DeepCAVE: An Interactive Analysis Tool for Automated Machine Learning

René Sass[1]   SASS@TNT.UNI-HANNOVER.DE
Eddie Bergman[2]   BERGMANE@CS.UNI-FREIBURG.DE
André Biedenkapp[2]   BIEDENKA@CS.UNI-FREIBURG.DE
Frank Hutter[2,3]   FH@CS.UNI-FREIBURG.DE
Marius Lindauer[1]   LINDAUER@TNT.UNI-HANNOVER.DE
[1] *Leibniz University Hannover,* [2] *University of Freiburg,* [3] *Bosch Center for Artificial Intelligence*

Abstract

Automated Machine Learning (AutoML) is used more than ever before to support users in determining efficient hyperparameters, neural architectures, or even full machine learning pipelines. However, users tend to mistrust the optimization process and its results due to a lack of transparency, making manual tuning still widespread. We introduce DEEPCAVE, an interactive framework to analyze and monitor state-of-the-art optimization procedures for AutoML easily and ad hoc. By aiming for full and accessible transparency, DEEP-CAVE builds a bridge between users and AutoML and contributes to establishing trust. Our framework's modular and easy-to-extend nature provides users with automatically generated text, tables, and graphic visualizations. We show the value of DEEPCAVE in an exemplary use-case of outlier detection, in which our framework makes it easy to identify problems, compare multiple runs and interpret optimization processes. The package is freely available on GitHub https://github.com/automl/DeepCAVE.

1. Introduction

Experimental design can be tedious and error-prone in practice, particularly if there is little feasible insight into the metric to be optimized and high-dimensional problems with many (hyper-)parameters are tackled. A typical scenario of experimental design is the tuning of machine learning systems, including data processing, various architectures of deep neural networks, and several hyperparameters for all components in the pipeline. Automated Machine Learning (AutoML; see Hutter et al., 2019, for an overview) has alleviated practitioners from this manual task and support users in achieving peak performance of machine learning (ML) systems. However, since the selection of these ML design decisions is automatically made by the optimization process, the transparency decreases, and the question of how and why a particular pipeline or configuration was chosen remains open. The lack of insights in current AutoML systems (Drozdal et al., 2020) goes so far that some users even prefer manual tuning as they believe they can learn more from this process (Hasebrook et al., 2022). Even more importantly, in safety-critical applications like automated driving or the medical domain, transparency and interpretability of both the ML model and the AutoML process are strongly needed to gain trust in the systems.

The brisk development of machine learning (Pedregosa et al., 2022) and AutoML (e.g. Lindauer and Hutter, 2020, for neural architecture search) creates new challenges for interpretation and analysis tools. For example, *multi-objective* optimization (see, e.g., Miettinen, 1998) allows for simultaneously optimizing all possible non-dominated trade-offs of multiple objectives (e.g., minimizing network size while maximizing classification accuracy (Benmeziane et al., 2021)). Further, incorporating multiple budgets into the process (a.k.a. *multi-fidelity* optimization) is used to speed-up optimization by estimating the final performance on lower budgets such as epochs (Falkner et al., 2018; Li et al., 2018) or subsets of the data (Jamieson and Talwalkar, 2016; Klein et al., 2017). Given these multitudes of different approaches and considering that optimization runs may still take several days, depending on the problem, interpretation tools for AutoML need to be flexible, interactive, and ad-hoc.

We introduce DEEPCAVE, successor of *CAVE*[1] (Biedenkapp et al., 2018), an interactive dashboard for analyzing AutoML runs to bring the human back into the loop. Our framework mainly focuses on Hyperparameter Optimization (Bischl et al., 2021, HPO) but can also be used with Neural Architecture Search (Elsken et al., 2019, NAS) and Combined Algorithm and Hyperparameter Selection (Thornton et al., 2013, CASH) by encoding architectural, algorithmic choices or ML pipelines as part of the configuration. DEEPCAVE enables users to interactively explore and analyze outputs of optimizers in the form of texts, tables, and graphics already *while* the optimization process is running. Users can dive *deep*er into their particular interesting topics as our proposed framework is structured in modules (or plugins), all of which complement each other to get an idea of what happened behind the black-box facade of the optimizer.

DEEPCAVE is designed with AutoML for deep learning in mind, a.k.a. AutoDL. First, the real-time monitoring of DEEPCAVE allows users to quickly gain insights even for costly and long-running optimization runs, as a typical trait of AutoDL. Second, pure black-box optimization is too expensive for training several DNNs; hence, multi-fidelity optimization is one of the key approaches to making AutoDL feasible (Zimmer et al., 2021). DEEPCAVE is the first tool to support analysis of multi-fidelity optimization procedures. Last, with the diverse applications of deep learning, several objectives need to be taken into account by AutoDL. Therefore, we believe that DEEPCAVE is well suited to become a go-to tool for deep learning practitioners using AutoML.

In this paper, we explain how certain framework design decisions benefit the users and why DEEPCAVE can be easily used by different user groups, including researchers, data scientists, and machine learning engineers. Finally, we show how our framework can be utilized to answer crucial questions: Have there been any issues in the optimization process? How much of the space is already covered? Are the selected budget steps efficient? Which configuration should be selected if multiple objectives are important? Which hyperparameters and design decisions are the most important ones?

Our contributions are twofold: i) We present a novel framework that enables analysis of AutoML optimization processes while increasing the transparency and bringing the human back in the loop. ii) In an exemplary study on outlier detection, we show how our framework can be used to answer questions typically faced when using AutoML tools.

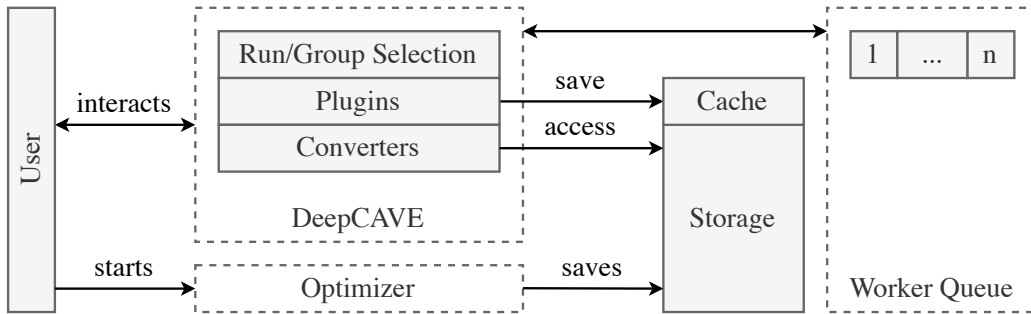1. CAVE stands for Configuration, Evaluation, Visualization and Evaluation

Figure 1: Relation between user, optimizer and DeepCAVE.

2. Framework

We begin by giving an overview of DeepCAVE and describe how a typical workflow looks like. Finally, we explain how certain requirements shaped the implementation.

Overview The interconnection between a user, optimizer, and DeepCAVE is shown in Figure 1. After starting DeepCAVE, the user can select optimizer runs or stagger runs as groups. DeepCAVE accesses the optimizer data on the file system via *converters*, which monitor both finished processes and running processes that regularly write new results to disk, as long as a suitable converter is available. At release, DeepCAVE natively supports well-known AutoML packages such as BOHB (Falkner et al., 2018), DEHB (Awad et al., 2021), SMAC (Lindauer et al., 2022), Auto-Sklearn (Feurer et al., 2022), and Auto-PyTorch (Zimmer et al., 2021), but also provides a Python interface to record the optimizer's output directly. After converting the data, the user can start interpreting the optimization process through the plugins, all of which can incorporate texts, tables, and interactive graphics to show useful information such as simple statistics, hyperparameter importance, or how an optimization run behaves over time. Since some plugins might require more heavy computation, a worker queue ensures responsiveness at all times.

Design Decisions DeepCAVE is implemented on top of Dash (Plotly Technologies Inc., 2015) and supports interactive visualizations natively provided by Plotly. With the ease of extensibility in mind, our framework is written purely in Python as it is the most common programming language in the field of ML (Raschka et al., 2020). Thus, (Auto)ML researchers have a low barrier of entry when designing new plugins for DeepCAVE. Moreover, we structured plugins into input, filter, and output blocks, all of which are rendered by easy-to-understand HTML-like elements. All plugins have access to various base functionalities due to a unified run interface, which is instantiated based on the selected runs or groups by a converter at runtime. We integrated a powerful interface to provide maximal flexibility to all plugins: The run instance holds meta data, configurations $\boldsymbol{\lambda} \in \Lambda$ from the configuration space Λ, objectives $C = \{c_1, \ldots, c_n\}$, budgets $B = \{b_1, \ldots, b_m\}$ and further helper methods. Additionally, the whole optimization history with its K trials $\{\lambda^k, b^k, C(\lambda^k, b^k), \text{status}, \text{additional}\}_k^K$ is saved, in which each trial consist of a configuration $\boldsymbol{\lambda}^k \in \Lambda$, a budget $b^k \in B$, and the obtained objective values $C(\lambda^k, b^k)$. Furthermore, trials are marked with a status (e.g., successful, crashed or timed-out) and equipped with additional information (e.g., traceback).
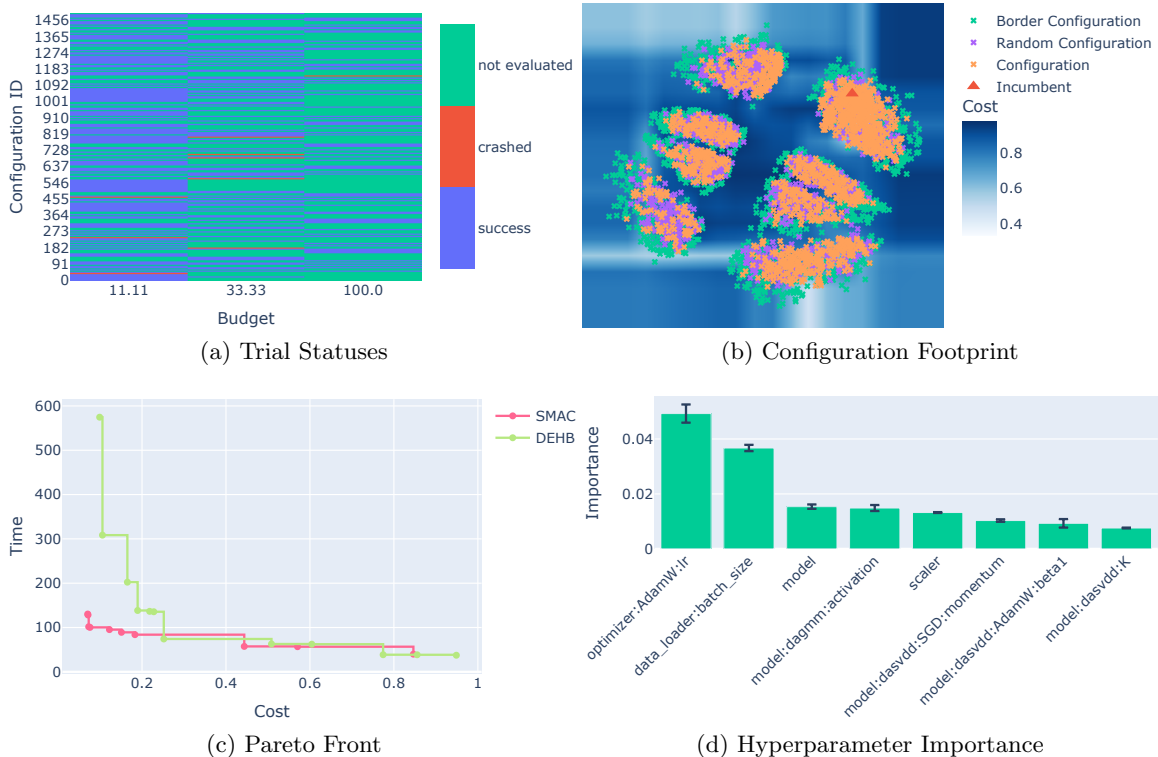
(a) Trial Statuses

(b) Configuration Footprint

(c) Pareto Front

(d) Hyperparameter Importance

Figure 2: DeepCAVE plots using a group of three SMAC runs on the highest budget.

DeepCAVE's dynamically generated texts, hover tooltips, and integrated documentation enables and assist users in understanding the plugins and interpreting the data efficiently. Finally, the integrated cache ensures the quality of life improvements as repeated requests do not trigger redundant computation with the current state of the run in mind. In particular, hashes of the optimizer's outputs are compared, deciding whether both the cache and run instance are up-to-date or have to be updated.

3. Exemplary Study using DeepCAVE

In the following, we demonstrate DeepCAVE's capabilities and answer questions that typically arise when using AutoML tools. We emphasize that DeepCAVE is a modular system that can potentially answer many more questions than we explore in this example study. We provide example questions for each plugin in the documentation (`https://automl.github.io/DeepCAVE`) to help the user to choose the right plugin for each question.

Experimental Setup   Our exemplary ML-task is outlier-detection on the *pendigits* (Keller et al., 2012) dataset with a contamination ratio of 15%. We use two HPO optimizers SMAC3 (Lindauer et al., 2022) and DEHB (Awad et al., 2021), both of which maximize the area under the precision-recall (AUPR) curve to determine a well-performing DL pipeline. Since the optimizers minimize by default, we refer to *cost* as $1-\text{AUPR}$ in the following. The configuration space consists of 39 hyperparameters, in which the optimizer has to select a

model between AE (Rumelhart et al., 1985), VAE (Kingma, 2013), DASVDD (Hojjati and Armanfard, 2021), and DAGMM (Zong et al., 2018). The optimizers make use of epochs for multi-fidelity optimization with budgets $\{11, 33, 100\}$.

Q1: Have there been any issues in the optimization process?   The question can be answered using the *Overview* plugin. In addition to displaying meta data, the optimizer's budgets and objectives, DeepCAVE dynamically generates a short status report:

> Taking all evaluated trials into account, 96.66% have been successful. The other trials are crashed (3.24%). Moreover, 47.96%/30.11%/21.78% of the configurations were evaluated on budget 11.11/33.33/100.0, respectively.

The status report is further supported by a barplot and a heatmap and is ideal for sanity checks. The heatmap (see Figure 2a) shows the status of trials. In particular, crashed configurations are not evaluated on higher budgets, and only some configurations are performing well enough to invest more computational resources. The plugin offers further details to understand why certain trials have failed. Unsuccessful configurations are listed with their status and traceback, making it straightforward to identify the issue.

Q2: How much of the space is already covered?   The plugin *Configuration Footprint* reveals the structure and coverage of the configuration space. Using the multi-dimensional scaling (MDS), it generates a 2D projection of the configuration space (Biedenkapp et al., 2018; Zöller et al., 2022), see Figure 2b. Both evaluated (orange) and unevaluated configurations are shown in the plot, with the latter further divided into border[2] (green) and random (purple) configurations. Border configurations ensure that the entire space is spanned, whereas the random configurations highlight unexplored areas. The plot thus helps users to see if the optimizer already densely sampled the space or might need more evaluations to achieve better coverage. We note that, in this example, not all parts of the MDS space can be covered because of the hierarchical structure induced by the configuration space.

Q3: Are the selected budget steps efficient?   For multi-fidelity optimization, it is desirable to use budgets such that the performance across budgets is strongly correlated since multi-fidelity allows an optimizer to discard poorly performing configurations on the lower budgets quickly. In the plugin *Budget Correlation*, the correlation between all budget combinations is shown. DeepCAVE tells us directly that all of our budget combinations have a very strong correlation in our example. Hence the chosen budgets are appropriate – not shown here.

Q4: Which configuration should be selected if both cost and time are important?
DeepCAVE can highlight the best configurations with respect to both objectives using the plugin *Pareto Front*. The lines in Figure 2c depict the Pareto front for cost and (training) time. Depending on the application, a user could easily choose a configuration optimized for time, cost, or somewhere in between. This plugin also supports comparison between multiple runs or groups so that, in our case, *SMAC* and *DEHB* can be quickly compared. Hovering over a point reveals the configuration, its objective values on the highest budget, and selected hyperparameters. The points are clickable, taking the user directly to a detailed page with information about its origin, objective values, hyperparameter visualizations, and auto-generated code that can be directly copied to use the configuration in Python.

---

2. A border configuration uses only hyperparameters with min and/or max bounds.

Q5: Which hyperparameters are the most important ones? The plugin *Importances* let us choose between LPI (Biedenkapp et al., 2018) and fANOVA (Hutter et al., 2014). The plugin's output (see fANOVA results in Figure 2d) shows that the learning rate, batch size, and model have the largest impact on performance. However, the plot does not reveal, e.g., which model was used in the best-performing configurations. For this, we can use the *Configuration Footprint* (Figure 2b) again. Having a closer look at the incumbent, i.e., the best performing configuration, and using hover information from its neighbors, we find that *DAGMM* is the best choice on the given dataset. In a further optimization run, we could utilize this information, e.g., to prune the configuration space, focusing only on more promising regions that include the incumbent's neighborhood.

4. Related Work

The design of DeepCAVE is driven by the need to make AutoML more trustworthy by means of increasing transparency. Other tools, such as CAVE (Biedenkapp et al., 2018) for algorithm configuration, IOHanalyzer (Doerr et al., 2020) for discrete optimization and XAutoML (Zöller et al., 2022) for AutoML, strive for similar goals, but have major limitations compared to DeepCAVE. First of all, all three can only be used *after* the optimization process has finished. This restriction prevents users from discovering potentially non-fatal errors while the optimizer is running. Furthermore, DeepCAVE is the only tool that is designed with multi-objective and multi-fidelity optimization in mind.

CAVE only generates static reports, which have several drawbacks, e.g., it is nigh on impossible to explore a specific problem deeper. Since *CAVE* is designed for experts only, it is challenging for inexperienced users to understand the given information. Our novel framework mitigates these shortcomings by providing an interactive interface, intuitive visualization, and online documentation directly available where users need it, i.e., when making choices and when results are presented. Thus, we view DeepCAVE as a tool that is both accessible and useful for AutoML users of all experience levels.

XAutoML focuses on analysis for classification tasks and does not support multi-fidelity or multi-objective optimization. DeepCAVE mitigates these shortcomings by being optimizer agnostic, enabling the use of multi-fidelity and multi-objective optimization, thereby avoiding a strong focus on specific target applications. Although DeepCAVE is built for AutoML, in principle, it can be used to analyze any black-box function optimizer.

5. Conclusion

We introduced DeepCAVE, an interactive framework to analyze AutoML runs in real-time, and discussed in detail how our design decisions make our framework easy to use with *any* multi-fidelity AutoML optimizer as well as its extensibility through design. Based on an exemplary study of outlier detection, we demonstrated how central questions commonly faced by users of AutoML tools could be answered. This study further shows how our tool improves the transparency of AutoML and how the application aids the user's understanding of the optimization process in-depth. Finally, we believe that our interactive tool DeepCAVE and its diverse collection of default plugins bring us closer to human-centered AI and will help to increase the trustworthiness of AutoML tools.

Acknowledgement

Funded by the European Union

References

N. Awad, N. Mallik, and F. Hutter. DEHB: Evolutionary hyberband for scalable, robust and efficient hyperparameter optimization. In Z. Zhou, editor, *Proceedings of the 30th International Joint Conference on Artificial Intelligence, IJCAI'21*, pages 2147–2153. ijcai.org, 2021.

H. Benmeziane, K. El Maghraoui, H. Ouarnoughi, S. Niar, M. Wistuba, and N. Wang. A comprehensive survey on hardware-aware neural architecture search. *arXiv:2101.09336 [cs.LG]*, 2021.

A. Biedenkapp, J. Marben, M. Lindauer, and F. Hutter. CAVE: Configuration assessment, visualization and evaluation. In R. Battiti, M. Brunato, I. Kotsireas, and P. Pardalos, editors, *Proceedings of the International Conference on Learning and Intelligent Optimization (LION)*, Lecture Notes in Computer Science. Springer, 2018.

B. Bischl, M. Binder, M. Lang, T. Pielok, J. Richter, S. Coors, J. Thomas, T. Ullmann, M. Becker, A. Boulesteix, D. Deng, and M. Lindauer. Hyperparameter optimization: Foundations, algorithms, best practices and open challenges. *arXiv:2107.05847 [stat.ML]*, 2021.

C. Doerr, F. Ye, N. Horesh, H. Wang, O. Shir, and T. Bäck. Benchmarking discrete optimization heuristics with IOHprofiler. *Applied Soft Computing*, 88:106027, 2020.

J. Drozdal, J. D. Weisz, D. Wang, G. Dass, B. Yao, C. Zhao, M. J. Muller, L. Ju, and H. Su. Trust in AutoML: exploring information needs for establishing trust in automated machine learning systems. In F. Paternò, N. Oliver, C. Conati, L. D. Spano, and N. Tintarev, editors, *Proceedings of the 25th International Conference on Intelligent User Interfaces (IUI'20)*, pages 297–307. ACM, 2020.

T. Elsken, J. Metzen, and F. Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

S. Falkner, A. Klein, and F. Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning (ICML'18)*, volume 80, pages 1437–1446. Proceedings of Machine Learning Research, 2018.

M. Feurer, K. Eggensperger, S. Falkner, M. Lindauer, and F. Hutter. Auto-sklearn 2.0: The next generation. *Journal of Machine Learning Research (JMLR)*, 2022. To appear.

N. Hasebrook, F. Morsbach, N. Kannengießer, J. Franke, F. Hutter, and A. Sunyaev. Why do machine learning practitioners still use manual tuning? A qualitative study. *arXiv:2203.01717 [cs.LG]*, 2022.

H. Hojjati and N. Armanfard. DASVDD: deep autoencoding support vector data descriptor for anomaly detection. *arXiv:2106.05410 [cs.LG]*, 2021.

F. Hutter, H. Hoos, and K. Leyton-Brown. An efficient approach for assessing hyperparameter importance. In E. Xing and T. Jebara, editors, *Proceedings of the 31th International Conference on Machine Learning, (ICML'14)*, pages 754–762. Omnipress, 2014.

F. Hutter, L. Kotthoff, and J. Vanschoren, editors. *Automated Machine Learning: Methods, Systems, Challenges.* Springer, 2019. Available for free at http://automl.org/book.

K. Jamieson and A. Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In A. Gretton and C. Robert, editors, *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 51. Proceedings of Machine Learning Research, 2016.

F. Keller, E. Muller, and K. Bohm. Hics: High contrast subspaces for density-based outlier ranking. In *2012 IEEE 28th International Conference on Data Engineering*, pages 1037–1048, 2012.

D. Kingma. Fast gradient-based inference with continuous latent variable models in auxiliary form. *arXiv:1306.0733 [cs.LG]*, 2013.

A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter. Fast Bayesian optimization of machine learning hyperparameters on large datasets. In A. Singh and J. Zhu, editors, *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 54. Proceedings of Machine Learning Research, 2017.

L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018.

M. Lindauer and F. Hutter. Best practices for scientific research on neural architecture search. *Journal of Machine Learning Research*, 21(243):1–18, 2020.

M. Lindauer, K. Eggensperger, M. Feurer, A. Biedenkapp, D. Deng, C. Benjamins, T. Ruhkopf, R. Sass, and F. Hutter. Smac3: A versatile bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research (JMLR) – MLOSS*, 23(54):1–9, 2022.

K. Miettinen. *Nonlinear multiobjective optimization*, volume 12 of *International series in operations research and management science*. Kluwer, 1998.

F. Pedregosa, T. Maharaj, A. Kucukelbir, R. Das, V. Borghesani, F. Bach, D. Blei, and B. Schölkopf. Retrospectives from 20 years of JMLR, 2022. URL `https://jmlr.org/news/2022/retrospectives.html`.

Plotly Technologies Inc. Collaborative data science, 2015. URL `https://plot.ly`.

S. Raschka, J. Patterson, and C. Nolet. Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. *Information*, 11:193, 2020.

D. Rumelhart, G. Hinton, and R. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing*, volume 1, chapter 8, pages 318–362. MIT Press, 1985.

C. Thornton, F. Hutter, H. Hoos, and K. Leyton-Brown. Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In I. Dhillon, Y. Koren, R. Ghani, T. Senator, P. Bradley, R. Parekh, J. He, R. Grossman, and R. Uthurusamy, editors, *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'13)*, pages 847–855. ACM Press, 2013.

L. Zimmer, M. Lindauer, and F. Hutter. Auto-PyTorch Tabular: Multi-Fidelity MetaLearning for Efficient and Robust AutoDL. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 3079 – 3090, 2021.

M. Zöller, W. Titov, T. Schlegel, and M. F. Huber. XAutoML: A Visual Analytics Tool for Establishing Trust in Automated Machine Learning. *arXiv:2202.11954 [cs.LG]*, 2022. URL `https://arxiv.org/abs/2202.11954`.

B. Zong, Q. Song, M. Renqiang Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *Proceedings of the International Conference on Learning Representations (ICLR'18)*, 2018. Published online: `iclr.cc`.