Learned Data Structures for Per-Flow Measurements

Andrea Monterubbiano
University of Rome - Sapienza
Rome, Italy

Raphael Azorin
Huawei Technologies, France
Paris, France

Gabriele Castellano
Huawei Technologies, France
Paris, France

Massimo Gallo
Huawei Technologies, France
Paris, France

Salvatore Pontarelli
University of Rome - Sapienza
Rome, Italy

ABSTRACT

This work presents a generic framework that exploits learning to improve the quality of network measurements. The main idea of this work is to reuse measures collected by the network monitoring tasks to train an ML model that learns some per-flow characteristics and improves the measurement quality re-configuring the memory according to the learned information. We applied this idea to two different monitoring tasks, we identify the main issues related to this approach and we present some preliminary results.

ACM Reference Format:
Andrea Monterubbiano, Raphael Azorin, Gabriele Castellano, Massimo Gallo, and Salvatore Pontarelli. 2022. Learned Data Structures for Per-Flow Measurements. In *CoNEXT Student Workshop 2022 (CoNEXT-SW '22), December 9, 2022, Roma, Italy.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3565477.3569147

1 INTRODUCTION

Measurements are fundamental for network operations and management. However, due to the high load and limited device resources, accurate per-flow monitoring is not a viable solution. To overcome this problem, network monitoring systems employ approximated data structures, *sketches*, to estimate relevant traffic metrics. Sketches can be used to estimate a variety of traffic metrics such as frequency e.g., Count-Min Sketch (CMS) [1], to identify Heavy Hitters, or inter-arrival time (IAT) distribution e.g., Distributed Distribution Sketch (DDSketch) [4], to identify performance degradation. These data structures trade measurement accuracy for memory efficiency making traffic monitoring viable for memory and computational resource-constrained programmable switches. Optimal sketches configuration is tightly related to traffic characteristics. However, sketches are designed (and configured) for the general case, leading to possible performance degradation in the case of adversarial flow patterns. Machine Learning (ML) models have been recently used in the networking community to capture complex traffic behaviors in the context of traffic classification [5], security [2], etc. We claim that simple ML models can successfully predict traffic characteristics at a reasonable cost to hint towards optimal (and dynamic) sketches configuration.

In this work, we explore the design and potential benefits of *learned sketches*: ML-enhanced probabilistic data structures. In Section 2, we first present the generic concept of the proposed framework and its preliminary evaluation for two sketches: CMS and DDSketch. Finally, we conclude the paper with the next steps that we identified as future work.

2 LEARNED DATA STRUCTURES

The design of the proposed framework is depicted in Figure 1. Collected measures are used to estimate traffic metrics and train an ML model that learns relevant per-flow characteristics, or *hints*. Such hints are used to configure the data structure(s) for the next measurement epoch, with the objective of reducing the approximation error. In the following, we describe the key points of the proposed framework.

Feature selection. ML input features can be of 2 types: *i*) ubiquitous packet headers fields, e.g., full or partial 5-tuple, or *ii*) complex per-flow features such as IAT, packet size, etc. of the first *N* packets. The first approach has the advantage of providing hints to any packet. However, such hints are inherently related to the network layout rather than to the actual traffic behavior. The second approach requires additional memory to store the features relative to the first *N* packets, but also reduces the number of inferences needed since short flows are naturally neglected.

Model hints. *Hints* are used to configure data structure(s) to improve the monitoring task quality. While the ML output is task-dependent, two orthogonal characteristics can be identified: *i*) coarse *vs* fine grain, and *ii*) generic *vs* specific. *Coarse* grain hints are expressed with very few possible values e.g., long/short flow. These hints can be used to allocate the flow to the most appropriate data structure based on its class. *Fine* grain hints, instead, can be used as input parameters for the data structure e.g., a hint could select the sampling rate for each flow to monitor. Hints can than be *generic* e.g., the flow size used, or *specific* and directly related to the measure of interest e.g., the max IAT value to correctly size DDSketchs.

Misprediction cost. The proposed framework relies on the hints' correctness. However, ML models are hardly 100% accurate and a robust measurement system must be able to tolerate mispredictions. We address this issue by allocating additional memory and preventing unbounded errors in case of incorrect hints. In the following, we describe such mechanism both for CMS and DDSketch.

We evaluated our approach on a 1-hour traffic trace from the CAIDA dataset and, for each data structure, we compared its standard implementation with its "learned" counterpart in terms of cost-performance trade-off.

CoNEXT-SW '22, December 9, 2022, Roma, Italy
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9937-1/22/12.
https://doi.org/10.1145/3565477.3569147

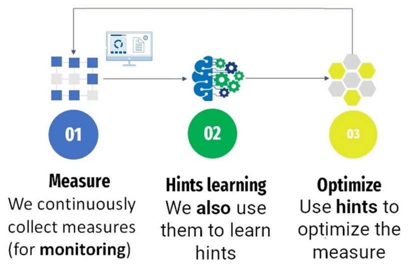

01
Measure
We continuously collect measures (for monitoring)
02
Hints learning
We also use them to learn hints
03
Optimize
Use hints to optimize the measure

Figure 1: Schematic view of a learned data structure

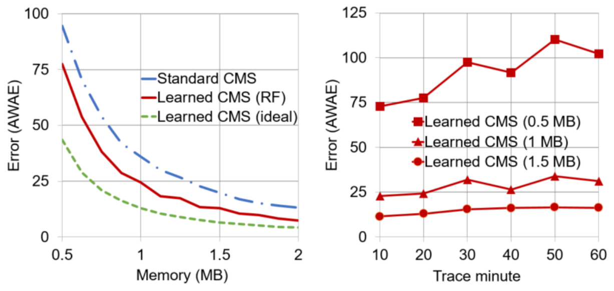
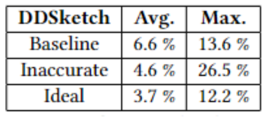


Figure 2: (left) Comparison of flow size estimation. (right) Comparison of three RF-based learned CMS



DDSketch	Avg.	Max.
Baseline	6.6 %	13.6 %
Inaccurate	4.6 %	26.5 %
Ideal	3.7 %	12.2 %

Figure 3: Comparison of a standard DDSketch against two learned DDSketch.

2.1 Learned Count-Min Sketch

In a learned CMS [3], thanks to the hints provided by an ML model, flows predicted to be elephants are counted in dedicated buckets while mice are estimated using an L by K CMS. This improves the overall accuracy by reducing hash collisions caused by elephants and at the same time exactly counting them. Our approach differs from [3] in two aspects:
1) Sketch sizing. Since elephant flows are counted separately, CMS counters size can be reduced e.g., from 32 to 16 bits. However, in case of mispredictions, this spoils the CMS by inserting elephant flows, which can trigger counters overflow. To address this problem, we use 32-bit buckets in the first CMS row and 16-bit ones for the remaining $L - 1$ rows. Our simulations confirm that this solution is sufficient to deal with misprediction effects.
2) ML architecture. To predict the flow class, i.e., elephant or mice, in [3] a Recurrent Neural Network (RNN) is trained with the flow 5-tuple as input. We use a Random Forest (RF), trained on the same input features, which achieves comparable accuracy. RFs present the double advantage of being extremely parallelizable and easier to implement in switches. To assess the performance of our proposal, we perform some experiments using *i*) a standard CMS, *ii*) a learned CMS fed by hints from a trained RF model (occupying ~113KB of memory) and *iii*) a learned CMS fed by a 100% accurate model i.e., oracle. Figure 2 (left) compares such data structures in terms of average weighted absolute error against available memory on the 20th minute of the network trace (training is performed on the first 7 minutes). Results show that the learned CMS requires less memory than the standard CMS with both RF and oracle models and the RF model provides similar results to the RNN model from [3]. However, due to the ever-evolving nature of network traffic, our RF-based model struggles to generalize well as we count flows from later minutes in the trace. Figure 2 (right) reports the evolution of the RF-based learned CMS estimation error every 10 minutes with 0.5, 1, and 1.5 MB memory.

2.2 Learned DDSketch

DDSketch [4] is a data structure used to estimate the distribution of a set of real positive values relative to a flow, e.g., IAT in our case. The range of possible IAT values is first divided into N buckets, each counting the number of values falling within its boundaries. To the best of our knowledge, a learned version of such sketch has never been studied in the literature. A possible learned DDSketch aims at restricting the range of possible IAT values by using hints from an ML model. Also in this case, a range misprediction can lead to completely wrong results. We mitigate the risk of model prediction error using some additional *safety bins*. If the model overestimates the lower bound of the flow IATs range, then at insertion-time, all IATs values that fall below this threshold will be considered as zeroes. To overcome this misprediction effect we allocate some buckets to cover the values falling between 0 and the predicted range's lower bound. These *safety bins* allow the learned DDsketch to guarantee a bounded relative error even in case of a wrong model prediction. To assess learned DDSketch benefits, we compared *i*) a DDSketch with 64 buckets per flow, *ii*) one with simulated inaccurate hinted range (i.e., the predicted values have a maximum relative error of 20%), and *iii*) one with an oracle. The DDSketch fed by an inaccurate model features *safety bins* by allocating 32 buckets to the predicted range and 32 buckets for the values that fall outside the ML hinted range. Figure 3 presents the average and maximum relative error for the 95-quantile estimation with the three data structures. Both the oracle and inaccurate learned DDSketches over-perform the baseline on average. We also observe that the max error increases when using the inaccurate learned DDsketch. However, the safety bins ensure that such error is bounded.

3 FUTURE WORK

The work presented in this paper just scratches the surface of the broader learned data structures topic. A lot of work remains to be done toward a realistic implementation for programmable devices. Several directions could be taken to optimize the ML model memory footprint like in [6], where match-action rules are used to implement decision trees in a switch. Additionally, model performance could benefit from feature engineering to harness more information from the flow 5-tuple like in [2]. Finally, and most importantly, the online training of learned data structures remains unexplored. Incrementally re-training models and adapting data structures accordingly should unveil interesting challenges.

REFERENCES

[1] Graham Cormode et al. 2005. An improved data stream summary: the count-min sketch and its applications. In *Journal of Algorithms*.
[2] Luca Gioacchini et al. 2021. DarkVec: Automatic Analysis of Darknet Traffic with Word Embeddings. In *ACM CoNEXT*.
[3] Chen-Yu Hsu et al. 2019. Learning-Based Frequency Estimation Algorithms. In *International Conference on Learning Representations*.
[4] Charles Masson et al. 2019. DDSketch: A fast and fully-mergeable quantile sketch with relative-error guarantees. In *VLDB*.
[5] Zhanyi Wang. 2015. The applications of deep learning on traffic identification. In *BlackHat USA*.
[6] Zhaoqi Xiong et al. 2019. Do switches dream of machine learning? Toward in-network classification. In *Proc. of the 18th ACM workshop on hot topics in networks*.