Digital Discovery

ROYAL SOCIETY OF CHEMISTRY

PAPER

View Article Online
View Journal

Cite this: DOI: 10.1039/d5dd00525f

A user's guide to your first self-driving liquid handling lab

Apostolos P. Maroulis,†[a] Dylan M. Waynor,†[a] Quinn M. Gallagher,[b] Roshan A. Patel,[b] Matthew Tamasi,[a] D. Christopher Radford,[a] Michael A. Webb *[b] and Adam J. Gormley *[a]

Experimentation is inherently difficult because most methods require substantial refinement, calibration, and validation before high-quality, reliable data can be collected. In most cases, experimental outcomes are impacted by multiple variables, thus requiring their simultaneous optimization for single and multi-objective targets. Traditional experimental approaches rely on trial-and-error methods guided by rational decision making, but these become increasingly inefficient and ineffective as complex interactions between inputs limit our ability to capture underlying trends using conventional statistical approaches. Machine learning and active learning (ML/AL) combined with automation represents an approach that can bolster future laboratory productivity. However, a steep initial learning curve and high costs of instrumentation pose substantial barriers to adoption. To democratize access, we herein comprehensively cover both the computational skills and hardware implementation necessary for self-driven experimental workflows. The accompanying open-source, low-cost liquid handling platforms offer practical templates for researchers adopting self-driving lab (SDL) methodologies. Complete tutorials and build guides are provided at https://gormleylab.github.io/SDLGuide.

Received 25th November 2025
Accepted 25th March 2026

DOI: 10.1039/d5dd00525f

rsc.li/digitaldiscovery

1 Introduction

Major goals of scientific research include advancing fundamental understanding of phenomena and developing technologies. In practice, this often involves using experimentation to determine how controllable input parameters (*e.g.*, processing conditions, molecular structure, composition) influence a complex output property of interest (*e.g.*, material performance, yield, selectivity, and stability). Traditionally, researchers have selected experiments by leveraging their own intuition (*e.g.*, rational experimental design) or more systematic approaches such as classical design of experiments (DOE),[1,2] which are often supplemented by simple statistical methods to model the relationship between experimental parameters and the property of interest.[3] While powerful, these approaches can face limitations when the input parameter space is high-dimensional, the relationship between input parameters and output property is highly nonlinear, and experiments are time and/or resource-intensive.

Recent advances in machine learning (ML) and automation offer new ways to overcome these barriers and accelerate scientific research.[4–6] Specifically, supervised ML algorithms can provide flexible modeling frameworks to predict complex output properties from input parameters with unprecedented fidelity.[7–9] When systematically probed and interpreted, supervised ML algorithms can yield insights into how input parameters and their interactions influence output properties.[10–12] In addition, active learning (AL) algorithms can be used to strategically select the 'next best experiment(s)', typically to improve model fidelity (thus bolstering knowledge) or to optimize output properties with respect to input parameters.[12–16] Complementing these algorithmic advances, robotic platforms and automated high-throughput characterization tools allow experiments to be performed rapidly, reproducibly, and in parallel, in effect, providing more data to resolve the connection between input parameters and output properties in a shorter period of time.[17–19] Furthermore, data collected when applying these tools allow for greater repeatability of experimentation through the reduction of human error when generating samples. Thus, mitigating uncontrolled variations captured by the model and when capturing experimental baselines.

Together, the aforementioned components can be linked together in a Design-Build-Test-Learn (DBTL) workflow,[20] as illustrated in Fig. 1. In a DBTL workflow, high throughput/automated machinery expedite the experimental preparation (build) and characterization (test) of samples, supervised ML enables modeling and projection of the collected data (learn), and AL leverages the learned information to select new

[a]Department of Biomedical Engineering, Rutgers, The State University of New Jersey, Piscataway, NJ 08854, USA. E-mail: adam.gormley@rutgers.edu

[b]Department of Chemical and Biological Engineering, Princeton University, Princeton, NJ 08544, USA. E-mail: mawebb@princeton.edu

† These authors contributed equally: A. P. M. and D. M. W.

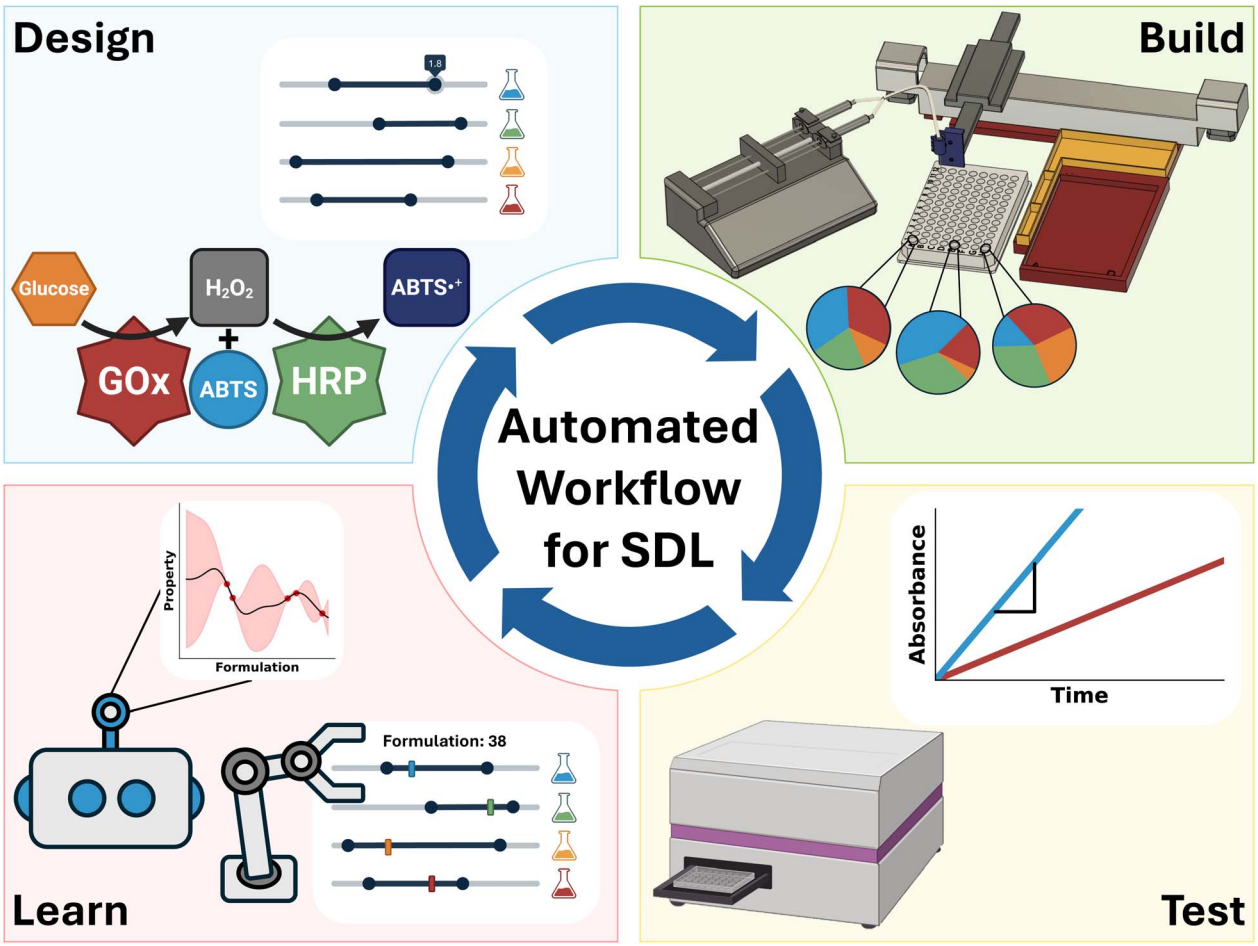

Fig. 1 Schematic of a self-driving laboratory (SDL) workflow based on a closed-loop design-build-test-learn paradigm for automated experimentation. Experiments are autonomously performed using a custom and low-cost liquid handler. The collected data, alongside design parameters, are fed into a machine learning pipeline that guides data acquisition with active learning principles. This framework can be used to understand system behavior across an input domain or to autonomously suggest and test parameter combinations to yield desired functionality.

experiments (design). This process can be repeated towards the advancement of defined scientific objectives. If well-integrated, such that the process can sustain without human intervention, the assembly of these components in a DBTL workflow forms a self-driving lab (SDL).[6,21] Distinguished by rapid and efficient operation, SDLs have the potential to revolutionize scientific discovery and otherwise influence how scientific research is conducted.

Despite their potential, implementing automated DBTL workflows can present substantial challenges that add methodological complexity with minimal immediate gain. Currently, the time-investment and expansive skill set required—which can span programming, robotics, and domain expertise—often drives researchers toward seeking commercial solutions for their bespoke objectives. Commercial turn-key systems offer reliable, out-of-the-box operation that alleviates these problems, which can be attractive to a user with sufficient capital. However, direct investment may be improbable for a researcher due to high initial cost. Companies, such as Opentrons, have helped lower entry barriers with more accessible (*i.e.*, low-cost and open-source) turn-key instruments,[22] but the investment in compatible devices and infrastructure still may exceed that of many research laboratories.[23] This creates a critical gap between the promise of SDL methodologies and their practical adoption in typical research settings.

To address aforementioned barriers related to skills and resources, there are growing efforts to democratize access to SDLs through open-source software and hardware.[22,24] For example, the Jubilee Project's extensible multi-tool motion platform demonstrates this idea through community built open-source hardware for automation applications.[23] Additionally, previous examples of low-cost open-source tools for prototyping SDLs (*e.g.*, SDL-light and claude-light) illustrate more community-driven approaches for automated experimentation.[25,26] In particular, wet lab SDLs based on liquid handling devices have demonstrated success across diverse applications.[6,27–32] Nevertheless, a significant gap remains between these demonstrations and the step-by-step educational resources that researchers need to successfully and timely implement SDL workflows.

Herein, we address both an educational and infrastructure gap by presenting a comprehensive guide for building and

implementing an open-source, low-cost liquid handling platform capable of SDL applications. Our approach combines hardware construction with software development, featuring Python-based control scripts and tutorials that researchers can customize for their specific needs. This provides a resource to bolster familiarity with AL/SDL concepts as well as a practical development tool, guiding users through the complete process from hardware assembly to crafting autonomous workflows. Additionally, we believe that the time investment and technical expertise gained through the utilization of our guides will embolden researchers to then tailor learned techniques towards their specific domain. This work represents the second installment for our User's Guide Series, building upon the foundational ML concepts covered in our first installment;[33] readers are referred to that installment for essential prerequisite knowledge and terminology. Through explanations of both theoretical principles and practical implementation steps, this guide aims to lower barriers for widespread adoption of SDL workflows across diverse research disciplines.

The rest of the document is structured as follows. Section 2 reviews essential ML/AL terminology and requisite background knowledge needed for understanding the guide. Section 3 provides instructional information for the creation and operation of low-cost liquid handlers for the purpose of automation. Section 4 provides a simple yet non-trivial demonstration of an autonomous experiment optimizing an enzyme assay, incorporating the developed liquid handling device and covered topics. Finally, Section 5 summarizes the key topics featured in this guide and provides perspective for future consideration. We note that supporting resources, including complete tutorials and build guides, are available at https://gormleylab.github.io/SDLGuide.

2 Essentials of active learning

2.1 Overview of active learning and Bayesian optimization

For an SDL to be self-driving, it must have the ability to autonomously propose and then pursue new experiments. The autonomous proposition can be accomplished using AL, which provides a principled framework based on ML and information theory to select experiments judiciously based on previously acquired data. In previous work, AL has been successfully used to develop accurate ML models with minimal training data or optimize outputs from a large input space using minimal evaluations.[12,34–37] Thus, by implementing this process to function iteratively, the SDL can efficiently acquire data without human intervention.

Applications of AL for SDLs follow a standard workflow, the components of which will be further described in subsequent sections. A schematic of this workflow is shown in Fig. 2. Initially, high-throughput preparation and characterization capabilities are used to perform a first set of measurements with conditions randomly selected or using another strategy. We refer to this initial data selection as data seeding. An ML model, termed the surrogate model, is trained on this data for target property prediction. The surrogate model then predicts the target property on the unmeasured portions of the input space. Then, new points are selected based on the predictions of the surrogate model. Specifically, an acquisition function is defined, typically using the outputs of the surrogate model, and new points are selected by optimizing the acquisition function. The SDL then obtains data for the recommended points, after which the surrogate model is retrained and the process repeats. This AL "loop" continues until an end criterion is met. Common criterion include exhausting an experimental resource or time budget, the plateauing of surrogate model accuracy, or the identification of a set of experimental conditions that yield a target property.

This standard AL workflow is commonly employed in two distinct contexts. First, AL can be used to select samples that best represent the variation of a property across the input space. In other words, AL is used to train a maximally accurate surrogate model for the prediction of the target property for any possible input. In this context, the acquisition function may rely on the uncertainty of the surrogate model. ML models can make predictions with uncertainty using a variety of the methods

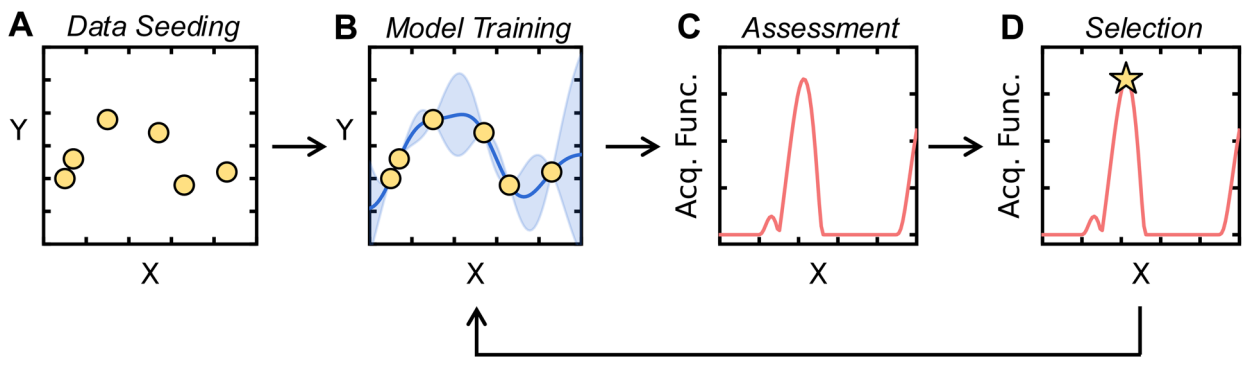

Fig. 2 Schematic of active learning framework. (A) Data are initially chosen using a data seeding method. Here, *X* is the range of inputs, *Y* are the property values, and the initial data are plotted as yellow markers. (B) A surrogate model is trained on this data and used to make predictions on the input space. The mean predictions of the surrogate model are plotted as a blue line, while the blue shaded region depicts the uncertainty of the surrogate model. (C) An acquisition function is computed from these predictions and uncertainties. Here, the acquisition function is plotted as a red line over the input space *X*. (D) The point which maximizes the acquisition function is chosen for further evaluation, as shown by the yellow star. After evaluating the new point, the surrogate model is retrained on the new dataset, completing the active learning loop.

discussed in Section 2.3. By choosing points for which the surrogate model is maximally uncertain, the SDL minimizes uncertainty in model predictions across the input space, resulting in a (hopefully) maximally accurate surrogate model.

Second, AL can be used to find points in an SDL's input space that optimize a target property using minimal experiments. This form of AL is often referred to as Bayesian optimization (BO). In this context, the acquisition function considers surrogate model predictions and uncertainties to select new points from the input space which are likely to improve over the current optimal input. Since SDLs are often employed for optimizing experimental conditions or discovering materials with optimal properties, BO is commonly used to guide autonomous data selection in SDLs. Therefore, we emphasize the use of BO in this User's Guide.

For the remainder of Section 2, we provide a more thorough discussion of the components of AL/BO. To provide readers with a hands-on demonstration of the methods and workflow described here, we have created a Google Colab notebook for performing BO on a toy function. Specifically, we show how BO can be applied in various ways to the optimization of the Müller–Brown potential, a two-dimensional function from theoretical chemistry commonly used to benchmark optimization protocols. All code required to recreate the results of Section 2 is available in the Google Colab.

2.2 Data seeding

Choosing a set of data to initiate AL/BO is called data seeding. The initial dataset may have been previously collected, or it may be obtained intentionally at the outset of an AL-guided campaign. If the latter, it is desirable to use a diverse set of initial points that can supply baseline coverage across the input space. To generate such diverse initial datasets, space-filling algorithms can be employed;[38] examples include Latin hypercube sampling (LHS)[39] and Sobol sampling[40] when variables on the domain are independent and bounded, while maximin[41] and cluster-based[42] samples can be employed for arbitrarily shaped domains. While many options exist, so long as the initial training set covers the input domain, most reasonable options should suffice, as the importance of initial training data selection is expected to decline as more data is acquired during AL.[37] Fig. 3 shows examples of popular data seeding algorithms applied to the Müller–Brown potential, as shown in the Google Colab.

In AL/BO workflows, it is necessary to decide what fraction of an experimental budget should be allocated to data seeding. There is no consensus on the best amount of data to use for seeding, and the ideal amount is likely problem dependent. As a result, seed dataset sizes are usually chosen based on data acquisition logistics. If data acquisition occurs sequentially, as is often the case for manual experimentation, the seed dataset could be a single measurement. If data acquisition occurs in batches, as is often the case for high-throughput experimental equipment, then a seed dataset size can be chosen for compatibility with experimental protocols (*e.g.*, a size of 96 for experiments conducted in 96-well plates). If one has the capacity to choose a seed data size independent of equipment constraints, then it can be decided by considering the AL/BO algorithm being used. If an exploitative BO algorithm is employed, it may be beneficial to allocate a larger portion of one's experimental budget to space-filling, suggesting a larger seed dataset. If a BO algorithm has an exploratory component, it may be beneficial to provide the algorithm more iterations for optimization, suggesting a smaller seed dataset. We emphasize that these are general considerations and that the relationship between seed data and AL/BO outcomes is not fully understood.[19]

2.3 Surrogate model training

Using the initial dataset, an ML model is trained and used to make predictions across the entire input space. While any ML model can be used for AL/BO, some common examples include Gaussian processes (GPs), random forests, and neural networks. GPs are the usual method employed for BO. For most

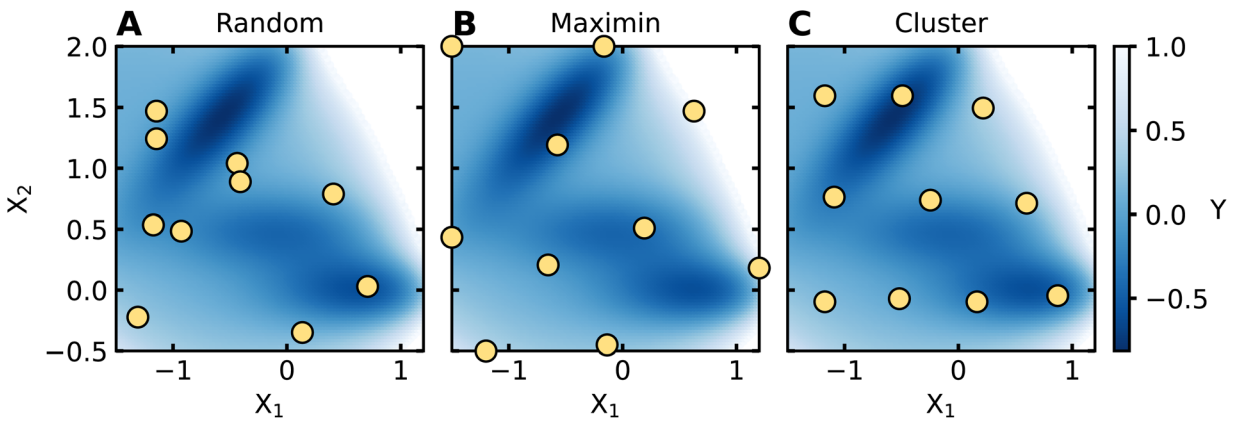

Fig. 3 Examples of space-filling algorithms. Common space-filling algorithms are applied to the two-dimensional Müller–Brown dataset. X_1 and X_2 are the features of the Müller–Brown potential, and the labels are shown using a blue color map, where darker regions denote smaller values. The chosen sample is shown by yellow points. The results of (A) random, (B) maximin, and (C) cluster-based sampling algorithms applied to the Müller–Brown dataset are shown.

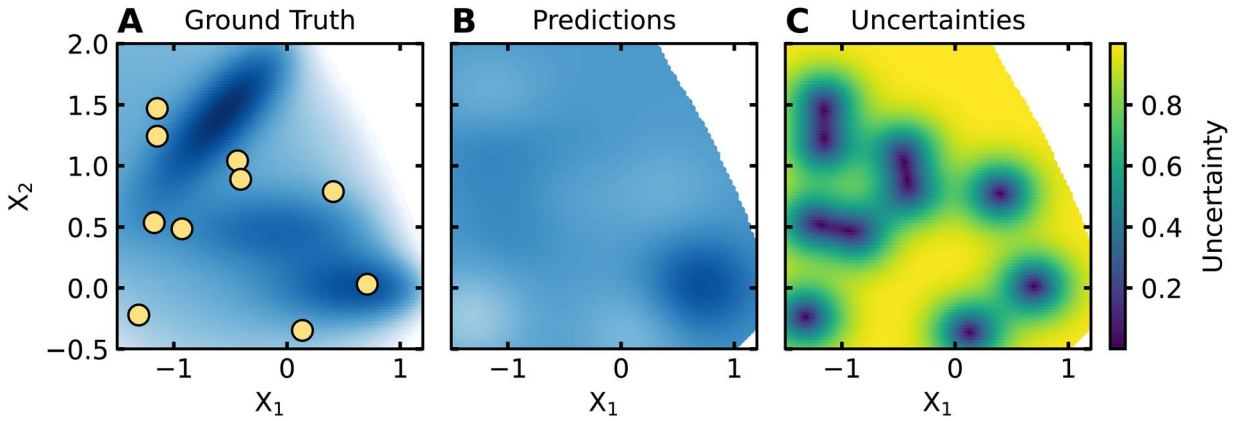

Fig. 4 Example of surrogate model outputs. A GP is trained on the dataset shown in (A). The predictions of this GP on the Müller–Brown domain are shown in (B), while the uncertainties of the GP on the Müller–Brown domain are shown in (C). The GP accurately predicts a local minimum in the bottom right corner of the domain. The GP has minimal uncertainty near the training data, but has high uncertainty far from the training data.

AL/BO workflows, surrogate models are used to make predictions with uncertainty (*i.e.*, quantifying a range of possible predictions), since surrogate model uncertainty is often used to guide the selection of new experiments. Fig. 4 shows a GP's predictions and uncertainties for the Müller–Brown dataset after being trained on the seed dataset shown in Fig. 3A.

ML models differ in how they compute uncertainties, which can impact AL/BO performance. GPs inherently measure uncertainty by predicting distributions of labels directly, which encourages their use in AL/BO workflows. Random forests, being ensembles of decision trees, typically compute uncertainty by measuring the spread in predictions from individual decision trees. For neural networks, uncertainty is often calculated by training several neural networks with different initial parameters in a process called ensembling. In principle, all ML models are capable of uncertainty estimation using ensembling or bootstrap aggregation, where uncertainties are calculated using the spread of predictions from several models trained on different subsets of the training data. Uncertainty quantification for ML models of varying architectures is an active area of research, and we refer interested readers to more complete treatments of uncertainty estimation.[43–47]

2.4 Acquiring new data

After the ML model is trained, new points in the input space are selected for evaluation based on the predictions and uncertainties of the ML model. A new candidate is selected if it maximizes an acquisition function. Acquisition functions usually depend on both predicted values and uncertainties of surrogate models. If researchers are interested in maximizing ML model accuracy across the domain, a suitable acquisition function may simply be the model uncertainty. In other scenarios, researchers are interested in finding the experimental inputs that maximize a property; numerous acquisition functions comport with this objective, with opportunity to balance both predicted values and uncertainties when choosing the next candidate from the domain. Researchers may also be interested in simultaneously recommending several points (*i.e.*, batch selection) rather than one point at a time (*i.e.*, sequential selection),[48] and acquisition functions can be devised or adapted for either context.

Numerous acquisition functions are used for BO. Fig. 5 shows the expected improvement (EI) acquisition function calculated for the predictions and uncertainties on the Müller–Brown domain shown in Fig. 4 and the point which maximizes it. The most common acquisition functions for BO include EI, probability of improvement (PI), and upper confidence bound (UCB). PI calculates the probability that a candidate will improve upon the best current result (sometimes referred to as the incumbent). EI extends this concept by also weighting probability of improvement by the magnitude of expected improvement; thus, the consideration evolves from asking simply whether something will be better to how much better it might be. UCB is calculated as a weighted sum of the prediction and uncertainty, thereby choosing points based on their likely maximum value. Variations on these acquisition functions can be formulated with a hyperparameter that modulates the balance of exploitation (*i.e.*, choosing points with predicted optimal values) or exploration (*i.e.*, choosing points with high uncertainty). Additional acquisition functions exist based on different considerations (*e.g.*, entropy search, knowledge gradient, Thompson sampling, *etc.*).[49] Likewise, ensembles of acquisition functions can be used to combine the intuitions of each approach; this has proven especially useful in batch selection settings.[50,51]

High-throughput experimental equipment developed for SDLs is often capable of making measurements in batches. Therefore, it is useful for AL/BO algorithms to recommend batches of points rather than individual measurements for data acquisition. A naïve approach to select batches of b measurements involves choosing the b measurements that optimize an acquisition function (*e.g.*, the b points with the highest EI). However, this approach does not consider correlation among proposed measurements, possibly leading to redundancy in acquired data. Alternative batch selection methods attempt to choose desirable measurements while reducing redundancy. Some methods sequentially choose batches of points while

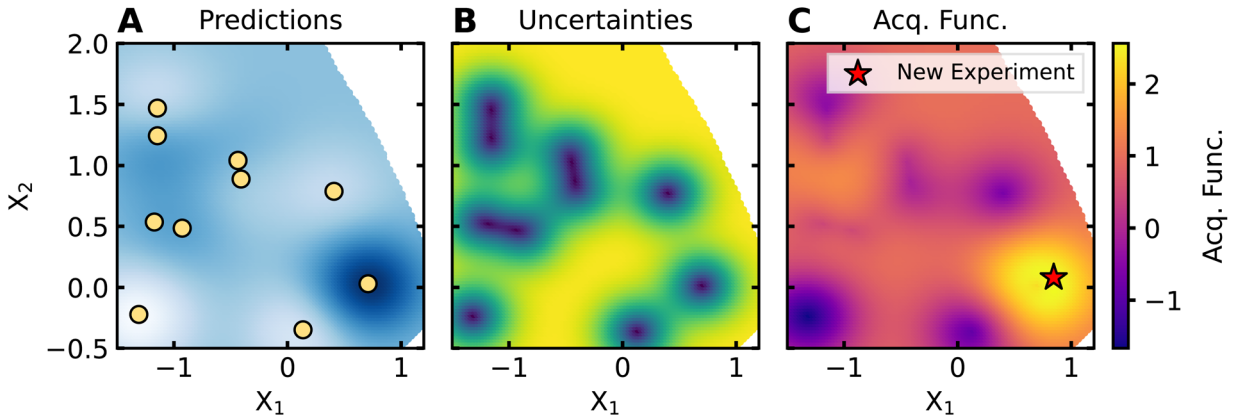

Fig. 5 Example of an acquisition function. The expected improvement acquisition function is computed for the predictions and uncertainties of the surrogate model. (A) The scaled predictions of the surrogate model. (B) The uncertainties of the surrogate model. (C) The expected improvement acquisition function. The proposed point which maximizes the acquisition function is shown as a red star.

discouraging selection near previously chosen points, such as local penalization[52] and Kriging believer batch selection.[48] Other methods rely on stochasticity to prevent similar measurement selection, like Thompson sampling and the multipoint probability of optimality.[53] Batch selection methods can also use filters, such as clustering algorithms[54] or diversity metrics,[55] to limit redundancy in the selected batch. Batch selection in AL/BO workflows is still an active area of research, and no single approach has proven optimal across a wide array of optimization problems.

2.5 Closing the loop

After a new point is recommended, the SDL prepares the necessary experiment, measures the result, and the process is repeated. Specifically, the ML model is retrained on the updated set of measurements, predictions and uncertainties are made on the domain, and the point that maximizes the updated acquisition function is chosen and measured by the SDL. This process is repeated until a stopping criterion is reached, whether that is exhaustion of the SDL budget, identification of an effective input, or sufficient convergence of model accuracy (as suitably measured by cross-validation or a held-out test set).

2.6 Successful applications of AL/BO for SDLs

The growing popularity of AL/BO for SDLs is supported by many successes. Tamasi *et al.* used BO (specifically, a GP with the EI acquisition function) to discover copolymers capable of successfully stabilizing three enzymes under thermal stress by measuring only ~0.1% of the total domain.[12] Angello *et al.* used BO (specifically, a novel BO formulation called Gryffin[56]) to not only optimize the photostability of light-harvesting donor–acceptor molecules in solution sampling only 1.5% of the total domain, but they also gained chemical "knowledge" by identifying previously unknown molecular features correlated with their target property during their BO campaign.[57] Szymanski *et al.* used an AL-driven SDL to explore the domain of synthetically accessible inorganic powders, integrating computation, theory, AL, and robotics to propose synthesis recipes and search for novel compounds.[31] These are just a few examples of a growing effort to use AL/BO and SDLs to accelerate scientific discovery.[58–62] We anticipate that continued research into maximally data-efficient AL/BO protocols will only further increase the successes of SDLs.

3 An open-source SDL platform build

While Section 2 introduced readers to the essential facets of AL, the goal of this section is to deploy these methods in a simple, low-cost, and open-source platform for self-driving wet lab experimentation. Fig. 6 shows how the principles of AL/BO can be incorporated into an automatic experimental platform to achieve this goal. We view our platform as an accessible and customizable alternative to commercial devices such as the Opentrons OT-2 robot. Inspired by SDL-Light and Jubilee,[23,25] we provide an SDL build guide that provides two lower cost and open-source alternatives to commercially available liquid handlers. By developing this guide, we also hope to encourage the use of automatic liquid handlers in experimental workflows and provide a starting point for the iterative refinement of low-cost, automated laboratory equipment.

3.1 Custom-built liquid handlers

Automated liquid handlers rely on three systems: the spatial system, fluidic system, and control system. The spatial system includes mechanical parts capable of movement, the fluidic system includes equipment capable of drawing and dispensing fluid, and the control system includes the software necessary to coordinate the spatial and fluidic systems to execute user-defined tasks. Based on these systems, we prototyped two liquid handling platforms which offer lower cost alternatives to commercial instruments. Fig. 7 shows these platforms, referred to as the pen plotter- and pipette-driven liquid handlers, which differ in their spatial and fluidic systems. The platforms strike a balance between dispensing accuracy, precision, and speed, while being built for less than $1000 (excluding the cost of the syringe pump, electronic pipette, and UV-vis spectrophotometer, which are common in most labs).

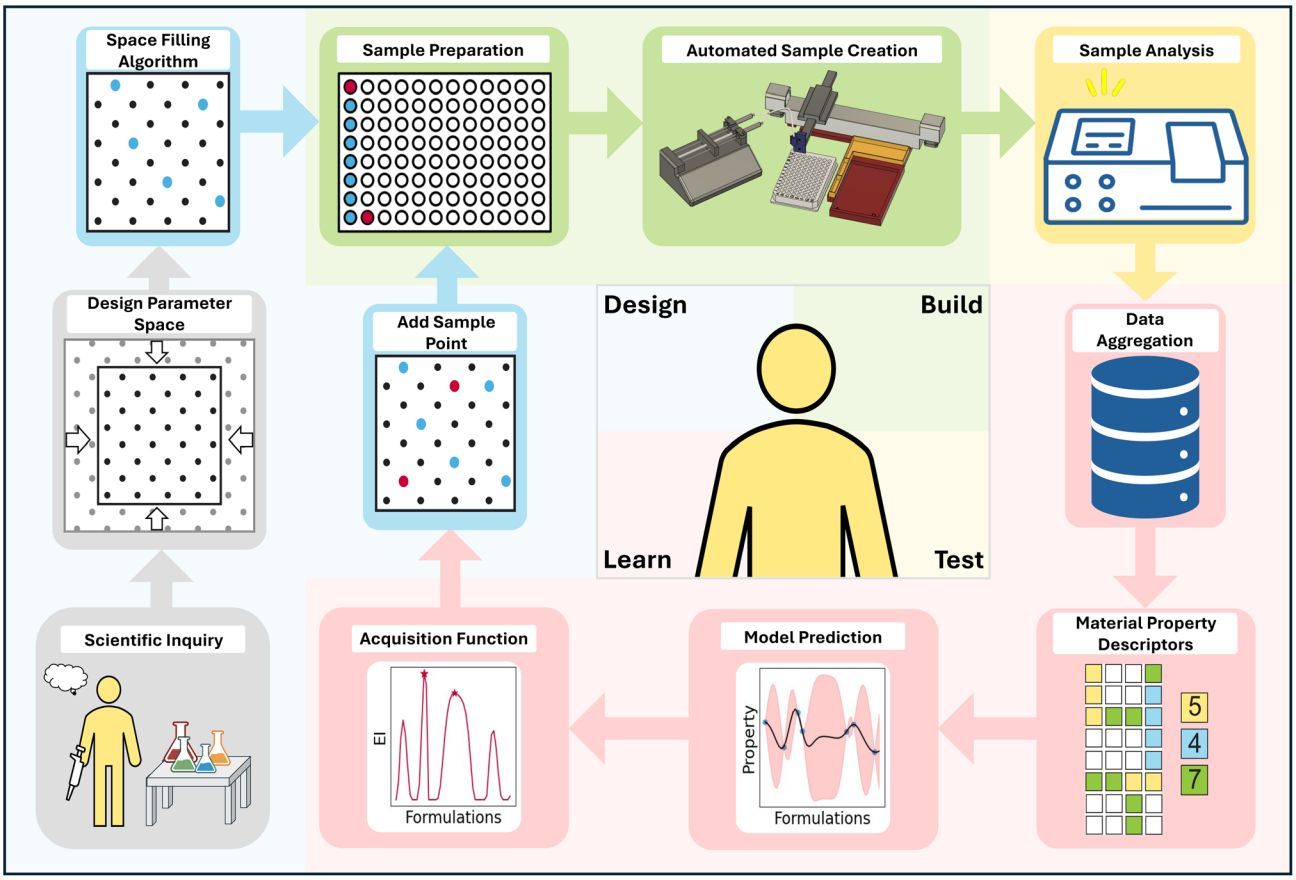

Fig. 6 Schematic overview of an SDL. Once an experimental question is generated and the researcher designs a parameter space of relevant resources, the task is sent through the automated platform to iteratively uncover the function–property relationships in the domain. The automated platform uses a designated space filling algorithm, prepares and analyzes chosen samples, aggregates the data into a ML friendly format, and performs BO by applying a chosen ML model and acquisition function to suggest the next samples.

The pen plotter driven liquid handler was created by integrating a Fusion 200X syringe pump by ChemyX Inc., an AxiDraw V3 pen plotter by Evil Mad Scientist (now sold by Bantam Tools as NextDraw), various fluidic components obtained from Chrom-Tech, a supplier of IDEX Health and Science fluidic components, and 3D-printed parts designed and printed using a Bambu Lab X1C 3D printer. The pen plotter was used as a controllable, three-axis, pre-built spatial system for precise movement. The Fusion 200X syringe pump was chosen as the basis of our fluidic system for its precision bulk withdrawal and infusion at large volumes. Due to the commercial availability and well documented APIs of the discussed devices, it is simple to build a programmable control system. The platform is designed to work with a SpectraMax M2 UV-vis spectrophotometer plate reader (*i.e.*, from Molecular Devices Inc.) to collect absorbance data from microplates, as can be seen in Fig. S2. A 3D printed platform is designed to fit auxiliary labware such as reagent holders, waste, and cleaning solution with designated space opening and closing the plate reader drawer. This allows for transfer of materials from the reagent holders to the sampling plate inserted into the plate reader. A demonstration of this system running can be seen in Video S1.

To provide an alternative to the pen plotter-driven liquid handler that may serve different needs within a laboratory environment, we provide a build guide for a pipette-driven liquid handler. Here, the fluidic system is comprised solely of an electronic pipette (VIAFLO, Integra Biosciences) which individually withdraws and dispenses in small volumes with the commercially determined precision. The fluidic system is mounted on a custom gantry that acts as the spatial system, as shown in Fig. 7. With carriages along the *X*/*Y*-axes and a *Z*-axis linear actuator, the spatial system provides ample space for greater customizability. However, a specific user defined task dictates the design of a robust liquid handler, ultimately requiring careful planning and iterative improvement. A demonstration of the liquid handler running can be viewed in Video S2. A component breakdown and build guide for creating this specific handler is provided in a GitHub repository at https://github.com/GormleyLab/Pipette-Liquid-Handler.

3.2 Pen plotter liquid handler

For the remainder of this user guide, we consider the pen plotter liquid handler. In addition to the build guide, we provide an extended explanation of the low-level decisions related to its operation and production.

The integration of a finely tuned fluidic system becomes the major focus in a liquid handler using a prebuilt spatial system.

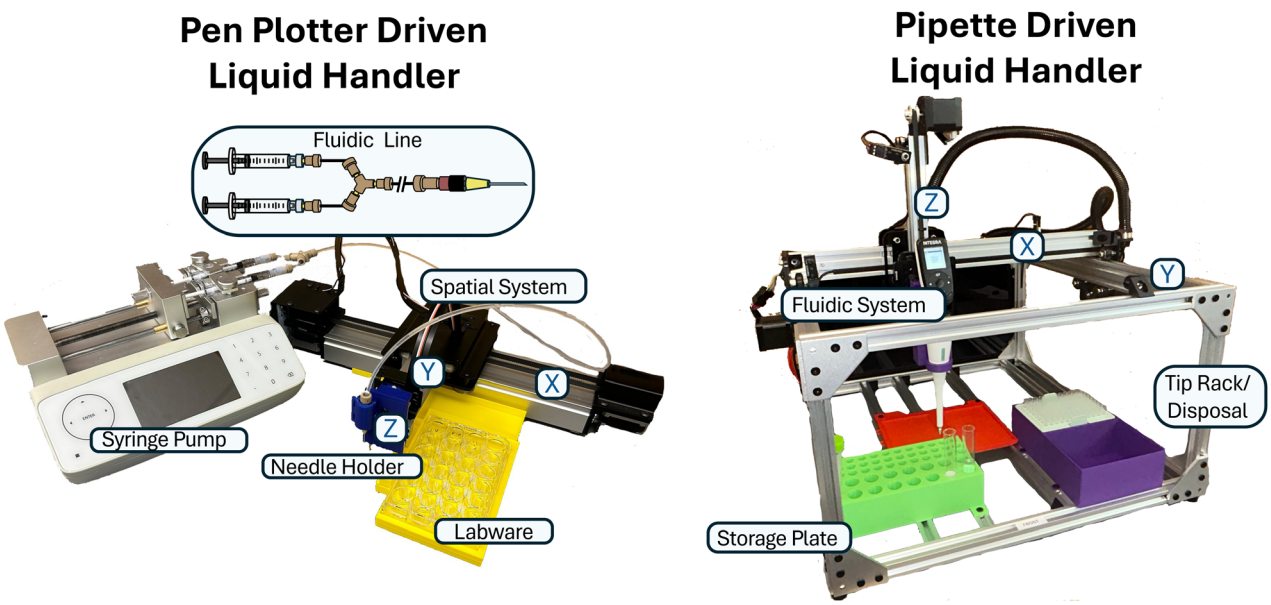

Fig. 7 Completed examples of compatible low-cost liquid handler platforms for an SDL (pen plotter driven system described in length in this paper). Includes both fluidic schematics, and spatial movement systems. The pen plotter driven liquid handler's spatial system is comprised of a modified pen plotter while the fluidic system is comprised of a user-developed fluidic line driven by a controllable syringe pump. The pipette driven liquid handler's spatial system is comprised of a user-developed gantry while the fluidic system is comprised of a wirelessly controlled electronic pipette. Build guides for both liquid handlers including parts lists, CAD files, and software can be found in GitHub at https://gormleylab.github.io/SDLGuide.

A fluidic line was developed to produce the desired connection between the syringe pump and pen plotter to aspirate and dispense reagents. The developed configuration utilizes two parallel 1 mL syringes clamped into the syringe pump joined by a Y-assembly to a single tubing line ending with a needle assembly. The needle assembly is placed in a 3D printed holder on the *z*-axis linear actuator for withdrawal and dispensing by the pen plotter (Fig. 7). Two 1 mL syringes were chosen to leverage a smaller internal diameter to increase the reliability of dispensing at specified flow rates. However, larger syringes can be used for an increased storage capacity, albeit at the cost of lower dispensing accuracy and precision.

3.2.1 Preliminary setup. Here, we review important aspects related to the setup of the pen plotter liquid handler.

3.2.1.1 Fluidic line priming. Before the fluidic line can be used in the fluidic system, the line must be prepared. This is done by filling the line and syringes with an incompressible liquid (*i.e.* water), taking care to mitigate the number of air bubbles in the system. If the fluidic line exceeds the amount of volume present in the syringes, it may be advantageous to fully submerge the syringe in a container of the incompressible liquid with its connecting piece to fill the line without introducing unnecessary air bubbles. The loaded fluid transmits the forces generated by the syringe pump within the system to execute dispensing orders consistently. To eliminate interactions with the liquid already in the system, a deliberate air gap is formed to separate drawn fluids.

3.2.1.2 Syringe pump fluidic calibration. Due to inaccuracies inherent in a pressure-driven syringe pump and variation in syringe sizes, it is necessary to scale the user-submitted volumes when dispensing. A scaling factor is determined by performing a calibration, which involves computing the ratio between actual and user-specified volumes for triplicate dispenses. When user-specified volumes are scaled by this factor, the liquid fluidic system dispenses volumes more accurately.

3.2.2 Fluidic capabilities. The pen plotter liquid handler was designed to maximize accuracy and precision over the largest range of dispensing volumes possible while satisfying cost constraints, as described in Fig. S1. However, there is a clear lower limit for accurate volume transfers. At low volumes, it is measured to have higher dispensing error, likely due to start up effects of the syringe pump. This start up effect is present over the entire dispensing range as there is a consistent loss in energy due to compression at the air gap separating the incompressible fluid and the drawn reagent. However, this effect is remedied at higher volumes using the defined correcting factor.

3.3 Program

The software that provides programmable control of the liquid handler is key to developing automated workflows. The software must consider limitations of the spatial and fluidic systems to execute experiments of interest. Our software is built to work with the AxiDraw V3 pen plotter (now sold by Bantam Tools as NextDraw), Fusion 200X ChemyX syringe pump, and SpectraMax M2. The software was built using Python 3.12.0 and requires the ChemyX API, AxiDraw API and Molecular Devices SDK, as well as several packages for data analysis and interpretation like NumPy and Pandas. The program is organized into a modular structure to support flexibility and maintainability. At the top level, a `requirements.txt` file defines all necessary dependencies installed in a virtual

environment. The application is launched *via* `main.py`. The core functionality resides in the `src` directory, which includes five modules. The `sdlgui.py` module contains all tkinter-based UI elements, including buttons and entry boxes used to configure experiments. The `guifunctions.py` module is responsible for processing, organizing, and manipulating user input, and contains the functions that execute the experimental workflow. The `liquidhandler.py` module controls the AxiDraw V3 and Fusion 200X syringe pump used for automated liquid handling. The `dataprocessing.py` module handles data acquisition from the SpectraMax M2 and implements the ML and BO algorithms that enable self-driving functionality. Finally, `sdlvariables.py` stores shared global variables such as directory paths and pandas DataFrames that are accessed and modified across the modules. The project also includes a data directory for storing raw and processed experimental data and a resources directory that contains static assets such as logos and icons used in the user interface.

When running the program, we provide a user interface that allows for the manual setup of experimental parameters. Configuring experiments begins in the 'home' tab, where a list of reagents is provided with corresponding concentrations. The order of the reagents determines the order of dispensing. When reagents are put into groups, the program selects one reagent from each group. Reagents that act as buffers must be selected as buffers in the dropdown menu and include a corresponding pH. Additional tabs can be added for different experiments. Currently, the enzyme assay tab is the only self-driven task that can be chosen. Under the enzyme assay tab, reagent bounds can be chosen in the format "([Lower Bound], [Upper Bound])". The number of samples is the seed library size and must be chosen carefully to ensure the dispensing maximum is not exceeded. Finally, the desired enzymatic activity value is selected as the single-objective optimization target for the AL/BO pipeline. By providing example software, we hope to provide a starting point which users can add to or modify for their own experimental procedures.

4 SDL demonstration: enzyme assay optimization

In this section, we provide a representative example of our SDL in operation. Specifically, we apply the SDL pipeline to the optimization of an enzyme assay, a common objective in wet labs that can be challenging due to its dependence on many input parameters. We chose glucose oxidase (GOx) as the enzyme assay of interest due to its common use, colorimetric nature, and ability to integrate with our designed system. Within the AL framework, reagent concentrations are inputs, and enzymatic activity is the target property we seek to maximize. We discuss the development process step-by-step as a tool for improved understanding of the SDL pipeline.

4.1 Design

The GOx assay involves a two-step enzymatic reaction. The first step involves the oxidation of glucose to hydrogen peroxide, which is catalyzed by GOx. The second step involves the oxidation of 2,2′-azino-bis(3-ethylbenzothiazoline-6-sulfonic acid) (ABTS), which is catalyzed by horseradish peroxidase (HRP). Our target is a specified enzymatic activity of GOx, which can be measured by colorimetric analysis of the oxidized ABTS using a spectrophotometer.

To optimize the GOx assay, we must determine the viable space of input parameters. The input variables consist of concentrations of assay reagents, which includes GOx, HRP, glucose, and ABTS concentrations. Allowable ranges for these inputs must be chosen according to domain knowledge; for example, since our goal is to measure the enzymatic activity of GOx, it is necessary that HRP has a high enough concentration so that it is not the limiting reactant. We choose a GOx concentration range of 2.6 to 13 nM, while HRP has a concentration of 22.7 to 114 nM. ABTS and glucose were set at concentration ranges of 168.2 to 841 μM and 69.4 to 347 mM, respectively. Once we determine our input variable ranges, we generate a library of possible combinations of input variables. From this library, we employ the LHS space-filling algorithm (as discussed in Section 2.2) to generate the seed dataset. This results in an initial set of input parameters with a large coverage of the input space that can be used to initiate AL/BO.

4.2 Build and test

After determining a set of reagent concentrations to measure, we use the pen plotter liquid handler to measure enzyme activity. Determining a viable and reproducible protocol for running the enzyme assay on the pen plotter liquid handler requires substantial effort. For example, to measure enzymatic activity, we require kinetic sampling of the initial rate of change of absorbance (ΔOD) produced by the oxidation of ABTS to yield a blue-green product. Therefore, order of reagent addition is important; the reagent which initiates the reaction is added last to reduce time variance between wells. In this case, glucose drives the initial enzymatic event of the assay, so it is added last. Once the experimental procedure is determined, the program calculates and formats the dispensing protocol for the selected inputs, which is then executed autonomously by the pen plotter liquid handler. Leveraging automation minimizes human labor and reduces the potential for experimental errors.

Enzyme activities are measured by the SDL using an absorbance read method previously created using external Softmax software. We choose an initial seed dataset of sixteen reaction conditions, which was the maximum number of samples that our fluidic system could run in a single synthesis step. After data seeding, we acquire data in batches of six measurements, which was chosen as a reasonable compromise between time and resource consumption for our system. To provide the software enough time to collect kinetic data for a 96-well plate, the sampling rate is standardized for all iterations. We test samples using a 405 nm wavelength in fifteen-second intervals for two and a half minutes, permitting the measurement of (ΔOD) at standard time points. By using the pen plotter liquid handler described in Section 3, we can directly translate generated experimental designs into fully executed and characterized assays, providing high-quality data to train the surrogate model.

4.3 Learn

After measuring enzyme activities for the reagent concentrations chosen in the seed dataset, we train a surrogate model on the available data. We choose a GP model as the surrogate. We choose the GP kernel to be a summation of the Dot Product, RBF, and White kernels. We choose the Dot Product and RBF kernels due to their popularity; we choose the White kernel to capture noise in our data originating from formulation and testing. Kernel hyperparameters are chosen by performing a grid search, and we choose hyperparameters which maximizes R^2 calculated using five-fold cross-validation. After training our GP surrogate model, we compute the EI acquisition function and identify the reagent concentrations which maximize EI. These reagent concentrations are identified, measured by the liquid handler, and the appropriate data is added to the dataset. This loop repeats until our stopping criterion is reached; here, we stop after four rounds of BO.

Fig. 8A provides a summary of BO performance, reporting both the distribution of measured enzyme activities and prediction accuracies of surrogate models for each generation (*i.e.*, iteration). Fig. 8A shows a clear increase in the distribution of enzyme activities from round-to-round, with the primary increases occurring at rounds one and two. Such a distribution shift in the enzymatic activity of the seed data and later generations is indicative of a successful BO campaign; the SDL effectively biases recommendations in regions likely to maximize enzyme activity.

We also consider how surrogate model accuracy varies across the five rounds of data acquisition. Fig. 8A shows the cross-validation R^2 score for the GP model for each generation. Initially, the surrogate model has relatively high predictive power, with an $R^2 = 0.65$. The surrogate model experiences a decrease in accuracy after the first round of active learning, with an $R^2 = 0.33$, but increases in accuracy for all subsequent generations, with a maximum accuracy of $R^2 = 0.80$ by the final round. To better understand model accuracy at each generation, we analyze parity plots of predicted *vs.* measured enzymatic activity at each iteration, which we show in 8C. At early generations, the model struggles to make accurate predictions

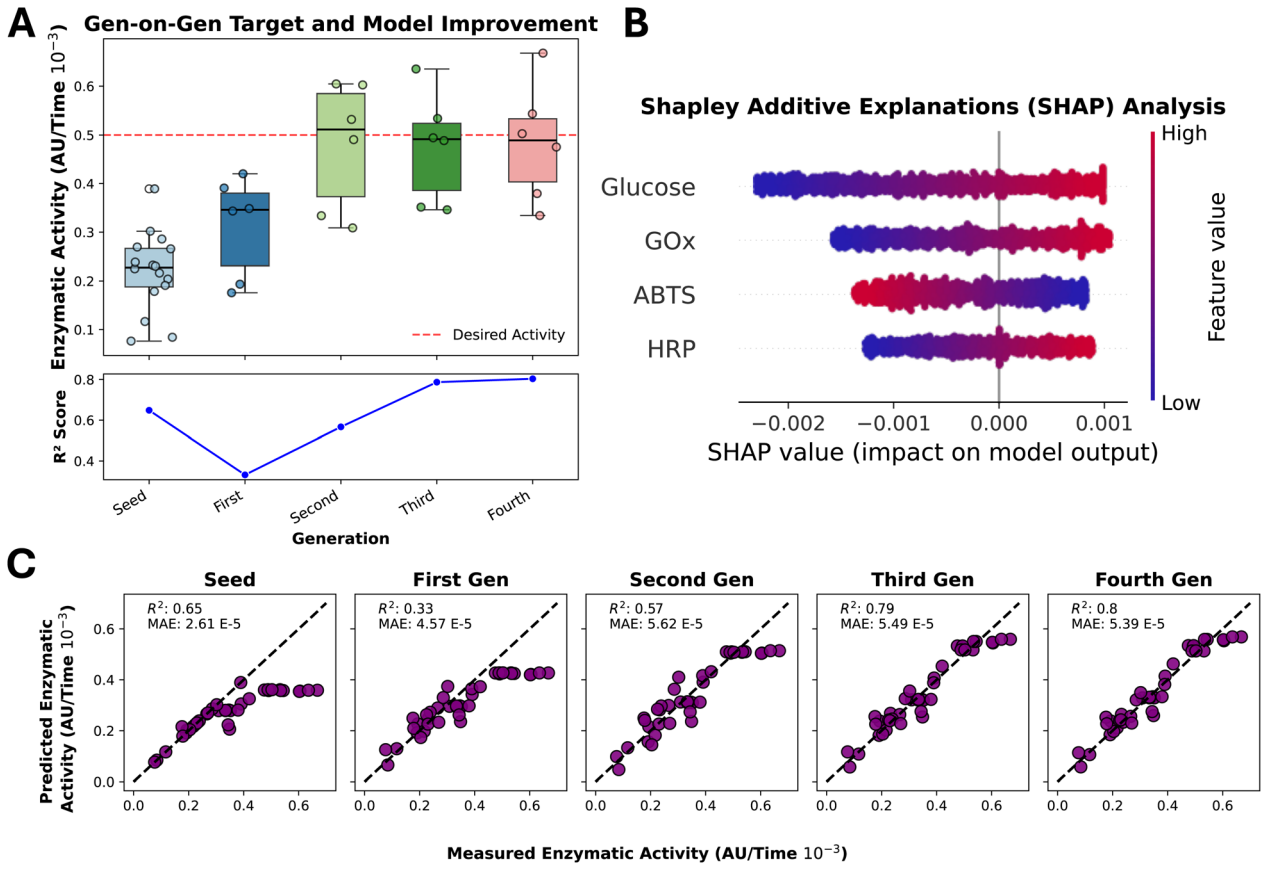

Fig. 8 AL campaign performance and feature influence (A) formulation performance with respect to target enzymatic activity per generation overlaid with model performance per generation. Each formulation activity value is shown on a box and whisker plot over each iteration of the DBTL loop, with the target formulation value designated by a red dashed line. Performance per generation is shown by the line plot (Blue) of the quantitative metric of R^2. (B) Shapley additive explanations (SHAP) analysis plot of GOx assay GPR model. The relative impact of each feature on model output (SHAP value) is displayed across the *x*-axis, and the relative magnitude of the feature value for each data point is color-coded across a gradient (red = high value, blue = low value). (C) Model performance displayed over each subsequent generation of the DBTL loop. Each diagonal black line shows a "perfect" model's prediction of the target feature, with values below the line representing an underprediction of the actual enzyme activity and values above the line representing an overprediction. Quantitative metrics of model performance (Gaussian process regressor R^2 score and MAE) are listed in the top left corner of each graph.

on reagent concentrations that result in high enzymatic activity. As the generations continue, high-performing predictions fall closer to the parity line and model accuracy increases. These results suggest that at early generations, model error is high due to inaccurate predictions for high-performing candidates. Model accuracy initially decreases as more high-performing candidates are added to the training set in the first round of BO. As the dataset size increases, the model improves its accuracy when predicting high-performing candidates. The overall increase in model accuracy and increased identification of high-performing reagent concentrations indicates a successful BO scheme.

An accurate surrogate model can be used to identify how changes in input parameters produce changes in experimental outcomes. Specifically, explainable AI (xAI) methods, such as Shapley Additive explanations (SHAP), provide a way to quantify the contribution of each reagent to model predictions of enzyme activity. In situations where the relationship between reagent and output is well-established, such analysis can assess whether the model has accurately captured the expected physical phenomena. 8B shows the results of applying SHAP analysis to our final surrogate model. The top two rows of 8B indicate that, according to the surrogate model, higher values of GOx and glucose concentrations contribute to increases in predicted enzymatic activity. This agrees with our physical intuition, since increasing concentrations of catalyst and substrate should increase reaction rates. The third row shows that higher concentrations of ABTS result in reductions in predicted enzyme activity; this agrees with prior work which shows that ABTS can act as an agonist at higher concentrations. The fourth row shows that concentrations of HRP have the least impact on predicted enzyme activities. Since HRP concentrations were specifically chosen to prevent HRP from being a limiting reactant, it is sensible that HRP concentrations would have a negligible impact on the resulting GOx assay. The results in 8B indicate that our model has accurately identified the expected physical relationship between reagent concentrations and enzymatic activity.

Due to the limitations of the device allowing for a maximum of 96 samples per experiment, it requires a manageable problem to solve before exhausting the resources of the system. The BO scheme was able to solve and understand the mechanisms of the assay within four generations of DBTL cycling after detecting sufficient convergence of model performance ($R^2 \geq 0.8$ and no significant improvement from previous generation). The optimization of GOx to achieve a certain enzymatic activity through iterative testing validates the SDL and BO scheme to educate and learn about SDL workflows.

5 Summary and perspectives

The goal of this perspective is to provide readers with a foundational understanding of AL principles in the context of developing self-driving systems capable of performing iterative experimentation. Our described system provides an open-source, customizable platform for automated liquid handling, reducing costs while maintaining precision and scalability for small-to medium-scale laboratory workflows. Through a representative study, we validated our low-cost automated liquid handler platform, providing readers with the tools and hardware necessary to apply SDLs to their own areas of expertise. We believe that the development of tutorials describing low-level component analysis of automation platforms and programming tasks shows noteworthy progress in the goal of democratizing user-developed automation infrastructure. By focusing on increased accessibility, there is a greater opportunity for accelerated adoption of SDLs as technical and financial barriers to entry continue to fall.

Author contributions

A. P. M. and D. M. W. designed and validated low-cost liquid handler systems with build guides, designed and implemented user interface and program for liquid handling systems and wrote tutorial notebook 2. Q. M. G. and R. A. P. created tutorial notebook 1. A. P. M., D. M. W., Q. M. G., and R. A. P. wrote the initial draft. All authors revised the manuscript. M. A. W. and A. J. G. conceived the project and supervised the work. All authors contributed and approved the manuscript.

Conflicts of interest

The authors declare no competing interests.

Data availability

The datasets and code generated during this study, including experimental logs, build guides, device-control code, and Jupyter notebooks, are openly available *via* our GitHub repository (https://github.com/gormleylab/SDLGuide). A citable, archived version is available on Zenodo (https://doi.org/10.5281/zenodo.19190906). An easy-to-navigate project landing page can be found at https://gormleylab.github.io/SDLGuide/.

Supplementary information (SI) is available. See DOI: https://doi.org/10.1039/d5dd00525f.

Acknowledgements

This work was funded by the National Institutes of Health (NIH), National Institute of Biomedical Imaging and Bioengineering (NIBIB) (R01EB037022), National Institute of General Medical Sciences (NIGMS) (R35GM138296), and the National Science Foundation Division of Materials Research (DMREF-2118860, DMREF-2118861, and 2237470). This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-2039656. We would like to thank C. Matty, G. Palahnuk, and the members of the GormleyLab for their indispensable contributions to the instruction and guidance used to create the systems and guides discussed in this paper.

References

1 E. L. Lehmann, *Fisher, Neyman, and the Creation of Classical Statistics*, Springer New York, New York, NY, 2011, pp. 63–76, DOI: 10.1007/978-1-4419-9500-1_5.

2 B. Durakovic, *Design of experiments application, concepts, examples: State of the art*, Periodicals of Engineering and Natural Sciences, 2017, 5, pp. 421–439, DOI: 10.21533/pen.v5i3.145.

3 A. I. Khuri and S. Mukhopadhyay, Response surface methodology, *Wiley Interdiscip. Rev. Comput. Stat.*, 2010, **2**, 128–149.

4 K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev and A. Walsh, Machine learning for molecular and materials science, *Nature*, 2018, **559**, 547–555, DOI: 10.1038/s41586-018-0337-2.

5 R. Vasudevan, G. Pilania and P. V. Balachandran, Machine learning for materials design and discovery, *J. Appl. Phys.*, 2021, **129**, 070401, DOI: 10.1063/5.0043300.

6 G. Tom, *et al.*, Self-Driving Laboratories for Chemistry and Materials Science, *Chem. Rev.*, 2024, **124**, 9633–9732, DOI: 10.1021/acs.chemrev.4c00055.

7 G. Pilania, C. Wang, X. Jiang, S. Rajasekaran and R. Ramprasad, Accelerating materials property predictions using machine learning, *Sci. Rep.*, 2013, **3**, 2810.

8 A. Agrawal and A. Choudhary, Deep materials informatics: Applications of deep learning in materials science, *MRS Commun.*, 2019, **9**, 779–792.

9 S. Chibani and F.-X. Coudert, Machine learning approaches for the prediction of materials properties, *APL Mater.*, 2020, **8**, 080701.

10 F. Oviedo, J. L. Ferres, T. Buonassisi and K. T. Butler, Interpretable and Explainable Machine Learning for Materials Science and Chemistry, *Acc. Mater. Res.*, 2022, **3**, 597–607, DOI: 10.1021/accountsmr.1c00244.

11 M. Umehara, H. S. Stein, D. Guevarra, P. F. Newhouse, D. A. Boyd and J. M. Gregoire, Analyzing machine learning models to accelerate generation of fundamental materials insights, *npj Comput. Mater.*, 2019, **5**(1), 34, DOI: 10.1038/s41524-019-0172-5.

12 M. J. Tamasi, R. A. Patel, C. H. Borca, S. Kosuri, H. Mugnier, R. Upadhya, N. S. Murthy, M. A. Webb and A. J. Gormley, Machine Learning on a Robotic Platform for the Design of Polymer–Protein Hybrids, *Adv. Mater.*, 2022, **34**, e2201809.

13 B. Shahriari, K. Swersky, Z. Wang, R. P. Adams and N. De Freitas, Taking the human out of the loop: A review of Bayesian optimization, *Proc. IEEE*, 2015, **104**, 148–175.

14 B. Settles, *Active Learning; Synthesis lectures on artificial intelligence and machine learning*, Springer International Publishing, Cham, 2012.

15 Y. An, M. A. Webb and W. M. Jacobs, Active learning of the thermodynamics-dynamics trade-off in protein condensates, *Sci. Adv.*, 2024, **10**, eadj2448, DOI: 10.1126/sciadv.adj2448.

16 R. A. Patel, S. S. Kesharwani and F. Ibrahim, Active learning and Gaussian processes for the development of dissolution models: An AI-based data-efficient approach, *J. Controlled Release*, 2025, **379**, 316–326, DOI: 10.1016/j.jconrel.2025.01.003.

17 M. Tamasi, S. Kosuri, J. DiStefano, R. Chapman and A. J. Gormley, Automation of controlled/living radical polymerization, *Adv. Intell. Syst.*, 2020, **2**, 1900126.

18 R. Upadhya, S. Kosuri, M. Tamasi, T. A. Meyer, S. Atta, M. A. Webb and A. J. Gormley, Automation and data-driven design of polymer therapeutics, *Adv. Drug Delivery Rev.*, 2021, **171**, 1–28, DOI: 10.1016/j.addr.2020.11.009.

19 R. A. Patel and M. A. Webb, Data-driven design of polymer-based biomaterials: High-throughput simulation, experimentation, and machine learning, *ACS Appl. Bio Mater.*, 2024, **7**, 510–527.

20 M. J. Tamasi and A. J. Gormley, Biologic formulation in a self-driving biomaterials lab, *Cell Rep. Phys. Sci.*, 2022, **3**, 101041.

21 M. Abolhasani and E. Kumacheva, The rise of self-driving labs in chemical and materials sciences, *Nat. Synth.*, 2023, **2**, 483–492, DOI: 10.1038/s44160-022-00231-0.

22 G. Moukarzel, Y. Wang, W. Xin, C. Hofmann, A. Joshi, J. W. Loughney and A. Bowman, Automation of biochemical assays using an open-sourced, inexpensive robotic liquid handler, *SLAS Technol.*, 2024, **29**, 100205, DOI: 10.1016/j.slast.2024.100205.

23 K. Dunn, C. Feng and N. Peek, Jubilee: A Case Study of Distributed Manufacturing in an Open Source Hardware Project, *J. Open Hardw.*, 2023, **7**, 4, DOI: 10.5334/joh.51.

24 B. Pelkie, *et al.*, Democratizing self-driving labs through user-developed automation infrastructure, https://chemrxiv.org/engage/chemrxiv/article-details/67a4ffb6fa469535b94a3ad9.

25 S. G. Baird and T. D. Sparks, What is a minimal working example for a self-driving laboratory?, *Matter*, 2022, **5**, 4170–4178, DOI: 10.1016/j.matt.2022.11.007.

26 J. R. Kitchin, The evolving role of programming and LLMs in the development of self-driving laboratories, *APL Mach. Learn.*, 2025, **3**, 026111–026122, DOI: 10.1063/5.0266757/20506556/026111_1_5.0266757.pdf.

27 F. Kong, L. Yuan, Y. F. Zheng and W. Chen, Automatic Liquid Handling for Life Science: A Critical Review of the Current State of the Art, *J. Lab. Autom.*, 2012, **17**, 169–185, DOI: 10.1177/2211068211435302.

28 Q. Zhu, *et al.*, An all-round AI-Chemist with a scientific mind, *Natl. Sci. Rev.*, 2022, **9**, nwac190.

29 A. Adamo, R. L. Beingessner, M. Behnam, J. Chen, T. F. Jamison, K. F. Jensen, J.-C. M. Monbaliu, A. S. Myerson, E. M. Revalor, D. R. Snead, T. Stelzer, N. Weeranoppanant, S. Y. Wong and P. Zhang, On-demand continuous-flow production of pharmaceuticals in a compact, reconfigurable system, *Science*, 2016, **352**, 61–67.

30 H. Quinn, G. A. Robben, Z. Zheng, A. L. Gardner, J. G. Werner and K. A. Brown, PANDA: a self-driving lab for studying electrodeposited polymer films, *Mater. Horiz.*, 2024, **11**, 5331–5534.

31 N. J. Szymanski, *et al.*, An autonomous laboratory for the accelerated synthesis of novel materials, *Nature*, 2023, **624**, 86–91, DOI: 10.1038/s41586-023-06734-w.
32 C. W. Coley, *et al.*, A robotic platform for flow synthesis of organic compounds informed by AI planning, *Science*, 2019, **365**, 6453–6557.
33 T. A. Meyer, C. Ramirez, M. J. Tamasi and A. J. Gormley, A User's Guide to Machine Learning for Polymeric Biomaterials, *ACS Polym. Au*, 2023, **3**, 141–157, DOI: 10.1021/acspolymersau.2c00037.
34 D. A. Cohn, Z. Ghahramani and M. I. Jordan, Active learning with statistical models, *J. Artif. Intell. Res.*, 1996, **4**, 129–145.
35 K. Shmilovich, R. A. Mansbach, H. Sidky, O. E. Dunne, S. S. Panda, J. D. Tovar and A. L. Ferguson, Discovery of self-assembling π -conjugated peptides by active learning-directed coarse-grained molecular simulation, *J. Phys. Chem. B*, 2020, **124**, 3873–3891.
36 J. S. Smith, B. Nebgen, N. Lubbers, O. Isayev and A. E. Roitberg, Less is more: Sampling chemical space with active learning, *J. Chem. Phys.*, 2018, 148.
37 Q. M. Gallagher and M. A. Webb, Data efficiency of classification strategies for chemical and materials design, *Digital Discovery*, 2025, **4**, 135–148.
38 R. A. Fisher, *The design of experiments*, 1937.
39 M. D. McKay, R. J. Beckman and W. J. Conover, A comparison of three methods for selecting values of input variables in the analysis of output from a computer code, *Technometrics*, 2000, **42**, 55–61.
40 I. M. Sobol, The distribution of points in a cube and the approximate evaluation of integrals, *USSR Comput. Math. Math. Phys.*, 1967, **7**, 86–112.
41 M. H. Tan, Minimax designs for finite design regions, *Technometrics*, 2013, **55**, 346–358.
42 L. Pronzato, Minimax and maximin space-filling designs: some properties and methods for construction, *J. Soc. Fr. Stat.*, 2017, **158**, 7–36.
43 R. K. Tripathy and I. Bilionis, Deep UQ: Learning deep neural network surrogate models for high dimensional uncertainty quantification, *J. Comput. Phys.*, 2018, **375**, 565–588.
44 H. M. D. Kabir, A. Khosravi, M. A. Hosen and S. Nahavandi, Neural network-based uncertainty quantification: A survey of methodologies and applications, *IEEE Access*, 2018, **6**, 36218–36234.
45 L. Hirschfeld, K. Swanson, K. Yang, R. Barzilay and C. W. Coley, Uncertainty quantification using neural networks for molecular property prediction, *J. Chem. Inf. Model.*, 2020, **60**, 3770–3780.
46 A. P. Soleimany, A. Amini, S. Goldman, D. Rus, S. N. Bhatia and C. W. Coley, Evidential Deep Learning for Guided Molecular Property Prediction and Discovery, *ACS Cent. Sci.*, 2021, **7**, 1356–1367, DOI: 10.1021/acscentsci.1c00546.
47 F. Grasselli, S. Chong, V. Kapil, S. Bonfanti and K. Rossi, Uncertainty in the era of machine learning for atomistic modeling, *Digital Discovery*, 2025, **4**, 2654–2675, DOI: 10.1039/D5DD00102A.
48 D. Ginsbourger, R. Le Riche and L. Carraro, *Kriging is well-suited to parallelize optimization*, 2010.
49 P. I. Frazier, A Tutorial on Bayesian Optimization, *arXiv*, 2018, preprint, arXiv:1807.02811, DOI: 10.48550/arXiv.1807.02811.
50 A. I. Cowen-Rivers, W. Lyu, R. Tutunov, Z. Wang, A. Grosnit, R. R. Griffiths, A. M. Maraval, H. Jianye, J. Wang and J. Peters, Hebo: Pushing the limits of sample-efficient hyper-parameter optimisation, *J. Artif. Intell. Res.*, 2022, **74**, 1269–1349.
51 W. Lyu, F. Yang, C. Yan, D. Zhou and X. Zeng, Batch Bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design, *Proc. Mach. Learn. Res.*, 2018, **80**, 3306–3314.
52 J. González, Z. Dai, P. Hennig and N. D. Lawrence, Batch Bayesian Optimization via Local Penalization, *arXiv*, 2015, preprint, arXiv:1505.08052, DOI: 10.48550/arXiv.1505.08052.
53 J. Fromer, R. Wang, M. Manjrekar, A. Tripp, J. M. Hernández-Lobato and C. W. Coley, Batched Bayesian Optimization by Maximizing the Probability of Including the Optimum, *J. Chem. Inf. Model.*, 2025, **65**, 4808–4817, DOI: 10.1021/acs.jcim.5c00214.
54 M. J. Tamasi, R. A. Patel, C. H. Borca, S. Kosuri, H. Mugnier, R. Upadhya, N. S. Murthy, M. A. Webb and A. J. Gormley, Machine Learning on a Robotic Platform for the Design of Polymer–Protein Hybrids, *Adv. Mater.*, 2022, **34**, 2201809, DOI: 10.1002/adma.202201809.
55 Q. Nguyen and A. B. Dieng, Quality-Weighted Vendi Scores And Their Application To Diverse Experimental Design, *arXiv*, 2024, preprint, arXiv:2405.02449, DOI: 10.48550/arXiv.2405.02449.
56 F. Häse, M. Aldeghi, R. J. Hickman, L. M. Roch and A. Aspuru-Guzik, Gryffin: An algorithm for Bayesian optimization of categorical variables informed by expert knowledge, *Appl. Phys. Rev.*, 2021, **8**, 031406, DOI: 10.1063/5.0048164.
57 N. H. Angello, D. M. Friday, C. Hwang, S. Yi, A. H. Cheng, T. C. Torres-Flores, E. R. Jira, W. Wang, A. Aspuru-Guzik, M. D. Burke, C. M. Schroeder, Y. Diao and N. E. Jackson, Closed-loop transfer enables artificial intelligence to yield chemical knowledge, *Nature*, 2024, 1–8, DOI: 10.1038/s41586-024-07892-1.
58 D. N. Cakan, E. Oberholtz, K. Kaushal, S. P. Dunfield and D. P. Fenning, Bayesian optimization and prediction of the durability of triple-halide perovskite thin films under light and heat stressors, *Mater. Adv.*, 2025, **6**, 598–606, DOI: 10.1039/D4MA00747F.
59 A. Dave, J. Mitchell, S. Burke, H. Lin, J. Whitacre and V. Viswanathan, Autonomous optimization of non-aqueous Li-ion battery electrolytes via robotic experimentation and machine learning coupling, *Nat. Commun.*, 2022, **13**, 5454, DOI: 10.1038/s41467-022-32938-1.
60 H. Ros, Y. Abdalla, M. T. Cook and D. Shorthouse, Efficient discovery of new medicine formulations using a semi-self-driven robotic formulator, *Digital Discovery*, 2025, **4**, 2263–2272, DOI: 10.1039/D5DD00171D.
61 A. Thelen, M. Zohair, J. Ramamurthy, A. Harkaway, W. Jiao, M. Ojha, M. U. Ishtiaque, T. A. Kingston, C. L. Pint and C. Hu, Sequential Bayesian optimization for accelerating

the design of sodium metal battery nucleation layers, *J. Power Sources*, 2023, **581**, 233508, DOI: 10.1016/j.jpowsour.2023.233508.

62 S. Ament, M. Amsler, D. R. Sutherland, M.-C. Chang, D. Guevarra, A. B. Connolly, J. M. Gregoire, M. O. Thompson, C. P. Gomes and R. B. van Dover, Autonomous materials synthesis via hierarchical active learning of nonequilibrium phase diagrams, *Sci. Adv.*, 2021, 7, eabg4930, DOI: 10.1126/sciadv.abg4930.