An Approach to Teaching Applied Machine Learning with Autonomous Systems Integration

Chad Mello, Adrian De Freitas and Troy Weingart
Department of Computer & Cyber Sciences, United States Air Force, Colorado Springs, CO 80840, U.S.A.

Keywords: Unmanned Aerial Vehicles, UAV, Autonomous Systems, Artificial Intelligence, AI, Machine Learning, ML, Computer Science Education, Computing Education, Drones, Multicopters, Sensors, Computer Vision.

Abstract: We propose an applied machine learning course that teaches students with no machine learning background how to train and use deep learning models for deploying aerial drones (multi-copters). Our unique, hands-on curriculum gives students insight into the algorithms that power autonomous systems as well as the hardware technology on which they execute. Students learn how to integrate Python code with serial communications for streaming sensors and imagery to deep learning models. Students use OpenCV, Keras, and TensorFlow to learn about computer vision and deep learning. The final project (see Figure 1) provides the opportunity for students to plan and develop an end-to-end, fully autonomous, self-contained product (i.e. all systems physically residing on the drone itself) that is integrated with heavy-payload drones and computer vision in a scenario centered around an outdoor search and rescue mission. With no human in the loop, students deploy drones in search of a missing person. The drone locates and identifies the individual, delivers a care package to their location, and then reports the individual's geolocation to ground rescuers before returning home. The novel helper code and solutions are built in-house using Python and open technologies. Results from a pilot offering in the spring of 2021 indicate that our approach is effective and engaging for computer and cyber science students who have previously taken a basic artificial intelligence course and who have 1-2 years of programming experience. This paper details the design, focus, and methodology behind our *Autonomous Systems Integration* curriculum as well as the challenges we encountered during its debut.

1 INTRODUCTION

For all the good internships may do for undergraduate students' preparedness for industry, merely 60% of undergraduate students in the United States participate in some form of internship program (Smith and Green, 2021). A recent study with 536 multi-institutional computer science (CS) students confirmed this trend, showing that only 57.5% of them completed an internship prior to graduation (Kapoor and Gardner-McCune, 2020). Many students rely on their college degrees for job prospects after graduation (Valstar et al., 2020), and while there is little difference in academic performance between students who intern and those who do not, those graduating without the industry experience an internship affords often find themselves less employable, because they lack practical experience, good technical/interpersonal skills, and the ability to work effectively in teams (Kapoor and Gardner-McCune, 2020).

We set out to address some of these issues for computer science (CS) undergraduate majors by designing a hands-on machine learning (ML) curriculum aimed at improving students' understanding and application of machine learning, teamwork, and newly acquired skills. Specifically, we focus on skills and insight that are considered relevant and valuable to both pubic and private industry: artificial intelligence (AI), ML, simulation, computer vision, and large, heavy-payload, unmanned aerial vehicles (UAVs). Work on large, complex projects is never done in a vacuum; therefore, we felt it was important to integrate teamwork directly into the curriculum. Our course, *Computer Science CS 472: Autonomous Systems Integration* was designed to give students hands-on experiences with these state-of-the-art technologies.

The curriculum offers challenges wherein third and fourth-year CS undergraduates can succeed by leveraging and building upon their current skills and knowledge. The chosen prerequisites for CS 472 are basic programming skills with Python, exposure

Mello, C., De Freitas, A. and Weingart, T.
An Approach to Teaching Applied Machine Learning with Autonomous Systems Integration.
DOI: 10.5220/0011092700003182
In *Proceedings of the 14th International Conference on Computer Supported Education (CSEDU 2022) - Volume 2*, pages 204-212
ISBN: 978-989-758-562-3; ISSN: 2184-5026
Copyright © 2022 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

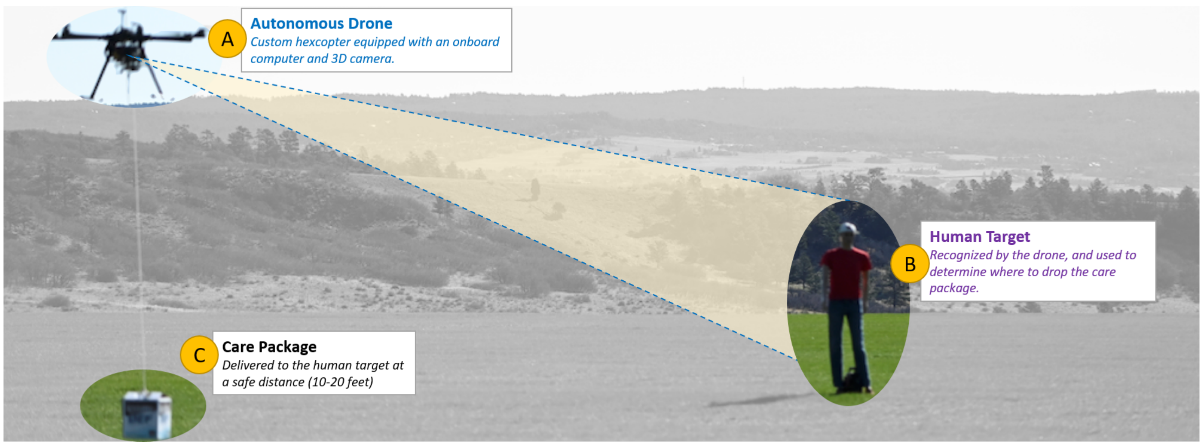

A
Autonomous Drone
Custom hexcopter equipped with an onboard computer and 3D camera.
B
Human Target
Recognized by the drone, and used to determine where to drop the care package.
C
Care Package
Delivered to the human target at a safe distance (10-20 feet)

Figure 1: A summary of CS 472's final project. This image shows an actual deployment in action; drone is currently delivering a package to the ground. Students deploy a fully autonomous drone (a) that uses machine learning to recognize a stranded human from the air and calculate a safe ground distance from the individual where a much-needed care package is to be delivered (b); the payload is then safely lowered to the ground and detached from the drone (c).

to Linux, intermediate knowledge and understanding of AI (CS 471), and a grasp of basic, college-level math. To make the course accessible to computer science students, exercises focus on the algorithms and ML models that control the UAV while also exposing students to the hardware stack and sensor streaming when required. An advanced understanding of Linear Algebra (*e.g.,* vectors and matrices) isn't required; we cover what they need during the course.

This paper makes three contributions. First, we present a curriculum that helps CS students gain practical skills in drone technologies and applied ML. We detail design considerations encompassing three key areas: core autonomous systems concepts, machine learning, and real-world integration, testing, and deployment. Secondly, we explore the resources and coordination required to implement such a course. Third, we share a retrospective of our experience with teaching CS 472 and challenges we encountered throughout the 2021 Spring semester.

This paper is structured as follows. In the first section, we provide an overview of related works and show the need for a course that combines theoretical ML education and real-world exercises. We then provide an overview of CS 472, describe the hardware/software used, and the lesson layout. Finally, we discuss our experiences piloting CS 472 during the Spring 2021 semester, and conclude with a discussion of lessons learned and areas for improvement.

2 RELATED WORKS

A modern approach to addressing the gap between academia and industry is to couple theory and application, embedding a practical learning experience into the curriculum. This is the cornerstone of vocational approaches to learning, where expertise is acquired through engagement and experience prompted by contextualized meaningfulness (Stevenson, 2020). Internships overlap vocational-oriented programs that incorporate many aspects of *experiential learning* (Kolb et al., 2014) (or vice versa), which may help to promote self efficacy through the transformation of the experience built into such programs (Manolis et al., 2013).

It is generally accepted that ML is currently held in high regard insomuch as the promise it holds in many areas of science, medicine, and engineering (Akbilgic and Davis, 2019; Farjo and Sengupta, 2021; Rutherford, 2020; Trister et al., 2017; Lürig et al., 2021; Toole et al., 2019; Azari et al., 2020; von Lilienfeld and Burke, 2020); therefore, we feel it is important to have a portion of CS curriculum devoted to it. Yet, according to (Shapiro et al., 2018), educational offerings in CS departments do not reflect this reality; in addition, they point out that more research into *how* people learn ML is necessary so as to encourage the creation of ML courses specifically designed to be effective across a broad range of students with varying backgrounds and interests.

It is important to note that our course is designed as an introductory ML course, albeit heavily applied. The curriculum encourages students to "roll up their sleeves" and dive into the material with quick startup lessons and labs. We loosely incorporate ideas from two studies that suggest differing pedagogical approaches to teaching ML that aim to accomplish similar results for non-CS majors, (Sulmont et al., 2019) and (Fiebrink, 2019). (Sulmont et al., 2019) point out that ML algorithms are not as difficult to teach as is using ML in design decisions and comparing contrasting models; insightful information that we took into account when building out the hands-on projects included in the curriculum.

Work by (Fiebrink, 2019) suggests approaches to teaching ML for creative practitioners (*i.e.,*. artists, musicians, etc.) while it draws on substantial data from some of the first courses in the world focused on teaching ML to non-CS majors. (Sulmont et al., 2019) mapped *Structure of Observed Learning Outcomes* (SOLO) stages from the seminal work published by Biggs and Collis (Biggs and Collis, 2014) to an instructor-based taxonomy of ML learning goals for non-CS majors. Our take is that incorporating elements from these works into our design allows for the program to be an effective, robust introductory course to ML for CS majors while allowing for it to be tailored to work outside of a CS major track in the near future.

3 COURSE OVERVIEW

CS 472 is a 3-credit hour course consisting of 40 one-hour classes. Students are expected to spend between 2 and 3 hours outside of class per hour in class on homework, labs, and assignments. The course centers on integrating machine learning with autonomous systems found in open drone tech stacks to provide applied solutions to real-world problems.

The culminating event of the class is a multi-day exercise in which students deploy a fully-autonomous drone in support of a search and rescue mission (see Figure 1). The drone must find and recognize a stranded, possibly injured person from the air (represented as a dressed mannequin for safety reasons) using a camera, infrared depth sensor and trained machine learning model. It must then lower and deliver a care package (payload) within ten feet of that person, report the location of the individual to human rescue crews, and return home. The assignment is accomplished with teams of two students, each team having its own drone. Each team integrates the drone hardware with custom-written, in-house helper libraries and software to enable the functionality necessary to complete the mission. Each team may use any code covered in class as well as any code created by its own individual team members; no cross-team code sharing is permitted. Teams go head-to-head, competing for best all-around execution.

Due to the complexity of the final project, we were unable to find cost effective commercial products that aligned our course goals; therefore, we created our own drone software suite, relying heavily on open hardware and software architectures and technologies whenever available. In this section we outline the hardware and software choices we made, and demonstrate how these choices give students the opportunity to experience a rich environment where implementation and execution is fused with traditional college-level theory.

3.1 Hardware Choices

Budget, safety concerns, and feasibility must all be considered when choosing hardware for a course that includes UAVs. Smaller drones are typically less expensive and even considered more practical under certain constraints (like available air space for flying or storage space). However, we have ample outdoor space for flying large machines, and we have good storage facilities as well; therefore, we chose to purchase seven custom, fully assembled large Tarot X6-based hexacopter platforms from *UAV Systems International* (International, 2019). Figure 2 details the basic drone stance as well as the locations where we retrofitted the assistive computer, camera, and distance sensor onto the drone. A number of reasons went into our decision for purchasing these larger machines:

- We wanted to afford students the opportunity to work with state-of-the-art technology while having all of the necessary embedded equipment onboard the drones. The autopilot, NVIDIA GPUs, 1TB data storage, cameras, distance finders, etc. are 100% onboard. Smaller drones would make this difficult, if not impossible.
- We desired to have all necessary hardware for the AI execute on the drone independently, without the need for a remote computer connection. This significantly reduces latency between commands and execution. Note that instructors have RC transceivers on hand at all times to override drone functions in cases where student code might create a dangerous situation. In addition, a laptop with a wireless radio is used to monitor real-time telemetry while the drones are in flight.
- We wanted to create interesting and engaging challenges for this course. The Tarot X6 is capable of carrying up to 11 pounds of cargo in addition to the onboard batteries. We felt that missions requiring the delivery of a large payload would be appealing and interesting for students.
- As the course matures, we may integrate more objectives that require additional sensors and other equipment. These platforms are capable of scal-

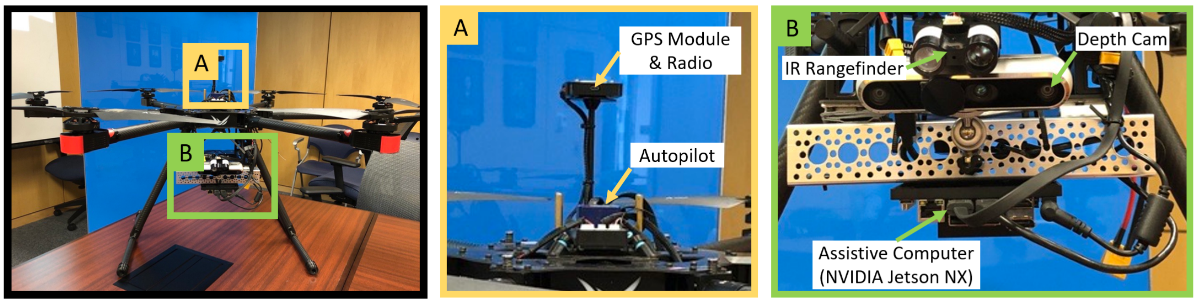


Figure 2: A close-up of the Tarot X6 hexacopters used in CS 472. In addition to the standard radio, autopilot, and GPS systems (A), we attached a depth sensing camera, IR range finder, and an independent assistive computer (**b**). The drone is capable of running machine learning models to classify objects and altering its behavior without commands from a base station.

ing up with the course *via* their larger footprint and payload capacity.

We purchased several long-range *PowerBox System* Radio Core RC Systems (powerbox systems.com, 2022) for managing the drones when outdoors. Our handsets store an individual profile for each of our drones, allowing instructors to override any drone from any handset. In addition, each drone is equipped with an *Intel® RealSense™* D455 depth-sensing camera (intelrealsense.com, 2021), the PX4 (ArduPilot) Cube Blue autopilot (ardupilot.org, 2021), the NVIDIA® *Jetson Xavier™ NX* for accelerated AI execution (nvidia.com, 2022) - 6-core ARM CPU, 384 GPU cores, 8 GB ram, and 1TB SSD. Drones also have onboard depth sensors and payload cable latch systems.

3.2 Software

We relied heavily on cross-platform software when designing CS 472. This was done out of necessity. All of the students at our institution are issued Windows laptops during their Freshman year. Consequently, we wanted to make sure that the software solutions would run on their systems, as well as with the course hardware–much of which utilizes Linux-based operating systems. We designed and produced Python source modules specifically for this course that provide students with all the necessary core code they'll need for several major projects. It contains ML helpers, safe drone interfacing functions, standardized logging, and starter code for major projects.

Throughout the course students utilize several pieces of open source software tools. *Python 3.8* (Sanner et al., 1999) was chosen for the programming language. All coding exercises are performed in Python. PyCharm (JetBrains.com, 2022) was chosen for the Python development environment. PyCharm is a free open-source, robust development environment that offers easy virtual environment creation, debugging and syntax checking. Students utilize two drone-related Python API libraries: (1) Pymavlink and (2) DroneKit. These libraries communicate with drone hardware over serial communications using the *MavLink* protocol. Machine learning and computer vision are facilitated by TensorFlow, Keras, and Open-CV. Finally, we include the PyRealSense Python library for interfacing with Intel's camera system (described in section 3.1).

To configure the drone hardware, upload flight plans to drones, and monitor drones while in flight, students use *Mission Planner* (Oborne, 2016). We also provided students with a simulated environment that they use to test their drone code. The software for that is provided by the *ArduPilot open source project*, referred to as *Software in the Loop* (SITL) (ArduPilot.org, 2016). SITL allows students to execute their code on a simulated drone so that it can be safely tested and debugged before deploying onto the actual devices. To visualize the simulated drone flight, we incorporated an open-source flight simulator to sit on top of SITL, *FlightGear* (Perry, 2004). The SITL environment, as we have it configured for this course, contains everything required for virtualized drone flight testing and experimentation.

4 CURRICULUM DESIGN

CS 472's curriculum is split into two halves. The first half of the course (lessons 1-15) consists mostly of a traditional classroom arrangement. Here, students attend lectures and complete labs covering the use of drones, multi-copter flight theory and mission planning, machine learning, sensors and data streaming, drone hardware calibration, and drone safety. Quizzes, homework, and testing are administered as expected under a traditional college-level setting.

The second half of the course (Lesson 16 onward) is designated for hands-on experimentation and implementation workshops, fused with lectures and gate checks (progress and readiness checks) designed to cover ML in greater detail and depth. The hands-on lessons are designed to help students concretely apply their theoretical knowledge to the physical hardware and environment. To form a complete picture of an end-to-end solution, each major lesson block concludes with a either real-world exercise where students must program the UAV hardware to accomplish a subset of the final mission, or a real-world example that includes building/training ML models to be used in our mission(s).

There are *many* topics wrapped into a course of this type, and any one of these topics alone could consume an entire semester. Consequently, material had to be compressed to fit within our 16-week semester. It was important to consider assimilable material, how much of it would be supplied to the students, and where we should expect students to *"fill in the blanks"*. For example, when designing labs and projects, we considered the amount of starter code that should be supplied and where assignments should focus students' work and attention to ensure optimal learning towards the primary purpose of a given exercise. An important outcome for this course is to produce students who have both knowledge and implementation skills, yet we considered it *unreasonable* to expect students to design, build, discover, and code everything from scratch.

4.1 Programming Exercises

Although CS 472 is not a programming course *per se*, it does require students to write a significant amount of Python code to get their drones working. To ease this burden, we provided two forms of starter code to the students:

- Custom *Python libraries* that contain common functions for communicating with our drones, along with commonly-used commands and processes
- *Skeleton code* that provides a working MAIN function and stubs that students are required to complete

These items provide a key starting point for all projects, allowing students to quickly spin up and focus on the major tasks at hand. In addition, labs include *live coding* in the classroom, where students can follow along with the instructor. This gives us an opportunity to explain the code to the students while giving them functional examples that they can then modify as needed when working on the final project.

4.2 Textbooks

The advantages to using trade books over more traditional textbooks in this context are (1) quality publications provide readers with robust theory to go with content while filtering all but the essentials for good instruction, (2) trade books are often considered more readable, making it possible to approach readily-applicable examples and instruction and (3) students who go on to become professionals are likely to turn to trade books when working in industry (Schultz, 2014; Smolkin et al., 2013). With this in mind, we chose 3 texts for the course:

- Build a Drone: A Step-by-Step Guide to Designing, Constructing, and Flying Your Very Own Drone (Davies, 2016) **required**
  I Sa et al - 2016
- Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow 2nd Edition (Géron, 2019) **required**
  A Géron - 2019
- Machine Learning with Python for Everyone (Fenner, 2019) **supplemental**
  M Fenner - 2019

4.3 Lesson Organization

CS 472's lessons are organized into five blocks. In this section, we outline the structure of each block, the learning outcomes, and the tools used to assess student learning.

4.3.1 Block One: Drones, Multi-copters, and Mission Planning

The first 6 lessons of CS 472 introduce the course curriculum while taking a fresh look at modern drones, their importance, and application in industry and defense. Four lessons focus on multi-copter drones, flight theory surrounding them, and general tools and application.

The key objectives of these introductory lessons are to help students: (1) understand the difference between automated vs. autonomous systems, (2) know the immense potential in applied autonomous systems as well as their limitations, (3) become familiar with the devices that enable autonomous flight & operation, (4) describe the role of Software In the Loop (SITL) and Mission Planner, and (5) understand the technology stack used throughout the course. We use a combination of PowerPoint slides, videos, and hands-on labs to teach the material. Two in-class labs, one quiz, as well as homework activities are used to assess how well these objectives are met.

4.3.2 Block Two: Machine Learning

This block consists of 10 classes centered around ML. The first 3 lessons focus on some general machine learning approaches as well as their uses. The next 4 lessons focus on feed-forward neural networks, convolutional neural networks (CNNs), and computer vision. The final 2 lessons in the block focus on transfer learning.

Through these lessons, students are able to: (1) understand what ML is and what it is used for, (2) understand how ”learning” takes place in ML, (3) gain awareness of inherent challenges of training and using ML, (4) understand deep feed-forward neural networks as well as deep CNNs, (5) gain a rudimentary understanding of how convolutions work, (6) learn the architecture of a basic CNN, (7) obtain hands-on experience using TensorFlow, Keras and Open-CV, (8) learn how to train and evaluate a simple CNN, (9) experiment with computer vision, object recognition and tracking, (10) learn about transfer learning, and (11) gain skills in transfer learning using Keras and TensorFlow.

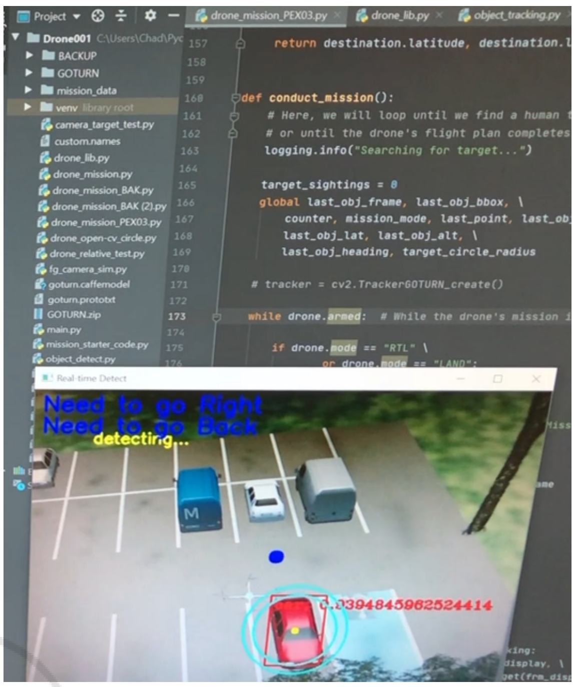
As in block one, we use a combination of PowerPoint slides, videos, and hands-on labs to teach the material. 3 in-class labs, 2 quizzes, as well as homework activities are used to assess how well these objectives are met. The midterm exam is also administered and evaluated at the end of this block.

4.3.3 Block Three: Integrating with Autonomous Systems (AS)

By the third block, students are ready to learn how their AI/ML systems will work with real-time data collected by sensors on the drone. For the next 6 lessons, students complete hands-on lab experiments *via* SITL as well with the actual drone hardware. Students learn how to enable communications between the assistive computer and the autopilot, then they begin uncovering ways to coordinate functionality between them to accomplish tasks.

These lessons give students: (1) a deeper understanding the drone's autopilot, configuration, and sensors, (2) an understanding of how communications work between the autopilot and the assistive NVIDIA computer, (3) time to explore the ethical concerns of using autonomous drones, (4) an overview of drone safety and precautions, (5) manual drone flight training, and (6) experience processing and utilizing drone telemetry and sensor information in preparation for the following block's autonomous target acquisition and landing exercise.

Performance is assessed though in-class labs, completion of manual drone flight training, an ethics paper, and one quiz.



```
    return destination.latitude, destination.l


def conduct_mission():
    # Here, we will loop until we find a human t
    # or until the drone's flight plan completes
    logging.info("Searching for target...")

    target_sightings = 0
    global last_obj_frame, last_obj_bbox, \
        counter, mission_mode, last_point, last_ob
        last_obj_lat, last_obj_alt, \
        last_obj_heading, target_circle_radius

    # tracker = cv2.TrackerGOTURN_create()

    while drone.armed:  # While the drone's mission i

        if drone.mode == "RTL" \
                or drone.mode == "LAND":
```

Figure 3: A student submission for Mission Exercise 1. Students must demonstrate that their code (top) can safely operate the drone in the SITL environment (bottom) before being run on real-world hardware.

4.3.4 Block Four: Mission Exercise 1: Automated Drone Target Acquisition

In the fourth block we transition from lectures and labs to hands-on workshops focused on executing missions. Here, students must program their drones to fly a search pattern and locate a ground target (*e.g.,* an area of terrain with a particular color) using the onboard camera and computer vision algorithms. Once that target is located, the drone will center itself over the target and safely land on top of it.

Key objectives are: (1) learning how to control drone movement via Python, (2) learning how to code, debug & test drone missions in the SITL environment, and (3) to successfully complete the in-field drone mission execution.

To prevent students from trying to complete the assignment in an ”all-nighter” session (and, consequently, to reduce the risk of the drones being damaged), we divided the assignment into two ”progress checks”, which were separated by a week. In the first progress check, students are evaluated on how well their code initially runs in the SITL environment (Figure 3). The second progress check occurs the day before the real-world flight, and tests all aspects of the mission (*i.e.,*target acquisition, navigation, and landing); if a team's code fails the SITL test, it will not be executed on the actual drone, and students will receive *at best* a 70% for the exercise. The final step is executing the mission outdoors where the target is placed in a random location on the ground in a very large, outdoor field. A flight plan unknown to students is uploaded into the drone. The mission is graded on 2 items: (1) the overall time to acquire target and land and (2) how closely to the target the drone lands. If the mission fails, the best grade a team receives is 80%.

4.3.5 Block Five: Mission Exercise 2: Saving Lives with Autonomous Search and Rescue

The final 12 lessons of CS 472 are dedicated to the final project. Students are presented with a hypothetical scenario in which an individual is injured and stranded in a hard-to-reach place. This person must be found, receive medical attention, and ultimately be rescued. Students are asked to devise a way to locate this person, deliver much-needed medical supplies, and then report the location of the injured person to search and rescue personnel. They are asked to consider what is involved with locating and identifying a stranded person and then delivering a care package to that person. There are many technical challenges that the students need to overcome in order to accomplish this mission. The first challenge is determining how the drone can identify a person from the air. The class is asked to think of a *better, more refined* object tracker over the tracker used in the previous mission to track an object (the person) once the individual has been identified. Other problems that the students must solve include:

- How might they estimate the stranded individual's location?
- How could a package be delivered to the person's location?
- What code could be repurposed from the previous mission?
- What are some new elements that need to be created?

The mission starts with the drone on the ground in a designated field with a care package attached to it by a cable (package contains 3 eggs). The drone will lift off with the package and fly a predetermined flight pattern; could be any pattern loaded by the instructor, students will not know the pattern ahead of time. The objective is to find the stranded person (dressed, mannequin) in the field, safely deliver the care package within ten feet of that person, report the geolocation of the individual and return home. Each team will put together all the code necessary required to complete the mission. Teams may use any code covered in class as well as any code team members created throughout the course.

Teams must determine how to prepare and implement the object recognition model, which requires the use of transfer learning on Yolo4-tiny using the VisDrone dataset (Zhu et al., 2018) to get the model to learn to recognize people from the air, how to locate and track the person in need of help, which requires the implementation of an object tracker, and how to deliver the care package undamaged, which requires the use of the a range finder and a simple trigonometric formula. While some aspects of the project cannot be easily tested with SITL before executing the actual mission, many of the important pieces can. It is up to each team to discover how SITL can help as they build out and test their code.

The final project grade is determined by considering the following elements:

- First, how well the drone was able to identify the objective
- Second, how close to the 10-foot radius the drone was able to deliver its payload without injuring the person (dummy)
- Third, how much damage the payload incurred during the process (ideally, none)
- Forth, how quickly the drone was able to complete its mission
- Fifth, how smoothly the mission was executed overall

5 VALIDATION

CS 472 debuted in Spring 2021 with a total of nine 3rd-year and 4th-year undergraduate CS majors–all of which voluntarily enrolled in our class. All students had some initial exposure to Python from prior courses, and were found to be moderately proficient in the language. All students in the group had taken an elementary AI course in the prior semester, but had no experience with machine learning. As a result, they came to the class with a theoretical understanding of AI concepts and algorithms.

Table 1 summarizes students' grades. In addition to traditional grade performance, we evaluated student feedback so as to improve the next offering of the course. Some of the more important feedback we considered follows:

- 86% of respondents felt that the course activities (readings, lectures, discussions, labs, projects,

Table 1: Overall Class Performance.

Assignment	Avg Grade (%)
AI/ML Application & Eval	87.38%
Mission 1: Target Acquisition	97.71%
Mission 2: Search & Rescue	83.33%
Midterm	76.00%
Quizzes	79.45%
In-class & Take-home Labs	88.60%
Ethics paper	94.56%
Gate Checks	88.64%
Overall	**86.33%**

etc.) were effective in helping them to accomplish the learning goals set forth in the course; 14% disagreed.

- 57% of students felt that graded events (exams, quizzes, homework, projects, papers, etc.) provided them the opportunity to demonstrate their accomplishments related to the course objectives; 28% disagreed and 14% had no opinion.
- For every 1 hour in class, students spent on average 2.4 hours outside of class completing coursework.
- Nearly all respondents indicated that they regard the hands-on nature of the assignments as the *most-liked* aspect of the course.
- The majority of respondents indicated that they felt the course moved too quickly from learning to application.
- 86% of respondents were either satisfied or very satisfied with the knowledge they gained from the course.
- 100% of respondents felt that they got out of the course what they desired to know and understand about ML and autonomous drones.

6 CONCLUSIONS AND FUTURE WORK

We acknowledge the fast-paced nature of this course, and we worked hard to ensure students did not feel overwhelmed with learning material and work. However, some students expressed frustration with putting various elements together to form a solution when working on the major mission projects. 25% of students felt they were being “thrown into the fire” at times. We are reexamining the course material and considering ways that might alleviate frustration for some students. We are using student feedback to determine where hand-holding makes more sense and when to take off the proverbial training wheels. Most of this can be addressed with some additional lab instruction as well as more skeleton code that better organizes drone state during mission execution. We also realize the need for further evaluation of this course that includes comparisons with more traditional ML courses. We plan to further evaluate and compare our methods after Spring 2022 ends.

COVID-19 presented unique challenges for the new course. Much of our hardware was delivered late - well after the semester started. The original intent was to have students interact with the hardware early on, within the first two weeks. We wanted to motivate students by demoing the hardware and allowing students to learn to manually fly the drones very early in the course. Critical hardware items did not arrive until five weeks into the course. This delayed proving out the two main drone mission exercises before issuing them to the students; uncertainty surrounded the challenges that could arise during execution outside of the virtual environment. We encountered a range of issues while initially executing missions outdoors: scattered light, high winds, ground shadows, background trees, even varying visual patterns on the ground. In addition, we encountered unforeseen technical issues with sensors and defective hardware. These problems required quick solutions that allowed the instructor to guide students towards successful project execution.

COVID-19 jeopardized our ability to conduct the course altogether, since much of the course design required students to be in-person and on-site for most labs and projects. The first several weeks from the beginning of the semester was conducted over Microsoft Teams, forcing us to rearrange the order of certain lessons to accommodate remote learning. Fortunately, our class size was small, allowing us to practice social distancing during in-person class time, and by mid-semester we were able to consistently be on-site for labs.

We are considering expanding drone exercises for the next offering (Spring 2022). We are currently experimenting with elements such as targeting moving objects and coordinating exercises between ground-based rovers and aerial drones. We may outfit our drones with gimbals that enable cameras and sensors to move independently from the drone’s body coordinates.

ACKNOWLEDGMENTS

This work is partly sponsored by the Air Force Office of Scientific Research (AFOSR) under Grant FA9550-20-S-0003 as part of the Dynamic Data and Information Processing portfolio of Dr. Erik Blasch. The views and conclusions contained herein are those

of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Academy or the U.S. Government.

REFERENCES

Akbilgic, O. and Davis, R. L. (2019). The promise of machine learning: When will it be delivered?

ArduPilot.org (2016). Sitl simulator (software in the loop).

ardupilot.org (2021). The cube overview.

Azari, A. R., Lockhart, J. W., Liemohn, M. W., and Jia, X. (2020). Incorporating physical knowledge into machine learning for planetary space physics. *Frontiers in Astronomy and Space Sciences*, 7:36.

Biggs, J. B. and Collis, K. F. (2014). *Evaluating the quality of learning: The SOLO taxonomy (Structure of the Observed Learning Outcome)*. Academic Press.

Davies, B. (2016). *Build a drone: a step-by-step guide to designing, constructing, and flying your very own drone*. Simon and Schuster.

Farjo, P. D. and Sengupta, P. P. (2021). Ecg for screening cardiac abnormalities: The premise and promise of machine learning.

Fenner, M. (2019). *Machine learning with Python for everyone*. Addison-Wesley Professional.

Fiebrink, R. (2019). Machine learning education for artists, musicians, and other creative practitioners. *ACM Transactions on Computing Education (TOCE)*, 19(4):1–32.

Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media.

intelrealsense.com (2021). Introducing the intel® realsense™ depth camera d455.

International, U. S. (2019). Aurelia x6 standard - ready to fly.

JetBrains.com (2022). The python ide for professional developers.

Kapoor, A. and Gardner-McCune, C. (2020). Exploring the participation of cs undergraduate students in industry internships. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, pages 1103–1109.

Kolb, D. A., Boyatzis, R. E., and Mainemelis, C. (2014). Experiential learning theory: Previous research and new directions. In *Perspectives on thinking, learning, and cognitive styles*, pages 227–248. Routledge.

Lürig, M. D., Donoughe, S., Svensson, E. I., Porto, A., and Tsuboi, M. (2021). Computer vision, machine learning, and the promise of phenomics in ecology and evolutionary biology. *Frontiers in Ecology and Evolution*, 9:148.

Manolis, C., Burns, D. J., Assudani, R., and Chinta, R. (2013). Assessing experiential learning styles: A methodological reconstruction and validation of the kolb learning style inventory. *Learning and individual differences*, 23:44–52.

nvidia.com (2022). Nvidia jetson xavier nx for embedded & edge systems.

Oborne, M. (2016). Dronecode project inc. *Mission Planner Open-Source Ground Station Software. Available online: http://ardupilot. org/planner/(accessed on 10 January 2021)*.

Perry, A. R. (2004). The flightgear flight simulator. In *Proceedings of the USENIX Annual Technical Conference*, volume 686.

powerbox systems.com (2022). Radio system core.

Rutherford, S. (2020). The promise of machine learning for psychiatry. *Biological Psychiatry*, 88(11):e53–e55.

Sanner, M. F. et al. (1999). Python: a programming language for software integration and development. *J Mol Graph Model*, 17(1):57–61.

Schultz, L. (2014). Readability analysis of programming textbooks: Traditional textbook or trade book? In *Proceedings of the Information Systems Educators Conference ISSN*, volume 2167, page 1435.

Shapiro, R. B., Fiebrink, R., and Norvig, P. (2018). How machine learning impacts the undergraduate computing curriculum. *Communications of the ACM*, 61(11):27–29.

Smith, K. N. and Green, D. K. (2021). Employer internship recruiting on college campuses: 'the right pipeline for our funnel'. *Journal of Education and Work*, 0(0):1–18.

Smolkin, L. B., McTigue, E. M., and Yeh, Y.-f. Y. (2013). Searching for explanations in science trade books: What can we learn from coh-metrix? *International Journal of Science Education*, 35(8):1367–1384.

Stevenson, J. (2020). *Developing vocational expertise: principles and issues in vocational education*. Routledge.

Sulmont, E., Patitsas, E., and Cooperstock, J. R. (2019). What is hard about teaching machine learning to non-majors? insights from classifying instructors' learning goals. *ACM Transactions on Computing Education (TOCE)*, 19(4):1–16.

Toole, A. A., Pairolero, N. A., Forman, J. Q., and Giczy, A. V. (2019). The promise of machine learning for patent landscaping. *Santa Clara High Tech. LJ*, 36:433.

Trister, A. D., Buist, D. S., and Lee, C. I. (2017). Will machine learning tip the balance in breast cancer screening? *JAMA oncology*, 3(11):1463–1464.

Valstar, S., Krause-Levy, S., Macedo, A., Griswold, W. G., and Porter, L. (2020). Faculty views on the goals of an undergraduate cs education and the academia-industry gap. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, pages 577–583.

von Lilienfeld, O. A. and Burke, K. (2020). Retrospective on a decade of machine learning for chemical discovery. *Nature communications*, 11(1):1–4.

Zhu, P., Wen, L., Du, D., Bian, X., Ling, H., Hu, Q., Nie, Q., Cheng, H., Liu, C., Liu, X., et al. (2018). Visdrone-det2018: The vision meets drone object detection in image challenge results. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0.