Online-Journals.org

iJET International Journal of Emerging Technologies in Learning

iJET | eISSN: 1863-0383 | Vol. 20 No. 4 (2025) | OPEN ACCESS

https://doi.org/10.3991/ijet.v20i04.55567

PAPER

Explainable Artificial Intelligence with MicroPython: Lightweight Neural Networks for Students' Deeper Learning

Dennis Klinkhammer (✉)

FOM University of Applied Sciences, Cologne, Germany

dennis.klinkhammer@fom.de

ABSTRACT

This study explores the integration of neural networks onto resource-constrained microcontrollers in academic teaching and learning settings in order to enhance students' understanding of artificial intelligence. By leveraging MicroPython, the open-source framework AI-ANNE: (A) (N)eural (N)et for (E)xploration is used to facilitate the transfer of pre-trained models from high-level artificial intelligence libraries like TensorFlow and Keras to microcontrollers. A comparative analysis was conducted to assess students' comprehension of neural networks under different instructional methods. Statistical evaluations using Tukey's HSD post-hoc test revealed that students who implemented neural networks from scratch in MicroPython demonstrated significantly higher learning outcomes compared to those using TensorFlow and Keras in Python. However, no significant difference was observed between students who implemented MicroPython models on standard computing environments and those who deployed them onto microcontrollers. This suggests that explicit implementation in MicroPython fosters a deeper conceptual understanding of neural networks. While high-level artificial intelligence libraries like TensorFlow and Keras provide efficiency, they may obscure fundamental learning processes when it comes to trust, transparency, understandability, and usability as key concepts of explainable artificial intelligence approaches. Thus, direct implementation in MicroPython enhances comprehension and prepares students for responsible artificial intelligence development.

KEYWORDS

explainable artificial intelligence, neural networks, MicroPython, microcontroller

1 INTRODUCTION

Current research indicates that some students might perceive neural networks, a key component of artificial intelligence, as complex and opaque due to the abstraction provided by high-level artificial intelligence libraries such as TensorFlow and

Klinkhammer, D. (2025). Explainable Artificial Intelligence with MicroPython: Lightweight Neural Networks for Students' Deeper Learning. *International Journal of Emerging Technologies in Learning (iJET)*, 20(4), pp. 72–80. https://doi.org/10.3991/ijet.v20i04.55567

Article submitted 2025-03-17. Revision uploaded 2025-08-14. Final acceptance 2025-08-14.

Keras in the Python programming language [9]. While such tools are essential for practical applications, they often obscure fundamental concepts and mechanisms [8]. In contrast, the implementation of neural networks using MicroPython as a programming language is supposed to offer substantial educational benefits, particularly in academic teaching and learning settings [12]. Implementing neural networks in MicroPython requires students to explicitly define operations such as matrix multiplications, activation functions, and suitable performance parameters [9, 20]. This hands-on approach might demystify the mathematical principles underlying neural networks, allowing students to better understand the relationships between data, weights, biases, and related outputs [15, 16]. Furthermore, debugging the code without guidance by high-level artificial intelligence libraries might reinforce this knowledge and provide insights into how errors propagate and can be corrected. This might not only lead to students reflecting trust in artificial intelligence but might also increase transparency, understandability, and usability of artificial intelligence itself, which are defined as key criteria for any explainable artificial intelligence approach [7, 16].

1.1 Explainable artificial intelligence with MicroPython

Since MicroPython is an efficient, lightweight implementation of the Python programming language designed to run on microcontrollers and embedded systems with constrained resources [3], it offers a streamlined interpreter and a subset of Python's standard libraries. Therefore, students need to code neural networks from scratch, which might foster a transparent understanding of how data flows through the neural network, how parameters are adjusted, and how the architecture and learning processes work [9]. In order to make these artificial intelligence insights accessible to students with heterogeneous prerequisites, AI-ANNE: (A) (N)eural (N)et for (E)xploration can be used, which is specifically designed for academic teaching and learning settings. In this open source framework, neurons, layers, density, and activation functions as the underlying foundation of neural networks are pre-coded in MicroPython in order to be able to reconstruct the neural networks that were originally trained with TensorFlow and Keras in Python [7, 12]. While functional neural networks are already pre-programmed, they can simply be tried out and expanded by the students as required. Especially the elaborated foundations of data science and analytics by Fayyad and Hamutcu [4], as well as the investigation of skill requirements in artificial intelligence and machine learning job advertisements by Verma et al. [19], were inspiration for the development of AI-ANNE: (A) (N)eural (N)et for (E)xploration as an explainable artificial intelligence approach.

Furthermore, coding a neural network from scratch, rather than relying on high-level artificial intelligence libraries such as TensorFlow and Keras, offers the flexibility to tailor the implementation to specific needs [9]. This includes reducing the number of layers, optimizing the activation functions, even inventing new ones, or using specialized operations to reduce computational load [14]. While students can log and analyze intermediate results in MicroPython, such as hidden layer outputs and weight updates, providing a transparent view of the neural network's learning process [6], this step-by-step visibility in MicroPython might also help to identify and understand common challenges like vanishing or exploding gradients, fostering a more intuitive grasp of how neural networks operate [9]. Students can also explore how biases in training data might influence model behavior, the implications of design choices on interpretability, and the importance of building explainable

and accountable systems. As a result, this approach might foster students' critical thinking and creativity, which are valuable skills in the evolving artificial intelligence landscape [4, 19]. The use of MicroPython might even cultivate a deeper awareness of the societal impact of artificial intelligence and the responsibility of artificial intelligence practitioners [2, 6].

1.2 Artificial intelligence on microcontrollers

Machine Learning and Deep Learning are increasingly driving innovation across various fields [10], with a notable expansion into embedded systems, bringing their capabilities closer to data sources. This trend, known as TinyML, is especially prominent in microcontrollers and Internet of Things devices. TinyML offers several advantages over cloud-based artificial intelligence, including improved data privacy, lower processing latency, energy efficiency, and reduced dependency on connectivity [13]. One key application of TinyML is in condition monitoring, where neural networks can be used to detect anomalies directly within sensors so that immediate corrective actions can be taken automatically [1]. Therefore, when these microcontrollers are connected to microscopes in medical diagnostics or machines for industrial production, they are referred to as embedded systems, which can provide hands-on insights for students.

While many high-level artificial intelligence libraries like TensorFlow and Keras for Python are designed for powerful hardware such as GPUs, these frameworks are unsuitable for embedded systems due to their limited computational resources [3]. To address this issue, the architecture of neural networks needs to be reconstructed in a lightweight programming language such as MicroPython. Therefore, neural networks trained on high-performance hardware need to be transferred onto resource-constrained devices such as microcontrollers while achieving the same outputs and results [13]. However, training or developing neural networks directly on embedded systems remains a complex challenge. AIfES: (A)rtificial (I)ntelligence (f) or (E)mbedded (S)ystems has shown initial success here [18]. It is a flexible software framework designed by Fraunhofer to run deep learning models on small, low-power devices such as microcontrollers [18]. It simplifies the process of building, training, and running models directly on these devices, without needing powerful external systems, but without being specifically designed for academic teaching and learning settings.

As a result of their limited computational resources, microcontrollers introduce students to the challenges of optimizing code for efficiency and performance and could therefore be beneficial in terms of explainable artificial intelligence [3]. An example is the lack of microcontrollers to handle large-scale floating-point operations efficiently, which are typically used in neural network computations [13]. In order to address such issues, the previously stated transparency of neural networks coded in MicroPython might open up didactic application potential in terms of explainable artificial intelligence [2, 15]. As a result, there are distinct advantages to coding neural networks from scratch in MicroPython for microcontroller deployment. The primary benefit lies in full control over the network's architecture and computation, allowing for significant customizations that align closely with the constraints of the microcontroller. By working within these constraints, students might gain a deeper appreciation for the trade-offs involved in artificial intelligence development, particularly in the context of embedded systems. As such, this practical experience may close the gap between theoretical knowledge and real-world

applications [2, 11]. Moreover, working with microcontrollers makes artificial intelligence education more accessible, as these devices are inexpensive and widely available, lowering barriers for students and institutions with limited resources.

1.3 Research question

This study deals with the research question of whether students in academic teaching and learning settings can gain a better understanding of how neural networks operate by transferring neural networks from Python to MicroPython while coding directly on microcontrollers. The mentioned limitations of microcontrollers and the lack of high-level artificial intelligence libraries in MicroPython should therefore foster a more fundamental approach regarding the basics of neural networks. A pretest-posttest design will be used to investigate the following hypotheses:

- **H1:** Students who implement neural networks from scratch in MicroPython via Anaconda Cloud demonstrate higher learning outcomes compared to those using TensorFlow and Keras in Python.
- **H2:** Students who implement neural networks from scratch in MicroPython on microcontrollers demonstrate higher learning outcomes compared to those using TensorFlow and Keras in Python.
- **H3:** Students who implement neural networks from scratch in MicroPython on microcontrollers demonstrate higher learning outcomes compared to those using MicroPython via Anaconda Cloud.

A corresponding evaluation of different academic teaching and learning settings via the Tukey method [17] is therefore the methodological approach of this study.

2 METHODS

For the present analysis of the different academic teaching and learning settings, a learning assessment will be the instrument of measurement, and Tukey's HSD, the Honestly Significant Difference post-hoc test [17], will be the method of analysis. A description of the sample, the underlying design, and the intervention is explained in the following sections, and results will be based upon pretest and posttest comparison.

2.1 Sample description

A total of 144 students were randomly divided into three groups so that each group contained a total of 48 students. The students were first-year students with limited knowledge of how neural networks operate. Each group will be confronted with a different academic teaching and learning setting as intervention. Corresponding learning assessments in a pretest-posttest design were conducted within three consecutive hours. The pretest learning assessment was based on 20 questions (that accordingly resulted in a 20-point scale in which each correctly answered question yielded one point) on basic definitions and principles of neural networks as well as their interpretation (refer to Table 1):

Table 1. Pre-test learning assessment

	Group 1	Group 2	Group 3
Mean Score (SD)	02.54 (01.16)	02.62 (01.19)	02.50 (01.23)

The pretest before the intervention indicates no significant differences between the groups.

2.2 Intervention: A binary classification task

All groups were confronted with a binary classification task, based upon the IRIS dataset. The IRIS dataset is a widely recognized dataset in statistics, machine learning, and deep learning and is often used for classification problems. Introduced by the British biologist and statistician Fisher [5], it contains 150 instances of iris flowers, each with four attributes that describe their physical characteristics. These attributes include the sepal length, sepal width, petal length, and petal width, all measured in centimeters. Based on these four attributes, the objective of the dataset is to classify each flower into one of three species: 1) Setosa, 2) Versicolor, and 3) Virginica. The IRIS dataset serves as an ideal example for demonstrating and testing neural networks, particularly for classification tasks. Its manageable size, clear distinctions between classes, and straightforward nature make it a popular choice for exploring classification techniques. However, classifying some of the species, particularly Versicolor and Virginica, can be challenging due to their overlapping characteristics, making the task more complex for certain machine learning algorithms and neural networks, such as deep learning models.

The intervention was carried out in three different academic teaching and learning settings: In the first setting, students learn the basics of neural networks in Python and with recourse to high-level artificial intelligence libraries such as TensorFlow and Keras by using Anaconda Cloud. In the second setting, students familiarize themselves with neural networks in MicroPython and without high-level artificial intelligence libraries, but also in the Anaconda Cloud. The third setting will be without the Anaconda Cloud, and the students will code neural networks in MicroPython and on microcontrollers. The second and third settings will make use of the MicroPython codes provided by AI-ANNE: (A) (N)eural (N)et for (E)xploration, once with and once without the use of a microcontroller. It is the intention of the intervention that the architecture and performance of the neural networks did not differ, only the programming language and the additional use of a microcontroller were decisive for the differences between the academic teaching and learning settings.

3 RESULTS

For an assessment, the posttest learning assessment made use of slightly different questions compared to the pretest learning assessment, and Tukey's HSD will be used as a method of analysis. As a one-way ANOVA, it is similar in interpretation to the unpaired Student's t-test, which would be limited to the comparison of two groups [17]. First of all, it should be noted that all groups experienced a significant increase within the posttest learning assessment, probably as a result of the intervention. However, the main differences become apparent when the different academic teaching and learning settings are compared with each other (refer to Table 2):

Table 2. Post-test learning assessment

	Group 1	Group 2	Group 3
Mean Score (SD)	13.27 (02.35)	15.37 (02.38)	16.12 (02.33)

Based on the three groups, each with a different intervention, the following findings can be derived: The difference between group 1 and group 2, according to the Tukey method, is 02.10 (95% CI: 00.96 to 03.23) and significant (p = .0001). Furthermore, the difference between group 1 and group 3 is 02.85 (95% CI: 01.71 to 03.98) and also significant (p = .000). However, the difference between group 2 and group 3 is only 0.75 (95% CI: –0.38 to 1.88) and not significant (p = .2659). From this it can be deduced that programming a neural network in MicroPython contributes more to understanding the underlying functionality than an approach with TensorFlow and Keras in the Anaconda Cloud, which suggests that the data sufficiently supports hypotheses H1 and H2. However, the use of an additional microcontroller does not seem to contribute to a better understanding of how neural networks work, which suggests that the data does not sufficiently support hypothesis H3. Regarding the rejection of hypothesis H3, it should be noted that the students were confronted with two predefined neural networks as part of the intervention, which—for the purpose of comparability—did not exhaust the limitations of the microcontroller.

4 DISCUSSION

This study examined the integration of neural networks into resource-constrained microcontrollers to create a cost-effective learning environment and to enhance students' understanding of artificial intelligence within academic teaching and learning settings. To facilitate this, the open-source framework AI-ANNE: (A) (N)eural (N)et for (E)xploration was used as intervention, providing a transparent and lightweight solution for transferring pre-trained models from high-level artificial intelligence libraries such as TensorFlow and Keras to microcontrollers using MicroPython. Thus, by implementing key neural network components such as neurons, layers, density, and activation functions directly in MicroPython, students get in touch with core concepts of artificial intelligence, in accordance with the explainable artificial intelligence goals such as trust, transparency, understandability, and usability.

Therefore, the study analyzed the impact of different teaching and learning approaches on students' comprehension of neural networks, particularly in the context of explainable artificial intelligence. While all students exhibited significant learning gains through the provided academic teaching and learning settings, key differences emerged based on the methodological approach used. Statistical analysis using Tukey's HSD post-hoc test revealed that students who implemented neural networks using TensorFlow and Keras in Python gained relatively lower learning outcomes than those who programmed the same models from scratch in MicroPython. An even greater discrepancy was observed when comparing the TensorFlow and Keras group with students who executed their MicroPython implementations directly on a microcontroller. However, among students utilizing MicroPython, no statistically significant difference was found between those who ran their models on a standard computing environment and those who deployed them on microcontrollers.

These findings suggest that implementing neural networks in MicroPython, regardless of whether a microcontroller is used or not, leads to a better conceptual grasp of neural networks compared to relying on high-level artificial intelligence libraries like TensorFlow and Keras.

From an educational and practical standpoint, the findings of this study highlight key implications regarding the explainable artificial intelligence approach. First, the relatively lower learning outcomes associated with using TensorFlow and Keras suggest that the high level of abstraction in these frameworks makes it more challenging to internalize model architectures and learning processes. Direct implementation in MicroPython fosters a deeper, more transparent understanding of neural network functionality, which might benefit trust and transparency in artificial intelligence outputs. Second, the superior learning outcomes of students using MicroPython highlight that explicit implementation might enhance conceptual clarity. By requiring students to manually construct models, this approach seems to strengthen fundamental knowledge about artificial intelligence, which addresses the goal of understandability of the explainable artificial intelligence approach. The third point is aimed at the usability of artificial intelligence: While TensorFlow and Keras offer ease of use, they may inadvertently obscure key principles, particularly for students with limited programming experience. A balanced educational strategy should integrate both high-level efficiency and low-level comprehension to optimize learning outcomes. In summary, this method transforms artificial intelligence education into a transparent and immersive process, preparing students for both academic research and professional applications while encouraging responsible engagement with artificial intelligence technologies. Notably, the neural networks deployed on microcontrollers achieve the same results as their TensorFlow and Keras counterparts, yet offer a more interpretable and accessible approach in terms of explainable artificial intelligence.

In conclusion, this study demonstrates that implementing neural networks in MicroPython, with and without the additional use of microcontrollers, can serve as an effective educational tool for enhancing explainable artificial intelligence. By bridging theoretical concepts with practical application, this approach empowers students with the knowledge and skills necessary to critically engage with artificial intelligence technologies. Despite these promising results, certain limitations must be acknowledged. The study focused on only two predefined neural network models and did not comprehensively explore the constraints of microcontrollers in artificial intelligence applications. Future research should investigate whether further optimization and implementation of neural networks on microcontrollers enhance students' understanding even further. Additionally, expanding the scope of network architectures and increasing sample diversity could yield more generalized insights.

5 OPEN-SOURCE CODE

The Micropython code for the intervention and AI-ANNE: (A) (N)eural (N)et for (E)xploration is available online: https://www.statistical-thinking.de/ai-anne.html. Included are Jupyter Notebooks for neural networks with TensorFlow and Keras in Python, as well as the MicroPython code for the same neural networks that can be transferred to microcontrollers.

6 REFERENCES

[1] R. Cioffi, M. Travaglioni, G. Piscitelli, A. Petrillo, and F. De Felice, "Artificial intelligence and machine learning applications in smart production: Progress, trends, and directions," *Sustainability*, vol. 12, no. 2, p. 492, 2020. https://doi.org/10.3390/su12020492

[2] C. Collier and A. Powell, "Data analyst competencies: A theory-driven investigation of industry requirements in the field of data analytics," *Journal of Information Systems Education (JISE)*, vol. 35, pp. 325–376, 2024. https://doi.org/10.62273/SPYC4248

[3] G. Delnevo, S. Mirri, C. Prandi, and P. Manzoni, "An evaluation methodology to determine the actual limitations of a TinyML-based solution," *Internet of Things*, vol. 22, p. 100729, 2023. https://doi.org/10.1016/j.iot.2023.100729

[4] U. Fayyad and H. Hamutcu, "Toward foundations for data science and analytics: A knowledge framework for professional standards," *Harvard Data Science Review*, vol. 2, no. 2, 2020. https://doi.org/10.1162/99608f92.1a99e67a

[5] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936. https://doi.org/10.1111/j.1469-1809.1936.tb02137.x

[6] M. Frank, D. Drikakis, and V. Charissis, "Machine-learning methods for computational science and engineering," *Computation*, vol. 8, no. 1, p. 15, 2020. https://doi.org/10.3390/computation8010015

[7] A. B. Haque, A. K. M. N. Islam, and P. Mikalef, "Explainable artificial intelligence (XAI) from a user perspective: A synthesis of prior literature and problematizing avenues for future research," *Technological Forecasting and Social Change*, vol. 186, p. 122120, 2023. https://doi.org/10.1016/j.techfore.2022.122120

[8] D. Klinkhammer and K. Keller, "Grundlagenwissen Künstliche Intelligenz," in *Kompetenzen für die Arbeitswelten der Zukunft*, S. Fichtner-Rosada, T. Heupel, C. Hohoff, and J. Heuwing-Eckerland, Eds., FOM-Edition, Springer Gabler, Wiesbaden, 2024, pp. 327–342. https://doi.org/10.1007/978-3-658-44959-9_21

[9] S.-C. Kong, W. M.-Y. Cheung, and G. Zhang, "Evaluating artificial intelligence literacy courses for fostering conceptual learning, literacy and empowerment in university students: Refocusing to conceptual building," *Computers in Human Behavior Reports*, vol. 7, p. 100223, 2022. https://doi.org/10.1016/j.chbr.2022.100223

[10] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015. https://doi.org/10.1038/nature14539

[11] G. Li, C. Yuan, S. Kamarthi, M. Moghaddam, and X. Jin, "Data science skills and domain knowledge requirements in the manufacturing industry: A gap analysis," *Journal of Manufacturing Systems*, vol. 60, pp. 692–706, 2021. https://doi.org/10.1016/j.jmsy.2021.07.007

[12] C. Meske, E. Bunde, J. Schneider, and M. Gersch, "Explainable artificial intelligence: Objectives, stakeholders, and future research opportunities," *Information Systems Management*, vol. 39, pp. 53–63, 2020. https://doi.org/10.1080/10580530.2020.1849465

[13] P. P. Ray, "A review on TinyML: State-of-the-art and prospects," *Journal of King Saud University – Computer and Information Sciences*, vol. 34, pp. 1595–1623, 2022. https://doi.org/10.1016/j.jksuci.2021.11.019

[14] F. Sakr, R. Berta, J. Doyle, A. De Gloria, and F. Bellotti, "Self-learning pipeline for low-energy resource-constrained devices," *Energies*, vol. 14, p. 6636, 2021. https://doi.org/10.3390/en14206636

[15] R. Scherer, "Learning from the past–the need for empirical evidence on the transfer effects of computer programming skills," *Front. Psychol.*, vol. 7, 2016. https://doi.org/10.3389/fpsyg.2016.01390

[16] P. Schmidt, F. Biessmann, and T. Teubner, "Transparency and trust in artificial intelligence systems," *Journal of Decision Systems*, vol. 29, pp. 260–278, 2020. https://doi.org/10.1080/12460125.2020.1819094

[17] J. W. Tukey, "Comparing individual means in the analysis of variance," *Biometrics*, vol. 5, pp. 99–114, 1949. https://doi.org/10.2307/3001913

[18] L. Wulfert *et al.*, "AIfES: A next-generation edge AI framework," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, pp. 4519–4533, 2024. https://doi.org/10.1109/TPAMI.2024.3355495

[19] A. Verma, K. Lamsal, and P. Verma, "An investigation of skill requirements in artificial intelligence and machine learning job advertisements," *Industry and Higher Education*, vol. 36, pp. 63–73, 2021. https://doi.org/10.1177/0950422221990990

[20] M. Yin, J. Wortman Vaughan, and H. Wallach, "Understanding the effect of accuracy on trust in machine learning models," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019, pp. 1–12. https://doi.org/10.1145/3290605.3300509

7 AUTHOR

Dennis Klinkhammer is a Professor of Social Sciences, especially Empirical Research at the Institute for Empirical Research and Statistics, FOM University of Applied Sciences, Cologne, Germany (E-mail: dennis.klinkhammer@fom.de).