The Journal of Open Source Software

HARDy: Handling Arbitrary Recognition of Data in Python

Maria Politi[1¶], **Abdul Moeez**[*2], **David Beck**[13], **Stuart Adler**[1], **and Lilo Pozzo**[1]

1 University of Washington, Department of Chemical Engineering, Seattle, WA, USA **2** University of Washington, Department of Materials Science and Engineering, Seattle, WA, USA **3** eScience Institute, University of Washington, Seattle, WA, USA ¶ Corresponding author

DOI: 10.21105/joss.03829

Software

- Review
- Repository
- Archive

Editor: George K. Thiruvathukal

Reviewers:

- @andrewtarzia
- @alberto-battistel

Submitted: 26 April 2021
Published: 10 March 2022

`HARDy` is a python-based package that helps evaluate differences in data through feature engineering coupled with kernel methods. The package provides an extension to machine learning by adding layers of feature transformation and representation. The workflow of the package is as follows:

- *Configuration*: Sets attribute for user-defined transformations, machine learning hyper-parameters or hyperparameter space
- *Handling*: Imports pre-labelled data from `.csv` files and loads into the catalogue. Later the data will be split into training and testing sets
- *Arbitrage*: Applies user defined numerical and visual transformations to all the data loaded.
- *Recognition*: Machine Learning module that applies user defined hyperparameter search space for training and evaluation of model
- *Data-Reporting*: Imports result of machine learning models and reports it into dataframes and plots

Statement of Need

High Throughput Experimentation (HTE) and High Throughput Testing (HTT) have exponentially increased the volume of experimental data available to scientists. One of the major bottlenecks in their implementation is the data analysis. The need for autonomous binning and classification has seen an increase in the employment of machine learning approaches in discovery of catalysts, energy materials and process parameters for design of experiment (Becker et al., 2019; Williams et al., 2019). However, these solutions rely on specific sets of hyperparameters for their machine learning models to achieve the desired purpose. Furthermore, numerical data from experimental characterization of materials carries diversity in both features and magnitude. These features are traditionally extracted using deterministic models based on empirical relationships between variables of the process under investigation. As an example, X-ray diffraction (XRD) data is easier to characterize in linear form as compared to small angle X-ray scattering data, which requires transformation of axis to log-log scale.

One of the most widely applied strategy to enhance the performance of machine learning model is Combined Automatic Machine Learning (AutoML) for CASH (Combined Alogrithm Selection and Hyperparameter Optimization) (Hutter et al., 2019). However, these packages are only limited to hyper-parameter tuning and data features remain untouched. To improve the effectiveness of machine learning models, some of the popular feature engineering strategies used for simple numerical data include binning, binarization, normalization, Box-Cox Transformations and Quantile Sketch Array (QSA) [Zheng & Casari (2018)](Nargesian et al., 2017). Moreover, Deep Feature Synthesis has also shown promising results. Here features are generated from

[*]co-first author

relational databases by performing multi-layer mathematical transformation operations (Kanter & Veeramachaneni, 2015).

HARDy presents an infrastructure which aids in the identification of the best combination of numerical and visual transformations to improve data classification through Convolutional Neural Networks (CNN). HARDy exploits the difference between human-readable images of experimental data (i.e. Cartesian representation) and computer-readable plots, which maximizes the data density presented to an algorithm and reduce superfluous information. HARDy uses configuration files, fed to the open-source package Keras-tuner, removing the need for the user to manually generate unique parameters combinations for each neural network model to be investigated.

Description and Use Case

The python-based package HARDy is a modularly structured package which classifies data using 2D convolutional neural networks. A schematic for the package can be found in figure 1.

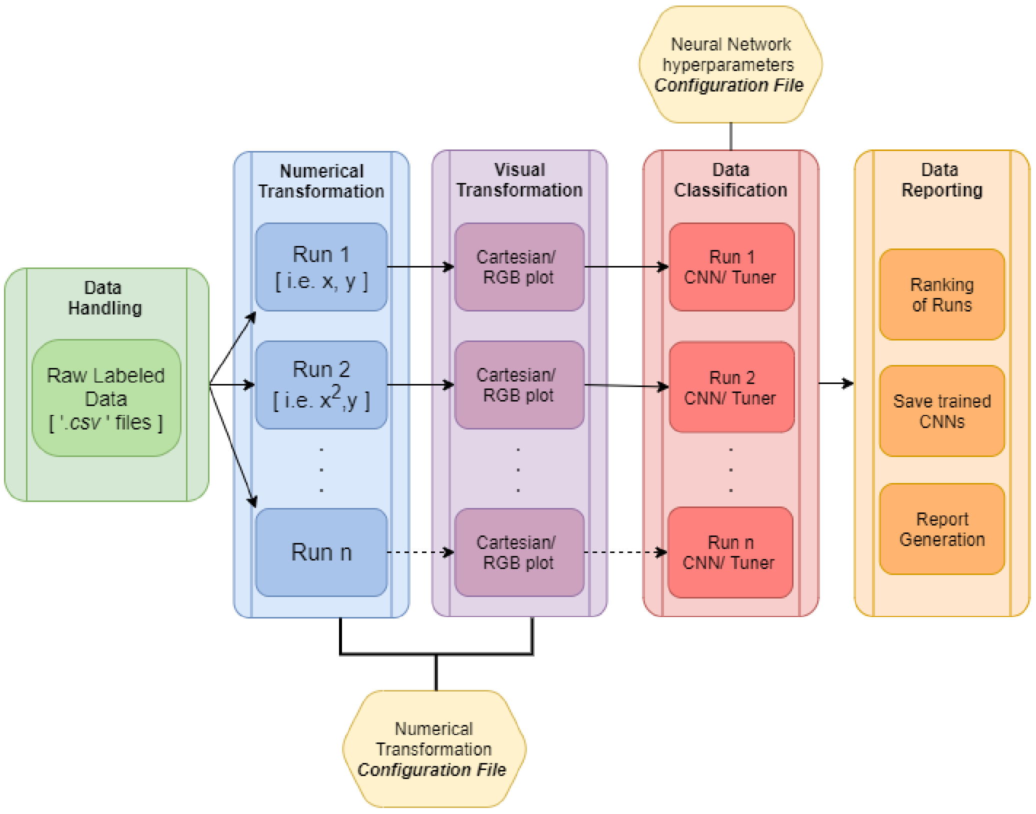
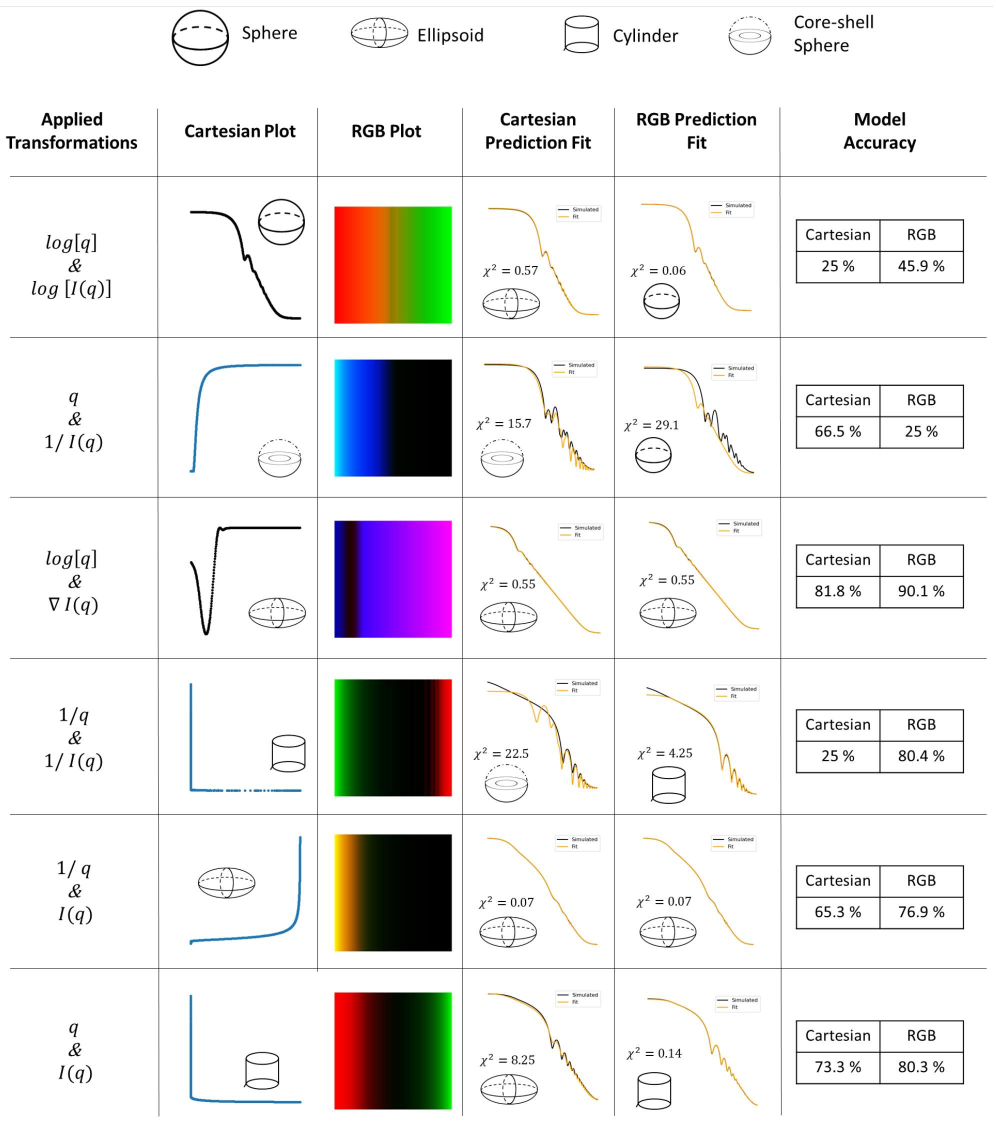


Figure 1: Workflow schematics for HARDy. The data files, left-most column, are subject to numerical and visual transformations, according to the rules outlined in the user defined configuration files. The images are then fed into either a CNN or a tuner, for which the hyperparameter space is controlled through another configuration file. Finally, each transformation produces a report comprising of the best trained model file, the log of training session and the model validation result.

The package was tested on a set of simulated Small Angle Scattering (SAS) data to be classified into four different particle models: spherical, ellipsoidal, cylindrical and core-shell spherical. A total of ten thousand files were generated for each model. The data was generated using *sasmodels*. The geometrical and physical parameters used to obtain each spectrum were taken from a published work discussing a similar classification task (Archibald et al., 2020).

The name of each SAS model was used as label for the data, allowing for further validation of the test set results. These models were selected as they present similar parameters and data features, which at times make it challenging to distinguish between them.



Applied Transformations	Cartesian Plot	RGB Plot	Cartesian Prediction Fit	RGB Prediction Fit	Model Accuracy: Cartesian	Model Accuracy: RGB
$log[q]$ & $log\,[I(q)]$			$\chi^2 = 0.57$	$\chi^2 = 0.06$	25 %	45.9 %
q & $1/\,I(q)$			$\chi^2 = 15.7$	$\chi^2 = 29.1$	66.5 %	25 %
$log[q]$ & $\nabla\,I(q)$			$\chi^2 = 0.55$	$\chi^2 = 0.55$	81.8 %	90.1 %
$1/q$ & $1/\,I(q)$			$\chi^2 = 22.5$	$\chi^2 = 4.25$	25 %	80.4 %
$1/\,q$ & $I(q)$			$\chi^2 = 0.07$	$\chi^2 = 0.07$	65.3 %	76.9 %
q & $I(q)$			$\chi^2 = 8.25$	$\chi^2 = 0.14$	73.3 %	80.3 %

Figure 2: Summary table of few transformations visualized using cartesian coordinate and RGB representation along with the respective fittings. The left-most column shows the transformations applied to the data: the scattering vector *q* and scattering intensity *I(q)*. The final model accuracies are also provided. The original data label corresponds to the icon on the left-most graph, whereas the icon under each fit correspond to the model prediction.

First, the pre-labelled data was loaded. A subset of the files, three thousand files in total, was identified as the testing set. All the ML models initialized in the same code run were validated using the same testing set. A user-provided list of transformations, inputted through a configuration file, was then applied to the data. Different trials can be specified, so that multiple sets of transformations can be investigated. Both Cartesian and RGB plots representations were compared. The latter visualization option was obtained by encoding the data into the pixel values of each channel composing a color image, for a total of six-channels available (i.e., 3 RGB channels in horizontal/vertical orthogonal directions).

The data was then fed into a convolutional neural network, whose hyperparameters and structure were defined using another configuration file. Alternatively, it is also possible to train multiple classifiers for a single transformation trial through the use of a tuner, by instead providing a hyperparameter space and a search method. The classification results, as well as the best performing trained model were saved for each transformation run. The package also allows to visually compare, through parallel coordinates plots (see documentation), the performance of each transformation. Figure 2 shows a summary of few runs comparing the two visualization strategies and their best performing model accuracies.

Comprehensive results for all transformations tested are available in the documentation. It can be noticed that data representation using Cartesian coordinate plots yielded a higher number of instances in which the accuracy of the trained machine learning model was ~25%. This value corresponds to machine learning model's inability to recognize differences in a four-class classification task. On the other hand, the RGB plots show, on average, higher accuracy for the same combinations of numerical transformations. To further validate the results, mathematical fitting was performed on a test set using the SASmodels package. The fitting was based on probabilities determined by the ML model for each label. In scenarios where the output probability was below 70%, the data was also fitted using the second highest possible SAS model.

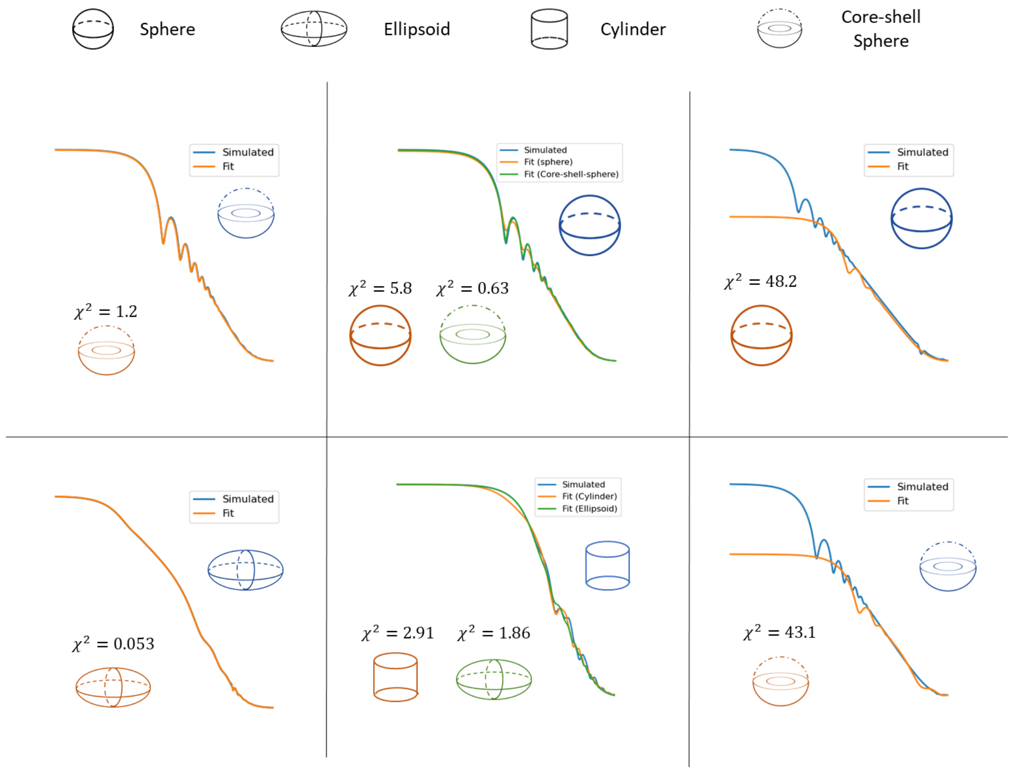


Figure 3: Model fitting examples for various validation files used under the following transformation: *log(q)* & *derivative I(q)*. The model was trained using RGB images and yielded the highest classification accuracy of 90.1%. The bottom labels used correspond to the predicted model that was assigned to the data by the model, whereas the top label correspond to the original model used to generate the data.

The average chi-square parameter of the fitted data was determined to be 7.5. Approximately 11 % of the data had a probability lower than 70%. In all cases,as seen in figure 3, if the neural network was not able to correctly label the data with the highest probability label, the second highest probability label was the correct one.

In conclusion, HARDy can significantly improve data classification so that automatic data fitting and modeling can be executed without human intervention and without compromising on reliability. We also note that data representation for computer-classification tasks may not follow human intuition and/or standard conventions. HARDy serves a key role in the optimization of visual data representations for CNN classification tasks. Finally, the flexibility of HARDy allows for deployment of the task on a supercomputing cluster system, possibly removing the limitations given by the high computational power required to run these ML algorithms. All configuration files and scripts used to run the example presented in this paper can be found in the package documentation.

Acknowledgements

This project was supported by: National Science Foundation through NSF-CBET grant no. 1917340, the Data Intensive Research Enabling Clean Technology (DIRECT) National Science Foundation (NSF) National Research Traineeship (DGE-1633216), the State of Washington through the University of Washington (UW) Clean Energy Institute and the UW eScience Institute. This work was facilitated through the use of advanced computational, storage, and networking infrastructure provided by the Hyak supercomputer system and funded by the STF at the University of Washington.

References

Archibald, R. K., Doucet, M., Johnston, T., Young, S. R., Yang, E., & Heller, W. T. (2020). Classifying and analyzing small-angle scattering data using weighted k nearest neighbors machine learning techniques. *Journal of Applied Crystallography*, *53*(2), 326–334. https://doi.org/10.1107/s1600576720000552

Becker, P., Márquez, J. A., Just, J., Al-Ashouri, A., Hages, C., Hempel, H., Jošt, M., Albrecht, S., Frahm, R., & Unold, T. (2019). Low temperature synthesis of stable γ-CsPbI3 perovskite layers for solar cells obtained by high throughput experimentation. *Advanced Energy Materials*, *9*(22), 1900555. https://doi.org/10.1002/aenm.201900555

Hutter, F., Kotthoff, L., & Vanschoren, J. (2019). *Automated machine learning: Methods, systems, challenges*. Springer Nature.

Kanter, J. M., & Veeramachaneni, K. (2015). Deep feature synthesis: Towards automating data science endeavors. *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 1–10. https://doi.org/10.1109/dsaa.2015.7344858

Nargesian, F., Samulowitz, H., Khurana, U., Khalil, E. B., & Turaga, D. S. (2017). Learning feature engineering for classification. *IJCAI*, 2529–2535. https://doi.org/10.24963/ijcai.2017/352

Williams, T., McCullough, K., & Lauterbach, J. A. (2019). Enabling catalyst discovery through machine learning and high-throughput experimentation. *Chemistry of Materials*, *32*(1), 157–165. https://doi.org/10.1021/acs.chemmater.9b03043.s001

Zheng, A., & Casari, A. (2018). *Feature engineering for machine learning: Principles and techniques for data scientists.* " O'Reilly Media, Inc.".