Hierarchical Training using SHAP for Interpretable Organometal Halide Perovskite Bandgap Prediction

Kenneth Kjærgaard Malowanczyk
kenneth.kier95@gmail.com
Aalborg University
Aalborg, Denmark

Andreas Dahl Nielsen
andreas.dahl.nielsen@gmail.com
Aalborg University
Aalborg, Denmark

Sebastian Reidar Petersen
sebpetersen@outlook.com
Aalborg University
Aalborg, Denmark

ABSTRACT

During the last years, model interpretability has become an increasingly researched aspect in machine learning. Its ability to provide an explanation of the model can, from one side, increase the trustworthiness of the predictions and from the other side help in identifying hidden trends, thus going beyond the use of machine learning as a black box. In this paper, we propose a hierarchical training method to interpret convolutional neural networks trained on tabular data, and apply it for bandgap prediction of organometal halide perovskites, by assigning importance values to features. The feature space includes properties of the elements, precursors, and perovskite crystal structures, for a total of 39 features, which can be combined together. Using a Weight Parameter Saving Method, we are able to reuse previously trained network's weights for training the next network, achieving faster convergence and better prediction performances. Using Shapley Additive Explanations for approximating feature importance and hierarchical training, a minimal feature set needed for bandgap prediction (within a squared error of 0.1) is found. This has the effect of reducing the feature space, while preserving the predictive performance of the model.

KEYWORDS

Machine Learning, Interpretability, Hierarchical Training, Material Science

ACM Reference Format:
Kenneth Kjærgaard Malowanczyk, Andreas Dahl Nielsen, and Sebastian Reidar Petersen. 2021. Hierarchical Training using SHAP for Interpretable Organometal Halide Perovskite Bandgap Prediction. In *Gold Coast '21: 30th ACM International Conference on Information and Knowledge Management, November 01–05, 2021, Gold Coast, Queensland, Australia.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Modern day machine learning has become a critical function within many fields. Its ability to go through large amounts of data and make accurate predictions in areas from playing videogames [7] to recommending items based on website behavior [1] has made it a great subject of focus. While machine learning models offer great benefits in many areas, it usually comes at the cost of its understandability. Most modern machine learning algorithms are considered black boxes, which means that the predictions they make are without explanation and the underlying models' behavior is hard, if not impossible, to understand. This makes machine learning a more sceptical choice in fields, where the need for understanding the model is crucial in order to trust its predictions [8]. The area of model interpretability tries to solve this issue via the introduction of algorithms, that can interpret black box models while attempting to maintain a high accuracy. This interpretation allows for better insight into how and why a certain prediction was made and understand the underlying strategy that the model utilizes [8]. Such knowledge will allow for further adoption of machine learning in various fields, since better model understanding will increase the usefulness of machine learning for other fields than computer science, making the model more transparent, trustworthy, and instructive. An example of a field where hesitation of machine learning adoption exists is material science, where machine learning can accelerate the calculations, while interpretations are needed to disclose the structure-property relationships [10]. Interpretable machine learning for material science can help with understanding how the material works, as well as to identify trends and descriptors, which can accelerate the discovery of novel materials [4].

In this paper, we present a post-hoc interpretation of a model within the field of computational materials, used for predicting the bandgap of Organometal Halide Perovskites (OMHP) [9]. The bandgap of the photoactive material is a descriptor of the efficiency of a solar cell [3]. An OMHP is formed by organic molecules in an inorganic template. There is virtually several thousand molecules which can be used, making the investigation space too large to be simulated via atomistic quantum mechanical simulations, and only a small subset has so far been studied [2]. The interpretable method we propose can assist material scientists by getting a better understanding of the machine learning model they utilize, to help them understand which features are important and how to better tune them for achieving an expected bandgap.

This work uses the model and data from research by Saidi Et. al. [9], where they propose a hierarchical convolutional neural network for predicting the bandgap of various OMHP. Our research builds upon their model with the proposal of an interpretable method using SHapley Additive exPlanations (SHAP) [6] to assign importance values to features, while retaining the models prediction capabilities. Unlike [5][12], which assume that the features are independent and may give very limited explanations, this work considers the relations among features and creates multi-level feature combinations.

The model is trained hierarchically and the unimportant features are removed iteratively. A weight parameter saving method is proposed to reduce training time of each iteration. We test our method on the full feature set described in [9], where our method is able to find features of similar importance.

2 METHOD

A convolutional neural network is used that is trained with a hierarchical method, using SHAP values to tune input features and a parameter saving method to reduce training time. The model takes as input a set of feature combinations F^n describing an OMHP and outputs a bandgap prediction B. The features F^n are computed by taking the vector of features F and multiplying with the transpose F^T, combining the features with each other. Any redundant features are removed leaving only unique combinations. This is to create combinations that preserve relations between features due to using a convolutional neural network, where the kernel makes nearby pixels in an image more strongly related than distant ones, and SHAP which assumes that features are independent [6]. The features are standardized by subtracting the mean value of each feature, and dividing it by the standard deviation [9].

2.1 SHAP

SHAP is a game theoretic approach that unifies several post-hoc interpretability methods under one set of algorithms for computing feature importance through the use of Shapley values [6]. In this paper, we use the Deep SHAP method [6] which combines the ideas of Deeplift [11] with Shapley values. In Deeplift, feature importance is computed by assigning each neuron y_i in a network a reference value, which is used to compute how much it causes the output o_j of its connected neuron in the next layer to deviate from its original value. Deep SHAP assumes that features are independent and the model is linear, by linearizing all nonlinear components of the model. Feature contribution is approximated by splitting up the network and computing SHAP values recursively for smaller parts of the model, summing up to the whole model. This is done through the usage of Deeplift's multipliers which can be seen in Equation 1 and is computed in a backpropagation manner.

$$m_{y_i o_j} = \frac{\phi_i\left(o_j, y\right)}{y_i - E\left[y_i\right]} \tag{1}$$

The multiplier $m_{y_i o_j}$ represents how much a neurons y_i differentiation from its reference value causes its output neuron o_j to differ from its reference value. The multipliers are computed for all the neurons in the model with respect to their connected neurons, one layer at a time. ϕ_i is the feature attribution and $E\left[y_i\right]$ is represented by the Deeplift summation-from-delta equation $\sum_{i=1}^{n} C_{\Delta y_i \Delta o} = \Delta o$. When computing a neurons contribution to the models output o_{out}, we sum all multipliers for neuron y_i and subsequently all previous neurons x_j, leading up to the output as seen in Equation 2. This is similar to the chain-rule.

$$m_{y_i o_{out}} = \sum_{j=1}^{2} m_{y_i o_j} m_{x_j o_{out}} \tag{2}$$

Once the contribution effect of each neuron on the output has been computed, the attribution value (SHAP value) of feature f can be approximated to be its multiplier, as seen in Equation 3.

$$\phi_i\left(o_{out}, f\right) \approx m_{y_i o_{out}}\left(y_i - E\left[y_i\right]\right) \tag{3}$$

2.2 Hierarchical Training Method

The Hierarchical training method can be seen in Figure 1. An initial dataset is used from which features are extracted and standardized into a feature set, as seen by the leftmost box in Figure 1. From here, the initial second-level features are computed, which is done following Equation 4, where F^2 is the set of second-level features, f^1 is a single feature, N is the number of features and F^1 is the set of single features. These features consists of pair-wise combinations created by multiplying the single features with each other.

$$F^2 = \{f_k^1 \times f_m^1 | 1 \leq k < m \leq N, f^1 \in F^1\} \tag{4}$$

After computing F^2 features, the model is trained. This first training run will serve as a baseline for how the model performs with all features. From here, the trained model and the feature set will be used together in SHAP to compute the feature importance for all features, as seen by the rounded box in Figure 1. These SHAP values are used to determine unimportant features which then will be removed. For the first run, 10% of the worst features are removed and the model is trained again on the updated feature set, following the topmost line in Figure 1. If performance increases, an additional 10% features will be removed, if performance decreases, half of the most recently removed features will be added back. This loop of adding/removing features will continue until no more features can be removed. Between each training run, the model will reuse the weight parameters from the past run, to reduce the needed training time, as symbolized by the second topmost line in Figure 1. This is explained in greater detail in subsection 2.3.

$$F^n = \{f_k^{n-1} \times f_m^{n-1} | \exists f_o^{n-2} \in F^{n-2}, f_o^{n-2} \subset f_k^{n-1} \wedge f_o^{n-2} \subset f_m^{n-1}, \\ 1 \leq k < m \leq N^{n-1}, f^{n-1} \in F^{n-1}\} \tag{5}$$

When no more features can be removed without decreasing performance, the best model and its feature set is selected, and the method starts anew as symbolized by the bottom line in Figure 1. From here, 3rd-level features are generated from the remaining 2nd-level features, by examining the initial characters of the features, as shown on Equation 5, where F^n is the n'th level features. As an example, using 4th-level features ABCD and ABCE, with the same 3rd-level subset ABC, the feature ABCDE will be computed.

2.3 Model Parameter Saving

To decrease training time, we propose a Model Parameter Saving method. Using the hierarchical training method we make minor changes to the model each iteration, by removing the least important features in the feature set. As we only remove the least important features, the impact on the functions plane will also be small and the plane will be very similar with regards to the global minimum. As we use a convolutional layer in the first layer of the network, no matter how many input features we have, the first level will be the same, however the output from the convolutional layers is different for the distinct input features. Therefore, instead of training again with randomized weights, we take the weights from the best model and apply them to the new network, taking into consideration that it might overwrite some knowledge, as it is near impossible to know exactly what to save and what to remove from this area of the network.

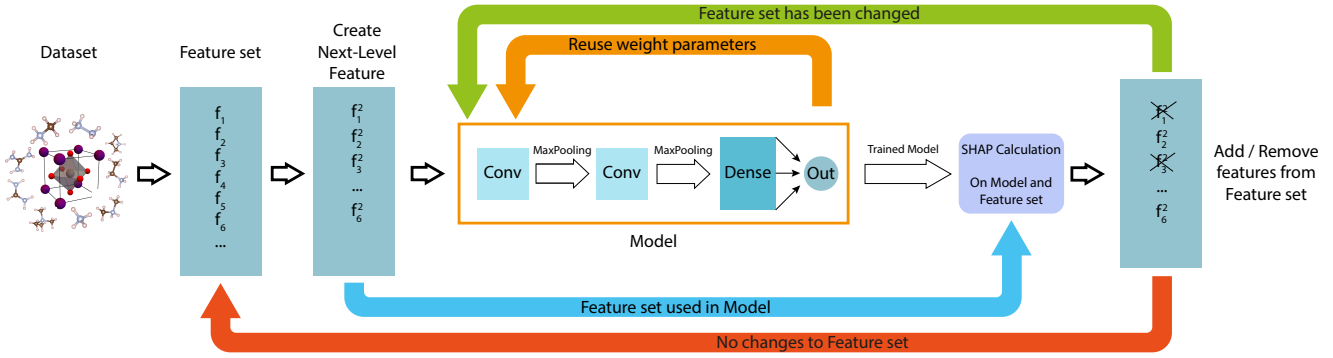

Figure 1: Overview of the Hierarchical training method

3 EXPERIMENT

In this section, we show the benefits of combining features into higher level features, reducing the feature space, by removing less important features and the performance benefits of the Weight Parameter Saving method. These experiments are run on the smaller feature set from [9] to keep it simple and allow for comparison with their results. Lastly, we also test the performance of our method on the full feature set and find the importance of each feature to compare with the 12 features used in the small feature set.

3.1 Experimental Settings

We conduct experiments using two feature sets, which each consists of 862 Organometal Halide Perovskites (OMHP) compounds. We use a smaller feature set (hence referred to as the small feature set) with 12 features as used in [9], and a larger feature set (hence referred to as the full feature set) with 39 features from which the 12 features were hand-picked as being more useful for predicting the eV band gap of the compound.

The general formula for OMHP is $\mathbf{ABX}_3$, where **A** is the organic molecule, **B** a cation (Pb or Sn), **X** an anion (Cl, Br, I, or their combinations. To cover this, we replace $\mathbf{X}_3$ with **CDE**). The feature set is divided in elemental, precursor, and perovskite features. The elemental features comprise the 5 first and second (except for the A-cation which has +1 charge) ionization potentials and electron affinities (**I?**, **IE?**, **E?**, **EPB2?**, respectively. A **?** indicates **A**, **B**, **C**, **D**, or **E**) , the elemental radii (**R?**), the Shannon radii (except for the molecule, **RS?**), the dipole of the molecule (**DIP**). The precursor features are formed by the bandgap of the **AX** and $\mathbf{BX}_2$ precursors of the OMHP syntesis process (**BG_AX**, **BG_BX2**) as well as their volumes (**V_AX**, **V_BX2**) and formation energy (**FE_AX**, **FE_BX2**). All these features can be collected from materials databases. The calculated features are the perovskites ones and are comprised of the band gap (**bg_tetra**), the optimized lattice parameter (**opt_a**), the octahedron rotation (**OT**), and the tolerance factors (**Tol1, Tol2, Tol3**). The 12 features selected in Saidi's work are **IA**, **EA**, **BG_AX**, **FE_AX**, **RA**, **V_AX**, **opt_a**, **OT**, **RB**, **RC**, **RD** and **RE**.

The model is a convolutional neural network consisting of 5 layers. The first layer is the input layer, followed by a convolutional layer with 64 filters, another convolutional layer with 128 filters, a fully connected layer with 100 neurons and a single neuron output layer. Between each layer, the output is run through a Rectified Linear Unit activation function. Padding is applied to the convolutional layers, such that the output has the same size as input. The convolutional layers also apply maxpooling to downsize the input by a factor of two. The fully connected layer employs a dropout of 0.2 to avoid overfitting. The optimizer used is the Adam Optimizer with a learning rate of $1 * 10^{-4}$. Early stopping has been implemented on the validation set, with a patience of 500 epochs.

3.2 Model Performance on small feature set

As a baseline we use the code received from [9] and obtain a RMSE score of 0.2286. Running our code on the 12 features from the small feature set and Equation 4, we compute 78 unique features that the model trains on and receive a RMSE of 0.2880. This shows that by only keeping unique features the model is still capable of outputting satisfactory scoring. After hierarchical training, the model has 69 unique features, with a RMSE of 0.2236. This shows that removing redundant features helps the network by removing noise for other inputs. With this, we have proven that removal of features can lead to better performance in terms of RMSE. A test on the full feature set with 39 features, is run to find how the model performs in this scenario, getting a RMSE of 0.3067.

3.3 Feature Combination and Importance

Figure 2 shows the heatmap for all second level features in the model and their importance value as a percentage, summing up to 100, and shows how big of an influence a feature has in predicting the output. These importance values are calculated using SHAP values. The column shown as "single_f" represents all the single features by themselves, as the features and the feature combinations are both given as input to the model. Half of the map is empty, as features are not combined with themselves and we only compute unique features, as mentioned in section 2. We see that a feature like **opt_a** is very important for the model, having a 1.86% importance on the output by itself. The most important feature is the combined feature of **RB** and **opt_a** which has a percentage value of 4.06%. This shows that we are able to find the importance of single features as well as their combinations which can lead to a higher importance.

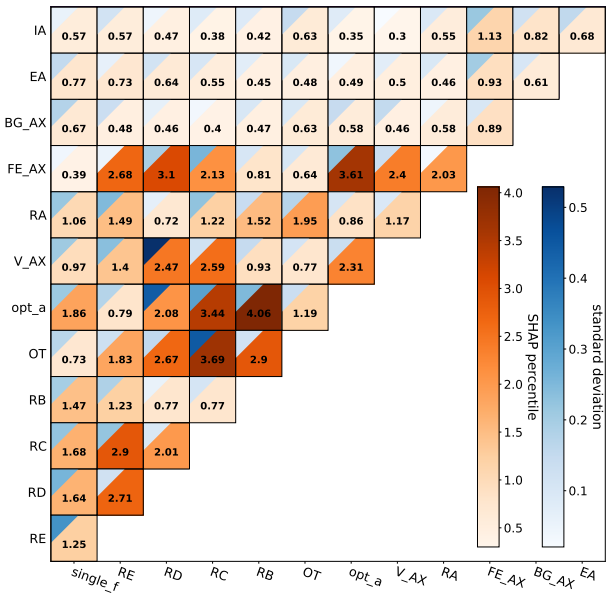

Figure 2: Heatmap showing the run done with all features of the first and second level. Values represent a percentage of importance.

3.4 Benefit of Weight Parameter Saving

For this experiment we have completed two runs on the small feature set, which can be seen on Figure 3.

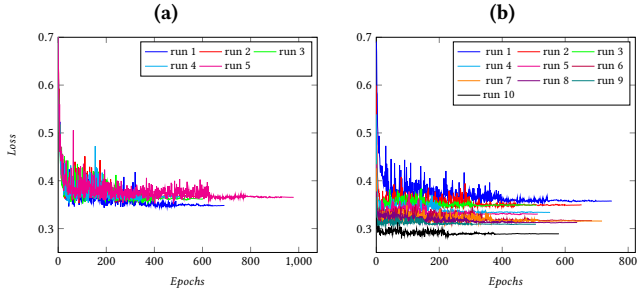

Figure 3: (a) Loss with weight parameter saving turned off. (b) Loss with weight parameter saving turned on.

Looking at Figure 3a, the models loss does not decrease after each training run. In contrast to Figure 3b, where the model retains knowledge from previous runs, we see that the loss decrease in almost every run. This shows, that weight parameter saving has performance benefits, since by retaining knowledge from previous runs we can get a better loss. From a time perspective both Figure 3a and Figure 3b are nearly equivalent with and average of 301 seconds with Weight Parameter Saving off, and 283 seconds with it on.

3.5 Best Features from Full Feature Set

As part of the experiment the method ran on the full feature set with 39 features, to see how important each feature is and how they

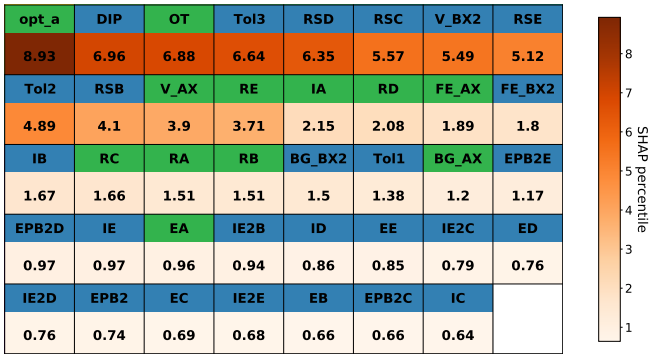

Figure 4: SHAP values for full feature set. Green indicates features used in the small feature set.

differ from the 12 features used in the small feature set. Looking at the results in Figure 4, we find the importance of each feature. Of the 12 best features, only **Opt_a**, **OT**, **V_AX**, **RE** are used in the small feature set, however **IA**, **RD**, **FE_AX** are ranked near the top 12. The features used in [9] are good features, ranging from very important like **Opt_a**, to average importance like **RC**, **RA** and **RB**, with some exceptions of example **BG_AX** and **EA**, which have a low SHAP value. Meanwhile the features **DIP**, **Tol3**, **RSD** which are found to be very important, with a high SHAP value, were not picked as important features for the small feature set.

4 DISCUSSION

In this section, we will discuss the results and ideas used throughout the article. When training the model, a hyper parameter is set, which is set to a margin of 0.1 of the squared error, and decide if features should be removed from or added back into the feature set. This is implemented using the loss of the best performing model in the current run, we check if the newly trained models loss is within a margin of error. If it is, we remove more features, else we add features back. However, the margin of error can be both helpful in reducing the amount of runs needed to run the algorithm, but also a problem in the way that it stops us from exploring performance on even lower amount of features.

The weight parameter saving idea should be useful for any model within the same area of research, as removing less important features have a low impact on the model. However, the adding and removing of features needs to be able to work for several different methods. There is the possibility that a feature's removal has a low or no impact on the model's performance because its information could be saved within the models parameters. This means that reusing the weights does not give a proper removal of a feature.

Currently features are removed with regards to the model with the lowest loss value. But other removal strategies could be an interesting area to research. As an example the method could be changed to instead remove as many features as possible, while keeping the accuracy close to the baseline.

5 CONCLUSION

We propose a method that, given a model can help reduce the size of the input, while retaining the predictive performance. This shows that intepretability can be used to give insight into the model and fine-tune the feature space. Furthermore we propose Weight Parameter Saving during training, which is shown to converge faster towards the minimum, as well as higher level feature combinations, preserving the relations between several features for better predictions.

ACKNOWLEDGMENTS

We would like to thank Kaixuan Chen from Aalborg University for supervising our master thesis, providing general feedback, and helping us with understanding the customs of conferences. We would also like to thank Ivano Eligio Castelli from Technical University of Denmark for assisting us throughout the project, by providing us with all the necessary knowledge about OMHP, assisting in writing the general formula (see 3.1) and providing general feedback. Lastly we would like to thank Ivano and Wissam A. Saidi from the University of Pittsburgh, for providing us with the code used as a baseline.

REFERENCES

[1] Robert Bell, Yehuda Koren, Yahoo Research, and Israel Volinsky. 2008. The BellKor 2008 solution to the Netflix Prize. *AT&T Research* (01 2008).

[2] Ivano E. Castelli, Juan María García-Lastra, Kristian S. Thygesen, and Karsten W. Jacobsen. [n.d.]. Bandgap calculations and trends of organometal halide perovskites. *APL Materials* 2, 8 ([n. d.]), 081514. https://doi.org/10.1063/1.4893495

[3] Ivano E. Castelli, Thomas Olsen, Soumendu Datta, David D. Landis, Søren Dahl, Kristian S. Thygesen, and Karsten W. Jacobsen. [n.d.]. Computational screening of perovskite metal oxides for optimal solar light capture. *Energy Environ. Sci.* 5, 2 ([n. d.]), 5814–5819. https://doi.org/10.1039/c1ee02717d

[4] Stefano Curtarolo, Gus Hart, Marco Buongiorno Nardelli, Natalio Mingo, Stefano Sanvito, and Ohad Levy. 2013. The high-throughput highway to computational materials design. *Nature materials* 12 (03 2013), 191–201. https://doi.org/10.1038/nmat3568

[5] Ioana Giurgiu and Anika Schumann. 2019. Additive Explanations for Anomalies Detected from Multivariate Temporal Data. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (Beijing, China) *(CIKM '19)*. Association for Computing Machinery, New York, NY, USA, 2245–2248. https://doi.org/10.1145/3357384.3358121

[6] Scott Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. 4765–4774.

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. [n.d.]. Human-level control through deep reinforcement learning. *Nature* 518, 7540 ([n. d.]), 529–533. https://doi.org/10.1038/nature14236

[8] Christoph Molnar. 2020. *Interpretable Machine Learning*. Lulu.com. https://www.ebook.de/de/product/39178906/christoph_molnar_interpretable_machine_learning.html

[9] Wissam A. Saidi, Waseem Shadid, and Ivano E. Castelli. [n.d.]. Machine-learning structural and electronic properties of metal halide perovskites using a hierarchical convolutional neural network. *npj Computational Materials* 6, 1 ([n. d.]). https://doi.org/10.1038/s41524-020-0307-8

[10] Kristof T. Schütt, Michael Gastegger, Alexandre Tkatchenko, and Klaus-Robert Müller. 2019. Quantum-chemical insights from interpretable atomistic neural networks. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Lecture Notes in Computer Science, Vol. 11700. Springer, 311–330.

[11] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning Important Features Through Propagating Activation Differences. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 3145–3153. http://proceedings.mlr.press/v70/shrikumar17a.html

[12] Kyle Young, Gareth Booth, Becks Simpson, Reuben Dutton, and Sally Shrapnel. 2019. Deep Neural Network or Dermatologist?. In *Interpretability of Machine Intelligence in Medical Image Computing and Multimodal Learning for Clinical Decision Support - Second International Workshop, iMIMIC 2019, and 9th International Workshop, ML-CDS 2019, Held in Conjunction with MICCAI 2019, Shenzhen, China, October 17, 2019, Proceedings (Lecture Notes in Computer Science, Vol. 11797)*, Kenji Suzuki, Mauricio Reyes, Tanveer Syeda-Mahmood, Ender Konukoglu, Ben Glocker, Roland Wiest, Yaniv Gur, Hayit Greenspan, and Anant Madabhushi (Eds.). Springer, 48–55.