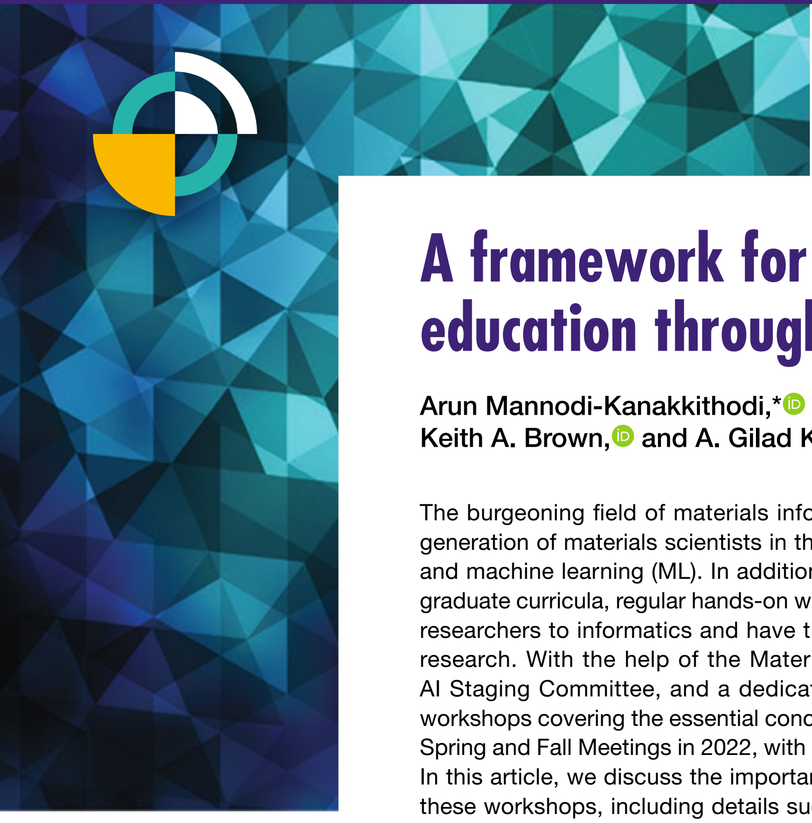


A framework for materials informatics education through workshops

Arun Mannodi-Kanakkithodi,* Austin McDannald, Shijing Sun, Saaketh Desai, Keith A. Brown, and A. Gilad Kusne

The burgeoning field of materials informatics necessitates a focus on educating the next generation of materials scientists in the concepts of data science, artificial intelligence (AI), and machine learning (ML). In addition to incorporating these topics in undergraduate and graduate curricula, regular hands-on workshops present the most effective medium to initiate researchers to informatics and have them start applying the best AI/ML tools to their own research. With the help of the Materials Research Society (MRS), members of the MRS AI Staging Committee, and a dedicated team of instructors, we successfully conducted workshops covering the essential concepts of AI/ML as applied to materials data, at both the Spring and Fall Meetings in 2022, with plans to make this a regular feature in future meetings. In this article, we discuss the importance of materials informatics education via the lens of these workshops, including details such as learning and implementing specific algorithms, the crucial nuts and bolts of ML, and using competitions to increase interest and participation.

Importance of materials informatics education

Materials discovery has enormously benefited from the application of state-of-the-art machine learning (ML) and artificial intelligence (AI) methods to materials data sets, both historical and newly generated. Combining large quantities of materials data with AI/ML algorithms to develop predictive models and optimization schemes, perform high-throughput screening, and automate or accelerate experiments and computations in general, has come to be known as materials informatics. Many successful discoveries of new battery electrolytes, solar-cell absorbers, catalysts, thermoelectrics, dielectrics, and high-entropy alloys, among myriad other materials applications, are thanks to informatics-based approaches.[1] Informatics has become an indispensable tool to both computational and experimental materials scientists looking to make the next big materials discovery. The field has matured to such an extent that well-established researchers feel the need to embrace these methods even late in their careers, while beginners have been tasked with learning AI/ML concepts during their materials science education.

Although some roadblocks remain to fully incorporate materials informatics into mainstream curriculum, most materials scientists recognize the importance of teaching students best data practices, ways to extract qualitative and quantitative relationships from data, and getting comfortable with programming languages such as Python. The barrier to applying AI/ML toward materials research is lower than ever before, with the sheer number of tools and resources available online. For instance, the Scikit-learn Python library contains essentially every possible ML regression, classification, clustering, and dimensionality reduction algorithm, any of which can be easily applied to suitably structured input–output data.[2] With just a few lines of code, researchers can call necessary functions and train supervised ML models based on random forests, ridge regression, neural networks, and other regression algorithms, as well as immediately elevate their work using sophisticated data visualization and processing techniques. Another source providing ready access to materials informatics tools is repositories hosted on GitHub and similar services such as nanoHUB,[3] typically containing reactive code notebooks such as Jupyter or Pluto notebooks,[4,5] Python scripts,

Arun Mannodi-Kanakkithodi, Materials Engineering, Purdue University, West Lafayette, USA; amannodi@purdue.edu
Austin McDannald, Materials Measurement Science Division, National Institute of Standards and Technology, Gaithersburg, USA; austin.mcdannald@nist.gov
Shijing Sun, Toyota Research Institute, Los Altos, USA; shijing.sun@tri.global
Saaketh Desai, Center for Integrated Nanotechnologies, Sandia National Laboratories, Albuquerque, USA; saadesa@sandia.gov
Keith A. Brown, Boston University, Boston, USA; brownka@bu.edu
A. Gilad Kusne, Materials Measurement Science Division, National Institute of Standards and Technology, Gaithersburg, USA; aaron.kusne@nist.gov
*Corresponding author
doi:10.1557/s43577-023-00531-6

and executables that can be adapted and applied toward any new materials data sets.

Materials informatics workshops

In their recent article, Sun et al.[6] talk about the lessons learned from conducting an ML workshop for a materials science audience. As the successor to the workshop discussed in their article, we organized and successfully conducted an in-person hands-on tutorial titled “DS00: Machine Learning in Materials Sciences – From Basic Concepts to Active Learning,” at the Materials Research Society (MRS) Spring Meeting in Honolulu in May 2022, as well as at the MRS Fall Meeting in Boston in November 2022. The idea behind making such a tutorial a necessary and recurring feature of all MRS meetings emerged from extensive discussions between members of the MRS AI Staging Committee, which consists of several leading materials informatics researchers and is tasked with “establishing and promoting AI as a crosscutting research topic within MRS.” The tutorial provided a basic introduction to concepts of ML, covered some examples of how it is applied for materials design, and went through several detailed case studies using reactive code notebooks, which were accessible to all of the audience members who were encouraged to execute code snippets in parallel with the instructors. Despite challenges posed by the COVID-19 global pandemic and conferences still slowly building up to pre-2020 attendance, the tutorial was positively received, with between 50 and 100 people in attendance at its peak at both meetings, which has helped establish a baseline for how to conduct such introductory workshops and how best to interactively teach essential concepts of data visualization, feature generation, model training and testing, and new predictions for discovery. **Figure 1** shows pictures from the workshop conducted at the 2022 MRS Fall Meeting.

Figure 1. Snapshots from the workshop held at the MRS Fall Meeting in November 2022, at peak audience in the morning (top) and at the end of the day (bottom).

Given the fact that tutorial days at MRS meetings are generally “day 0” before the official technical symposium sessions begin, and the fact that there are many other tutorials going on at the same time, the audience for tutorial DS00 constituted a satisfactory number. What was also striking was the breakdown of audience members: there were arguably as many experienced researchers as there were beginner-level students, with an equal degree of enthusiasm among all, as gauged by their active participation in opening and running through the reactive code notebooks, asking questions, and engaging in meaningful discussion during the dedicated Q&A periods, and even taking part in the ML challenge at the end of the day to apply what they learned during the tutorial and give themselves a shot at finishing at the top of a results leaderboard (discussed in the “Raising curiosity and confidence via competitions” section). Some of the audience members were indeed researchers already applying informatics-based approaches in their work, looking to consolidate their knowledge and interact with like-minded people. While the daylong tutorial is meant to be no more than a crash course in applying AI/ML toward materials science problems, it provides a vital boost to both beginners and active researchers on (a) the importance and inevitability of materials informatics, and whether/how the underlying approaches apply to their work; (b) how to get started and become comfortable with specific algorithms, keeping in mind best data and software practices; and (c) how to incorporate materials informatics education within their own research groups and institutions. In “Nuts and bolts of ML,” “Teaching via reactive code notebooks,” “Demonstrating essential concepts using an example: ML-driven perovskite discovery,” “Neural networks,” “Gaussian processes and active learning,” and “Raising curiosity and confidence via competitions” sections, we discuss some details of our ML workshops, which followed a general technical outline presented in **Figure 2**. Some of the topics of discussion include the nuts and bolts of ML that all researchers need to be familiar with, the advantages of using reactive code notebooks, specific approaches such as neural networks and active learning, and finally, the importance of raising curiosity and incentivizing the use of competitions.

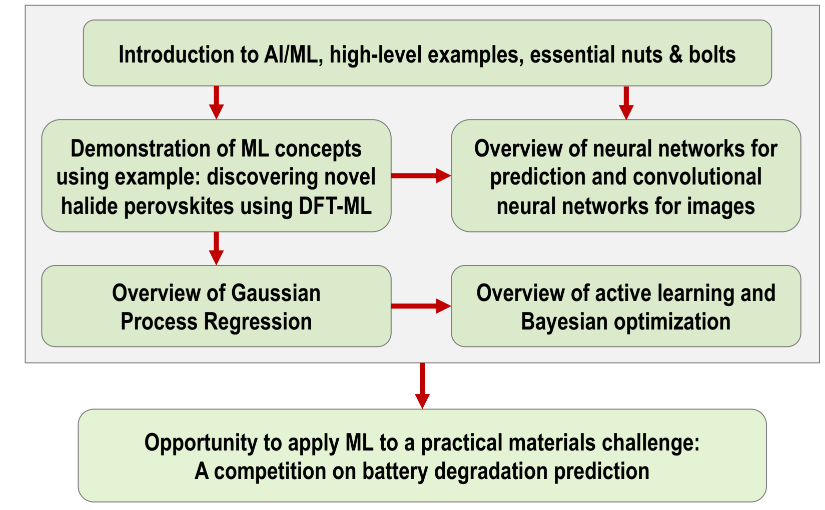

Introduction to AI/ML, high-level examples, essential nuts & bolts
Demonstration of ML concepts using example: discovering novel halide perovskites using DFT-ML
Overview of neural networks for prediction and convolutional neural networks for images
Overview of Gaussian Process Regression
Overview of active learning and Bayesian optimization
Opportunity to apply ML to a practical materials challenge: A competition on battery degradation prediction

Figure 2. An outline of our materials informatics workshop. DFT, density functional theory.

Nuts and bolts of ML

The simplest version of applying ML to materials is when an input–output materials data set (inputs being easily attainable descriptors, outputs are properties of interest) is compiled and a supervised or unsupervised learning algorithm is applied to obtain some kind of a predictive model that can be deployed on many unseen datapoints. To this end, there are a number of essential concepts, referred to here as the "nuts and bolts of ML," which need to be understood, and were the subject of discussions in the early part of the tutorial. These nuts and bolts include:

- Materials data: Perhaps the most important component of a materials informatics framework is the data set. This could be computational or experimental data, it may range from a few dozen to hundreds or even millions of datapoints depending on the problem, and it could be obtained from databases such as the Materials Project[7] or be freshly generated. The ability to procure reasonably accurate and sufficiently large quantities of data is key to ML predictive power and utility.
- Materials descriptors: An enormous body of work exists on critical examination of various types of features, fingerprints, or descriptors that can be used to effectively represent materials in a data set structured for ML.[8,9] Such descriptors must be easy to obtain (or learn), uniquely defined for each material, and generalizable across the entire chemical space. Common examples include using one-hot encoded composition vectors, tabulated elemental properties such as ionic radius and electronegativity, Coulomb matrices, atomic overlap functions, and cheaper computational information. Descriptors can also be learned using ML methods, especially for high-dimensional data sets such as images.
- Data pre-processing: Once the input (descriptors) - output (properties) data are compiled, some operations are necessary before ML tasks could be performed. These include straightforward visualization of inputs versus outputs, standardization and normalization of the inputs, dimensionality reduction techniques, and lower-dimensional projection and visualization of data using methods such as principal component analysis (PCA) and t-SNE.[10] These exercises help us gain an intuition for our data sets, setting prior expectations of the ML models subsequently developed.
- Supervised versus unsupervised learning: Depending on the problem at hand, either type of approach could be applicable. Whereas supervised learning techniques find unknown functions connecting inputs to outputs from labeled data based on interpolation and extrapolation of patterns, unsupervised learning aims to find patterns in unlabeled data (i.e., "outputs" do not exist) to yield a clustering of samples.[11] A typical workflow of a supervised learning algorithm is presented in **Figure 3**, where many of the highlighted components are the same as the ML nuts and bolts being discussed here.
- ML algorithm: The choice of specific ML regression, classification, or optimization algorithm, although less significant than the quality of the data and descriptors, is still quite important. Typical studies involve testing of multiple algorithms before the best one is selected, or the use of an algorithm that worked well on similar data sets in the past. Neural networks work best with large data sets where physical understanding of contributions from specific features may not be key, and is frequently applied for predicting atomic forces and formation energies of solids and molecules. Methods such as random forest yield feature importance values, while Gaussian process regression helps obtain reliable uncertainty estimates along with predictions.[12] There are a whole host of other methods that have been successfully applied on materials data sets and readily available via Scikit-learn and other packages.
- Training, validation, and testing: In practice, the data set is split into training and testing sets, with the training set used exclusively for optimizing the ML model in terms of parameters and accuracy of prediction, whereas the testing set serves as a true test of the model on unseen datapoints. The training set itself is split into multiple validation sets in a process that is called cross-validation, wherein model parameters are chosen so as to minimize error in prediction on subsets of the training set over multiple cycles of splitting the data, training then validating. Furthermore,

one would visualize the effect of training data size using learning curves, which show the test prediction error as a function of the number of datapoints used to train and optimize the model, in an attempt to determine the saturation point. Such curves could also be made between number (or combinations therein) of input features or algorithm parameters and the error.

- Hyperparameter optimization: Every ML algorithm has a set of "hyperparameters" that need to be optimized for best predictive performance. For instance, hyperparameters in random forest regression include the number of trees in the forest, maximum depth of each tree, number of features to consider when looking for the best node split, and the minimum number of samples required to be at a leaf node. Typically, finding the optimal set of all parameters is done by employing a grid-search approach where the algorithm cycles through many combinations, but there are more efficient ways to do this, such as via Bayesian optimization using a function set to minimize the validation error.[13]
- New discovery and understanding: The ultimate objective of applying ML to materials data is to attain quantitative or qualitative models that can reveal atom/composition/structure–property relationships that were previously unknown, and enable predictions over a combinatorial space, which can then be used for screening and discovery.[12–14] Once rigorously optimized, ML models need to be deployed for new discovery and understanding, which may involve enumerating new datapoints and making predictions, performing inverse design using approaches such as genetic algorithm or generative neural networks[15] to obtain new materials with desired properties, and generally making best models available to be coupled with ongoing computations or experiments, often within an active learning framework[16] for continuous improvement and discovery.

Teaching via reactive code notebooks

Reactive code notebooks, such as Jupyter and Pluto,[4,5] represent some of the most frequently used web-based computing environments, which allow the user to author the notebook, write and run their code interactively, and add any number of details in the form of markdowns. Such notebooks are often self-sufficient and independent of related publications, data sets, or codes. They allow the inclusion of images, equations, narrative texts, plots, and videos, each designed and inserted at specific locations throughput the notebook to explain different components of the overall code and project. The reactive code notebook allows users to edit and run code from right within their browsers, and perform live visualization of results as tabulated values, plots, or in other formats. Notebooks contain kernels that allow executing code in different programming languages such as MATLAB and Python, as well as different types of cells that include live code, headings, markdowns, and raw information.

It is no wonder that one (or multiple) all-encompassing reactive code notebook(s) is the preferred medium for instructors conducting a hands-on AI/ML workshop. Specific benefits include the following: (a) all code and underlying information can be shared with the audience via public repositories such as on GitHub, with a universal url accessible to anybody; (b) the notebook (and associated data files) could either be downloaded to one's personal machine and run locally (assuming all code dependencies are installed and up to date), or be opened using an application such as Colaboratory[17] to be run on Google's processors, ensuring that attendees can run the code (alongside the instructor) on their own laptops, making local edits or updates as and when needed; (c) attendees can easily copy particular code cells to other notebooks and adapt them toward their research projects; and (d) community-informed updates can periodically be made to the notebook, such that it is improved and enhanced every time it is used for a tutorial. It should be noted that today, in materials informatics circles, it has become common practice to release such notebooks as supporting documents alongside journal publications or conference presentations, which goes a long way toward ensuring that materials data remain FAIR (findable, accessible, interoperable, and reusable).[18] The fact

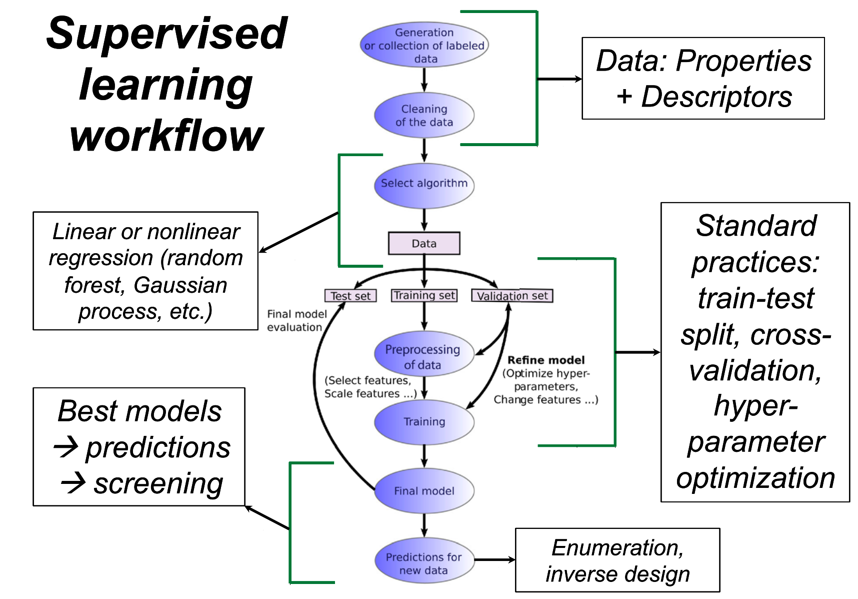

Supervised learning workflow
Generation or collection of labeled data
Cleaning of the data
Data: Properties + Descriptors
Select algorithm
Linear or nonlinear regression (random forest, Gaussian process, etc.)
Data
Test set
Training set
Validation set
Final model evaluation
Preprocessing of data
(Select features, Scale features ...)
Refine model
(Optimize hyper-parameters, Change features ...)
Standard practices: train-test split, cross-validation, hyper-parameter optimization
Training
Best models → predictions → screening
Final model
Predictions for new data
Enumeration, inverse design

Figure 3. A typical supervised learning workflow, showing the various nuts and bolts of machine learning discussed in this article. Figure adapted with permission from Springer Nature.[1]

that the notebooks can as well be used to educate students and other researchers on the important aspects of the research project is an added bonus.

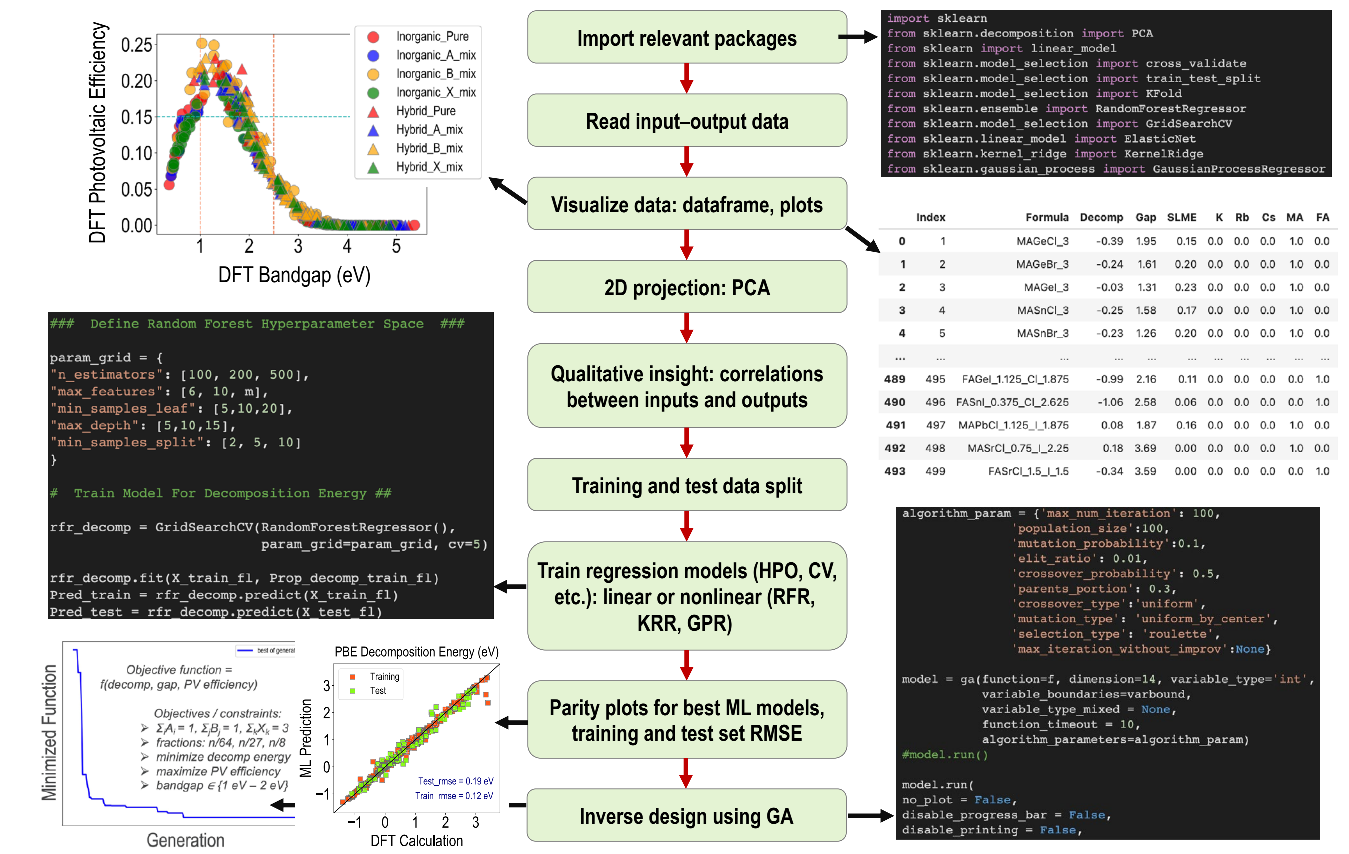
Demonstrating essential concepts using an example: ML-driven perovskite discovery

An "ML for materials science" workshop can take many different forms depending on the subject matter it aims to cover. It could, for instance, be focused on something very specific such as understanding and applying graph-based neural networks for materials discovery,[19] accelerating molecular dynamics via ML-predicted atomic forces and energies,[20] or performing automated and efficient experimental discovery using active learning.[21] In an introductory/overview type of workshop, our focus is on using an intuitive, easy-to-understand, and preferably published example that involves the use of the specific concepts covered in the first part of the tutorial: collecting materials data, obtaining or learning descriptors, processing, optimizing, and visualizing results from various learning approaches, and deploying ML models for new discovery. As such, we opted for a project involving the discovery of novel halide perovskite alloys for optoelectronic applications, using ML models trained upon a substantial computational data set of relevant properties.[14,22]

A Jupyter notebook laying out this entire project can be found on GitHub[23] as well as on nanoHUB;[24] this includes all code necessary for reading and visualizing the data (properties + different types of descriptors), performing clustering and examining correlations, training different types of regression models, and coupling the best predictive models with genetic algorithm to discover new perovskite compositions that show a targeted set of properties. In addition, several boxes of descriptive text and images detail all the technical content, motivation, methodology, and primary results/observations of the project. This work involved performing high-throughput density functional theory (DFT) computations on a data set of ≈500 compounds, yielding their energies of decomposition (measure of stability), electronic bandgaps, and theoretical photovoltaic efficiencies. To determine suitable semiconductors for use as absorbers in solar cells or in related optoelectronic applications such as LEDs, lasers, photodiodes, and sensors, one must screen materials that are stable, have a suitable bandgap within the visible spectrum, and a high absorption efficiency. The computational expense of accurate DFT (hours to days of HPC time needed for a single compound) and the very combinatorial nature of the ABX_3 perovskite chemical space (where A and B are cations while X is a halogen anion) given the number of options for A/B/X species and prospects of alloying at each site, DFT cannot endlessly be performed for millions of candidates, and needs to be coupled with ML models that can make predictions in mere seconds. Thus, using one-hot encoded vectors of the ABX_3 composition as well as elemental properties such as electron affinities and ionic radii of the A/B/X species, we trained predictive models using various linear and nonlinear regression algorithms, leading to highly accurate predictions of the three properties of interest for any number of new perovskite compositions.

A detailed outline of the perovskite DFT-ML notebook is provided in **Figure 4** as a flowchart, along with accompanying screenshots of code and plots showing some of the results. The workshop provided the attendees with a brief background of this project, introducing the data set and ML approaches involved, before delving into details with the help of the notebook. The notebook begins with a description of the project, references to relevant publications, and acknowledgment of the authors who contributed to it, followed by an introductory code cell that imports all necessary Python packages. The data are next read from CSV (comma separated value) files and formatted as a pandas dataframe[25] that includes necessary compound labels, computed properties, and all descriptor dimensions. One of the concepts explored in this work is "multi-fidelity learning,"[26] which involves generating data from two different levels of theory within DFT, namely the semilocal GGA-PBE functional (or PBE),[27] which is cheaper but has low accuracy compared to experiment, and the hybrid HSE06 functional (or HSE),[28] which is more expensive but also more accurate; while the PBE data set contains nearly 500 points, the HSE data set contains half of that. The next code cells present a visualization of the two data sets in terms of one property plotted against another, a PCA-based 2D projection, and Pearson coefficients of linear correlation calculated between descriptor dimensions and properties, thus providing qualitative insights into the effect of different A/B/X species and their specific elemental properties on increasing or decreasing the perovskite stability and optoelectronic properties.

The next code cells involve splitting the data randomly into training and test sets, followed by rigorous training of three types of regression models individually for each property: elastic net regression (ENR, representing a linear model), random forest regression (RFR), and kernel ridge regression (KRR). The latter two techniques represent some of the most commonly used nonlinear regression algorithms in materials informatics. The training process for each algorithm involves defining a grid of hyperparameters, optimizing them over the training set based on minimizing the validation error from a fivefold cross-validation technique, and making final predictions for all training and test set datapoints. The subsequent code cell visualizes model performance in the form of parity plots showing ML predictions on the *y*-axis and actual DFT ground truth on the *x*-axis, as well as root mean square errors (RMSEs) on the training and test set. It should be noted here that the choice of ENR, RFR, and KRR are arbitrary and simply for demonstrating larger concepts, whereas the RMSE is only one type of error metric that was used here, but could easily be replaced by the mean absolute error or R^2 value. Regression models are trained for the PBE and HSE data sets individually, as well as for the PBE and HSE data sets combined into one: the latter uses the principles of multi-fidelity learning,


DFT Photovoltaic Efficiency
0.25
0.20
0.15
0.10
0.05
0.00
1
2
3
4
5
DFT Bandgap (eV)
Inorganic_Pure
Inorganic_A_mix
Inorganic_B_mix
Inorganic_X_mix
Hybrid_Pure
Hybrid_A_mix
Hybrid_B_mix
Hybrid_X_mix
Import relevant packages
Read input–output data
Visualize data: dataframe, plots
2D projection: PCA
Qualitative insight: correlations between inputs and outputs
Training and test data split
Train regression models (HPO, CV, etc.): linear or nonlinear (RFR, KRR, GPR)
Parity plots for best ML models, training and test set RMSE
Inverse design using GA
Minimized Function
best of generat
Objective function = f(decomp, gap, PV efficiency)
Objectives / constraints:
➢ $\Sigma_i A_i = 1$, $\Sigma_j B_j = 1$, $\Sigma_k X_k = 3$
➢ fractions: n/64, n/27, n/8
➢ minimize decomp energy
➢ maximize PV efficiency
➢ bandgap ∈ {1 eV – 2 eV}
Generation
PBE Decomposition Energy (eV)
Training
Test
ML Prediction
3
2
1
0
−1
−1
0
1
2
3
DFT Calculation
Test_rmse = 0.19 eV
Train_rmse = 0.12 eV

```
import sklearn
from sklearn.decomposition import PCA
from sklearn import linear_model
from sklearn.model_selection import cross_validate
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import ElasticNet
from sklearn.kernel_ridge import KernelRidge
from sklearn.gaussian_process import GaussianProcessRegressor
```

	Index	Formula	Decomp	Gap	SLME	K	Rb	Cs	MA	FA
0	1	MAGeCl_3	-0.39	1.95	0.15	0.0	0.0	0.0	1.0	0.0
1	2	MAGeBr_3	-0.24	1.61	0.20	0.0	0.0	0.0	1.0	0.0
2	3	MAGeI_3	-0.03	1.31	0.23	0.0	0.0	0.0	1.0	0.0
3	4	MASnCl_3	-0.25	1.58	0.17	0.0	0.0	0.0	1.0	0.0
4	5	MASnBr_3	-0.23	1.26	0.20	0.0	0.0	0.0	1.0	0.0
...	...	...	...	...	...	...	...	...	...	...
489	495	FAGeI_1.125_Cl_1.875	-0.99	2.16	0.11	0.0	0.0	0.0	0.0	1.0
490	496	FASnI_0.375_Cl_2.625	-1.06	2.58	0.06	0.0	0.0	0.0	0.0	1.0
491	497	MAPbCl_1.125_I_1.875	0.08	1.87	0.16	0.0	0.0	0.0	1.0	0.0
492	498	MASrCl_0.75_I_2.25	0.18	3.69	0.00	0.0	0.0	0.0	1.0	0.0
493	499	FASrCl_1.5_I_1.5	-0.34	3.59	0.00	0.0	0.0	0.0	0.0	1.0

```
###  Define Random Forest Hyperparameter Space  ###

param_grid = {
"n_estimators": [100, 200, 500],
"max_features": [6, 10, m],
"min_samples_leaf": [5,10,20],
"max_depth": [5,10,15],
"min_samples_split": [2, 5, 10]
}

#  Train Model For Decomposition Energy ##

rfr_decomp = GridSearchCV(RandomForestRegressor(),
                          param_grid=param_grid, cv=5)

rfr_decomp.fit(X_train_fl, Prop_decomp_train_fl)
Pred_train = rfr_decomp.predict(X_train_fl)
Pred_test = rfr_decomp.predict(X_test_fl)
```

```
algorithm_param = {'max_num_iteration': 100,
                   'population_size':100,
                   'mutation_probability':0.1,
                   'elit_ratio': 0.01,
                   'crossover_probability': 0.5,
                   'parents_portion': 0.3,
                   'crossover_type':'uniform',
                   'mutation_type': 'uniform_by_center',
                   'selection_type': 'roulette',
                   'max_iteration_without_improv':None}

model = ga(function=f, dimension=14, variable_type='int',
           variable_boundaries=varbound,
           variable_type_mixed = None,
           function_timeout = 10,
           algorithm_parameters=algorithm_param)
#model.run()

model.run(
no_plot = False,
disable_progress_bar = False,
disable_printing = False,
```

Figure 4. Outline of the reactive code notebook on the perovskite design work, showing some code snippets and density functional theory (DFT)/machine learning (ML)/genetic algorithm (GA) plots. PCA, principal component analysis; RFR, random forest regression; KRR, kernel ridge regression; GPR, Gaussian process regression; RMSE, root-mean-square error; HPO, hyperparameter optimization; CV, cross-validation; PV, photovoltaic.

wherein correlations between properties from the two levels of theory and better predictions for the larger PBE data set are exploited to improve the prediction accuracy for HSE.

The punchline of the notebook comes in the form of how the best predictive models could be used for discovering promising new candidates. This has been shown in two different ways:

- Enumeration of thousands of new compounds followed by prediction of their properties and screening based on suitable property values (e.g., a negative decomposition energy and bandgap between 1 and 2.5 eV). A markdown in the notebook that follows the ML models presents graphics showing screening more than ≈18,000 compounds leading to 392 materials that satisfy multiple objectives of interest.[14]
- Performing inverse design by exploiting the DFT-ML surrogate models to iteratively produce new candidates that take you closer to the desired set of properties; this is demonstrated here using a genetic algorithm (GA),[29] where starting from a random population of perovskite compositions, new generations are continuously produced, using operations such as elitism, crossover, and mutation, so as to minimize a multi-objective function that ensures chemical feasibility (maintain stoichiometry, fractions of multiple components at A/B/X sites must sum to 1), a negative decomposition energy, bandgap between 1 and 2.5 eV, and a PV efficiency as high as possible. The GA framework includes parameters that can be tuned and models can be run for any number of generations, at the end of which a list of best candidates is obtained.

Enumeration and screening as well as the GA-based design is performed using both PBE- and HSE-level estimates, with knowledge of the general accuracies of both functionals used for determining the suitable property ranges. Final cells of the notebook present a visualization of the GA results in terms of the objective function plotted against the GA generation as well as the chemical constituents occurring at A/B/X sites in the best performing perovskite compositions. Subsequently, the same perovskite data set is utilized to demonstrate neural network regression as described in the "Neural networks" section. The many concepts explored and explained through this notebook – types of descriptors (compositional versus physical properties), visualization and clustering of data, extracting correlations, splitting data and optimizing regression models, etc. – would eventually serve the materials informatics challenge provided to all attendees at the final session of the tutorial.

Neural networks

Building on the foundations laid in the first session of the tutorial, the next reactive code notebook introduces the audience to neural network regression, utilizing the same perovskite data set that the audience is now familiar with. The inputs again are descriptors that encode compositional information and elemental properties, and the output is chosen to be the DFT predicted bandgap at the PBE level. This notebook uses the TensorFlow[30] and Keras[31] libraries to help audiences quickly set up and train neural networks, with minimal code. The notebook begins by obtaining the raw data set afresh as a CSV file, and next, the data are converted to a Pandas dataframe and “featurized” as in the previous notebook, creating input and output columns. The next code cell carves a train, validation, and test set from the dataframe, reiterating to the audience the importance of these data sets. Following this, the notebook sets up a simple neural network with a few hidden layers, with the instructor encouraging the audience to try out their own variations to this simple architecture. The flexibility of reactive code notebooks allows each audience member to rapidly get comfortable with trying different neural network architectures. After training the model, subsequent code cells visualize the training via learning curves and check for common neural network training pitfalls such as overfitting. Finally, the neural network makes PBE bandgap predictions on test data, encouraging the audience to compare their predictions against previous regression models used in the workshop. A snippet of the neural network code is presented in **Figure 5**a.

Following the regression demonstration, we slightly change tracks and introduce the audience to a classification problem that is solved using neural networks. This problem uses images as inputs, and is aimed at materials scientists working with microstructural data such as electron/optical microscropy images or diffraction data, introducing them to the idea of learning from image data using neural networks. To demonstrate image classification using neural networks, the reactive code notebook uses the FashionMNIST[32] data set, which consists of thousands of clothing images, and attempts to classify them into their correct categories (T-shirts, trousers, boots, etc.). Because this data set is new to the audience, the instructor spends some time introducing and pre-processing the data, making sure the inputs and outputs are clear. Then, the concept of “convolution operations” to learn features from images is illustrated, before showing the code for a neural network composed of convolutional layers. Once again, audience members are encouraged to try out their own variations of the network architecture by modifying parameters. As with the regression tutorial, the model is trained, after which the notebook visualizes learning curves, and makes predictions on the test data set.

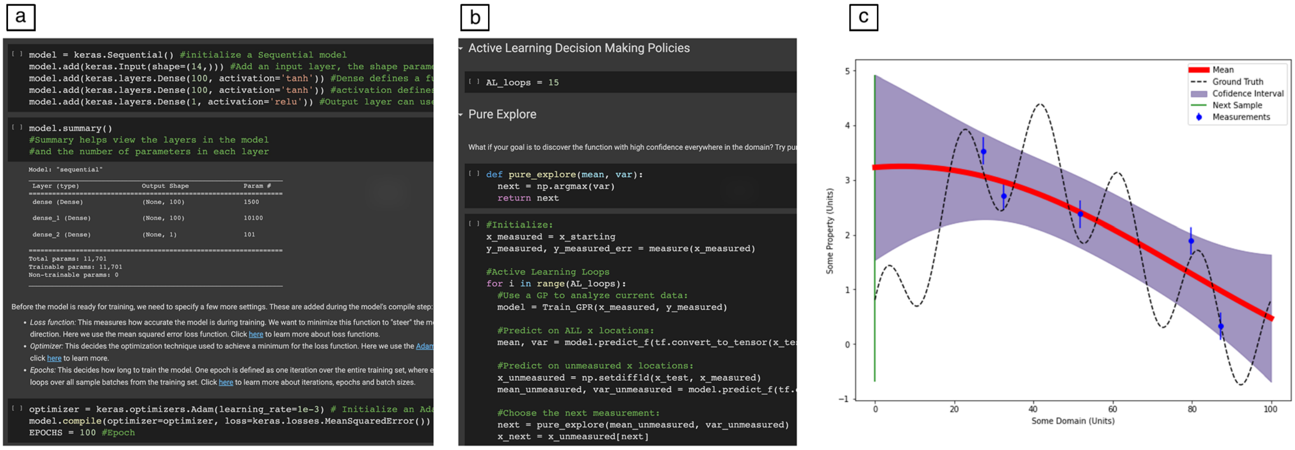
These Jupyter notebooks can be found on nanoHUB,[24] along with other basic machine learning tutorials.[33]

Gaussian processes and active learning

A separate reactive code notebook was used to introduce the audience to Gaussian processes (GPs), a Bayesian technique.[23] The goal of Bayesian methods such as GPs is to robustly determine the following: given the data so far and some knowledge of the problem, what can be concluded about new locations in the domain? GPs are particularly well suited to smaller data sets, such as could be encountered in experimental measurements or computationally intensive simulations. GPs natively output uncertainty information with the predictions, which make them natural choices for “active learning” campaigns, where the objective is to iteratively add new datapoints to minimize overall uncertainty and maximize reward. This Jupyter notebook tutorial aims to provide intuitions about the implications of how the GPs are used, so that they can be applied in the next reactive code notebook on active learning (AL) for autonomous experimental materials science.

The notebook presents GPs as a collection of related probability distributions over the input domain. Knowledge about the problem (the priors in Bayesian inference parlance) is specified by the design of the mean and kernel (also called covariance) functions. The mean function depends on the global position in the domain (e.g., a function such as $f(x) = x^2$). In contrast, the kernel functions only depend on the relative similarity between locations in the input domain. The next cells present the implications of choosing some popular kernels. For example, using the radial basis function kernel implies that the user expects the function to be smooth (i.e., infinitely differentiable). However, choosing the Matern $\frac{3}{2}$ kernel would imply that discontinuous jumps are expected in the slope of the function. The notebook concludes with an illustration of how GPs could be used for regression. In addition to executing code alongside the instructor, the workshop attendees were encouraged to use the notebook for generating example data and training regression models with different choices of kernels and hyperparameters. Further examples are provided showing common pitfalls and demonstrating how to propagate the measurement uncertainty.

In the next section, a second notebook builds upon the GP background to demonstrate AL with a simulated autonomous experiment campaign. This notebook covers the basic concepts of several common decision-making policies that use both the predictions of the function of interest and the uncertainties in those predictions. Presenting this material as a notebook allows the attendees to run the simulated AL campaigns. They were encouraged to modify the AL policy and the model design, run the campaigns, and compare and contrast the obtained results. This notebook concludes by presenting several performance metrics for the success of the AL campaigns. These metrics evaluate the success of the AL campaigns toward the goals of either learning a global map of the function, or learning global optimum. A code snippet from



```
model = keras.Sequential() #initialize a Sequential model
model.add(keras.Input(shape=(14,))) #Add an input layer, the shape parame
model.add(keras.layers.Dense(100, activation='tanh')) #Dense defines a fu
model.add(keras.layers.Dense(100, activation='tanh')) #activation defines
model.add(keras.layers.Dense(1, activation='relu')) #Output layer can use
```

```
model.summary()
#Summary helps view the layers in the model
#and the number of parameters in each layer
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 100)               1500

 dense_1 (Dense)             (None, 100)               10100

 dense_2 (Dense)             (None, 1)                 101

=================================================================
Total params: 11,701
Trainable params: 11,701
Non-trainable params: 0
_________________________________________________________________
```

Before the model is ready for training, we need to specify a few more settings. These are added during the model's compile step:

- *Loss function:* This measures how accurate the model is during training. We want to minimize this function to "steer" the m direction. Here we use the mean squared error loss function. Click here to learn more about loss functions.
- *Optimizer:* This decides the optimization technique used to achieve a minimum for the loss function. Here we use the Adam click here to learn more.
- *Epochs:* This decides how long to train the model. One epoch is defined as one iteration over the entire training set, where e loops over all sample batches from the training set. Click here to learn more about iterations, epochs and batch sizes.

```
optimizer = keras.optimizers.Adam(learning_rate=1e-3) # Initialize an Ada
model.compile(optimizer=optimizer, loss=keras.losses.MeanSquaredError())
EPOCHS = 100 #Epoch
```

Active Learning Decision Making Policies

```
AL_loops = 15
```

Pure Explore

What if your goal is to discover the function with high confidence everywhere in the domain? Try pu

```
def pure_explore(mean, var):
  next = np.argmax(var)
  return next
```

```
#Initialize:
x_measured = x_starting
y_measured, y_measured_err = measure(x_measured)

#Active Learning Loops
for i in range(AL_loops):
  #Use a GP to analyze current data:
  model = Train_GPR(x_measured, y_measured)

  #Predict on ALL x locations:
  mean, var = model.predict_f(tf.convert_to_tensor(x_te

  #Predict on unmeasured x locations:
  x_unmeasured = np.setdiff1d(x_test, x_measured)
  mean_unmeasured, var_unmeasured = model.predict_f(tf.

  #Choose the next measurement:
  next = pure_explore(mean_unmeasured, var_unmeasured)
  x_next = x_unmeasured[next]
```

Figure 5. (a) Snippet of code from the neural network notebook. (b) Snippet of code from the active learning notebook. (c) Example plot showing the success of determining an unknown function by sampling new points using active learning.

the AL notebook and a plot showing results at a particular step of the AL campaign are presented in Figure 5b–c, respectively.

Raising curiosity and confidence via competitions

Organizing an in-class competition after the tutorials provides participants with hands-on experience to apply ML and data handling techniques introduced during the workshop, without answers provided. This learning format encourages participants to follow their curiosity and explore different ML methods, serving as a bridge to apply similar methods to their own research challenges. There are several advantages to hosting a data science competition along with workshops. First, it provides participants with a platform to receive timely feedback and benchmark their answers with others. Having completed a materials data challenge independently also boosts newcomers' confidence in further research in materials informatics. Launching the competition during or immediately after the tutorials allows for fresh memories, and an in-person Q&A session is particularly friendly for beginners, who may have questions such as installing a Python package or checking data downloading bottlenecks, which are easier to resolve when an instructor takes over the computer screen.

Both competitions we ran at MRS Fall Meeting Tutorial Days in 2021 and 2022 consisted of an introductory session explaining the data challenge's significance in materials science, followed by sharing the data and a Google Colab notebook to help participants get started. In the 2021 workshop, we simulated a data set where participants were asked to predict y values (materials property) using x values (composition) by applying supervised ML and AL techniques. In the 2022 workshop, the competition was hosted in partnership with the Toyota Research Institute and the University of Maryland. Participants were given an industry-relevant battery degradation challenge and asked to predict battery cycle life using an experimental battery testing data set from literature.[34] Through this partnership, MRS tutorial participants have the opportunity to be invited as finalists for the Battery Informatics & ML Hackathon, which is open to a wider audience, including battery researchers and ML professionals. Because the submission platform and live leaderboard were open to all, participants from diverse academic and research backgrounds can benchmark their solutions against each other. To achieve the educational goals of attracting more participants from diverse fields while ensuring the fairness of the competition, two separate Judges' awards were set up to reward best models received from MRS tutorial attendees and community participants, respectively.

Outlook and next steps

Informatics-based approaches are the norm in all materials design endeavors today, and the importance of educating materials scientists in essential concepts and practical aspects of AI/ML cannot be overstated. There are numerous virtual and in-person workshops being regularly organized by institutions and professional societies the world over to address gaps in materials informatics training. Furthermore, such topics have made their way into existing and new courses in many materials engineering departments, as it becomes imperative that students learn and apply data science methodologies to their research. Through this article, we attempted to shed light on why, how, and what we should be teaching beginners to put them on the path to becoming seasoned AI/ML practitioners. MRS conferences represent the foremost gathering of materials researchers from across the world, and there is no better venue to host an introductory workshop like what is described in detail in this article. This workshop is planned for many more upcoming meetings, always with a rolling list of instructors, and necessary updates and additions as the state of the art evolves and new techniques and capabilities come to light. Once again, we will present this tutorial at the 2023 MRS Spring Meeting and beyond as we continue to help normalize and popularize materials informatics education and training.

Acknowledgments

A.M.K. acknowledges support from the School of Materials Engineering at Purdue University under Account No. F.10023800.05.002. S.S. acknowledges A. Threwatha and S. Torrisi for setting up the challenge, data set, and notebook shared by the workshop at MRS and the battery informatics hackathon launched in November 2022, B. Storey for supporting materials informatics workshops, and R. Braatz for the fruitful discussions. S.D. acknowledges funding under the Beyond Fingerprinting Sandia Grand Challenge Laboratory Directed Research and Development (GC LDRD) program. Center for Integrated Nanotechnologies (CINT) is an Office of Science user facility operated for the US Department of Energy. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the US Department of Energy National Nuclear Security Administration under Contract No. DE-NA0003525.

Data availability

No new data sets were generated or analyzed during this study.

Conflict of interest

On behalf of all authors, A.M.K. states that there is no conflict of interest.

Disclaimer

The identification of any commercial product or trade name does not imply endorsement or recommendation by the National Institute of Standards and Technology (NIST). These opinions, recommendations, findings, and conclusions do not necessarily reflect the views or policies of NIST or the United States Government.

References

1. J. Schmidt, M.R.G. Marques, S. Botti, M.A.L. Marques, *NPJ Comput. Mater.* **5**(1), 83 (2019)
2. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, *J. Mach. Learn. Res.* **12**, 2825 (2011)
3. K. Madhavan, L. Zentner, V. Farnsworth, S. Shivarajapura, M. Zentner, N. Denny, G. Klimeck, *Nanotechnol. Rev.* **2**(1), 107 (2013)
4. D. Mejia, Jupyter Examples - Jupyter Notebook (202105) (2023). https://nanohub.org/resources/jupyterguide70
5. F. van der Plas, Interactive Julia Programming Environment (2023). https://github.com/fonsp/Pluto.jl
6. S. Sun, K. Brown, A.G. Kusne, *Matter* **5**(6), 1620 (2022)
7. A. Jain, S.P. Ong, G. Hautier, W. Chen, W.D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, K.A. Persson, *APL Mater.* **1**(1), 011002 (2013)
8. L. Himanen, M.O.J. Jäger, E.V. Morooka, F. Federici Canova, Y.S. Ranawat, D.Z. Gao, P. Rinke, A.S. Foster, *Comput. Phys. Commun.* **247**, 106949 (2020)
9. A. Seko, A. Togo, I. Tanaka, *Descriptors for Machine Learning of Materials Data* (Springer, Singapore, 2018), pp. 3–23
10. J. Pareek, J. Jacob, "Data Compression and Visualization Using PCA and t-SNE," in *Proceedings of Advances in Information Communication Technology and Computing (AICTC 2019)*, ed. by V. Goar, M. Kuri, R. Kumar, T. Senjyu (Springer, Singapore, 2021), pp. 327–337
11. T. Mueller, A.G. Kusne, R. Ramprasad, *Machine Learning in Materials Science* (Wiley, New York, 2016), pp. 186–273
12. A. Mannodi-Kanakkithodi, M. Chan, *J. Mater. Sci.* **57**, 10736 (2022)
13. A. Mannodi-Kanakkithodi, X. Xiang, L. Jacoby, R. Biegaj, S.T. Dunham, D.R. Gamelin, M.K. Chan, *Patterns* **3**(3), 100450 (2022)
14. A. Mannodi-Kanakkithodi, M.K.Y. Chan, *Energy Environ. Sci.* **15**, 1930 (2022)
15. R. Gómez-Bombarelli, J.N. Wei, D. Duvenaud, J.M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T.D. Hirzel, R.P. Adams, A. Aspuru-Guzik, *ACS Cent. Sci.* **4**(2), 268 (2018)
16. T. Lookman, P.V. Balachandran, D. Xue, R. Yuan, *NPJ Comput. Mater.* **5**, 21 (2019). https://doi.org/10.1038/s41524-019-0153-8
17. E. Bisong, "Google Colaboratory," in *Building Machine Learning and Deep Learning Models on Google Cloud Platform* (Apress, Berkeley, 2019), pp. 59–64
18. M. Wilkinson, M. Dumontier, U.J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. Bonino da Silva Santos, P.E. Bourne, J. Bouwman, A.J. Brookes, T. Clark, M. Crosas, I. Dillo, O. Dumon, S. Edmunds, C.T. Evelo, R. Finkers, A. Gonzalez-Beltran, A.J.G. Gray, P. Groth, C. Goble, J.S. Grethe, J. Heringa, P.A.C. 't Hoen, R. Hooft, T. Kuhn, R. Kok, J. Kok, S.J. Lusher, M.E. Martone, A. Mons, A.L. Packer, B. Persson, P. Rocca-Serra, M. Roos, R. van Schaik, S.-A. Sansone, E. Schultes, T. Sengstag, T. Slater, G. Strawn, M.A. Swertz, M. Thompson, J. van der Lei, E. van Mulligen, J. Velterop, A. Waagmeester, P. Wittenburg, K. Wolstencroft, J. Zhao, B. Mons, *Sci. Data* **3**(1), 160018 (2016)
19. C. Chen, W. Ye, Y. Zuo, C. Zheng, S.P. Ong, *Chem. Mater.* **31**(9), 3564 (2019)
20. A.P. Bartók, M.C. Payne, R. Kondor, G. Csányi, *Phys. Rev. Lett.* **104**, 136403 (2010)
21. A.G. Kusne, A. McDannald, B. DeCost, C. Oses, C. Toher, S. Curtarolo, A. Mehta, I. Takeuchi, *Front. Phys.* **10**, 71 (2022)
22. J. Yang, A. Mannodi-Kanakkithodi, *MRS Bull.* **47**(9), 940 (2022)
23. A. Mannodi-Kanakkithodi, DFT-ML for Perovskites (2022). https://github.com/mannodiarun/perovs_dft_ml
24. P.T. Manganaris, S. Desai, A. Mannodi-Kanakkithodi, MRS Computational Materials Science Tutorial (2022). https://nanohub.org/resources/mrsicmsnotes
25. T. Pandas Development Team, pandas-dev/pandas: Pandas (2020). https://doi.org/10.5281/zenodo.3509134
26. W. Ye, X. Li, C. Chen, Y. Zuo, S.P. Ong, *Nat. Comput. Sci.* **1**, 46 (2021)
27. J.P. Perdew, K. Burke, M. Ernzerhof, *Phys. Rev. Lett.* **77**, 3865 (1996)
28. J. Heyd, G.E. Scuseria, M. Ernzerhof, *J. Chem. Phys.* **118**(18), 8207 (2003)
29. P.C. Jennings, S. Lysgaard, T. Bligaard, *Nat. Comput. Sci.* **5**(46) (2019)
30. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous distributed systems (2016), Preprint, arXiv:1603.04467
31. F. Chollet (2015). https://keras.io/getting_started/faq/#how-should-i-cite-keras
32. H. Xiao, K. Rasul, R. Vollgraf, Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms (2017), Preprint, arXiv.1708.07747
33. J.C.V. Gastelum, A. Strachan, S. Desai, Machine Learning for Materials Science: Part 1 (2019). https://nanohub.org/resources/mseml
34. K.A. Severson, P.M. Attia, N. Jin, N. Perkins, B. Jiang, Z. Yang, M.H. Chen, M. Aykol, P.K. Herring, D. Fraggedakis, M.Z. Bazant, S.J. Harris, W.C. Chueh, R.D. Braatz, *Nat. Energy* **4**(5), 383 (2019) □

Publisher's note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Arun Mannodi-Kanakkithodi is an assistant professor in materials engineering at Purdue University. He received his PhD degree in materials science and engineering from the University of Connecticut in 2017, and worked as a postdoctoral researcher at the Center for Nanoscale Materials in Argonne National Laboratory from 2017 to 2020. His research involves using first-principles computational modeling, machine learning, and materials informatics to drive the design of new materials for energy-relevant applications. Kanakkithodi can be reached by email at amannodi@purdue.edu.

Austin McDannald is a staff material scientist at the National Institute of Standards and Technology (NIST). He received his BS degree in physics from the Worcester Polytechnic Institute in 2010, and his PhD degree in materials science and engineering from the University of Connecticut in 2016. At NIST, he works on developing autonomous experimental material science systems. His particular interest is on encoding physics into the machine learning algorithms used to drive autonomous experiments. McDannald can be reached by email at austin.mcdannald@nist.gov.

Shjijing Sun is a senior research scientist in the Energy & Materials Division of the Toyota Research Institute (TRI). Her research specializes in materials informatics, energy harvesting and storage. Prior to joining TRI, she was a research scientist of mechanical engineering at the Massachusetts Institute of Technology, where she led a team developing low-cost solar cells leveraging high-throughput experiments, artificial intelligence, and advanced characterization using synchrotron facilities. She obtained her BA, MSci, and PhD degrees in materials science at Trinity College, University of Cambridge, UK. Sun can be reached by email at shijing.sun@tri.global.

Saaketh Desai is a postdoctoral appointee at Sandia National Laboratories. He received his PhD degree in materials engineering from Purdue University in 2020, and his undergraduate degree in metallurgical and materials engineering from the Indian Institute of Technology Roorkee, India, in 2015. His research focuses on combining machine learning methods with atomistic and mesoscale computational materials science methods to discover and design materials such as carbon fibers, shape-memory alloys, and thin films. He actively engages in machine learning outreach via workshops and tutorials at conferences, or at nanoHUB.org. Desai can be reached by email at saadesa@sandia.gov.

Keith Brown is an associate professor of mechanical engineering, materials science and engineering, and physics at Boston University. He earned his PhD degree in applied physics at Harvard University, and an SB degree in physics from the Massachusetts Institute of Technology. Following his doctoral work, he was an International Institute for Nanotechnology Postdoctoral Fellow at Northwestern University. The KABlab studies polymers and smart fluids to determine how useful properties emerge from the interplay between composition, processing, and hierarchical structure. A considerable focus is developing approaches that increase the pace of experimental materials research using autonomous experimentation, miniaturization, parallelization, and machine learning. Brown can be reached by email at brownka@bu.edu.

A. Gilad Kusne is a scientist with the National Institute of Standards and Technology, an adjunct professor with the University of Maryland, and a Fellow of the American Physical Society. He received his BS and PhD degrees from Carnegie Mellon University. He leads the machine learning team of an international, cross-disciplinary effort building autonomous research systems to advance solid-state, soft, and biological materials. For these systems, machine learning performs experiment design, execution, and analysis. He is the lead founder of the annual Machine Learning for Materials Research Bootcamp and Workshop educating next-generation and mid-career materials scientists in machine learning. Kusne can be reached by email at aaron.kusne@nist.gov.