THURSDAY: A Web Platform to Support AutoML

CHASE CARTHEN, CHRISTOPHER LEWIS, VINH LE, ALIREZA TAVAKKOLI, FREDERICK C. HARRIS, JR., and SERGIU DASCALU, University of Nevada, Reno, USA

THURSDAY is a web platform that aids users in building machine learning models by providing easily accessible tools to either create models manually, or through the use of automated machine learning (AutoML) libraries like AutoKeras. As part of THURSDAY's key innovations, users are given the opportunity to configure and run multiple machine learning models. The results of these model executions can then be compared with built-in performance metrics. Finally, THURSDAY allows users to analyze hyperparameter changes, as well as the changes created by AutoML libraries, in order to provide a vital tool that aids in the revision of existing models. To meet the high volume demands of machine learning, THURDAY adopted a microservice-based design pattern that supports containerization, orchestration, and scalability. In this paper, the design, implementation, and impact of the THURSDAY system is explored in detail. In order to evaluate the capability of THURSDAY, its core functionality is compared against similar platforms that provide machine learning support.

CCS Concepts: • **Human-centered computing** → User interface toolkits; *Information visualization*; *Visualization toolkits*; • **Software and its engineering** → Formal software verification.

Additional Key Words and Phrases: automl, visualization, deep learning, human computer interaction, neural networks

ACM Reference Format:
Chase Carthen, Christopher Lewis, Vinh Le, Alireza Tavakkoli, Frederick C. Harris, Jr., and Sergiu Dascalu. 2022. THURSDAY: A Web Platform to Support AutoML. 1, 1 (April 2022), 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The inclusion of machine learning (ML) in research projects often presents itself as a steep risk for new researchers, due to the sheer amount of time it requires to learn and configure a workflow. Finding the optimal ML model and the parameters associated to that model can steal away a significant amount of time, even for professionals and those savvy with ML. Researchers employing ML in their workflow may soon find that they spend more time on learning ML and contending with parameter tuning than the actual research of their domain.

Interestingly, many researchers have identified this gap between learning/configuring ML and utilizing it in their domain research. It is through this need that the sub-field of Automated Machine Learning (AutoML) emerged. AutoML's express purpose is to bridge this research gap by developing tools to automate the process of searching for optimal models and identify why the tuning of certain parameters produce more effective results. THURSDAY is designed with this purpose in mind and operates as a means to integrate AutoML tools into a usable platform with powerful tools and services to cater to researchers seeking a potential ML component in their domain research.

THURSDAY itself is a platform consisting of several powerful ML tools designed to ease the burden of researchers. One such tool is a customized visualization designed with AutoML in mind. This visualization provides a visual breakdown of differences between ML models generated from AutoML libraries, but although it supports more than one type of model, it is aimed primarily at deep learning models. The visualization presents differences between models, and these differences include metrics such as the loss, accuracy, precision, recall, computation time, and memory usage. Along with comparing models generated from AutoML libraries, users are able to create a machine learning model themselves and compare it against other models. This functionality is included so that the user may explore any form of model that may be of interest. These functionalities require that THURSDAY's software be robust enough that it can scale to accommodate multiple users.

In execution, THURSDAY is a full stack application designed so that it can be placed on clustering infrastructures, like Kubernetes. At the time of this deployment, THURSDAY is deployed and configured with Kubernetes in mind. THURSDAY's various subsystems and component are all designed and implemented with Docker containers as its primary source of containerization. This is done so that a certain degree of scalability can be ensured as development continues. To allow THURSDAY to be used by almost anyone, this scalability is core to the implementation of THURSDAY, as not every user has a powerful enough machine to develop using AutoML or ML libraries. A database is used to keep track of previous models and datasets used by THURSDAY. This paper covers only the use case of THURSDAY with one AutoML library, AutoKeras [12]. The design and implementation of this software is further discussed later in the paper.

The rest of the paper is structured as follows: Section 2 describes a general background of AutoML and the problems that THURSDAY is being designed to solve; Section 3 describes similar work related to THURSDAY and this problem domain; Section 4 covers how THURSDAY is designed and implemented; along with some limitations; Section 5 does a comparison against existing AutoML tools and other behavior when implemented; Section 6 describes the core functionality and unique aspects of THURSDAY; and lastly Section 7 outlines the future directions of THURSDAY and the overall contributions it brings.

Authors' address: Chase Carthen, chase@nevada.unr.edu; Christopher Lewis, christopher_le1@nevada.unr.edu; Vinh Le, vle@unr.edu; Alireza Tavakkoli, tavakkol@unr.edu; Frederick C. Harris, Jr., fred.harris@cse.unr.edu; Sergiu Dascalu, dascalus@cse.unr.edu, University of Nevada, Reno, 1664 North Virginia Street, Reno, USA, 89557.

XXXX-XXXX/2022/4-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 BACKGROUND

Within the AutoML field regarding deep learning, researchers have been working on improving a number of topics. A few topics that we want to highlight are: hyperparameter tuning, meta-learning, neural architecture search, and explainable autoML. Hyperparameter tuning refers to searching for the best set of parameters that would lead to the best machine learning model. An example of a hyperparameter would be the bias of a neuron in an artificial neural network (ANN), or the number of folds to split a dataset into. Typically, hyperparameters are static in nature and must be specified by the developer of the model.

The goal behind meta-learning is for a machine learning model to retain what it has learned from one problem. Then, when the machine learning model is trained for another problem, the retained information from the past plays an active part of the new solution. In more layman terms, meta-learning's goal is for a machine learning model to be able to apply past knowledge to new problems, and continuing this process again and again onto future problem(s).

Neural architecture search (NAS) consists of searching for the best neural network that will solve a problem in an automated fashion. Methods of NAS can be broken down into three different properties: search space, search strategy, and performance estimation strategy [6]. Search space encompasses looking for the best architecture for a given model that may constraints such as the number of neural network layers or number of neurons in a neural network. Search strategy is how the search space explored such as random search, breadth first search, and other types of searchers. Performance estimation strategy is how to explore the estimation of a neural network.

Explainable AutoML (xAutoML) aims to make AutoML more explainable and improve its transparency. Major talking points within xAutoML include subjects like hyperparameter importance, automatic ablation studies, visualizing hyperparameter effects and sampling process [8]. Hyperparameter importance refers to finding which hyperparameters are important globally. Automatic abalation studies refers to finding what configuration of an AutoML tool was important after changes. Visualizing hyperparameters effects and sampling process refers to visualizing the results caused by altering the hyperparameters and the sampling process. In THURSDAY's case, this system would categorize as an implementation of the xAutoML methodology, as it's main purpose is to assist a user in interpreting the changes made by AutoML libraries.

Typically, the pipeline of AutoML can be broken down as: data preparation, feature engineering, model generation, and model evaluation [10]. The steps for the pipeline of AutoML are shown visually in Figure 1. Data preparation characterizes how data has to be changed, details added, or even corrected in order for a machine learning model to use the data. In order for a machine learning model to learn a specific solution, it often requires some feature within a dataset that can be designed by the user or discovered by the model in the case of deep learning. This process is known as feature engineering. In order for a task within machine learning to be achieved, a model must be built. Model generation is often done by hand and the model is chosen based on the type of problem that is presented. The goal of AutoML is to automate this task of building the model's architecture in model generation. Training these models can take time and would be inefficient if evaluated this way. There are techniques for estimating how well the model performs without fully training the model and this process is known as model evaluation. The combination of model generation and model evaluation encompasses encompasses NAS.

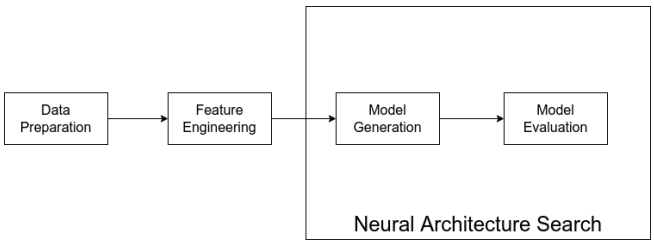


Fig. 1. The steps of building a machine learning model and the overall pipeline of AutoML [10] from start to finish. Neural architecture search encompasses both model generation and model evaluation.

In order to know if an AutoML framework or library is producing an effective machine learning model, a set of metrics are needed to validate whether a model is better performing than another model. These metrics are dependent on the type of output the model has, which could be classification or regression. These metrics can include mean squared error, training loss, accuracy, precision, and recall [5]. In practice, metrics for validation determine whether the changes to the data or machine learning models produce better performance in an AutoML library. THURSDAY visualizes these types of metrics dynamically based on the AutoML library that it interfaces with and what metrics that library displays natively. The visualization of these metrics, with a representation of the machine learning models, can help inform a user of what produces a better result for a given problem.

3 RELATED WORK

While it remains a relatively new field, there are existing libraries for AutoML, including: Auto-WEKA [13], Auto-Sklearn [7], Auto-PyTorch [21], and Auto-Keras [12]. Both Auto-PyTorch and Auto-Keras support deep learning and have been built specifically to do AutoML with Neural Architecture Search and Hyperparameter Tuning. Auto-sklearn and Auto-WEKA supports algorithm selection in terms of machine learning and Hyperparameter Tuning. Researchers have also been working on getting AutoML to run in a scalable environment such as Katib in Kubernetes [20]. Even major tech companies, like Microsoft, Google, and Amazon have started to support AutoML through various tools [1]. For the sake of evaluating THURSDAY's capabilities, the tools that evaluated here in depth are Katib, CAVE, and ATMSeer [3, 19, 20].

Katib is a platform and system that runs inside Kubernetes [20]. It is specialized for doing hyperparameter and NAS inside Kubernetes. Users can specify the machine learning model and AutoML workflow they want to run. While Katib is highly scalable and has a database for keeping track of models and metrics, it does not keep track of multiple users like ATMSeer and THURSDAY. Katib implemented its own functionality for doing AutoML unlike THURSDAY that is

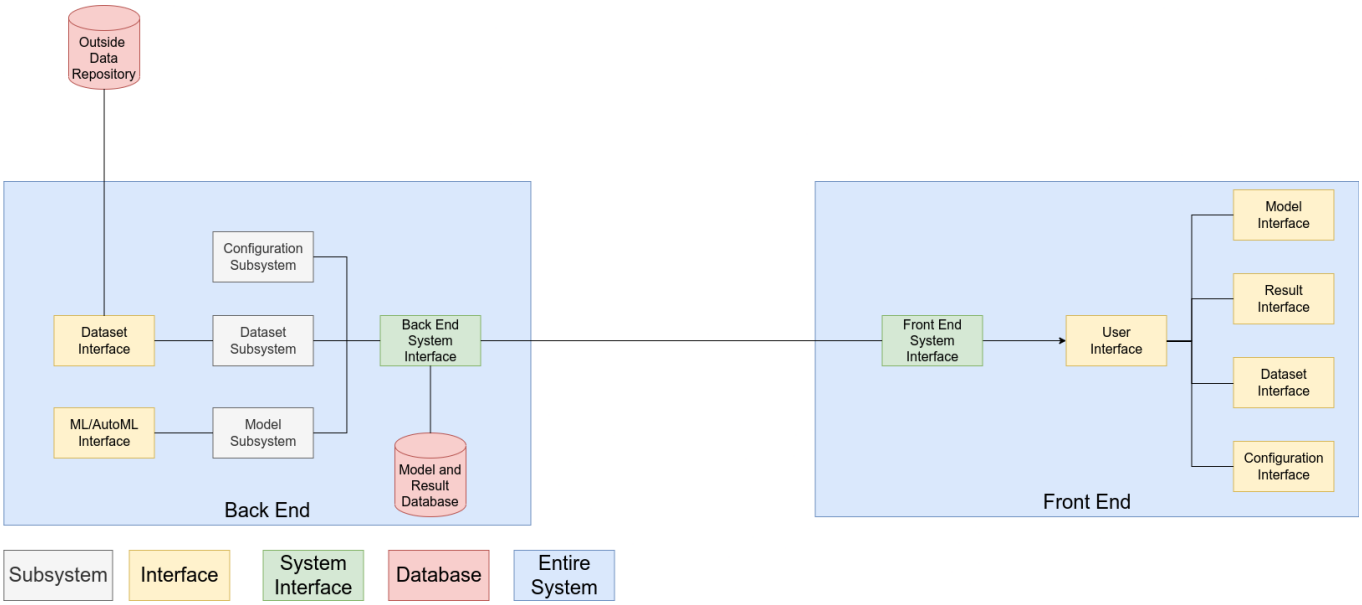


Fig. 2. The high level diagram of THURSDAY.

relying upon AutoML libraries like Auto-Keras. Katib is similar to THURSDAY in that they are both microservice oriented and both have functionality for performing NAS and hyperparameter tuning.

Configuration Assessment, Visualization and Evaluation, or CAVE [3], was created with the express goal of helping researchers understand how different configurations of an algorithm impact the results of the solution, but in an automated fashion. This tool fits within the AutoML field, because hyperparameters are a large part of the configuration process in machine learning. The authors of CAVE, in their paper, focused on showcasing it with an SAT solver. This tool is very similar to THURSDAY in that it does performance analysis, feature importance, parameter importance, and configurator importance. Where THURSDAY differs in comparison to CAVE, is that THURSDAY supports and even encourages deep learning models and is designed as a portal for multiple users.

ATMSeer [19] is a visualization tool built on top of Auto Tune Models (ATM), an AutoML System built for multiple users [18]. ATM is novel because it supports multiple users, as most AutoML systems operate as standalone applications. ATM uses a MySQL database and Amazon S3 to contain information about models, datasets, configurations, and output from runs. ATMSeer is able to interface with ATM's various functionalities, like being able to pause a model being trained and look at different metrics. THURSDAY relates to both in that it uses an SQL database to store models and are user-facing portals. However, THURSDAY differs in that it is more focused towards the deep learning side of machine learning, and it uses a suite of existing AutoML libraries, unlike ATMSeer and ATM that implement their own code base.

THURSDAY is very similar to ATMSeer, in that it is built for multiple users and visualize results from training models. CAVE is comparable to THURSDAY, but isn't built as a system for multiple users. Katib is a general AutoML platform built on Kubernetes, that allows users to run models in a generic fashion with yaml declaration, but isn't built specifically for helping find what changes made by AutoML are particularly relevant. THURSDAY is built specifically to be scalable and for multiple users, like ATM and Katib, while also helping users find what changes made by AutoML are relevant, like CAVE and ATMSeer.

4 IMPLEMENTATION

4.1 Requirements and Use Cases

THURSDAY's requirements were generated from use cases. Requirements and use cases were all created based on two main factors: usability and visualization. Requirements and use cases involving usability allow the user to do some action inside of the system. Requirements and use cases involving visualization allow the user to see data THURSDAY generates, or actions that THURSDAY is currently executing.

THURSDAY had the usability requirements that allowed the user to: load tensorflow and labeled datasets; limit the number of layers and size of the models; allow the model parameters to be configurable; allow the user to set the number of epochs, validation techniques, and number of layers; allow an in-training model to be stopped or started; save and load a model, and the code associate with it; specify a time and epoch limit for run duration; allow an AutoML library to be used and configured; and store, display, and evaluate past models.

THURSDAY had the visual requirements that allowed the user to visualize: different colored graphs for each model; a model's epoch, accuracy, and what validation techniques were used on it; a model's running accuracy and error rate; the current status of the model (running, running duration, finished, and error status messages); and previous models and their evaluations.

Figure 2 demonstrates THURSDAY's system at a high level. The back end consists of several subsystems that handles the configuration, datasets, and models brought into the system. A ML/AutoML interface handles connecting to different machine libraries associated to THURSDAY in the Back End. A dataset interface handles fetching external datasets for THURSDAY. THURSDAY's frontend consists of a user interface that uses a model interface, result interface, dataset interface, and a configuration interface. These interfaces on the frontend allow the system to take in a user's specifications and make changes in the backend. The implementation of these requirements and ideas behind this figure required the front-end and back-end of THURSDAY to run in concert with one another and update when data is generated.

4.2 Back End

THURSDAY's back end currently uses AutoKeras [12] as the AutoML framework that is being used to compare ML models. THURSDAY uses a Postgresql [17] database to store both the ML model being worked with and results from training a model. Flask [15] is implemented as a connection between the front end, the database, and the AutoML framework. Each object in the back end is also containerized using Docker [11]. This includes the AutoML framework functions, the Postgresql database, and the Flask code that opens up end points for a web interface to access. These containers are then orchestrated with Docker-Compose [11], though Kubernetes [2] is being considered as a replacement when doing future work for THURSDAY. Using this system, THURSDAY is able to expose the back end to the front end in a controlled way that also allows further scalability in the future. Another result of containerizing THURSDAY is that, once hosted, it allows multiple users to use THURSDAY independently from one another. Once this hosting takes place, users won't need to use their own machine for machine learning.

4.3 Front End

THURSDAY's front end uses Angular [9] to connect to the back end's exposed endpoints, created in Flask. The UI is created using Angular [9], Bootstrap [4], and Plotly.js [16]. Figure 3 shows the evaluation tab from the front end UI. This figure also shows a graph that is made in Plotly.js off of selected results from the back end. A more detailed example can be seen in Figure 4. Figure 3 also shows bootstrap in the form of navigation tabs at the top bar of the screen, dropdown menus for data selection, and a button for refreshing the graph. The front end is also containerized using Docker. When all of the containers are orchestrated together using Docker-Compose or Kubernetes, this is called a pod. This pod represents one total application for one single user. Any of these containers in a pod can be swapped out for either updated containers, or to be replaced entirely with whatever the user might want at a time. For example, if we found a better replacement for our Postgresql database, we could create a similar container to the Postgresql database container and then have an automated script pull all of the data out of each Postgresql database for each pod. This data would then need to be put into the new container so that users don't lose data, but the actual swap between the databases would be almost instant after the data was moved in each individual pod.

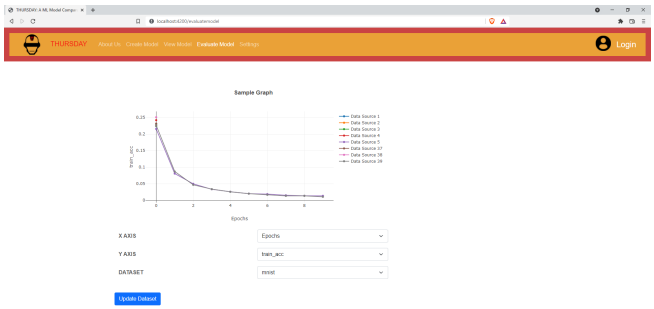
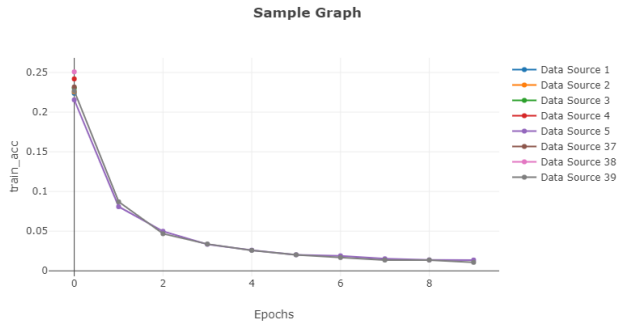
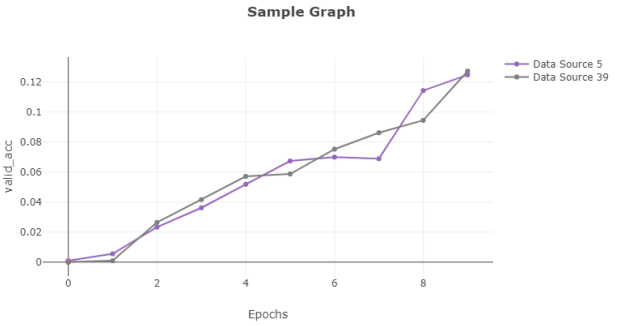


Fig. 3. A graph of the MNIST dataset in THURSDAY's user interface.



(a)



(b)

Fig. 4. For both figures 4(a) and 4(b) a model was ran for 10 epochs with AutoKeras. On figure 4(a) is the training loss and on figure 4(b) is the validation loss for each epoch.

5 EVALUATION

This section describes what features were implemented within THURSDAY and how it currently compares with existing software out there. It also describes what is left to implement into THURSDAY, to make a complete and feature rich application.

Feature	ATMSeer	CAVE	Katib	THURSDAY
Platform/Language	TypeScript	Python	Kubernetes	Docker
Multitenancy	Yes	No	Yes	*Yes
Run AutoML Libraries	No	Yes	Yes	Yes
Fault Tolerant	Yes	No	Yes	*Yes
Metric Storage	Yes	No	Yes	Yes
Metric Collection	Yes	Yes	Yes	Yes
Scalable	Yes	No	Yes	Yes
Configurable	Yes	Yes	Yes	Yes
Visualization	Yes	Yes	No	Yes
Make a Custom Model	Yes	Yes	Yes	*Yes
Uploading data	Yes	No	Yes	*Yes
Deep Learning Models	No	No	Yes	Yes
Compare Libraries	No	No	No	Yes

Table 1. A feature comparison of THURSDAY to ATMSeer, CAVE, Katib, and THURSDAY. * **are planned for implementation.**

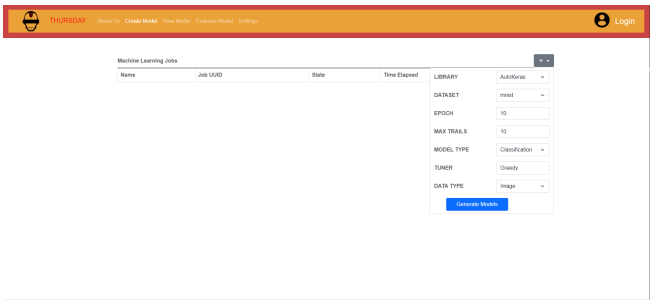


Fig. 5. Example of how model runs can be configured in the user interface of THURSDAY.

5.1 Features Implemented

Core functionality for THURSDAY has already been implemented. These features include:

- The front end and back end THURSDAY are implemented and connected together.
- The front end can display data from the database for previous model runs such as loss, accuracy, precision, recall for both training and validation datasets.
- The back end stores machine learning models and configurations sent down by the user.
- The front end can start and display AutoML jobs on a select set of datasets from TFDS.
- AutoKeras has been integrated with the back end.
- We have dockerized and orchestrated all services for this project into Docker-Compose.

Essentially, we have implemented sufficient functionality for the whole workflow of THURSDAY to work. For example, a user can pick the MNIST dataset with any number of parameters as shown in Figure 5. See the results within some time like those shown in Figure 3. AutoKeras was chosen as it seems to be the easiest library to work with given the documentation made available for it. The datasets provided by TFDS were chosen because Google provides easy access to the datasets, and the API to interface with them is intuitive. The readily available documentation for Angular and Bootstrap made it possible for us to build a good and functional user interface. Flask made it quick and efficient to set-up endpoints to be observed by the front end. Plotly allowed for models runs to be easily visualized within a webbrowser and certainly made it easy to include it in this paper in Figure 4.

With these implemented functionalities, more functions can be implemented with ease. These features include: being able to visualize models stored in the database, using existing models to evaluate against other datasets, allowing users to create custom models, and the ability to upload custom datasets. Adding these new functionalities will allow expanding on different areas that can be explored. For example, if a researcher finds that a particular model generated by an AutoML library produces an interesting result, they can try creating that model in the custom interface and tweak some aspect of it. The plans for the future work are discussed further in Section 7.

5.2 Software Comparison

THURSDAY, as specified in Section 3, is comparable to CAVE, Katib, and ATMSeer in its functionality. A comparison of the functionality with these software packages can be seen in Table 1. The features being compared are as follows: the major platform or language; whether the software has multiple users or multitenancy; can run AutoML libraries; is fault tolerant; can store metrics in some database; can collect metrics; is scalable; is configurable; has visualization for different metrics; users can upload data or models; and has the capability to work with deep learning models. Both ATMSeer and Katib hold most of the functionality that THURSDAY provides. Katib is able to work with many different machine learning libraries and could use an AutoML library like AutoKeras, but it would be counter productive as Katib itself is a system built to do AutoML with existing machine learning libraries. Katib is very similar to ATM and AutoKeras. Also, Katib is not built to be a visualization tool to compare machine learning libraries. ATMSeer, being a visualization tool, allows for the user to create custom models and is built on top of the ATM AutoML system. ATMSeer, however, does

not provide the functionality to compare machine learning libraries against themselves or to run them. ATM was designed as a system that works separately from other AutoML libraries and does not seem to perform generation from deep learning based networks.

6 CONCLUSION

In this paper, we discussed how THURSDAY is implemented and what other tools or systems it is related to. THURSDAY is implemented as a full stack application that has a back end and a front end. The back end is comprised of a database and some web services that can send information from the database and start the generation of machine learning models with AutoML libraries. The front end allows for the user to interface with the back end to do things such as configuring the generation of machine models from an AutoML library, or visualizing results from a previous model with one dataset.

THURSDAY is a unique and modernized example of software engineering due to the inclusion of containerized elements with Docker, orchestrated containers with Docker-Compose, and the inclusion of Flask and Angular for abstracted user functions and front end design. THURSDAY is also very modular due to this containerization, so implementing other AutoML libraries and other datasets will be considerably easier than most other applications that aren't containerized. THURSDAY also has the possibility to scale and be used by many users due to the containerization and orchestration done by Docker and Docker-Compose.

7 FUTURE WORK

The core functionality of this project has been implemented, but there is still a lot that the authors wish to add to make this work truly impactful in the machine learning field. We would like to expand on the number of available AutoML libraries that THURSDAY currently uses. Some AutoML libraries that may be added in the future include Auto-PyTorch, Auto-WEKA, and Auto-Sklearn. Like ATMSeer, we would like to have the functionality for multiple users where users can upload their own models and data. For example we could include a visualizer, similar to Netron [14], so that the user can see what a model looks like, or alternatively, use it to build a model for comparing against other models in the system.

To further improve THURSDAY at a system level we are planning to integrate THURSDAY into a Kubernetes cluster, similar to Katib, and test it with ongoing projects at the University of BLIND REVIEW along with preexisting datasets. Since THURSDAY is already containerized with Docker it will be trivial to integrate with some modifications. At the University of BLIND REVIEW there is a dataset being generated with lidar sensors along a local street, this tool could be used to build a set of classification models for this data. We also want to compare how similar configurations within AutoML libraries perform on existing datasets versus existing known implementations. This is to give a more empirical analysis on AutoML models versus traditionally made models.

ACKNOWLEDGMENT

This material is based in part upon work supported by the National Science Foundation under grant numbers Grant1 and Grant2. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] altexsoft. 2022. Comparing Machine Learning as a Service: Amazon, Microsoft Azure, Google Cloud AI, IBM Watson. https://www.altexsoft.com/blog/datascience/comparing-machine-learning-as-a-service-amazon-microsoft-azure-google-cloud-ai-ibm-watson/

[2] The Kubernetes Authors. 2022. Production-Grade, Container Orchestration. https://kubernetes.io/

[3] André Biedenkapp, Joshua Marben, Marius Lindauer, and Frank Hutter. 2019. CAVE: Configuration Assessment, Visualization and Evaluation. In *Learning and Intelligent Optimization*, Roberto Battiti, Mauro Brunato, Ilias Kotsireas, and Panos M. Pardalos (Eds.). Springer International Publishing, Cham, 115–130.

[4] bootstrap. 2022. Bootstrap: The Most Popular HTML, CSS, and JS library in the world. https://getbootstrap.com/

[5] Sibanjan Das and Umit Mert Cakmak. 2018. *Hands-on Automated Machine Learning: A beginner's Guide to Building Automated Machine Learning Systems using AutoML and python.* Packt Publishing, Livery Place 3 Livery Street Birmingham B3 2PB, UK. https://www.packtpub.com/product/hands-on-automated-machine-learning/9781788629898

[6] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Neural Architecture Search: A Survey. *Journal of Machine Learning Research* 20, 55 (2019), 1–21. http://jmlr.org/papers/v20/18-598.html

[7] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and Robust Automated Machine Learning. In *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Eds.). Curran Associates, Inc., 10010 North Torrey Pines Road, La Jolla, CA 92037, 2962–2970. https://papers.nips.cc/paper/5872-efficient-and-robust-automated-machine-learning.pdf

[8] Freiburg-Hannover. 2022. Hyperparameter optimization. https://www.automl.org/

[9] Google. 2022. Angular: The Modern Web Developer's Platform. https://angular.io/

[10] Xin He, Kaiyong Zhao, and Xiaowen Chu. 2021. AutoML: A survey of the state-of-the-art. *Knowledge-Based Systems* 212 (2021), 106622. https://doi.org/10.1016/j.knosys.2020.106622

[11] Docker inc. 2022. Empowering App Development for Developers. https://www.docker.com/

[12] Haifeng Jin, Qingquan Song, and Xia Hu. 2019. Auto-Keras: An Efficient Neural Architecture Search System. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, Association for Computing Machinery, New York, NY, USA, 1946–1956.

[13] Lars Kotthoff, Chris Thornton, Holger H. Hoos, Frank Hutter, and Kevin Leyton-Brown. 2017. Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA. *Journal of Machine Learning Research* 18, 25 (2017), 1–5. http://jmlr.org/papers/v18/16-261.html

[14] Lutzroeder. 2021. Lutzroeder/netron: Visualizer for Neural Network, Deep Learning, and Machine Learning Models. https://github.com/lutzroeder/netron

[15] Pallets. 2022. Welcome to Flask - Flask Documentation (2.1.x). https://flask.palletsprojects.com/en/2.1.x/

[16] Plotly. 2022. Plotly: The Front End for ML and Data Science Models. https://plotly.com/

[17] PostgreSQL. 2022. PostgreSQL: The World's Most Advanced Open Source Database. https://www.postgresql.org/

[18] Thomas Swearingen, Will Drevo, Bennett Cyphers, Alfredo Cuesta-Infante, Arun Ross, and Kalyan Veeramachaneni. 2017. ATM: A distributed, collaborative, scalable system for automated machine learning. In *2017 IEEE International Conference on Big Data, BigData 2017, Boston, MA, USA, December 11-14, 2017*. IEEE, Boston, MA, USA, 151–162. https://doi.org/10.1109/BigData.2017.8257923

[19] Qianwen Wang, Yao Ming, Zhihua Jin, Qiaomu Shen, Dongyu Liu, Micah J. Smith, Kalyan Veeramachaneni, and Huamin Qu. 2019. *ATMSeer: Increasing Transparency and Controllability in Automated Machine Learning*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300911

[20] Jinan Zhou, Andrey Velichkevich, Kirill Prosvirov, Anubhav Garg, Yuji Oshima, and Debo Dutta. 2019. Katib: A Distributed General AutoML Platform on Kubernetes. In *2019 USENIX Conference on Operational Machine Learning (OpML 19)*. USENIX Association, Santa Clara, CA, 55–57. https://www.usenix.org/conference/opml19/presentation/zhou

[21] Lucas Zimmer, Marius Lindauer, and Frank Hutter. 2021. Auto-Pytorch: Multi-Fidelity MetaLearning for Efficient and Robust AutoDL. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43, 9 (2021), 3079–3090. https://doi.org/10.1109/TPAMI.2021.3067763