www.solar-rrl.com

Identifying Performance Limiting Parameters in Perovskite Solar Cells Using Machine Learning

*Oliver Zbinden, Evelyne Knapp, and Wolfgang Tress**

Herein, it is shown that machine learning (ML) methods can be used to predict the parameter that limits the solar-cell performance most significantly, solely based on the current density–voltage (*J–V*) curve under illumination. The data (11'150 *J–V* curves) to train the model is based on device simulation, where 20 different physical parameters related to charge transport and recombination are varied individually. This approach allows to cover a wide range of effects that could occur when varying fabrication conditions or during degradation of a device. Using ML, the simulated *J–V* curves are classified for the changed parameter with accuracies above 80%, where Random Forests perform best. It turns out that the key parameters, short-circuit current density, open-circuit voltage, maximum power conversion efficiency, and fill factor are sufficient for accurate predictions. To show the practical relevance, the ML algorithms are then applied to reported devices, and the results are discussed from a physics perspective. It is demonstrated that if some specified conditions are met, satisfying results can be reached. The proposed workflow can be used to better understand a device's behavior, e.g., during degradation, or as a guideline to improve its performance without costly and time-consuming lab-based trial-and-error methods.

1. Introduction

The efficiency of perovskite solar cells (PSCs) has almost doubled to more than 25%[1] in the last decade. This increase in efficiency, together with a plethora of favorable properties such as a good light absorption coefficient,[2,3] relatively long diffusion length and carrier lifetime,[4,5] high defect tolerance,[6,7] tunable bandgap,[8] as well as easy and cheap production, makes perovskites a very interesting group of materials for solar cell application. However, the Shockley–Queisser limit[9] has not been reached yet, reproducibility and stability issues have to be overcome as well. Around the same time PSCs were invented, artificial intelligence became popular (again), especially the field of machine learning (ML). A variety of many different research fields have been profiting from this development and several studies have shown where ML tools can be applied in the field of PSCs, for example, to find materials that form perovskite structure,[10] predict the bandgap energy E_g,[11] find lead free perovskite materials,[12] predict photovoltaic parameters and power conversion efficiency,[13] or predict long-term performance and/or degradation.[14–16] In this work, we want to identify quantities that limit the PSC performance based on the devices' current–voltage curves. In a first step, we generate current density–voltage (*J–V*) curves of different PSCs using the drift–diffusion device simulator *Setfos*.[17] Based on the simulated data, we want to train an ML algorithm that is able to detect limiting parameters in a PSC. Known applications of ML in the field of PSC that go in a similar direction are, for example, identifying the dominant recombination loss,[18] or finding the optimum annealing temperature and solvent ratios, determined based on *J–V* curves.[19] Similar to what we present here, parameter variation and comparison to a reference device[20] have been shown to be able to detect the major loss mechanism upon degradation. In this last example, the classification was done without ML. What is new in our work is that we combine ML with standard simulation and analyzing tools, to identify limiting parameters.

Our approach can be seen as an assistant in the lab to guide the manufacturer to changes in the recipe or procedure toward devices with higher efficiencies. Furthermore, our approach can be used to suggest the most likely parameter that changed, e.g., if an applied passivation layer really does what it is supposed to do, compared to the reference.

O. Zbinden, E. Knapp, W. Tress
Institute of Computational Physics
Zurich University of Applied Sciences
Technikumstrasse 71, Winterthur 8400, Switzerland
E-mail: wolfgang.tress@zhaw.ch

O. Zbinden
Institute for Computational Science
Universität Zürich
Winterthurerstrasse 190, Zurich 8057, Switzerland

The ORCID identification number(s) for the author(s) of this article can be found under https://doi.org/10.1002/solr.202300999.

DOI: 10.1002/solr.202300999

2. Simulation

We used device simulation because making many identical devices or changing only one device parameter in a defined way is experimentally challenging without affecting other properties. To our knowledge, there is no public data base that fulfills our requirements in a reasonable amount to get enough data for ML, so device simulation is a suitable tool to overcome this problem. The goal of these simulations is not to reach theoretically perfect, but realistic results that are close to what is commonly reached and reported.

Starting from three simulated reference devices (described in left of **Figure 1** and Table S1, Supporting Information), we systematically varied one device parameter at a time to generate ≈11150 *J*–*V* curves, as shown in Figure S5–S14, Supporting Information. These reference devices (*J*–*V* curves in Figure 1) have slightly different parameters. Devices 1 (blue) and 2 (orange) only differ in electron and hole capture rate, device 2 has a capture rate that is reduced by a factor of 10 compared to device 1. These defects are responsible for Shockley-read-hall (SRH) recombination. In reference device 3 (green), compared to the other two, the trap energies and densities are reduced from 5.10^{17} to $1.2\ 10^{16}\ \text{cm}^{-3}$; capture rates are asymmetric, ion densities, as well as μ_e and μ_h in perovskite are different. A detailed description can be found in Table S1, Supporting Information. Parameters that were kept the same for all models are listed in Table S2, Supporting Information. The reason for using different reference devices is that this allows the ML models to better generalize their classification task because the ML model is trained with a larger data set that covers more than only one case. Note that, in the device stack illustrated in Figure 1a, the bottom and top perovskite interlayers allow to control surface recombination and charge transport across the interface in the simulation. Variation of different parameters mimics effects that lower the overall performance which can be related to problems during fabrication, or environmental conditions that cause degradation.

In total, there are 20 varied parameters: internal quantum efficiency/generation coefficient, thickness of the hole transport layer, perovskite, the electron transport layer, acceptor and donor doping in the perovskite layer, anion and cation density, and recombination in the bulk and at the surface, different electron and hole mobilities, acceptor doping in and hole mobility in the hole-transport layer (HTL), donor doping in the electron-transport layer (ETL), trap energy and trap density in perovskite, as well as different bottom and top electrode work functions. The default parameters for ETL and HTL were chosen based on what is known for the common representatives TiO_2 and 2,2′,7,7′-Tetrakis[N,N-di(4-methoxyphenyl)amino]-9,9′-spirobifluorene (spiro-MeOTAD). Together with the respective sweep ranges, all parameters are listed in Table S3, Supporting Information. We only vary one parameter at a time. Otherwise, there is a chance of correlation and it would be difficult to reach the main objective of this work, which is to find the one parameter that limits the device performance most. We neglected hysteresis in our model, and only the steady state is simulated under one sun illumination. The resulting sets of *J*–*V* curves will then be used for the ML training.

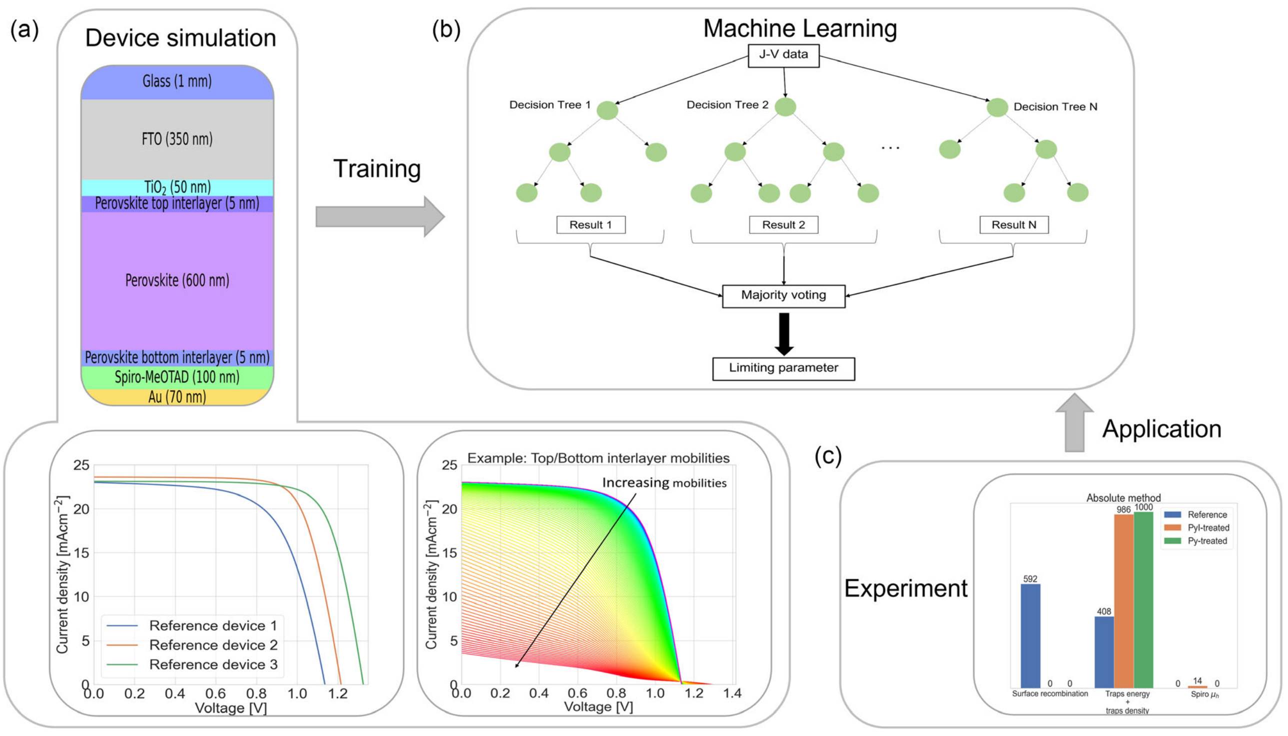

Figure 1. Used framework to identify limiting parameters in perovskite solar cells. a) Device simulation is described in Section 2. The general architecture we used is shown on top, bottom left shows the *J*–*V* curves of the different models that were used, and right to it, one *J*–*V* curve set of a parameter variation is shown. Interlayers are introduced into the simulation to describe surface recombination and interfacial charge transport mimicked by bulk properties of these thin interlayers. b) The simulated data was then used to train different ML models. Here, for illustration, a Decision Tree and a confusion matrix are shown in a simplified and reduced version, based on a test run trained with a small fraction of the data set. This is the topic of Section 3. c) To show the practical relevance of this study, experimental data from literature is applied to the ML models, and the results are discussed in Section 4.

3. ML

3.1. Methods

With the simulation results in hand, we want to find out if it is possible to identify the one parameter that changed with respect to the reference device based on information from the *J*–*V* curves.

The targets we want to classify for are the changed parameters. From now on, this method is called *absolute method*. It is important to mention that we do not want to make predictions about performance or stability, e.g., forecast the device's lifetime, but say which parameter contributed most to a detected loss in performance, or what limits a freshly made cell most, compared to a similar device without such a drawback.

As a starting point, we tested different tree-based methods, such as Decision Trees (DTs) and Random Forests (RFs),[21] both with Scikit-learn,[22] XGBoost (XGB),[23] neural networks (NNs, with Keras[24]), and support vector machines (SVMs).[25] All code is implemented in Python.[26]

We tried different inputs, such as many points on the *J*–*V* curve, systematically reduced the size of the input vectors, and compared the changes in test accuracies. It turned out that it is sufficient to take the open-circuit voltage V_{oc}, short-circuit current J_{sc}, fill factor ff, and maximum power conversion efficiency mpce as input parameters. We decided to only use four features, where one could argue if mpce is really needed since it depends on the other three figures of merit. It will be discussed later why this information seems to be enough for ML to distinguish between the different cases and predict changing parameters correctly with high accuracy.

The results in **Table 1** show that it is possible for different algorithms to differ between simulated changes and assign the correct changing parameter with accuracy well above 80%. For DT, RF, and XGB, the accuracies were determined with a 1000 times repeated random sub-sampling validation, NN and SVM, due to calculation time, it is a 100 times repeated random sub-sampling validation. The last two performed much weaker, both around 75%, so we excluded them from further consideration. The reason for the low performance of NN and SVM is not because they are incapable of such tasks, but because our hardware resources did not allow optimization within reasonable timescales. The used train-test split was 75:25 and contained all data based on the simulated reference devices. It is not surprising that RF and XGB perform better than DT, because the simpler DTs are more prone to overfitting, which means that it might memorizes peculiarities from the training data instead of finding predictive rules.[27] The reason we did not proceed with XGB was that its performance was very close to the RF, but calculation time was much longer. Note that an optimized design for the NNs might be a good candidate, as well as an SVM with suitable choices of hyperparameters.

Table 1. This table shows results of the 1000 times repeated random sub-sampling validation for both, DT and RF. The standard deviation for the different cases allows uncertainty quantification.

ML algorithm	Absolute		Difference	
	Test accuracy [%]	Standard deviation σ [%]	Test accuracy [%]	Standard deviation σ [%]
RF	86.59	0.63	82.81	0.69
DT	84.86	0.74	80.73	0.72

However, because we start with defined reference devices and compare absolute values with respect to the sweeping parameter, the model might be not as flexible, and comparison with devices that differ from the simulation in parameters, not in architecture, can be difficult. This problem can be overcome, or at least reduced, if we do not train the model with all absolute values, but by the difference with respect to the respective reference device (V_{oc}–$V_{oc,ref}$, J_{sc}–$J_{sc,ref}$, ff–ff_{ref}, and mpce–$mpce_{ref}$, where the subscript $_{ref}$ denotes the reference). This would also have the benefit that, when testing the algorithm on experimental data, PSCs can be compared directly with a reference cell. We call this the *difference* method.

3.2. Discussion/Validation

Randomly picking one *J*–*V* curve from Figure S5–S14, Supporting Information, and saying which parameter is limiting seems extremely difficult, so one might get the impression that the surprisingly good results may not be justified. We will now discuss how these high accuracies are possible, and under which conditions.

Figure 2 shows data extracted from Figure S5–S14, Supporting Information, the individual traces represent the different parameter variations. This gives an insight how V_{oc}, J_{sc}, and ff are correlated when a certain parameter is varied. One can already see from these plots that it is hard to distinguish the different parameter variations in some area, close to the initially modeled reference devices, and easier in other areas. This leads us to the first restriction. The changed parameter should cause a notable change in the *J*–*V* curve, or at least in one input parameter (V_{oc}, J_{sc}, ff, and mpce). In Figure 2, bottom right (and Figure S4, Supporting Information), one can get a feeling what "notable" means in this sense. There, we show all the misclassified candidates from the test set. Comparing this to the bottom left of Figure 2, we see that all errors occur where a lot of overlapping is found. In this crowded region, where misclassifications most likely occur, fluctuations are also likely to appear in experimental measurements. Furthermore, two fabricated devices that are assumed to be identical may differ slightly. Of course, data in the vicinity of the simulated reference devices could be excluded because it is very difficult to get any useful information from this region. However, putting this in perspective of Figure S3, Supporting Information, this impression is questionable. One can see that even in the region where one assumes that it is very hard to make correct predictions, as implied by Figure S4, Supporting Information, there are still correct classifications and, furthermore, they outweigh the incorrect ones by far. One can get a better impression when we change to a 3D V_{oc}–J_{sc}–ff space as shown in **Figure 3**. Of course, changes in different parameters may lead to similar solutions, which can cause misclassification in the algorithms. Therefore, it is important to understand how changes in one parameter influence the *J*–*V* curve, as shown in Figure S5–S14, Supporting Information. For example, one can see that acceptor doping and cation density

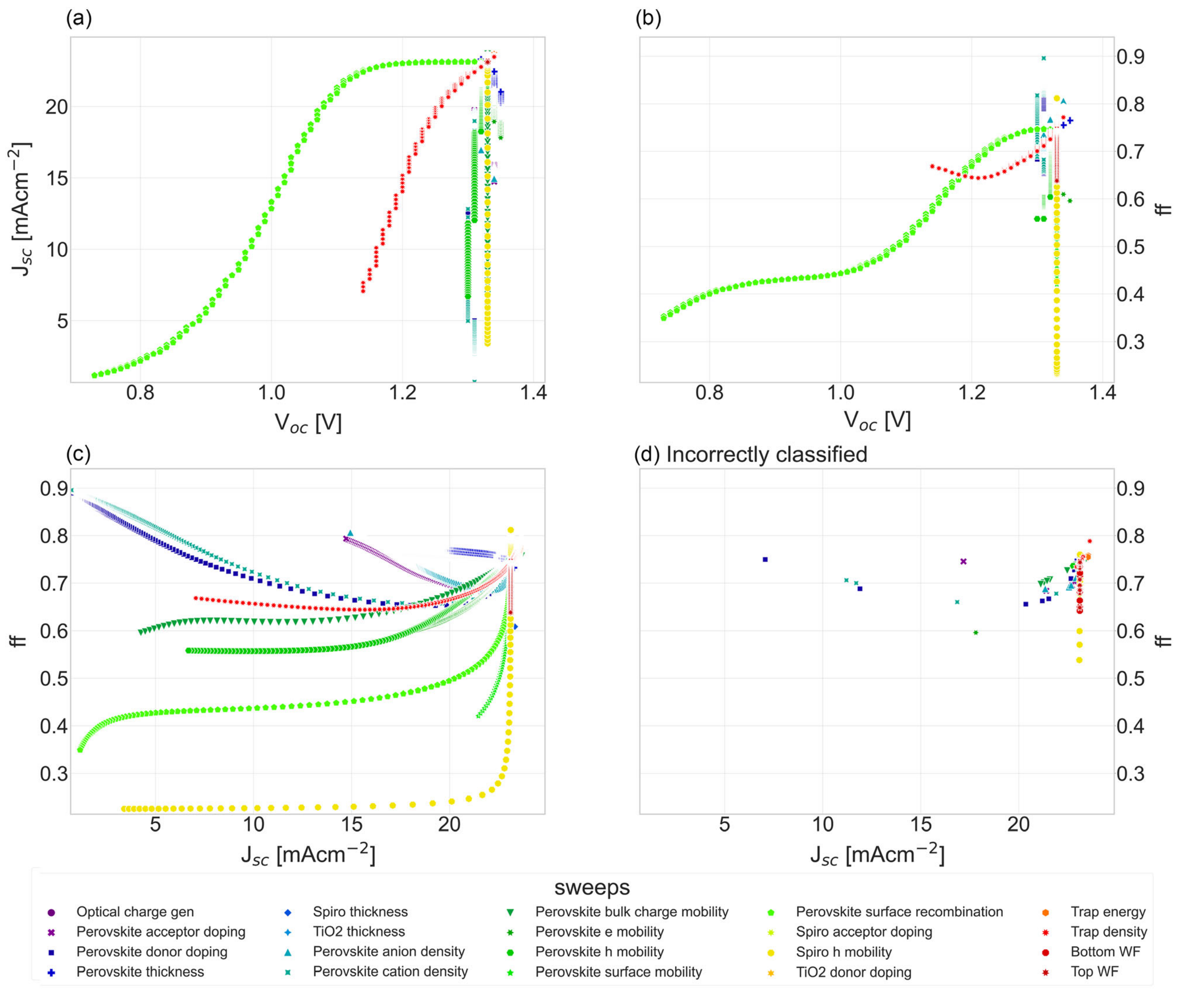

Figure 2. a) Evolution of J_{sc}–V_{oc}, b) ff–V_{oc}, and c) ff–J_{sc} dependent on changing parameter from one of our reference devices. One can see that J_{sc}–V_{oc} and ff–V_{oc} do not show unique combinations due to overlapping. Only the surface recombination and trap density are clearly separable from the rest. Ff–J_{sc} provides much more insight, at least when there is some notable change with respect to the reference model. There, most changes in parameters follow an individual trend yield unique V_{oc}–J_{sc}–ff combinations. d) Misclassified objects in ff–J_{sc} plane for one test run from the entire data set, based on all three simulated reference devices. The distribution of these devices is mainly explained by the third restriction.

in Figure 2 and 3 show a very similar behavior. An explanation for this can be found in Table S3, Supporting Information, and Poisson's equation, where both quantities contribute equally in a sum as space charges.

Because we sweep over the same, or at least overlapping, regions, the confusion in classification of these parameters can be explained. Or, concluding this first restriction, if the measurement is too far away from the training data, the classification may fail because the chosen tree based classifiers are weak in extrapolation.[28] Adding more, or using different, simulations could solve this problem.

The second restriction is that the device(s) under examination should be close to one of the simulated reference devices if the most significant limitation was absent. This is based on the assumptions on which the reference devices are simulated, and because we only swept one physical parameter at a time. If the actual device differs significantly from the simulated model, the classification will not be reliable because of the weakness in extrapolation mentioned in restriction 1. If there are several effects that limit the device in a similar amount, it is possible that other interaction effects play a role, and that the device would not just lie between the two respective curves in the diagram. One possibility to overcome this problem could be to add models that consider several changing parameters at a time, which was not part of this work.

In short, the second restriction is related to parameter variation and data simulation. We only changed one parameter at a time. Multiple changes are much more complex to interpret, because it is hard to say which one of those limits device performance most.

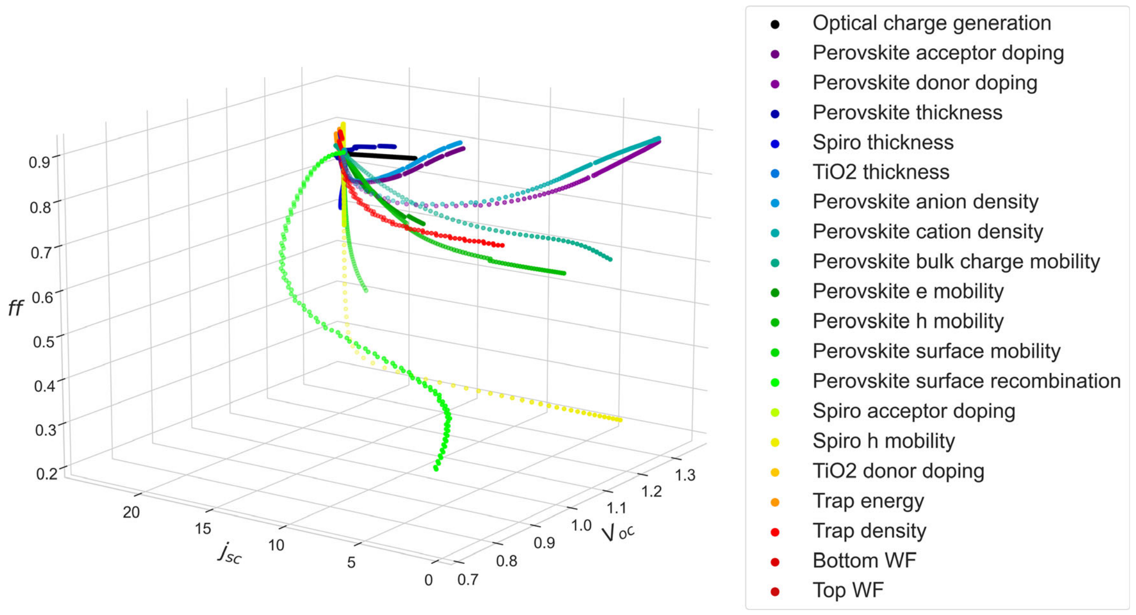

Figure 3. A perspective of the same data as in Figure 2 in 3D. This gives a much clearer picture how changes of one parameter change these key figures and what trends they follow. It also helps to understand what patterns the different algorithms learn to reach the reported accuracies.

Third, if the measured device is already close to the simulated reference, there is a high chance of misclassification, as can be inferred from Figure 2 and 3. In this case, there is either no need for further optimization because the manufactured device already behaves as desired, or the model has to be improved.

The last restriction relates to simulation and ML. The ML algorithms are trained with simulated data. Simulation always includes some simplifications and assumptions. It has to be decided how the simulated reference cell should look like, and physical parameters have to be fixed. It was, for example, not possible to include a mesoporous layer, or to add information about grain sizes in perovskite. To overcome these issues, ML training should be done with experimental data when they become available in the future.

To test the robustness of the model, noise was added and the results were compared to the clean simulation data. Unsurprisingly the accuracy decreased, but only caused by data points in the region that already caused problems without noise (Figure S4, Supporting Information).

4. Testing/Applications

It is important to not only use simulation, but also to apply the results to experimental data for demonstration purposes. To show the benefit of our approach, we will apply the RF to data from literature, where the device architecture is close enough to one of our reference devices. The experimental data have to obey the already mentioned restrictions and must be reported for specific cells. In all cases, we trained and tested the RF 1000 times, as before. After each training cycle, the model is then applied to the experimental data set each time, to have statistically significant results.

In the first example from Belich et al.[29], (Data from Figure S5, supporting information), a PSC was encapsulated and then exposed to standard conditions under an ambient environment for more than 1000 h. The reference cell (before stressing) is close to the optimum of our simulation, where it can be difficult to make accurate predictions. In the ff–J_{sc} plane, it is very close to the path of changes in anion density, which is detected by the RF very well. Since it is difficult to make predictions for cells close to the optimum, we should not pay too much attention to this device. In contrast, because their initial device is so close to our simulation, this makes the device after stressing an ideal candidate to test both, the absolute method as well as the difference method. After stressing, traps that cause SRH recombination are identified as limiting parameter (**Figure 4**, left). This is consistent with the conclusion of the authors.[29] Comparing the output before and after stressing by the difference method (Figure 4, right), the vast majority suggests that it is surface recombination. Because this method does not suggest the limiting parameter of an individual *J–V* curve, but the difference between curves, this is not a contradiction to the finding with the other method. Here, it suggests what has to be changed in a device to get from one *J–V* curve to another one. Surface recombination goes hand in hand with (surface) traps and therefore the result from the difference method is still consistent with the one obtained by the absolute method. With the difference method, perovskite thickness is also identified as limiting parameter in a small fraction of test runs. We will now explain why the algorithm could come to this result. The *J–V* curves corresponding to different perovskite layer thicknesses in the right graph in Figure S10, Supporting Information, show a similar behavior as the measurements before and after stressing the devices from[29] listed in **Table 2**. V_{oc} decreases slightly as J_{sc} increases, also causing the ff to decrease. The algorithm interprets this (unexpected) experimental observation as a change in perovskite layer thickness.

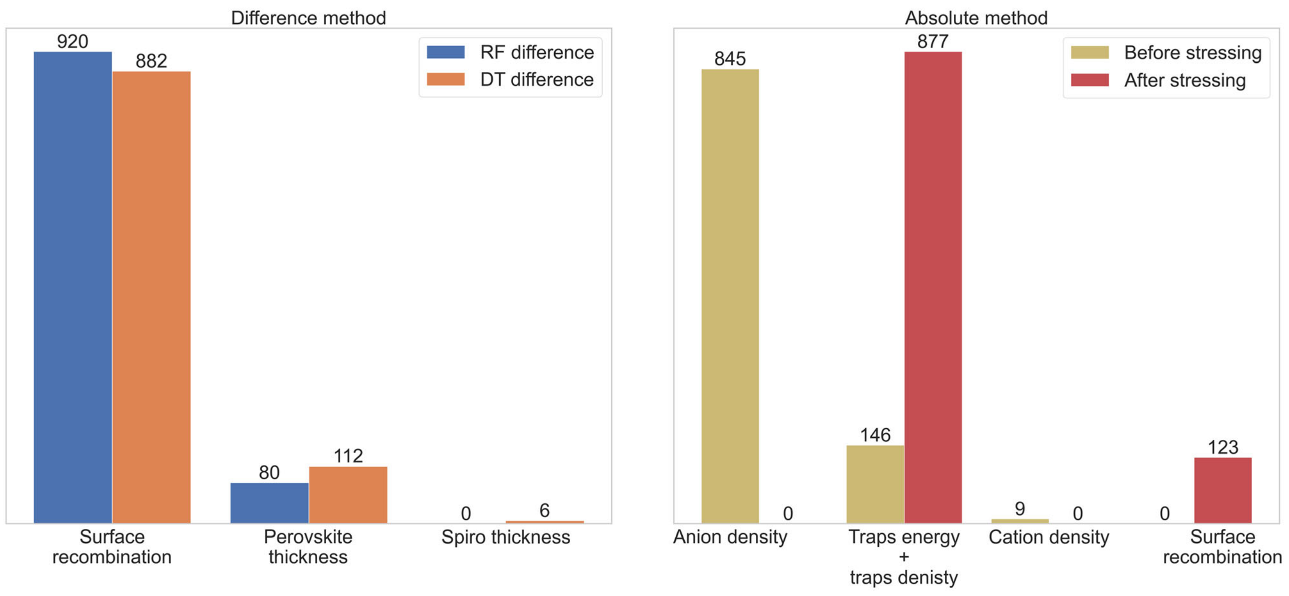

Figure 4. LHS: Before stressing the cell, it seems that the most limiting factor is related to ion density. After exposing the device to standard conditions, the RF finds traps energy to be most influential. RHS: Comparing the device before and after stressing, both DT and RF propose surface recombination to be the parameter that changed the most.

Table 2. List of key figures taken from refs. [29,30] that we applied on our ML algorithms.

Data source	V_{oc} [V]	J_{sc} [mA cm^{-2}]	ff	mpce
Belich et al.				
Before stressing	1.09	22.6	0.78	19.2
After stressing	1.07	23.3	0.75	18.9
Du et al.				
Untreated	0.974	22.50	0.693	15.20
PyI treated	1.146	22.88	0.773	20.26
Py treated	1.138	22.43	0.715	18.26

In the second example, Du et al.[30] (data from Table S3, Supporting Information, forward scan, Supporting Information) applied different materials for surface passivation. Again, we first look at the method where the individual device measurements are fed to the algorithm. The algorithm suggests surface recombination as the most limiting factor for the untreated device. After the different treatments with Py or PyI to passivate the perovskite surface, this seems to be not an issue any more, the results can be found in **Figure 5**. The treated devices are now very close to one of our reference devices, which means that there is a higher uncertainty of misclassification. To find the parameters that limit these optimized devices, new reference devices could be added to the training set. The important conclusion is that the reference device, especially V_{oc} and ff, seems to be limited by surface recombination, as found by our algorithm. A decrease of surface recombination could be explained by the reported reduction of surface roughness shown there in Figure 2.[30] Considering the differences before and after the

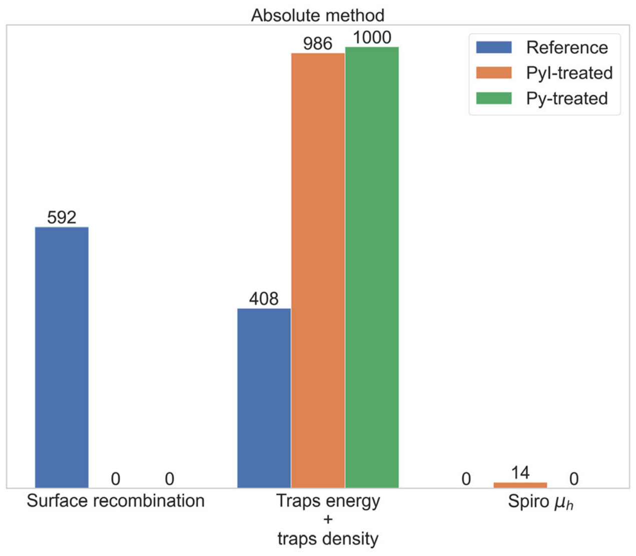

Figure 5. Surface recombination is the most likely limiting effect in the reference cell. This completely vanishes after both treatments. Because this is the goal of surface passivation, it could be an indication that the limiting effect can be predicted correctly. After treatment, mainly traps are identified.

treatments, traps are found to be responsible for the changes for both RF and DT exclusively.

5. Conclusion

We have introduced a new application of ML that can, based on J–V data, identify the device parameter which limits a PSC device's efficiency most. The used models are based on simulated data, with ML classification accuracies well above 80%. The models were applied to experimental data, where devices were stressed, or where cells were passivated and compared to a reference cell. Theoretically understanding how a specific change influences a device, the ML results can further be validated. Additionally, with small adaptations in the data input (difference method), we can determine which parameter changes most upon a certain treatment, such as passivation or stressing a device. There are some restrictions on the applicability of our models: 1) the measurement should be not too far away from the simulated data; 2) In absence of the most significant limiting factor, the cell should perform comparable to a simulated reference device; and 3) There is a high chance of misclassification if the measured device is close to the simulated ref. [4]). The ML algorithms are trained with simulated data. Keep in mind that simulation always includes restrictions and simplifications compared to the real physical world.

Having these restrictions in mind, our approach can be used in different situations. As a quality control, e.g.: if a lab follows a standard recipe and some cells do not perform as expected. With a simulated model of the desired architecture, one can narrow the limiting effect and may adapt fabrication. This will save time and material, because repeated trial and error can be avoided. It could also be interesting for future industrial application, to check if changes in the recipe or architecture, for example, a passivation layer, lead to better results. Another suggested application is to identify the most limiting parameter after stress or degradation. This can help to understand processes and the device itself better, and it could potentially lead to more stable devices.

In this work, we focused on one PSC architecture. However, it can be generalized or adapted to other designs and specific problems easily. Simulation and ML training take some time, but testing data with the classifier only takes seconds to minutes. This is a huge advantage compared to trial-and-error-based methods in the lab to find a source of undesired performance loss, because once device simulation and ML training are completed, classification can be performed within seconds to minutes, depending on the number of iterations chosen.

Supporting Information

Supporting Information is available from the Wiley Online Library or from the author.

Acknowledgements

This research received funding from the European Union's Horizon 2020 research and innovation program under grant agreement no. 851676 (ERC StGrt). O.Z. and W.T. acknowledge funding by ZHAW digital in the framework of a DIZH fellowship.

Conflict of Interest

The authors declare no conflict of interest.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Keywords

machine learnings, optimizations, perovskite solar cells

Received: December 1, 2023
Revised: January 12, 2024
Published online: February 10, 2024

[1] Best Research-Cell Efficiency Chart Plotted by National Renewable Energy Laboratory, USA, https://www.nrel.gov/pv/cell-efficiency.html (accessed: March 2023).
[2] S. De Wolf, J. Holovsky, S.-J. Moon, P. Löper, B. Niesen, M. Ledinsky, F.-J. Haug, J.-H. Yum, C. Ballif, *J. Phys. Chem. Lett.* **2014**, *5*, 1035.
[3] M. Kato, T. Fujiseki, T. Miyadera, T. Sugita, S. Fujimoto, M. Tamakoshi, M. Chikamatsu, H. Fujiwara, *J. Appl. Phys.* **2017**, *121*, 115501.
[4] S. D. Stranks, G. E. Eperon, G. Grancini, C. Menelaou, M. J. P. Alcocer, T. Leijtens, L. M. Herz, A. Petrozza, H. J. Snaith, *Science* **2013**, *342*, 341.
[5] G. Xing, N. Mathews, S. Sun, S. S. Lim, Y. M. Lam, M. Grätzel, S. Mhaisalkar, T. C. Sum, *Science* **2013**, *342*, 344.
[6] D. R. Ceratti, Y. Rakita, L. Cremonesi, R. Tenne, V. Kalchenko, M. Elbaum, D. Oron, M. A. C. Potenza, G. Hodes, D. Cahen, *Adv. Mater.* **2018**, *30*, 1706273.
[7] K. X. Steirer, P. Schulz, G. Teeter, V. Stevanovic, M. Yang, K. Zhu, J. J. Berry, *ACS Energy Lett.* **2016**, *1*, 360.
[8] Z. Zhang, M. Wang, L. Ren, K. Jin, *Sci. Rep.* **2017**, *7*, 1918.
[9] W. Shockley, H. J. Queisser, *J. Appl. Phys.* **1961**, *32*, 510.
[10] X. Li, Y. Dan, R. Dong, Z. Cao, C. Niu, Y. Song, S. Li, J. Hu, *Appl. Sci.* **2019**, *9*, 24.
[11] G. Pilania, A. Mannodi-Kanakkithodi, B. P. Uberuaga, R. Ramprasad, J. E. Gubernatis, T. Lookman, *Sci. Rep.* **2016**, *6*, 19375.
[12] S. Lu, Q. Zhou, Y. Ouyang, Y. Guo, Q. Li, J. Wang, *Nat. Commun.* **2018**, *9*, 3405.
[13] J. Li, B. Pradhan, S. Gaur, J. Thomas, *Adv. Energy Mater.* **2019**, *9*, 1901891.
[14] M. Srivastava, J. M. Howard, T. Gong, M. Rebello Sousa Dias, M. S. Leite, *J. Phys. Chem. Lett.* **2021**, *12*, 7866.
[15] J. M. Howard, E. M. Tennyson, B. R. Neves, M. S. Leite, *Joule* **2019**, *3*, 325.
[16] R. J. Stoddard, W. A. Dunlap-Shohl, H. Qiao, Y. Meng, W. F. Kau, H. W. Hillhouse, *ACS Energy Lett.* **2020**, *5*, 946.
[17] Semiconductor Simulator (Setfos), https://www.fluxim.com/ (accessed: December 2021).
[18] V. M. Le Corre, T. S. Sherkar, M. Koopmans, L. J. A. Koster, *Cell Rep. Phys. Sci.* **2021**, *2*, 100346.
[19] Z. Ren, S. Tian, T. Heumueller, E. Birgersson, F. Lin, A. Aberle, S. Sun, I. M. Peters, R. Stangl, C. J. Brabec, T. Buonassisi, F. Oviedo, H. Xue, M. Thway, K. Zhang, N. Li, J. D. Perea, M. Layurova, Y. Wang, in *2019 IEEE 46th Photovoltaic Specialists Conf. (PVSC)*, IEEE, Chicago, IL, USA **2019**.
[20] A. Julien, J.-B. Puel, J.-F. Guillemoles, *Energy Environ. Sci.* **2023**, *16*, 190.
[21] T. K. Ho, in *Proc. of 3rd Int. Conf. on Document Analysis and Recognition*, IEEE, Montreal, QC, Canada **1995**.
[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg,

J. Vanderplas, D. Passos, A. Cournapeau, M. Brucher, M. Perrot, E. Duchesna, *J. Mach. Learn. Res.* **2011**, *12*, 2825.
[23] T. Chen, C. Guestrin, CoRR **2016**.
[24] Keras, https://keras.io (accessed: June 2022).
[25] C. Cortes, V. Vapnik, *Mach. Learn.* **1995**, *20*, 273.
[26] Python Language Reference, Version 3.6, http://www.python.org (accessed: December 2021).
[27] T. Dietterich, *ACM Comput. Surv.* **1995**, *27*, 326.
[28] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, *Nat. Rev. Phys.* **2021**, *3*, 422.
[29] N. A. Belich, A. A. Petrov, P. A. Ivlev, N. N. Udalova, A. A. Pustovalova, E. A. Goodilin, A. B. Tarasov, *J. Energy Chem.* **2023**, *78*, 246.
[30] Y. Du, J. Wu, X. Zhang, Q. Zhu, M. Zhang, X. Liu, Y. Zou, S. Wang, W. Sun, *J. Energy Chem.* **2021**, *52*, 84.