AIBEE: AI-Driven Block-coding Educational Environment for Teaching, Learning, and Prototyping AI

Calvin Zhou
W3B Education
San Jose, CA
calvinzhou@w3beducation.org

Abstract— **Block-coding environments such as Scratch and App Inventor have long been used as visual and interactive tools for STEM education and, in recent years, for teaching AI. However, studies show that the existing block-based applications are less effective in AI education due to drawbacks such as their black box-style presentation, difficulties in creating teaching content, and challenges in applying the knowledge to real-world tasks. This paper proposes AIBEE, an AI-driven Block-coding Educational Environment, providing functional AI blocks facilitating the full AI modeling process from data importing and processing to model training and testing. The AIBEE system aims to make AI education simpler and accessible for all by allowing users to interactively create an AI model using drag-and-drop blocks and instantly observing the running results without writing any code. This paper describes the design and implementation of the AIBEE system, as well as the planned experimental tests to comparatively evaluate its performance against other block-based educational environments.**

Keywords— **Block-based Coding, Educational Technology, AI Learning Platform**

I. INTRODUCTION

Among various AI educational tools or systems currently in active use, block-based programming or block-coding environments have been widely leveraged in computer science and recently in AI education, demonstrating decent performance and effectiveness for both teaching and learning [1]. Distinct from conventional text-based or other visual programming approaches, block coding uses connected puzzle pieces or boxes as visual cues to help the user establish the logic and processes of computational entities and data flows [2]. The most popular block-coding environments are MIT Scratch [3] [4] and MIT App Inventor [5], both having tens of millions of users worldwide.

Despite the ease of use and extensive acceptance of block-coding tools like Scratch and App Inventor, studies have shown their drawbacks, including:

- Teaching AI in black box-style blocks only demonstrates results without revealing the components inside to help the learners understand the conceptual foundations [6]
- Lack of comprehensive curriculum and interactive instructional support due to difficulties with creating teaching content and the lack of assisted teaching solutions [7]

Furthermore, most of these environments do not allow the learners to export the created models for other applications or deploy them for the learners' real-life use cases, causing a gap between learning and actual usage.

To address these challenges, this paper proposes AIBEE, an AI-driven Block-coding Educational Environment, aiming to solve the above problems with the following functionalities:

- Different levels and types of AI blocks that support high-level conceptual education as well as low-level model fine-tuning
- AI-driven content generation that assists curriculum creation and development
- An interactive AI teaching assistant that provides in-context instructional support
- Importing/exporting functionality for AI models to and from standard formats to facilitate an open system
- Integrated AI application hosting that allows the users to run the created models after learning/prototyping

II. METHODOLOGY

The AIBEE project consists of the following approaches: UI design, system architecture, and data flow.

A. UI Design

The AIBEE system is implemented as a cloud-hosted web application, with the main UI shown in Fig. 1.

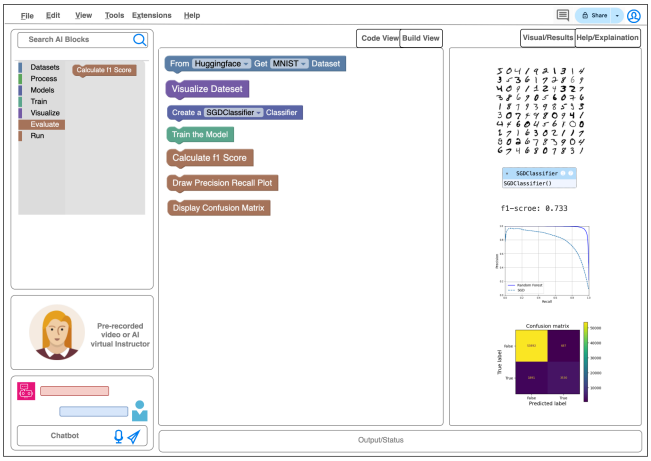

Fig. 1 Web UI Design

The web app UI adapts a 3-column layout with a left panel, central build space, and right display space. The upper part of the left panel is the block space including a search box, where the user can explore the AI blocks available to build an AIBEE Blockwork. The blocks are categorized into functional groups such as datasets, processing, models, training, visualization, evaluation, and running. Users can drag a block from the block space and drop it into the central build space.

The lower section of the left panel serves as the AI teaching zone. This area includes a chatbot for text-based conversational assistance and an AI instructor. Initially, the AI instructor will be a pre-recorded video player, but a virtual AI instructor is planned for future implementation. The central space is the work area where the user can build the AIBEE Blockwork to learn and test AI models. The default block view can be switched to the code view which displays the Python code generated from the current Blockwork. The right-side space displays the output of the Python code corresponding to the blocks, which is similar to the work style of a Jupyter Notebook. The visual output view can be switched to a text view which shows the related tutorials or the help documents of this system.

B. *System Architecture*

As shown in Figure 2, the AIBEE system is developed as a web application hosted on cloud infrastructure such as AWS. The web front end (WFE) is built using the Node.js framework and Google Blockly library [8] [9]. The service layer uses the microservice design, including web service APIs exposed as a user, dataset, model, code conversion, and block management. The services are routed using an AWS API gateway connecting the WFE. The backend storage uses AWS S3 file storage for saving blockwork files and AWS RDS for saving system data. The AI model training and running are driven by a Jupyter [10] Server hosted on AWS SageMaker service.

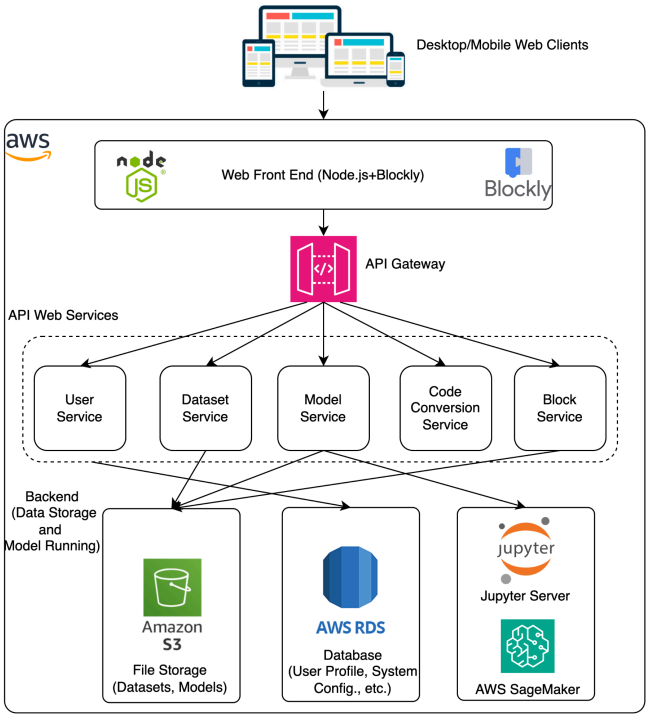

Fig. 2 System Architecture

C. *Data Flow*

The main data flow of this system uses a client-server style as shown in Figure 3. The core service handling the data flow is a custom node.js (AIBEE) module which validates the blockwork design and converts it to Jupyter notebook (ipynb) code. All the blocks in the AIBEE system are created as custom Google Blockly blocks, together with predefined variables, functions, and corresponding Python code.

III. EXPERIMENTAL TESTS

To evaluate the performance of AIBEE in actual learning and teaching, experimental AI education sessions have been scheduled in June and July as free, 5-day summer camps. We will collect approximately 60 registrations of high school students and randomly assign them to 3 groups: AIBEE, Scratch, and App Inventor. Each session will introduce students to one machine-learning module and one deep-learning module. At the end of the 5 days, students will be required to complete a survey for qualitative analysis and a quiz for quantitative evaluation of AIBEE's effectiveness.

IV. CONCLUSION

This paper describes the background, idea, and approach of the AIBEE system, an AI-driven Block-coding Educational Environment. This project has great potential as an effective and innovative solution to assist K-12 schools and higher education institutions in making AI education easier and accessible for all.

REFERENCES

[1] D. Weintrop, "Block-based Programming in Computer Science Education – Communications of the ACM," *Commun. ACM*, vol. 62, no. 8, pp. 22–25, Aug. 2019, doi: 10.1145/3341221.

[2] D. Weintrop and U. Wilensky, "Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms," *ACM Trans. Comput. Educ.*, vol. 18, no. 1, pp. 1–25, Mar. 2018, doi: 10.1145/3089799.

[3] J. Maloney, L. Burd, Y. Kafai, N. Rusk, B. Silverman, and M. Resnick, "Scratch: a Sneak Preview," in *Proceedings. Second International Conference on Creating, Connecting and Collaborating through Computing, 2004.*, Jan. 2004, pp. 104–109. doi: 10.1109/C5.2004.1314376.

[4] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond, "The Scratch Programming Language and Environment," *ACM Trans. Comput. Educ.*, vol. 10, no. 4, p. 16:1-16:15, Nov. 2010, doi: 10.1145/1868358.1868363.

[5] B. Magnuson, "Building Blocks for Mobile Games : A multiplayer framework for App Inventor for Android," Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2010. Accessed: Apr. 24, 2024. [Online]. Available: https://dspace.mit.edu/handle/1721.1/61253

[6] C.-B. Fleger, Y. Amanuel, and J. Krugel, "Learning Tools Using Block-based Programming for AI Education," in *2023 IEEE Global Engineering Education Conference (EDUCON)*, May 2023, pp. 1–5. doi: 10.1109/EDUCON54358.2023.10125154.

[7] C. Gresse Von Wangenheim, J. C. R. Hauck, F. S. Pacheco, and M. F. Bertonceli Bueno, "Visual tools for teaching machine learning in K-12: A ten-year systematic mapping," *Educ. Inf. Technol.*, vol. 26, no. 5, pp. 5733–5778, Sep. 2021, doi: 10.1007/s10639-021-10570-8.

[8] N. Fraser, "Ten things we've learned from Blockly," in *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)*, Oct. 2015, pp. 49–50. doi: 10.1109/BLOCKS.2015.7369000.

[9] E. Pasternak, R. Fenichel, and A. N. Marshall, "Tips for creating a block language with blockly," in *2017 IEEE Blocks and Beyond Workshop (B&B)*, Oct. 2017, pp. 21–24. doi: 10.1109/BLOCKS.2017.8120404.

[10] T. Kluyver *et al.*, "Jupyter Notebooks—a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas (Proceedings of the 20th International Conference on Electronic Publishing)*, Göttingen, Germany: IOS Press, 2016, pp. 87–90. doi: 10.3233/978-1-61499-649-1-87.