Survey on Automated End-to-End Data Science?

Djallel Bouneffouf, *IBM Research*, Yorktown, USA
Charu Aggarwal, *IBM Research*, Yorktown, USA
Horst Samulowitz, *IBM Research*, Yorktown, USA
Beat Buesser, *IBM Research*, Dublin, Ireland
Thanh Hoang, *IBM Research*, Dublin, Ireland
Udayan Khurana, *IBM Research*, Yorktown, USA

Sijia Liu, *MIT-IBM*, Cambridge, USA
Tejaswini Pedapati, *IBM Research*, Yorktown, USA
Parikshit Ram, *IBM Research*, Yorktown, USA
Ambrish Rawat, *IBM Research*, Dublin, Ireland
Martin Wistuba, *IBM Research*, Dublin, Ireland
Alexander Gray, *IBM Research*, Yorktown, USA

***Abstract*—Data science is labor-intensive and human experts are scarce but heavily involved in every aspect of it. This makes data science time consuming and restricted to experts with the resulting quality heavily dependent on their experience and skills. To make data science more accessible and scalable, we need its democratization. Automated Data Science (AutoDS) is aimed towards that goal and is emerging as an important research and business topic. We introduce and define the AutoDS challenge, followed by a proposal of a general AutoDS framework that covers existing approaches but also provides guidance for the development of new methods. We categorize and review the existing literature from multiple aspects of the problem setup and employed techniques. Then we provide several views on how AI could succeed in automating end-to-end AutoDS. We hope this survey can serve as insightful guideline for the AutoDS field and provide inspiration for future research.**

***Index Terms*—Machine learning, Data science, Automation**

I. Introduction and Motivation

Data science covers the whole spectrum of data processing, beginning from data integration, distributed architecture, automating machine learning, data visualization, dashboards and BI, data engineering, deployment in production mode, and automated and data-driven decisions (Figure 1). A key part of data science is machine learning in which the system learns from data-driven examples in order to make predictions about examples in which some of the attributes are missing. In supervised learning, this process of modeling uses fully specified examples, referred to as training data, whereas in unsupervised learning, this process is done with incompletely specified examples. Beyond these core areas, many parts of data science such as data acquisition, data integration, and data visualization (which are traditionally not considered machine learning) are essential to the effective deployment of data science and machine learning solutions.

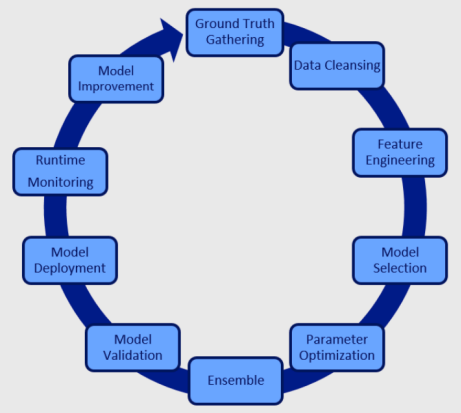

Fig. 1. Steps involved in a typical data science workflow.

Much of data science requires manual intervention in the form of the choice of software (and its set up) and the overall learning pipeline; this makes the use of available resources over diverse settings very challenging. For example, machine learning often requires significant manual exploration of the input data before the data science pipelines are experimented for various tasks. This is a time consuming task, and is often the primary bottleneck in the ad hoc effort required from the analyst. Given a set of machine learning tools and a set of problems, how does one choose what type of resource to deploy in a particular setting? How does one decide how to set its parameters? When training has been performed with a particular data set, how does one decide that the constructed model has now become stale? All these questions often require manual intervention from a user who

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

must experiment with various settings in trial-and-error mode, until decisions on deployment have been made. This type of situation militates against the successful use of wide resources over many settings. A fundamental question in automated data science is whether one can automate large parts of the learning process so that key *meta-decisions* on model choice, (hyper)parameter tuning, and model freshness can be made with minimal user intervention. Indeed, the holy grail of being able to make machines truly intelligent is to automate large portions of machine learning that remain hand crafted to a large extent even today. Even though deep learning has already helped in automating many tasks such as feature engineering (that were earlier hand crafted in traditional machine learning), there remain significant lacunae in what is truly possible in terms of reducing the need for human intervention.

The challenges in automating data science may arise at individual stages of the pipeline, or in the construction of the entire pipeline itself. Furthermore, each stage in the pipeline poses its own set of unique challenges – for example, challenges in the data integration stage are quite different from those in the machine learning stage. We will discuss these challenges in the context of automated data science solutions. Rather than an exhaustive survey, our purpose in this paper is mainly to provide a qualitative overview and editorial outlook for the topic of end-to-end automation of data science. We begin by describing what artificial intelligence (AI) currently automates in the parts of the data science pipeline, then highlight some general approaches that could address the entire process more holistically.

II. What Does AI Automate Today?

Although the goal of end-to-end automated data science is still quite far, many parts of the data science pipeline have been automated with significant success. Here we provide a broad overview of many such cases.

A. Automated Data Collection

Data engineering is the aspect of data science that focuses on the *ground truth gathering* step with the focus on data collection and analysis. For all the questions that data scientists answer using large data sets, there have to be mechanisms for collecting and validating those relevant information. In this context, much of modern hardware technology has already automated data collection; simple transactions today such as phone usage or credit card swipes lead to data collection behind the scenes. The real issue is that there is a *cost* in even using the data collected unless there are ways of parsing out parts of the data relevant and useful for the application at hand. The field of *active learning* [1] provides methods capable of selectively collecting appropriate and relevant data for a particular application. This process integrates data collection and knowledge discovery, and automates large parts of the process to minimize manual effort.

B. Automated Data Integration

The process of integrating data from heterogeneous sources into a single, unified view has been a significant barrier to many data science tasks – a variety of infrastructures generate different data formats with varying levels of contamination in the raw data. This necessitates custom ETL (Extract, Transform, Load) code for *data cleaning and integration*.. This process has been automated with a declarative interface [2] allowing for extendable domain-specific data models; a AI planner performs the integration, optimizing for the plan completion time. This tool is designed for schema-less data querying, code reuse within specific domains, and is robust to messy unstructured data, demonstrated by its capability of integrating data from diverse sources such as web click-stream logs and the census. However, the application of such data cleaning and integration tools have been relatively limited, and there is various open research challenges to wider automation. For example, for any given set of data sources, a truly autonomous system should be able to automatically detect the nature of the data and the ETL steps required for the application at hand.

C. Automated Feature Engineering

The use and effectiveness of different approaches to *feature engineering* is heavily dependent on the nature of the data and the learning task. Problems involving images, videos and texts have seen significant success with deep neural networks where the feature engineering is an integral part of the modeling step, and explicit separate feature engineering has received less attention. In contrast, tabular and relational data rely heavily on explicit feature engineering (automated or otherwise) – it is often the most crucial step in terms of the downstream predictive performance, and consequently being the most time consuming, making it a prime candidate for automation. We categorize the automation schemes based on the type of data they are applied to:

Tabular data: For tabular data, the basic set of features can be extended to transform the data into a feature space that benefits specific models. The set of transformations is usually very large and efficient automation cannot be achieved by exhaustively trying all possible

transformations. One way of reducing the size of the set is by learning the effectiveness of different transformations for a given "type" of prediction problem on data seen in the past by the system [3]. This relies on characterizing the problem (data features, targets, objectives) and effectiveness of each transformation. For any new data, the system suggests transformations based on its learned experience. It does so by learning meta-predictors (which themselves are classification or regression models) on the historical data and statistics. The meta-learning step requires essential preprocessing of the data through quantile sketching which converts data of different shapes and sizes to a canonical format. The prominent advantage of this approach is the low runtime cost, which is due to a handful of inference steps.

Relational data: According to a recent survey among more than 8000 data scientists [4], 65% are frequently working with relational data, making it the most popular data type. However, manual feature engineering with relational data is a tedious task requiring multiple SQL queries in an error-prone trial-and-error fashion until desired results are obtained. Therefore, automated feature engineering for multi-relational data has recently received increasing attention. There are two basic problems: the first involves the choice of appropriate joining paths in entity relation graphs to collect relevant data for a given prediction target, and the second requires choice of the right transformations/aggregations to turn the joined tables into useful features. An early method learned inductive decision trees by propositionalizing relational data (transforming relational data with multiple tables into a single table with features) [5]. The problem has been recently revisited with Deep Feature Synthesis (DFS) [6], but limited to numerical data. The One Button Machine (OneBM) extends support for complex transformations on non-numerical data [7]. Both automated schemes are rule-based where the transformations and joining path choices are predefined based on heuristics. While choosing the right join is a computationally intractable problem, transformations can be learned from relational data using deep neural network [8].

D. Automated Machine Learning

Automated Machine learning (AutoML) has received increasing attention, starting with hyper-parameter optimization (HPO) to determine the most appropriate parameters for a ML model (for example, the number of trees in a random forest), to automating selection of a ML pipeline (such as feature transformation & selection combined with predictive modeling). AutoML automates the *model selection*, *(hyper)parameter optimization*, and even *ensembling* steps of the data science workflow, addressing a wide range of difficult technical challenges ranging from HPO, automated feature engineering, to neural network design. The neural network paradigm has itself been a fillip to the development of automated data science by enabling automated feature engineering to a large extent, and has provided successful end-to-end solutions in specific domains like machine translation [9] which traditionally relied on large amounts of hand-crafted features. Nevertheless, the move to fully automated systems has been incomplete because they continue to require human-centric tuning of large numbers of parameters or design choices. AutoML bridges this additional gap to a large extent; in addition, newer technologies such as reinforcement learning can play a significant role. Auto-WEKA [10] and Auto-sklearn [11] are the main representatives for solving AutoML by so-called sequential parameter optimization. Both apply Bayesian Optimization to find useful ML pipelines. Auto-sklearn improves upon Auto-WEKA by utilizing *meta-learning* [12] and *ensembling*.

E. Visualizations and Decision Making

At the end of the day, the results of machine learning are often fed into visualization and/or decision making tools. This is a particularly tricky part of the process, since the nature of the visualizations and decisions are highly application-specific. This makes automation less feasible. Nevertheless, progress has been made in some partial respects.

The ability to create good visualizations became a must-have skill for all data analysts. In current data visualization tools, users need to know their data well in order to create good visualizations. However, the users need tools to automatically recommend visualizations rather than hand-craft highly customized tools. The authors in [13] propose a system for automatic data visualization that tackles the problem of Visualization recognition where given a visualization, it provides a prediction of whether it is "good" or "bad". This is achieved by training a binary classifier to model the quality of visualization. They also study the Visualization ranking problem, which provides a relative ordering of two given visualizations. This is achieved with a supervised learning-to-rank model, although expert knowledge is also considered with the use of expert rules. Therefore, the approach is not a fully automated system yet, and it uses human criteria in the form of expert rules. Nevertheless, this still provides a modicum

of automation in the drive towards automated visual systems.

Beyond visualization, the final goal of data science is to support decision making. Automated Business Intelligence systems are software applications that utilize automated processes in order to extract actionable organizational knowledge. Authors in [14] proposes an architecture to guide the development of such systems and in so doing outlines a feasible approach by which organizations can adopt them in support of their strategic decision making processes. The goal is to gain a competitive advantage by utilizing information garnered from web sources to inform corporate decision making. Similar to [14], the authors in [15] proposed to automate the extraction, processing, and display of indicators provide useful and current data for operational meetings. The feasibility of extracting specific metrics from information systems was evaluated as part of a longer-term effort to build a business intelligence architecture. Analytics were performed on the data, a process that generated indicators in a dynamic Web-based graphical environment that proved valuable in discussion and root cause analysis. However, this type of decision making is still not fully automated. A key aspect that distinguishes it from truly intelligent (human-like) systems is the trial-and-error process that is endemic to all forms of intelligent decision making. This will be the topic of discussion in the next section.

III. How can AI Automate End-to-End Data Science?

Much of what AI automates today remain as parts of the data science pipeline. However, to be able to work from raw sensory inputs to final decisions is a key challenge that is required for a high level of automation. This forms the basic goal of *end-to-end* data science, as opposed to the automation of individual parts of the data science process, discussed in the previous sections. Here we highlight general frameworks and approaches which offer possible avenues for considering more holistic automation.

A. Reinforcement Learning

The main challenge that arises in building fully automated systems at the level of humans is the fact that the construction of a machine learning system requires a large number of design choices, and the specific *combination* of these design choices can regulate the effectiveness of the system at hand. In many cases, these decisions need to be made *sequentially*. For example, if one is to construct a neural network for a particular task, then the choice of the number of layers naturally precedes the choice of the number of units in each layer. Humans are naturally prone to experimenting with these large numbers of decision choices, and are often able to construct a reasonably accurate system with a relatively modest number of iterations of trial and error. Therefore, creating systems that *automatically perform trial and error, and learn from successes and failures is the key to creating a truly intelligent automated system.* This type of setting is naturally the domain of *reinforcement learning* in which a system can learn from the success and failure of trials [16]. Reinforcement learning is used extensively for video games [17], [18], in which one makes a set of sequential decisions in order to win virtual rewards in the form of game points or victories, and the success of a particular set of decisions can be easily judged. It is noteworthy that many of these solutions work with raw sensory inputs (e.g., pixels) and provide the final decisions as the result, which is a high level of end-to-end automation. This situation applies perfectly to the automated machine learning paradigm where the success of a particular design choice can be easily evaluated.

Reinforcement learning is also used through the bandit framework [19]–[23]. For instance, the authors in [24]–[28] introduce an algorithms that tackles this dilemma in Context-Based Information Retrieval, Context-Based recommender system and in active learning. It is based on dynamic exploration/exploitation and it can adaptively balance the two aspects by deciding which situation is most relevant for exploration or exploitation. Using combinatorial bandit frameworks, authors of [29] tackle the online feature selection problem by addressing the combinatorial optimization problem in the stochastic bandit setting with bandit feedback, utilizing the Thompson Sampling algorithm.

B. Deep Learning

The problem of Automated Deep Learning (ADL) boils down to designing a strategy that given a dataset and a task along with some constraints, yields a well-trained deep learning model that can be used for solving the task. Designing an optimizer that given a search space, looks for an optimal architecture, forms the key component of ADL methods. The current trends in ADL methods explore a variety of approaches to solve this optimization problem, most of which build upon the well established theories in reinforcement learning (RL) and evolutionary algorithms (EA). It should come as no surprise that both of these ideas borrow heavily from the success of the biological paradigm.

As a specific example, consider the case where one wants to learn the best neural architecture for a particular data domain in a problem-specific way. The RL-based optimizers involve learning a controller which is trained to output actions that lead to architecture encodings which correspond to high-performance deep learning models. [30] and [31] were one of the first works to explore RL-based approaches for architecture search. Both these works build networks by sequentially choosing its layers and its set of operations. The validation accuracy of the final trained network is used as a reward to update the controller. [32] take a different approach. Starting from a given architecture, they learn how to modify it in order to improve its performance. In this context, an interesting reinforcement learning system is proposed in [31], which creates an optimal convolutional neural network architecture for image classification. The convolutional neural network whose architecture is to be constructed is considered the *child network*, whereas the neural network that creates the architecture is referred to as the *controller network*. The controller network is a recurrent neural network. The controller network is used to make decisions about the parameters of the child network, and correct decisions are rewarded by higher accuracy of prediction. These types of controller-child combinations are coupled with a boiler-plate reinforcement learning algorithm such as *policy gradients* [33]. The basic idea is that the controller network (together with the reinforcement learning algorithm) experiments with the parameters of the child network and encodes correct decisions within the parameters of the controller network. The controller network is essentially a *policy network* that outputs probabilities of specific decisions about the design of the neural architecture. This broad approach is very similar to how a human experiments with the parameters of a neural network and learns from the experiences obtained with the resulting performance. Other variants of techniques that use reinforcement learning to design neural architectures are discussed in [30]. The main drawback of the reinforcement learning paradigm is that it is computationally expensive, and it can sometimes be difficult to repeatedly train systems with different sets of parameters in order to judge the success of a particular setting.

A key bottleneck in these methods that the evaluation of an architecture requires expensive and repeated training. This problem can be alleviated by reusing parameters of smaller or shallower networks with function-preserving operations [34]. The resulting "warm-start" model requires fewer epochs to train a new network [32], [35]. In another line of work, a large common network is trained, and it is hypothesized that sampling architectures from this large network (either uniformly [36] or according to an optimizer [37]) will yield a useful ranking of the sampled architectures. Some works relax the assumption that the choice of operations has to be discrete and learn a parameterized architecture choice jointly with model parameters [38], [39]. Other works extrapolate the learning curve of a network [40] by predicting whether a training run will improve the current best solution (and terminating the run early).

It is worth noting that high accuracy is often not the only objective of a deep learning model. Often additional practical constraints like number of parameters or inference time need to be incorporated. This task is either solved by aggregating all objective functions to a single one and solving it with standard ADL optimizers [41] or approaches which search for pareto-optimal solutions [42].

Search spaces are a useful component in defining the "'scope" within which an optimizer will look for an optimal architecture. For example, in the convolutional neural network setting, does one use a conventional network like VGG as the "base" model, or does one use a modern skip-connection-based *ResNet* as the base model? The nature of the search space in the latter is more complex. The search spaces have progressively become complex in their definitions over the course of developments in automated deep learning approaches. The cell-based search space, initially proposed by [43], enables an easy transfer of deep learning models across datasets and tasks. The task for the ADL optimizer reduces to select cells which are stacked to derive the final architecture. A cell is built as a combination of blocks, each block has two inputs and two corresponding operations and a combination operator to output a new state. The flexibility of picking input states allows for learning artifacts like parallel convolutions, branches, and skip-connections.

Architecture search is not the only component of deep learning modeling pipeline that has benefited from automation. Additional components like search for heuristics of optimization methods [44], a search for suitable activation functions [45], storing few experiences [46], and automated data augmentation [47] has also been investigated for automation in some of more recent works.

C. Black-Box Optimization

One of the major challenges in designing automated learning system is its *black-box* optimization nature

when explicit expressions of the gradients are difficult to obtain. For example the problems of architecture search and feature engineering do not offer a continuous loss function that can be optimized with gradient descent. The only mode of interaction with the learning system is by submitting inputs and receiving feedback. In AutoDS, the commonly-used black-box optimization methods include Bayesian optimization (BO) [48], zeroth-order (ZO) optimization [49], derivative-free trust region method (DF-TRM) [50]. BO has been largely used in AutoDS for the model selection and hyper-parameter tuning tasks [10]. However, it suffers from poor scalability with increasing dimensionality. ZO optimization mimics first-order descent-type methods and have been applied in AutoDS for solving multi-armed bandit problems [51]. In contrast to BO, ZO optimization has provably favorable convergence. However, it suffers from high query complexity and the smoothness requirement in the objective function. DF-TRM approximates the black-box function with a parametric surrogate function (such as linear or quadratic functions) fit on the available function evaluations, which allows for efficient constrained optimization on the surrogate [50], [52]–[54]. For the continuous variables they are restricted to a neighborhood of the current point (the trust region) and for the discrete variables the number of changes is restricted. The size of the trust region corrects for the discrepancy between the parametric surrogate and the black-box. DF-TRM is computationally intensive, impeding its widespread use compared to BO and ZO optimization. At this point its also worthwhile mentioning that one can combine multiple AI technologies such as leveraging AI planning in this context [55] to tackle the various challenges.

D. Evolutionary Algorithms

Reinforcement learning methods follow the ubiquitous biological paradigm of reward-based trial and error in order to create automated systems for AI. Another natural paradigm that copies biological principles is that of evolutionary algorithms. These methods encode solutions to problems (e.g., neural architectures) as codes that are akin to chromosomes in the biological DNA. Just as biological organisms evolve and become intelligent through (Darwinian) reward-driven trial-and-error, evolutionary algorithms (EA) use a Darwinian approach of selecting highly-performing architectures and recombining their best characteristics to explore the search space. It is not difficult to see that this is a rough simulation of how the biological brain has evolved over the millenia. EA-based optimizers follow a standard optimization strategy wherein a population of networks is maintained and a set of evolution steps are carried out until termination. This traditionally involves 1) selecting "parents" (i.e., highly performing architectures) in a "Darwinian" way 2) applying mutation and recombination operations to create new "individuals" 3) evaluating the "fitness" of the recombined individuals. This is followed by repeating the Darwinian process of selecting the most fit survivors of the population and recombining them again. Evolutionary approaches span a highly diverse set of ADL methods each of which vary in their definitions of encodings, set of mutations and selection strategies, notable ones being [56], [57]. The main challenge arises from the large search space and over which one must optimize the population. Indeed, the success of biological evolution is owed to the fact that the evolution of the ecosystem can be viewed as a massively (but loosely) parallel process that has occurred over hundreds of millions of years over millions of species. We have nothing approaching that kind of computational power today.

E. Meta-Learning

A truly automated system can make reward-driven decisions, just like any intelligent organism. Although this goal is achieved to some extent by reinforcement learning, the main problem with reinforcement learning solutions is that they require massive amounts of data, and are therefore mostly good for learning in artificial settings like board/video games, where one can generate unlimited amounts of data with self-play. Similarly, although reinforcement learning robots can learn to work on their own in virtual simulators relatively easily, this is much harder to generalize to physical robots, where the speed of data collection is constrained by the limitations of the number of tasks a physical robot can perform in a specific period of time, and also by the fragility of physical robots to the consequences of "bad" trials. On the other hand, most humans can learn from relatively limited amounts of data. For example, a child does not need too many examples of a toy truck to learn what it is, and can easily generalize the idea to real-world trucks of completely different shape, size, and color. This difference between humans and automated learning systems can be partially explained by the fact that the former is far more powerful in terms of its ability to perform unsupervised learning. Humans take in massive amounts of sensory input, which is processed in an unsupervised way in order to perform continuous learning over time. This unsupervised learning can be viewed as a continuous process of massive "pre-

training" that makes it easier to learn specific tasks with a smaller amount of data. Furthermore, much of the unsupervised *and* supervised learning is encoded in the highly "regularized" neural structure of the brain, which has evolved over millions of years and is inherited from one generation to the next. This inheritance from one generation to the next is a form of *transfer learning* [58], which is required to be incorporated more seamlessly in automated systems than is possible with the highly customized systems available today. Ideas surrounding learning over many past problems in order to learn more efficiently for the current problem is often called *meta-learning* or *learning to learn* [59].

IV. Outlook and Perspective

Whatever AI approach will succeed in automating end-to-end data science it will need to overcome a number of challenges: **Generalization.** The first is the issue of overfitting that is further amplified by automation since a machine can essentially fine-tune the flow until generalization is non-existent. In some cases, the original goals of the application are compromised due to an automated system, such as an RL system, overfitting to difficult-to-specify rewards. While generalization has been widely studied, the implications of fully automated systems in this respect are not quite as well studied [60].
Safety. An alarming observation in this respect correspond to safety issues– RL systems regularly learn cheats and hacks in video games that were not originally intended. It is not useful to have a robot that makes messes and then cleans them up– or, more darkly, a robot nurse with similar behavior.
Deep vs Non-Deep Learning. Many of the ideas developed solely in the context of deep learning can be applied more generally to data science with any ML method, and vice versa. We expect and hope to see more transfer between these bodies of work. In some data modalities, such as images, what might otherwise be considered data preparation transformations (at least certain ones) are handled by layers in the network itself, blending data preparation and model into one *end-to-end* training process. This general idea can be extended beyond deep learning. Likewise, in practice, not all of the data preparation needed for practical problems is always effectively captured in network layers, and thus the kinds of treatments of the data preparation transformations outside of deep learning can be applied to systems that employ deep learning as well.
Domain Knowledge. It is notable that this survey has given little discussion to the notion of domain knowledge, which is so important to most data scientists. Domain knowledge can range from simple derived features such as the body mass index (BMI) of a person, to encoding the syntax of a language for an NLP application. While it is an article of faith among data scientists that data science solutions benefit from domain knowledge, its role in fully automated data science remains an open question. A key observation is that its track record has been mixed in more advanced forms of AI. With limited data, domain knowledge has indeed been extremely useful as a natural regularizer. However, with increased data, a surprising observation has been that completely data-driven learners (with zero domain knowledge) have consistently outperformed systems with encoded domain knowledge. For example, machine translators with zero knowledge of syntax now routinely outperform domain rule-based machine translators, and chess learners with zero knowledge (e.g., *AlphaZero*) now routinely outperform chess programs like *Stockfish* in which chess grandmasters have spent years in fine tuning the evaluation function. This is simply due to the fact that domain knowledge can be a "biased" upper-bound to the intelligence we expect the system to eventually have. To give a biological analogy, a child can learn a lot from the domain knowledge given by parents, but it is a poor substitute to what the child can learn through their own experiences. There is also no common-sense reasoning in a machine setting — let alone reasoning at an domain expert level. However, there will always be some situations in which domain knowledge is useful. Early work already exists [61], but it still has a long way to go.

Lifelong learning and unsupervised learning. Another is that of life-long learning - while various works exists on this topic already it is incredibly complex to automatically facilitate life-long learning. One reason is the missing taxonomy of data. For instance, when the predictions of the model influence the newly incoming data it is often not valid to use those data points for retraining. Unsupervised learning remains a significant challenge, because it is hard for systems to know which parts of the massive amounts of unsupervised data will be useful in future applications. To a large extent, it is expected that truly automated and intelligent systems will be obtained by combining reinforcement learning, unsupervised learning, and transfer learning in a way that is not fully understood today. While reinforcement learning systems require massive amounts of data, this problem will be alleviated by unsupervised learning over large periods of time that can take in massive amounts of data, and learn the portions that can obtain rein-

forcement rewards. At the same time, transfer learning will be required in order to inherit the generalizable knowledge over different situations in much the same way as humans inherit the highly regularized structure of their neurons across generations. In this sense, transfer learning can be viewed as a simpler and faster alternative to what one tries to achieve with the use of evolutionary algorithms. This ability to combine unsupervised learning, supervised learning, and reinforcement learning in a reward-driven context has remained the most important problem in artificial intelligence in recent years. Recent successes in unsupervised learning, such as the ability to realistically replicate intricate data objects like images with the use of *generative adversarial networks* has brought significant advancements to unsupervised learning — however, a complete and seamless integration of supervised, unsupervised, and reinforcement learning remains elusive.

Computational cost. The single largest impediment to automated data science is the computational aspect of it. One must view the limited success of automated data science by comparing it to what biological organisms have benefited from– we have nothing close to the computational power that simulates the massively parallel "computations" that have occurred in the (implicit) "computational" process of biological evolution over billions of organisms. One should view our relative successes and failures to biological learning and automation from the perspective of the humbling limitations we work with today. However, with new hardware paradigms on the horizon, such as quantum computing, it is difficult to know how much progress we will make along these lines— the only certainty is that one should expect the unexpected.

REFERENCES

[1] C. C. Aggarwal, X. Kong, Q. Gu, J. Han, and S. Y. Philip, "Active learning: A survey," in *Data Classification*, Chapman and Hall/CRC, 2014.

[2] G. Kougka, A. Gounaris, and A. Simitsis, "The many faces of data-centric workflow optimization: a survey," *IJDSA*, vol. 6, no. 2, pp. 81–107, 2018.

[3] F. Nargesian, H. Samulowitz, U. Khurana, E. B. Khalil, and D. Turaga, "Learning feature engineering for classification," in *IJCAI*, 2017.

[4] Kaggle, "The state of data science & machine learning," *https://www.kaggle.com/surveys/2017*, 2017.

[5] A. Knobbe, A. Siebes, and D. van der Wallen, "Multi-relational decision tree induction," in *PKDD*, 1999.

[6] J. M. Kanter and K. Veeramachaneni, "Deep feature synthesis: Towards automating data science endeavors," in *DSAA*, 2015.

[7] T. L. Hoang, J. Thiebaut, M. Sinn, B. Chen, T. Mai, and O. Alkan, "One button machine for automating feature engineering in relational databases," *CoRR*, vol. abs/1706.00327, 2017.

[8] T. L. Hoang, T. N. Minh, M. Sinn, B. Buesser, and M. Wistuba, "Learning features for relational data," *CoRR*, vol. abs/1801.05372, 2018.

[9] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:09.144*, 2016.

[10] C. Thornton, H. H. Hoos, F. Hutter, and K. Leyton-Brown, "Auto-weka: Automated selection and hyper-parameter optimization of classification algorithms," *CoRR*, vol. abs/1208.3719, 2012.

[11] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient automated machine learning," in *NeurIPS*, 2015.

[12] J. Vanschoren, "Meta-learning: A survey," *arXiv preprint arXiv:1810.03548*, 2018.

[13] Y. Luo, X. Qin, N. Tang, and G. Li, "Deepeye: Towards automatic data visualization," in *ICDE*, 2018.

[14] D. S. Soper, "A framework for automated web business intelligence systems," in *HICSS*, 2005.

[15] P. G. Nagy, M. J. Warnock, M. Daly, C. Toland, C. D. Meenan, and R. S. Mezrich, "Informatics in radiology: automated web-based graphical dashboard for radiology operational business intelligence," *Radiographics*, vol. 29, no. 7, pp. 1897–1906, 2009.

[16] R. S. Sutton and A. G. Barto, *Reinforcement learning - an introduction*. Adaptive computation and machine learning, MIT Press, 1998.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[18] R. Noothigattu, D. Bouneffouf, N. Mattei, R. Chandra, P. Madan, K. R. Varshney, M. Campbell, M. Singh, and F. Rossi, "Teaching AI agents ethical values using reinforcement learning and policy orchestration," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pp. 6377–6381, 2019.

[19] R. Allesiardo, R. Féraud, and D. Bouneffouf, "A neural networks committee for the contextual bandit problem," in *Neural Information Processing - 21st International Conference, ICONIP 2014, Kuching, Malaysia, November 3-6, 2014. Proceedings, Part I*, pp. 374–381, 2014.

[20] B. Lin, D. Bouneffouf, G. A. Cecchi, and I. Rish, "Contextual bandit with adaptive feature extraction," in *2018 IEEE International Conference on Data Mining Workshops, ICDM Workshops, Singapore, Singapore, November 17-20, 2018*, pp. 937–944, 2018.

[21] D. Bouneffouf and R. Féraud, "Multi-armed bandit problem with known trend," *Neurocomputing*, vol. 205, pp. 16–21, 2016.

[22] A. Balakrishnan, D. Bouneffouf, N. Mattei, and F. Rossi, "Incorporating behavioral constraints in online AI systems," in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019.*, pp. 3–11, 2019.

[23] D. Bouneffouf, S. Parthasarathy, H. Samulowitz, and M. Wistuba, "Optimal exploitation of clustering and history information in multi-armed bandit," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pp. 2016–2022, 2019.

[24] D. Bouneffouf, A. Bouzeghoub, and A. L. Gançarski, "A contextual-bandit algorithm for mobile context-aware recommender system," in *International Conference on Neural Information Processing*, pp. 324–331, Springer, 2012.

[25] D. Bouneffouf, A. Bouzeghoub, and A. L. Gançarski, "Exploration/exploitation trade-off in mobile context-aware recom-

mender systems," in *Australasian Joint Conference on Artificial Intelligence*, pp. 591–601, Springer, 2012.
[26] D. Bouneffouf, A. Bouzeghoub, and A. L. Gançarski, "Contextual bandits for context-based information retrieval," in *International Conference on Neural Information Processing*, pp. 35–42, Springer, 2013.
[27] D. Bouneffouf, A. Bouzeghoub, and A. L. Gançarski, "Hybrid-ε-greedy for mobile context-aware recommender system," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 468–479, Springer, 2012.
[28] D. Bouneffouf, R. Laroche, T. Urvoy, R. Féraud, and R. Allesiardo, "Contextual bandit for active learning: Active thompson sampling," in *International Conference on Neural Information Processing*, pp. 405–412, Springer, 2014.
[29] D. Bouneffouf, I. Rish, G. A. Cecchi, and R. Féraud, "Context attentive bandits: Contextual bandit with restricted context," in *IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pp. 1468–1475, 2017.
[30] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," *arXiv preprint arXiv:1611.02167*, 2016.
[31] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.
[32] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, "Efficient architecture search by network transformation," in *AAAI*, 2018.
[33] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.
[34] T. Chen, I. J. Goodfellow, and J. Shlens, "Net2net: Accelerating learning via knowledge transfer," *CoRR*, vol. abs/1511.05641, 2015.
[35] M. Wistuba, "Deep learning architecture search by neuro-cell-based evolution with function-preserving mutations," in *ECML/PKDD*, 2018.
[36] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le, "Understanding and simplifying one-shot architecture search," in *ICML*, 2018.
[37] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *ICML*, 2018.
[38] H. Liu, K. Simonyan, and Y. Yang, "DARTS: differentiable architecture search," *CoRR*, vol. abs/1806.09055, 2018.
[39] S. Xie, H. Zheng, C. Liu, and L. Lin, "SNAS: stochastic neural architecture search," in *ICLR*, 2019.
[40] B. Baker, O. Gupta, R. Raskar, and N. Naik, "Accelerating neural architecture search using performance prediction," *CoRR*, vol. abs/1705.10823, 2017.
[41] C. Hsu, S. Chang, D. Juan, J. Pan, Y. Chen, W. Wei, and S. Chang, "MONAS: multi-objective neural architecture search using reinforcement learning," *CoRR*, vol. abs/1806.10332, 2018.
[42] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via lamarckian evolution," in *ICLR*, 2019.
[43] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *ICLR*, 2018.
[44] I. Bello, B. Zoph, V. Vasudevan, and Q. V. Le, "Neural optimizer search with reinforcement learning," in *ICML*, 2017.
[45] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," in *ICLR*, 2018.
[46] M. Riemer, T. Klinger, D. Bouneffouf, and M. Franceschini, "Scalable recollections for continual lifelong learning," in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019.*, pp. 1352–1359, 2019.
[47] E. D. Cubuk, B. Zoph, D. Mané, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation policies from data," *CoRR*, vol. abs/1805.501, 2018.
[48] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," *IEEE*, vol. 104, no. 1, pp. 148–175, 2016.
[49] Y. Nesterov and V. Spokoiny, "Random gradient-free minimization of convex functions," *FCM*, vol. 2, no. 17, pp. 527–566, 2015.
[50] A. R. Conn, K. Scheinberg, and L. Vicente, "Global convergence of general derivative-free trust-region algorithms to first-and second-order critical points," *SIAM Journal*, vol. 20, no. 1, pp. 387–415, 2009.
[51] A. Agarwal, O. Dekel, and L. Xiao, "Optimal algorithms for online convex optimization with multi-point bandit feedback," in *COLT*, 2010.
[52] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to derivative-free optimization*, vol. 8. Siam, 2009.
[53] A. Choromanska, B. Cowen, S. Kumaravel, R. Luss, M. Rigotti, I. Rish, P. Diachille, V. Gurev, B. Kingsbury, R. Tejwani, and D. Bouneffouf, "Beyond backprop: Online alternating minimization with auxiliary variables," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pp. 1193–1202, 2019.
[54] S. Liu, P. Ram, D. Bouneffouf, G. Bramble, A. R. Conn, H. Samulowitz, and A. G. Gray, "Automated machine learning via ADMM," *CoRR*, vol. abs/1905.00424, 2019.
[55] A. Biem, M. Butrico, M. Feblowitz, T. Klinger, Y. Malitsky, K. Ng, A. Perer, C. Reddy, A. Riabov, H. Samulowitz, D. M. Sow, G. Tesauro, and D. S. Turaga, "Towards cognitive automation of data science," in *AAAI*, 2015.
[56] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *ICML*, 2017.
[57] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Aging evolution for image classifier architecture search," in *AAAI*, 2019.
[58] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.
[59] S. Thrun and L. Pratt, *Learning to learn*. Springer Science & Business Media, 2012.
[60] C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, and A. Roth, "The reusable holdout: Preserving validity in adaptive data analysis," *Science*, vol. 349, no. 1, pp. 636–638, 2015.
[61] Z. Hu, Z. Yang, R. R. Salakhutdinov, L. Qin, X. Liang, H. Dong, and E. P. Xing, "Deep generative models with learnable knowledge constraints," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pp. 10522–10533, 2018.