D10.1 – Active learning package/module demonstrated to work on pre-generated simulation or experimental data sets

VERSION

VERSION	DATE
1.0	25.02.22

PROJECT INFORMATION

GRANT AGREEMENT NUMBER	957189
PROJECT FULL TITLE	Battery Interface Genome - Materials Acceleration Platform
PROJECT ACRONYM	BIG-MAP
START DATE OF THE PROJECT	1/9-2020
DURATION	3 years
CALL IDENTIFIER	H2020-LC-BAT-2020-3
PROJECT WEBSITE	big-map.eu

DELIVERABLE INFORMATION

WP NO.	10
WP LEADER	KIT
CONTRIBUTING PARTNERS	ULIV, ITU, BASF, NVOLT, Fraunhofer, FZJ, CID, DTU, KIT
NATURE	Demonstrator
AUTHORS	H. Stein, F. Rahmanian, J. Flowers, P. Schäfer, A. Bhowmik, I. Cekic-Laskovic, F. Hanke, S. Clark, I.E. Castelli, E. Ayerbe, P. Cerejio, C. Ganunza, R. Ciria, B. Zhang, L. Fischer, A. Sanin, I. Suvrau, G. Pizzi, H. Hajiani, N. Safei, J. Carlsson, M. Vogler, P.B. Jørgensen
CONTRIBUTORS	All partners within the work package 10
CONTRACTUAL DEADLINE	28.02.2021
DELIVERY DATE TO EC	25.02.2022
DISSEMINATION LEVEL (PU/CO)	PU

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957189. The project is part of BATTERY 2030+, the large-scale European research initiative for inventing the sustainable batteries of the future.

ABSTRACT

BIG-MAP aims at accelerating battery research and thus needs methods to explore phases, formulations, and compositions, i.e., the chemical space in an efficient manner. This is since the chemical space, spanned by the combinatorics of mixing electrolytes, electrodes, and processes to synthesize solid electrolyte interphases is so vast and fine-grained that it is intractable using brute force screening. Work packages like WP3 and WP6 may perform simulations and experiments at a greatly accelerated pace, but without guidance, there is no merit in accelerating throughput alone. Through the interfaces developed in WP9, robotic interfaces in WP4, the protocols in WP8, and the common language (i.e., ontology) in WP7, WP10 aims at acting as an accelerator to guide experimentalists and theorists alike towards joint accelerated materials discovery, i.e., deploy AI/ML (artificial intelligence / machine learning) algorithms (with uncertainty quantification developed in WP11) to efficiently sample the chemical search-space through iterative guidance of experiments and simulation but also offering the possibility to trigger an experiment from a simulation.

The WP is divided into four tasks and corresponding deliverables. This deliverable report, D10.1, is concerned with demonstrating active learning on pre-generated simulation or experimental data sets, which is the prerequisite to deploying one or multiple centralized AI agent/s in BIG-MAP.

Therefore, the work package closely works with the data-generating WPs (e.g., WP6, WP2) to ingest data from both theory and experiment and echo back measurement requests (in collaboration with WP4) for multishot/many-cycle active learning.

This deliverable report first demonstrates **active learning on pre-generated experimental data** from WP6 to optimize electrolyte conductivity. Then we demonstrate partial **autonomous materials discovery in a one-shot active learning demonstration** in full exploitation mode. We secondly demonstrate **active learning deployed to two machines, with on-the-fly machine learning,** and lastly, **a new way of deploying active learning to simulation AND experiments simultaneously,** allowing them to trigger each other.

The newly developed **f**ast **in**tention **a**gnostic active **le**arning **s**erver (*finales*) concept allows the incorporation of experimental or simulation methods that do not (yet) have APIs, i.e., where there are humans in the loop.

We thereby demonstrated the capability of active learning in BIG-MAP and deployed two (one new) methods to accelerate active learning to the battery research community.

Overall, this deliverable report is therefore structured such that we first demonstrate active learning on a pre-generated dataset from an experiment from WP6 (FZJ), then a framework that is capable of deploying active learning to instrumentation and codes, and finally, a framework is demonstrated that performs multimodal optimization across fidelities together with WP3, WP6, WP11, WP8, WP7, and WP9.

The overall modular approach allows for a facile exchange of underlying optimizers and “to be orchestrated” agents and with the newest development of *finales* that is, to the best of our knowledge, **the first demonstration of an autonomous orchestrator tapping in simulation-, experimental- and human-agents.**

1. Active learning on pre-generated experimental data

Even with the great throughput of some experimental or theoretical methods, there are limitations in cost and time that prevent an exhaustive and fine sampling of the chemical space. Therefore, methods need to be deployed that help in the efficient exploration and exploitation of the chemical space. Typically, in active learning, there is a workflow as depicted below in Figure 1, where an example active learning cycle is shown.

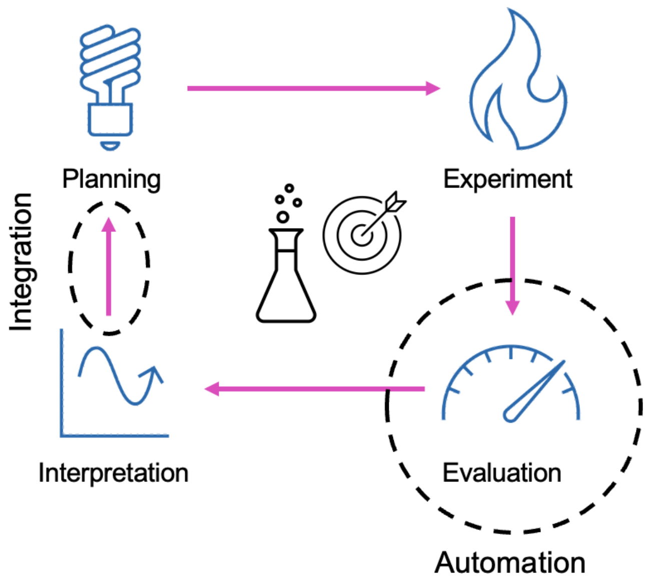


Figure 1. Emblematic high-level active learning cycle consisting of a planning, experiment, evaluation, and interpretation/suggestion phase. The bottlenecks in any of these are the integration and automation of all tasks, i.e., the seamless interconnection and intervention-free operation.

In a prototypical active learning cycle, some experimental (or simulation) device measures some data, which is then analyzed. This analysis is added to a database from which some regression or classification algorithm is trained that can also calculate aleatoric and epistemic uncertainties (though not necessarily differentiate between them). This trained model is then used to predict some materials' property and uncertainty for a fine grid of possible, yet unexplored, input parameters. From the predictions and uncertainties, an acquisition function is built.
The acquisition function can incorporate both the prediction and uncertainty (possibly even the different uncertainty quantifications of aleatoric and epistemic uncertainty) to suggest subsequent experiments that are likely to exhibit better functional properties or where the model is highly uncertain. Suppose one does not consider the uncertainty and only aims at maximizing the outcome.

In that case, the acquisition strategy is referred to as being in full *exploitation mode*. In contrast, a sole focus on performing experiments that have maximum uncertainty (i.e., where the model is likely to learn most from) is typically referred to as being in full *exploration mode*. There are several different acquisition strategies possible; we have below implemented several strategies like expected improvement of maximal arbitrage.

With the final goal of WP10 being the accelerator for AI-guided materials discovery, an interface for experiments and simulations is needed. There is thus a critical need to build the necessary infrastructure for AI-accelerated materials discovery consisting of modules for performing the uncertainty aware:

1) active learning
2) data analysis
3) data reassembly

These modules need to be part of an open framework in which machinery and software can communicate with each other.

The demonstration and deliverable of active learning on experimental or simulation data is therefore structured into four sections:

1) Demonstration of active learning on a pre-generated dataset demonstrating that it works in principle on the chosen data for battery-related experiments
2) A suggestion of new experiments in a semi-autonomous workflow in a full exploitation mode for one-shot active learning
3) Demonstration of an active learning framework with hardware in the loop
4) Demonstration of an active learning framework that bridges theory and experiment

1.1 Demonstration of active learning on a pre-generated dataset

As discussed above, the deployment of active learning can only work when data science-related modules can freely communicate with the data management and measurement devices. To assess the amenability of active learning to battery-related research, we choose electrolyte conductivity at different temperatures as a first demonstration due to the relative “smoothness” of the problem.

This demonstration of active learning on a pre-generated dataset is performed on data generated in WP6 (as part of D6.2 – first deployment of experimentally derived high-throughput data) at FZJ and measures the electrolyte conductivity of formulations containing ethylene carbonate (EC), propylene carbonate (PC), lithium hexafluorophosphate (LIPF6), and ethylmethylcarbonate (EMC) mixtures. A detailed description of the experimental setup is discussed in Deliverable D6.3.

Summarized, FZJ has a setup that is capable of mixing electrolyte formulations of liquid and solid compounds at almost arbitrary concentrations and measuring the conductivity as a function of temperature. While the measurements were done by performing electrochemical impedance spectroscopy on individual formulations at different temperatures, we consider the experiments at different temperatures to be independent and test the active learning there independently.

A very first test of the feasibility study was done by allowing a maximum of 20 measurements with random initialization of one example and a vanilla random forest regressor (typically bad performance in extrapolation). Already here, we see an acceleration factor of 3x after only 18 active learning cycles for temperatures above 0°C, as shown in Figure 2.

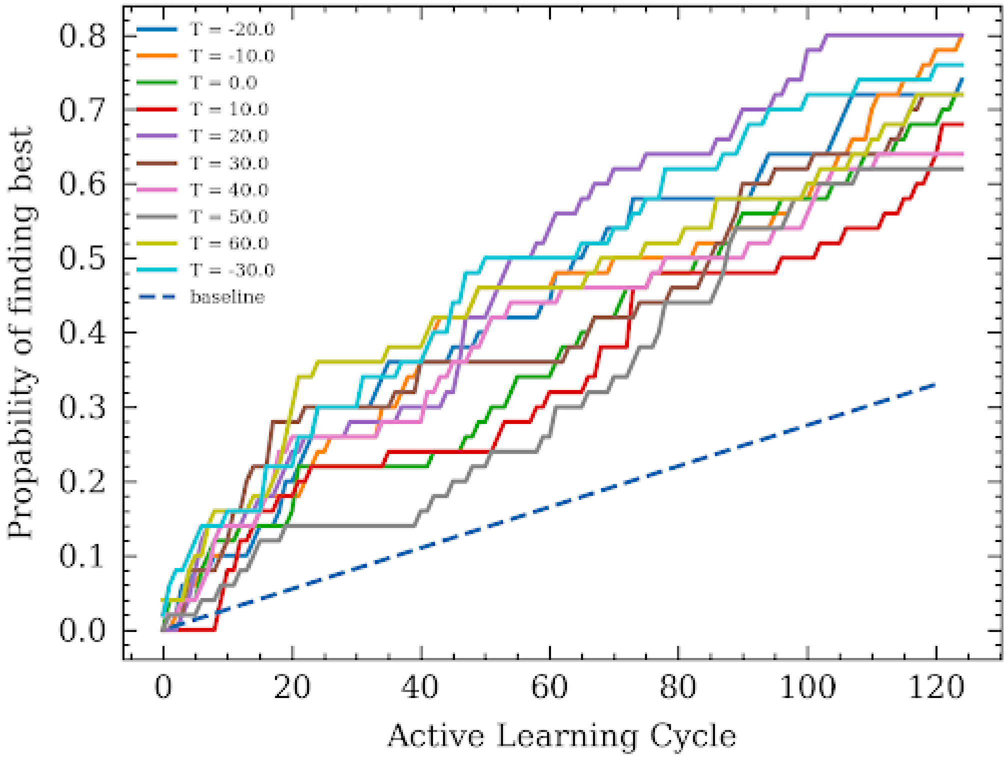


Figure 2. Active learning example for a maximum of 120 active learning cycles for different temperatures Even though the model is run in full exploration mode using a mix of aleatoric and epistemic uncertainty (i.e., the lowest gain in possible optimization speed) an acceleration factor of up to 5x is observed.

A more fundamental problem in active learning is a machine's learned modelling ability to extrapolate towards high or low values/properties, as exemplarily shown in Figure 3. There, a regression model has been trained with data from only the middle percentile, i.e., from 15-85% of the data. At the same time, the examples of high and low y-values (here, we observe the 2-dimensional Schwefel function, which has many local minima and maxima) are used as a test data set. The first plot shows the data distribution, the second shows the error upon training with this holdout, and the bottom shows the error when the dataset is split randomly. This is to exemplify the problem in active learning that when models are used that only try to perform interpolation type regression, there is generally a bad expected behaviour when the model is asked to extrapolate to high or low unseen values. It should also be mentioned that this top/bottom type holdout is not an extrapolation in the x/y input space.

Overall, this well-known problem motivates the further investigation of autonomous hypothesis generation as done in collaboration with WP11. The BIG models developed in WP11 will, in the course of the project, allow WP10 to deploy physics and chemistry informed (instead of regressed) models to BIG-MAP.

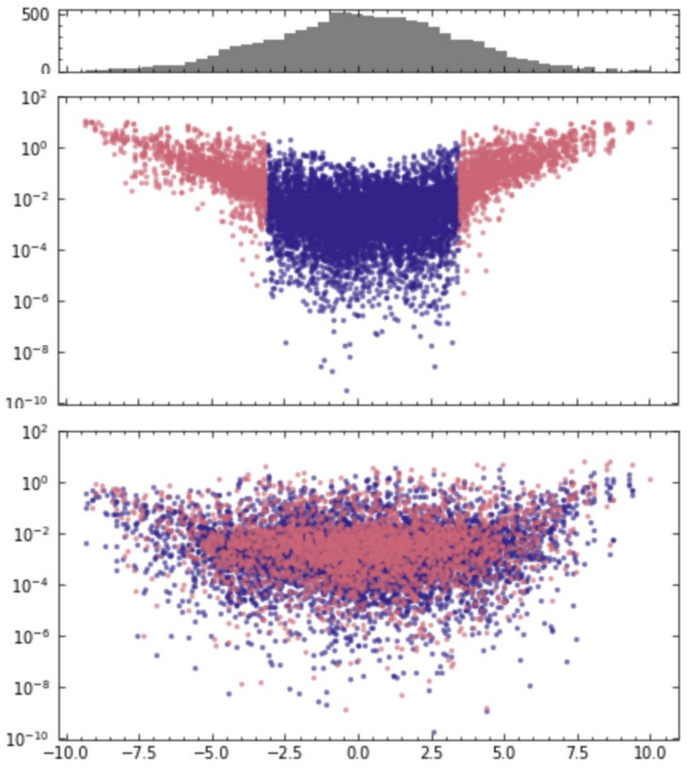


Figure 3. Example of a machine learning model's inability to predict high and low output values if it is only trained with values from the 15-85 percentile. This situation (as shown in the middle plot) is most emblematic to research done for extrapolation and, therefore, in an active learning run. Measuring the uncertainty in these extrapolation settings is of utmost importance; hence there is a strong interlinking with the uncertainty quantification efforts in WP11.

Closest to these models developed in WP11 are polynomial regression models with strong regularization. With those models, it is possible to fit a response surface for testing how an active learning study could look like at a certain temperature. In the next chapter, a one-shot active learning study is demonstrated using this data. Figure 4 shows a simulated active learning study that demonstrates the possibility to accelerate electrolyte optimization by up to 5 x even in a fully exploratory (the slowest for acceleration) setting.
As part of the HELAO package (see below), we have implemented a series of optimizers and acquisition functions that offer the entire BIG-MAP community and beyond to deploy active learning to experiments and simulations. HELAO is part of the BIG-MAP App Store. Together with WP4, these methods are also deployed to a more efficient exploration of liquid extraction methods. An exhaustive active learning run on a fit response surface (to demonstrate how well an active learning run could work) is shown in Figure 2.

1.2 Demonstration of active learning in a one-shot mode

In the previous section, we have demonstrated that, in principle, electrolyte conductivity is amendable for active learning. We have furthermore used a strongly regularized polynomial regression with the addition of piecewise polynomial regression. The reason is that in scenarios with very few data points (<30 measurements) at reasonably high dimensionality (>3), models like random forests underfit gaussian processes work only upon careful hyper tuning but then fail at large dataset sizes. The ensemble polynomial regression is based on ensemble piecewise polynomial regression. This can approximate functions through fitting many polynomial equations to random subsets of the data. Whilst for ensemble linear regression, an extrapolation tends not to diverge, any polynomial will diverge in an extrapolation scenario. We therefore only consider polynomial regressors to be predictors within their input data range unless heavily regularized as the extremes of the chemical space spannable in the experiments are fixed this an amendable strategy.

Below, predictions for electrolyte conductivity at different temperatures are shown that yield interesting trends. The plots in Figures 4-7 suggest that at low temperatures, a narrow optimum at lower PC to EC and comparably high $LiPF_6$ ratios would be beneficial for the electrolyte conductivity. These findings are in stark contrast to high temperatures, where a trimodal optimum exists.

The figures below show the composition space investigated, which is spanned by the LiPF6 to EC+PC ratio and the PC to EC+PC ratio. The EC+PC to EMC ratio was held constant. The underlying data was supplied through D6.3 by FZJ as part of that deliverable.

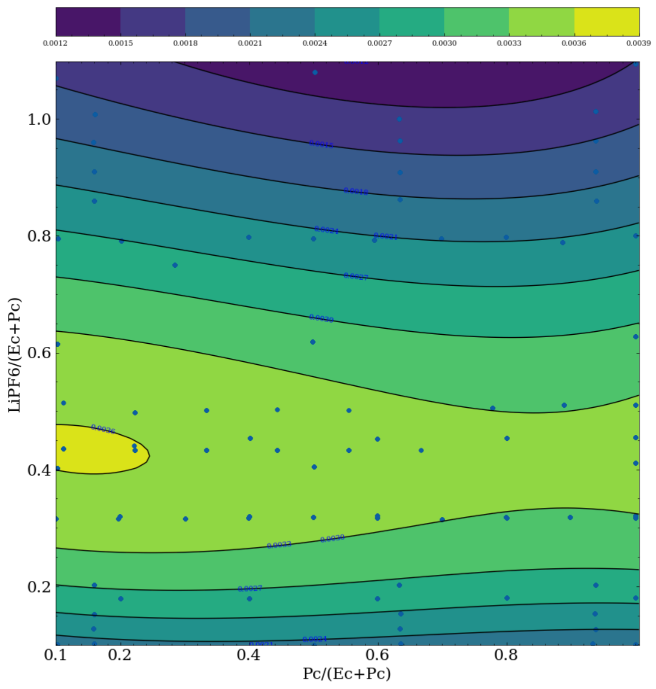


Figure 4. Electrolyte conductivity in a quaternary Li-ion battery electrolyte (LiPF6, EC, PC, EMC) at -10°C. The blue dots indicate where measurements were made by FZJ. The color overlay corresponds to the predicted electrolyte conductivity from the regression model. At -10°C, a narrow optimum exists at comparably low LiPF6 ratios and low PC ratios.

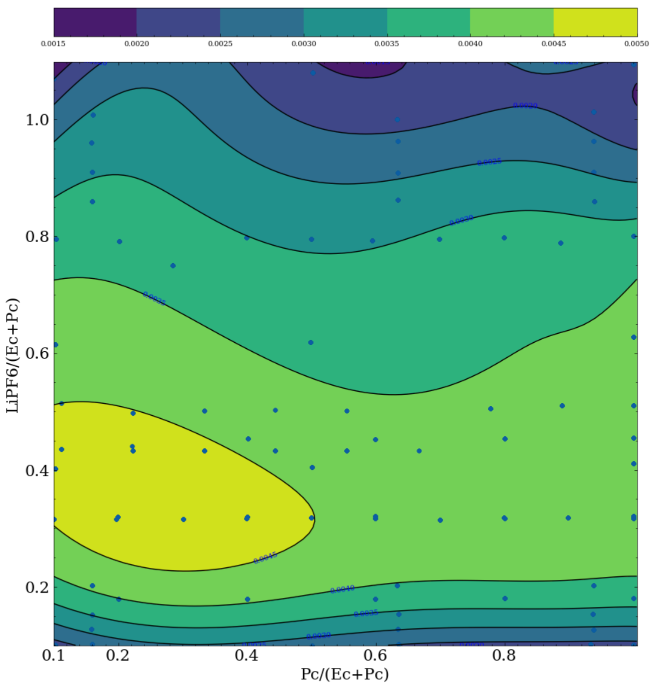


Figure 5. Electrolyte conductivity in a quaternary Li-ion battery electrolyte (LiPF6, EC, PC, EMC) at 0°C. The blue dots indicate where measurements were made by FZJ. The color overlay corresponds to the predicted electrolyte conductivity from the regression model. At 0°C, a broader optimum that at -10°C exists at comparably low LiPF6 ratios and low PC ratios.

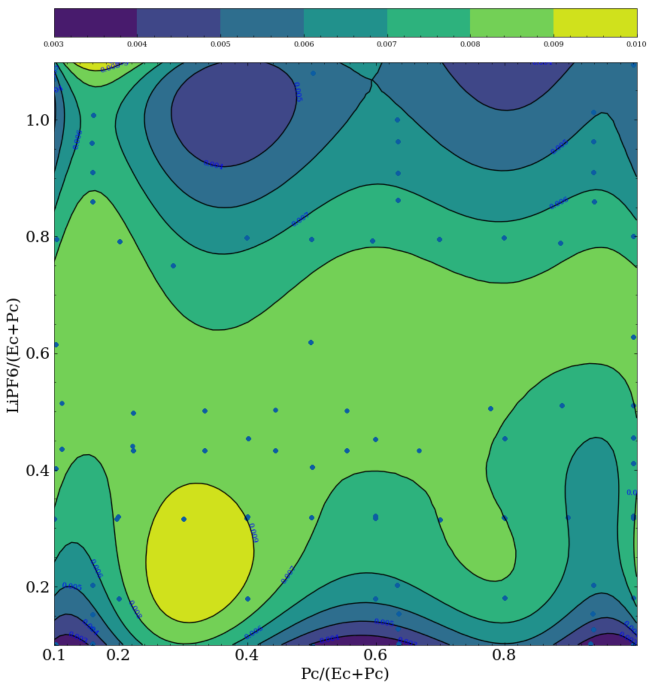


Figure 6. Electrolyte conductivity in a quaternary Li-ion battery electrolyte (LiPF6, EC, PC, EMC) at 30°C. The blue dots indicate where measurements were made by FZJ. The color overlay corresponds to the predicted electrolyte conductivity from the regression model. At 30°C, a first indication of multiple local optima appears. At 30°C, it seems like even lower LiPF6 ratios are better.

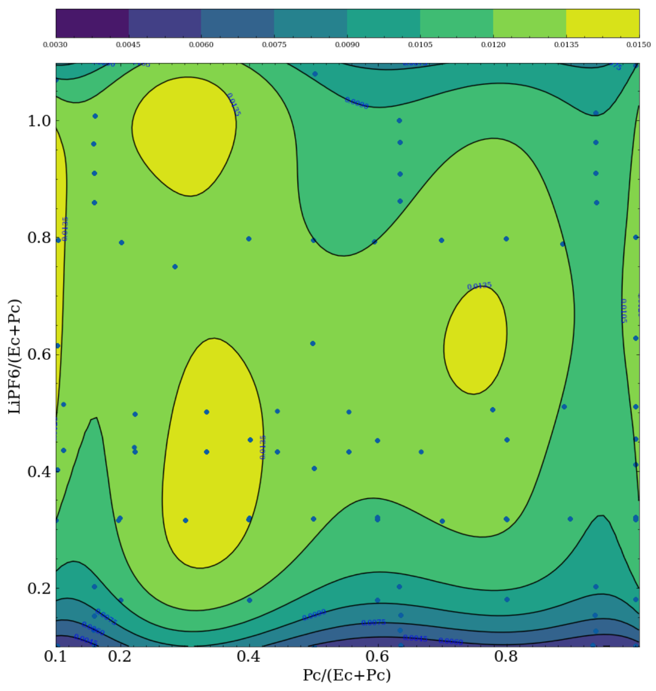


Figure 7. Electrolyte conductivity in a quaternary Li-ion battery electrolyte (LiPF6, EC, PC, EMC) at 60°C. The blue dots indicate where measurements were made by FZJ. The color overlay corresponds to the predicted electrolyte conductivity from the regression model. At 60°C, a multimodal optimum seems to exist for the highest electrolyte conductivity. Interestingly, two optima - high PC and high LiPF6 at low PC ratio - are yet unexplored.

Figures 4-7 suggest that unexpected new electrolyte compositions are still to be discovered to optimize LiPF6 salt-based electrolytes. Using these models, KIT, DTU and FZJ performed a one-shot active learning study in which, using the simple model of multiple polynomials, new formulations were suggested to FZJ to try. Figures 8-10 demonstrate the excellent agreement between the predictions from KIT and experimental results from FZJ for low temperatures. Predictions were made for all temperatures ranging from -30°C to 60°C in 10°C intervals, but to conserve space, only a selection is shown in Figure 8-10. To put these results into perspective: typical blends of $LiPF_6$ containing electrolytes exhibit conductivities of 5-10 mS/cm at room temperature[1]. There is a good agreement in conductivity, especially for low temperatures (<0°C). With some of our predictions and the corresponding results reaching ca. 2 mS/cm at -30°C, we seem to have hit the physical maximum of conductivity in this chemical system at ultra-low temperatures. The high temperatures, especially above 50°C, tend to be less reliable in finding an optimized electrolyte candidate formulation suggesting that the previously identified trimodal optimum is not real. The herein used model does not have uncertainty quantification and therefore motivates the use of more adapted acquisition functions that incorporate both exploitative (as done here) and explorative methods (as shown above in Figure 3).

[1] See the exhaustive review by Hubble et al. from 2022 DOI: https://doi.org/10.1039/D1EE01789F

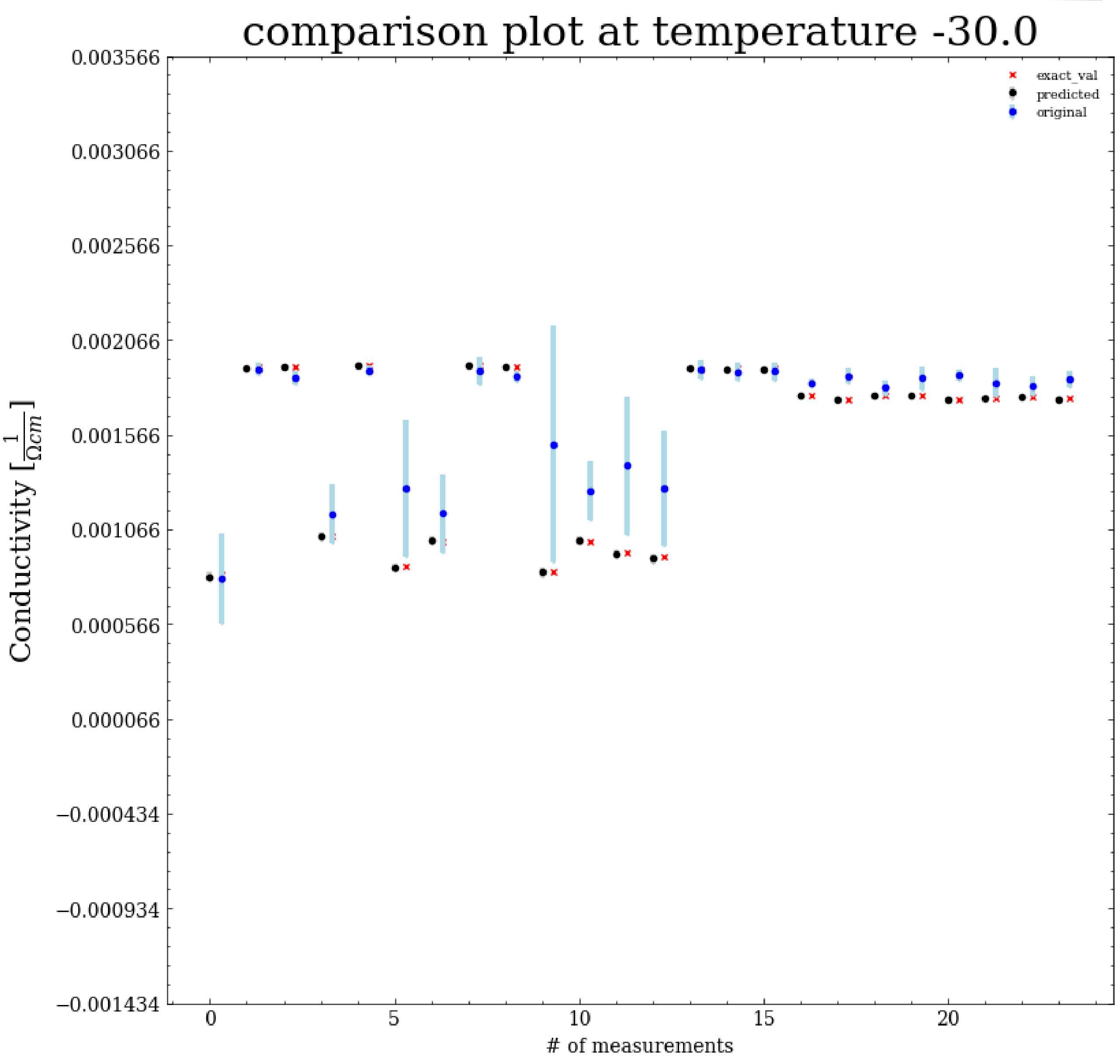


Figure 8. Predicted conductivities from the hypothetical top percentile at -30°C, measured conductivities, and adjusted predictions. An adjustment was necessary as predicted formulations could not be reached exactly. The exact would-be predictions are shown as red crosses. The error bars are obtained from the min/max values of the repeated measurements, with the average shown as a blue dot. The predicted values all lie within a single standard deviation of the experiments (black dots).

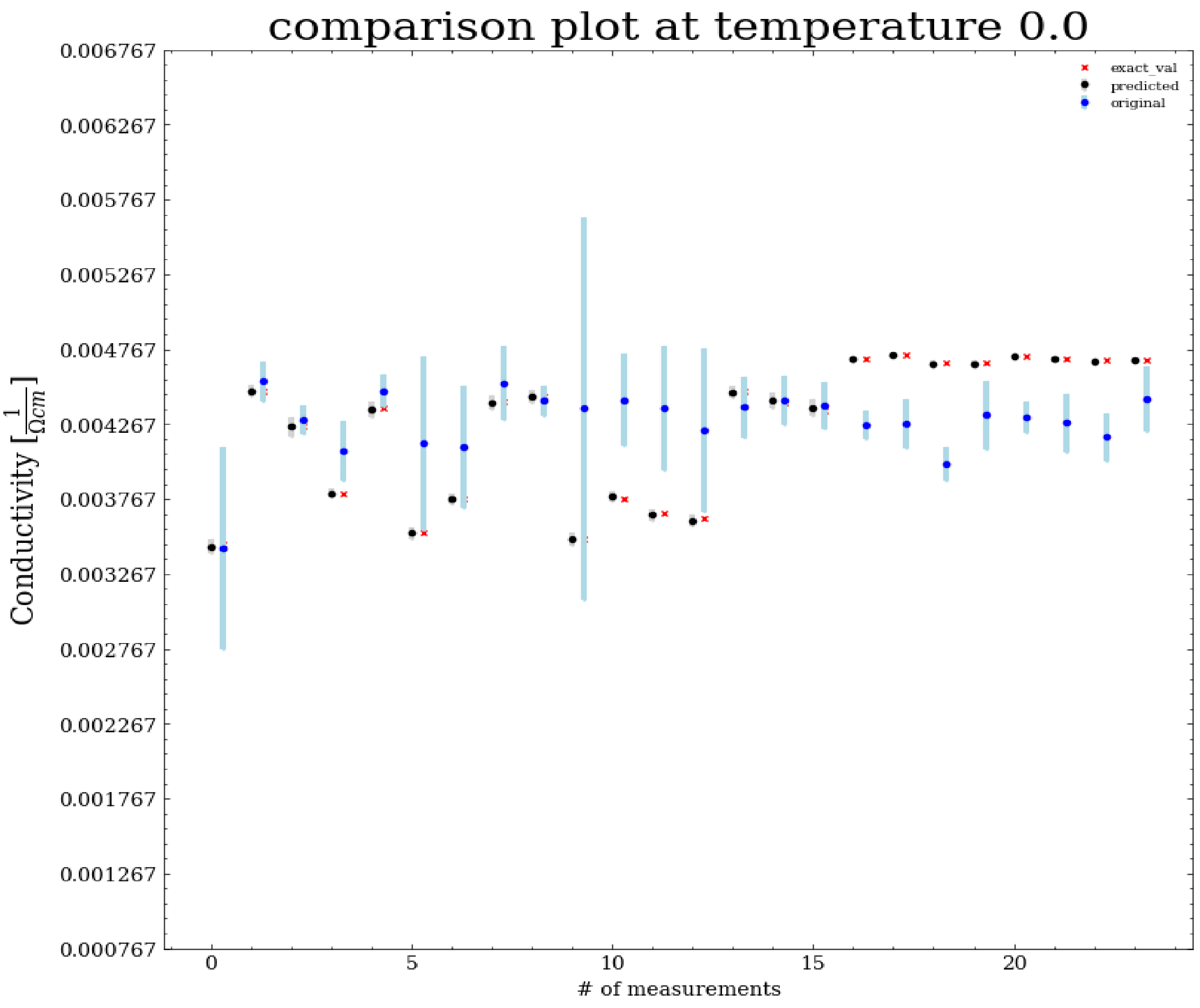


Figure 9. Predicted conductivities from the hypothetical top percentile at -30°C, measured conductivities, and adjusted predictions. An adjustment was necessary as predicted formulations could not be reached exactly. The exact would-be predictions are shown as red crosses. The error bars are obtained from the min/max values of the repeated measurements, with the average shown as a blue dot. The model tends to predict conductivities well for most formulations.

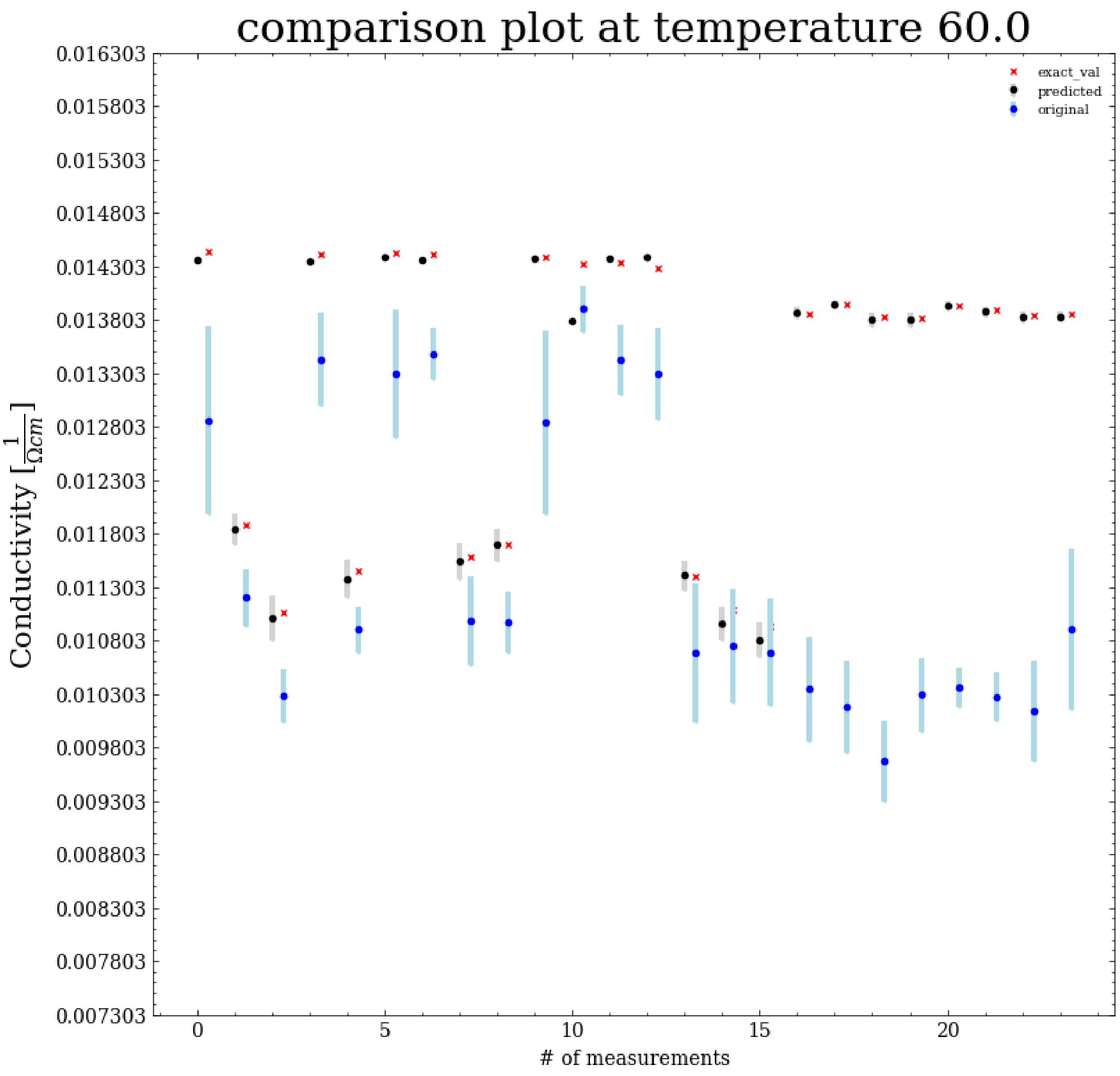


Figure 10. Predicted conductivities from the hypothetical top percentile at 60°C, measured conductivities, and adjusted predictions. An adjustment was necessary as predicted formulations could not be reached exactly. The exact would-be predictions are shown as red crosses. The error bars are obtained from the min/max values of the repeated measurements, with the average shown as a blue dot. Comparison to the fitted response curve from above suggests that some of the local maxima were wrongly predicted; however, the experimentally obtained conductivities of up to 13.8 mS/cm are extremely high.

To summarize, we have thus demonstrated that simulated active learning on pre-generated data is possible and that we are capable of performing one-shot active learning and obtaining high-performing electrolytes at arbitrary temperatures.

2. The critical need for active learning orchestration frameworks in experimental chemistry

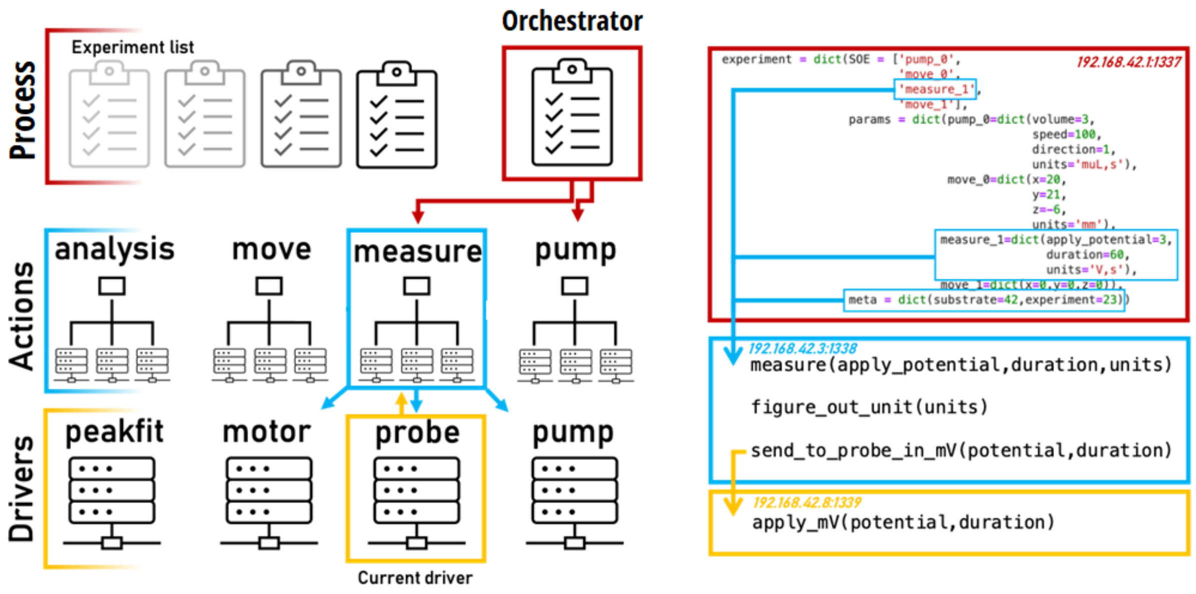
The previous demonstrations operated on a pre-existing dataset and did not require direct interfacing of the algorithms with hardware, the existence of a data analysis framework, or proper protocols and standards in place. Therefore, the next step towards the deployment of active learning is an autonomous orchestration framework capable of performing distributed active learning on a fleet of machines and software. This is necessary as in BIG-MAP, several partners can perform experiments and simulations and offer some form of interfacing to them. Referring back to Figure 1, there is thus a need for a framework that incorporates all steps and exposes them through openly accessible APIs. These research tasks, therefore, need to be tightly integrated and autonomous. We have thus demonstrated a method for this integration for research tasks within a hierarchical experimental laboratory automation and orchestration (HELAO) framework. The code is available through the BIG-MAP app store at https://big-map.github.io/big-map-registry/apps/HELAO.html and the manuscript under the CC-BY-NC at https://doi.org/10.1002/admi.202101987. We demonstrate the capability of orchestrating distributed research instruments that can incorporate data from experiments, simulations, and databases. HELAO offers interfacing laboratory hardware and software distributed across several computers and operating systems for executing an experiment, data analysis, provenance tracking, and autonomous planning. Autonomous planning requires seamless integration of numerous accelerated research tasks, which this framework is capable of orchestrating in a facile manner. Research acceleration by instrument parallelization in terms of reduction of total experimental time is demonstrated to be close to 2x. This new kind of optimization paradigm in materials science requires device sharing and asynchronous multithreading, which to the best of our knowledge, is demonstrated for the first time in an active learning run. This acceleration is made possible by elevating data management from being an archiving method to being an integral part of the research execution process. HELAO derived data enables scientists to view, disseminate, and analyze data and recreate the experiment.

2.1 Active learning with hardware in the loop

Active learning with hardware in the loop has been demonstrated by Rahmanian et al. in Advanced Materials Interfaces 2022, 2101987, published by Wiley-VCH GmbH under the Creative Commons CC-BY-NC license from which we have adapted the text below.

In the early phase leading up to this demonstrator, few standards and protocols existed for describing what an experiment is, how it could be communicated across groups and how this would interface with the ontologies developed in WP7. As WP7 is working on the Battery Interface Ontology, we in WP10 saw the need to translate it into an operational ontology. See also the last section of this document on *finales* for further details on a direct mapping effort of an ontology for battery measurements. We base the standard for interfacing with hardware in an active learning setting from a bottom-up hardware perspective, as shown in Figure 11. Here, a research instrument to be interfaced is made up from a set of devices. A device is a piece of laboratory equipment, defined as the largest "thing" that has a dedicated communication stream, i.e., a multi-channel potentiostat or a motor control board.

Devices are typically shipped with a drivers that enable access to the device functions, i.e., measuring a current. From the top-down perspective, a user or operator wishes to perform a series of experiments that are each a list of actionable events (which we define as "actions") that are to be executed in some order with predefined or variable parameters. A list containing experiments is given to an orchestrator that governs their sequential execution from a queue, thereby constructing an virtual ad-hoc instrument. Everything that happens within that instrument originates from within the orchestrator.



```
experiment = dict(SOE = ['pump_0',
                         'move_0',
                         'measure_1',
                         'move_1'],
                  params = dict(pump_0=dict(volume=3,
                                            speed=100,
                                            direction=1,
                                            units='muL,s'),
                                move_0=dict(x=20,
                                            y=21,
                                            z=-6,
                                            units='mm'),
                                measure_1=dict(apply_potential=3,
                                               duration=60,
                                               units='V,s'),
                                move_1=dict(x=0,y=0,z=0)),
                  meta = dict(substrate=42,experiment=23))
```

```
measure(apply_potential,duration,units)

figure_out_unit(units)

send_to_probe_in_mV(potential,duration)
```

```
apply_mV(potential,duration)
```

Figure 11. A schematic representation of HELAO where experiments are executed by sequentially calling actions which are high-level wrappers for lower-level driver instructions. Communication goes hierarchically down from the orchestrator level to actions, which may communicate among each other, to the lowest level of drivers, which can only communicate with actions. The orchestrator, actions, and drivers are all exposing python class functions through a web interface, allowing for highly modular and distributed hosting of each item. Experiments are dictionaries containing a sequence of events (SOE) that outlines in which the actions are to be executed. Many experiments form a process. All actions require parameters and are supplied by metadata that is echoed back. The figure is taken from Rahmanian et al. Adv. Mat. Interfaces 2022, 2101987 under the Creative Commons CC-BY-NC license.

For the ensemble of devices to operate in concert as a single instrument, it is convenient to assemble the various elements listed above into a single software framework. Hierarchically from bottom to top, each device driver (internally communicating through, e.g., serial, TCP/IP commands, or a dynamic link library) is exposed through an uvicorn web server through fastAPI. Construction of more complicated but actionable functions ("actions") is based on these device driver's exposed API calls.

An example of an action to pump a mixture of three fluids would therefore be initiated by the orchestrator calling the action to pump a mixture. This action then calls the pump driver server. Internally the driver server is sequentially called by the mixture action as the hardware requires first initializing each pump channel, prime the pumps, and only then turning the pumps. The orchestrator will receive a nested reply from the action that entails all information exchanged, down to the lowest level, i.e., initialization, priming, execution.

A rigorous commitment to data management is foundational to this framework's implementation philosophy. Requests to the driver and action servers track all functions called, as well as all (echoed) input parameters and outputs of those functions. The orchestrator tracks additional metadata, such as the time at which an action was performed or the point on a substrate at which an experiment was conducted, in addition to accepting arbitrary custom metadata. These are then automatically saved (redundantly) in the native file format (if applicable) and in an hdf5 file together with the parameters and metadata. Methods for depositing the hdf5 file into institutional repositories like Kadi4Mat[22] or MEAD[19] repositories are automatically executed after each session. From Kadi4Mat[22], experimental data can be accessed internally and shared with the community through Materials Cloud or AIIDA[23,24], as per the agreed-upon data management plan in BIG-MAP.

Due to each element of our software framework being a server, a very high degree of modularity is achieved that allows, for instance, a single instance of a device to fulfill requests from multiple action servers or a single action server to communicate with multiple orchestrators, thereby allowing for distributed hosting of devices on different machines potentially scattered across the EU.

A goal of HELAO is to enable active learning accelerated experiments across a wide range of laboratory instruments. Active learning does, however, require automatic data analysis and machine learning-based suggestion of the next best subsequent experiment.

Therefore, what HELAO needs to be able to do is alter a subsequent experiment based on previous input. An active learning action within a SOE requires access to the data of one or all previous actions and the possibility to alter a parameter for future action of the current SOE to be executed. The data analysis servers or actions act like any other action/driver server combination with the exception that they provide pointers to the location in the orchestrator memory of where relevant data is stored. Likewise, the server dedicated to machine learning for active learning needs to be pointed to the input values of the experiments and the resulting target values from data analysis. Inside the active learning action, the datasets are aggregated on the fly from the orchestrator temporary storage (what is later the hdf5 file being uploaded). A suggestion can be made from a list of candidate parameters or be freely decided by the algorithm depending on the chosen optimizer. This active learning action then passes the suggested parameter pair through a special native command pointing to this "source value" returned by the action. This special native action is called after the active learning action and before the action to be modified. The active learning suggested parameter is then copied over to the "target" parameter of the subsequent experiment.

These functionalities allow for autonomous operation where the user only has to define the budget of active learning runs, pointers to the input and output values, and the choice of optimizer and the estimator.

The active learning server contains a broad range of optimizers and regression algorithms. Also, several acquisition functions have been implemented, including expected improvement (EI) probability of improvement (POI).
As some ML algorithms require significant computational resources within a fastAPI thread and some actions are data transfer intensive, servers may become unresponsive. To solve this issue, the most computationally expensive tasks like machine learning can be wrapped inside a Celery server when necessary to distribute high workloads across compute clusters. We empirically observed this necessity for long-running active learning runs with a high degree of freedom.

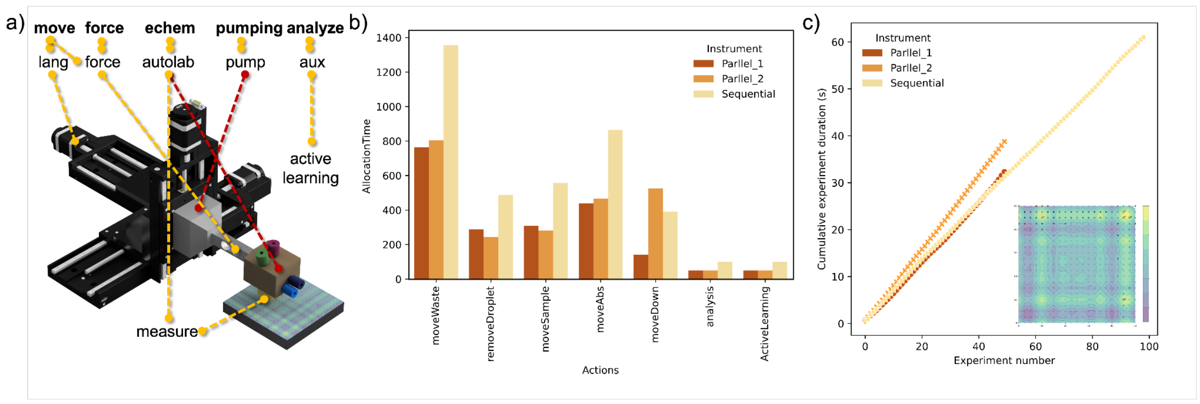


Figure 12. Schematic drawing of the HELAO hardware in the loop active learning run. Compared to two mock scanning droplet cells, SDCs running in parallel, this instrument has no pump, and the potentiostat is replaced by a measure driver that returns a function value from the Schwefel function depending on the visited substrate position. b) Time spent at each action for a sequential and a parallel run with two instruments c) total time spent per run. The time spent does not form a perfectly straight line as some actions need different times (i.e., movements are shorter or longer). The inset in c shows the parallel run and highlights visited points in black and red depending on whether they were visited by instrument 1 or 2. The figure is taken from Rahmanian et al. Adv. Mat. Interfaces 2022, 2101987 under the Creative Commons CC-BY-NC license.

A hardware-in-the-loop demonstration run of HELAO is shown in Figure 12. The instrument is copied two times where one setup was run in a fume hood, and another one was run in a glove box (hosting the orchestrator). The learner and optimizer are the same for both instruments as the orchestrator centrally runs it. Compared to a standard scanning droplet cell (SDC), the only difference is that there is no potentiostat and no pump used. This is to point the attention to the (arguably already complex) software-hardware interaction and not to the electrochemistry. Replacing the potentiostat in this hardware in the loop demonstration is done through the implementation of a device called measure. This device returns a scaled Schwefel function depending on the position where the SDC touches down on a substrate. The active learning run is stopped once a threshold value (top percentile) is found. Actions in this run consist of, e.g., “move to waste”, “remove the droplet”, “move to sample offset”, “move to the defined point”, “move down to substrate”, “get output value”, “predict the next best position using active learning algorithm”. One experiment takes about 108 seconds. Depending on the number of data points, the learning step requires more time. During the measurement, all data is constantly logged from all devices and subsequently uploaded to the data management repository (Kadi4Mat). The overall time required for the entire run was a little less than 3 hours and allowed for a fine-grained analysis of which action consumes the most experimental time as shown in Figure 12b From this analysis, it is, for instance, evident that motor movement between measurements consumes a substantial fraction of the experimental time, motivating efforts to enable faster movement. With 3856 seconds for the sequential run, there is a significant speedup when the experiment is run in parallel where instrument 1 (in the fume hood) takes 2041, and instrument 2 requires 2424 seconds to complete. As is evident from these numbers, the speedup is a little less than 2x for a parallel active learning run as asynchronous locks, and the machine learning consumes some of the time. To the best of our knowledge, this is the first demonstration of an active learning run involving two spatially distributed instruments and involving more than one operational PC and instruments distributed across space.

3. Linking experiment and theory in one active learning framework

So far, we have demonstrated the capability of optimizing battery electrolytes in simulated active learning, we have demonstrated the capability of active learning in a one-shot approach and obtained a physically optimal electrolyte formulation at -30°C for applications in low-temperature settings. We have then demonstrated a framework capable of orchestrating multiple machines in a laboratory at once through a central orchestrator. What is missing is the interconnection of different labs' different methods and decoupling experiments from intentions.

So far, active learning was concerned with altering the input parameters of experiment description, i.e., where a measurement took place on a sample or how much of a solvent was mixed into a vial. All orchestration frameworks and workflows (including HELAO as demonstrated above) are operationally close to the instrumentation/software they operate on.

As an example, the optimization of an electrolyte in HELAO would likely mean the change of a formulation, which would mean pumping different volumetric ratios into a probe-head. Changing the acquisition function would mean changing the function call in the sequence of events (SoE). In "pipeline pilot" (pipeline pilot is a 3DS workflow designer), this would mean changing input parameters to the simulation.

In both cases, an expert user would need to know what input parameters to the simulation code of what coordinates they need to change, i.e., they need to know about the underlying method or hardware. We believe that to link experiments and modelling efforts (in collaboration with D9.2), we are in dire need of a common language such that the suggestion of a measurement is decoupled from the underlying method with that something is measured as only this way a true multi-fidelity, multi-modal and multi-method autonomous platform is possible. BIG-MAP can only become a reality if we find an operational language (backed by an ontology) by which measurement requests can be triggered regardless of the underlying method being a modelling or experimental one.

Together with the colleagues from WP9, WP8, WP4, WP2, and WP3, we have therefore developed a schema using the pydantic Python library. This allows for facile integration into secure web frameworks like fastAPI.

Below is an example for a request of a quaternary formulation with DMC, PC, EMC, and LIPF6 is shown that requests and experiments are to be done. Changing the experiment to a simulation request can be done by simply changing the origin type to “simulation”.

To achieve this high degree of interoperability, we needed to design around potential miscommunications between experimentalists and theorists. First and foremost, all physical quantities are either Volumes or Mol, allowing for a simple exchange of data and greatest compatibility to BattInfo.

```
DMC = schemas_pydantic.Chemical(smiles='COC(=O)OC', name='DMC', reference='DMC Elyte 2020')
PC = schemas_pydantic.Chemical(smiles='CC1COC(=O)O1', name='PC', reference='PC ELyte 2020')
EMC = schemas_pydantic.Chemical(smiles='CCOC(=O)OC', name='EMC', reference='EMC ELyte_2020')
LIPF6 = schemas_pydantic.Chemical(smiles='[Li+].F[P-](F)(F)(F)(F)F', name='LiPF6',
                                                                   reference='LiPF6_Elyte_2020')
amnt500mmol = schemas_pydantic.Amount(value=0.5, unit='mol')
amnt100mmol = schemas_pydantic.Amount(value=0.1, unit='mol')

A = schemas_pydantic.Compound(chemicals=[DMC,LIPF6],
                              amounts=[amnt500mmol,amnt100mmol],
                              name='LiPF6 salt in DMC_5:1')
B = schemas_pydantic.Compound(chemicals=[PC,LIPF6],
                              amounts=[amnt500mmol,amnt100mmol],
                              name='LiPF6 salt in PC 5:1')
C = schemas_pydantic.Compound(chemicals=[EMC,LIPF6],
                              amounts=[amnt500mmol,amnt100mmol],
                              name='LiPF6 salt in EMC 5:1')

  form = schemas_pydantic.Formulation(compounds=[A, B, C], ratio=[0.3, 0.2, 0.5],
                                      ratio_method='volumetric')
  temp = schemas_pydantic.Temperature(value=380, unit='K')
  orig = schemas_pydantic.Origin(origin='experiment')
  fom = schemas_pydantic.FomData(value=42, unit="g/cm**3", origin=orig,
                                 measurement_id='123', name='Density')
  meas = schemas_pydantic.Measurement(formulation=form, temperature=temp,
                                      pending=False, kind=orig, fom_data=fom)
  ans  = requests.post(f"http://{config.host}:{config.port}/api/broker/post/measurement",
                       data=meas.json()).json()
```

Any type of optimizer (human or algorithmic) or even a simple exploratory study would thus consist of posting a measurement request as shown above to a server. Here we call the server to which the measurement request is being sent a “broker server”, as it brokers between experiments, AI, and modelling. The experiment client or the simulation client then simply asks, whenever they want, if a new request for their method is available. If so, they can choose to execute it. Each request and then later executed measurement receives unique identifiers from the broker that make the entire workflow trackable from suggestion to result and downstream. This setup allows interconnecting arbitrary workflow orchestrators as long as they exhibit the possibility to send web requests and ingest json schemas. This high-level *finales* workflow is shown in Figure 13.

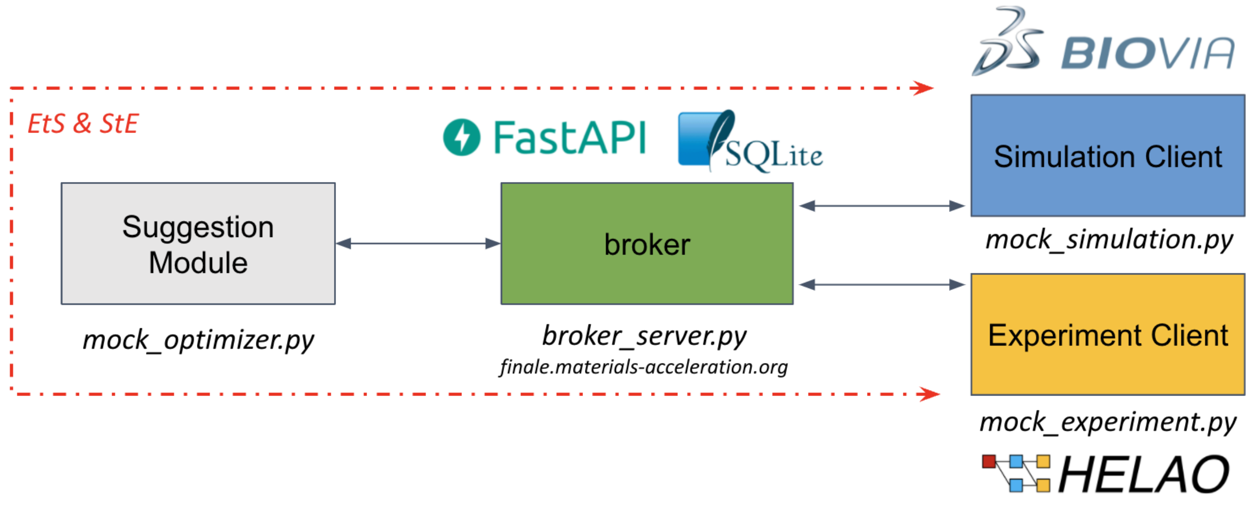


Figure 13. High-level setup of the *finales* workflow. All measurement requests and results are posted asynchronously to the broker server. This allows for an arbitrary number of Experiment, Simulation, and Suggestion Clients and even allows for humans in the loop. Stress testing on an instance running on an M1 Apple MacBook Air laptop puts a limit of roughly 30 clients posting one measurement result per second each.

The code is deposited in the (currently private) GitHub repository https://github.com/BIG-MAP/finales where users have the possibility to build a docker container for simple deployment. The *finales* framework is completely documented and follows the openAPI standard for exposing an API.

A live public demo is hosted at http://stein.hiu-batteries:49157, as shown in Figure 14 with the documentation available at http://stein.hiu-batteries:49157/docs. A simple user interaction is shown in Figure 15. Upon publication of the manuscript regarding *finales,* the code will be made publicly accessible and added to the BIG-MAP App Store.

First uses of *finales* include the triggering on experiments to measure density and viscosity of the compounds shown below:

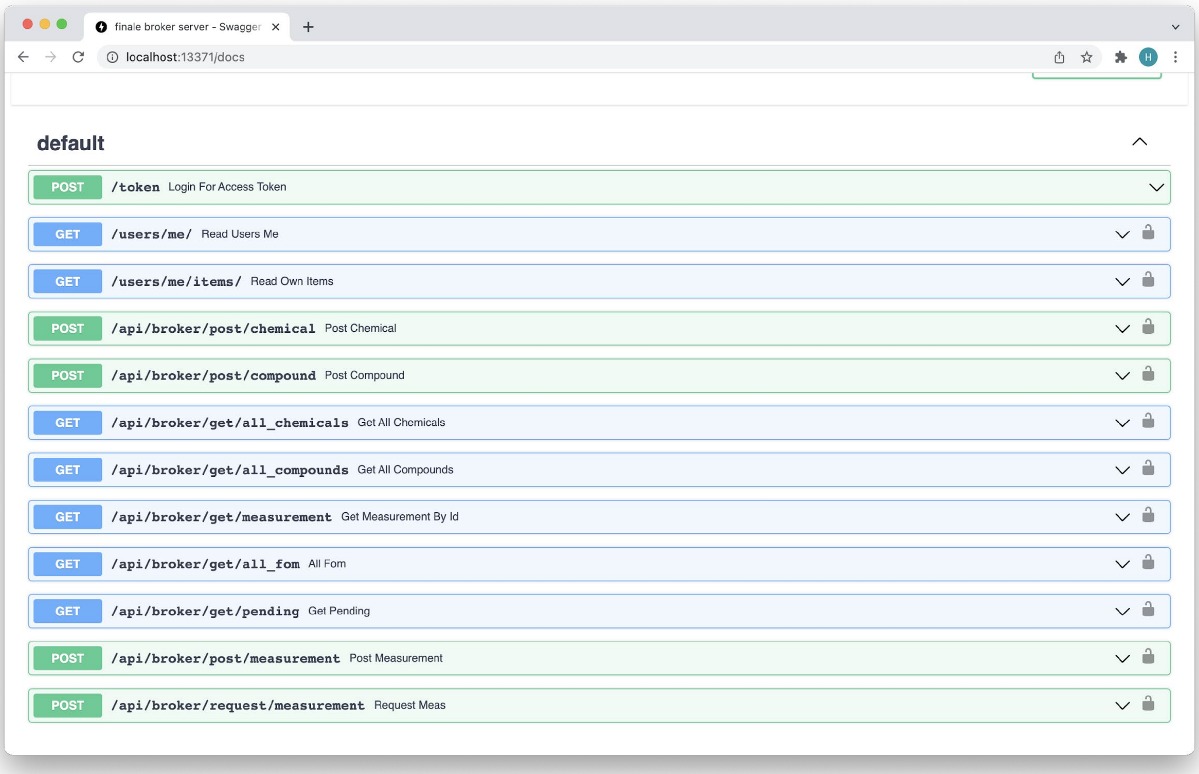
```
LiPF6_EC_DMC = schemas_pydantic.Compound(chemicals=[
    schemas_pydantic.Chemical(smiles='C1COC(=O)O1', name='EC', reference='EC_Elyte_2020'),
    schemas_pydantic.Chemical(smiles='COC(=O)OC', name='DMC', reference='DMC_Elyte_2020'),
    schemas_pydantic.Chemical(smiles='[Li+].F[P-](F)(F)(F)(F)F', name='LiPF6', reference='LiPF6_Elyte_2020')],
    amounts=[schemas_pydantic.Amount(value=0.47, unit='mol.-%'),
    schemas_pydantic.Amount(value=0.46, unit='mol.-%'),
    schemas_pydantic.Amount(value=0.07, unit='mol.-%')],
    name='LiPF6_salt_in_EC_DMC_1:1')

EC_DMC = schemas_pydantic.Compound(chemicals=[
    schemas_pydantic.Chemical(smiles='C1COC(=O)O1', name='EC', reference='EC_Elyte_2020'),
    schemas_pydantic.Chemical(smiles='COC(=O)OC', name='DMC', reference='DMC_Elyte_2020')],
    amounts=[schemas_pydantic.Amount(value=0.51, unit='mol.-%'),
    schemas_pydantic.Amount(value=0.49, unit='mol.-%')],
    name='EC_DMC_1:1')

LiPF6_EC_EMC = schemas_pydantic.Compound(chemicals=[
    schemas_pydantic.Chemical(smiles='C1COC(=O)O1', name='EC', reference='EC_Elyte_2020'),
    schemas_pydantic.Chemical(smiles='CCOC(=O)OC', name='EMC', reference='EMC_ELyte_2020'),
    schemas_pydantic.Chemical(smiles='[Li+].F[P-](F)(F)(F)(F)F', name='LiPF6', reference='LiPF6_Elyte_2020')],
    amounts=[schemas_pydantic.Amount(value=0.31, unit='mol.-%'),
    schemas_pydantic.Amount(value=0.60, unit='mol.-%'),
    schemas_pydantic.Amount(value=0.09, unit='mol.-%')],
    name='LiPF6_salt_in_EC_EMC_3:7')

EC_EMC = schemas_pydantic.Compound(chemicals=[
    schemas_pydantic.Chemical(smiles='C1COC(=O)O1', name='EC', reference='EC_Elyte_2020'),
    schemas_pydantic.Chemical(smiles='CCOC(=O)OC', name='EMC', reference='EMC_ELyte_2020')],
    amounts=[schemas_pydantic.Amount(value=0.34, unit='mol.-%'),
    schemas_pydantic.Amount(value=0.66, unit='mol.-%')],
    name='EC_EMC_3:7')
```

These were then also measured using the molecular dynamics tools available at 3DS. Furthermore, an interface to AIIDA at EPFL exists. First implementations of multi-modal, multi-fidelity active learning are running, but they are still ongoing due to their time requirements.



finale broker server - Swagger

localhost:13371/docs

default

POST /token Login For Access Token

GET /users/me/ Read Users Me

GET /users/me/items/ Read Own Items

POST /api/broker/post/chemical Post Chemical

POST /api/broker/post/compound Post Compound

GET /api/broker/get/all_chemicals Get All Chemicals

GET /api/broker/get/all_compounds Get All Compounds

GET /api/broker/get/measurement Get Measurement By Id

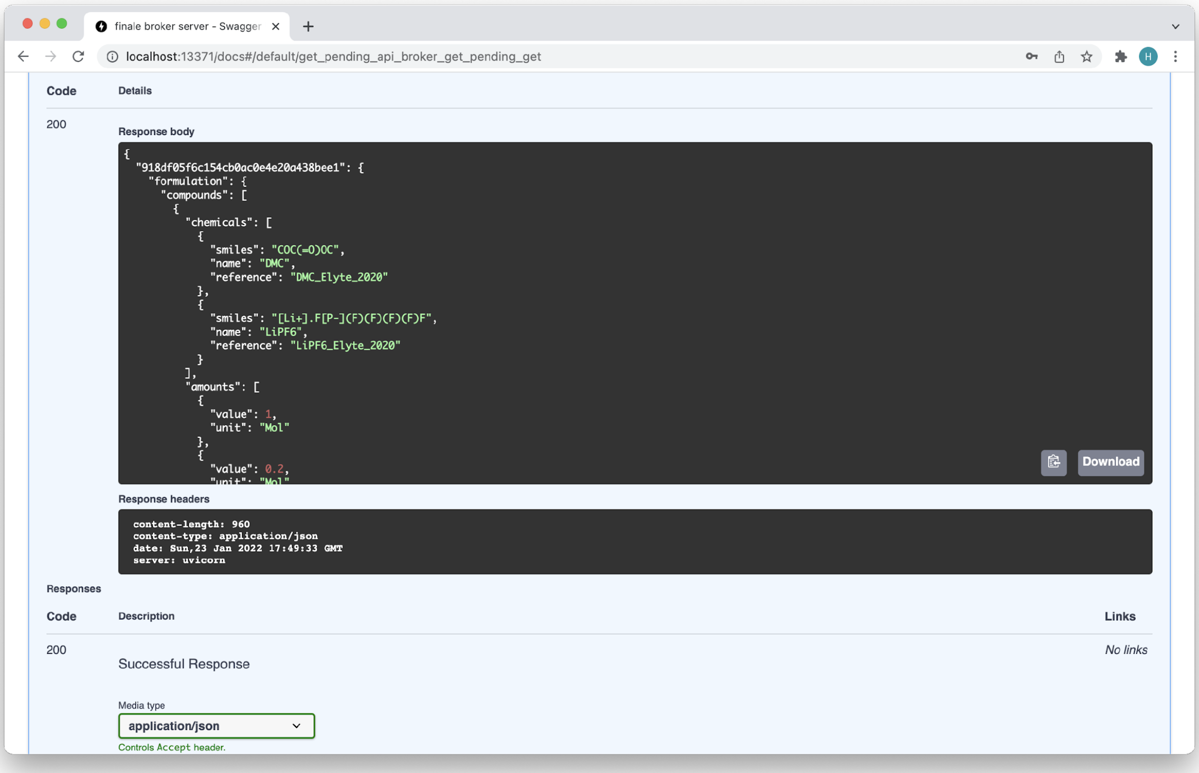
GET /api/broker/get/all_fom All Fom

GET /api/broker/get/pending Get Pending

POST /api/broker/post/measurement Post Measurement

POST /api/broker/request/measurement Request Meas

Figure 14. Broker server of *finales* running locally (deployed through docker, also available online at stein.hiu.batteries.de:13371) showing the different API calls possible. The framework was written with strong authentication (industry-standard OAuth2 workflow with jwt tokens), allowing fine-grained access control. All API functions are documented and adhere to schemas compatible with BattInfo



finale broker server - Swagger

localhost:13371/docs#/default/get_pending_api_broker_get_pending_get

Code | Details

200

Response body

```
{
  "918df05f6c154cb0ac0e4e20a438bee1": {
    "formulation": {
      "compounds": [
        {
          "chemicals": [
            {
              "smiles": "COC(=O)OC",
              "name": "DMC",
              "reference": "DMC_Elyte_2020"
            },
            {
              "smiles": "[Li+].F[P-](F)(F)(F)(F)F",
              "name": "LiPF6",
              "reference": "LiPF6_Elyte_2020"
            }
          ],
          "amounts": [
            {
              "value": 1,
              "unit": "Mol"
            },
            {
              "value": 0.2,
              "unit": "Mol"
```

Response headers

```
content-length: 960
content-type: application/json
date: Sun,23 Jan 2022 17:49:33 GMT
server: uvicorn
```

Responses

Code	Description	Links
200	Successful Response	*No links*

Media type

application/json

Controls Accept header.

Figure 15. Example of a pending measurement from the broker server returned as an ID key measurement pair

4. Uncertainty quantification and deployment to industrial partners

An optimal data-informed approach to allocate time and resources regardless of experiment or a simulation to solve is important for the acceleration of battery materials R&D at BASF. Lead times, resources, and practical accuracy (or relevance to the business challenge) differ dramatically between simulation and experiment. Such a data-informed approach, however, is still lacking in our R&D workflow, and thus the models developed herein are currently under assessment to be deployed in production at BASF.

The proposed active learning approach incorporating both theory and experiment has the potential to make better use of computational resources, reduce experimental effort, and speed up R&D. BASF envisions accelerating several steps in the development of cathode active materials by combining carefully planned physicochemical simulations and targeted experiments. The herein proposed framework would constitute a game-changing tool for accelerating research.

In order to expand the active learning usage scenarios beyond electrolyte formulation studies, CIDETEC shared a Python script to analyze electrochemical data. CIDETEC provided an intuitive script, which allows physicochemical parameter extraction from the usual GITT test. These parameters are an essential part of continuous models, such as DFN. Precisely, this Python tool solves the Fick diffusion equation via an analytic approach.

5. Summary and outlook

In summary, this deliverable report contains demonstrations of active learning on pre-generated experimental data, the first demonstrations of hardware in the loop active learning using mathematical functions with non-periodicity and multiple local minima, and first multi-modal, multi-fidelity optimization efforts and frameworks where experiments trigger simulations and vice versa, as shown in Figure 16. This is only possible through the interaction of work packages from the entire breadth of the BIG-MAP project highlighting the necessity for a broad interdisciplinary project like this. Together with the strong interaction between WP11, WP9 and WP10, we have demonstrated, for the first time, a seamless linking of experiments, modelling, machine learning, and humans in one joint optimization workflow.

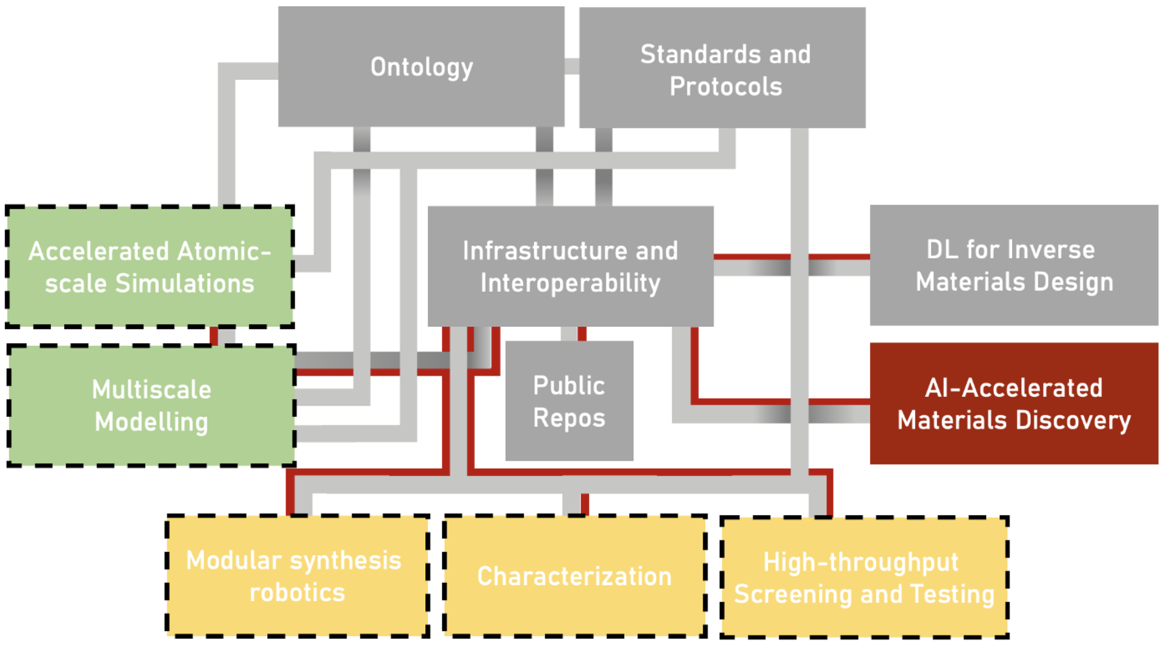

Ontology
Standards and Protocols
Accelerated Atomic-scale Simulations
Infrastructure and Interoperability
DL for Inverse Materials Design
Multiscale Modelling
Public Repos
AI-Accelerated Materials Discovery
Modular synthesis robotics
Characterization
High-throughput Screening and Testing

Figure 16. Work packages in BIG-MAP. Red-lined connections are those directly used by WP 10. Color highlighted work packages directly benefit from WP10 as an accelerator. Grey-colored work packages act as intermediaries allowing for streamlined communication.