# AutoLearn: Learning in the Edge to Cloud Continuum


Alicia Esquivel Morel
ace6qv@mail.missouri.edu
University of Missouri
Columbia, MO, USA

William Fowler
william.fowler@tufts.edu
Tufts University
Medford, MA, USA

Kate Keahey
keahey@mcs.anl.gov
Argonne National Laboratory
Lemont, IL, USA

Kyle Zheng
kyle996032@my.yosemite.edu
Modesto Junior College
Modesto, CA, USA

Michael Sherman
shermanm@uchicago.edu
University of Chicago
Chicago, IL, USA

Richard Anderson
rianders@docs.rutgers.edu
Rutgers University
New Brunswick, NJ, USA



## ABSTRACT

Technological advancements have led to an increase in teaching the fundamentals of cloud computing, robotics and autonomous systems and their importance, relying on strong hands-on practical experimentation. The National Science Foundation (NSF)-supported testbeds have opened the doors for experimentation and support in the next era of computing platforms and large-scale cloud research. In this paper, we present an educational module that conveys accessibility to education, aiming to prepare learners for technological career paths with the motivation to bring hands-on sessions, and on the idea of building a freely available set of artifacts that can serve the educational community. Specifically, we present AutoLearn: Learning in the Edge to Cloud Continuum, an educational module that integrates a collection of artifacts, based on a small scale open-source self-driving platform that leverages the Chameleon Cloud testbed to teach cloud computing concepts, edge devices technology, and artificial intelligence driven applications.



## CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

## KEYWORDS

computer science education, cloud computing

**ACM Reference Format:**
Alicia Esquivel Morel, William Fowler, Kate Keahey, Kyle Zheng, Michael Sherman, and Richard Anderson. 2023. AutoLearn: Learning in the Edge to Cloud Continuum. In *Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis (SC-W 2023), November 12–17, 2023, Denver, CO, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3624062.3624101


## 1 INTRODUCTION

We live in exciting times: rapid advances in Machine Learning (ML), robotics, the Internet of Things (IoT), and automation herald the Fourth Industrial Revolution (I4.0) [5]. These changes create significant opportunities in our society, but they also transform it, resulting in a rapidly developing need for skills in new areas of technology and the consequent shift in the job market [1, 19]. Academic institutions responded to this need by introducing courses in robotics, autonomous systems, ML, cloud computing, and data science [21] – however, these topics can only be taught through hands-on learning that gives the students a taste of the theory behind the innovations, a more intuitive feel for the real-world problems and solutions that make it up. Hands-on learning confronts students with the reality of new technology, demands and inspires critical thinking, ingenuity in problem solving, and hones the ability to adapt to fluid technological landscape. However, a precondition for such technology exploration is having access to not only high-end resources but also to digital curricula that can support such experiential exploration and can also rapidly adapt to technology changes.

In recent years, the National Science Foundation (NSF) has established multiple experimental platforms for computer science research. Testbeds such as Chameleon [17], CloudLab [10], FABRIC [3], and the PAWR testbeds [4, 6, 20, 22, 25] collectively provide capabilities where any topic in computer science can be experimented with – or taught. This creates a powerful opportunity that both amortizes the costs of procurement, and democratizes access to innovative and often expensive resources. It is also a capability that can potentially disrupt the space of digital artifact sharing – since digital artifacts typically require some form of computational capability to interpret, and that in turn relies on access to infrastructure that can provide it. Computer science education is one of the areas that could benefit the most from this development, as much of the computer science learning, especially in the area of systems, requires experiential learning. However, while there are well-established ways of sharing traditional course curricula that adapt knowledge for the purposes of learning e.g., in the form of textbooks or exercise books, the momentum behind sharing digital artifacts is only just developing, giving rise to questions such as: What do digital educational artifacts comprise? How should they be shared? What are the digital learning use cases (classes, self-learning, etc.) How should they be supported by infrastructure or infrastructure-related services? How can digital artifacts be sustained?

This paper presents AutoLearn: Learning in the Edge to Cloud Continuum, an educational module that integrates a collection of



*SC-W 2023, November 12–17, 2023, Denver, CO, USA*



ACM ISBN 979-8-4007-0785-8/23/11…$15.00
https://doi.org/10.1145/3624062.3624101

educational artifacts, based on DonkeyCar, an open-source self driving platform for small scale cars [24] that leverages the Chameleon testbed to teach topics relating to autonomous driving ranging through engineering, data collection, interactions in edge to cloud continuum, various types of ML, to digital twin models. The artifacts offer several digital learning pathways and allow educators to tailor the content to different learning objectives; for example, by emphasizing engineering aspects by focusing on modifications to the self-driving cars or tracks, or shifting emphasis to ML by using simulators for car driving and focusing on model training and validation instead. We describe how the artifacts leverage various features of the Chameleon testbed, as well as the Trovi digital artifact hub [14]. Lastly, we offer thoughts on how digital artifacts can rely on community support by borrowing a model from open source development practices.

## 2 EXPERIMENTAL PLATFORMS FOR EDUCATION

Enabling the academic research community with novel cloud architectures such as "bare-metal access", and improving access to experimentation is one of the main goals of experimental platforms or testbeds that are supported by the NSF. These testbeds represent an unique infrastructure, fundamentally important to advance and support experimentation. Efforts in [13] developed educational modules for topics in networking, security and cloud computing demonstrating how instructors and students can benefit from these topics and especially with the experimentation in real-world testbeds. Hands-on experiments based on modules that are openly available and that can be used by institutions to enhance their educational curricula. The Fair Use Building And Research Labs (FUBAR) supports and maintains FOOCars, a low cost racing autonomous vehicles project [2]. Authors highlight the importance of these testbeds to test assumptions and learn not only about topics related to cloud computing but also training on ML, sensors, micro-controllers or edge devices. The team created a use-case with an autonomous remote controller car platform with the main goal to provide students with the experience of transitioning from the physical world to remote devices experimentation. Authors also claim how this project can benefit students, teachers or researchers in new fields, including testing models for autonomous driving.

Trying to minimize the gap between the transition on building use-cases that can span from controlling a device remotely or hosting new development on local physical devices is also necessary for this learning process. Chameleon Cloud [16] has successfully supported educational projects, and it has structured the learning experience pipeline in a way to provide students and teachers with unique features such as Chameleon edge testbed, called CHameleon Infrastructure (CHI) at Edge (CHI@Edge) [15]. This has allowed the experimentation with application containerization onboard small and remote controlled cars. In addition, the orchestration aspect of resource reservations, availability of images and libraries, and most importantly getting access to both edge and cloud resources makes the learning experience undoubtedly easy to interact with.

As a starting point, we built AutoLearn, a first of its kind educational module that leverages these testbed capabilities and delivers a set of instructional materials, cloud resources, and ultimately a complete loop pipeline to a holistic learning experience. For building this educational module, our goal is based on the idea of building a freely available set of modules that can serve the educational community and to bring instructional materials across the learning process. We built our work on the basis of creating not only educational materials that can teach about cloud computing or networking, but to build instruction to offer access to learning topics such as engineering, robotics or ML. Most notably, we have seen the higher barrier of entry to these areas of research, and with a scarcity of well-organized, free and easy to follow educational materials. Our contributions are built having extensibility, flexibility, affordability and adaptability in mind. With different options available in the market, we rely on DonkeyCar, an open source small self-driving car platform, with a very simple and organized set of instructions that aims students to take inputs and return outputs in the pipeline. From data collection to training models to ultimately evaluating the learning experience, we consider this platform not only for its minimalism, but also for its modularity. Our approach is built on top of the Chameleon testbed, aiming to provide free and accessible education for individuals of all backgrounds and skill-levels to devise education initiatives, benefiting society by spreading knowledge and lowering barriers to entry.

## 3 A SHAREABLE DIGITAL EDUCATIONAL ARTIFACT

### 3.1 Autonomous Car Module

Our autonomous cars educational module has been designed to build on the existing efforts described in the previous section and adapt them to take advantage of resource availability provided by open research and educational platforms, specifically the Chameleon testbed and CHI@Edge, thus putting this type of learning within reach of many students, either as part of directed or self study. We supplement these platforms by making specific recommendations for purchase of inexpensive ~($200) and generally available cars kits and accessories that minimize the configuration time for this type of course [23]. Similar to the existing approaches, the learning outcomes for this module span the following: familiarity with assembling hardware, basic familiarity with systems topics (basic knowledge of UNIX, understanding how to configure hardware and software, etc.), basic familiarity with cloud and edge computing, basics of computer simulation, and ML topics spanning data collection and cleaning, training a ML model, and finally actuating a successful ML model with an autonomous car. The following section describes the pre-conditions for setting up the module, the expected time investment on the part of instructors and students, and additional resources and capabilities available by leveraging the Chameleon testbed.

### 3.2 Cloud and Edge Hardware Requirements

Our educational module is built on top of the Chameleon platform, and in particular the CHI@Edge segment of Chameleon to manage the autonomous cars and integrate them with the testbed. Chameleon is an NSF-funded testbed that supports computer science research and education; to gain access all educational users need to do is request a project in computer science education. The

testbed contains a large collection of diverse hardware resources over several sites. In particular, it has a large investment in accelerators ranging from 40 nodes with a single Nvidia RTX6000 GPU for general use, to sets of 4 nodes each with 4x Nvidia V100, P100, or A100 Datacenter GPUs and InfiniBand interconnects to scale larger experiments, for example HPC or ML training workloads. Smaller numbers of nodes with other architectures (Nvidia M40, K80, AMD MI100) round out a variety of choices. All hardware is available either on-demand or via advance reservations so that users can reserve required resources ahead of time, for example, to manage resource scarcity or to guarantee resource availability at a specific time slot for a class or a demonstration. The hardware is re-configurable on bare metal level to support research on topics such as power management or provide a controlled environment for performance measurement. Further, the two principal Chameleon sites are connected to the FABRIC networking testbed creating potential to support cloud experiments with managed latency.

Once part of a Chameleon project, users can log into the testbed with their institutional credentials via federated identity login and then interact with it via a GUI, or programmatically via the command line and python interfaces. To support experiment development and sharing, Chameleon integrates the programmatic interfaces with Jupyter so that users can package their experiments more easily and combine experimental environment creation, experiment body, and analysis in one set of notebooks. To make sharing such experiments viable, the system also provides Trovi [14], an experiment hub integrated with the testbed (and other testbeds in the future via open APIs) so that users can not only find experimental artifacts, but interact with them easily.

Since it first came online in mid-2015, Chameleon has served 8,000+ users, working on 1,000+ projects, whose scientific output resulted in 600+ publications. In addition to providing access to resources in the datacenter, Chameleon also supports the Bring Your Own Device (BYOD) paradigm that allows users to add their own devices to the testbed for limited sharing. In this paradigm, users can add devices to the testbed by downloading a CHI@Edge command line utility and SD card image; the utility registers the device with the testbed, and configures the SD card image to be flashed onto the device. Once booted up, the image contains a daemon that connects the device to the testbed and configures whitelist-based access policies for the added device. From there on, the added device can be allocated via the standard Chameleon methods, i.e., federated identity login, resource discovery, and advance reservations – though it is reconfigured by deploying a Docker container rather than bare-metal reconfiguration. In particular, users can manage the device through our Jupyter interface and within the same Chameleon session as the data center resources. This allows users to create and manage complex resource topologies and interactions in the edge to cloud continuum from one Jupyter notebook.

## 3.3 The Educational Module: Structure

The structure of the educational module is designed to give students, self-learners and teachers the necessary guidance and walk-through in an easy to follow set of instructions. The structure is divided into three main sub-components with artifacts, computational components and ultimately extensions and assignments as can be observed in Fig. 1, which can be used to reinforce, apply, and assess the new learned skills. Below, we describe the composition and function of each component.

**Collecting and cleaning data:**
Collecting data from manual driving sessions is the first step students need to get through in the AutoLearn training module. Fig. 2 illustrates three different data collection paths. Sample datasets, and data collected through the Unity game platform via simulation, and through the real physical car. For those students who have access to the physical car, the most interesting method is to drive the car around an actual track. Students can drive the car using a physical joystick controller, or use the DonkeyCar web controller that provides the same functionality via a web interface and sends the commands to the car. Both modes provide a variety of options such as setting the throttle as constant (useful if the car is used in races with a pilot that will steer but does not control throttle). By default, all data is stored on the Raspberry Pi /car/data and can be manually transferred to the cloud using SSH for either storage as sample dataset or for training (this workflow is part of AutoLearn instructions, see [11]). Alternatively, students can use the DonkeyCar simulator which allows them to do the same – but with a virtual car in a simulated track environment. The simulator includes several different tracks to choose from. After the simulator is set up, all other functionality relating to driving the car and collecting data is the same. The simulator is not computationally intensive and can be installed on various OS, including Windows, Linux and MAC machines, i.e., students' laptops.

**Additional data collection:**
Learners will likely generate some bad data consisting of mistakes (i.e., crashes or images that are off-side) while driving; this data need to be deleted for the training set to represent a valid scenario. This step is done manually by using the tubclean utility included in the DonkeyCar python package, which plays a video of the collected images; users watch the video, select the parts that need to be deleted, which the program then correlates to invalid data records that need to be cleaned up.

**Sample datasets:**
The AutoLearn package also contains sample datasets that students can use to train models without having to drive the car. The sample datasets were collected by manually driving the car around a track, and through the DonkeyCar simulator. We used a default track that was made with an orange tape oval shape with the following dimensions; inner line length: 330 in, outer line length: 509 in and average width: 27.59 in. Students can replicate the default track following the dimensions and as it can be seen in Fig. 3 (a). Fig. 3 (b) corresponds to the Waveshare track, which is a commercial track. Each of the existing datasets contains 10-50K records, records that consist of .catalog files, images directory, and manifest files. .Catalog files consist of steering and throttle values that were recorded while driving. Each of these corresponds to an image in the images directory based on their id number. Catalog_manifest files store information about each catalog file and the manifest.json file is where certain records are marked for deletion. These sample datasets are stored in our educational module's GitBook, and can be accessed in [11]. This can be seen as a good entry level exercise that gets students familiar with the car setup; they can be generated either

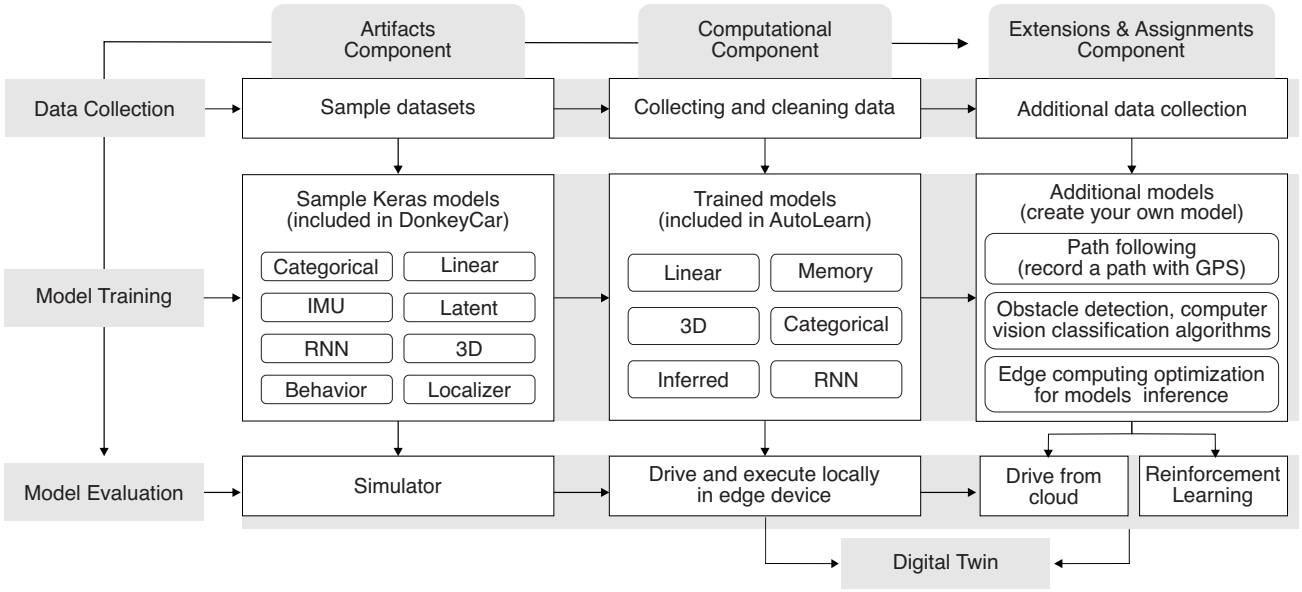


**Figure 1: AutoLearn offers three main components, artifacts, computation and extension/assignments, within a comprehensive ML pipeline including data collection, model training and evaluation. Students and instructors can make use of sample datasets that can be trained with sample Keras models (included in DonkeyCar), collect / clean data and try out models (included in AutoLearn). Lastly, more advanced students can collect additional data that can aim to create new models for training.**

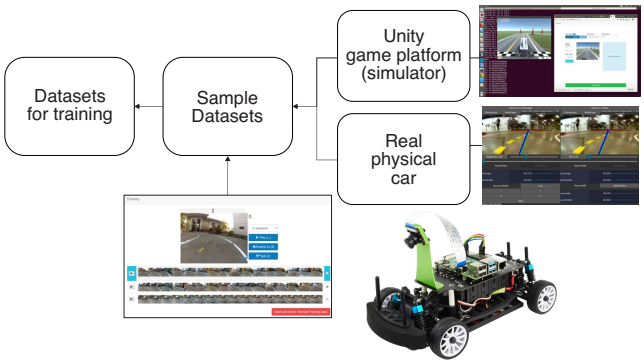


**Figure 2: AutoLearn provides three different data collection paths. Sample datasets, data collected through the Unity game platform via simulation, and through the real physical car. Ultimately, these datasets serve for training the ML models.**

**(a) Default track** **(b) Waveshare track**

**Figure 3: Sample datasets can be collected by manually driving the car around a track, and through the DonkeyCar simulator. We used a default track that was made with an orange tape oval shape and a commercial conventional track (Waveshare).**

by driving the actual car or via the DonkeyCar virtual driving environment. Possible variations include modifying the shape of the track, varying the car configuration and/or driving conditions e.g., by changing the surface of the track, or modifying the car itself. This is a "beginner level" assignment that allows students to easily experiment with effects of different datasets on different training models.

**Model training:**

Once the data is obtained by any of the methods described above, students can proceed to training models. The AutoLearn module provides a Jupyter notebook that reserves Chameleon hardware, deploys Ubuntu 20.04 CUDA image with accelerator support, and then installs and configures all the required dependencies including Donkey, Tensorflow, and CUDNN drivers. DonkeyCar provides by default several Keras models to choose for training a self-driving model. By default, a learner can start with the Linear model with an easy to understand pipeline. AutoLearn comes with six tested models, including linear, memory, 3D, categorical, inferred, and RNN; other models can be also tried, but they require doing extra configuration and / or hardware. We tested this process on a

range of GPU nodes available via Chameleon including A100, V100, v100NVLINK, RTX6000, and P100. Once the instance is deployed and configured with all the dependencies, the student copies the training data using rsync command and can begin the training process.

**Training Additional Models:**

This is a good area for further study that additionally lends itself to competitions between student teams. Extensions and exercises relating to model training include comparisons between different models (e.g., we found that the inferred model was best because it gave the car the ability to speed fast, while still being accurate); path following (record a path with GPS and have the car follow that path); obstacle detection; various computer vision classification algorithms (example: camera identifies color of object placed in front of it; red means stop, green means go); and edge detection/line following (camera used to identify the edge of the track or a center line and keep the car following that). Students might also compete to train models yielding a combination of fastest speed with fewest errors, or accuracy following tracks of different shapes.

**Model Evaluation:**

For this segment, teachers or TAs make the Raspberry Pi devices associated with the cars (and therefore the cars themselves) available via the BYOD functionality of CHI@Edge (see Section 2). Students can thus treat the cars as any other Chameleon resource, download the trained models onto them for inference, and drive them around the track measuring qualities of interest (speed, number of errors, etc.) as suggested in the AutoLearn documentation. As in the case of data collection, students without access to a physical car, can use DonkeyCar simulator to perform the evaluation virtually. Extensions and assignments associated with this stage include modules exploring the edge to cloud interaction by attempting to run inference models in the cloud, constructing hybrid edge cloud inference models, or using reinforcement learning. Lastly, combining the simulator and real-life validation can lead to interesting exploration of digital twin modeling.

## 3.4 The Teaching Module

The teaching module is structured to support multiple learning pathways depending on learning objectives, available equipment, time, other resources, and learning ability. The overall learning pathway consists of the rough three phases illustrated in Fig. 1: data collection, model training, and model evaluation. However, each of the phases has multiple alternatives that can be used to customize the student's learning pathway. For example, instead of collecting actual data, students can use the datasets provided in AutoLearn. Further, students can use one of the packed pre-trained models or explore new models with different training objectives, or with default pre-trained models that are included in DonkeyCar. Lastly, they can validate the models running an actual car, or by using the simulator if the car is not available, or even combine both to use digital twin exploration. Providing alternatives in each of the phases is an interesting way to extend the module creating potential for additional exercises or homework assignments ranging in level from beginner to advanced. For example, additional data collection could be a good beginner/warmup exercise while in the validation phase students can extend the module by exploring in-situ versus in the cloud inference or experiment with reinforcement learning providing the opportunity for more advanced assignments. Finally, a range of interesting projects can be based on developing a digital twin model based on comparing the simulation output with real-life model evaluation. Different pathways through this educational module could also shift the focus of the exploration from engineering to ML: using available datasets and a simulator does not require a car, focuses on training and evaluating models, and can be used as part of a ML course. At the same time, an engineering course can focus on data collection with different car configurations and evaluate them using ready-made training modules.

## 3.5 The Educational Module: Implementation

**Chameleon and CHI@Edge support:**

The educational module is implemented in CHameleon Infrastructure (CHI) at Edge (CHI@Edge) to manage the car; Chameleon's integration with Jupyter to implement the various instructional elements in a way that combines explanations in text, with implementation, and instructional videos and pictures; and the Trovi experiment hub integrated with GitBook [11] to share the artifact. The artifact thus consists of a series of Jupyter notebooks that can be imported/exported to the GitBook to leverage integration with Chameleon on one side, and contribution and feedback features on the other.

**CHI@Edge and BYOD functionality:**

To manage interactions with the car we used CHI@Edge, adding the car via the BYOD functionality. This allows a student to launch a container on the car's Raspberry Pi using a Docker image which pre-installs all DonkeyCar dependencies simply by executing one cell in the corresponding Jupyter notebook; this provides a "zero to ready" configuration pathway with minimum time and effort. A further advantage of using CHI@Edge is that after launching a container, there is a built-in console in Jupyter for running commands on the Raspberry Pi. This was helpful though we had to work around the fact that text editing is not supported in the console at the present time.

**Chameleon's Basic Jupyter Server Appliance:**

To provide a seamless "Jupyter experience" that covers both container establishment and the data collection programs that run inside the container, we used Chameleon's Basic Jupyter Server Appliance [7] and included it in AutoLearn Docker image; this allows students to access the Jupyter Notebook executing on the Raspberry Pi (and containing all the data collection functionality) from their own laptops using an SSH tunnel. To implement model training we used Chameleon's datacenter resources to reserve a bare-metal node with a v100 GPU (though other GPU resources like A100 would work as well as documented in our instructions); this allowed us to train a model in reasonable amount of time.

**Trovi and Chameleon's artifacts**

In addition to using Chameleon's resources, we were also able to leverage and augment experiment patterns from artifacts published on Trovi, and other artifacts [18] using CHI@Edge; this speed up our development considerably. Similarly, others can extend or modify our notebooks to implement new training models as part of extensions or exercises in our educational module. Leveraging the programmatic interface to the system via Jupyter notebook was

in general very helpful as it allowed us to streamline often complex configuration of highly programmable resources by combining them in Jupyter cells that can be executed with one click.

**Chameleon's Object Store:**

The collected datasets and the pre-trained models are stored in Chameleon's object store [9] and can be combined with other components of the system in a "mix and match" pathway.

**Educational materials:**

The AutoLearn educational materials include documentation supporting different roles and different settings. For directed learning, we provide documentation for educators including course objectives, explanations of what hardware to buy and alternatives, proposed project extensions, and a one-page TA checklist. To support students, our GitBook [11] is documented with extensive comments with instructions and videos on how to set up and drive the car for data collection and cleaning. Finally, we provide a special documentation pathway for digital self-learners that contains a combination of teacher's and student's documentation modules in a more streamlined form as self-learners are likely to play both roles.

## 4 CONTRIBUTIONS AND FEEDBACK

Our contributions are tangible through an exhaustive digital content freely available that can be followed in three different pathways, i.e. regular, classroom, and digital path, based on student's interests, background or goals. In addition, Chameleon's Trovi allows users to import and export artifacts to / from our GitHub repository as the best path to support collaborative development and at the same time offering a link to experimental infrastructure. Understanding how the educational module can be supported as a long-term project and as a foundational resource for teaching and learning is also important to us. We provide a set of instructions in our GitBook artifact, in which learners can start their own educational module. This can be synced and learners can make additional changes to the module, make extensions or improvements. Through collaborative support and learning, students can make a merge request to the original repository so then the learning community can have access to different versions and updates of the project. As the educational module gains contributions and momentum, we are positive that it can continually improve. In addition, we facilitate a Google Group [8] and a set of instructions for providing feedback or sharing case study information about how the educational materials benefited or what improvements can be made. As we are in the early stage of developing this educational module, we hope to be very responsive to any feedback from instructors, students, or any member of the community.

## 5 IMPACT OF THE EDUCATIONAL MODULE

One important question is how much impact a learning module might have; this is particularly important if we are to create incentives for module authors and contributors who are likely to be motivated by the impact. While it is too early to provide a definitive answer to this question in the context of this particular module, we suggest a couple of avenues of doing that. One is to rely on quantitative distribution metrics provided by the venue through which the module is distributed such as number of downloads, views, executions, review rankings, likes, etc. In our case, those can be obtained from Trovi [14], which for each artifact lists the number of views as well as executions (benefit of platform integration), defined as the execution of at least one cell in the artifact packaging. In addition, Trovi provides a clear view of the artifact life-cycle management, where we can keep track of new versions, and apply metadata such as related tags, description or author lists.

Possible integration with social network-like mechanisms could in the future provide a more sophisticated feedback by adding a more fine-grained and verbose comment feedback; in the context of our packaging this role in our "Contributing Community" and "How to Provide Feedback modules" [11]. As of this writing, since its publication in September 2023, the numbers for our artifact in Trovi are modest: 35 total number of launch button clicks, 9 users who clicked the launch button, 2 users who executed at least one cell, and it has been published 8 versions of the artifact. The advantage of these mechanisms is that the information they provide can be collected in an automated fashion without placing a reporting burden on the users of the artifact. The disadvantage is that they represent an outcome rather than impact: in other words, we can say how many people used an artifact but not how much they were able to achieve using it. The latter information is more valuable but harder to obtain and collecting it usually requires some participation from targeted student groups.

In our case, we were surprised to find that the two REU students who participated in the development of the educational artifact, once they became proficient in it, it took only two weeks each to develop an independent research project that resulted in two posters accepted to the ACM student competition at SC'23 [12, 26]. Each of the students followed different avenues of exploration recommended in the various extensions described in Section 3.3 and were able to focus on the chosen research question immediately. This suggests that the module represents not only a good preparation for independent research but also provides a framework on which research investigations can be built, in effect providing a "shortcut" to independent exploration.

## 6 CONCLUSIONS AND FUTURE WORK

In this paper, we focus on the development and design of instructional materials that can be used in introduction to engineering classes, robotics, ML, hardware, edge devices and cloud platforms. Our goal is primarily to serve as a starting point for future educational modules that can integrate several fields. We aim to offer a comprehensive source of practical exercises and open materials to enable learners to understand and apply cloud and edge computing paradigms into real-world deployments. Our instructional materials are based on DonkeyCar, an open-source self driving platform for small scale cars that leverages the Chameleon testbed, and that can serve different learning pathways i.e. regular, classroom, and digital path. Our educational module brings a lot of potential for future directions or improvements including further validation. Our next step is to improve our validation steps with real-classroom deployment. We are interested in collecting data on how well the materials are instructed or learned, in addition to cover additional topics that can aim students to learn more about cutting-edge technologies and real-world applications or develop ideas on how to integrate multiple disciplines within these areas. Usability studies

within different ranges of students, including undergrads, graduates and self-learners will also add value to enhance AutoLearn. Ideally, these would take the case of the applicability of AutoLearn to final projects, ideas for integration with other technologies and / or testbeds, or the extension of the work for research purposes.

As a clear road map and long-term impact, understanding students' perceptions would definitely add value to the student's learning process within our educational module. In addition, since this area and field is constantly evolving we are also aware of potential future improvements or adaptations that need to be added to the educational module. AutoLearn can be extended in other technologies within these areas including the integration of other intelligent autonomous vehicles in general such as unmanned aerial vehicles or drones, in addition to other applications such as precision agriculture that can lead to a broader application integration including sensors or robots.

## ACKNOWLEDGMENTS

This paper's results were obtained with support from the National Science Foundation through the Chameleon testbed (Award Number 2027170) and the FOUNT project (Award Number 2230077). The opinions, findings, conclusions, or recommendations expressed in this publication are solely those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

## REFERENCES

[1] Mohammad Akour and Mamdouh Alenezi. 2022. Higher education future in the era of digital transformation. *Education Sciences* 12, 11 (2022), 784.

[2] Richard Anderson. 2023. *fubarlabs: Low Cost Racing Autonomous Vehicles: RC Cars to Power Wheels Racers.* Retrieved August 4, 2023 from https://github.com/fubarlabs/foocars/

[3] Ilya Baldin, Anita Nikolich, James Griffioen, Indermohan Inder S Monga, Kuang-Ching Wang, Tom Lehman, and Paul Ruth. 2019. Fabric: A national-scale programmable experimental network infrastructure. *IEEE Internet Computing* 23, 6 (2019), 38–47.

[4] Leonardo Bonati, Pedram Johari, Michele Polese, Salvatore D'Oro, Subhramoy Mohanti, Miead Tehrani-Moayyed, Davide Villa, Shweta Shrivastava, Chinenye Tassie, Kurt Yoder, et al. 2021. Colosseum: Large-scale wireless experimentation through hardware-in-the-loop network emulation. In *2021 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*. IEEE, 105–113.

[5] Christopher Alan Bonfield, Marie Salter, Alan Longmuir, Matthew Benson, and Chie Adachi. 2020. Transformation or evolution?: Education 4.0, teaching and learning in the digital age. *Higher education pedagogies* 5, 1 (2020), 223–246.

[6] Joe Breen, Andrew Buffmire, Jonathon Duerig, Kevin Dutt, Eric Eide, Mike Hibler, David Johnson, Sneha Kumar Kasera, Earl Lewis, Dustin Maas, et al. 2020. POWDER: Platform for open wireless data-driven experimental research. In *Proceedings of the 14th International Workshop on Wireless Network Testbeds, Experimental evaluation & Characterization*. 17–24.

[7] Chameleon Cloud. 2023. *Basic Jupyter Server Appliance.* Retrieved August 2, 2023 from https://www.chameleoncloud.org/experiment/share/39ae6822-b078-4707-8323-76eaf3a7e213

[8] Chameleon Cloud. 2023. *Chameleon Education Google Group.* Retrieved August 1, 2023 from https://groups.google.com/g/chameleon-education/

[9] Chameleon Cloud. 2023. *Chameleon Object Store.* Retrieved August 5, 2023 from https://chameleoncloud.readthedocs.io/en/latest/technical/swift.html

[10] Dmitry Duplyakin, Robert Ricci, Aleksander Maricq, Gary Wong, Jonathon Duerig, Eric Eide, Leigh Stoller, Mike Hibler, David Johnson, Kirk Webb, et al. 2019. The design and operation of {CloudLab}. In *2019 USENIX annual technical conference (USENIX ATC 19)*. 1–14.

[11] Alicia Esquivel Morel, William Fowler, Kate Keahey, Kyle Zheng, Michael Sherman, and Richard Anderson. 2023. *CHI@Edge Education.* Retrieved August 4, 2023 from https://chi-education.gitbook.io/chi-edge-or-education/

[12] William Fowler, Kate Keahey, and Alicia Esquivel Morel. 2023. Road To Reliability: Optimizing Self-Driving Consistency With Real-Time Speed Data. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC'23 Poster)*.

[13] Fraida Fund. 2023. *Teaching on Testbeds.* Retrieved August 1, 2023 from https://teaching-on-testbeds.github.io/

[14] Keahey Kate, Anderson Jason, Powers Mark, and Cooper Adam. 2023. Three Pillars of Practical Reproducibility. In *rewords23: 3rd Workshop on Reproducible Workflows, Data Management, and Security During eScience'23.*

[15] Kate Keahey, Jason Anderson, Michael Sherman, Cody Hammock, Zhuo Zhen, Jenett Tillotson, Timothy Bargo, Lance Long, Taimoor Ul Islam, Sarath Babu, et al. 2022. CHI-in-a-Box: Reducing Operational Costs of Research Testbeds. In *Practice and Experience in Advanced Research Computing*. 1–8.

[16] Kate Keahey, Jason Anderson, Zhuo Zhen, Pierre Riteau, Paul Ruth, Dan Stanzione, Mert Cevik, Jacob Colleran, Haryadi S Gunawi, Cody Hammock, et al. 2020. Lessons learned from the chameleon testbed. In *2020 USENIX annual technical conference (USENIX ATC 20)*. 219–233.

[17] Kate Keahey, Pierre Riteau, Dan Stanzione, Tim Cockerill, Joe Mambretti, Paul Rad, and Paul Ruth. 2019. Chameleon: a scalable production testbed for computer science research. In *Contemporary High Performance Computing*. CRC Press, 123–148.

[18] Bobadilla Leonardo, Tsen Jonathan, Anderson Jason, and Keahey Kate. 2021. Choosing Optimal Edge Topologies for Real-Time Autonomous Fish Surveys.. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC'21 Poster)*.

[19] Ling Li. 2022. Reskilling and upskilling the future-ready workforce for industry 4.0 and beyond. *Information Systems Frontiers* (2022), 1–16.

[20] Vuk Marojevic, Ismail Guvenc, Rudra Dutta, Mihail L Sichitiu, and Brian A Floyd. 2020. Advanced wireless for unmanned aerial systems: 5G standardization, research challenges, and AERPAW architecture. *IEEE Vehicular Technology Magazine* 15, 2 (2020), 22–30.

[21] Alexandre Oliveira, Heitor Assumpcao, Jonas Queiroz, Luis Piardi, Javier Parra, and Paulo Leitao. 2022. Hands-on learning modules for upskilling in industry 4.0 technologies. In *2022 IEEE 5th International Conference on Industrial Cyber-Physical Systems (ICPS)*. IEEE, 1–6.

[22] Dipankar Raychaudhuri, Ivan Seskar, Gil Zussman, Thanasis Korakis, Dan Kilper, Tingjun Chen, Jakub Kolodziejski, Michael Sherman, Zoran Kostic, Xiaoxiong Gu, et al. 2020. Challenge: COSMOS: A city-scale programmable testbed for experimentation with advanced wireless. In *Proceedings of the 26th annual international conference on mobile computing and networking*. 1–13.

[23] Waveshare. 2023. *Waveshare, PiRacer Pro AI Kit.* Retrieved August 3, 2023 from https://www.waveshare.com/wiki/PiRacer_Pro_AI_Kit

[24] Roscoe Will and Conway Adam. 2019. *Donkey car: An opensource DIY self driving platform for small scale cars.* Retrieved August 4, 2023 from https://www.donkeycar.com/

[25] Hongwei Zhang, Yong Guan, Ahmed Kamal, Daji Qiao, Mai Zheng, Anish Arora, Ozdal Boyraz, Brian Cox, Thomas Daniels, Matthew Darr, et al. 2022. ARA: A wireless living lab vision for smart and connected rural communities. In *Proceedings of the 15th ACM Workshop on Wireless Network Testbeds, Experimental evaluation & Characterization*. 9–16.

[26] Kyle Zheng, Kate Keahey, and Alicia Esquivel Morel. 2023. Chasing Clouds with Donkeycar: Holistic Exploration of Edge and Cloud Inferencing Trade-Offs in E2E Self-Driving Cars. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC'23 Poster)*.