Applying Machine Learning for Sensor Data Analysis in Interactive Systems: Common Pitfalls of Pragmatic Use and Ways to Avoid Them

THOMAS PLÖTZ, School of Interactive Computing, College of Computing, Georgia Institute of Technology, USA

With the widespread proliferation of (miniaturized) sensing facilities and the massive growth and popularity of the field of machine learning (ML) research, new frontiers in automated sensor data analysis have been explored that lead to paradigm shifts in many application domains. In fact, many practitioners now employ and rely more and more on ML methods as integral part of their sensor data analysis workflows—thereby not necessarily being ML experts or having an interest in becoming one. The availability of toolkits that can readily be used by practitioners has led to immense popularity and widespread adoption and, in essence, pragmatic use of ML methods. ML having become mainstream helps pushing the core agenda of practitioners, yet it comes with the danger of misusing methods and as such running the risk of leading to misguiding if not flawed results.

Based on years of observations in the ubiquitous and interactive computing domain that extensively relies on sensors and automated sensor data analysis, and on having taught and worked with numerous students in the field, in this article I advocate a considerate use of ML methods by practitioners, i.e., non-ML experts, and elaborate on pitfalls of an overly pragmatic use of ML techniques. The article not only identifies and illustrates the most common issues, it also offers ways and practical guidelines to avoid these, which shall help practitioners to benefit from employing ML in their core research domains and applications.

CCS Concepts: • **Human-centered computing → Human computer interaction (HCI)**; **Ubiquitous and mobile computing**; • **Computing methodologies → Machine learning;**

Additional Key Words and Phrases: Sensor data analysis, machine learning applications

ACM Reference format:
Thomas Plötz. 2021. Applying Machine Learning for Sensor Data Analysis in Interactive Systems: Common Pitfalls of Pragmatic Use and Ways to Avoid Them. *ACM Comput. Surv.* 54, 6, Article 134 (July 2021), 25 pages.
https://doi.org/10.1145/3459666

1 INTRODUCTION

Many contemporary computing systems are based on some form of automated sensor data analysis. Prominent examples are innovative and more intuitive input modalities such as voice and gesture [37], or automated activity logging and analysis [47]. Such prominence requires robust and reliable methods that can cope with the challenges of real-world applications and systems, of

Author's address: T. Plötz, School of Interactive Computing, College of Computing, Georgia Institute of Technology, 85 5th Street N.W., Atlanta, GA, 30332; email: thomas.ploetz@gatech.edu.

0360-0300/2021/07-ART134 $15.00
https://doi.org/10.1145/3459666

which there are many: noisy sensor readings; often ambiguous, sometimes erroneous annotation (labeling [43]); small sample datasets; hard real-time constraints for analysis; and so on [60]. As a key component of sensor data analysis, many researchers now employ **machine learning (ML)** techniques, especially those related to the automated analysis of time-series data [8]. ML aims at deriving algorithms that can automatically adapt towards a specific problem or application domain through estimating parameters of statistical models from example data [7, 52]. ML methods are flexible and thus have the potential to help solving challenging problems without having to reinvent entire analysis pipelines from scratch every time the application changes ever so slightly. In recent years the field of ML has seen an explosion in growth and sophisticated methods now do exist that are key enablers for many application areas, such as, Natural Language Processing [36], Computer Vision [65, 69], or finance [54], to name but a few.

The tremendous uptake of ML methods is—at least partially—correlated with the availability of user-friendly toolkits and software frameworks that nicely package complex ML methods and thus make them accessible to practitioners, i.e., non-experts in ML research. Examples of popular ML toolkits are Matlab [49], Weka [22], scikit-learn [59], Orange [16] and—more recently—the various deep learning frameworks (e.g., TensorFlow [1] or PyTorch [57]). Arguably, packaging *away* the complexity of ML methods is both a blessing and a curse. It is favorable because it opens up a rich toolset to practitioners who are interested in benefitting from the tremendous potential ML methods have for the automated analysis of complex data. However, it is problematic because it lowers the barrier for (accidental) misuse of methods without practitioners even being aware of problems related to incorporating ML techniques into their systems. Such problems can range from models not fitting the analyzed data, overfitting and thus lack of generalization through small, biased training datasets, to issues related to the training and validating ML methods in scenarios where ground truth annotation is hard to obtain or ambiguous. Without careful considerations of appropriateness of methods for specific problems practitioners are at risk of falling victim to "flaky" conclusions [14].

There have been efforts within the core ML research community to illustrate what ML techniques can do [20], as well as to raise awareness of what they cannot [70]. There have also been efforts in the broader computing community, most notably in the Human Computer Interaction (HCI) research field, to promote the appropriate use of data analysis methods that employ techniques from statistics and, ultimately, ML [10]. This article extends these efforts by targeting it directly at the current generation of computer scientists who want to employ ML methods. I advocate the enormous potential ML methods have for current and next generations of computing applications and systems—specifically targeting sensor data analysis problems that are at the core of most interactive systems. I do so by carefully unpacking typical ML-based procedures with a view on identifying common pitfalls that an overly pragmatic use of ML methods inevitably brings, which leads to those aforementioned "flaky" conclusions. Through identifying and analyzing these pitfalls I offer ways to avoid these, which serve as guidelines for practitioners working with ML methods but who are not necessarily experts in core ML research. My analysis is focused on when (not) to use ML methods, practical aspects of training and validating ML models, issues of ground truth annotation of sample data, realistic method evaluation and reporting of results, guidance on modeling paradigms, and on practical deployments of ML methods in real-world applications.

Formulating these guidelines serves two purposes: *(i)* The wider computing community develops a huge range of innovative applications and many take great pride in this strict application focus [2]. With great rigor researchers and practitioners explore, develop, and evaluate new ways of computing that go (way) beyond the traditional desktop paradigm. This article shall give hands-on advise for practitioners who want to apply ML methods in their work—thereby maximizing

their effectiveness and as such the practical outcomes. *(ii)* When exploring existing related work, practitioners can easily get overwhelmed by technical details that might complicate suitability assessments of the methods used for their own research. The guidelines shall serve as a checklist that allows to draw conclusions for one's own work. The observations that led to this article are based on my experience from years of working in the field, from reviewing (and publishing) countless papers in the area, and from teaching and working with numerous students. The target audience for this article is the community of practitioners who employ ML techniques in their research and are not necessarily core ML experts. Some of the aspects discussed in this article may seem obvious to some (if not many). However, according to my experience, even if that was the case it is worth reminding ourselves as a community to be rigorous and systematic in our work and I argue that this is of special importance when applying ML techniques.

2 MACHINE LEARNING FOR SENSOR DATA ANALYSIS IN INTERACTIVE SYSTEMS

The field of machine learning aims at developing algorithms that are capable of adapting towards new situations and circumstances [7, 52]. ML methods are typically based on probabilistic models whose parameters are derived from example data through statistical learning [32]. It is the flexibility of these learning methods that render them very attractive for many challenging application domains that come with uncertainty about the data themselves, their annotation, i.e., the actual meaning of the data, or about the general circumstances of the domain. ML methods learn representations [5] and models from example data, thereby aiming for generalization beyond those training data such that the resulting models and methods can effectively be used for accurate analysis of unseen, yet related data. A plethora of methods has been developed over the years, which can be categorized into two main classes: *(i)* supervised learning, i.e., methods that derive models from example data and their associated class annotations [11]; and *(ii)* unsupervised learning, i.e., methods that derive models purely from unlabelled sample data [4]. Variants and combinations of these two main categories exist [77, 78]. Main algorithmic challenges for ML research lie in issues related to convergence towards stable and optimal solutions, generalization that avoids over- or under-fitting to the example data, and reproducibility and intelligibility, to name but a few.

ML has been used in contemporary computing systems primarily for sensor data analysis tasks, of which there are many. The vast majority of such systems, which include mobile, wearable, ubiquitous (or pervasive), and interactive computing [41], is based and relies on various physical or virtual sensors. Two main categories of sensors can be discriminated: *(i)* those that record static data, such as cameras; and *(ii)* those that record sequential, i.e., time-series data. Many systems utilize the latter category of sensors incl., for example, **inertial measurement units (IMUs)** for movement analysis; environmental sensors capturing, e.g., light, temperature, humidity, or noise levels; microphones for analysis of spoken content or other acoustic conditions; or sensors integrated into the infrastructure such as those that measure water pressure, electric current, and so on [15, 24, 58].

The main application case for ML in such settings is the analysis of sensor data for capturing and assessing the context a user is in [17]. This context awareness enables proactive and situated actions provided by a computer system that shifts the focus of the user from being preoccupied with interacting with the machine in a typically non-intuitive manner to implicit interaction that allows for focusing on the actual tasks that shall be accomplished [73]. An important context factor is location, and a range of ML techniques have been developed and deployed to infer the location of a user [33], to predict future locations and purpose of trips [23], and to infer the semantic meaning of locations [42]. For outdoor localization typically GPS is used, which provides spatio-temporal data streams that have to be analyzed using time-series analysis methods. For indoor localization, the sensing mechanism has to be different (which still represents a non-trivial challenge), yet the

general characteristics of the sensor data analysis and as such of the ML methods used are similar [75]. Activities are another important context factor and a wealth of sensing approaches and especially ML methods for sensor-based **human activity recognition (HAR)** have been developed and deployed [10, 13].

Sensor data analysis is challenging and as such the requirements and constraints on ML methods are substantial [60]: *(i)* sensor data are notoriously noisy, complicating model training; *(ii)* typically only small amounts of *labelled* sample data exist, which impacts supervised learning methods; *(iii)* annotation is often ambiguous if not erroneous, which leads to the challenge of training under uncertainty [43]; *(iv)* many scenarios come with resource restrictions, which limits the choice of ML methods [45]; and *(v)* interactive systems impose hard (near) real-time constraints on each system component in general, and on ML methods in particular [55].

3 POSTULATES OF APPLIED MACHINE LEARNING

For developing new ways of computing, practitioners adopt and adapt methods and techniques from a wide range of research areas, for example, sensing, signal processing, image processing and computer vision, communication and networking, and many more. ML methods are now widely used as well to tackle challenging sensor data analysis problems. While many computer scientists primarily adopt ML techniques for their specific needs, the field also gives back to the core ML research domain and pushes the state-of-the-art there. Very prominently, the recent adoption of Deep Learning techniques for human activity recognition based on wearable sensing platforms has led to a substantial push on developing ways to effectively train such complex models in scenarios where there are only small labeled datasets available for model training and validation [44, 71]; where the quality of the data to be analyzed varies including missing, noisy, and ambiguous data [26]; and where hard real-time and resource constraints need to be met for applications in wearable computing scenarios.

Arguably, many computer scientists would not characterize themselves as ML experts but rather as users of ML methods. With the availability of ML toolkits practitioners now actually do not have to be core ML experts but are rather in a position to employ ML methods in a very pragmatic way, which is of substantial benefit for pushing their core research agenda by being able to develop systems and applications that can integrate ML-based data analysis. Yet, such (overly) pragmatic use of ML methods comes with the danger of (accidental) misuse. Ian Witten, project lead of the popular Weka ML toolkit [74], summarized the danger of "blindly" using ML toolkits as follows:

> "In some research papers [researchers using Weka] have come to some *flaky* conclusions … in a way that really isn't contributing in any way to science." [14]

A common example of such "flaky" conclusions is the overconfidence in a model's generalization capabilities. Often the training and validation datasets used for developing ML-based sensor data analysis methods are simply too small or not representative enough to draw meaningful conclusions about the effectiveness of a resulting model in real-life application scenarios. As such, impressive recognition accuracy values on the (small) test set may have much less practical meaning than what they seem to suggest.

In what follows I will explore the most common pitfalls of applied ML for sensor data analysis in interactive computing systems as it is of fundamental importance for the community of non-ML researchers and practitioners. In addition to identifying pitfalls I will offer ways to avoid these and thus to prevent those "flaky conclusions" (according to Witten), which shall serve as practical guidelines for researchers and practitioners who want to benefit from applying ML techniques without falling victim to—often hidden—flaws. In combination, this exploration constitutes

a collection of *Postulates of Applied Machine Learning* that shall be of value to researchers and practitioners who are interested in gaining the most from ML-based sensor data analysis methods.

3.1 When (Not) to Use Machine Learning

The very first pitfall of applied machine learning that I want to elaborate on here may come as a surprise:

1. Not Everything is a Machine Learning Problem.

Key elements of contemporary computing systems are context sensitivity and implicit interaction, which often rely on automated sensor data analysis. Given the multitude of sensing capabilities and the challenges of "in-the-wild" application scenarios this constitutes a formidable task comprising many aspects of signal processing and pattern recognition. The nature of unconstrained application scenarios with vague, ambiguous interactions; uncertainty in sensor data and their annotations; missing data, and so on, often results in hard data analysis problems. As such it is all but understandable that practitioners are attracted by solutions that come with the promise of solving these problems "out of the box." ML techniques seem to reduce the analysis problem to a data collection and annotation problem. With enough data, so the hope, it will be possible to solve all analysis problems at least to some satisfying extent. Whether or not this is actually realistic obviously depends on the application and problem domain.

There are plenty of very successful applications of ML techniques in, for example, mobile and wearable computing domains, such as activity or gesture recognition, localization, or context inference in general [2]. There is, per se, nothing wrong with employing ML methods for challenging data analysis tasks like these. It becomes a problem, however, when employing ML techniques turns into an automatism—almost like a "knee-jerk reflex"—that is, when practitioners stop questioning whether tackling a specific problem actually requires complex methodologies like ML techniques. It is worth reminding that ML techniques are by definition probabilistic methods. That means that—to some extent—the comfort of deterministic model behavior is deliberately given up for uncertainty. If not done properly (as discussed later and addressed by Witten [74]) then the employment of ML techniques can bring one—often unnecessarily—closer to those "flaky conclusions," that is, to systems that may not actually work all that well in practice.

As it is good scientific practice one should always choose those analysis methods that are appropriate for a specific problem. There is no such thing as a universal problem solver and one has to be careful to not fall victim to Maslow's Law of the Instrument: "If the only tool you have is a hammer, it is tempting to treat everything as it were a nail" [48, p.16]. The danger with unquestioned, reflex-like employment of ML techniques lies in overfitting to example data, i.e., training scenarios, and thus leading to limited generalization capabilities, which typically results in systems that do not work well in real-world scenarios. For many applications this danger can be avoided altogether—by *not* employing ML techniques! In line with Maslow's hammer analogy, researchers and practitioners have access to a huge, properly filled toolbox and as such can choose from a wealth of analysis methods that may be more appropriate for certain scenarios. One prominent application example shall serve as illustration for the first postulate of applied ML: step-counting through pedometers.

Pedometers are wearable devices that consist of a movement sensor (typically tri-axial accelerometers if not full inertial measurement units) that record a wearer's activities in form of continuous sensor data streams. By employing automated analysis methods these sensor data streams are then analyzed with regards to how many steps the wearer has walked. This analysis problem can be broken down into either recognizing characteristic patterns in the sensor data streams—individually or through their combination in the magnitude signal—that represent

individual steps and then count them, or into recognizing periods of walking and then inferring the number of steps for these walking bouts. Neither of these two analysis variants require machine learning.

In the first case, signal cleaning through, for example, range normalization and de-noising through smoothing will lead to typically very clean signals that can easily be analyzed through thresholding. One can exploit the fact that with every footfall there will be a significant peak (followed by a trough) in the acceleration signal, and thus flank detection through thresholding will very reliably identify individual steps—without any learning component. In the second case, preprocessing would focus, for example, on spectral analysis to filter out those portions of the data that are related to walking and then apply auto-correlation analysis that unveils the principal walking frequency, which combined with information about the bout length will lead directly to step counts—without any learning component. These straightforward solutions require an understanding of the underlying phenomenon (steps) and how it manifests in the data (periodic patterns with characteristic peaks and troughs in, for example, acceleration in direction of earth's gravity), which can easily be obtained through observation or studying the physical phenomenon that is analyzed.

And yet, entire cohorts of students resort almost immediately to ML techniques when confronted with the assignment of developing a pedometer. The typical reflex is to employ some form of the standard activity recognition pipeline (summarized, for example, in Reference [10]): Record and annotate example data, extract analysis frames through a sliding window procedure, extract (heuristic) features, and train a classifier. Moreover, even published academic papers on (variants of) such rather fundamental movement analysis problems employ—unnecessarily—ML methods.

Practical Guidelines

In practice it has turned out to be very fruitful to carefully think about a sensor data analysis problem *not* with the initial assumption that ML techniques shall be used. Instead, it is often helpful to first think about employing more conventional yet more focussed methods such as classical signal processing including filtering, normalization, and so on, and, for example, thresholding and rule-based analysis for decision making. ML techniques should be resorted to if those conventional, non-ML techniques do not solve the specific analysis problem. In essence, this approach applies Occam's razor, and thus good scientific practice [19, 63]. In my experience this procedure often helps with better understanding the problem that is to be tackled, ultimately leading to better, i.e., more effective solutions—either based or not based on ML. Guidelines on when (not) to consider ML techniques for sensor data analysis can be summarized as follows:

Robustness over Generalization: When aiming for real-world applications, robustness of the developed systems becomes imperative. As such researchers should always consider whether a seemingly more generic ML-based solution in practice can live up to the expectation of robustness in real-world applications, or if a conventional, more specific sensor data analysis method that is tailored to the particular use case is more appropriate. If in doubt, a researcher should always apply Occam's razor and opt for the simpler, that is, less complex method—which in many cases will not be ML-based. One should solve the problem that needs to be solved—no less, but also no more.

Constraints and Restrictions in Practical Applications: Extending the previous point, real-world scenarios typically not only have hard requirements on robustness but also constraints and restrictions on available computational resources. This needs to be taken into account when making the decision for or against using ML techniques. Even with the galloping progress in hardware

development and thus ever increasing computational capabilities, there will always remain some form of resource constraint that will lead to bottlenecks. Addressing these through tailored sensor data analysis methods is typically a promising endeavor. For real-time applications one might want to favor simple yet effective signal preprocessing operations over more complex, generic methods. For example, in many recognition scenarios it is sufficient to use distribution-based sensor data representations like ECDF—that can be computed with linear time complexity [29]—over, for example, feature learning [31, 61].

Probabilistic vs. Deterministic: The main attraction of ML methods lies in their ability to adapt towards specific scenarios through learning from sample data. Probabilistic models are derived through estimating model parameters that optimize a cost function, which is evaluated on validation data. While there is no doubt about the flexibility and the generalization capabilities of such methods, one has to bear in mind that such probabilistic models—by definition—always produce *probabilistic* predictions. With a view on reproducibility, which links to user expectations, such—strictly speaking—non-deterministic behavior may not always be desired or appropriate, even if probabilistic predictions may be more accurate on average. Before employing ML, one should consider this aspect carefully.

Interpretability of Results: The majority of ML techniques essentially resemble "black box" methods, i.e., it is typically not possible to retrace how—let alone why—a model made a specific decision or prediction. Usually, this is a desired behavior as the potential of ML techniques lies in exploiting and modeling higher level, typically hidden relationships within the data. In fact, the success of ML methods for complex recognition tasks proves that it is of benefit to learn from the data rather than merely replicating what human experts already know. However, for many application scenarios one might need to understand the reasons why a certain prediction was made. This can be for liability reasons or for assurance in mission critical systems. As such when making the decision to use or not use ML one should always consider whether results need to be interpretable. If that is the case, then the majority of *current* ML methods cannot be used "as is." Note, however, that the wider ML research community has now started to move towards studying intelligibility of ML methods [25].

Insights into Physical Phenomena: In similar vein, it is of interest to understand and exploit certain physical phenomena. For example, in activity assessments through wearables, insights into concrete movement parameters are of essence. Alternatively, in on-body communication scenarios it is often interesting to measure and understand signal propagation through tissue or bones. For these, and many other scenarios, "black box" ML techniques may not be the best choice, as their predictions would effectively hide the underlying characteristics.

3.2 Training and Testing Machine Learning Models

Once it has been established that an analysis problem at hand indeed justifies the use of ML techniques, then the actual work on designing, training, and validating an ML-based sensor data analysis pipeline begins—and with that the majority of pitfalls have to be expected and dealt with. ML-based methods learn from sample data thereby aiming for generalization of the analysis capabilities to related, yet unseen test data. This generalization is crucial, as the overall suitability of an approach relies on it. Oftentimes ML applications seem to see an astonishingly high proportion of cases where recognition accuracies of way above 90%, often approaching perfect results of 100%, are reported. While it is, of course, desirable and—if achieved—very fortunate to reach such outstanding analysis capabilities, a healthy skepticism is advisable, which I summarize in the second postulate on applied machine learning:

2. If you reach 100%, think about the data.[1]

To understand the gravity of this postulate it is worth exploring the possible cases that would lead to near perfect recognition accuracy. In general, reaching 100% accuracy implies that the analysis problem at hand has been solved. In the absence of any procedural flaw this, of course, would be the ultimate, desirable goal. Once it has been reached, one should, however, revisit the first postulate on applied machine learning (Section 3.1) to validate whether the analysis problem at hand actually required the use of ML techniques in the first place.

To shine light on what could be possible procedural flaws, which, I argue, is the more realistic conclusion from reaching (near) perfect recognition accuracy, it is worth reconsidering the general principle of ML. In ML, the parameters of probabilistic models are estimated on a sample (set) S using some learning algorithm, which will eventually produce a hypothesis h. The sample S is drawn from an overarching distribution D. Typically, D is unknown and S is represented by the (annotated) example data that are recorded during system design. In statistical terms the sample S is randomly drawn from D and the objective of the learning algorithm is to produce a hypothesis h (the model) that generalizes from S to the unknown D. Within this framework 100% accuracy corresponds to the hypothesis h always being correct, which is extremely unlikely in any probabilistic framework—if executed correctly.

That leaves three possibilities for (near) perfect accuracy: *(i)* $S == D$; *(ii)* there is an issue with the distribution D; or *(iii)* there is an issue with the sample S. The first case represents the trivial case that all data points that could possibly originate from a problem domain are known and covered by the example data, which is unlikely. If that was the case, then by definition no ML or any kind of sophisticated analysis is required as the recognition problem is essentially reduced to a lookup task, given that all possible instantiations from a problem domain are known and accessible. The second possibility implies that the studied problem domain, which is represented by the (unknown) distribution D, is "wrong" or that some issue pertains to it. This is either an ill-conceived notion or more fundamental issues with defining the problem domain exist. In the context of this article, this possibility can thus safely be ignored, which leaves the third possibility as the only remaining one that would explain near perfect analysis results.

In fact, many issues can be associated with the sample S that is used for training and validating a ML model. The following two scenarios are the most common ones: First, S could be non-representative for the distribution D, which is a typical problem as data recording and especially annotation is a non-trivial endeavor in real-life application scenarios. Special care needs to be taken to record a non-biased, representative dataset, which often can be accomplished by focusing data collection efforts on realistic scenarios such as in "in the wild" or field studies. Second, and more critically, the separation of S into disjoint training- and test-sets can be compromised leading to the violation of a fundamental principle in ML research as described in what follows.

Training and evaluating an ML model is based on example data. It is an almost trivial endeavor to obtain very high, if not perfect performance on the training data. However, this training accuracy is not very informative, because the objective of model optimization is to derive parameters that generalize beyond the example data, namely, to unseen test data and ideally to the entire distribution D. For that reason, it is important to test the model capabilities on unseen data for which the sample S has to be separated into disjoint subsets S_{train} and S_{test} where $S_{\text{train}} \cap S_{\text{test}}$. This separation into training and test data is typically done randomly. However, with a view on representativeness it is advisable to consider the underlying class distribution and to balance the selection. In an ideal case one would create three disjoint datasets: S_{train}, S_{val}, and S_{test}, with the

[1]Note that the inverse—"If you reach 0%, think about the data"—essentially corresponds to the same phenomenon/problem.

already discussed training and test sets plus a separate validation set. Model training is performed on S_{train}, optimization of (hyper-)parameters is done on the validation set S_{val}, and the test set S_{test} is held back and only used for final testing. Results should be reported for the test set.

All three sample sets need to be of reasonable size, representative, and annotated. For many scenarios these are hard constraints and often researchers struggle to satisfy these. As a consequence it is often necessary to divert from the ideal case of having access to three separate datasets for model training, validation, and evaluation. For more economic—yet biased—use of available data often the ideal of having access to a dedicated validation set is dropped. In this case model optimization has to be performed entirely on the training set, which bears the risk of overfitting. In many practical scenarios the available amount of annotated sample data is even smaller, and thus the creation of reasonably sized, balanced, and annotated disjoint training and test sets becomes impossible. In such cases, researchers often resort to an approximative protocol, namely, to k-fold cross validation [3]. The existing dataset is divided into k subsets—folds—of (roughly) equal size thereby matching the class distributions of each fold to the global distribution of the overall dataset S (stratification). Model training (and optimization) is then performed on $k - 1$ folds, and testing on the remaining k-th one. All unique combinations of $k - 1$ training folds are used and final results are averaged over all k experiments. Cross validation makes the most economic, yet unbiased use of a dataset. Variants exist where the folds are defined, for example, by recording session or subject.

Within this framework—and following the line of argumentation so far—(near) perfect analysis results can only stem from the fact that training- and test-sets are in-fact *not* disjoint. In practice, that either means that model testing was performed on (parts of the) training data, i.e., $S_{\text{test}} \subset S_{\text{train}}$, or, vice versa, that model training was performed on (parts of the) test data, i.e., $S_{\text{train}} \subset S_{\text{test}}$. Both cases must be avoided as they lead to overoptimistic if not flawed evaluation results. In static data analysis scenarios, such as image analysis, mixing training and test data is a problem that typically can easily be identified and rectified. Identically and independently distributed (*i.i.d.*) data points are randomly or as per some stratification strategy distributed into the separate subsets, be it the hold out datasets S_{train}, (S_{val}), S_{test}, or the k folds for the cross-validation protocol. However, for the analysis of time series data, such as sensor data streams, it can be more difficult and less obvious to recognize that training and test data have been mixed. A common practice for sensor data analysis is to segment continuous data streams into overlapping analysis frames and then to analyze these frames in isolation [10]. Random distributions of these frames into training (and validation), and test set, however, violate the *i.i.d.* assumption for these frames as—clearly—subsequent samples are not independent of each other. The problem becomes even worse for overlapping frames, as here identical data points are distributed into more than one frame. It has been shown that for a typical $k = 10$-fold cross-validation the likelihood of assigning subsequent, hence highly correlated, if not partially identical, data points into different subsets—training and test—is 99%, resulting in an epidemic of violating the principle of working with disjoint training and test data [30].

Practical Guidelines

The primary guideline for any researcher and practitioner who employs ML techniques for sensor data analysis is to preserve a healthy skepticism with regards to analysis results that seem suspiciously good (or bad). In my experience, for many real-world and thus non-trivial analysis scenarios near perfect accuracy values almost always indicate some issue with the experimentation protocol. In addition to this initial "sanity check," the following practical guidelines on training and testing procedures can be formulated:

Careful Preparation and Execution of Training and Evaluation Protocol: For most, if not all, application scenarios it is of utmost importance to make most economic use of sample data. While it is usually straightforward to record virtually unlimited amounts of sensor data, obtaining labels for these is often hard and requires great efforts, and sometimes it is even impossible. As such, proper planning and experiment design is essential. The use of hold out datasets for model training, validation, and testing represents the scenario that most closely resembles real-world use cases and thus should be prioritized if sufficient enough amounts of representative data are available. If that is not the case, then k-fold cross-validation should be employed. Care needs to be taken when determining the number of folds k and distributing the data across the folds. Every fold should contain a representative number of samples with balanced class distribution. Harder but, depending on the application case, possibly more realistic protocols aim at session-wise evaluation where, for example, data from one user is held back for evaluation while training is based on sessions from other users only. This scheme is typically combined with k-fold cross-validation and provides insights into how well a method generalizes. Especially for highly biased datasets, such as when, for example, the null class dominates the whole dataset, it is worth considering data augmentation or condensation techniques that would rebalance the dataset for improved model training.

Evaluation Metrics: The scientific toolbox offers a huge variety of methods that have been designed for evaluation purposes ranging from classification accuracy, precision, recall and F-scores, to area under curve, to name but a few [62]. One has to be careful to choose the metric that is most relevant and realistic. Classification accuracy is thereby usually not the best measure as it does not capture well the reality of typical scenarios with often heavily imbalanced class distributions. F-measures, and in particular F1 scores, are very popular as they focus on both negative and positive predictions. Again, care has to be taken for imbalanced datasets. While both weighted and average F1 scores take class imbalances into account, especially the former only amplifies the problem of skewed distributions and thus inflates recognition results. While there is no single one evaluation measure that would be ideal for all possible application scenarios, starting with F1-scores, that is, F1 scores averaged over all classes, is typically a reasonable strategy.

In addition, one needs to consider what level of evaluation is of interest. In scenarios where high sampling rates are used and long-term recordings are analyzed (e.g., Reference [18]), reporting results at sample-level may not be appropriate as the typically dominant null class will lead to inflated results that are not very informative. Instead, one could consider reporting results at event level [72] as in many practical applications it is of interest to report how well specific instances of activities can be predicted rather than focusing on arbitrary sample precision.

3.3 Annotation and Labelling

Annotation of sample data refers to the manual provision of information regarding the class identity of data points that are collected during system development. These annotations often come in form of value pairs where samples $\mathbf{x}$ are associated with labels y that originate from a finite set of M possible options and a NULL class y_0: $y \in \{y_0, y_1, \ldots y_M\}$. In reality, annotators observe data—typically in form of video recordings that accompany a sensor data collection effort—and then label instances of interest along the timeline, which then implicitly translates into a sample-wise annotation. These expert annotations serve two purposes: *(i)* they provide the basis for supervised learning methods (as described below); and *(ii)* they provide the basis for quantitative evaluations of the recognition capabilities of the derived ML models and thus enable rigorous assessment of the capabilities of ML models, as described in Section 3.2. As such, annotation plays a crucial role for the development and evaluation of ML-based systems.

Because of its central role for model training and evaluation, annotating example data is a common procedure and thus a routine task when deploying ML methods. However, it remains an underestimated problem with many pitfalls that, if not addressed properly, can lead to substantial issues that can harm the effectiveness of a system, which leads me to the formulation of the third postulate of applied machine learning:

3. There is no ground truth.

The term "ground truth" is often used to refer to manual data annotations. The underlying assumption is that expert annotators label example data in a way that is: *(i)* accurate; *(ii)* non-ambiguous; *(iii)* consistent; and *(iv)* at the required level of granularity. However, the reality of many applications is challenging when it comes to data annotation and as such often one or more of these requirements cannot be met. Unfortunately, it is also a reality that even if it was possible to adhere to these annotation criteria, sometimes not enough care is taken and one or more of the requirements are—not necessarily willingly—violated.

Intuitively, providing accurate annotation is the requirement that causes the least problems and thus could be met easily. Primarily, accurate annotation refers to annotators not making mistakes while labelling data. With proper training, a realistic timeline that includes breaks for annotators to address inevitable fatigue, and especially with multiple annotators labeling the same data, accuracy in manual annotation can be achieved to a very high degree. The latter aspect should be part of the annotation routine, as it is in many other disciplines that rely on manual observations, e.g., clinical psychology. An annotation routine that has worked well in practice and scales reasonably well to larger scale endeavors can be summarized as follows: Labels for sensor data are provided by an even number of independent, trained human annotators (at least two). These labels are then compared and accepted if the majority of annotators agree on the same label for a datapoint. In case of no majority consensus, a "tie-breaker" is called in who will then decide on the final annotation, possibly after having discussed the discrepancies with the annotators. This procedure is conducted for the whole dataset, potentially organizing annotations by session or any other smaller unit. One should keep track of the frequency of discrepancies occurring during this annotation process. Frequent labelling disagreements are an indicator for either poorly trained human annotators, or—more likely—for issues with the annotation instructions, and require modifications of the process. Annotator agreement can be quantized through specific measures, such as kappa statistics (inter-rater reliability) [50]. However, it is not always straightforward to implement such a formalization due to ambiguities inherent to the data.

Specific to interactive computing scenarios, especially in mobile settings, are challenges related to obtaining data annotations from "in-the-wild" deployments where technology is used outside lab environments. Here, it is often very challenging, if not impossible, to have human annotators label sample data even if they have received expert training and aforementioned multi-annotator procedures are implemented. It is simply impossible to follow users for the sake of annotation without changing their behavior (and thus skew the data) or intruding their privacy. Approaches to ask the users themselves to provide data annotation have been proposed and studied extensively (e.g., Reference [53]). However, these user annotations are, strictly speaking, not considered "ground truth," as they may (and will) vary from user to user and as such not the same rigor can be applied as for multi-annotator labeling procedures such as the one outlined above.

Human annotators need to be trained to create high-quality, accurate labelling for sample data. The training procedure typically comprises familiarization with annotation tools, the general application scenario, and—most importantly—the specifications of what and how shall be annotated. Especially the latter aspect is seemingly often neglected, as all too often researchers fall back to assumed common sense and thus rather implicit definitions, and as such detailed labelling

specifications are often not provided (or at least hardly ever reported). As a prominent example, in human activity recognition often the activities of interest are only informally described (running, walking, sitting, standing, etc.). Even though most humans do have good intuitions about what constitutes such activities, the details are much more challenging. For supervised learning approaches, such as the widely used, standard activity recognition chain [10], one would typically need sample-precise annotation or otherwise runs the risk of hurting both model training and evaluation by compromised sample data [40]. For such annotations it needs to be specified when a certain activity starts, when it ends, and what exactly defines it and discriminates it from others (e.g., running vs. walking).

One has to be rigorous in this aspect and can build on expertise and practices in other fields where human observations and annotations play a key role in research. For example, clinical psychologists or veterinarians use very detailed specifications for annotating behaviors. Depending on the specific application domain such specifications are referred to through different terminologies such as behavioral topographies, operational definitions, taxonomies, or behavior ethograms. Most importantly, very concrete and especially non-ambiguous descriptions are spelled out that are used for annotator training and serve as documentation of label definitions. Having access to such detailed operational definitions enables high-quality, that is, non-ambiguous, annotation. Arguably these specifications should be included into publications or at least be made available to the readership through other means, e.g., on an accompanying website or in an appendix. The reason for this is that other researchers may want to evaluate their own methods on an existing dataset, and domain experts are, of course, interested in the findings and thus would need access to such a detailed, non-ambiguous specification.

Practical Guidelines

The aforementioned requirements on manual annotations of sensor data are strict, and it is often challenging to meet them in practical application scenarios. The absence of ground truth is not a reason to not employ ML, though. However, in the interest of transparency, rigor, and replicability as well as reusability (of datasets and methods) practitioners need to adhere to certain standards and best practices for manual data annotation, for which the following practical guidelines can be formulated:

Operational Definitions: As explained before it is of utmost importance to specify and use operational definitions for the annotation of sensor data streams. For the benefit of the community—who may want to replicate published results, want to understand in more detail and build upon some published findings that are based on employing ML techniques for sensor data analysis, or who want to work with published datasets to improve and validate methods—these operational definitions need to be made accessible to the interested readership. Ideally, and space permitting, they should be an integral part of an evaluation section in any academic paper that integrates methods that are based on manual annotations. If space is limited, then overviews should be given in the main publication and full details should be given in external addenda such as websites or appendices.

Multiple, Independent Labelers: Even the best, most intensively trained human annotators make mistakes. Human error is an aspect that can be minimized but never be eliminated, and as such it should be acknowledged. The best way to deal with human errors during the annotation process is to have multiple, independent annotators label the sensor data in parallel and then derive the final annotation through consolidation among all available labels. That way the likelihood of inaccurate annotation is effectively minimized. At the same time having multiple annotators serves as an effective means for checking for consistency and practicality of the annotation process.

Accept Ambiguity but Provide Transparency: Disambiguation for manual annotation is an ideal that, unfortunately, is not achievable for many application domains. As such, one often has to work with imperfect annotations. A reasonable strategy to deal with this situation is to aim for transparency. One should acknowledge potential ambiguities either explicitly or implicitly by disclosing full details of the annotation procedure including operational definitions, logistics in terms of tools used, number of annotators and their training as well as expertise, details of the consolidation process employed for resolving conflicting annotations, and so on. Managing expectations is possible if transparency is provided.

Annotation Is Expensive: When embarking on a research and development endeavor that includes a manual annotation component, one should be aware that this annotation process will come with considerable costs. These costs are related to the annotation infrastructure, including software tools and computers for parallel, independent labelling through multiple annotators, training (time and material) of the annotators, and the annotation labor itself. Thorough annotation takes time and as such is expensive. For the benefit of the community one should always aim for making datasets and their annotations available to the public. Dedicated web portals such as the UCI repository of ML datasets [21] exist that facilitate dataset sharing.

3.4 Realistic Evaluations

Many practitioners aim at building systems that are designed and developed for actual, that is, out-of-the-lab deployments. As such, ML methods will almost always have to analyze sensor data that are recorded "in the wild," i.e., outside lab-like, controlled environments. A logical consequence of this application focus is that experimental evaluations in general and ML methods in particular need to focus on realistic, real-world application scenarios, which leads to the formulation of the fourth postulate:

4. Field test, or it did not happen.

With the application focus of many practitioners this postulate may seem somewhat obvious. However, still a lot of (published) work reports evaluation results on lab-based data. Focusing on lab-collected data only for evaluating ML-based systems arguably comes with the danger of limited generalization. If a ML model simply has never seen real-world data, then it is unlikely that it will function well enough in non-artificial settings. At the very least, lab-based evaluations do not provide any actual evidence for a system working well in the intended application domains.

As reader of publications in the field one should be skeptical if it is claimed that a system that is trained and tested on lab-data only would work well "in the field." Even for scenarios that seem obvious with regard to reflecting real life conditions, one needs to insist on evidence from actual real-world deployments. Take, for example, the case of accelerometry-based activity recognition that aims at automatically logging activities of daily living [68]. As mentioned before (and elsewhere, e.g., Reference [53]) it is often very challenging to obtain sample data annotation in such scenarios, and thus it may be tempting to record data in scripted user studies—a practice that is employed rather frequently [12, 64]. In such scenarios users are asked to follow a pre-defined protocol, which certainly makes data annotation much easier, as it reduces the task to instructing participants and making sure they are following the protocol. However, scripted interactions hardly qualify as "natural" and following a protocol certainly has an effect on the participant's behavior, ranging from observer (Hawthorne [66]) effect to plain mismatches of regular routines that are not captured by even the most sophisticated data collection protocols. As such, one should refrain from employing scripted evaluation protocols.

As author and system developer one should embrace the application focus and be opportunistic about what seemingly comes with more effort and is more challenging than working with data that were recorded in well controllable lab environments. In fact, it has even been shown that training ML models on more challenging real-world data improves generalization capabilities and robustness of models [27]. This is in stark contrast to the common intuition that model training should be based on the "best possible data," that is, "clean" data with high-fidelity data annotation. Oftentimes such clean, lab-based data collections simply do not represent the reality of actual application scenarios, and as such model training methods will struggle to reach generalization. Deriving models from "less clean" data is certainly more challenging but will lead to increased robustness and, in the end, to better modeling.

Arguably, the straightforward recommendation for practitioners who want to employ ML techniques is to focus their data collection efforts on real-world application scenarios, that is, to deploy their systems "in-the-field." There is a rich body of literature that provides guidelines on how to plan and conduct field studies (see, for example, the excellent overview in Reference [9]). Practitioners who want to employ ML techniques should follow these guidelines, as it will allow them to record meaningful data that will be of value for training and evaluating their ML models.

Practical Guidelines

Real-world deployments, i.e., field tests, should be the main objectives of researchers who employ ML methods in their systems, for which the following practical guidelines can be formulated:

Real-world Evaluation: As a golden rule, even if it is tempting and more convenient to record data in the lab or through scripted procedures, practitioners should not shy away from out-of-the-lab deployments. Real-world applications are a distinctive ambition and they should also remain within the core focus when ML methods are employed. This does not necessarily exclude method evaluation on existing, that is, not self-recorded, benchmark datasets. However, such benchmark datasets need to be realistic and challenging enough by essentially representing the field tests as discussed in this section.

Number of Participants/Size of Datasets: Study design and statistical evaluation are well-researched fields. As such, there is a well established body of knowledge on how participants should be recruited—most importantly: *how many*. ML models are derived from example data that are drawn from an underlying (typically unknown) distribution. To make this process as unbiased as practically possible, a representative and large enough dataset must be collected, which translates into recruiting a sufficient number of participants into a field test. Statistics provides a rich toolbox with methods that allow to determine how large a sample set needs to be to be able to evaluate the effectiveness of a method in a meaningful way (see also next section). A prominent example is statistical power analysis that facilitates determining the effect size for a given sample size and method [34]. Practitioners must use such methods as well to not fall victim to overfitting of their methods to too small, potentially biased datasets. However, the majority—if not all—of these standard methods assume that an analysis method (here, the ML method that is applied) is fixed and all parameters are known such that, for example, a power analysis can be performed. Unfortunately, for many applications this is not the case, as the analysis method itself is subject to development and thus to change, which often rules out standard statistical determinations of sample set size. This is a well-known problem in the ML literature and no definite, analytical solution exists. For practitioners, however, there are a number of suggestions that can minimize the risk of overfitting:

(i) The number of samples required for robust model training and evaluation is correlated with the number of parameters that need to be estimated for an ML model [56]. Per parameter a number of samples should be recorded, e.g., $n = 5$.

(ii) The number of parameters is not only dependent on the particular model that is employed, with a huge variability among the wealth of ML methods that exist. The main dependency is on the dimensionality of the data, which should be taken into consideration during system design and evaluation [38].

(iii) Start with a conservative estimation of required sample set size, for example, by calculating the number of parameters of a standard modeling method (e.g., a Gaussian Mixture Model) thereby taking into account the considerations *(i)* and *(ii)*. This estimation on standard models can enable the calculation of a minimum sample set size to be used for drawing meaningful conclusions from an experimental evaluation.

Recruit Actual Users: A lot of research is conducted in universities and by students. As such, it is tempting, and in fact somewhat straightforward, and thus very common, to recruit participants for field tests from this cohort. However, with a view on the realistic evaluations that I am advocating here, one has to be careful to not bias the evaluation. If a system aims at general, possibly population-wide, applicability, then the experimental evaluation should be based on a representative sample of this wider population and include, for example, not only college-age, technology-savvy individuals. If such a recruitment is not possible or not wanted because a system might actually be targeted towards a very specific population, then this should be disclosed and explained. By any means, in such cases researchers need to refrain from overclaiming with regard to generalization capabilities. In any case it is important to provide full details of the demographics of the participant cohort such that readers will be able to judge the validity of a study and its conclusions. It is also worth mentioning that study participants should ideally not be closely related to the researchers (which includes friends) as this would also bias the results. Interestingly, experiments in other fields have shown that proxy populations may actually be utilized for evaluating certain *tasks* though not necessarily models and automated assessment systems, as they are targeted by ML methods for sensor data analysis. In Reference [35] software engineering tasks were conducted by a group of students and a group of professionals. The authors did not see substantial differences in task fulfillment between the two groups, which suggests that students could be utilized as proxy population for assessing such tasks. However, concerns regarding enacting target user behavior by such a proxy population remain as outlined above and the suggestion stands to recruit actual users of ML-based analysis systems.

System Developers Are Taboo for Evaluation: As a direct yet most important extension of the previous aspect it is necessary to point out that system developers should never be part of the evaluation studies. Anyone working on developing a system is—if only implicitly or subconsciously—biased, as they know how the system works and as such are very likely to use the system and behave in a way that is not representative for naive users. While this seems to be a rather obvious aspect, it is surprising to see how little it is actually known and adhered to in practice, which is, arguably, dangerous, as this can easily compromise all findings. As researchers, we need to be rigorous as always—even if that complicates data collection and the general logistics of field tests.

"No Secret Sauce": It is worth reiterating that all details of an evaluation need to be documented in a way that allows readers to understand and assess the procedure and the results. Ideally, the description should be in a way that experiments and results can be repeated by others. If possible (anonymized), then datasets should be made available and all sources and experimentation scripts including documentation should be published. Many ML methods require the optimization

of hyper-parameters, that is, parameters that are not directly related to a specific method or model, but are rather related to the way model training is performed. These hyper-parameters and the values used need to be published and explained as otherwise it will not be possible to replicate the work, which limits its impact. Note that ensuring reproducibility should be common practice in any scientific field. Yet, specifically when employing ML methods with their various hyper-parameters and potential dependence on specific configurations of model training procedures due diligence needs to be applied when describing the methods such that fellow researchers and practitioners can replicate the presented work.

3.5 Proper Reporting of Results

The effectiveness of ML techniques is primarily evaluated through quantitative methods. When employing ML methods, inevitably one has to focus on such quantitative evaluation for system development. Note that oftentimes a quantitative evaluation comes in addition to other means of evaluation including qualitative, ethnographic methods [9]. Conducting quantitative evaluations and especially reporting results requires rigor and careful execution, which leads to the formulation of the fifth postulate:

5. Proper reporting, or it did not happen.

One would assume that reporting results of a quantitative evaluation should be straightforward with little room (if any) for ambiguity. Arguably, unlike for qualitative evaluations it is typically much easier to define a framework that reduces the evaluation to raw numbers. However, in practice it is very often the case that the results of quantitative evaluation studies are reported poorly or incompletely, which then easily can, and often does, lead to false conclusions. The main dimensions for transparently reporting results of quantitative evaluations for the majority of applications of ML-based sensor data analysis are as follows (cf. Section 3.2 for more details on how to properly evaluate):

(i) Evaluation metric: For many applications one class often dominates the sensor data analysis scenario. Most prominently, the "always on" character of contemporary computing systems usually leads to an overrepresentation of the NULL class, i.e., sensor data that are recorded when the system is *not* actively being used. As argued before, such a biased (or skewed) class distribution can lead to inflated evaluation results that overestimate the actual capabilities of an analysis system. Care needs to be taken to choose a suitable evaluation metric that realistically reflects the capabilities of an analysis system [62].
(ii) Evaluation protocol: Rigorous evaluation is difficult, yet one must not shy away from doing it properly (Section 3.2). Similarly important is the proper documentation of all relevant details of the evaluation protocol(s) employed for the assessment. This includes dataset descriptions, as well as specifications (and publication) of how the datasets are separated into training, validation, test—or into folds if k-fold cross-validation is employed. The chosen evaluation protocol needs to be motivated and explained through the application scenario. With that it needs to become clear, why, for example, the more challenging **leave-one-subject-out (LOSO)** protocol was not applied and instead k-fold cross-validation was used. The primary objective of such transparency is to allow fellow researchers to replicate the work and especially the results that are reported.
(iii) Significance Test: Based on the observation that the term "significance" is often used in a misguiding, if not erroneous, way, it is worth reminding that statistical significance is a defined concept in hypothesis testing that essentially quantifies whether a result is likely to reject the null-hypothesis of being random or not. Only if that null-hypothesis can be rejected, a result can be considered statistically significant. As such, authors have to be careful when using this term,

because it requires that an actual significance test was conducted (which should be good academic practice anyway), and that results are reported including their confidence intervals. Using the framework of statistical significance testing, all technical details of the procedure need to be documented. These details include the justification for the particular type of significance test, the calculated confidence intervals, and the conclusion regarding significance (or not).
(iv) Level of Granularity: Sensor data can be analyzed at various levels of granularity: samples, frames, sequences, sessions, and so on. While there may always be good reasons for choosing one over the other, for the sake of replication and fair comparison one has to be transparent about it, which includes specification and justification. For example, human activity recognition systems are often based on analyzing frames of consecutive portions of sensor readings, which is reasonable given that the amount of information carried by an individual sensor reading is rather limited. However, to be able to compare different analysis methods the evaluations should be based on the same temporal horizon of the analysis, i.e., the same framesize needs to be used. Without specifying the details of the level of granularity for an evaluation, system comparison is difficult if not impossible.

Practical Guidelines

Trust in experimental evaluations of ML-based systems only comes from rigorous assessment protocols and transparent reporting of results. A number of pitfalls exist that shall be avoided when it comes to reporting results, for which the following practical guidelines can be formulated:

Transparency about the Choice of Evaluation Metric: Even though F1 scores seem to be the de facto standard for many applications of ML-based sensor data analysis, there may be reasons for utilizing different measures. A large number of evaluation metrics exists (cf., e.g., Reference [76]) and each and everyone of them has their justified use case. In any case, authors need to justify their choice of evaluation metric—even if the standard F1 scores are used. This justification needs to directly link to the application case and especially to the particulars of the analyzed dataset(s) and explain what the chosen evaluation metric actually measures with regards to the targeted application scenario.

Assess Complexity of Analysis Task: It is not always straightforward to assess how challenging a data analysis task is and what results can realistically be expected in terms of, for example, classification accuracy. With the looming issues of potential errors with the evaluation protocols, authors should always make an effort to explain and discuss the complexity of an analysis task, which essentially will shine light on how difficult it is. In addition, such an analysis will also illustrate how good is "good enough" for an application scenario. It is very much application dependent if near perfect classification results are required, for example, in mission-critical scenarios, or if higher margins of error are tolerable. The data analysis could, for example, only be one part of a larger processing pipeline with potential for post-classification correction, e.g., through integration with other system parts. First and foremost, authors need to be transparent about expectations on and potential of ML-based sensor data analysis.

Always Report Details of Significance Test: The importance of significance tests cannot be stressed enough. Especially when it comes to comparing ML methods, and, based on this, making design decisions for a computing system, rigorous evaluation is required. As a simple rule of thumb, practitioners should *always* provide the details of statistical significance tests when it comes to reporting results of quantitative evaluations. For practically all deployments of ML techniques the training and validation protocols, or even the analysis methods themselves, include some elements of randomness. As such, the most straightforward evaluation is to repeat all analysis multiple times

(standard is 30 repetitions) and then average over the results and provide standard deviations as confidence intervals. Alternatively, standard significance tests shall be applied as appropriate and all details reported. Care has to be taken with regard to the potential necessity for applying score correction methods (such as Bonferroni, or Benjamini-Hochberg correction [51]) if multiple hypotheses are tested. It is far beyond the scope of this article to provide detailed guidelines on which significance test has to be applied for specific scenarios and how. The reader is referred to a rich volume of literature that covers the mechanics of statistical significance tests in great detail. Here, I am strongly advocating for actually employing significance tests and reporting all relevant details.

Report Not Only Peak Performance: Arguably, it is tempting for authors to limit reporting the results of experimental evaluations to peak performance, i.e., to only present the best results. From a practitioner's perspective, ML-based sensor data analysis typically represents only one part of the overall computing system and as such one is primarily interested in the best performing ML system that contributes to the overall system. However, there are good reasons to report more than only peak performance. Fellow practitioners typically face similar yet not identical application scenarios and thus requirements on the ML methods they employ for their sensor data analysis. By providing insights into *how* peak performance was achieved, it becomes much easier for other researchers to understand how a particular analysis method can be adapted towards their own requirements. Few studies have been published that investigate the impact of the various parameters of ML methods on assessment results, independent of actual peak performance [28]. In the core ML research community, and especially in the Deep Learning community, it has now become standard to report results of ablation studies in which the impact of various processing steps or parameter optimizations is reported. This is good practice, which ML practitioners should adopt.

3.6 Modeling for Data

The overarching paradigm of ML is to develop algorithms that adapt based on example data. This adaptation essentially corresponds to estimating parameters of typically probabilistic models—a process referred to as "learning." With this, the challenge is not only in designing the algorithms, which in itself is a non-trivial endeavor, but also in obtaining suitable amounts of representative example data. Contemporary ML methods are becoming more and more data-hungry, notably the popular deep learning techniques with their millions of parameters. As such, it is an all but too familiar request to ask for more training data: "There is no data, like more data!"[2] However, for many practical application domains of ML it is not straightforward, if not impossible, to record large amounts of labeled sample data, as it would be required for the predominant approach of supervised learning. Instead, every effort needs to be made to "make the most" of existing datasets, which leads to the formulation of the sixth postulate:

6. Data rule, models serve.

In theory, any (reasonable) probabilistic classification system will produce perfect recognition results if infinite amounts of annotated training samples are available. Arguably, for practical reasons this is not a realistic assumption and thus hardly any perfect ML system exists (Section 3.2). While it is often almost trivial for many applications to record large amounts of unlabelled sample data, it is typically challenging to also obtain label information for these data. For example, for health applications, such as in automated behavior monitoring using wearable sensors, labeling sensor data that were recorded in real-life scenarios is difficult for either logistical reasons, or even

[2]This quote is attributed to Robert L. Mercer (IBM Research) in the context of Hidden Markov Models for automated speech recognition and the relationship between their classification accuracy and the size of the training dataset [39].

unethical when it comes to serious events. One simply cannot ask participants of, for example, a fall study to deliberately fall "realistically" just for the sake of recording more training data.

Lamenting about small datasets for developing and validating ML methods is typically not only not very constructive, it may actually deflect from flaws in the overall modeling approach that may have led to the (perceived) lack of sample data in the first place. It is all but too tempting to follow current trends (in ML) and employ the latest methods that seem to promise spectacular recognition capabilities as reported in the literature. However, one should always bear in mind one of the most important theorems of pattern recognition and ML: "There is no free lunch!" (e.g., Reference [56]), i.e., no single classifier exists that universally performs best for each and every classification scenario. In fact, it can be shown that if no domain knowledge is incorporated into classifier design, i.e., when "off-the-shelf" methods are employed, then any classifier can be outperformed by random decisions. The direct consequence of this theorem[3] is that one always needs to carefully tailor the modeling approach towards the *existing* datasets, instead of simply just hoping for more data.

Practical Guidelines

In scenarios with strict constraints on availability (and quality) of example data, one needs to be extra careful on how to derive effective ML models, for which the following guidelines can be formulated:

Economic Use of Existing Data: First and foremost, it is important to actually use those datasets that are available in the best possible way. This includes, for example, training strategies that exploit datasets of limited size in an effective manner. Instead of, for example, trying to train a classifier in one holistic attempt using the entire training dataset, one could start with training a less complex classifier first, which is then used to initialize the training of the final classifier. As an example, it is well-known that the convergence of the **Expectation Maximization (EM)** algorithm heavily depends on a reasonable initialization [6]. Random initialization is typically not suitable, because EM-based optimization can easily get stuck in local optima. Instead, one could start with a more conventional clustering approach, such as k-means, and use the resulting codebook as initialization for the actual EM optimization step(s) [67]. Another strategy for economic use of small datasets is to reduce the dimensionality of the data themselves by eliminating redundancy in the data. Given that many feature representations inevitably capture at least some redundant information, it is reasonable to apply dimensionality reduction methods such as **Principle Component Analysis (PCA)**. With lower dimensionality typically come fewer model parameters that need to be estimated, and as such smaller datasets suffice.

Beyond these two exemplary approaches there are no limits for how to exploit the datasets one has at their disposal in the most effective way. These strategies typically rely on the intimate, practitioner's domain knowledge, which supports the overall paradigm of applied ML: Practitioners need to know their application domain and translate this knowledge into the design of the data analysis components of their systems.

Understand the Characteristics of the Data: The fundamental question that ML practitioners have to address is, which model to employ. With the incredible variety of models that have been developed over the years and are seemingly ready to be used, this task may be daunting. However, instead of instinctively either falling back to relatively simple models, or to follow the *zeitgeist* and (blindly) use models that are popular, one should carefully consider which model

[3]In fact, there is a whole family of "No free lunch" theorems, however, it is sufficient to limit the discussion here to the most prominent one.

to use. The key to such considerations is an understanding of the data that will be analyzed. By definition the practitioner possesses expert domain knowledge and as such should have a very detailed understanding of their application domain. In addition to this, a rich set of analysis tools exists that allows practitioners to understand their data with the ultimate goal of choosing the most suitable model for a data analysis scenario. It is beyond the scope of this article to even attempt a complete list of guidelines on how to characterize the data that have to be analyzed. To illustrate common pitfalls, I will give two examples: *(i)* Often models are employed that make assumptions on certain characteristics of the data. Most prominently, linear models (such as linear regression) are often chosen, because they are of limited complexity and thus seemingly straightforward to train. However, many data distributions actually are not linear and as such employing a linear model is at least limiting the analysis capabilities. *(ii)* Naive Bayes classifiers are incredibly popular, seemingly due to their low complexity and thus the implicit promise of easy trainability. However, Naive Bayes comes with the strong (naive) assumption of independence between data dimensions (features). Unfortunately, for a large set of applications such an independence assumption does not hold. As such, there is a substantial risk for overfitting to—especially small—training sets, which often is hard to recognize due to the small size and limited variability of the test data.

To not fall victim to "flaky" conclusions that are typically reasoned by the small annotated sample datasets practitioners have access to, it is important to thoroughly analyze the characteristics of the data that are to be analyzed and choose the analysis models accordingly.

(Hyper-)Parameter Optimization over Default Values: Extending the previous considerations, many contemporary computing systems that use ML techniques tend to use the analysis models with their default parameters. For example, Support Vector Machines come with a host of parameters (such as slack, or the configurations of the Gaussian when using **Radial Basis Function (RBF)** kernels)—yet often papers do not discuss how these hyper-parameters are optimized and one comes to think that default parameters have been used as the chosen toolkit provides them. With increasing complexity of the analysis models such an approach leads to problems of model robustness. Hardly ever will the default parameters be suitable for the specifics of an application domain that had never been addressed before.

3.7 The Challenges of Real-world Sensor Data Analysis

The final postulate again relates to the ambitions and focus of practitioners to build systems for real-world applications. The most important consequence of this ambition can be summarized as follows:

7. There is no oracle.

The nature of "always-on" systems is that they continuously record sensor data, and that these sensor data need to be continuously analyzed. While this sounds obvious, it is mainly actual products that comply with the strict always-on paradigm. A surprisingly high proportion of research papers violates this paradigm in one way or another, which is concerning specifically when real-world readiness is claimed and the systems are evaluated in field deployments. Specifically for ML-based sensor analysis this real-world focus comes with substantial challenges that are not necessarily related to the core analysis but rather to marginal cases such as outlier or exceptions. Unfortunately, these cases cannot be ignored, as it is expected that an automatic assessment system always provides reasonable results such that the overall computing system remains in a well-defined state.

Practical Guidelines

The key recommendations related to "always-on" capabilities of ML techniques for sensor data analysis can be summarized in the following practical guidelines:

Segmentation Is Crucial: Especially for time-series data, such as sensor data streams, but to some extent also for static data such as images, automated analysis is not limited to classification only, i.e., to assigning a class label to a data point. In addition, relevant data points need to be localized within the overall data stream, a process that is referred to as segmentation. The traditional approach to solving this dual analysis problem is to have a dedicated segmentation process that identifies relevant data portions, which are then forwarded to the classifier. Obviously, failures in the first stage of this process, i.e., missed segments are typically impossible to recover from, which motivates the popular sliding window procedure in which an analysis frame is shifted over the entire dataset (continuously) and every window is processed by the classifier. This procedure relies on a suitable configuration, which is often difficult to achieve [46]. Independent of which segmentation approach is utilized, it is important that segmentation is actually performed. It is of limited interest only to report results of the classification step only because that would assume the existence of an oracle that guides the classifier to the relevant portions of the sensor data. The oracle is the segmentation stage, and system developers need to invest into it, because the effectiveness of the data analysis procedure crucially relies on it.

Include Reject Models: Any analysis system needs to provide a decision for each and every data sample that is recorded. Assuming a functioning segmentation procedure, the classifier typically makes a decision between a finite set of classes, which corresponds to a forced decision making process. The harder problem for an ML method to tackle is to decide when it can *not* make a decision, for example, because certain data points are very dissimilar to anything the system was presented with during training. The majority of ML-based computing systems are based on the forced classification paradigm. Yet, for real-world applications it would be beneficial to provide a reject (or "do not know") classification rather than a false one. Starting points for such a reject class are, for example, class posterior probabilities, which can be used as confidence scores. If class posteriors for a data point are (close to) uniformly distributed, i.e., when no clear "winner" can be determined among the set of classes, then a reject classification could be a more reasonable option than deciding for the class with the only marginally maximum posterior.

Open-world Assumption: Extending the aforementioned concept of including a dedicated reject model, ML-based systems should adapt to changing circumstances during deployment. If, for example, data samples are frequently caught by the reject model, then it is reasonable to assume that deployment circumstances have changed. Rejected data points can be used for extending the analysis system by extending the set of known classes, thereby moving away from the implicit closed-world assumption. In such an open-world (or open-ended) scenario the ML system would dynamically be adapted towards the real-world circumstances it faces during deployment. Obviously, care has to be taken to prevent over- or mis-adaption for which again the class posteriors could be used combined with, e.g., a robust clustering strategy.

4 CONCLUSION

Applied machine learning is a driving force for much of the research within many areas of computer science. In fact, ML methods have now become a key enabler for many practical applications, because they essentially serve as the algorithmic backbone of sensor data analysis that is at the heart of many contemporary computing systems. As such, it is exciting to witness the enormous growth of core ML research, and at the same time the widespread adoption of ML methods by

practitioners and non-core ML researchers. Tremendous progress has been made by employing ML techniques in analysis domains that are considered challenging.

The main purpose of this article is to raise awareness for and focus the attention on pitfalls that are associated with an overly pragmatic use of ML techniques. Such a pragmatic use of ML methods can lead to issues with rigor that may result in questionable, if not wrong, conclusions that could lead to systems that do not work as intended. In the worst case, such issues can jeopardize the credibility of work that employs ML methods. Employing ML methods comes with a range of technical and logistical challenges related to, for example, amount and quality of data available for model training and evaluation. To set up ML pipelines one would also need to have a solid basis of foundational statistics allowing not only for correct interpretation of evaluation results but also for effective training that can lead to generalizable and thus usable models. While these are important aspects, which I have touched upon throughout this article, my main focus here was on a higher-level discussion of pitfalls that one can fall into when *applying* ML methods, as I have seen it frequently. Based on years of experience working and teaching in the field, I have analyzed the most common problems that need to be addressed when employing ML methods. In an attempt to provide constructive guidelines on those common pitfalls and especially how to avoid them, I have formulated the following *seven postulates of applied machine learning*:

1. *Not everything is a machine learning problem.*
2. *If you reach 100%, think about the data.*
3. *There is no ground truth.*
4. *Field test, or it did not happen.*
5. *Proper reporting, or it did not happen.*
6. *Data rule, models serve.*
7. *There is no oracle.*

Researchers and practitioners are invited to use these postulates and their associated guidelines as checklist for their own work, but also when assessing the value of existing work. These guidelines were developed over years of working with students with ambitions and curiosity, and great creativity to develop novel methods and systems of interactive computing. They have proven effective for quality as well as for reality checks when ML systems are employed (or not) in a range of challenging real-world sensor data analysis scenarios.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Retrieved from http://tensorflow.org/.

[2] Gregory D. Abowd. 2012. What next, Ubicomp? Celebrating an intellectual disappearing act. In *Proceedings of the International Conference on Ubiquitous Computing (UbiComp'12)*.

[3] Sylvain Arlot, Alain Celisse, et al. 2010. A survey of cross-validation procedures for model selection. *Statist. Surv.* 4 (2010), 40–79.

[4] Horace B. Barlow. 1989. Unsupervised learning. *Neural Comput.* 1, 3 (1989), 295–311.

[5] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* 35, 8 (2013), 1798–1828.

[6] Christophe Biernacki, Gilles Celeux, and Gerard Govaert. 2003. Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate Gaussian mixture models. *Comput. Statist. Data Anal.* 41, 3–4 (2003), 561–575.

[7] Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

[8] David R. Brillinger. 2001. *Time Series: Data Analysis and Theory*. SIAM.
[9] A. J. Bernheim Brush. 2010. Ubiquitous computing field studies. In *Ubiquitous Computing Fundamentals*, John Krumm (Ed.). CRC Press, 161–202.
[10] Andreas Bulling, U. Blanke, and Bernt Schiele. 2014. A tutorial on human activity recognition using body-worn inertial sensors. *Comput. Surv.* 46, 3 (Jan. 2014).
[11] Rich Caruana and Alexandru Niculescu-Mizil. 2006. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*. 161–168.
[12] R. Chavarriaga, H. Sagha, and D. Roggen. 2013. The opportunity challenge: A benchmark database for on-body sensor-based activity recognition. *Pattern Recog. Lett.* 34, 15 (2013), 2033–2042.
[13] L. Chen, J. Hoey, C. D. Nugent, D. J. Cook, and Z. Yu. 2012. Sensor-based activity recognition. *IEEE Trans. Syst. Man Cyber. Part C: Applic. Rev.* 42, 6 (2012), 790–808.
[14] Charlie Chung. 2014. If You've Got Data, Mine It Yourself: Ian Witten on Data Mining, Weka, and his MOOC. Retrieved from https://www.class-central.com/report/data-mining-weka-waikato/.
[15] Gabe Cohn, Sidhant Gupta, Jon Froehlich, Eric C. Larson, and Shwetak N. Patel. 2010. GasSense - Appliance-level, single-point sensing of gas activity in the home. In *Proceedings of the International Conference on Pervasive Computing (Pervasive'10)*. LNCS, Vol 6030, 265–282.
[16] Janez Demšar, Tomaž Curk, Aleš Erjavec, Črt Gorup, Tomaž Hočevar, Mitar Milutinovič, Martin Možina, Matija Polajnar, Marko Toplak, Anže Starič, et al. 2013. Orange: Data mining toolbox in Python. *J. Mach. Learn. Res.* 14, 1 (2013), 2349–2353.
[17] Anind K. Dey. 2001. Understanding and using context. *Pers. Ubiq. Comput.* 5, 1 (2001), 4–7.
[18] Aiden Doherty, Dan Jackson, Nils Hammerla, Thomas Ploetz, Patrick Olivier, Malcolm H. Granat, Tom White, Vincent T. van Hees, Michael I. Trenell, Christoper G. Owen, Stephen J. Preece, Rob Gillions, Simon Sheard, Tim Peakman, Soren Brage, and Nicholas J. Wareham. 2017. Large scale population assessment of physical activity using wrist worn accelerometers: The UK biobank study. *PLoS One* 12, 2 (Feb. 2017).
[19] Pedro Domingos. 1999. The role of Occam's razor in knowledge discovery. *Data Mining Knowl. Discov.* 3, 4 (1999), 409–425.
[20] Pedro M. Domingos. 2012. A few useful things to know about machine learning. *Commun. ACM* 55, 10 (2012), 78.
[21] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. Retrieved from http://archive.ics.uci.edu/ml.
[22] Frank Eibe, Mark A. Hall, and Ian H. Witten. 2016. The WEKA workbench. Online appendix for data mining: Practical machine learning tools and techniques. In Morgan Kaufmann (4th ed.). Retrieved on May 27, 2021 from https://www.cs.waikato.ac.nz/ml/weka/Witten_et_al_2016_appendix.pdf.
[23] Katayoun Farrahi and Daniel Gatica-Perez. 2011. Discovering routines from large-scale human locations using probabilistic topic models. *ACM Trans. Intell. Syst. Technol.* 2, 1 (Jan. 2011), 1–27.
[24] Jon Froehlich, Eric C. Larson, Tim Campbell, Conor Haggerty, James Fogarty, and Shwetak N. Patel. 2009. HydroSense - Infrastructure-mediated single-point sensing of whole-home water activity. In *Proceedings of the International Conference on Ubiquitous Computing (UbiComp'09)*. 235–244.
[25] Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. 2018. Explaining explanations: An overview of interpretability of machine learning. In *Proceedings of the IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA'18)*. IEEE, 80–89.
[26] Yu Guan and Thomas Ploetz. 2017. Ensembles of Deep LSTM learners for activity recognition using wearables. *Proc. ACM Interact., Mob., Wear. Ubiq. Technol.* 1, 2 (June 2017).
[27] Nils Hammerla, James Fisher, Peter Andras, Lynn Rochester, Richard Walker, and Thomas Ploetz. 2015. PD disease state assessment in naturalistic environments using deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI'15)*.
[28] Nils Hammerla, Shane Halloran, and Thomas Ploetz. 2016. Deep, convolutional, and recurrent models for human activity recognition using wearables. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'16)*.
[29] Nils Hammerla, Reuben Kirkham, Peter Andras, and Thomas Ploetz. 2013. On preserving statistical characteristics of accelerometry data using their empirical cumulative distribution. In *Proceedings of the International Symposium on Wearable Computing (ISWC'13)*.
[30] Nils Y. Hammerla and Thomas Ploetz. 2015. Let's (not) stick together: Pairwise similarity biases cross-validation in activity recognition. In *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing*.
[31] Harish Haresamudram, David V. Anderson, and Thomas Plötz. 2019. On the role of features in human activity recognition. In *Proceedings of the International Symposium on Wearable Computing (ISWC'19)*. 78–88.
[32] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning*. Springer.
[33] Jeffrey Hightower and Gaetano Borriello. 2001. Location systems for ubiquitous computing. *IEEE Comput.* 34, 8 (2001), 57–66.

[34] T. Hill and P. Lewicki. 2007. *STATISTICS: Methods and Applications*. StatSoft, Tulsa, OK.
[35] Martin Höst, Björn Regnell, and Claes Wohlin. 2000. Using students as subjects – A comparative study of students and professionals in lead-time impact assessment. *Empir. Softw. Eng.* 5, 3 (2000), 201–214.
[36] Nitin Indurkhya and Fred J. Damerau. 2010. *Handbook of Natural Language Processing*. Vol. 2. CRC Press.
[37] Julie A. Jacko. 2012. *Human Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications*. CRC Press.
[38] A. K. Jain and B. Chandrasekaran. 1982. 39 Dimensionality and sample size considerations in pattern recognition practice. In *Classification Pattern Recognition and Reduction of Dimensionality*. Handbook of Statistics, Vol. 2. Elsevier, 835–855.
[39] Frederick Jelinek. 2005. Some of my best friends are linguists. *Lang. Resour. Eval.* 39, 1 (Feb. 2005), 25–34.
[40] Reuben Kirkham, Aftab Khan, Sourav Bhattacharya, Nils Hammerla, Sebastian Mellor, Daniel Roggen, and Thomas Ploetz. 2013. Automatic correction of annotation boundaries in activity datasets by class separation maximization. *Adjunct Proceedings of the International Conference Ubiquitous Computing (UbiComp'13) – HASCA workshop*.
[41] John Krumm. 2018. *Ubiquitous Computing Fundamentals*. CRC Press.
[42] John Krumm and Dany Rouhana. 2013. Placer: Semantic place Labels from diary data. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. 163–172.
[43] Hyeokhyen Kwon, Gregory D. Abowd, and Thomas Plötz. 2019. Handling annotation uncertainty in human activity recognition. In *Proceedings of the International Symposium on Wearable Computing (ISWC'19)*. 109–117.
[44] Hyeokhyen Kwon, Catherine Tong, Harish Haresamudram, Yan Gao, Gregory D. Abowd, Nicholas D. Lane, and Thomas Plötz. 2020. IMUTube: Automatic extraction of virtual on-body accelerometry from video for human activity recognition. *Proc. ACM Interact. Mob. Wear. Ubiq. Technol.* 4, 3 (Sept. 2020). DOI: https://doi.org/10.1145/3411841
[45] Nicholas D. Lane, Sourav Bhattacharya, Petko Georgiev, Claudio Forlivesi, Lei Jiao, Lorena Qendro, and Fahim Kawsar. 2016. DeepX: A software accelerator for low-power deep learning inference on mobile devices. In *Proceedings of the 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'16)*. IEEE, 1–12.
[46] H. Li, G. Abowd, and T. Ploetz. 2018. On specialized window lengths and detector based human activity recognition. In *Proceedings of the International Symposium on Wearable Computing (ISWC'18)*.
[47] Henar Martín, Ana M. Bernardos, Josué Iglesias, and José R. Casar. 2013. Activity logging using lightweight classification techniques in mobile devices. *Pers. Ubiq. Comput.* 17, 4 (2013), 675–695.
[48] A. H. Maslow. 1966. *The Psychology of Science: A Reconnaissance*. Harper & Row.
[49] MATLAB. 2010. *version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, MA. http://citebay.com/how-to-cite/matlab/.
[50] Mary L. McHugh. 2012. Interrater reliability: The kappa statistic. *Biochem. Med.* 22, 3 (Oct. 2012), 276–282.
[51] Rupert G. Miller. 1981. *Simultaneous Statistical Inference*. Springer.
[52] Tom M. Mitchell. 1997. *Machine Learning*. McGraw Hill.
[53] Tudor Miu, Paolo Missier, and Thomas Ploetz. 2015. Bootstrapping personalised human activity recognition models using online active learning. In *Proceedings of the IEEE International Conferences on Ubiquitous Computing & Communications (IUCC'15)*.
[54] Sendhil Mullainathan and Jann Spiess. 2017. Machine learning: An applied econometric approach. *J. Econ. Perspect.* 31, 2 (2017), 87–106.
[55] Robert Nishihara, Philipp Moritz, Stephanie Wang, Alexey Tumanov, William Paul, Johann Schleier-Smith, Richard Liaw, Mehrdad Niknami, Michael I. Jordan, and Ion Stoica. 2017. Real-time machine learning: The missing pieces. In *Proceedings of the 16th Workshop on Hot Topics in Operating Systems*. 106–110.
[56] Richard O. Duda, Peter E. Hart, and David G. Stork. 2001. *Pattern Classification* (2nd ed.). Wiley-Interscience.
[57] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. Retrieved from http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.
[58] Shwetak N. Patel, Thomas Robertson, Julie A. Kientz, Matthew S. Reynolds, and Gregory D. Abowd. 2007. At the flick of a switch – Detecting and classifying unique electrical events on the residential power line. In *Proceedings of the International Conference on Ubiquitous Computing (UbiComp'07)*. 271–288.
[59] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* 12 (2011), 2825–2830.
[60] T. Ploetz and Yu Guan. 2018. Deep learning for human activity recognition in mobile computing. *IEEE Comput.* 51, 5 (2018), 50–59.

[61] Thomas Ploetz, Nils Hammerla, and Patrick Olivier. 2011. Feature learning for activity recognition in ubiquitous computing. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'11)* (2011).
[62] David Martin Powers. 2011. Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. *J. Mach. Learn. Technol.* 2, 1 (Dec. 2011), 37–63.
[63] Carl Edward Rasmussen and Zoubin Ghahramani. 2001. Occam's razor. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*. 294–300.
[64] A. Reiss and D. Stricker. 2012. Introducing a new benchmarked dataset for activity monitoring. In *Proceedings of the International Symposium on Wearable Computing (ISWC'12)*.
[65] Nicu Sebe, Ira Cohen, Ashutosh Garg, and Thomas S. Huang. 2005. *Machine Learning in Computer Vision*. Vol. 29. Springer Science & Business Media.
[66] Philip Sedgwick and Nan Greenwood. 2015. Understanding the Hawthorne Effect. *BMJ* 351 (2015), h4672.
[67] Emilie Shireman, Douglas Steinley, and Michael J. Brusco. 2015. Examining the effect of initialization strategies on the performance of Gaussian mixture modeling. *Behav. Res. Meth.* 49, 1 (Dec. 2015), 282–293.
[68] Niall Twomey, Tom Diethe, Xenofon Fafoutis, Atis Elsts, Ryan McConville, Peter Flach, and Ian Craddock. 2018. A comprehensive study of activity recognition using accelerometers. *Informatics* 5, 2 (June 2018), 27–38.
[69] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. 2018. Deep learning for computer vision: A brief review. *Comput. Intell. Neurosci.* 2018, Article 7068349 (2018). https://doi.org/10.1155/2018/7068349.
[70] Kiri L. Wagstaff. 2012. Machine learning that matters. In *Proceedings of the International Conference on Machine Learning (ICML'12)*.
[71] Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni. 2020. Generalizing from a few examples: A survey on few-shot learning. *ACM Comput. Surv.* 53, 3 (2020), 1–34.
[72] Jamie A. Ward, Paul Lukowicz, and Hans W. Gellersen. 2011. Performance metrics for activity recognition. *ACM Trans. Intell. Syst. Technol.* 2, 1 (Jan. 2011), 1–23.
[73] M. Weiser. 1991. The computer for the 21st century. *Sci. Amer.* 265, 3 (1991).
[74] Ian H. Witten, Eibe Frank, Mark A. Hall, and Christopher J. Pal. 2016. *Data Mining: Practical Machine Learning Tools and Techniques*. Elsevier.
[75] F. Zafari, A. Gkelias, and Kin K. Leung. 2019. A survey of indoor localization systems and technologies. *IEEE Commun. Surv. Tutor.* 21, 3 (2019), 2568–2599.
[76] Alice Zheng. 2015. *Evaluating Machine Learning Models*. O'Reilly Media.
[77] Xiaojin Zhu and Andrew B. Goldberg. 2009. Introduction to semi-supervised learning. *Synth. Lect. Artif. Intell. Mach. Learn.* 3, 1 (2009), 1–130.
[78] Xiaojin Jerry Zhu. 2005. *Semi-supervised Learning Literature Survey*. Technical Report. University of Wisconsin-Madison Department of Computer Sciences.

Received October 2020; revised January 2021; accepted March 2021