Challenges in Using ML for Networking Research: How to Label If You Must

Yukhe Lavinia[†], Ramakrishnan Durairajan[†], Reza Rejaie[†], Walter Willinger[‡]

[†] University of Oregon, [‡] NIKSUN, Inc.

ABSTRACT

Leveraging innovations in Machine Learning (ML) research is of great current interest to researchers across the sciences, including networking research. However, using ML for networking poses challenging new problems that have been responsible for slowing the pace of innovation and the adoption of ML in the networking domain. Among the main problems are a well-known lack of data in general and representative data in particular, an overall inability to label data at scale, unknown data quality due to differences in data collection strategies, and data privacy issues that are unique to network data. Motivated by these challenges, we describe the design of Emerge[1], a novel framework to support efforts to d**E**mocratize the use of **ML** for n**E**two**R**kin**G** r**E**search. In particular, Emerge focuses on the problem of providing a low-cost, scalable, and high-quality methodology for labeling networking data. To illustrate the benefits of Emerge, we use publicly available network measurement datasets from Caida's Ark project and create and evaluate data labels for them in a programmable fashion.

CCS CONCEPTS

• **Networks → Network measurement**; • **Computing methodologies → Learning paradigms**;

KEYWORDS

Labeling network data at scale, Weak supervision

ACM Reference Format:

Yukhe Lavinia[†], Ramakrishnan Durairajan[†], Reza Rejaie[†], Walter Willinger[‡]. 2020. Challenges in Using ML for Networking Research: How to Label If You Must. In *Workshop on Network Meets AI ML (NetAI'20), August 14, 2020, Virtual Event, NY, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3405671.3405812

1 INTRODUCTION

Motivated by the recent success of ML in domains such as computer vision [34] and autonomous driving/vehicles [24], we are witnessing enormous interest in applying ML to an ever-wider range of problems in the networking domain (*e.g.*, network automation [35], self-driving networks [20]). However, unlike other domains, the networking area poses several immediate and serious challenges that have impeded the rapid adoption of ML and have been responsible for the slow pace of innovation in ML-based networking research. Among the key challenges are a commonly-acknowledged and much-maligned lack of readily available data, questions concerning the representativeness of data, a general inability to label networking data at scale, and the privacy-sensitive nature of the data obtained from real-world networks.

At the same time, the networking area is experiencing a growing "digital divide" where a select few research groups in the industry can leverage their global-scale production networks as rich data sources for developing and training their latest ML models while most academic networking researchers typically lack access to any type of real-world network data. Motivated in part by the success that ImageNet [17] has had in fueling and democratizing ML-based research in computer vision, [23] argues that academic networking researchers should start leveraging their campus networks and utilize them as rich data sources for their ML-based research efforts. Whether or not the envisioned wide-spread use of campus networks will ensure a more equal playing field for networking researchers and will, in turn, revolutionize ML-based networking remains to be seen, though.

However, there is more to democratizing ML-based networking research than overcoming the lack of available rich data. For example, even if the data problem has been solved (*e.g.*, using campus networks as data sources), researchers are immediately faced with another formidable and largely unsolved problem: a general paucity of *labeled* networking data. To illustrate, the networking domain lacks in general well-established and commonly-agreed upon features for accurately describing different events of interest (*e.g.*, an onset of volumetric DDoS attack). This "fuzzy" nature of network data is further aggravated by the fact that the data is generally collected and curated in a highly piecemeal manner; *i.e.*, at different granularities (*e.g.*, packets vs. flows vs. logs) and locations (*e.g.*, edge vs. core), under different conditions (*e.g.*, congested vs. uncongested links), or with varying semantic information (*e.g.*, payloads vs. headers).

Unlike ImageNet where crowd-sourcing has been effectively leveraged to create a large database of hand-annotated images, network data is, in general, more complex than typical dog or cat pictures. That is, its correct interpretation or labeling often requires substantial domain knowledge (*e.g.*, protocols, configurations, policies) which in turn rules out the use of popular crowd-sourcing methods (as in the case of ImageNet) or more recently pursued out-sourcing efforts (as in the case of data used for autonomous

[1]To facilitate independent validations of our results and to catalyze community-based efforts, the source code of Emerge can be found at https://gitlab.com/onrg/emerge.

NetAI'20, August 14, 2020, Virtual Event, NY, USA

ACM ISBN 978-1-4503-8043-0/20/08…$15.00
https://doi.org/10.1145/3405671.3405812

driving) as low-cost, scalable and high-quality labeling approaches. Further, the networking area is unique with respect to the difficulties caused by privacy aspects associated with most network data. This aspect by and large rules out the sharing of raw or labeled data with third-party researchers and also limits the sharing of other ML research artifacts such as learning models due to possible privacy leaks [22, 40, 51]. In short, while attempts to democratize the use of ML in other areas have been wildly successful, they have been largely futile in the networking domain. To ensure that networking can similarly benefit from efforts where its use of ML becomes a collaborative and community-driven activity, new ideas are needed to specifically address the unique nature of network data (§ 2).

In this paper, we take a pro-active perspective. Assuming that researchers have ready access to rich network data (*e.g.*, from campus networks), we focus on the problem of how to provide them with a low-cost, scalable, and high-quality methodology for labeling their data. In particular, we describe the design of a framework called Emerge (§ 3) that seeks to extend the idea of the weak supervision-based data labeling technique proposed in prior work [32] and supports collaborative efforts by creating data labels at scale, of good quality (*e.g.*, dealing with bias in the data), at low cost (*e.g.*, supporting a community-based effort), and, if necessary, in ways that respect prevailing data privacy concerns.

We demonstrate the efficacy of Emerge using `traceroute` data from Caida's Ark project and creating data labels in a programmable fashion (§ 3.2). We start by performing exploratory analysis of the data to establish a threshold by which we distinguish good data from noise. We then split the data into training, validation, and test sets, and label ~20% of the overall data points for our validation and test sets. For feature extractions, we use the `tsfresh` module [16] and then generate feature combinations to train the automatic heuristic generation component of Emerge to produce probabilistic training labels. Subsequently, we train discriminative models (*e.g.*, Long Short Term Memory (LSTM)) using these probabilistic labels. Furthermore, in an effort to turn data labeling at scale into a genuine community-based effort, we consider steps to facilitate collaboration among third-party researchers working at the intersection of ML and networking by creating mechanisms to share relevant information (*i.e.*, metadata rather than raw data or ML models) in a privacy-preserving fashion. We also outline concrete design steps for dealing with bias in the data.

2 MOTIVATION AND RELATED WORK

In this section, we motivate our work with use cases and outline the prior efforts and their limitations.

2.1 Motivating Use cases

Our work is motivated by three concrete use cases that illustrate the range of problems that the pursuit of efforts to democratize the use of ML for networking entails.

2.1.1 **Addressing the Paucity of Labels.** Assuming a scenario where networking researchers have ready access to rich data, much of ML relies critically on the availability of large quantities of ground truth and/or high-quality labeled data to ensure that the developed learning models perform well "in the wild". This crucial role of labeled data is perhaps best illustrated by the following recent quote (slightly paraphrasing) in [5]:

> If data is "the new oil", then you don't want to be in the crude oil business but in the refinery business (*e.g.*, labeling).

While significant progress has been made in the recent past in creating "benchmarks" in computer vision (*e.g.*, ImageNet [17]) and autonomous driving (*e.g.*, Argoverse [2], nuScenes [6]), we posit that the data in these domains is significantly different from network data characteristics and properties. Some initial "benchmarking" attempts in the networking domain notwithstanding (see, for *e.g.*, [49]), a main reason why network data is unique and defies direct comparisons with data from other domains is its *fuzzy nature*. That is, objects in, say, the ImageNet database have key features that are specific to those objects. For example, to differentiate a cat from a dog using ML, a classifier can be trained by (a) using 1000s of labeled pictures of cats and dogs and (b) leveraging cat- and dog-specific features (*e.g.*, length of the animal's snout, shape of the animal's pupils).

Unfortunately, network data, by and large, lacks such specificity which in turn severely limits the researchers' ability to develop suitable learning models. For example, there are no standard or commonly agreed upon "k-best" features from NetFlow (or other types of network) data for accurately describing network events of interest such as a volumetric DDoS attack. While as a community, networking researchers have spent significant efforts on collecting and curating datasets [3, 4, 8], they have in general paid little to no attention to generating associated relevant metadata [41] or augmenting their data with high-quality data labels [32]. Importantly, the networking community as a whole currently lacks well-established and widely agreed-upon approaches for identifying or computing relevant feature sets for different network events of interest and including this or similar information in the datasets collected.

Complicating matters further is the largely uncoordinated nature of much of networking research and the already mentioned predominantly piecemeal approach to data collection. As a result, in the case of, say, network traffic measurements, the types of questions that can be answered using commonly-collected traffic data differ significantly based on the where (*e.g.*, vantage points), how (*e.g.*, granularity), when (*e.g.*, network conditions), or what (*e.g.*, semantics) of the data collection. Moreover, a number of real-world constraints (*e.g.*, competitive reasons, privacy concerns, business incentives) obviate most forms of data and/or label sharing.

Requirement 1. The networking domain is in dire need of solutions that (a) can identify and establish a standard set of relevant features for different events of interest, and (b) provide a framework for affordable and high-quality labeling of different types of network data at scale. Due to the fuzzy nature of network data, a key prerequisite to succeed in these efforts is to tap into and effectively leverage the networking community's vast amount of available domain knowledge.

2.1.2 **Supporting Privacy-preserving Collaborations.** The combination of a mostly piecemeal approach to solving networking problems, a growing chasm between the "haves" and "have-nots" [23] among networking researchers, and increasingly stringent policies for ensuring data privacy means that *sharing of raw data* among researchers in different universities or companies as the most straightforward approach to democratizing the use of

ML for networking is largely off-limits and not feasible in practice. While there has been great progress in privacy-preserving data sharing (*e.g.*, see [12, 18] and references), it is unclear how widely adopted the proposed solutions are and how they perform when faced with data from real-world production networks.

An obvious alternative to sharing of raw data is *sharing of ML models*. To illustrate, consider two researchers (A and B) working on two versions of the same network data. Assume A's version is more detailed (*e.g.*, proprietary version of the company's data) than B's version which was collected from a third-party perspective (*e.g.*, has fewer details and can be shared). Note that the propriety nature of A's data means that it cannot be shared with B "as is". In theory, this data sharing constraint notwithstanding, A and B can still collaborate; they both can train their own models with the data available to them, share the resulting models, and independently validate each other's model as well as try to explain observed differences between the two models. The problem is succinctly captured by the following quote extracted from [7]:

> The increasing utility of data from machine learning has a negative effect on privacy too. Disparate pieces of information – although individually of limited utility – become significant when combined with other types of information.

Unfortunately, there are several reasons why ML models cannot preserve data privacy in collaborations. First, several state-of-the-art ML techniques (*e.g.*, deep learning) are "black box" in nature. That is, while the input and the output (*e.g.*, decisions) from the ML model are known, it is unclear how the complex functions entrenched in the different layers arrive at the final mapping of the model's input to its output. Second, since the models are black box in nature, adversaries can craft model inversion attacks to leak private information [22, 40, 51] or launch re-identification attacks to de-anonymize sensitive data [19]. Third, most of the currently considered approaches to explain "black box" ML models are not applicable because they not only strive to explain the unknown mapping but also tend to produce explanations that are either not reliable or can be misleading [39].

We are thus left with one last alternative for enabling privacy-preserving collaborations among networking researchers using ML, namely *sharing of learning algorithms* via a "glass box" framework. That is, sharing only the developed learning algorithm but neither the available (labeled) data nor the resulting learning model, which makes this approach both feasible in practice and safe in theory (*e.g.*, no leak of private information). In fact, this approach promises that in the previously described setting, researchers A and B can be required to understand the impact of the quality of their data (and labels) on the accuracy of their respective predictions and incrementally improve the quality of their labels using the information (*e.g.*, metadata but no raw data) that they share. We argue that only a "glass box" mechanism can solve such unique collaboration requirements.

Requirement 2. To facilitate low-cost label creation, enable privacy-preserving collaborations among the third-party researchers, and support independent validation of separate research endeavors, the networking community is in need for a "glass box" framework so that learning algorithms and decision heuristics can be broadly shared *without* requiring any sharing of raw data or ML models.

2.1.3 **Dealing with Hidden Biases in the Data.** The following quote from [1] articulates the crux of the problem:

> Machine learning can actually amplify bias, and you can never be done checking for bias.

The combination of a lack of training labels—with both positive and negative examples—and the fuzzy nature of network data also results in non-representative data with known and unknown bias. For example, different network datasets collected by prior efforts are known to exhibit location-specific and sampling bias (*e.g.*, see [46, 50]), but quantifying these or other hidden biases is an open problem. Moreover, the piecemeal approach followed by the different research groups in collecting network data means that training data is in general created with a siloed perspective and does not accommodate diverse (positive and negative) examples.

Complicating the situation is the real-world nature of operational networks with their own outage and failure dynamics, an ever-increasing number of old and new attacks, and constantly-changing bandwidth needs of their users. These challenges posed by real-world production networks rarely align with the complex functions of the proposed "black box" models and often result in biased models [1]. It is not that researchers knowingly create biased ML models, but their models typically work only as intended when evaluated in the controlled settings (*e.g.*, in the lab) in which these models were developed in the first place.

Requirement 3. To create high-quality data labels at scale so as to ensure the development of new ML models that perform well in practice, the networking community has to design new mechanisms for dealing with (*i.e.*, identifying, quantifying, and correcting) hidden data biases and watching out for the emergence of novel types of biases in their data.

2.2 Prior Efforts and their Limitations

Applying supervised learning to networking problems has been of interest to the networking community for more than a decade (see surveys [33, 47] and references therein). These techniques construct predictive models by learning from a large number of training examples (*i.e.*, labeled data) but as discussed earlier, the networking domain lacks in general access to the necessary training data. Similarly, several efforts use unsupervised learning [43] to *e.g.*, detect anomalies in BGP [25], perform network traffic prediction and diagnosis [14, 26–28], or carry out event detection [42]. However, the problem with unsupervised learning is that not all types of clustering techniques are suitable for identifying events of interest in networking data. Among the main reasons for this shortcoming is the lack of a rigid mathematical definition for outliers and the data's fuzzy nature.

NoMoNoise [32], a recently proposed framework for denoising latency measurements, is most closely related to our work. It relies on two key ideas to tackle both the data labeling and data fuzziness problems. The first idea is to use weak supervision to combine and learn noisy labels from many weak sources to build a predictive model. Popular forms of weak supervision include distant supervision [13, 31] and crowd-sourcing with non-expert annotators [38, 48]. NoMoNoise models the joint distribution $P(x, l)$ of

features from the Internet measurement data x and labels l (*i.e.*, generative modeling) to describe the noise of the labeling functions and increase the accuracies of the labels, *without* access to the true labels. The second idea is that NoMoNoise opens up new possibilities in enhancing the utility of networking data by leveraging the benefits of both Snorkel [36, 37] and a data programming paradigm where users can programmatically create lower-quality training data via simple labeling functions or heuristics (*e.g.*, latencies above $\mu + 2\sigma$ denotes an outage) to address the data fuzziness problem.

While NoMoNoise is a first step towards democratizing the use of ML for Internet measurements, it is limited in terms of scale (*i.e.*, number of measurements) and is focused only on delay measurements. Further, NoMoNoise lacks capabilities to (a) process and label diverse networking data at scale, (b) identify and remove bias lurking in the data, and (c) facilitate collaboration among third-party researchers via sharing of privacy-preserved metadata and learning algorithms.

3 EMERGE: A FRAMEWORK TO SUPPORT ML FOR NETWORKING

To tackle the challenges outlined in § 2, we describe in this section the design of Emerge that seeks to support efforts to d**E**mocratize the use of **M**L for n**E**two**R**kin**G** r**E**search.

3.1 Design of EMERGE

At its core, the Emerge framework is an assembly of existing systems and consists of three main modules: (a) a module to create data labels at scale, extending ideas from the NoMoNoise framework [32]; (b) a module to ensure good quality labels (*e.g.*, dealing with bias in the data); and (c) a module to facilitate low cost (*e.g.*, community-based) labeling and sharing effort (in ways that respect prevailing data privacy concerns, if needed). In this work, we build an initial prototype of the first module and outline the design steps for the other two modules, leaving their implementation and evaluation as future work. In particular, assuming that users of Emerge (*e.g.*, researchers) have ready access to rich network data, we focus in the following on the problem of how to provide them with a low-cost, scalable and high-quality methodology for labeling their data.

The Emerge pipeline shown in Figure 1 includes Snuba [45], a component for automatic heuristics generation based on a set of unlabeled data, a set of labeled data, and extracted features from both sets. It then generates feature combinations and fits a simple classifier model (*e.g.*, logistic regressor, nearest neighbors, or decision tree) to each feature combination. Each classifier model will then assign probabilistic labels to different portions of the unlabeled data. With this approach, the simple classifiers are able to exploit different characteristics of the data and assign labels accordingly, thus alleviating the burden on users to have to write their own labeling functions. This approach also addresses the scalability issue in NoMoNoise, since writing labeling functions requires searching for the patterns that describe events of interest, and these patterns are different for different events of interest and different networking data types. As the quantity and diversity of the data increase, the task of studying the data to find the pattern could become a labor-intensive endeavor. The simple classifiers in Snuba substitute the human reasoning involved in searching for these patterns and assign labels based on these learned patterns.

The Snuba component is implemented without modification and is independent of NoMoNoise as Snuba can generate its own probabilistic labels. However, users can opt to connect Snuba to NoMoNoise's generative model to produce labels. Also, note that since Snuba requires its classifiers to learn from feature combination matrices, we employed `tsfresh` [16], a time series feature calculation tool, to obtain the features needed for Snuba. With the obtained probabilistic labels, Emerge users can train a classifier model such as LSTM to detect events of interest (*e.g.*, noise, anomaly, etc.).

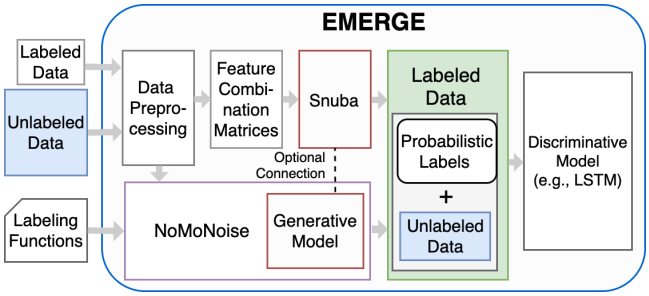

Figure 1: EMERGE pipeline.

3.2 Early Promise of EMERGE

Dataset. We use one day's worth of traceroute measurements from Caida's Ark project [10]. The data consists of 1,248,858 traceroutes. We use scamper's `warts2csv` tool [9, 30] to convert the data into CSV format and extract the round-trip time (RTT) values between 28 source-destination (SD) pairs, amounting to 75,359 total RTT measurements. Each SD pair contains between 2,000 and 4,000 data points.

3.2.1 **Labeling Data at Scale.** We present our preliminary results on labeling data at scale using Emerge.

Experiment. Our goal is to compare the accuracy of Emerge against the accuracy achieved by using different naïve methods that we describe in more detail below. More precisely, for each SD pair dataset, we perform a manual exploratory analysis to establish a threshold value with which we label the RTTs as noise (-1) or good (+1). Since the number of noisy data is almost always smaller than the good data, we over-sample the minority (noisy) class to balance the number of samples by generating random values that are greater than the threshold and adding them to the original data. This way we ensure that our classifier component has abundant good and bad examples to learn from.

To prepare the input for Snuba, we use `tsfresh` to calculate the statistical features of the data. We choose eight features: length, maximum, minimum, mean, median, standard deviation, sum, and variance. Next, for each SD pair dataset, we partition the data into training, validation, and test sets. We take 500 data points to be our test set and then take 20% of the remaining data to be our validation set. The rest becomes our training set. The validation and test sets become our labeled data while the train set becomes our unlabeled data. Our generative model component will produce probabilistic

training labels and these are the labels we use in our discriminative model component. Since the temporal order of the RTT values in our data matters, we choose LSTM as our discriminative model.

When we compare the discriminative model accuracy of EMERGE against the accuracy of the different considered naïve methods, we use the same training set to learn the LSTM model and test it on the validation and test sets. We fine-tuned our LSTM hyperparameters by trying out various numbers of epochs, batch sizes, number of LSTM units, and learning rates. We apply L2, dropout, and early stopping regularization to avoid overfitting (see Appendix B). While EMERGE uses our generated probabilistic training labels, the considered naïve methods employ a variety of different heuristics (*e.g.*, μ, $\mu + 1\sigma$, $\mu + 2\sigma$, $\mu + 3\sigma$, KMeans, local outlier factor (LOF), elliptic envelope (EE), overly robust covariance estimation (ORCE), and isolation forest (IF)) to assign labels to the training set. That is, our naïve method 1 uses μ as threshold to label data, naïve method 2 uses KMeans to label data, etc. Note that we establish the threshold value for each considered naïve method *before* we augment the data. We do so because adding synthetic noise will alter the statistical characteristics of the dataset that the naïve methods depend on.

We use F1 scores as our evaluation metrics. The reason for using this metric is that we are mainly interested in measuring how well EMERGE performs in (a) identifying actual good measurement data (low false positives, thus high precision), and (b) recognizing as many good measurement data as possible (low false negative, thus high recall). Finally, for each SD pair, we compare the performance of our method against nine different naïve methods and average the obtained F1 scores across the 28 SD pair datasets.

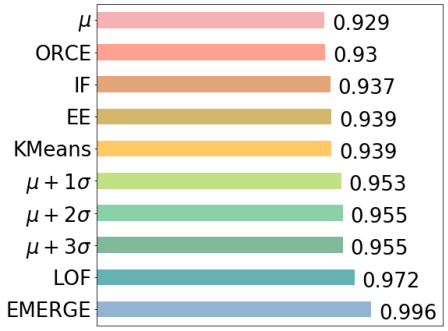

Figure 2: Average F1 scores of discriminative models trained using EMERGE and naive methods.

Results. Figure 2 depicts the average F1 scores produced by the LSTM models trained using the different naïve labeling methods and the probabilistic labels generated by EMERGE. With an F1 score of 0.996, EMERGE outperforms all the considered naïve methods and is successful in distinguishing good and noisy measurement data.

Among the naïve labeling methods, μ results in the lowest F1 score of 0.929, while LOF achieves the highest score with 0.972. The μ method's lowest F1 score comes as no surprise. Given that the recall score captures the number of false negatives, which in our case corresponds to the number of RTT values that are falsely considered as noise, using μ as threshold to determine noisy measurements leads to a high number of (falsely) predicted noisy data. As the value of false negative increases, the recall score decreases, leading to a lower F1 score. On the other hand, LOF proves to be the best naïve technique to separate good data from noisy data. By considering the local density of data points and comparing this density to that of their neighbors, LOF can detect data points that are "obviously different", which may fit the characteristics of actual noisy data points. EMERGE, however, still outperforms LOF. By employing different classifiers to learn from the various feature combinations of the data, EMERGE is able to learn from different segments of the data that may have different noise characteristics. With this approach, EMERGE avoids making the mistake of applying a heuristic that may have been based on previous assumptions or bias in the data.

3.2.2 **Supporting Privacy-preserving Collaborations.** The twin goals of this experiment are to (a) facilitate sharing of information among researchers in a privacy-preserving fashion and (b) reduce the cost of labeling by pooling available resources. Importantly, EMERGE guarantees privacy: only the metadata that provide sufficiently detailed descriptions of the data (*e.g.*, labeling functions and heuristics) but no raw data, and no ML models are shared in the process of enhancing the accuracy of labeling.

Experiment. We use the same 28 SD pairs considered earlier to conduct our experiment. Using the NoMoNoise component in EMERGE, we write labeling functions using different heuristics and combine the heuristics to see if combining them will boost the F1 score. The goal is to show the possible benefits of combining labeling functions written by different research groups. To illustrate, we write labeling functions using some of the familiar heuristics listed in the previous experiment to generate probabilistic labels, train an LSTM model using these labels, and average the F1 scores. Our LSTM hyperparameter settings are: 0.001 learning rate, 100 units, 20 epochs, 0.25 dropout rate, and a batch size of 128.

Results. Tables 1, 2, and 3 show the average F1 scores of several labeling functions and their combinations. From Tables 1 and 2, we see that combining EE with LOF and $\mu + 2\sigma$ with EE increases the score compared to using the different naïve methods individually. However, combining μ with EE or with LOF results in scores that are lower than those obtained when using EE and LOF individually. Similarly, when combining $\mu + 2\sigma$ with LOF, the score is 0.810 compared to 0.836 which is the score from using only LOF. Also, combining $\mu + 2\sigma$ and EE & LOF is shown to decrease EE & LOF's score by 0.052. Interestingly, when we combine $\mu + 2\sigma$ and EE, we see an increase of their respective F1 scores, and combining μ, μ+2σ, EE, and LOF boosts the F1 score to 0.914.

μ	EE	LOF	$\mu + 2\sigma$
0.637	0.759	0.836	0.721

Table 1: Average F1 scores (one labeling function).

The fact that one labeling function's involvement in a combination of different labeling functions decreases the F1 score of the combination may indicate certain assumptions that may not fit the characteristics of the data and can lead to an increase in either false positives or false negatives. As discussed earlier, using only μ to distinguish good from noisy data can lead to a high number of falsely identified noisy data, which negatively affect the F1 score. This is also accentuated in the results shown in Table 3. There, μ's involvement in the EE & LOF combination also drags down EE & LOF's combined F1 score of 0.890 to 0.733.

Another possible reason for F1 score decreases when combining labeling functions is the number of overlapping labeling functions in different segments of the data. In such overlapping segments, multiple labeling functions may disagree with each other. Although learning from the agreement/disagreement of these labeling functions has been shown to be critical in NoMoNoise [32], it is important to note that the *number* of labeling functions involved plays an important role. In many machine learning settings, a classifier benefits from a large number of features as these features provide ample opportunities for learning. Correspondingly, since Emerge learns from the agreement and disagreement of the labeling functions, the larger the number of labeling functions, the more learning opportunities exist. This can also explain the increase of the F1 score when four labeling functions are combined (see Table 3). In this example, Emerge may have learned from more labeling functions and was able to correct the mistakes it made when it is only provided with only three labeling functions.

μ & EE	μ & LOF	EE & LOF	$\mu + 2\sigma$ & EE	$\mu + 2\sigma$ & LOF
0.720	0.821	0.890	0.826	0.810

Table 2: Average F1 scores (two labeling functions).

Additionally, labeling function combinations that boost the functions' individual scores may indicate better coverage of the possibly diverse noise characteristics in the data. A dataset may contain subsets that exhibit noise characteristics that are unique to those subsets and thus require a different approach for separating good data from noise. The increasingly higher F1 scores we obtain as we add more labeling functions demonstrate a promising scenario where collaborative efforts between different research groups, each contributing with its own set of labeling functions, will (a) improve the quality of data labeling, (b) reduce the cost of labeling, and (c) ensure data privacy.

μ & EE & LOF	$\mu + 2\sigma$ & EE & LOF	μ & $\mu + 2\sigma$ & EE & LOF
0.733	0.838	0.914

Table 3: Average F1 scores (3+ labeling functions).

3.3 Dealing with Hidden Bias in the Data

Motivated by [44], we propose to extend Emerge with multi-task learning (MTL) to tackle hidden bias in the data via information sharing between several tasks [15]. Key to MTL is the idea of *implicit data augmentation* which increases sample size for better model generalization, which in turn improves bias reduction [11]. While single-task models typically perform well when applied to the specific problem for which they were designed for (*e.g.*, outage detection), they often discard information that is not relevant for the problem at hand but may be potentially useful for other models (*e.g.*, anomaly detection, noise detection) that are trained on the same dataset. This drawback of single-task models suggests that MTL can better generalize than several models, each trained to perform its own task independently from the other tasks/models. Further, the single-task modeling approach requires the training process to be repeated for each task. Since this results in the multiplication of training overhead by the number of tasks [29], MTL has the potential to greatly reduce training time (unless training can be done in parallel) while maintaining or even improving overall accuracy. We leave the implementation of MTL for future work.

4 DISCUSSION

This paper barely scratches the surface of the challenges that arise from the use of ML for networking. For example, creating an end-to-end pipeline of Emerge and turning it into an effective collaborative tool (*e.g.*, in the spirit of Binder [21]) requires innovative new ideas about describing network events in terms of commonly agreed-upon features or quantifying the "strength" of different data labeling heuristics. However, the future success of ML for networking will depend on more than just ensuring ubiquitous access to voluminous amounts of labeled data. In particular, to be relevant in practice, ML for networking will have to develop solutions that are unbiased (*e.g.*, "fairness") and robust (*i.e.*, "safety") and that network operators can trust (*e.g.*, "explainability"). Frameworks such as Emerge that enable the reproducibility of relevant ML research artifacts will be key to succeeding in this endeavor.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their insightful feedback. This work is supported by NSF CNS 1850297 award. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF.

REFERENCES

[1] Analyzing preventing unconscious bias in machine learning. https://www.infoq.com/presentations/unconscious-bias-machine-learning/.
[2] Argoverse. https://www.argoverse.org/.
[3] CAIDA Ark Datasets. http://www.caida.org/projects/ark/topo_datasets.xml.
[4] CRAWDAD Datasets. https://crawdad.org/.
[5] If data is the new oil, these companies are the new baker hughes. https://fortune.com/2020/02/04/artificial-intelligence-data-labeling-labelbox/.
[6] nuscenes. https://www.nuscenes.org/.
[7] Privacy-preserving machine learning 2018: A year in review. https://medium.com/dropoutlabs/privacy-preserving-machine-learning-2018-a-year-in-review-b6345a95ae0f.
[8] RIPE Atlas. https://atlas.ripe.net, 2018.
[9] Sc_warts2csv, Mar. 2018.
[10] The ipv4 routed /24 topology dataset, Nov. 2019.
[11] Baxter, J. A model of inductive bias learning. *Journal of artificial intelligence research 12* (2000), 149–198.
[12] Bhumiratana, B., and Bishop, M. Privacy aware data sharing: balancing the usability and privacy of datasets. In *Proceedings of the 2nd International Conference on PErvasive Technologies Related to Assistive Environments* (2009), pp. 1–8.
[13] Bunescu, R., and Mooney, R. Learning to extract relations from the web using minimal supervision. In *ACL* (2007).
[14] Camacho, J., Pérez-Villegas, A., García-Teodoro, P., and Maciá-Fernández, G. PCA-based Multivariate Statistical Network Monitoring for Anomaly Detection. *Computers & Security* (2016).
[15] Caruana, R. Multitask learning. *Machine learning 28*, 1 (1997), 41–75.
[16] Christ, M., Braun, N., Neuffer, J., and Kempa-Liehr, A. W. Time series feature extraction on basis of scalable hypothesis tests (tsfresh – a python package). *Neurocomputing 307* (2018), 72 – 77.
[17] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (2009), Ieee, pp. 248–255.
[18] Dunning, L. A., and Kresman, R. Privacy preserving data sharing with anonymous id assignment. *IEEE Transactions on Information Forensics and Security 8*, 2 (2012), 402–413.
[19] El Emam, K., Jonker, E., and Luk Arbuckle, B. M. A systematic review of re-identification attacks on health data. *PloS one 6*, 12 (2011).
[20] Feamster, N., and Rexford, J. Why (and how) networks should run themselves. *arXiv preprint arXiv:1710.11583* (2017).
[21] Forde, J., Bussonnier, M., Fortin, F.-A., Granger, B., Head, T., Holdgraf, C., Ivanov, P., Kelley, K., Pacer, M., Panda, Y., et al. Reproducing machine learning research on binder. In *NIPS Workshop on Machine Learning Open Source Software* (2018).
[22] Fredrikson, M., Jha, S., and Ristenpart, T. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the*

22nd ACM SIGSAC Conference on Computer and Communications Security (2015), pp. 1322–1333.

[23] Gupta, A., Mac-Stoker, C., and Willinger, W. An effort to democratize networking research in the era of ai/ml. In *Proceedings of the 18th ACM Workshop on Hot Topics in Networks* (2019), pp. 93–100.

[24] Janai, J., Güney, F., Behl, A., and Geiger, A. Computer Vision for Autonomous Vehicles: Problems, Datasets and State of the Art. *arXiv e-prints* (2017).

[25] K. Xu and J. Chandrashekar and Z.L. Zhang. A First Step toward Understanding Inter-domain Routing Dynamics. In *ACM SIGCOMM workshop on Mining network data* (2005).

[26] Lakhina, A., Crovella, M., and Diot, C. Diagnosing Network-wide Traffic Anomalies. In *ACM SIGCOMM* (2004).

[27] Lakhina, A., Crovella, M., and Diot, C. Mining Anomalies Using Traffic Feature Distributions. *ACM SIGCOMM* (2005).

[28] Li, X., Bian, F., Zhang, H., Diot, C., Govindan, R., Hong, W., and Iannaccone, G. MIND: A Distributed Multi-Dimensional Indexing System for Network Diagnosis. In *IEEE INFOCOM* (2006).

[29] Liu, T., Alibhai, S., Wang, J., Liu, Q., He, X., and Wu, C. Exploring transfer learning to reduce training overhead of hpc data in machine learning. In *2019 IEEE International Conference on Networking, Architecture and Storage (NAS)* (2019), IEEE, pp. 1–7.

[30] Luckie, M. Scamper: a scalable and extensible packet prober for active measurement of the internet. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement* (2010), pp. 239–245.

[31] Moore, A. W., and Zuev, D. Internet traffic classification using bayesian analysis techniques. In *ACM SIGMETRICS* (2005).

[32] Muthukumar, A., and Durairajan, R. Denoising internet delay measurements using weak supervision. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)* (2019), IEEE, pp. 479–484.

[33] Nguyen, T. T., and Armitage, G. A survey of techniques for internet traffic classification using machine learning. *IEEE Communications Surveys & Tutorials 10*, 4, 56–76.

[34] Nixon, M., and Aguado, A. *Feature extraction and image processing for computer vision.* Academic Press, 2019.

[35] Rafique, D., and Velasco, L. Machine learning for network automation: Overview, architecture, and applications [invited tutorial]. *Journal of Optical Communications and Networking 10*, 10 (2018), D126–D143.

[36] Ratner, A., Bach, S. H., Ehrenberg, H., Fries, J., Wu, S., and Ré, C. Snorkel: Rapid training data creation with weak supervision. *VLDB Endowment* (2017).

[37] Ratner, A. J., De Sa, C. M., Wu, S., Selsam, D., and Ré, C. Data programming: Creating large training sets, quickly. In *Advances in neural information processing systems* (2016), pp. 3567–3575.

[38] Rekatsinas, T., Chu, X., Ilyas, I. F., and Ré, C. Holoclean: Holistic data repairs with probabilistic inference. *VLDB Endowment* (2017).

[39] Rudin, C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence 1*, 5 (2019), 206–215.

[40] Shokri, R., Stronati, M., Song, C., and Shmatikov, V. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)* (2017), IEEE, pp. 3–18.

[41] Sommers, J., Durairajan, R., and Barford, P. Automatic metadata generation for active measurement. In *ACM IMC* (2017).

[42] Syamkumar, M., Mani, S. K., Durairajan, R., Barford, P., and Sommers, J. Wrinkles in Time: Detecting Internet-wide Events via NTP. In *proceedings of IFIP Networking* (2018).

[43] Usama, M., Qadir, J., Raza, A., Arif, H., Yau, K.-L. A., Elkhatib, Y., Hussain, A., and Al-Fuqaha, A. Unsupervised machine learning for networking: Techniques, applications and research challenges. *IEEE Access 7* (2019), 65579–65615.

[44] Vaidya, A., Mai, F., and Ning, Y. Empirical analysis of multi-task learning for reducing model bias in toxic comment detection. *arXiv preprint arXiv:1909.09758* (2019).

[45] Varma, P., and Ré, C. Snuba: Automating weak supervision to label training data. *Proc. VLDB Endow. 12*, 3 (Nov. 2018), 223–236.

[46] Weidmann, N. B., Benitez-Baleato, S., Hunziker, P., Glatz, E., and Dimitropoulos, X. Digital discrimination: Political bias in internet service provision across ethnic groups. *Science 353*, 6304 (2016), 1151–1155.

[47] Williams, N., Zander, S., and Armitage, G. A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification. *ACM SIGCOMM CCR* (2006).

[48] Yuen, M.-C., King, I., and Leung, K.-S. A survey of crowdsourcing systems. In *IEEE SocialCom* (2011).

[49] Zerwas, J., Kalmbach, P., Henkel, L., Rétvári, G., Kellerer, W., Blenk, A., and Schmid, S. Netboa: Self-driving network benchmarking. In *Proceedings of the 2019 Workshop on Network Meets AI & ML* (2019), pp. 8–14.

[50] Zhang, B., Iosup, A., Pouwelse, J., Epema, D., and Sips, H. Sampling bias in bittorrent measurements. In *European Conference on Parallel Processing* (2010), Springer, pp. 484–496.

[51] Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530* (2016).

A RESEARCH REPRODUCIBILITY

The source code of Emerge can be found at https://gitlab.com/onrg/emerge.

B OVERFITTING AND PERFORMANCE

To avoid the issue of overfitting, we take the following steps.

In the first experiment where we compare the performance of the naïve methods and our Emerge models, we fine-tune the following hyperparameters: batch size (16, 32, 64, 128, or 256), number of epochs (5, 10, 20, 25, or 30), number of LSTM units (32, 64, 128), and learning rate that we apply on Adam optimizer. To prevent overfitting, we apply L2 regularization, dropout, and early stopping. We set our early stopping regularization to monitor the validation loss. When the model finds that the decrease of validation loss is less than 0.01, it considers the change as no improvement and waits for another 5 epochs before stopping. Note that we fine-tune our models for each naïvely labeled train set and Emerge labeled train set per each SD pair. We note that the hyperparameter values are the same in some SD pairs while different in others.

We train our models and test their performance on the validation set. After fine-tuning the model to avoid overfitting, we select the hyperparameters used for the models that generate the best F1 score on the validation set to be the hyperparameters for our models that will be tested on the test set. On i5-8279U CPU, the execution time for the training and testing (either on validation or test set) is about 45-60 seconds depending on the number of epochs, batch size for each heuristic per one SD pair dataset, and thus about 9-10 minutes for all nine naïve models and our Emerge model for each SD pair dataset.

In summary, *these results demonstrate the fact that* Emerge*—in addition to facilitating privacy-preserving collaborations—can provide good-quality labels at scale (for tens or hundreds of thousands of data points) within a reasonable period of time.*