence in Education," Computers and Education: Artificial Intelligence, vol. 1, 100002.

[2] M., Sundararajan, A. Taly, and Q Yan, "Axiomatic attribution for deep networks," arXiv:1703.01365, 2017.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, "Attention is all you need," presented at the 31st Conference on Neural Information Processing Systems, Long Beach, CA, USA, 2017.

[4] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis, A. Mohiuddin, L. Kaiser, D. Belanger, L. Colwell, and A. Weller, "Rethinking attention with performers," presented at the Ninth International Conference on Learning Representations, 2021.

[5] F. H. Wang, "A feasible study of a deep learning model supporting human-machine collaborative learning of object-oriented programming," IEEE Transactions on Learning Technologies, 1-15, 10.1109/TLT.2022.3226345, 2022.

[6] G. Montavon, W. Samek, and K.-R. Muller, "Methods for interpreting and understanding deep neural networks," arXiv preprint arXiv:1706.07979, 2017.

[7] W. Samek, T. Wieg and, K.-R. Müller, "Explainable artificial intelligence: understanding, visualizing and interpreting deep learning models," arXiv preprint arXiv:1708.08296, 2017.

[8] D.-V. Finale, and K. Been, "Towards a rigorous science of interpretable machine learning," arXiv preprint arXiv:1702.08608, 2017.

[9] L. M. Zintgraf, T. S. Cohen, T. Adel, and M. Welling, "Visualizing deep neural network decisions: prediction difference analysis," arXiv preprint arXiv:1702.04595, 2017.

[10] A., Nguyen, J. Yosinski, J. Clune, "Multifaceted feature visualization: uncovering the different types of features learned by each neuron in deep neural networks," arXiv preprint arXiv:1602.03616, 2016.

[11] G. Montavon, S. Bach, A. Binder, W. Samek, and K.-R. Muller, "Explaining nonlinear classification decisions with deep Taylor decomposition," Pattern Recognition, vol. 65, pp. 211–222, 2017.

[12] V. J. Shute, "Focus on formative feedback," Review of Educational Research, vol.78, no. 1, pp. 153–189, 2008.

[13] W. L. Johnson, and E. Soloway, "Intention-based diagnosis of novice programming errors," presented at AAAI Conference, pp. 162–168, 1984.

[14] F. H. Wang, "An efficient performer-based model for automatic feedback generation for teaching object-oriented programming," presented at the 13th annual International Conference on Education and New Learning Technologies, 2022.

[15] A.Vizcaíno, "A simulated student can improve collaborative learning," Int. J. of Artificial Intelligence in Education, vol. 15, pp. 3–40, 2005.

[16] G. J. Hwang, H. Xie, B. W. Wah, and D. Gašević, "Vision, challenges, roles and research issues of artificial intelligence in education," Computers & Education: Artificial Intelligence, vol. 1, pp. 1–5, 2020.

[17] G. J. Hwang and C.-Y. Chang, "A review of opportunities and challenges of chatbots in education," Interactive Learning Environments, pp. 1-14, 2021.

[18] M. B. Magolda, "Learning partnerships model: A framework for promoting self-authorship," In M. B. Magolda & P. M. King (Eds.), Learning partnerships: Theory and models of practice to educate for self-authorship (pp. 37–62). Sterling, VA: Stylus, 2004.

[19] F. H. Wang, "A semantics-preserving approach to augmentation of Java codes for training deep learning classification models," presented at the 25th TANET, Taiwan, 2019.

AUTHOR

F. H. Wang received MS and Ph.D. in computer science and information engineering from National Taiwan University, Taiwan, in 1988 and 1993, respectively. He joined the Graduate School of Information Science and Engineering, Ming Chuan University, Taiwan, in 1995, where he is currently a professor. His research interests include deep learning technology, educational technology, and pedagogy research.