2018 ASEE ANNUAL CONFERENCE & EXPOSITION
JUNE 24–27, 2018
SALT PALACE CONVENTION CENTER
SALT LAKE CITY, UT

Paper ID #22725

Work in Progress: Designing Laboratory Work for a Novel Embedded AI Course

Dr. Mehmet Ergezer, Wentworth Institute of Technology

Mehmet Ergezer (S'06) received the B.S. and M.S. degrees in electrical and computer engineering from Youngstown State University, Youngstown, OH, USA, in 2003 and 2006, respectively. He received the D.Eng. degree in artificial intelligence from the Department of Electrical and Computer Engineering, Cleveland State University, Cleveland, OH, USA, in May 2014.

From 2003 to 2005, following his internship with U.S. Steel, he was a Graduate Assistant with Youngstown State University. In 2006, he was a Research Assistant with the Embedded Control Systems Research Laboratory, Cleveland State University, engaged in heuristic numerical optimization techniques. In 2008, he interned with the Digital Engineering Team, Philips Healthcare. In 2011, he worked on the the development of tracking algorithms for civilian aircraft as a Staff Engineer for ARCON in Waltham, MA, USA. In 2014, Dr. Ergezer joined the Research and Advanced Development signal processing team for Bose Corp. In 2017, he became an Assistant Professor for the Department of Computer Science and Networking at Wentworth Institute of Technology.

Dr. Ergezer is a member of ACM and Eta Kappa Nu.

Bryon Kucharski, Wentworth Institute of Technology

Senior computer engineering student at Wentworth Institute of Technology

Prof. Aaron Carpenter, Wentworth Institute of Technology

Professor Carpenter is an Assistant Professor at the Wentworth Institute of Technology. In 2012, he completed his PhD at the University of Rochester. He now focuses his efforts to further the areas of computer architecture, digital systems, cybersecurity, and computer engineering education.

Work-in-Progress - Designing Laboratory Work for a Novel Embedded AI Course

Mehmet Ergezer, Bryon Kucharski, Aaron Carpenter
{ergezerm, kucharskib, carpentera1}@wit.edu
Wentworth Institute of Technology

Abstract

Laboratory work is essential for students in the Science, Technology, Engineering, and Mathematics (STEM) fields. The laboratory work provides the students with practical experience of the theory and has the potential to increase enthusiasm by motivating the students to learn via experimentation. This is an important process for students as the active learning achieves positive educational results and prepares the students for real-world problems in the STEM fields.

This paper emphasizes the development of the Artificial Intelligence (AI) laboratory work for a new course in Embedded Artificial Intelligence (EAI). The course and the laboratory work are designed as an upper-level undergraduate elective to develop an AI system to run on an embedded device. It is open to all majors in the STEM fields who meet the prerequisite of a basic programming course and a linear algebra course. After an introduction to embedded programming and sensor interface, students will be introduced to machine learning and AI. During the corresponding lab sessions, the students are given a dataset to apply the theory in a sequential fashion. The laboratories start by employing traditional statistical classification algorithms, such as logistic regression, transitioning towards a deep neural network, such as a convolutional neural network (CNN). The accuracy of each model will be noted for each laboratory, starting with a lower accuracy but smaller model size for the statistical models and culminating with a relatively high accuracy with the CNN.

This paper outlines the design of the laboratories for the AI section of the EAI course, as well the feedback received during their development. The research to create and assess the AI exercises was conducted by a senior computer engineering student without any prior experience in AI. Two different forms of learning were taken into consideration: top-down approach where the student begins with a fully functional model and works backwards to understand each step of the processes and a bottom-up approach where the student begins from scratch and implements each component, working towards a fully functionally model. A comparative study of both approaches is presented from the point of view of the student. The assessment also asked the student to rate the assignment topics, to list how many hours were spent per each lab, and to propose suggestions for improvement.

1 Introduction

Laboratory work is essential for students in the Science, Technology, Engineering, and Mathematics (STEM) fields, and its importance is well-studied [1, 2]. Laboratory assignments offer students opportunities for practical applications of theory and have the potential to promote knowledge acquisition via experimentation. Hands-on learning is an important process for students, as active learning achieves positive educational results and prepares students for real-world problems in the STEM fields [3]. Laboratories allow students to frequently engage with the lecture material in a learner-centered, low-stakes environment that can enhance learning outcomes [4, 5, 6].

Traditionally, labs are offered in a face-to-face setting and designed to provide a hands-on, concrete experience with real systems. However, virtual and remote labs are becoming increasingly popular alternatives to traditional labs as they can be conducted at a lower cost (due to reduced hardware and space requirements) and reach a wider audience thanks to distance learning [7]. In this paper, we introduce the framework associated with developing laboratory material within the curriculum for a novel course. The course and its lab will be offered on the university premises, but the labs being developed will be applicable to both in-class and remote educational settings in order to widen their accessibility (to students with disabilities) and availability (to students with time or geographical constraints).

The proposed course is titled “Embedded Artificial Intelligence” (EAI) [8]. The course and the laboratory work are designed as an upper-level undergraduate fifteen-week-long elective to develop an AI system to run on an embedded device. It is open to all majors in the STEM fields who meet the prerequisite of basic programming knowledge. We have removed the original prerequisite of linear algebra to allow students from different majors to register for the course. We are preparing programming labs as an introductory course component that will cover topics from linear algebra alongside Python and Numpy.

Many undergraduate programs include artificial intelligence (AI) or embedded systems in their CS curricula [9, 10, 11, 12]. However, the authors could not find a course that would combine both topics. As the AI algorithms make their way into wearables, smart phones and Internet of Things (IoT) devices, having experience in real-time AI algorithms will benefit students in their professional careers.

The course is still under preparation and is planned to be offered in the summer of 2018. We are sharing our work-in-progress that illustrates the development of the labs for EAI. Two different forms of learning were taken into consideration: “top down” approach, where the student begins with a fully functional model and works backwards to understand each step of the processes and a “bottom up” approach, where the student begins from scratch and implements each component, working towards a fully functional model. A comparative study of both approaches is presented from the point of view of the student.

The rest of the paper is organized as follows: Section 2 explores topics covered in the proposed course. Section 3 discusses the lab assignments for the course; section 4 compares top-down and bottom-up approaches to the labs from the perspective of the undergraduate student researcher. Concluding remarks are presented in section 5.

2 Course Curriculum

The topics presented in the course can be divided into four categories: background, embedded algorithms, AI, and the final project. This section will introduce the topics under each of the four categories as well as provide references to the teaching materials.

It is possible for our departments to provide hardware to students, but that was not done in this preliminary case. Thus, the course will require students to purchase their own hardware to build a final project. To offset this cost, the class will rely on open source teaching materials when available. Consequently, most of the referenced books are available for free and provide the necessary technical content that is appropriate for our students' level.

Section 1: Background (Weeks 1-3)
This section refreshes analytical and programming topics that are needed for EAI.

In terms mathematical content, we will start with statistics, introducing topics such as averages and standard deviation [13, 14]. We will then cover topics such as matrix multiplication, matrix inverse, and eigenvalues/vectors from linear algebra [15, 16]. Simultaneously, we will also present the programming language of choice for the class: Python [17].

Section 2: Embedded algorithms (Weeks 4-7)
The second part of the course will focus on algorithms that can be deployed in near-real-time on embedded systems and on how to prepare the sensor data for processing. This will include presenting the commonly employed sensors, such as accelerometers, gyroscopes, ultrasonic sensors, and cameras [18], and the control and interface schemes, such as I^2S, I^2C, and SPI, necessary to interact with these sensors. We will then present common filtering algorithms, such as the alpha-beta filters or Kalman filters, used to minimize sensor noise [19]. Other practical real-time filters such as the moving average filter may also be discussed.

Section 3: AI (Weeks 8-12)
The next instructional part of the course presents selected topics from AI. We prioritize two AI problems: dimensionality reduction and supervised learning. We explain that we can minimize the computational power required to process data by extracting a subset of features using statistical learning algorithms, such as principal component analysis (PCA) and linear discriminant analysis (LDA) [20].

We, then, introduce prevailing techniques such as linear regression and linear classification that learn from labeled training data. These allow students to understand the underlying structure of neural-networks [21]. Finally, we introduce deep learning architectures, such as convolutional neural networks, and discuss their implementation via modern libraries, such as TensorFlow [22] and Keras [23].

Section 4: Final project (Weeks 13-15)
At the end of the fifteen-week semester, students present their projects that demonstrate their understanding of the AI topics and ability to implement it in an embedded system.

3 Laboratory Assignments

Many of the topics presented in Section 2 will have a lab assignment to promote a hands-on application of the theories presented during the lectures. Lab sessions are two hours long and conducted every week with the professor present in the class to offer assistance to students.

We are working on developing many of the labs to be programming-based and auto-graded. We are evaluating Jupyter notebooks and nbgrader to allow students to code, visualize their result, read instructions, and provide written responses [24]. The philosophy behind nbgrader sprouts from the belief that visualization aids understanding of complex information-processing systems [25]. Our current plan is to teach the course on-site and have lab time with the professor present during the labs to guide the students. However, nbgrader would enhance adaptability of lab assignments by providing scalability in student size and allowing remote learning.

Section 1: Background labs
The objective of these labs is to provide students with experience with Python programming language and some of the required mathematics knowledge, such as linear algebra and probabilities. For instance, the first lab can require students to install the tools and submit a sigmoid method as our “hello world!” and develop a function for multiplying rectangular matrices.

Once students can perform matrix operations comfortably, they will start using the computational library, NumPy, and will work with the visualization library Matplotlib to implement the analytical methods for image processing assignments. For this lab, students can experiment with cropping, transposing, and filtering images, such as converting images to gray scale.

In order to get experience with statistics, students will experiment with offline and online versions of mean and normalization algorithms and study their differences. These filters can be implemented on the popular machine learning datasets, such as breast cancer diagnosis or the wine dataset [26, 27].

Section 2: Embedded algorithms
After the students learn about the necessity of online algorithms, they will work on a real-time algorithm to estimate the steps taken by a user from cell phone accelerometer and gyroscope data [26]. They will then test their algorithm using the data collected from their own smart phones for different gaits.

We are working on a lab to implement a recursive filter to estimate the correct state using noisy sensor data. This approach is commonly employed in to estimating speed of a vehicle or estimating position based on space-based radio navigation data.

Section 3: AI labs
Students will refer to either the breast cancer diagnosis dataset or the wine dataset to test their implementation of the dimensionality reduction algorithms.

For classification and regressions experiments, we have created a subset of the dogs and cats dataset with 200 images [21] where students can achieve limited success using linear regression, linear classification and artificial and deep neural networks. Students will also practice visualizing classifier results using the confusion matrix. Their sigmoid and matrix multiplication methods from Section 1 labs will aid them in implementing the AI algorithms.

Finally, students will train convolutional neural networks on the cloud to classify the full dogs vs. cats dataset with 23,000 images and download their models to run on their laptops.

Section 4: Final project
Students will form groups to apply their knowledge from Sections 2 and 3 to solve a problem of their own choosing that runs in an embedded system or is a near-real-time application.

4 Top-down vs. Bottom-up Learning

STEM education researchers and practitioners have long been studying effective learning methods in computer science (CS) [28, 29]. Two conventional approaches to teaching programming are the top-down and bottom-up techniques. The top-down approach is applied in engineering as the analysis of the entire system before detailing the subsystems. Bottom-up approach refers to implementing the subsystems in increments and building up to the entire system. In the context of STEM education, a top-down approach can be defined as understanding the outcome before being exposed to the underlying implementations while a bottom-up approach is defined as implementation in small increments, building up to a full implementation. As [29] explains, "bottom-up approach primarily focuses on teaching the details of syntax and individual programming language elements first. After individual elements have been taught, more complex constructs are considered. Top-down approach starts with understanding the abstractions regardless of their physical implementation. After students understand these abstractions and their purpose, implementations are being taught."

Many CS classes are traditionally taught using the bottom-up approach. However, strong arguments in favor of the top-down approach have also been found in literature. [30] states that students "will feel uncomfortable for a while with the top-down approach as it forces them to deal with the abstraction so much of the time. Immersing them in this way of thinking should get them over this hurdle faster." [31] concludes that modern object-oriented programming languages are well-suited for the top-down approach and should facilitate teaching from object-oriented languages to procedural paradigm. On the other hand, some instructors point out that teaching the higher-level material can have its shortcomings. For instance, [32] found that when students were given a system with a top-level design, "there is little incentive for students to take the time to examine the internal construction of the completed IP core subcomponent modules even though the full HDL source code for these models was given to them."

We have experimented with both approaches with the help of our undergraduate student researcher (USR). From the perspective of the USR, the most efficient method of learning the new material is a hybrid of both the top-down and bottom-up strategies. The use of a hybrid instructional model will utilize the benefits from both approaches. A top-down style is beneficial in giving students an understanding of the bigger picture and encouraging them to determine the implementation steps. A bottom-up approach may be less complex than learning a state-of-the-art design and offer students insights into the shortcomings of the less-sophisticated method. Based on an investigation of unsuccessful and successful elements of different algorithms, students may have a better understanding on why the state-of-the-art design exists.

The course taught from a top-down approach could have students run classification methods with the greatest accuracy at the beginning of the course before any other knowledge is gained. The purpose of this instructional sequence would be to explain to students the current state of the art for classification. Then, students could dive into the implementation of the state-of-the-art designs. In contrast, the course taught from a bottom-up approach would have students begin at the lowest level using potentially less accurate methods of classification and work towards the state-of-the-art implementations that might provide higher accuracy but are more complicated. While initial bottom-up assignments may be more error-prone, an advantage of this approach would be to allow students to build knowledge in small increments relying on previously acquired knowledge. To make the most of the benefits that stem from the two instructional methods, the laboratory work for the embedded artificial intelligence course that is currently being developed will rely on a hybrid approach in hopes to improve dissemination of new knowledge and learning outcomes.

In general, the laboratory work will follow a bottom-up approach, starting with a lower accuracy, easier implementation methods and progress toward a higher accuracy, more complex implementation. Our hybrid approach is to first solve the given problem using a pre-build library, such as SciKit [33]. The second part of each lab will ask students to implement the same functionality from scratch using computational libraries such as NumPy. The purpose of this design is to give students an understanding of why the topic is important, in terms of accuracy, at the beginning of each lab. This will encourage students to implement the experiment from scratch as well as give them an accuracy to match in their own implementations. For instance, working with the breast cancer diagnostic dataset, students are provided with 30 real-valued features, such as the radius and the area of the cell nucleus. The students' task is to reduce the problem dimensionality from 30 to 2 using PCA. The result for the PCA can be obtained using SciKit's built-in function as plotted in Figure 1. Once students are familiar with the dataset, they implement their own PCA algorithm and compare their findings to SciKit. They also have to utilize their online normalization function from a previous lab to achieve a desirable reduction.

A similar hybrid approach can be applied to classification algorithms. Students will analyze the same 30 features for 569 patients and employ linear regression to predict the diagnosis, benign or malignant, for each patient. Figure 2 illustrates the performance of the algorithms as the training error minimizes. Figure 3 plots the confusion matrix for the designed model on the validation dataset. Both SciKit and low-level implementation is expected to classify over 90% of the cases correctly.

Figure 1: Top two principal components of breast cancer diagnostic dataset after normalizing the data

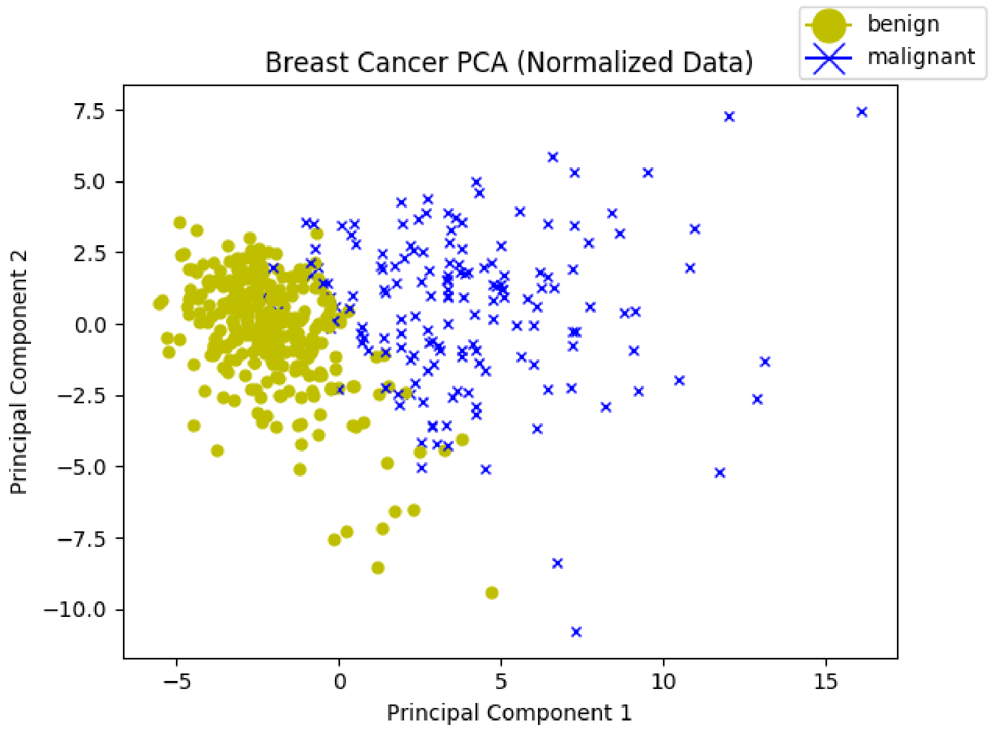


Figure 2: Training the logistic regression algorithm for the breast cancer diagnosis. Both SciKit and student implementations should perform similarly.

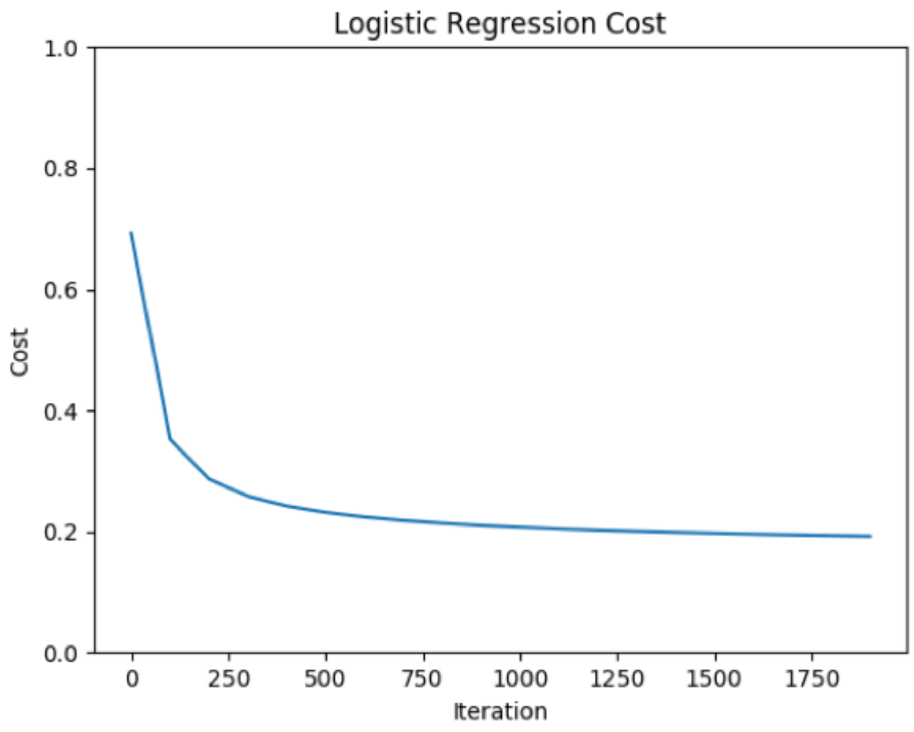
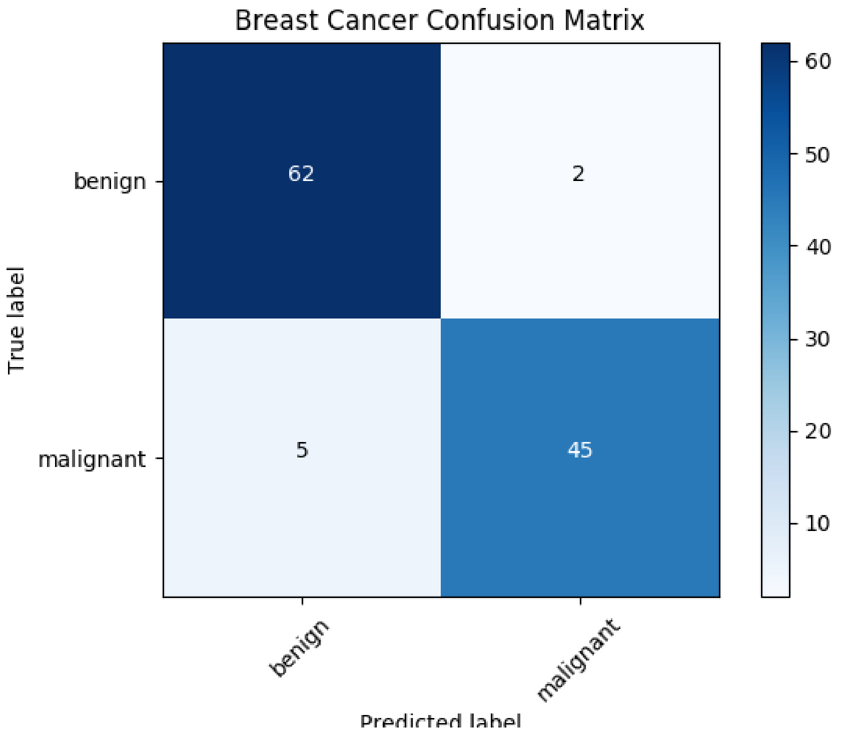


Figure 3: The performance of the designed linear regression algorithm is visualized using a confusion matrix for the breast cancer diagnostic dataset



5 Conclusions

This study presents our rationale for and design of the laboratory materials for a novel course titled Embedded AI. STEM education research suggests that lab work is a critical component of active learning and indicates that hands-on application of theory helps students to better understand and retain the course content. We have investigated two prevalent approaches for lab design: bottom-up and top-down. Based on the feedback from the USR, we have decided to root our laboratory assignments in a hybrid instructional model. This implies allowing students to take advantage of pre-build computational libraries to solve given problems. This approach enables students to understand the data and to visualize the expected output at a faster pace compared to designing their own functions. Once students gain confidence in their skills, they are expected to replace the library call with their own implementation.

We believe that this hybrid approach will promote students' interest in the problems that they are solving, which, in turn, may improve learning outcomes. We are also enabling the labs to be auto-graded to allow for distance learning. However, the professor will be available in the classroom to help students during the lab.

After we offer the class in the summer of 2018, we are planning to update our findings. An assessment will ask students to rate the assignment topics, list how many hours were spent per each lab, and propose suggestions for improvement.

It is our hope that this novel course with its laboratory component will equip students with the necessary tools for success in the rapid growing field of AI and prepare them for a rewarding career in the STEM fields.

References

[1] M. Abdulwahed and Z. K. Nagy, “Applying Kolb’s experiential learning cycle for laboratory education,” *Journal of engineering education*, vol. 98, no. 3, pp. 283–294, 2009.

[2] L. D. Feisel and A. J. Rosa, “The role of the laboratory in undergraduate engineering education,” *Journal of Engineering Education*, vol. 94, no. 1, pp. 121–130, 2005.

[3] J. Ma and J. V. Nickerson, “Hands-on, simulated, and remote laboratories: A comparative literature review,” *ACM Computing Surveys (CSUR)*, vol. 38, no. 3, p. 7, 2006.

[4] M. A. McDaniel, J. L. Anderson, M. H. Derbish, and N. Morrisette, “Testing the testing effect in the classroom,” *European Journal of Cognitive Psychology*, vol. 19, no. 4-5, pp. 494–513, 2007.

[5] S. Freeman, E. O’Connor, J. W. Parks, M. Cunningham, D. Hurley, D. Haak, C. Dirks, and M. P. Wenderoth, “Prescribed active learning increases performance in introductory biology,” *CBE-Life Sciences Education*, vol. 6, no. 2, pp. 132–139, 2007.

[6] S. Freeman, D. Haak, and M. P. Wenderoth, “Increased course structure improves performance in introductory biology,” *CBE-Life Sciences Education*, vol. 10, no. 2, pp. 175–186, 2011.

[7] R. Heradio, L. de la Torre, D. Galan, F. J. Cabrerizo, E. Herrera-Viedma, and S. Dormido, “Virtual and remote labs in education: A bibliometric analysis,” *Computers & Education*, vol. 98, pp. 14–38, 2016.

[8] M. Ergezer, B. Kucharski, and A. Carpenter, “Curriculum design for a multidisciplinary embedded artificial intelligence course,” in *49th ACM technical symposium on computer science education*, 2018.

[9] A. Salminen, J.-M. Vanhatupa, and H.-M. Järvinen, “Framework for embedded programming course,” in *Proceedings of the 11th Koli Calling International Conference on Computing Education Research*, Koli Calling ’11, (New York, NY, USA), pp. 54–59, ACM, 2011.

[10] J.-M. Vanhatupa, A. Salminen, and H.-M. Järvinen, “Organizing and evaluating course on embedded programming,” in *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, Koli Calling ’10, (New York, NY, USA), pp. 112–117, ACM, 2010.

[11] Joint Task Force on Computing Curricula, Association for Computing Machinery (ACM) and IEEE Computer Society, *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. New York, NY, USA: ACM, 2013. 999133.

[12] M. Sahami, “Computer Science Curricula 2013 (CS2013): AI and the Intelligent Systems Knowledge Area,” *AI Matters*, vol. 1, pp. 4–5, Mar. 2015.

[13] A. B. Downey, *Think Bayes: Bayesian Statistics in Python*. Green Tea Press, 1 ed., 2013.

[14] A. B. Downey, *Think Stats: Exploratory Data Analysis*. Green Tea Press, 2 ed., 2014.

[15] J. Hefferon, *Linear Algebra*. Orthogonal Publishing L3C, 3 ed., 2017.

[16] P. N. Klein, *Coding the Matrix: Linear Algebra through Applications to Computer Science*. Newtonian Press, 2013.

[17] A. B. Downey, *Think Python: How to Think Like a Computer Scientist*. Green Tea Press, 2 ed., 2015.

[18] A. B. Downey, *Think DSP: Digital Signal Processing in Python*. Green Tea Press, 1 ed., 2016.

[19] D. Simon, *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[20] J. VanderPlas, *Python data science handbook: Essential tools for working with data*. O'Reilly Media, Inc., 2016.

[21] J. Elson, J. J. Douceur, J. Howell, and J. Saul, "Asirra: a captcha that exploits interest-aligned manual image categorization," 2007.

[22] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning.," in *OSDI*, vol. 16, pp. 265–283, 2016.

[23] F. Chollet *et al.*, "Keras," 2015.

[24] J. B. Hamrick, "Creating and grading ipython/jupyter notebook assignments with nbgrader," in *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, pp. 242–242, ACM, 2016.

[25] D. Marr, "Vision: The philosophy and the approach," 1982.

[26] M. Lichman, "UCI machine learning repository," 2013.

[27] M. Forina, R. Leardi, C. Armanino, and S. Lanteri, "Parvus: an extendable package of programs for data exploration, classification and correlation, version 1.1," 1990.

[28] L. E. Winslow, "Programming pedagogy—a psychological overview," *ACM Sigcse Bulletin*, vol. 28, no. 3, pp. 17–22, 1996.

[29] S. Mohorovičić and V. Strčić, "An overview of computer programming teaching methods," in *XXII Central European Conference on Information and Intelligent Systems*, pp. 1–6, 2011.

[30] M. M. Reek, "A top-down approach to teaching programming," in *ACM SIGCSE Bulletin*, vol. 27, pp. 6–9, ACM, 1995.

[31] R. Decker and S. Hirshfield, "Top-down teaching: object-oriented programming in cs 1," in *ACM SIGCSE Bulletin*, vol. 25, pp. 270–273, ACM, 1993.

[32] E. Wells, "Using bottom-up techniques in a digital hardware design laboratory to better facilitate experiential learning," in *Proceedings of the 11th International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS'15)*, 2015.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.