Leveraging Novel Machine Learning in Engineering Education

Dr. JAMES WANLISS, Anderson University

James Wanliss is professor of general engineering at Anderson University. He is a winner of the NSF CAREER award, and works in experimental and computational plasma fluids, with interests in machine learning and data analysis.

©American Society for Engineering Education, 2024

Leveraging Novel Machine Learning in Engineering Education

Dr. James Wanliss, Professor
College of Engineering
Anderson University, SC

Abstract
It is vital to guarantee that engineering graduates have learned essential skills required to excel in a dynamic technological landscape. Today the proliferation of low-cost, high-speed computing devices offer opportunities for design and control of systems with varying levels of complexity. What this means in practice is that engineers increasingly need expert knowledge of various computer systems and software. Computing expertise once considered arcane must now become commonplace. We develop a novel Machine Learning (ML) course, designed for all undergraduate engineering majors with appropriate programming and mathematics background, to take as an elective in their junior or senior year. The course introduces deep learning and artificial intelligence (AI) as a basic tool engineers need to understand and utilize, even in an undergraduate engineering setting. Our paper shows how this course can be implemented in a new College of Engineering. The course uses the PyTorch machine learning framework as focus to guide students from basic ML concepts to the full deployment of models relevant to different areas of engineering.

1. Introduction

In the 21st century, a vital role for universities is to guarantee that by the time of graduation students have learned essential skills required to excel in a dynamic technological landscape. This is also true in engineering where our incoming students, though they have some technological experience, lack a specialized understanding of computers and programs relevant to the industry. Engineering systems feature intricate practical challenges for design and control. From the time that James Clerk Maxwell and Josiah Gibbs [1][2] elucidated engineering governors one can see the future introduction of computers into the realm of engineering, by the solution of mathematical equations, was virtually certain. Today the proliferation of low-cost, high-speed computing devices offer opportunities for design and control of systems with varying levels of complexity. What this means in practice is that engineers increasingly need expert knowledge of various computer systems and software. Computing expertise once considered arcane must now become commonplace.

Near the end of the 20th century it became the norm for undergraduate students to learn utilitarian programming software tools in order to accelerate their workflow [3][4]. Now, just in the past few years, deep learning and artificial intelligence (AI) has come to the place where it is no longer a wishful dream but a definitively powerful tool for addressing intricate scientific and engineering challenges. Thus, understanding and utilizing deep learning in undergraduate engineering education has become increasingly relevant, though still not widely practiced. Since it has become a core technology of today's Fourth Industrial Revolution (4IR or Industry 4.0) [21] it seems inconceivable that knowledge of fundamental deep learning tools should only be the realm of data science specialists or graduate students [32]. Accordingly, we propose to

develop a course in deep learning suitable for undergraduate engineering majors. Our paper will show how this might be implemented in a new College of Engineering at Anderson University, and how other schools may leverage our experiences.

There are many reasons why undergraduates will benefit from meaningful introduction to deep learning from an engineering perspective. But first it is necessary to define what is meant by 'deep learning'. In this paper deep learning refers to a subset of machine learning (ML) that involves using mathematical algorithms, specifically artificial neural networks, to assimilate and analyze large data sets and complex data patterns.

Given this definition, deep learning offers solutions to complex and intricate problems that may be challenging for typical algorithms in the traditional engineering curriculum. With the explosive proliferation of embedded systems, and the IoT, controlling complex actuators and sensors in real time is increasingly feasible. Many engineering applications generate vast amounts of data, and deep learning algorithms are effective at recognizing patterns and extracting meaningful insights from these large datasets [5]. Thus, an appreciation of deep learning methods will contribute to better engineering decision-making processes [6]. In addition, deep learning enables better automation, optimization, and control [7]. From design optimization to production planning, engineers can leverage deep learning to streamline workflows and enhance efficiency [8]. Fields like computer vision and signal processing benefit from deep learning with great success in areas as diverse as medical imaging, surveillance, and communications. In this regard, engineers involved in robotics and automation can benefit from deep learning in tasks such as object recognition, path planning, and control systems [9]. Given fragile electrical and industrial systems, an instance of which is the unsustainable US power grid, deep learning offers a pathway to forecast equipment failures and maintenance needs [10][13]. It offers a pathway to better optimize energy consumption and resource management in engineering systems, which can contribute to sustainability efforts and cost savings [11][12]. This is particularly crucial in engineering industries where unplanned downtime can be costly or, in the case of power grids, deadly [14]. Deep learning also has applications to monitor the structural health of buildings and infrastructure, thus enhancing safety and preventing catastrophic failures [15]. Positive examples of deep learning in the context of Natural Language Processing (NLP) are applications like automated documentation, translation, communication, facilitating collaboration among engineers across language barriers, and for development of sematic search engines to understand the context and meaning of user queries [16]. On the other hand, in the past few years concern has arisen[17] that deep learning techniques in natural language processing can destabilize political systems through, for example, chatbots on social networks [22].

In this paper we propose a curriculum for an undergraduate engineering introduction to deep learning. Our goal is to first apply this in a new course taught in the engineering program at Anderson University. The College of Engineering was founded in 2021. At present we offer Electrical, Computer, Mechanical, and General Engineering degrees, and will graduate our first engineering majors in 2025. It might seem natural to expect only computer or electrical engineers to take this proposed course, but we argue that the diverse applications of machine learning make such a course suitable to all engineering majors, so the course should be listed as a

general engineering elective course, and assign *ENGR 370: Machine Learning for Engineers* as a possible course code, suitable for junior or senior students. We center the curriculum around the open-source PyTorch machine learning library for Python as it is used in numerous areas, including image and speech recognition, natural language processing, computer vision, and more. Its user-friendly interface and dynamic nature make it an attractive option for both beginners and experienced practitioners in the field of deep learning [18]. In what follows we will detail in Section 2 explains course development, goals, and outcomes. Given these goals, Section 3 explains the necessary student preparation in order to succeed in an undergraduate general engineering introduction to ML. Section 4 describes the basic components of the course curriculum. Section 5 explains the course assessment and feedback tools to be used, and Section 6 discusses results and implementation.

2. Course Development, Objectives and Outcomes

As we develop our undergraduate *Machine Learning for Engineers* course our next goal is to design learning objectives and outcomes. Learning objectives and outcomes ensure that students acquire knowledge in deep learning concepts and their practical applications in engineering. Figure 1, after Reference [26], shows important elements of course design that we employ in this paper.

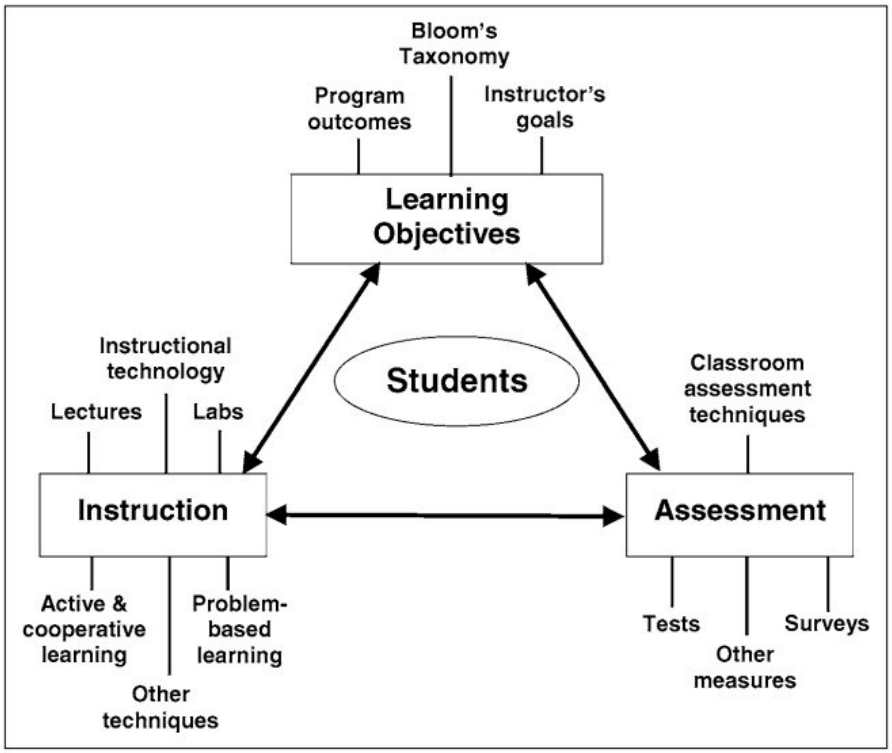

Figure 1. Important elements of course design (from Reference [26])

Reference [26] note that creating a course to achieve specified outcomes requires effort in the domains of "1): planning (identifying course content and defining measurable learning objectives for it); instruction (selecting and implementing the methods that will be used to deliver the specified content and facilitate student achievement of the objectives); and assessment and evaluation (selecting and implementing the methods that will be used to determine whether and how well the objectives have been achieved and interpreting the results."

Since course implementation can be hindered when the course does not align with appropriate accreditation criteria, we therefore seek to create appropriate course objectives and outcomes and align these with specific goals that meet the Accreditation Board for Engineering and Technology (ABET) criteria [19]. Since ABET accreditation criteria were introduced in 1996 they have spurred an intense national reassessment of the engineering curriculum. ABET accreditation and published goals provide a helpful yardstick to ensure that engineering programs meet certain standards [20], and that our new course fits these criteria. Table 1 shows how we propose to align our goals under the umbrella of the ABET criteria.

Table 1. Alignment between ABET criteria and the course learning objectives and outcomes.

ABET Criteria	Specific Goals
1. identify, formulate, and solve complex engineering problems by applying principles of engineering, science, and mathematics	Students acquire knowledge to identify how ML may be useful in technology; they master deep learning concepts and their practical applications in engineering.
2. apply engineering design to produce solutions that meet specified needs with consideration of public health, safety, and welfare, as well as global, cultural, social, environmental, and economic factors	Students analyze engineering problems and apply suitable deep learning models to address them.
3. communicate effectively with a range of audiences	Students develop collaboration and communication skills through group projects and presentations
4. recognize ethical and professional responsibilities in engineering situations and make informed judgments, which must consider the impact of engineering solutions in global, economic, environmental, and societal contexts	Through hands-on projects, students investigate and solve complex engineering problems using PyTorch and explore ethical impacts of the technology
5. function effectively on a team whose members together provide leadership, create a collaborative and inclusive environment, establish goals, plan tasks, and meet objectives.	Students engage in collaborative and professional coding practices and project work, aligning with ABET's emphasis on professionalism

6. develop and conduct appropriate experimentation, analyze and interpret data, and use engineering judgment to draw conclusions	Students master the use of PyTorch, a modern and widely adopted tool in the field of deep learning
7. acquire and apply new knowledge as needed, using appropriate learning strategies.	Students explore advanced topics to stay current with rapidly developing deep learning technologies

In 2024 our College of Engineering established an Advisory Board consisting of representatives from academia, government and industry, and held our first Advisory Board meeting on the university campus. At the meeting we reviewed the undergraduate engineering curriculum in electrical and computer engineering, mechanical engineering, and general engineering in order to learn about emerging needs that the board observes in their areas of expertise. After consultation with these stakeholders, we made the following list, given in Table 2, of learning objectives and outcomes aligned with ABET criteria. These serve as a guide for those who wish to implement a PyTorch rich ML course in their undergraduate curriculum.

Table 2. Course Learning Objectives.

Learning Objective	**Specific Goals**
1. Master Fundamental Deep Learning Concepts	Define and explain key deep learning concepts, including neural networks, activation functions, loss functions, and optimization algorithms.
2. Understand PyTorch Basics	Demonstrate proficiency in using PyTorch, including creating and manipulating tensors, understanding autograd, and working with computation graphs.
3. Build and Train Neural Networks	Develop the ability to design, train, and evaluate neural networks using PyTorch, emphasizing the importance of datasets and dataloaders.
3. Engage in Hands-On Projects	Apply PyTorch in practical projects to solve real-world engineering problems, demonstrating the ability to translate theoretical knowledge into practical applications.
4. Debugging and Troubleshooting in PyTorch	Develop effective debugging and troubleshooting skills for PyTorch models, showcasing the ability to identify and rectify common errors.
5. Explore Model Deployment	Understand the basics of deploying PyTorch models, considering deployment options, considerations, and demonstrating the deployment process.
6. Collaboration and Code Management	Work collaboratively on projects using version control systems like Git, emphasizing the importance of clean and modular code.

Upon completion of the PyTorch course, students should be able to demonstrate particular learning outcomes. Table 3 shows a list of possible outcomes which we have created that are in alignment with ABET standards.

Table 3. Course Learning Outcomes.

Learning Outcomes	Specific Goals
1. Demonstrate Proficiency in PyTorch	Independently use PyTorch to implement and analyze various deep learning models.
2. Apply Deep Learning to Engineering Problems	Apply deep learning techniques to solve engineering problems; demonstrate the ability to choose appropriate models for specific tasks.
3. Evaluate Model Performance	Evaluate the performance of deep learning models using relevant metrics and make informed decisions based on the results.
4. Collaborate Effectively	Collaborate with peers on coding projects, demonstrating effective communication, dissemination of code, and teamwork.
5. Demonstrate Critical Thinking in Model Design	Critically analyze engineering problems, choosing and justifying appropriate deep learning approaches for solving them.

3. Student Preparation for Machine Learning

Before our undergraduate engineering students can effectively master machine learning (ML) via PyTorch, it is essential for them to have a solid foundation in certain prerequisites. To ensure adequate preparation only students who have taken an introductory Python course, and completed mathematics courses up to Calculus IV, may enroll to take what we will call *ENGR 370: Machine Learning for Engineers*.

The key prerequisite skill that students should initially possess is proficiency in Python programming. At present this introductory Python course is taught in the second semester of the engineering program. Since students will already have had one intensive Python introductory course, they will have mastered syntax and string manipulation, data types and structures (lists, dictionaries, etc.), functions, and control structures, and the mathematics fundamentals necessary to understand ML concepts [23]. They will already be familiar with Python libraries like NumPy and Matplotlib, which are often used alongside PyTorch for numerical computations and data visualization [24], along with a basic understanding of linear algebra, calculus, and statistics. These courses will have prepared students for concepts like matrix multiplication, derivatives, and probability which are frequently used in deep learning and PyTorch operations. In addition, at the start of the course students will be introduced to fundamental PyTorch concepts such as tensors, and autograd (automatic differentiation) since it is essential that they should understand how to create and manipulate tensors.

The start of the course will introduce core principles of deep learning, such as supervised learning, unsupervised learning, classification, regression, and the basics of model training and evaluation [25]. Students will also need to develop a basic knowledge of neural networks, including the structure of computational neurons, layers, activation functions, and the concept of feedforward propagation. Although proficiency in data handling is crucial in ML, we will not take much time dealing with data preprocessing and subsequent data processing. Instead, we will simplify the learning flow by providing prepared datasets and simple, directed introduction to using and manipulating these data, an example of which will be demonstrated below.

In order to utilize PyTorch students will need to refresh basic command-line skills, and it will be helpful to create a familiarity with version control systems like Git, which is also valuable for collaborative coding projects that are common for engineering applications.

Once these various prerequisites are attained, our undergraduate engineering students will have a solid foundation to enable them to dive into PyTorch effectively and make the most out of their learning experience in deep learning and neural networks.

4. Curriculum Structure

By following the engineering criteria developed and listed above we can create a unifying framework for the development of the undergraduate *Machine Learning for Engineers* course and curriculum. This allows faculty who teach the course to have a coherent curriculum in which harmony is achieved for all aspects of course goals and objectives, design, syllabus, as well as methods of teaching and assessment [28][29]. We have identified five distinct concept areas, which can be taught as course modules.

4.1 Introduction to Deep Learning Concepts

At the very beginning of the course it is necessary, so to speak, to set the table on behalf of the students since they will have heard about ML, and even used it as a technology, but without an understanding of the limitations and concepts crucial to understanding ML. Thus, we will begin with definitions and key characteristics of ML and deep learning. In the era of intense student use of ChatGPT it is easy to see how students may think they understand so-called artificial intelligence, so it should be important to illustrate strengths, weaknesses, and limitations of what has been referred to as fake or ‘Potemkin AI’ [31].

Students will learn about the biological analogy of neurons with so-called artificial neurons, structured in neural network nodes and layers on a computational device. They will have an introduction to mathematical representations of feedforward processes, and the weights and biases which comprise such neural networks.

Following this relatively simple mathematical introduction, students will learn about staples of ML, including the backpropagation algorithm and the related mathematical details of chain rule and gradients, which are leveraged to explain ML gradient descent for weight optimization in the

neural network. They will review some basic optimization processes which they already covered in pre-requisite courses, for example least-squares, and learn about other popular ML loss functions and how to choose the appropriate loss function in a given task environment.

Next, we introduce the basics of training and evaluation, including the concepts of splitting data into training and testing sets, model training, and the various problems that can arise, such as model overfitting and underfitting [25].

4.2 Basics of PyTorch

With this strong foundation laid, in Module 2 it is time to introduce students to PyTorch, and to do so by first showing examples of its use in industry and society, and its origin as a leading open-source deep learning framework. We can compare PyTorch to other leading frameworks, such as TensorFlow, and simpler ones like Scikit-Learn.

Once preliminaries are dispensed with we turn to learning about the important idea of tensors. We will demonstrate basic tensor operations like tensor creation, manipulation, slicing, and how to convert between PyTorch tensors and the well-known NumPy arrays. Prior to deploying models we will add on to mathematics with which students are already familiar by introducing automatic differentiation, and the PyTorch `autograd` module for computing gradients. Numerous demonstrations of simple examples will showcase the automatic gradient computation feature.

In PyTorch, computation graphs are a fundamental concept used to represent the flow of data and operations within a neural network or any other computational process. They can take two forms in PyTorch: static and dynamic, which will be demonstrated to students.

At this point students will be ready to train a first simple model by building a basic neural network in PyTorch using the 'torch.nn' module. We give students the opportunity by a step-by-step approach to train the model with real-world time-series data, and using PyTorch's optimization algorithms (e.g., SGD) for training the model. Data will come from several areas of departmental research, including chronobiology [33], mechanical jitter, space engineering [34] [35], earthquakes [36], and signal processing [37]. Throughout this Module students will perform various data processing tasks such as splitting data into training, validation, and test sets, evaluating model performance on the validation and test sets, and exploring overfitting and underfitting issues as they employ common loss functions available in PyTorch (e.g., `nn.CrossEntropyLoss`, `nn.MSELoss`). We will explore how to select appropriate loss functions for different tasks.

In the previous required Python course (coded as *ENGR 130* at Anderson University) students have performed extensive coding with NumPy and Matplotlib. Utilizing this knowledge is important for visualization of results of training progress and model outputs, and to interpret the visualizations to gain insights into model behavior.

As they engage with PyTorch there will be ample opportunities for debugging and so at this juncture we will introduce common debugging techniques for PyTorch code.

4.3 Different Types of Neural Networks
Each week during the course includes lab exercises and this is how we will interact with different types of neural networks. Students will first learn about convolutional neural networks (CNNs) and then recurrent neural networks (RNNs), and will explore simple examples through guided coding exercises. They will have assignments to build and train simple neural networks, and lecturers will practice a structured approach to teaching the creation, training, and evaluation of basic neural networks.

4.4 Collaboration and Code Management
Since working in teams requires careful management we will discuss strategies collaboration using version control systems and maintaining clean code, with tools like Git.

4.5 Ethical Considerations
Anderson University is a Christian educational institution deeply concerned with the ethical consideration of technology. Because we have a vested interest in considering the impacts of ML on individuals, societies, and the world at large.

ML Models can perpetuate and exacerbate biases or preferential treatments, based on the data they are trained on. Therefore we will need to discuss and explore algorithmic decision-making and how bias migrates in data and in code. Given the vast volume of data that is available we will also need to discuss privacy and data protection concerns. Questions will be raised in a group setting to explore accountability and responsibility for the decisions suggested by ML systems.

5. Course Assessment and Feedback

Course assessment is essential so that students and other stakeholders have the opportunity to measure their progress with the learning expectations. To assess students' achievement of the learning goals and outcomes in the PyTorch course, a variety of assessments can be implemented. These assessments should cover theoretical understanding, practical application, and collaboration skills [26][29]. Table 4 lists the course assessments.

Table 4. Course assessments.

Assessment type	Description
1. Formative and Foundational	• In-class active Quizzes on ML Concepts: Assess understanding of fundamental deep learning concepts covered in lectures. • Exam on PyTorch Basics: Test knowledge of PyTorch basics, including tensors, autograd, and computation graphs through two written and computational tests.
2. Practical Application	• Neural Network Implementation Assignment: Students must correctly implement and train a basic neural network using

	PyTorch to demonstrate their understanding of model building and training. • CNN Image Classification Project: This will evaluate students' ability to apply Convolutional Neural Networks for image classification tasks.
3. Project-Based	• Hands-On Projects: We will assign real-world engineering projects that require the application of PyTorch. We will evaluate the quality of their solutions, the appropriateness of the chosen model, and the ability to analyze results. • Temporal Convolutional Neural Network (TCN) Project: Task students with implementing a TCN for a temporal sequence problem, assessing their understanding of advanced concepts.
4. Debugging and Troubleshooting	• Debugging Challenge: During in-class laboratory exercieses we will present students with a faulty PyTorch code and assess their ability to identify and fix errors, and allow them showcase their debugging skills.
5. Model Evaluation and Interpretation	• Model Evaluation Report: This will require students to evaluate the performance of a trained model, discuss metrics such as accuracy, precision, recall, and F1 score. Emphasize the interpretation of results.
6. Collaboration and Code Management	• Group Project Assessment: Students will be assigned a group projects; evaluation will be based on collaboration, effective use of version control (e.g., Git), and the quality of modular and well-documented code. Students will also perform group peer-evaluations to document their perceptions of professionalism and teamwork. • Code Review Assignment: Students will review and provide feedback on their peers' PyTorch code, assessing their ability to analyze and improve code quality.
7. Presentation and Communication	• Final Project Presentation: This is attached to the Group Project Assessment (#6 above). It will require students to present their final projects in a written and oral format, in which they explain their approach, results, and insights. This will evaluate their ability to communicate technical concepts effectively.
8. Exams	• Comprehensive Final Exam: This will test overall knowledge and understanding of the entire PyTorch course content, covering both theoretical and practical aspects.
9. Self-Assessment and Reflection	• Learning Journal: Throughout the course students will maintain a learning journal in which they reflect on their progress, challenges, and areas for improvement. This encourages self-assessment and continuous learning.
10. Problem-Solving Challenges	• Model Optimization Challenge: During several labs students will receive suboptimal PyTorch model code and will work in groups to optimize it, assessing their problem-solving skills.

11. Real-World Application Report	• Engineering Application Report: Students are tasked with identifying a specific engineering problem amenable to ML application. They will write a report on how they would apply PyTorch to solve this specific engineering problem and emphasize practical applications.

Although assessments are usually done for the benefit primarily of academic institutions and accrediting agencies, we will also provide constructive feedback to our undergraduate students throughout the course, via required open office-hours, in order to give them specific information about their performance, and so to help them understand what they did well and where they can improve. This will help develop the continuous improvement and future learning ethos that we strive to inculcate in engineers. As we offer positive and constructive feedback this should motivate students as the efforts are recognized, and this will help build their confidence and positive attitude towards learning. It will also help students to identify weaknesses and areas that they can target for improvement. As they reflect on this feedback, they will gain insights into their personal learning process and style, learn to ask questions that lead to a deeper understanding of ML, and show them areas where they can implement continuous improvement.

Requiring students, as groups and individuals, to attend open office hours not only provides feedback to them but also fosters a supportive learning environment where students feel that their efforts are valued, and they are supported in their academic journey.

6. Results and Discussion

Universities face challenges in demonstrating to key stakeholders, such as students, accreditors, employers, and government, that their engineering programs equip graduates with the required level of computational skills requisite to excel in a dynamic technological landscape. To that end we have developed a course to enable undergraduate engineering students to effectively master machine learning (ML) via a widely-used, modern open-source framework like PyTorch. Other educators may consider different tools. We call this course *ENGR 370: Machine Learning for Engineers*. Our study addresses the design and delivery of this course for all engineering majors and considered not only learning requirements, objectives, and assessments but also the concerns of various stakeholders for validating technological innovation.

We focused on the use of PyTorch since it is rapidly becoming important in numerous areas of practice [38]. Although it is a high-performance framework capable of advanced and complex uses, it is also sufficiently accessible and flexible via its modules [40] so that it advanced undergraduates (juniors or seniors in engineering) can learn fungible ML skills.

In Table 6, in the appendix, we provide a model 16-week course schedule. The schedule aims to cover a comprehensive range of topics, providing students with both theoretical knowledge and practical skills. Adjustments can be made based on the progress of the class, the level of engagement, and emerging trends in deep learning. The inclusion of guest lectures, case studies, and hands-on labs enhances the overall learning experience.

In summary, the course will equip our engineering students with a comprehensive skill set in deep learning using PyTorch. We emphasize practical applications, collaboration, and adherence to best practices. The combination of hands-on projects, collaborative learning, and a focus on real-world applications should prove effective in fostering a deep understanding of the subject matter. This will prepare students to apply their knowledge in both academic and industry settings, with a strong foundation for continuous learning in the rapidly evolving field of deep learning. It is likely that the integration of deep learning courses into undergraduate engineering programs will be a growing trend in response to the increasing importance of AI and ML in diverse engineering disciplines.

Acknowledgment

This research was partially funded by National Science Foundation Award AGS-2414513 and the NIH R-25 Program.

References

[1] J. Clerk Maxwell, “On governors,” Proceedings of the Royal Society of London, Vol. 16 (1867-1868), pp. 270-283.

[2] Mayr, O., 1971. “Victorian physicists and speed regulation: An encounter between science and technology.” Notes and records of the Royal Society of London, 26(2), pp.205-228.

[3] Edwards, P. A.; McKay, B. J.; Sink, C. W. First year chemistry laboratory calculations on a spreadsheet. J. Chem. Educ. 1992, 69, 648−650.

[4] Bell, P. C. Teaching Business Statistics with Microsoft Excel. INFORMS Trans. Ed. 2000, 1, 18−26.

[5] Jan, Bilal, Haleem Farman, Murad Khan, Muhammad Imran, Ihtesham Ul Islam, Awais Ahmad, Shaukat Ali, and Gwanggil Jeon. "Deep learning in big data analytics: a comparative study." Computers & Electrical Engineering 75 (2019): 275-287.

[6] Arpteg, Anders, Björn Brinne, Luka Crnkovic-Friis, and Jan Bosch. "Software engineering challenges of deep learning." In 2018 44th euromicro conference on software engineering and advanced applications (SEAA), pp. 50-59. IEEE, 2018.

[7] Kouhalvandi, Lida, Osman Ceylan, and Serdar Ozoguz. "Automated deep neural learning-based optimization for high performance high power amplifier designs." IEEE Transactions on Circuits and Systems I: Regular Papers 67, no. 12 (2020): 4420-4433.

[8] Regenwetter, Lyle, Amin Heyrani Nobari, and Faez Ahmed. "Deep generative models in engineering design: A review." Journal of Mechanical Design 144, no. 7 (2022): 071704.

[9] Patel, Priyanka, and Amit Thakkar. "The upsurge of deep learning for computer vision applications." International Journal of Electrical and Computer Engineering 10, no. 1 (2020): 538.

[10] Khodayar, Mahdi, Guangyi Liu, Jianhui Wang, and Mohammad E. Khodayar. "Deep learning in power systems research: A review." CSEE Journal of Power and Energy Systems 7, no. 2 (2020): 209-220.

[11] Runze, W. U., Bao Zhengrui, Song Xueying, and D. E. N. G. Wei. "Research on short-term load forecasting method of power grid based on deep learning." Modern electric power 35, no. 2 (2018): 43-48.

[12] Shuvro, Rezoan A., Pankaz Das, Majeed M. Hayat, and Mitun Talukder. "Predicting cascading failures in power grids using machine learning algorithms." In 2019 North American Power Symposium (NAPS), pp. 1-6. IEEE, 2019.

[13] Wanliss, J. A., and L. A. G. Antoine. "Geomagnetic micropulsations: Implications for high resolution aeromagnetic surveys." Exploration Geophysics 26, no. 4 (1995): 535-538. https://doi.org/10.1071/EG995535

[14] Busby, Joshua W., Kyri Baker, Morgan D. Bazilian, Alex Q. Gilbert, Emily Grubert, Varun Rai, Joshua D. Rhodes, Sarang Shidore, Caitlin A. Smith, and Michael E. Webber. "Cascading risks: Understanding the 2021 winter blackout in Texas." Energy Research & Social Science 77 (2021): 102106.

[15] Yuan, Fuh-Gwo, Sakib Ashraf Zargar, Qiuyi Chen, and Shaohan Wang. "Machine learning for structural health monitoring: challenges and opportunities." Sensors and smart structures technologies for civil, mechanical, and aerospace systems 2020 11379 (2020): 1137903.

[16] Zhang, Wei Emma, Quan Z. Sheng, Ahoud Alhazmi, and Chenliang Li. "Adversarial attacks on deep-learning models in natural language processing: A survey." ACM Transactions on Intelligent Systems and Technology (TIST) 11, no. 3 (2020): 1-41.

[17] Ibrishimova, Marina Danchovsky, and Kin Fun Li. "A machine learning approach to fake news detection using knowledge verification and natural language processing." In Advances in Intelligent Networking and Collaborative Systems: The 11th International Conference on Intelligent Networking and Collaborative Systems (INCoS-2019), pp. 223-234. Springer International Publishing, 2020.

[18] Doleck, Tenzin, David John Lemay, Ram B. Basnet, and Paul Bazelais. "Predictive analytics in education: a comparison of deep learning frameworks." Education and Information Technologies 25 (2020): 1951-1963.

[19] Alarifi, Ibrahim M. "Comparative Analysis on Regional (NCAAA) and International (ABET) Accreditation for Mechanical Engineering Program." Eng. Technol. Open Access J 3 (2021): 119-134.

[20] Alhorani, Rana AM, Wejdan Abu Elhaija, Subhi M. Bazlamit, and Hesham S. Ahmad. "ABET accreditation requirements and preparation: Lessons learned from a case study of Civil Engineering Program." Cogent Engineering 8, no. 1 (2021): 1995109.

[21] Sarker, Iqbal H. "Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions." SN Computer Science 2, no. 6 (2021): 420.

[22] Pashentsev, Evgeny. "The malicious use of artificial intelligence through agenda setting: Challenges to political stability." In Proceedings of the 3rd European Conference on the Impact of Artificial Intelligence and Robotics ECIAIR, pp. 138-144. 2021.

[23] Prokopiev, Mikhail Semenovich, Elena Zotikovna Vlasova, Tatiana N. Tretiakova, Maxim Anatolyevich Sorochinsky, and Rimma Alekseevna Soloveva. "Development of a

programming course for students of a teacher training higher education institution using the programming language Python." Propositos y representaciones 8, no. 3 (2020): 33.

[24] Ranjani, J., A. Sheela, and K. Pandi Meena. "Combination of NumPy, SciPy and Matplotlib/Pylab-a good alternative methodology to MATLAB-A Comparative analysis." In 2019 1st international conference on innovations in information and communication technology (ICIICT), pp. 1-5. IEEE, 2019.

[25] Raschka, Sebastian, Joshua Patterson, and Corey Nolet. "Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence." Information 11, no. 4 (2020): 193.

[26] Felder, Richard M., and Rebecca Brent. "Designing and teaching courses to satisfy the ABET engineering criteria." Journal of Engineering education 92, no. 1 (2003): 7-25.

[27] Anwar, Arif A., and David J. Richards. "A comparison of EC and ABET accreditation criteria." Journal of Professional Issues in Engineering Education and Practice (2018).

[28] Subic, Aleksandar, and Don Maconachie. "Strategic curriculum design: An engineering case study." European journal of engineering education 22, no. 1 (1997): 19-33.

[29] Boev, Oleg V., Norbert Gruenwald, and Guenter Heitmann. Engineering curriculum design aligned with accreditation standards. Scholars' Press, 2014.

[30] Qadir, Junaid. "Engineering education in the era of ChatGPT: Promise and pitfalls of generative AI for education." In 2023 IEEE Global Engineering Education Conference (EDUCON), pp. 1-9. IEEE, 2023.

[31] Sadowski J (2018) Potemkin AI: many instances of 'artifcial intelligence' are artificial displays of its power and potential. https://reallifemag.com/potemkin-ai; accessed 5 Feb. 2024

[32] Hu, Han, and Connor Heo. "Integration of Data Science Into Thermal-Fluids Engineering Education." In ASME International Mechanical Engineering Congress and Exposition, vol. 86694, p. V007T09A023. American Society of Mechanical Engineers, 2022.

[33] Wanliss J, Cornélissen G, Halberg F, Brown D, Washington B (2018) Superposed epoch analysis of physiological fluctuations: possible space weather connections. Int J Biometeorol 62:449–457. https://doi.org/10.1007/s00484-017-1453-7

[34] Cersosimo, D.O., Wanliss, J.A. "Initial studies of high latitude magnetic field data during different magnetospheric conditions." Earth Planet Sp 59, 39–43 (2007). https://doi.org/10.1186/BF03352020

[35] Wanliss, J. A., K. Shiokawa, and K. Yumoto. "Latitudinal variation of stochastic properties of the geomagnetic field." Nonlinear Processes in Geophysics 21, no. 2 (2014): 347-356. https://doi.org/10.5194/npg-21-347-2014

[36] Wanliss, James, Víctor Muñoz, Denisse Pastén, Benjamín Toledo, and Juan Alejandro Valdivia. "Critical behavior in earthquake energy dissipation." The European Physical Journal B 90 (2017): 1-8. https://doi.org/10.1140/epjb/e2017-70657-y

[37] Wanliss, J. A., and Grace E. Wanliss. "Efficient calculation of fractal properties via the Higuchi method." Nonlinear Dynamics 109, no. 4 (2022): 2893-2904. https://doi.org/10.1007/s11071-022-07353-2

[38] Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen et al. "Pytorch: An imperative style, high-performance deep learning library." Advances in neural information processing systems 32 (2019).

[39] Ketkar, Nikhil, Jojo Moolayil, Nikhil Ketkar, and Jojo Moolayil. "Introduction to pytorch." Deep Learning with Python: Learn Best Practices of Deep Learning Models with PyTorch (2021): 27-91.

[40] Chaudhary, Anmol, Kuldeep Singh Chouhan, Jyoti Gajrani, and Bhavna Sharma. "Deep learning with PyTorch." In Machine learning and deep learning in real-time applications, pp. 61-95. IGI Global, 2020.

Appendix

Table 5 shows a proposed 16-week schedule for the deep learning with PyTorch course, *ENGR 370: Machine Learning for Engineers*. Our study addresses the design and delivery of this course. The course will be 4 credit hours (CR). Each week consists of three lectures and one laboratory session. Please note that the schedule is flexible and can be adjusted based on the pace and needs of the students.

Table 5. Proposed course schedule.

Week	Specifications
Week 1: Introduction to Mathematical Tools for Deep Learning	

Week 1	• Lecture 1: Linear Algebra for Deep Learning
	  o Basic concepts such as vectors, matrices, operations
	  o Linear algebra
	• Lecture 2: Calculus for Deep Learning
	  o Derivatives, gradients, optimization
	• Lab 1: Practice
	  o Python exercises and problem solving related to linear algebra and calculus
	• Assignment
	  o Problem set to reinforce understanding of mathematical concepts

Week 2: Basics of Neural Networks	

Week 2	• Lecture 3: Overview of Neural Networks
	  o History and important applications
	  o Structure and functioning of artificial neurons
	  o Introduction to perceptrons and feedforward networks
	  o Activation functions and their role in neural networks
	• Lecture 4: Backpropagation and Training Neural Networks
	  o Basics of backpropagation algorithm
	  o Training process and loss functions
	• Lab 2: Neural Networks from Scratch

	○ Hands-on exercises on building and training basic neural networks using Python and numpy • Assignment ○ Implementing backpropagation algorithm for training a simple neural network
Week 3: Introduction to PyTorch and Tensors	
Week 3	• Lecture 5: Introduction to PyTorch Framework ○ Overview of PyTorch features and advantages • Lecture 6: Tensors and Operations in PyTorch ○ Generalizing vectors and matrices ○ Tensors and basic tensor operations • Lab 3: Working with Tensors in PyTorch ○ Practical exercises on creating and manipulating tensors • Assignment ○ Implementing tensor operations and visualizations in PyTorch
Week 4: Building Neural Networks with PyTorch	
Week 4	• Lecture 7: Building Blocks of Neural Networks ○ Understanding layers • Lecture 8: Further Building Blocks ○ Activation functions and loss functions • Lab 4: Building Neural Networks in PyTorch ○ Practical exercises on building and training simple neural networks • Assignment ○ Train a neural network on a real dataset and evaluate performance
Week 5: Training Neural Networks	
Week 5	• Lecture 9: How to Train your Neural Network ○ Overview ○ Forward and backward propagation • Lecture 10: More Training ○ Forward and backward propagation ○ Optimization algorithms • Lab 5: Training Neural Networks in PyTorch ○ Guided, hands-on training of neural networks using PyTorch • Assignment ○ Fine-tuning a pre-trained CNN model on a custom dataset
Week 6: Introduction to Convolutional Neural Networks (CNNs)	

Week 6	• Lecture 11: Basics of Convolutional Neural Networks o Overview and examples • Lecture 12: CNNs (II) o Convolutional layers, pooling layers, and their applications. • Lab 6: Implementing CNNs in PyTorch o Practical exercises on building and training CNNs for image classification tasks • Assignment o Implement a CNN architecture for image classification using PyTorch.
Week 7: CNN Architectures (III)	
Week 7	• Lecture 13: Advanced CNN Architectures o Understanding principles of CNN architecture design • Lecture 14: Transfer Learning with Pre-trained Models o How to fine-tune pre-trained CNN models for specific tasks • Lab 7: Transfer Learning with Pre-trained Models o Hands-on exercises on fine-tuning pre-trained CNN models for specific tasks • Assignment o Implement transfer learning on a pre-trained CNN model for a custom task
Week 8: Recurrent Neural Networks (RNNs)	
Week 8	• Lecture 15: Basics of Recurrent Neural Networks o Overview o Understanding RNN architecture • Lecture 16: RNNs and Sequence Modeling o Applications in sequence modeling • Lab 8: Implementing RNNs in PyTorch o Guided coding exercises on building and training RNNs for sequence prediction tasks • Assignment o Implement an RNN model for sequence prediction using PyTorch
Week 9: Long Short-Term Memory (LSTM) Networks	
Week 9	• Lecture 17: Introduction to LSTM Networks o Understanding the architecture and advantages of LSTM networks • Lecture 18: Implementing LSTM Networks o Details of operation of LSTM networks • Lab 9: n PyTorch

	○ Hands-on exercises on building and training LSTM models for sequence modeling • Assignment ○ Implement an LSTM model for time series prediction using PyTorch
Week 10: Model Evaluation and Validation	
Week 10	• Lecture 19: Model Evaluation Metrics ○ Overview of evaluation metrics ○ Understanding metrics like accuracy, precision, recall, and F1-score • Lecture 20: Use of metrics ○ Demonstrating use of metrics in models already used in the course • Lab 10: Model Evaluation in PyTorch ○ Practical exercises on evaluating and validating deep learning models using PyTorch. • Assignment ○ Evaluate the performance of trained models on different datasets
Week 11: Hyperparameter Tuning and Optimization Techniques	
Week 11	• Lecture 21: What are Hyperparameters ○ Overview • Lecture 22: Hyperparameter Tuning ○ Techniques for optimizing model performance through hyperparameter tuning ○ Introduction to NNI • Lab 11: Hyperparameter Tuning in PyTorch ○ Hands-on exercises on optimizing model hyperparameters using PyTorch. • Assignment ○ Optimizing the performance of a deep learning model through hyperparameter tuning
Week 12: Introduction to Generative Adversarial Networks (GANs)	
Week 12	• Lecture 23: Basics of Generative Adversarial Networks ○ Overview and History ○ Examples and use cases • Lecture 24: GAN Architecture ○ Understanding the concept and architecture of GANs • Lab 12: Implementing GANs in PyTorch ○ Hands-on exercises on building and training GAN models for generating synthetic data

	• Assignment o Implement a GAN model for generating synthetic images using PyTorch
Week 13: Model Deployment and Real-World Applications	
Week 13	• Lecture 25: Model Deployment Strategies o Understanding different deployment options for deploying deep learning models in real-world applications • Lab 13: Model Deployment in PyTorch o Practical exercises on sharing and deploying trained models using Git, and PyTorch Serve or Flask tensors • Assignment o Deploy a trained deep learning model as a web service using PyTorch
Week 14: Ethics and Bias in Deep Learning	
Week 14	• Lecture 25: Ethical Considerations in Deep Learning o Anderson University: The Christian Context o Exploring the ethical implications and biases associated with deep learning models • Lecture 26: Ethical Considerations (II) o Exploring the ethical implications and biases associated with deep learning models o Guest lecture and group discussions • Lab 15: Addressing Bias in Deep Learning Models o Hands-on exercises on identifying and mitigating bias in deep learning models • Assignment o Analyze and mitigate bias in a trained deep learning model on the basis of a Christian moral framework
Week 15: Advanced Topics and Project Work	
Week 15	• Lecture 26: Advanced Deep Learning Topic o Overview of advanced topics such as attention mechanisms, reinforcement learning, and self-supervised learning • Lab 15: Project Work and Consultation o tensors • Assignment o Dedicated time for students to work on their final projects with guidance from instructors
Week 16: Final Project Presentations and Course Conclusion	

Week 16	• Final Project Presentations: Students present their final projects to the class o Oral and written submissions in groups o Group and individual student assessments and feedback • Course Conclusion o Reflection on the course content, discussion on future directions in deep learning, and course evaluation