2019 5th International Conference for Convergence in Technology (I2CT)
Pune, India. Mar 29-31, 2019

A Review on Automated Machine Learning (AutoML) Systems

Thiloshon Nagarajah
University of Westminster
No 115, New Cavendish Street,
London, UK
+94 (0) 77 4209947
w1608489@my.westminster.ac.uk

Guhanathan Poravi
Informatics Institute of Technology
No 57, Ramakrishna Road,
Colombo 6, Sri Lanka
+94 (0) 77 342330
guhanathan.p@iit.ac.lk

***Abstract*—Automated Machine Learning is a research area which has gained a lot of focus in the recent past. But the various approaches followed by researchers and what has been disclosed by the available work is neither properly documented nor very clear due to the differences in the approaches. If the existing work is analyzed and brought under a common evaluation criterion, it will assist in continuing researches. This paper presents an analysis of the existing work in the domains of autoML, hyperparameter tuning and meta learning. The strongholds and drawbacks of the various approaches and their reviews in terms of algorithms supported, features and the implementations are explored. This paper is a results of the initial phase of an ongoing research, and in the future we hope to make use of this knowledge to create a design that will meet the gaps and the missing links identified.**

Keywords—autoML, hyperparameter, automation, AI

I. Introduction

This section introduces the domain 'Automated Machine Learning' with some background information, the identified problem in the domain and the necessity of addressing the problem.

A. Background

Machine learning is defined as the "field of study that gives computers the ability to learn without being explicitly programmed" [1]. Though machine learning makes it possible for computers to learn without human interaction, human effort is required when the machine learning systems are being built [2]. In low level, machine learning systems are just statistical modelling algorithms or ensembles of algorithms that are developed to work well in a specific use case. A machine learning model designed to identify cats from dogs will not work well on detecting human age from a picture. The statistical model built for one use case, was developed by choosing an algorithm from many possibilities and was tweaked as necessary so that it would work well, but only on that particular dataset. So there is a need to create separate statistical models to each and every dataset we come across making the machine learning process inefficient.

According to a KDNuggets poll conducted in 2015 [3], within next ten years, most of the expert level predictive data science tasks will be automated. This is supported by the fact many domain specific tools (e.g., Google Prediction API, Google CloudML, AzureML, BigML, Dataiku, DataRobot, KNIME, and RapidMiner) have surfaced in the near past to automate some parts of the machine learning pipeline, while many researches are underway to build tools that will completely automate the process. AutoML is a term coined in 2015 by CheLearn [4] to specify these set of tools and researches. AutoML is defined as a "software capable of being trained and tested without human intervention"

B. Problem & Motivation

Designing an effective learning model is often tedious and only done well by experts with deep knowledge of machine learning algorithms and domain expertise. Most others just choose an algorithm by intuition and go ahead with whatever the default values the parameters were assigned by the learning tool they are using. Sometimes they use brute force techniques to find the best performing model, but with infinite possibilities of algorithms and parameters, it becomes highly infeasible [5]. So even though the definition of machine learning makes it seem like there won't be any programming required, in reality it demands large hours of tedious programming to build a learning model.

Machine learning is arguably the center of Artificial Intelligence with many disciplines relying on it [6]. But lack of experts in machine learning makes the development of AI constrained and slow paced. An off-the-shelf solution that can be used to build learning models even by novice developers to any given use case, will be a huge turning point in AI world.

II. Study Setup

We started gaining the necessary domain knowledge with the literature survey. We identified three domains that are linked with automated statistical modelling,

1. AutoML
2. Hyperparameter Tuning
3. Meta-learning

We came across 28 primary researches under these domains and identified different types of approaches used, which is discussed in Section 4. We identified two main such approaches, fully-automated and semi-automated and explored more on these. We identified six main solutions developed by researchers with different pros and cons, as discussed in Section 6.

III. Auto-ML

The umbrella term *AutoML* coined from '**Auto**mated **M**achine **L**earning' [4] refers to the large scale automation of a wide spectrum of the machine learning process beyond the traditional model-creation, such as data pre-processing, meta-learning, feature learning, model searching, hyperparameter optimization, classification/regression, workflows generation, data acquisition and reporting. These black-box learning machines gained popularity after

978-1-5386-8075-9/19/$31.00 ©2019 IEEE

ChaLearn initiated AutoML Competitions [4] in 2015. Started as a 'benchmark for automated machine learning systems that can be operated without any human intervention' the challenge focused on automating hyperparameter tuning and model selection for classification learnings.

Even though there were many promising systems emerged from these competitions and recently we have been introduced to some commercial level AutoML systems by Google [7] and H2O.ai [8], the majority of the concepts and researches are in a very early stages. Researchers have used varieties of statistical theories like regularization, Bayesian priors, Minimum Description Length (MDL), Structural Risk Minimization (SRM), bias/variance tradeoff and genetic programming to tackle the autoML problem but ending up in a very distinct and inconsistent results. Further researches are required to find the best suiting techniques that are generic and works consistently.

Two main concepts in this research area, hyperparameter tuning and meta-learning are analyzed below,

A. *Hyper-parameter tuning*

Machine learning algorithms are in fact statistical functions aimed at minimizing some variant of a cost function. The cost function depends on a set of parameters called hyperparameters that make up the algorithm. For a specific dataset, to get the efficient learning model the parameters need to be set to the optimal value specific to that dataset. Till recent past, hand-tuning these parameters by domain experts have been the only way to find the parameter setting [9]. Since the hyperparameters can take up any value, choosing the optimal one becomes extremely tedious. Recently however, hyperparameter optimization techniques like Regression Trees, Gaussian Processes and density-estimation have been used to automate this tuning process.

The concept of hyperparameter optimization was originated in neural networks as there can be overwhelming number of parameters in a neural network. Later these concepts were needed even in machine learning domains with few hyperparameters, as datasets tend to become too large to hand-tune. Bayesian optimization has emerged as a successful candidate for hyperparameter tuning. It is a probabilistic model that captures relationship between hyperparameter settings and their performance, and uses this model to iteratively evaluate, choose and update the most promising settings. Techniques like random search and grid search [10] are used underneath these approaches.

B. *Meta-learning*

Meta-learning is yet another concept getting popularity in the recent times. Each model trained on a dataset contributes to the understanding of the data. Even if a model performed poorly, that says something about the dataset. These results combined, can create a knowledge system that can be used on similar datasets. This is the concept behind meta-learning. Meta-learning uses attributes like data set size, the number of features, and various aspects about the features along with the performance data. It is used to find good instantiations of learning models from the knowledge of previous tasks [11]. When this data generated by researches all over the world, is stored and continuously updated in a public repository, it creates an invaluable knowledge ecosystem that can help build an effective machine learning workflow.

IV. EXISTING WORK

The existing work on tracking framework evolution and code generation is presented in this section.

There has been substantial interest in researches around autoML systems in the recent past. Thornton [12] was one of the earliest researchers to propose hyperparameter tuning. He used Sequential Model-based Bayesian Optimization to achieve automation in parameter tuning. Several researchers followed his initial work and improved the concept further.

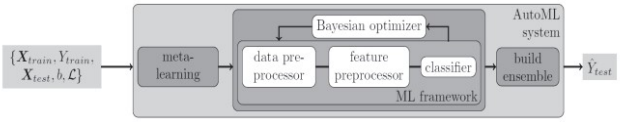

Fig. 1. Typical design of an autoML system

We can categorize possible approaches to track framework evolution, into two main types,

1) Automated approches
2) Semi-automated apporaches

A. *Automated approches*

Automated approaches try to completely automate the machine learning process. Even though it is the ultimate goal of all these researches, a complete automation has proved to be very challenging task even after the current technological advancements. We identified four researches which have achieved varied success in complete automation and analyzed their work.

Auto-WEKA (2012) [12] is the earliest research involved in automating the learning process. They formulated the problem into a formal study area dubbed 'Combined Algorithm Selection and Hyperparameter optimization' (CASH) problem. They came up with a concept to optimize empirical performance by automatically choosing an algorithm from WEKA Java package and its hyperparameters for a given dataset. Auto-WEKA used Bayesian Optimization techniques, and in particular Sequential Model-based Optimization that can work in both categorical and continuous hyperparameters. It iteratively calculates the dependence of cross-validation loss function of a hyperparameter setting, uses machine learning to choose the best candidate configuration of hyperparameters and updates the model with new datapoint obtained. In 2017 ***Auto-WEKA 2.0*** [13] was released improving on the first package by adding support for regression algorithms and parallel runs. It also considered tree-based Bayesian optimization methods as it yielded more promising results than the predecessor. In the comparisons below we consider both *Auto-WEKA* and *Auto-WEKA 2.0* to be the same.

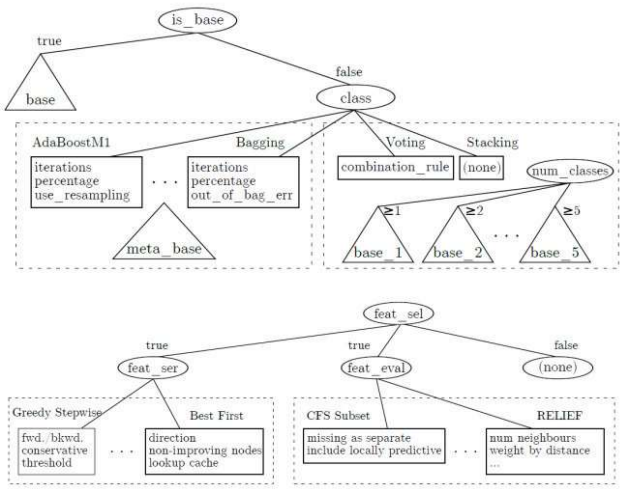

Fig. 2. Auto-WEKA Parameter Space

Hyperopt-Sklearn (2014) [14] was heavily inspired by Auto-WEKA with more focused on bringing similar functionalities to python. It uses the infamous Scikit-learn python package [15] as algorithms base with the underlying concepts being similar to Auto-WEKA. It sets up hyperparameters as a search space of random variables and uses Scikit-learn algorithms to train and validate the model based on the objective function and finally uses hyperopt optimization library to optimize parameters. The objective function is typically cross-validation, the negative degree of success on held-out samples. It also improved the scalability issues found in Auto-WEKA.

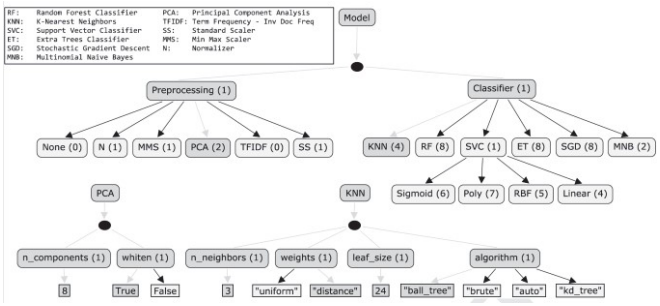

Fig. 3. Hyperopt Search Space

AUTO-SKLEARN (2015) [16] continued on the CASH problem introduced in Auto-WEKA and improved the existing autoML concepts like Bayesian optimization by making use of performance data of the past tries on similar dataset (meta-learning) and by creating ensembles of algorithms without sticking to one single algorithm as model (model ensemble). Meta-learning is done by collecting performance data and set of meta-features for large sets of datasets. The understanding of models and their performance results in initiating the models with the hyperparameters that are likely to suit best, making the automated process more effective and fast. This is used in compliment with Bayesian optimization so that best of both approaches are brought out. The few drawbacks on this system were it lacked support for regression algorithms and worked poorly on large datasets.

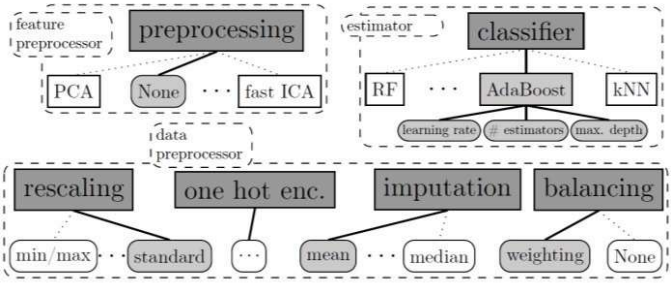

Fig. 4. AUTO-SKLEARN Configuration Space

TPOT (2016) [17] introduced a new approach to the autoML domain by using Genetic Programming (GP). Though focused on just classification algorithms and classification accuracy, it was successful in structuring machine learning process into set of incremental tasks of the entire workflow, making it easier to automate the tasks one by one. TPOT (Tree-based Pipeline Optimization Tool) contains three main tasks / Operators,

1. Feature Preprocessing Operators
2. Feature Selection Operators
3. Supervised Classification Operators

where each of these operators were treated as GP primitives and GP trees were constructed. As a structured task based approach was used, TPOT was very flexible, being able to scale and being able to add or remove more nodes in the pipeline easily. It also used the previous techniques of meta-learning and model ensembles to make it an effective solution.

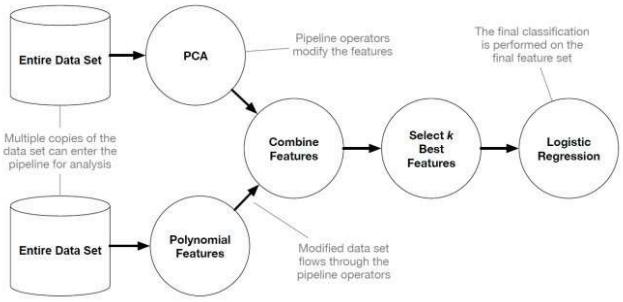

Fig. 5. TPOT Components

B. *Semi-automated approches*

Semi-automated approaches function mainly as an assistant to data scientists in the machine learning tasks and as an initial model builder for them to improve upon, rather than replacing them completely. This is because of the reason the autoML systems are not matured enough to work completely independent but can give a huge boost in performance when worked in coordination with a human involvement.

AutoCompete (2016) [18] was a semi-automated autoML solution developed for a completely specific purpose: to get an initial statistical model in machine learning and data science competitions. Thakur [18] spent two years observing and taking part in various competitions, to build, learn and validate this solution. This approach doesn't completely automate the process but builds the first predictive models for competitors to build on top of. Following the organized setup of TPOT, AutoCompete had following steps in the workflow,

1. Datatype Identifier & Data Splitter
2. Feature Selector & Stacker
3. Model & Hyper-parameter Selector

The system was even capable of automatically identifying machine learning type required (If text dataset is given, deploys Natural Language Processing algorithms), and avoiding over-fitting. On the downside, it was too focused on a specific use case, making it hard to be used as a generic solution and worked only on tabular datasets.

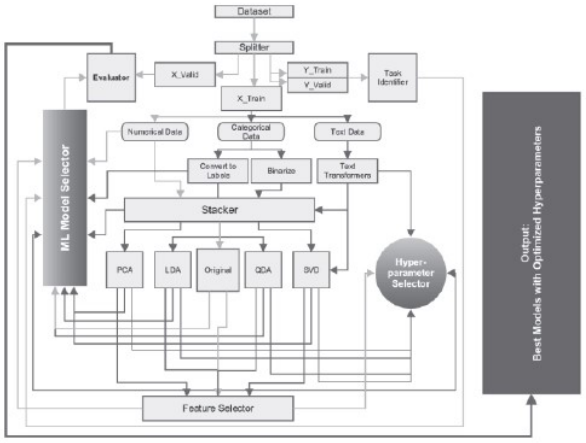

Fig. 6. AutoCompete Components

PennAI (2017) [18] is the first system to have a commercial appeal in the autoML landscape. Because the fully automated techniques are in very early stages, to have a proper product meant the system to not be fully automated. PennAI was introduced as a Learning Assistance that will not replace data scientists, rather help them find best models. While continuing exploration of genetic programming, this was solely focused on healthcare and biomedical domains. PennAI also had a very systematic and defined workflow, with human involvement (dubbed *Human Engine*) being an important part of the workflow. A new feature (dubbed *Knowledge Base*) of storing the models created in previous operations by different users and recommending for new operations was introduced along with a user friendly Graphical User Interface. Though the product had commercial appeal, it supported only a selected few algorithms in scikit-learn package.

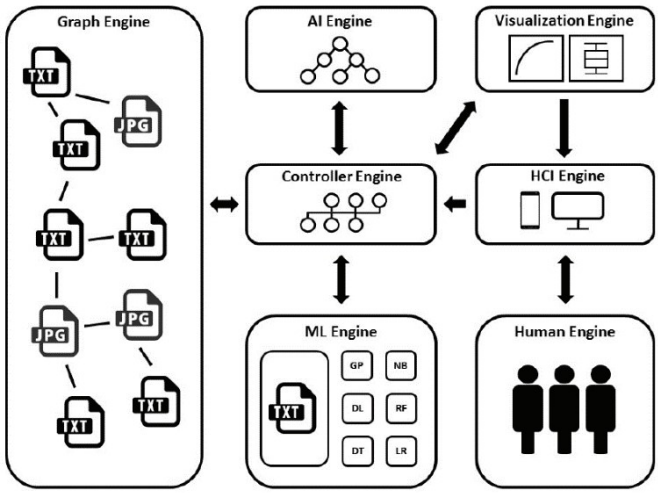

Fig. 7. PennAI Components

V. KEY COMPONENTS REQUIRED

This section covers the key components involved in developing the system regardless of the approaches identified through the literature survey. A typical autoML system will contain 4 important components overseeing 4 important tasks of the automated workflow. But in a more commercial oriented system there can be more than these 4 components. The important components that were identified are as follows,

A. Preprocessing Engine

Preprocessing the dataset is the very first operation done. It is important as the dataset input can be varied and can have many discrepancies. The preprocessing engine takes care of tidying up the data and performing few transformations so that the subsequent parts of the workflow can be run smoothly. Normalization, feature standardization, and missing-value patching are some common preprocessing done regardless of data type. More focused preprocessing can be done with dimensionality reduction, grouping modalities for categorical variables, discretization and nonlinear transformation (e.g. log transformation). There can also be datatype specific preprocessing like punctuation removal in Natural Language Processing related learnings.

B. Feature Engine

The next step is to identify and engineer the features of the dataset. Getting a proper feature set influences hugely on the success of the learning model. Common operations under this engine will be feature extraction, feature selection, dimensionality reduction, linear manifold transformations (e.g. Principle Component Analysis, ICA) and clustering (e.g. K-means) for unsupervised learnings. More specified and customized feature engine operations can be done with embedded feature learning of the algorithms and non-linear dimensionality reductions like KPCA, MDS, LLE and Laplacian Eigenmaps.

C. Predictor Engine

Predictor operation is the most important component of an autoML system. This creates the machine learning model or the predictor function which will be trained and evaluated in the automated process. The ultimate goal of this engine is to find the best candidates of hyperparameters and learning algorithms to be passed to the next engine. In all the previous works a fully functional package was selected to be the underlying layer of the predictor engine. For example, Auto-WEKA uses WEKA, a Java package and AUTO-SKLEARN uses Scikit-Learn, a Python package. Neural nets and Naive Bayes optimizations as predictor models with logistic loss function as predictor function is the most used and successful, so far in the existing systems. Alternatively, researchers have used genetic programming, ensembles of decision trees, linear methods, two-norm/one-norm regularization and nearest neighbors for classification learning as well.

D. Model Selection and Ensemble Engine

Here the best suiting prediction algorithm is chosen from a pool of candidates from *Predictor Engine*. These candidates can be as much as infinite possibilities or can be

refined by processes like meta-learning to a handful best suiting few. For model selection, techniques like cross-validation and leaderboards were used. Particularly in cross-validation, K-folds and leave-one-out and for ensembling, boosting, out-of-bag estimation and other bagging techniques were used. In addition, bi-level optimization and knowledge transfer from one engine to the next were widely used to make an effective model selection.

In a commercial oriented autoML system there can be few other components worth mentioning. For example, PennAI introduced few engines like Human Engine, Knowledge Base, Visualization Engine and GUI Engine. Though these are interesting concepts to make a user friendly autoML system, these doesn't contribute much to the novel research of autoML.

VI. Analysis of Existing Work

In this section we analyze the main six solutions under two categories, 1) The learning algorithms supported and 2) features and characteristics.

A. Analysis of Algorithms

Table 1 presents the comparison of the machine learning algorithms supported in the available work.

TABLE I. Learning Algorithms Supported by Auto-ML Systems

		Automated				Semi-automate		
Algorithm	**# of HP**	**Auto-WEKA**	**Hyperopt-Sklearn**	**AUTO-SKLEARN**	**TPOT**	**Auto-Compete**	**PennAI**	**Total**
Regression Learners								
Gradient Boosting	6					1	1	2
Linear Regression	3	1				1		2
Random Forest	7					1	1	2
Decision Stump	0	1						1
Decision Table	4	1						1
Decision Tree	4						1	1
ElasticNet							1	1
Gaussian Processes	10	1						1
IBk	5	1						1
k-Nearest Neighbors	3						1	1
KStar	3	1						1
Lasso						1		1
Logistic Regression	1					1		1
Multilayer Perceptron	8	1						1
Random Tree	11	1						1
REPTree	6	1						1
Ridge						1		1
SGD	5	1						1
Simple Linear Regression	0	1						1
SMOre	13	1						1
Support Vector Machine	4						1	1
Support Vector Regression						1		1
ZeroR	0	1						1
Classification Learners								
Random Forest	7	1	1	1	1	1	1	6
k-Nearest Neighbors	3		1	1	1	1	1	5
Logistic	1	1			1	1	1	4
Decision Tree	4			1	1		1	3
Gradient Boosting	6			1		1	1	3
Naïve Bayes Multinomial	2	1	1	1				3
Support Vector Machine	4			1		1	1	3
Gaussian Processes	10	1		1				2
kernel SVM	7			1			1	2
Linear Regression	3	1		1				2
Naïve Bayes	2	1				1		2
SGD	5	1	1					2
AdaBoost	4			1				1
BayesNet	2	1						1
Bernoulli Naïve Bayes	2			1				1
Decision Stump	0	1						1
Decision Table	4	1						1
ExtraTrees	8		1					1
extreml Random Trees	5			1				1
IBk	5	1						1
J48	9	1						1
JRip	4	1						1
KStar	3	1						1
LDA	4			1				1
LMT	9	1						1
M5P	4	1						1
M5Rules	4	1						1
Multilayer Perceptron	8	1						1
OneR	1	1						1
PART	4	1						1
passive aggressive	3			1				1
QDA	2			1				1
Random Tree	11	1						1
REPTree	6	1						1
Ridge Classifier						1		1
Simple Linear Regression	0	1						1
Simple Logistic	5	1						1
SMO	11	1						1
SMOreg	13	1						1
SVC	23		1					1
Voted Perceptron	3	1						1
XGBoost					1			1
ZeroR	0	1						1
Total Algorithms Supported	-	**41**	**6**	**15**	**5**	**14**	**13**	-

Here '*HP*' means Hyperparameters, 1 means available and missing values means not available.

B. Features and Behavoiurs

Table 2 presents the comparison of various features and behaviors found in available work.

TABLE II. CHARACTERISTICS OF AUTO-ML SYSTEMS

Feature	*Auto-WEKA*	*Hyperopt-Sklearn*	*AUTO-SKLEARN*	*TPOT*	*AutoCompete*	*PennAI*
Language	Java	Python	Python	Python	Python	Python
Algorithm Source	WEKA	Scikit-learn	Scikit-learn	Scikit-learn	-	Scikit-learn
Predictor Algorithm	Bayesian	Hyperopt	Bayesian	Genetic Programming	Grid-search	Genetic Programming
Ensembles	0	0	1	1	0	0
Meta Learning	0	0	0	1	0	1

VII. WHAT IS MISSING IN EXISTING WORK

In this section we will look into the findings of the survey focusing on what has been missing and the common issues noted in the existing systems.

Functional end products: So far the work in this domain is separated to two aspects of quality - Fully functional but research products and partly functional but proper end products. Since the results from fully automated products show inconsistency, only the semi-automated products have been used largely by public users. A fully automated industry-standard product with user friendly interface is still missing in this domain.

Accessible knowledge hub: Meta-learning techniques in the autoML systems rely on knowledge of previous similar tasks but there aren't any collaborative efforts in creating such libraries of statistical models. Following the initiatives of having open datasets for use, having a library of statistical models will help in advancing the autoML systems.

Python-centric researches: So far except for Auto-WEKA all other researches have been centered around Python. Though this is not a concern, R Language has been in rising popularity in the recent times in terms of data science researches. Having a research in R, a language designed solely for statistical computing will help explore avenues otherwise unexplored.

Using Neural Networks: Deep Neural Networks and Deep Belief Networks have become feasible in the recent times with the increasing computation powers. Tools like TensorFlow have been helping in AI researches and AutoML problem makes a good domain to try such advanced neural technologies.

VIII. CONCLUSION

Analyzing the results from the existing solutions, following conclusions were derived. The research area of autoML is something that formulated very recently and more initiatives are imminent before we get a fully automated industrial standard system. Even though several promising systems are developed, these are centered on a specific domain or use case, and not suitable to be used as a generic solution. By using ensembling and meta-learning the problem of automated hyperparameter tuning can be tackled efficiently. More technologies and statistical concepts unexplored in the autoML systems will make up the majority of future efforts while the knowledge of previous efforts need to be accumulated as knowledge hubs.

With this knowledge about the existing work, drawbacks of the available systems, and how they can be improved, we hope to come up with an architectural style in near future, towards an efficient automated machine learning system.

REFERENCES

[1] P. Simon, "Too Big to Ignore: The Business Case for Big Data," p. 25, 2013.

[2] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kegl, "Algorithms for Hyper-Parameter Optimization," p. 9.

[3] I. Guyon *et al.*, "A brief Review of the ChaLearn AutoML Challenge:," p. 10, 2016.

[4] I. Guyon *et al.*, "Design of the 2015 ChaLearn AutoML challenge," 2015, pp. 1–8.

[5] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential Model-Based Optimization for General Algorithm Configuration," in *Learning and Intelligent Optimization*, vol. 6683, C. A. C. Coello, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 507–523.

[6] R. S. Olson *et al.*, "A System for Accessible Artificial Intelligence," *ArXiv170500594 Cs*, May 2017.

[7] Q. Le and B. Zoph, "Using Machine Learning to Explore Neural Network Architecture," *Using Machine Learning to Explore Neural Network Architecture*, 17-May-2017. .

[8] H2O.ai, "AutoML: Automatic Machine Learning," *AutoML: Automatic Machine Learning*. .

[9] R. Bardenet, M. Brendel, B. Kégl, and M. Sebag, "Collaborative hyperparameter tuning," p. 9.

[10] K. Swersky, J. Snoek, and R. P. Adams, "Freeze-Thaw Bayesian Optimization," *ArXiv14063896 Cs Stat*, Jun. 2014.

[11] M. Feurer, J. T. Springenberg, and F. Hutter, "Initializing Bayesian Hyperparameter Optimization via Meta-Learning," p. 8.

[12] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms," *ArXiv12083719 Cs*, Aug. 2012.

[13] L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, and K. Leyton-Brown, "Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA," p. 5.

[14] B. Komer, J. Bergstra, and C. Eliasmith, "Hyperopt-Sklearn: Automatic Hyperparameter Configuration for Scikit-Learn," p. 7, 2014.

[15] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *ArXiv12010490 Cs*, Jan. 2012.

[16] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and Robust Automated Machine Learning," p. 9.

[17] R. S. Olson, "TPOT: A Tree-based Pipeline Optimization Tool for Automating Machine Learning," p. 9.

[18] A. Thakur and A. Krohn-Grimberghe, "AutoCompete: A Framework for Machine Learning Competition," *ArXiv150702188 Cs Stat*, Jul. 2015.