FAIR Modeling for Perovskite Solar Cells: An Open Source Machine Learning Pipeline

Nicholas Roberts

A thesis submitted in
partial fulfillment of the
requirements for the
degree of

Master of Science

University of Washington
2023

Reading Committee:
Lih Lin
Shih-Chieh Hsu

Program Authorized to Offer Degree:
Electrical Engineering

University of Washington

Abstract

FAIR Modeling for Perovskite Solar Cells: An Open Source Machine Learning Pipeline

Nicholas Roberts

Chair of the Supervisory Committee:
Lih Lin
Electrical Engineering

Perovskite solar cells (PSCs) show great promise for commercialization, rivaling traditional silicon-crystal solar cell efficiency despite their comparatively short research lifetime. This efficiency is achieved while being manufactured at low temperatures and in ambient conditions, lowering fabrication costs dramatically. Machine learning (ML) promises to significantly expedite further optimization by recommending novel configurations based on insight from existing literature. This paper utilizes the Perovskite Database Project (PDP) [1], an open source PSC database consisting of over 43,000 entries from published literature, to train three ML architectures with short circuit current density (J_{sc}) as a target. Using the XGBoost architecture, an RMSE of 3.73 $\frac{mA}{cm^2}$, R-value of 0.63, and MPE of 10.35% were achieved. This performance is comparable to the results reported in literature and through further investigation could likely be improved [2]. To overcome the challenges of manual database creation, an open-sourced data cleaning-pipeline was created to leverage the PDP. Through the creation of these tools this research aims to increase the availability of ML as a tool to promote improvement in novel device configurations for PSC while showing the already promising performance achieved.

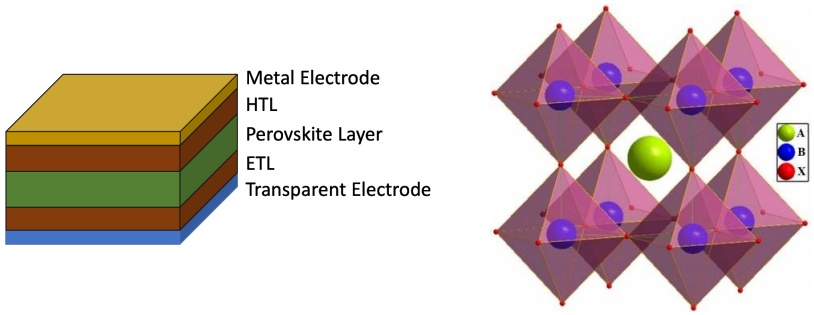
1 Introduction

While traditional silicon crystal solar cells have seen much improvement in efficiency over their 50-year history, a continued limitation remains in their fabrication process and rigid form factor. As the world moves towards increased demand for green energy, research has moved to investigating novel materials for photovoltaics that can offer equivalent or superior power conversion efficiency (PCE) with cost-effective and efficient fabrication techniques. Of these emerging materials, perovskite has shown great promise for commercialization because of its low-temperature facile manufacturing process and competitive quantum efficiencies. Despite having a much shorter research period in comparison, PSCs already rival the PCE of traditional silicon crystal solar cells (27.6%), increasing from 3.8% to 26.1% in just 14 years [3, 4]. Additionally, perovskite-silicon tandem cells have achieved a peak PCE of 32.5% [5], further demonstrating the material's viability in the photovoltaic space.

Perovskite device structure generally follows the format shown in Figure 1a. Additional layers can be added throughout the structure to serve a variety of purposes such as reducing auger recombination [6] or increasing ion transport efficiency. When discussing the PSC device the layers shown in Figure 1a are generally what is considered. In addition, perovskite itself is made up of two lattice structures and at least three elements as seen in Figure 1b. These materials each occupy a site within the material referred to as the A, B, and C sites[1] When generating novel compositions of perovskite, the materials in these sites are changed or mixed with other materials such as methylammonium (MA) or formamidinium (FA). These mixtures are often represented as molar coefficients, examples of which can be seen in Section 2.1 Table 1.

Incorporating perovskite into the solar cell absorption layer gives rise to complex interactions between device stack layers. To improve upon existing solar cell designs, researchers must consider how the structural and electrical properties of each layer contribute to the overall performance. In the past, such improvements largely relied on researcher intuition combined with trial-and-error experimentation. However, with the advent of machine learning (ML) and Big Data, it is now possible to expedite this process through analysis of large databases of previous experimental results. When compared to other modeling techniques, such as Density Functional Theory (DFT) or molecular dynamics, ML is better poised for practically modeling the non-linearities among many features found in perovskite solar cells [9]. These analyses can range from pattern recognition, which may

[1]The C site can also be referred to as the X site as seen in Figure 1b.



(a) (b)

Figure 1: *(a) Common device stack for a perovskite solar cell [7], (b) Visual representation of perovskite structure [8]*

shed light on a previously unknown relationship between material components, to predicting numeric values for solar cell performance. In particular, ML can predict the key performance metrics of solar cells such as short circuit current density (J_{sc}), open circuit voltage (V_{oc}), power conversion efficiency (PCE), and the fill-factor (FF), which taken together yield useful information on the performance of a particular solar cell configuration. Using these prediction methods while leveraging a researcher's own intuition can guide testing in more promising directions, speeding up the development cycle [10].

Critical components to the viability of a machine learning model are the quality and quantity of the data, the upper limit of performance being determined mainly by the quality of data used for training [11]. Previously, research conducted using ML to predict the electrical performance of PSCs [12, 13] required the creation of a custom training set from data collected from the literature. This can make it difficult to replicate or compare performances across models. The creation of a quality dataset of sufficient size for machine learning is no trivial task, and the conversion of raw data points into a format best suited for an ML investigation takes a significant amount of time that could be spent exploring the viability of different model frameworks.

When creating a machine learning pipeline based on human-collected data, whether through a literature search or from a database of experimental results, one of the initial tasks is to create a database devoid of impractical or erroneous data points. The data may not have been formatted for ML and must be converted to formats readable by ML models, some of which can only handle certain data types (e.g. numeric with neural networks). For PSCs, the categories of data available are varied, as materials, chemical ratios, and manufacturing processes can take the

form of numeric and categorical features. A typical ML development cycle involves many iterations of feature selection, model training, and testing with a variety of features and model types. This can lead to the time-consuming process of changing how information is encoded, balancing feature sets, and handling erroneous or missing information, along with a litany of other considerations that contribute to the ultimate quality of the training data.

To aid future research in using machine learning to guide perovskite solar cell manufacturing, we have created an open-source data cleaning and ML pipeline based on the Perovskite Database Project (PDP) [1]. Designed with FAIR data principles that emphasize findability, accessibility, interoperability, and reusability [14], the PDP offers a large open-source repository of PSC device information with detailed documentation. Because of its size and commitment to accessibility and openness, it is an ideal starting point for an open-source machine learning pipeline. Indeed, since the PDP debuted in 2022, studies have emerged using it for ML-based PSC performance predictions [7, 15]. In this paper, we outline our modular and flexible workflow for ML that is capable of handling any feature set or target found in the PDP. Inspired by FAIR data principles, we have included thorough documentation and have uploaded our codebase to GitHub (`https://github.com/linphotonicslab/ML_Pipeline/`). The pipeline's modular design for handling the formatting guidelines of the PDP, the layered nature of solar cell data, and common errors allows for any combination of features to be selected, cleaned and trained on. We anticipate that it can greatly reduce the time spent deploying models that can guide PSC fabrication by making useful performance predictions or by identifying better processes and materials.

2 Methods

2.1 Data Preparation

Data Selection

Data was selected from The Perovskite Database Project (PDP), an open-source database created by Jacobsson et al. to introduce more accessibility for perovskite device data [1]. The PDP consists of tabular data recording device parameters, electrical characteristics, and compositional information for perovskite solar cells. Since March 2009, entries for PSCs published through March 2023 have been incorporated into the PDP, amounting to 43,239 datapoints. A subset of which, containing 41,942 points that were published between February 2014 and December 2020, was selected for cleaning. The December 2020 end date was chosen as

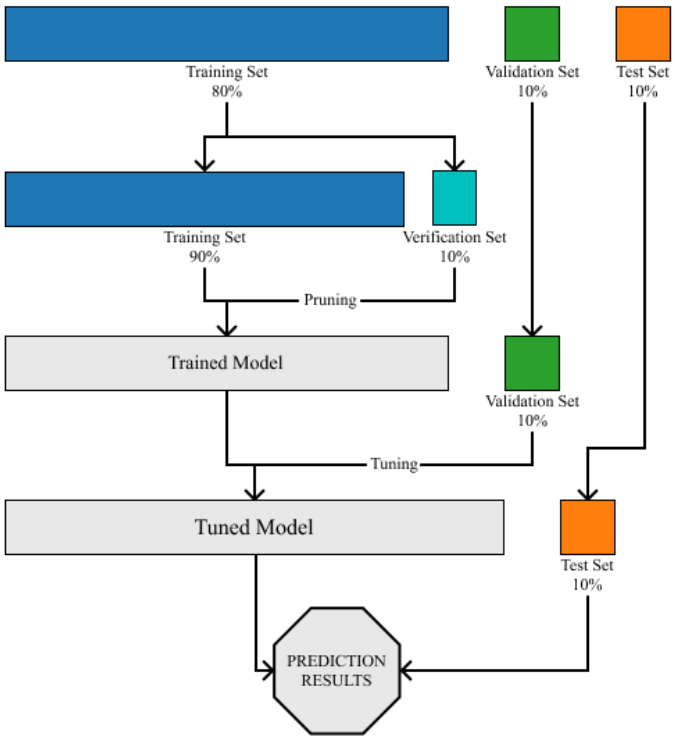


Figure 2: *Data splits showing the usages of each dataset in training, tuning hyperparameters, and predicting.*

this marked when the researchers open-sourced the project to the greater research community, allowing for crowd-sourced data additions. In its current form, this crowd-sourced data suffers from a higher frequency of errors than the data curated by the researchers and, as such, makes it difficult for future processing. A starting date of January 2014 was chosen as a time when PSC technology had begun to mature and high-quality devices began to be reported. This is confirmed when looking at the Best Research-Cell Efficiency Chart maintained by the National Renewable Energy Laboratory, which shows the highest accredited efficiencies of solar cells since 1976 [4]. The first cataloged, and thus accredited, entry for PSCs appeared in mid-2013.

Feature Selection

The Perovskite Database consists of 410 columns, of which a certain number must be selected as inputs to future ML models. To do this, categories of features were created for grouping the raw columns. These categories were "Perovskite Depo-

sition," "Perovskite Composition," "Electrical Characteristics," "Electrical Parameters," and "Substrate." This selection was guided by intuition from previous literature showing which features seemed to have greatest impact on perovskite performance and resulted in 26 of the original 410 columns being used. While this column reduction may seem extreme, many columns contain similar information that was deemed unnecessary to include. For example, only one column is required for a target metric, but there are over 50 columns pertaining to PSC performance. Once columns were grouped into these categories, valid entries were counted. A valid entry was defined to be a non-NaN entry. Once these counts were determined, only raw columns with more than 10,000 valid entries were considered to ensure enough data for training. A final list of all considered features can be found within the supporting information (S1).

As a target value, the short-circuit current J_{sc} was chosen. J_{sc} is proposed as a metric due to its more fundamental nature when compared to more traditional metrics like power conversion efficiency (PCE) or fill factor (FF). FF depends on the J-V characteristics of the PSC and represents the ratio of the max power that can be delivered by the PSC to the product of J_{sc} and open-circuit voltage V_{oc}. PCE is simply a ratio of generated output power to input power which itself is reliant on the FF (Equations 1, 2). V_{oc} was not chosen as the target metric despite it being similarly intrinsic due to its tendency to saturate at high values making variations in high V_{oc} materials much more difficult to differentiate.

$$\mathrm{FF} = \frac{P_{max}}{J_{sc} \times V_{oc}} \tag{1}$$

$$\mathrm{PCE} = \frac{P_{out}}{P_{in}} \tag{2}$$

J_{sc} is also ideal as a metric as it is agnostic to the load of any additional circuitry, allowing it to serve as a stand-in metric for other utilizations of perovskite, such as LEDs, which have very similar device stacks. Perovskite LEDs are a newer but growing perovskite application and can benefit from leveraging perovskite materials and device structure that result in high J_{sc} as a starting point for optimizing other properties, such as external quantum efficiency (EQE), which are more applicable to LEDs.

Data Encoding
Once the desired columns had been selected, these raw columns needed to be converted into features readable by a machine learning model; this was done with two strategies. One-hot encoding was used to represent the presence of a material in a specific device or the lack thereof. This strategy can be seen when encoding materials used in the hole or electron transport layers. If, for example, PEDOT:PSS is used as a hole transport layer material, it will be represented as a 1, and if it is not, then it will be represented as a 0.

However, not all information can be encoded in this way. An example of this would be the molar coefficients of cations in the perovskite. These are important for understanding the perovskite configuration and are present in different quantities; as such, encoding these molar coefficients as a 1 if it is present or a 0 if it is not will not work. For these types of columns, the second strategy, which will be referred to as two-column encoding, is used. As an example, the PDP utilizes two columns describing material concentrations in the A, B, and C sites. The materials used in the composition are first recorded in a column like "Perovskite_composition_a_ions," which would have an entry such as "FA;Cs" to designate that both FA and Cs are materials used for the A site in this material. This information is then paired with another column named "Perovskite_composition_a_ions_coefficients," which would have entries in the form "0.75;0.25." This information together would signify that FA and Cs are used in the A site with molar coefficients of 0.75 and 0.25, respectively. To encode this information into a readable form for the model, the materials are recorded and turned into column headers, and their associated coefficients are recorded as entries. An example of this conversion can be seen in Tables 1 and 2.

Table 1: **Raw Perovskite Configuration Format**

A Site Ions	*A Site Ion Coefficients*	*B Site Ions*	*B Site Ion Coefficient*	*C Site Ions*	*C Site Ion Coefficient*
FA; Cs	0.75; 0.25	Pb	1.0	Br; I	0.15; 2.85
FA	1.0	Pb	1.0	I	3.0
FA	1.0	Pb	1.0	I	2.5
MA; FA	0.5; 0.5	Pb	1.0	Cl; I	1.0; 2.0

Example perovskite configurations as they are represented in the Perovskite Database. The column headers have been changed for readability, within the database they follow the convention "Perovskite_composition_a_ions", "Perovskite_composition_a_ions_coefficients", etc.

Table 2: **Encoded Perovskite Configuration Formatting with Layers**

a_MA	a_FA	a_Cs	b_Pb	b_Sn	c_Br	c_Cl	c_I
0.0	0.75	0.25	1.0	0.0	0.15	0.0	2.85
0.0	1.0	0.0	1.0	0.0	0.0	0.0	3.0
0.0	1.0	0.0	1.0	0.0	0.0	0.0	2.5
0.5	0.5	0.0	1.0	0.0	0.0	1.0	2.0

Example perovskite configurations shown in Table 1 after being encoded into a format readable by the ML model.

In addition to these encoding methods, a higher level of organization is used to denote whether a particular section of the device structure (e.g., perovskite layer) consists of multiple layers. These layers are denoted in the database with a "|" character. Data is split around these characters, and the same encoding strategy is applied as in Figure 1, with the addition of an organizational tag denoting the layer specified. In reference to the earlier example of "FA;Cs", an entry with multiple layers would be represented as "Cs;Fa|MA." This string would show a two-layer perovskite structure in which the A site for the first layer consists of Cs and FA while the second layer has MA for the A site. This results in features expressing the material, its layer within the device, and the associated coefficient (Tables 3 and 4).

Table 3: **Raw Perovskite Configuration Format**

A Site Ions	*A Site Ion Coefficients*	*B Site Ions*	*B Site Ion Coefficient*	*C Site Ions*	*C Site Ion Coefficient*
FA; Cs| Cs	0.75;0.25|1.0	Pb| Pb	1.0|1.0	I| I	3.0|3.0
FA| Cs	1.0|1.0	Pb| Pb	1.0|1.0	I| I	2.5|3.0

Example multi-layered perovskite configurations as represented in the Perovskite Database.

After this encoding is completed for the entire raw dataset, a new dataset is created with columns denoting features to be used within the model.

Data Cleaning

Throughout the encoding process, there are four major checkpoints in which the

Table 4: **Encoded Perovskite Configuration Formatting with Layers**

a_FA_L0	a_FA_L1	a_Cs_L0	a_Cs_L1	b_Pb_L0	b_Pb_L1	c_I_L0	c_I_L1
0.75	0.0	0.25	1.0	1.0	1.0	3.0	3.0
1.0	0.0	0.0	1.0	1.0	1.0	2.5	3.0

Example perovskite configurations with layering information encoded.

data is cleaned to ensure all results are valid for training. The first begins with the two-column encoding of the perovskite composition. In order for an encoding to be valid, there must be the same number of entries in both the "Perovskite_composition_a_ions" and "Perovskite_composition_a_ions_coefficients" columns. An example of an invalid entry would be "Cs;FA" in the composition ions column and "0.7" in the composition ions coefficients column. Since there are two materials in the A site but only one coefficient is listed, these do not match, and the entry is considered invalid. This rule is true of any two-column encoding values. In addition, if the material has multiple layers, the same number of layers must be listed across the A, B, and C site coefficients to be valid. An example of a feature invalid in this sense could have an entry "Cs|FA" in the A site column, "Pb|Pb" in the B site column, and "Br" in the C site column. Since the C site only has one material layer listed, the second layer cannot be completed, and this entry is also considered invalid.

Table 5: **Raw Perovskite Configuration Format**

	A Site Ions	*A Site Ion Coefficients*	*B Site Ions*	*B Site Ion Coefficient*	*C Site Ions*	*C Site Ion Coefficient*
Valid	FA; Cs	0.75; 0.25	Pb	1.0	I	3.0
Invalid	FA; Cs	0.75	Pb	1.0	I	3.0
Valid	FA| Cs	0.75|1.0	Pb| Pb	1.0|1.0	Br| I	3.0|3.0
Invalid	FA| Cs	0.75|1.0	Pb| Pb	1.0|1.0	Br	3.0

Example multi-layered perovskite configurations as they are represented in the Perovskite Database with valid and invalid versions of encoding.

The second cleaning occurs after encoding all columns and consists of removing all NaN entries within the database. This cleaning step is optional, depending on the model being used. Specific models, such as tree-based algorithms, are capable

of working with incomplete entries. However, other models, such as neural networks, are not. Since this study focused on exploring the performance of different models using data from the Perovskite Database, this step was done to allow the utilization of neural networks.

The subsequent data cleaning occurs after all encoding has been completed and NaN entries removed, which is a statistical cleaning step. For this cleaning, each encoded column's means and standard deviations are calculated. If any row has an entry greater than three standard deviations away from its respective column's mean, that row is removed. An important note is that the target column and columns with one hot encoding are not considered for this statistical cleaning. The purpose of this cleaning step is to remove any data that may have been input into the database incorrectly; if, for example, a material coefficient was 50 instead of 5, this is likely incorrectly recorded and should be removed. Since the one hot encoding can only have a value of 0 or 1, it is impossible for incorrect recording to happen, and as such, it does not need to be cleaned in this way. In fact, cleaning these columns in this way could be detrimental if the calculated mean was too low; all data with ones could end up removed, thus destroying any information that the column held.

The final cleaning occurs post duplicate removal (see below). Since some rows are removed in each cleaning and duplicate removal step, there is the possibility that all non-zero entries for certain columns have been removed and thus contains no information. Any columns with all zero values are removed, leaving the final clean dataset. The final dataset consists of 7213 entries with 2126 features.

Duplicate Removal

To ensure a clear mapping between device configuration and the target value, it is useful to remove duplicated configurations within the database and standardize the remaining values. If these duplicates are not first removed, a single configuration will have multiple targets associated with it, making it harder for the model to determine the correct mapping between device configuration and target value, which in this case is J_{sc}. To address this issue, the J_{sc} values of duplicated entries were grouped, leaving each duplicated configuration with a distribution of J_{sc} values in which a new pseudo-J_{sc} value could be selected to represent this distribution (Figure 3). For this procedure, the J_{sc} value created was the average of the mean and median of the distribution. This value was used to balance between the mean's tendency to be right-skewed by a small number of particularly poorly performing devices and the median's tendency to be left-skewed due to researchers preferentially entering top-performing devices into the database more frequently.

All duplicates are then removed from the database before being replaced by this feature group and associated pseudo-J_{sc}.

After conducting all other cleaning steps besides the duplicate removal, there were 18,320 valid entries. However, after removing all duplicates, there are only 7,154 remaining valid entries. This reduction of more than 60% of the current database warranted further exploration to ensure this process was doing as expected. A natural place to start was to determine whether these cleaning steps caused the high number of duplicates or if it was intrinsic to the database. Upon inspection, 154 duplicates were found within the database before any columns or entries are removed.[2] While this accounted for a small number of duplicates, this was still far from the 60% seen within the cleaned database, meaning these duplicates were being created during cleaning. By examining the counts of each duplicate it can also be seen that there are not especially prevalent configurations, instead 2854 different configurations with at least one duplicate.

The more uniform distribution of duplicates across different configurations indicates that each group was sourced from a unique paper. Since the PDP contains DOI information, this was easily verified, and by tagging each duplicate by its associated DOI, it can be seen that this is generally the case (Figure 4). Most duplicates are caused by papers referenced within the PDP choosing a particular configuration and making minor changes such as varying the thickness of certain layers in the device stack. These alterations often cause a change in measured J_{sc} but are not accounted for in the cleaned database as this aspect was not selected for encoding. These entries are then treated as duplicates which is what causes the distributions seen in Figures 3 and 4. The average number of DOIs per distribution is 1.04, again showing that this intuition is likely correct. The averaging of the J_{sc} of duplicates is a trade-off that has been made in order to create a semi-uniform database that will allow a machine learning model to discern some underlying patterns between J_{sc} and device configurations. If all information were kept, there likely would not be enough information about any one particular aspect to make a reasonable prediction.

[2]For this analysis, some columns containing only metadata were removed (i.e., name of the person entering the data, date entered into the database, database ID, etc.).These were not considered relevant for checking for duplicates and were removed. A list of all columns removed for this can be found in S2

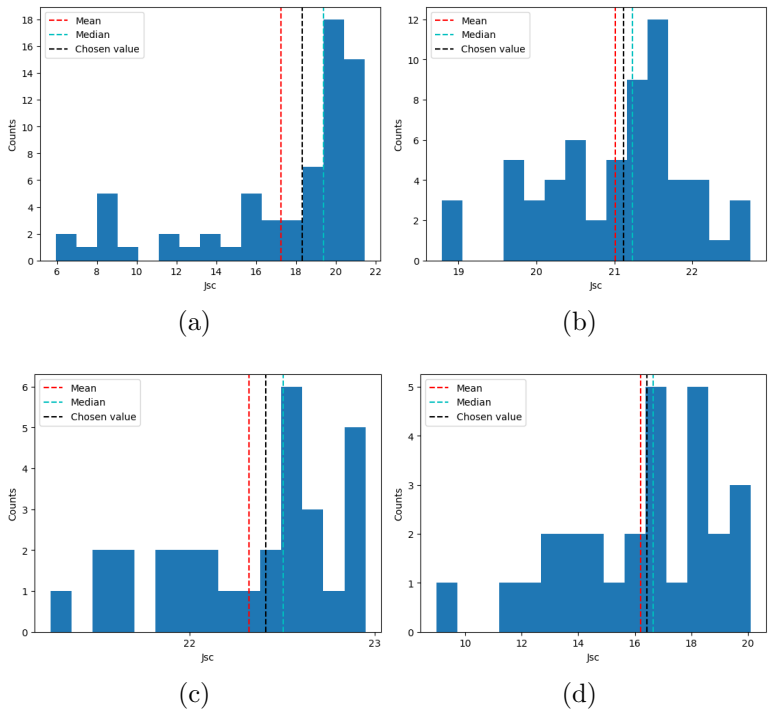


Figure 3: *Example distributions of J_{sc} for an assortment of different solar cell configurations. The mean, median, and pseudo-J_{sc} of each distribution are shown which was chosen to be representative of the distribution. The configurations of each distribution can be found in S3. (a) Mean: 17.24, Median: 19.38, Chosen Value: 18.31 (b) Mean: 21.01, Median: 21.23, Chosen Value: 21.12 (c) Mean: 22.35, Median: 22.46, Chosen Value: 22.41 (d) Mean: 16.21, Median: 16.65, Chosen Value: 16.43*

2.2 Model Training

The original database has now been cleaned and is ready to be divided into the training, validation, and test sets required for model training. For this study, an 80/10/10 split was used. The training set was used for weight optimization, the validation set was used for estimating optimal hyperparameters, and the test set was used to estimate unbiased model performance. Three model architectures were evaluated on this dataset to allow a comparison across different methods. Several model instances of each type were trained in order to evaluate optimal hyperparameter values. The models architectures evaluated were an XGBoost regressor, a deep neural network (DNN), and a Gaussian process regressor (GPR). XGBoost was selected to evaluate the performance of tree-based approaches which showed promise in mapping perovskite configurations to target metrics [7]. DNN was selected for its ability to map complex non-linear relationships which may

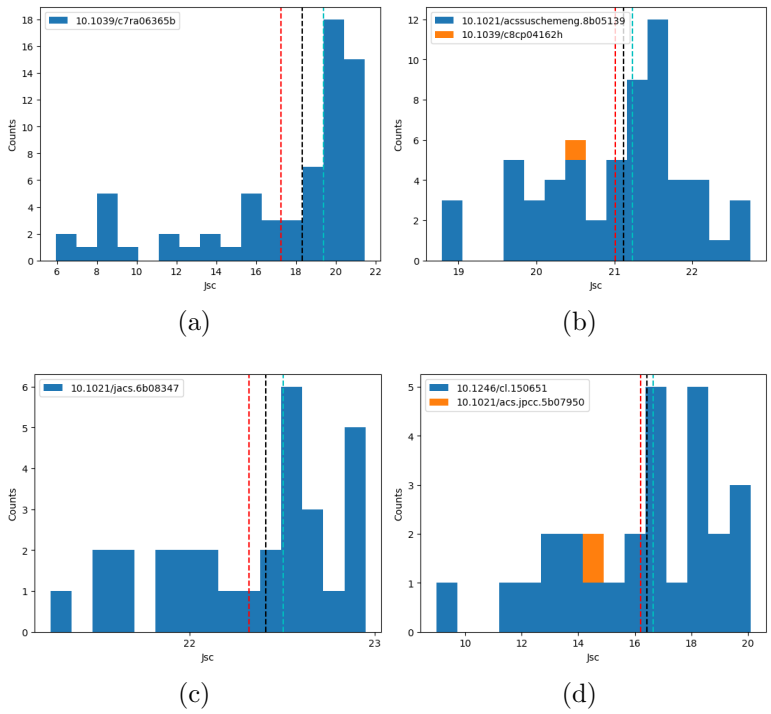


Figure 4: *Example distributions of J_{sc} from Figure 3, now showing the associated DOI for each J_{sc} included in the distribution.*

help in predicting values at the extreme of the test set target range. High J_{sc} is a desired characteristic that correlates with the performance of solar cells (Figure 5). GPR was chosen for its innate ability to pair a confidence interval with each prediction. This additional metric could also help in edge cases of the test set target values. If, for example, a particular configuration has a slightly higher predicted J_{sc} but the confidence interval is large, a slightly lower predicted J_{sc} with a smaller confidence interval could be selected.

Training of models follows the same general structure and begins by first loading the three pre-split datasets. Once they are all loaded the training set goes through an additional 90/10 split to create a 'verification' set. This set is used exclusively during hyperparameter tuning to ensure the model does not overfit the training set. To conduct hyperparameter tuning the Optuna library was used [16]. Optuna is an open-source hyperparameter tuning framework that can be implemented to increase the speed at which different hyperparameters can be tested. To conduct these tests, a base model, a parameter grid that defines the hyperparameter search space, and a desired number of test iterations are fed into Optuna. Optuna will then use a Bayesian sampler to minimize the chosen measurement metric while

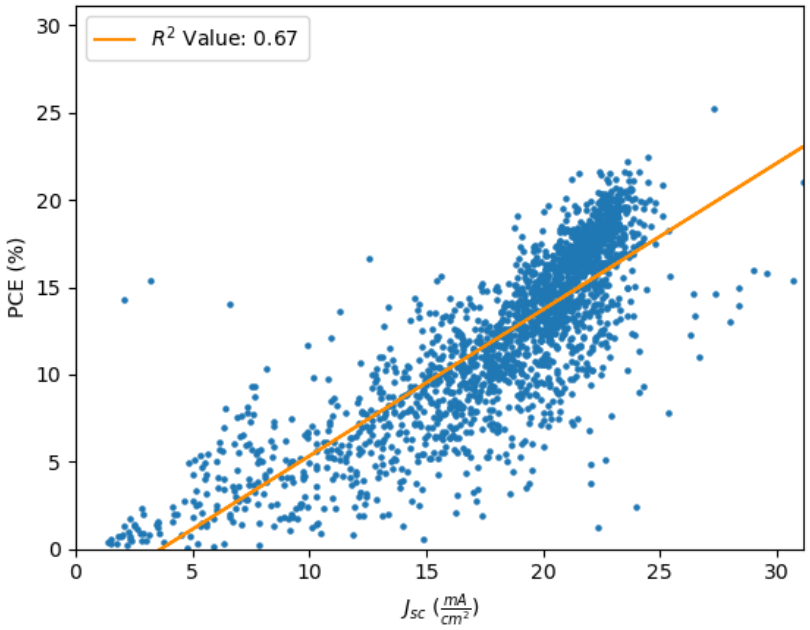


Figure 5: *Linear correlation plot between J_{sc} and power conversion efficiency (PCE) from within the Perovskite Database*

searching the hyperparameter space defined in the parameter grid. The default Optuna sampler Tree-structured Parzen Estimator (TPE) was used for this and all following models, along with a custom error metric used for hyperparameter tuning, as seen in Figure 6. This custom function was found to be a good balance between minimizing the MPE of the verification set which discouraged only predicting the mean of the distribution and minimizing the RMSE which helps to encourage overall accuracy. The verification set is used by Optuna to calculate this error metric which in turn dictates the hyperparameters it will test. Once this search is complete, the final model is tested on the validation set. If the model underperforms on the validation set the parameter grids are adjusted in Optuna and a search is re-conducted until a models performance on the validation set is acceptable. The performance on the validation set is measured as the RMSE and R-value of the predictions. This configuration is then used to predict the test set from the original split. RMSE, R-value, and mean percent error (MPE) are then recorded for later comparison against the other models architectures. XGBoost training follows these steps exactly while the neural network and GPR have some slight differences which are explored in the following sections. All parameter grids and model configurations used can be found in supplement section S4.

$$\text{Error} = \text{MPE(Verification)} + \text{RMSE(Verification)}$$

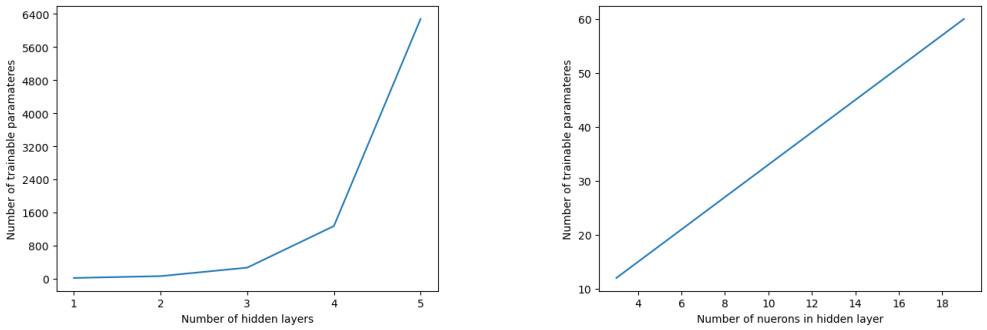
Figure 6: *Error function used for hyperparameter tuning.*

Deep Neural Network Training

Training was conducted slightly differently for the DNN compared to the other two model architectures for two main reasons, the first being neural networks' ability to train in epochs rather than all at once, and the second being the essentially infinite number of topologies a neural network can have. Beginning with the first change, having training be split into separate epochs allows for an examination of how well training is going to occur between each epoch. This allows for pruning of unpromising hyperparameter configurations, improving runtime as these unpromising models do not need to be fully realized. This pruning is done via Optuna, and the error metric seen in Figure 6 is again used to determine whether a specific configuration should be pruned as well as a configuration's overall performance. The successive halving pruner was utilized from the Optuna library for this. The second main difference is how the neural network is built. While the XGBoost and other tree-based models also contain nodes like a neural network (neurons in that case), the sheer number of connections means a seemingly small increase in the number of neurons leads to orders of magnitude more connections to calculate (Figure 7). Because of this, the number of neurons in each layer and the number of layers in a neural network is something that must be considered and cannot generally just be searched over with a hyperparameter optimizer like Optuna due to the runtime requirements. To help compensate for this, two main topologies are tested, and the better of those two is selected: a wide model and a deep model. A wide model consists of few layers with many neurons, whereas a deep model consists of many layers with fewer neurons.

To train each of these models, a mean squared error loss function was used while the optimizer for calculating new weights was selected by Optuna from a list consisting of the Adam optimizer, RMSprop optimizer, and stochastic gradient descent optimizer. In addition to this, a batch size of 64 samples was used for each model.

To determine the performance of each of these topologies, Optuna was utilized, first by restricting the number of layers within the search space to be between 1 and 3 but allowing each layer to have up to 2 times the input size numbers of neurons to create a wide network. Once optimized, the performance on the test set is recorded as RMSE and R-value, and the deep network is tested. The deep network training allowed 4 to 6 layers but restricted the number of neurons in



(a) (b)

Figure 7: *Growth rates of trainable parameters as (a) the number of hidden layers increases and (b) the number of neurons per hidden layer increases. For (a) an input size of 2, an output size of 1, and all hidden layers a size of 5 was used. For (b) an input size of 2, an output size of 1, and only a single hidden layer was used.*

each layer to a maximum of one-half the input size. The better performing model was selected based on RMSE and R-value. This selected model was then used to predict the J_{sc} of the test set.

Guassian Process Training

When training the GPR system memory on the computers conducting the model training was found to be a limitation. The system utilized had only 32 GB of RAM, which was insufficient for the required matrix manipulations. To compensate for this, a random sample of 1000 points was selected from the training set for each iteration of the Optuna tuning. This sample size was small enough to complete training with no memory issues while maintaining sufficient performance. A computer with more system memory could likely train utilizing the entire dataset and could see better results.

For GPR kernel creation, three basic kernels were considered to build the final kernel- these being the Matern kernel, rational quadratic kernel, and white noise kernel. Each of these was chosen due to their ability to represent different aspects of data, which Optuna can then piece together to best predict the target. Matern is a general form of the radial basis kernel, often used as a default kernel due to its flexibility. Rational quadratic kernels are effective at capturing general trends in data, which would be expected if specific perovskite configurations consistently perform better than others. Lastly, the white noise kernel can be used to filter out any noise in the data caused by differences in lab setups, tool measurement tolerances, or human error. Combinations of these three kernels are more likely to

represent the complex relationship between the input features and output target better than any single one could individually.

3 Results

With each model now trained, error metrics can be extracted from the final testing set and compared. Three metrics were chosen to represent the model performance: RMSE, R-value, and MPE (Equations 3-5). RMSE represents the average magnitude of our model's prediction errors, giving a clear indication of accuracy in a unit that is comparable to the observed data. The R-value, or correlation coefficient, measures the strength and direction of the linear relationship between the predicted and observed values, with a value close to 1 or -1 indicating a strong positive or negative linear correlation, respectively. The R-value complements the RMSE by highlighting the model's ability to capture trends in the data. Finally, MPE assesses the relative deviation of the predicted values from the observed values expressed as a percentage. This is useful in comparing performance across varying scales and units (e.g., comparing predictions of J_{sc} and V_{oc}, which are closely related but have different units).

$$\mathrm{RMSE}(y, \hat{y}) = \sqrt{\frac{\sum_{i=0}^{N-1}(y_i - \hat{y}_i)^2}{N}} \tag{3}$$

$$\mathrm{R} = \frac{\sum_{i=0}^{N-1}(x_i - \hat{x})(y_i - \hat{y})}{\sqrt{\sum_{i=0}^{N-1}(x_i - \hat{x})^2 \sum_{i=0}^{N-1}(y_i - \hat{y})^2}} \tag{4}$$

$$\mathrm{MPE}(y, \hat{y}) = \frac{\sum_{i=0}^{N-1} \left| \frac{\hat{y}_i - y_i}{y_i} \right|}{N} \times 100\% \tag{5}$$

As seen in Table 6, the deep neural network scored marginally worse, scoring a 1% higher RMSE and 13.3% lower R-value on the validation set. The wide network was chosen as the better topology and used for the results in Table 7. Of the three model architectures, XGBoost performed the best on the test set in all metrics.

While the XGboost outperforms both the GPR and DNN, the MPE between all three models is considerably higher than is expected, especially for the calculated

Table 6: **Deep and Wide Neural Network Performance on Validation Set**

	RMSE	*R-Value*
Wide Network	4.95	0.3
Deep Network	4.99	0.26

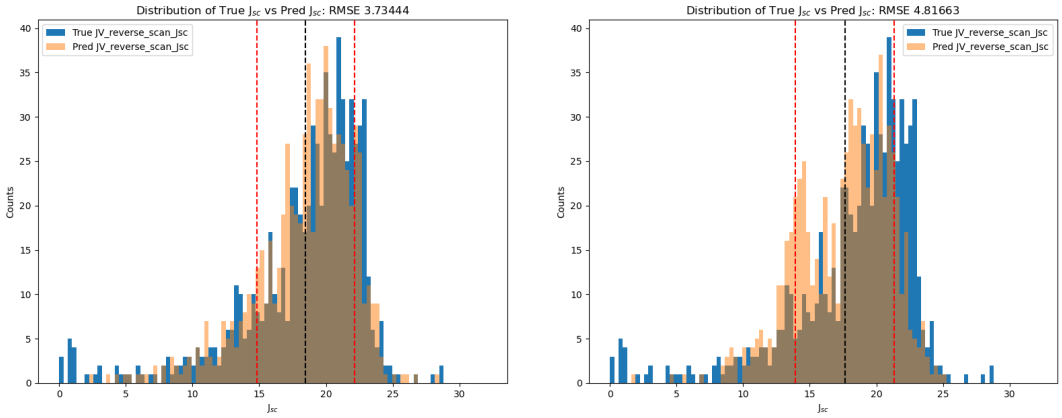
Table 7: **Model Performance on Test Set**

	RMSE	*MPE*	*R-Value*
Neural Network	4.82	79.9	0.36
XGBoost	3.73	86.8	0.63
Gaussian Process	13.02	94.1	-0.08

R-value of the XGBoost. This behavior becomes more apparent looking at the distributions of all models (Figures 8, 9). While the GPR appears to not learned from the data, instead only predicting an arbitrary value within the distribution, both the neural network and XGBoost appear to be much more effectively predicting the distribution.

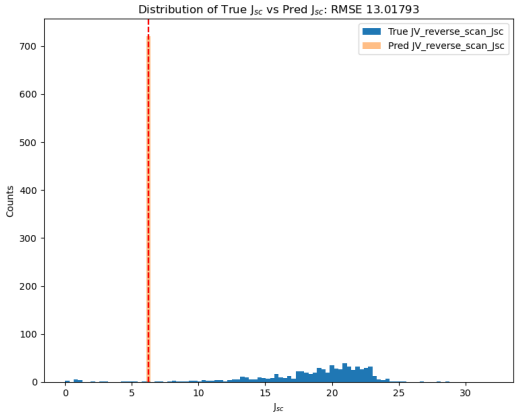
Figure 9 shows good prediction throughout the center samples (sample number 2000-5000) for both the XGBoost and DNN models but lower performance at the extremes of the test set. This can be seen in Figures 10 and 11 by examining the residuals of each prediction. This reduced ability to predict at the extremes of the dataset was likely due to an imbalance in the number of samples with J_{sc} values correlating to the center samples of Figure 9. This imbalance is a difficult problem to combat as the number of entries within that center region outnumber entries beyond it by a factor of 2:1 even before any cleaning has been done (Figure 12). Despite this, the residuals in the upper extremes of J_{sc} are considerably lower than in the lower extremes for both the XGBoost and DNN.

With this in mind and again considering equation 5, low performance at the lower ranges of the dataset will cause a massive increase in MPE. If, for example, the true value of the test set is 2 and the model predicts 6 that is a 200% error despite only being slightly above the RMSE. Ideally, the model would be able to predict at all ranges but in this situation the higher range is considerably more



(a)

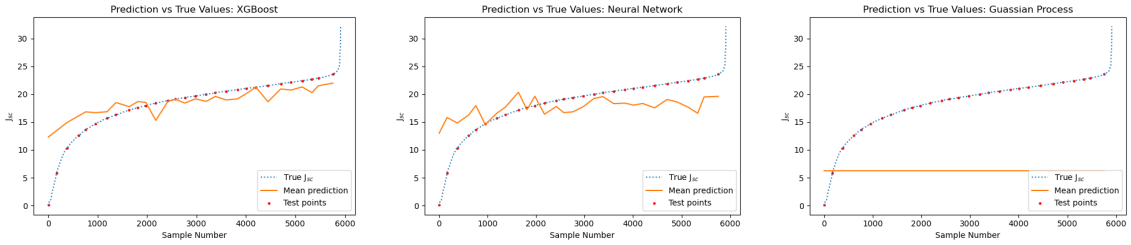
(b)



(c)

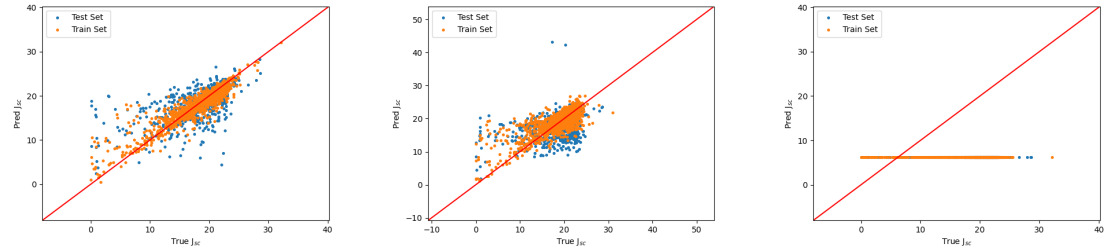
Figure 8: *Histograms showing the distribution of predicted and true J_{sc}, (a) XGBoost, (b) Neural Network, (c) Gaussian Process. The black dotted line denotes the mean value for the prediction distribution and the red dashed lines show the +/-1 standard deviation from that mean. (a) Mean: 18.47, Standard Deviation: 3.66, (b) Mean: 17.644, Standard Deviation: 3.704, (c) Mean: 6.15, Standard Deviation:* 3.54×10^{-6}*. The training dataset has a true mean of 18.40 and a standard deviation of 4.63*

important as the goal is to fabricate high J_{sc} devices. To compensate for this poor performance at the lower ranges skewing the MPE so heavily, an adjusted MPE is shown which considers the MPE of all points above the mid point of the J_{sc} range. This point for the test set is 14.38 $\frac{mA}{cm^2}$. Considering only predictions beyond this point still accounts for 84.6% of the test predictions but significantly



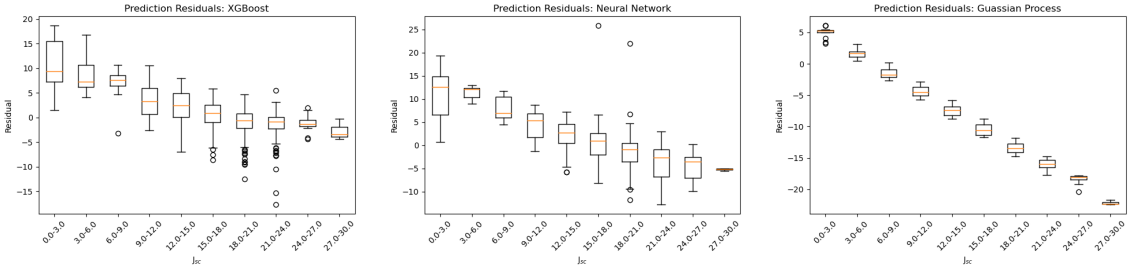
(a) (b) (c)

Figure 9: *Performances of various models on the test set. The orange line shows a models predictions at points indicated in red as well as the true J_{sc} value seen as the dashed blue line.*



(a) (b) (c)

Figure 10: *Plots showing the true J_{sc} value versus predicted J_{sc} value (a) XGBoost, (b) Neural Network, (c) Gaussian Process. Blue dots denote samples from the test set and orange from the training set.*



(a) (b) (c)

Figure 11: *Boxplots showing the residulas for different ranges of J_{sc}, (a) XGBoost, (b) Neural Network, (c) Gaussian Process. In combination with the above plots it can be seen areas where models predictive performance decreases. XGBoost, for example, shows much higher residuals in the low J_{sc} range but accuracy improves greatly at higher ranges.*

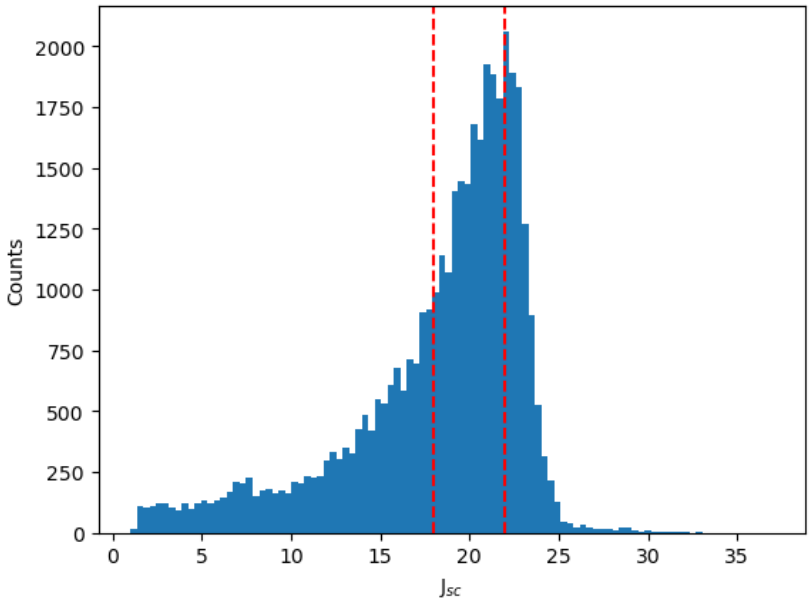


Figure 12: *Distribution of J_{sc} values before any database cleaning. The red lines show J_{sc} values for sample numbers 2000 and 5000, respectively, from Figure 9.*

improves the MPE (Table 8).

Table 8: **Adjusted Mean Percent Error**

	Adjusted MPE
Neural Network	15.66
XGBoost	10.35
Gaussian Process	68.32

This performance towards higher J_{sc} values shows these models are still useful in predicting at these ranges and could still be used in research to guide experimentation on high performance materials. Exact parameters for all models can be found in the supplement (S5).

4 Discussion

As can be seen from the results, the best-performing model was found to be the XGBoost regressor. While the results are promising, especially for one of the first utilizations of the PDP for ML-based studies, there is still clear room for improvement. One of the key areas that could be a promising direction for future research is the use of the additional nonstandard layer columns in the Perovskite Database in place of perovskite additives.[3]. Standard information regarding PSC device stack, perovskite composition, and deposition are always included as these are understood to be vital in predicting performance. However, in this experiment perovskite additive information was also included as a potential method for creating high J_{sc} devices. The justification for this being that if the goal is to optimize the performance of PSCs, the best way to do this is likely to improve the efficiency of the perovskite which may be done through additive materials. However, considering additional nonstandard layers may be more predictive of J_{sc} than initially believed, possibly by aiding ion transport after being generated in the perovskite or through passivation.

An aspect to consider when examining these results is the system memory bottleneck encountered while training the GPR. Sampling the original training set became necessary to compensate for this limitation, and while randomly sampling the training set should not significantly impact the training that is difficult to know for certain. Because this model notably had the worst performance, reconducting this experiment with a machine capable of handling the entire training set would be ideal to ensure accurate results.

J_{sc} was used for these tests, but there are many other potential metrics that could have been used, which may still be other interesting directions for exploration. The justification for J_{sc} was due to its intrinsic importance to the solar cell, being a needed quantity, along with V_{oc}, to calculate fill factor (FF). This fact lent itself well to the idea that it would be a good predictor of performance, which was supported by Figure 5. That does not mean it is the only possible metric, though, and another obvious choice could be V_{oc} due to its vital nature in calculating FF. V_{oc} was not used for these tests due to its tendency to begin to saturate at higher values and make it more difficult to distinguish between two high-performance devices. Despite that, it may be easier to predict than J_{sc} and thus generate more accurate models. This could be true of other more traditional metrics like PCE or FF which all provide other areas of study that could improve results.

[3]These columns are encoded with the prefix Add_lay_front or Add_lay_back within the database

Another unexplored avenue of research is the utilization of J_{sc} as a stand-in metric for other applications of perovskite. Besides solar cells, perovskite is becoming more common in color conversion and LED applications (PeLED). Because these are newer fields of study, there is not currently an existing database of PeLED performance on which to conduct ML experiments. Since J_{sc} is more a measure of the performance of the perovskite and its transport layers, both of which are present in LED applications, it could be used as a replacement metric to aid in the design of high-performance LEDs. Studies with this goal in mind could isolate specific perovskite phases (such as quantum dots, quasi-2D, or 3D) for optimal conductivity and retrain a J_{sc} prediction ML model to guide fabrication.

5 Conclusion

In this study, we presented a novel usage of the PDP by training three ML architectures to predict a target metric of J_{sc}. In doing this, we posit J_{sc} as an improved metric for predicting PSC performance due to its intrinsic nature to the performance of the perovskite materials and the PSC device structure. In addition, this target allows ML models trained on PSC data to be extrapolated to other applications of perovskite such as LEDs. Using data created from the PDP, we achieved a test RMSE of 3.73 $\frac{mA}{cm^2}$ and an R-value of 0.63, which correlated to an adjusted MPE of 10.35% using XGBoost. While this performance was commendable, future studies could improve this through further experimentation of different combinations of features and target metrics. In creating these models, we further present an open-source pipeline to aid in future studies of PSCs. This pipeline handles the encoding of the PDP into an ML-readable format while allowing flexibility in which features and targets are used. As perovskite solar cells continue to improve at exceptional rates, by increasing the accessibility of these tools, we hope to aid in this acceleration and propel their efficiency well beyond their silicon crystal counterparts.

References

[1] T. J. Jacobsson, A. Hultqvist, A. García-Fernández, A. Anand, A. Al-Ashouri, A. Hagfeldt, A. Crovetto, A. Abate, A. G. Ricciardulli, A. Vijayan *et al.*, "An open-access database and analysis tool for perovskite solar cells based on the fair data principles," *Nature Energy*, vol. 7, no. 1, pp. 107–115, 2022.

[2] Z. Liu, N. Rolston, A. C. Flick, T. W. Colburn, Z. Ren, R. H. Dauskardt, and T. Buonassisi, "Machine learning with knowledge constraints for process optimization of open-air perovskite solar cell manufacturing," *Joule*, vol. 6, no. 4, pp. 834–849, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542435122001301

[3] A. Kojima, K. Teshima, Y. Shirai, and T. Miyasaka, "Organometal halide perovskites as visible-light sensitizers for photovoltaic cells," *Journal of the american chemical society*, vol. 131, no. 17, pp. 6050–6051, 2009.

[4] NREL.gov, "Best research-cell efficiency chart," Available at https://www.nrel.gov/pv/cell-efficiency.html (2023/11/26).

[5] E. Aydin, E. Ugur, B. K. Yildirim, T. G. Allen, P. Dally, A. Razzaq, F. Cao, L. Xu, B. Vishal, A. Yazmaciyan *et al.*, "Enhanced optoelectronic coupling for perovskite-silicon tandem solar cells," *Nature*, pp. 1–3, 2023.

[6] C. Zou, C. Zhang, Y.-H. Kim, L. Y. Lin, and J. M. Luther, "The path to enlightenment: progress and opportunities in high efficiency halide perovskite light-emitting devices," *ACS Photonics*, vol. 8, no. 2, pp. 386–404, 2021.

[7] D. Mohanty and A. K. Palai, "Comprehensive machine learning pipeline for prediction of power conversion efficiency in perovskite solar cells," *Advanced Theory and Simulations*, p. 2300309.

[8] https://www.ucl.ac.uk, "Perovskite solar cells," Available at https://www.ucl.ac.uk/institute-for-materials-discovery/research/clean-energy/perovskite-solar-cells (2023/11/29).

[9] Y. Liu, X. Tan, J. Liang, H. Han, P. Xiang, and W. Yan, "Machine learning for perovskite solar cells and component materials: key technologies and prospects," *Advanced Functional Materials*, p. 2214271, 2023.

[10] J. Cai, X. Chu, K. Xu, H. Li, and J. Wei, "Machine learning-driven new material discovery," *Nanoscale Advances*, vol. 2, no. 8, pp. 3115–3130, 2020.

[11] A. Jain, H. Patel, L. Nagalapatti, N. Gupta, S. Mehta, S. Guttula, S. Mujumdar, S. Afzal, R. Sharma Mittal, and V. Munigala, "Overview and importance of data quality for machine learning tasks," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 3561–3562.

[12] W. Yan, Y. Liu, Y. Zang, J. Cheng, Y. Wang, L. Chu, X. Tan, L. Liu, P. Zhou, W. Li *et al.*, "Machine learning enabled development of unexplored perovskite solar cells with high efficiency," *Nano Energy*, vol. 99, p. 107394, 2022.

[13] Y. Liu, W. Yan, S. Han, H. Zhu, Y. Tu, L. Guan, and X. Tan, "How machine learning predicts and explains the performance of perovskite solar cells," *Solar RRL*, vol. 6, no. 6, p. 2101100, 2022.

[14] M. D. Wilkinson, "The fair guiding principle for scientific data management and stewardship: Comment," 2016.

[15] M. Mammeri, L. Dehimi, H. Bencherif, M. Amami, S. Ezzine, R. Pandey, and M. K. Hossain, "Targeting high performance of perovskite solar cells by combining electronic, manufacturing and environmental features in machine learning techniques," *Heliyon*, vol. 9, no. 11, 2023.

[16] optuna.org, "Optuna: A hyperparameter optimization framework," Available at https://optuna.readthedocs.io/en/stable/index.html (2023/11/27).