TalentSketch: LSTM-based Sketch for Adaptive and High-Precision Network Measurement

Yangsheng Yan†, Fuliang Li†✉, Wei Wang†, Xingwei Wang†✉
† Northeastern University, Sheyang 110819, China
2201898@stu.neu.edu.cn, {lifuliang, wangxw}@mail.neu.edu.cn, zero.wwei@outlook.com

2022 IEEE 30th International Conference on Network Protocols (ICNP) | 978-1-6654-8234-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICNP55882.2022.9940396

***Abstract*—With the massive and rapid growth of network traffic, sketching techniques are widely used to estimate a variety of network flow-level metrics, e.g., flow size, top-k flows, and the number of flows. However, how to cope with the constantly changing network environment and achieve high accuracy with limited switch resources is still a challenging problem. Existing studies do not well address the issues of identifying and correcting the inaccurately estimated flows. Therefore, we propose TalentSketch, an adaptive and high-precision hybrid measurement framework based on LSTM. TalentSketch uses LSTM to learn the sketch features with the captured traffic information. It could identify the inaccurately estimated flows with an error-prone flow model and correct them with a well-trained regression model. We conduct extensive experiments to verify the performance of TalentSketch. Experimental results show that, without increasing switch resource overhead, TalentSketch improves the measurement accuracy of different kinds of sketches by 12%~23%. More importantly, it can track network fluctuations and provide feedback on the overall accuracy of a sketch in real-time.**

I. Introduction

With the constantly expanding data center networks and the rapid growth of network traffic, the complexity of network measurement has significantly increased. Achieving accurate network monitoring while considering the limited switch memory resources has become a challenging problem.

The latest progress in network measurement includes network measurement algorithms and architectures. In terms of measurement algorithms, the programmability of the data plane broadens the possibility of deploying complex measurement algorithms. Network monitoring and management tasks, such as traffic engineering [1–7], anomaly detection [8–15], and forensics [16] require accurate and timely statistics of various flow-level metrics, such as sampling [8], top-k counting [8], and sketching [3]. The common practice is to use approximation methodologies. Sketching techniques have been widely used to estimate traffic flows, and statistics of flow-level metrics could be obtained by querying sketches [17–26]. Although a sketch sacrifices some precision to reduce resource demand for commodity switches, it also introduces some new problems. For example, static sketch configurations cannot cope with the changing network environment, and hand-derive dynamic sketch models rely on simplifying assumptions. As a result, the robustness of the sketch is poor, and its application abilities are restricted.

Regarding measurement architectures, we divide the current network measurement system into host-based, switch-based, and hybrid. The host-based architecture fully utilizes the hardware resources of the host to realize fine-grained network measurement. The switch-based architecture has a better internal view of the network, which can record and investigate various events within the network and provide accurate fault location abilities. For example, *Opensketch* [1] and *Scream* [27] deploy sketch algorithms on the switch to achieve high accuracy measurement with limited resources. The hybrid architecture is a promising framework taking advantage of the host and the switch. The switch processes the traffic first and reduces the traffic volume sent to the host. The host executes the query tasks that cannot be implemented on the switch. On the one hand, the internal view of the switch is fully utilized. On the other hand, the hardware resources of the host are explored for more complex query tasks. For example, *Sonata* [28] abstracts them into a streaming database, and *Marple* [29] abstracts them into a key-value database. The switch is designed as the cache of the database. Although the host has more computing and storage resources, its query accuracy is still constrained by the limited resources of the switch.

To improve the accuracy of sketch, some studies optimize sketch with deep learning technologies [30]. *RL-Sketch* applies reinforcement learning to improve *LD-Sketch* for identifying heavy hitters [31]. *ML-Sketch* [32] is another sketch-optimized model based on machine learning. It first identifies the error-prone flows, which are estimated by sketch with low precision. However, the identification accuracy cannot be guaranteed because the identification is heavily dependent on a manually set threshold. The follow-up regression model trained with the error-prone flows may learn the features of some high-precision flows. As a result, the overall accuracy of the sketch is not improved greatly.

As a supplement to the hybrid architecture and existing studies, this paper proposes TalentSketch, which introduces Long Short Term Memory to improve the accuracy of the sketch. First of all, to overcome the weakness of static configurations, we train the LSTM-based models to get real-time feedback on sketch accuracy, which helps adjust the sketch configurations dynamically. In addition, the model training samples are taken from a production network to reduce the dependence on traffic characteristics, significantly enhancing the application abilities. At last, to decrease the impact of limited switch resources on the accuracy, we use the host's computing resources to extract valuable information hidden in the sketches and further correct the low-precision flows. We

978-1-6654-8234-9/22/$31.00 ©2022 IEEE

note that model training increases the communication overhead between the switch and the host. To enhance the practicability of TalentSketch, we only choose a certain proportion of traffic as the training samples sent from the switch to the host. In this way, the extra communication overhead brought by TalentSketch is significantly reduced.

In summary, we make the following contributions.

- We propose TalentSketch to optimize the sketch-based hybrid measurement architecture. We reconfigure the sketch on the host based on the number of flows and the switch sketch size. To reduce the impact of sampling and restores the original features of the traffic, we propose a mapping policy to ensure that the ratio between counter values of the same flow is essentially constant across the reconfigured sketch and the switch sketch. We then train the proposed models with low bandwidth overhead with the restored sketch data.
- We propose an error-prone flow model to identify error-prone flows, which is utilized to react to network changes and further give instructions on how to adjust the sketch. We then apply a well-trained regression model to correct the error-prone flows. We also tune the neural network structures for these two models to achieve the linear time measurement.
- We evaluate TalentSketch with data-driven analysis. The actual dataset of a data center network is obtained from CAIDA [33]. Experiment results show that TalentSketch improves the accuracy by 12% through correcting the flows with low measurement accuracies. More importantly, TalentSketch can capture real-time network traffic dynamics and sketch accuracy variations.

II. Related Work and Motivation

A. Background

Sketch: sketches support upper-level decisions by measuring the flow-level information. For example, the flow size is a typical metric estimated by the sketch in network measurement applications. It is usually implemented by three kinds of sketches, *CM* sketch [34], *CU* sketch [35], and *CSM* sketch [36]. They have the same data structure and are composed of d counters. Each counter is associated with a hash function, and the counter length is w.

Long Short Term Memory: the deep learning model used in this paper is Long Short-Term Memory Network (LSTM). LSTM is an artificial recurrent neural network with feedback connections. It can process a single data point as well as a whole data sequence. LSTM is quite suitable for classification, processing, and prediction based on time series data because there are some unknown persistent correlations between important events in time series.

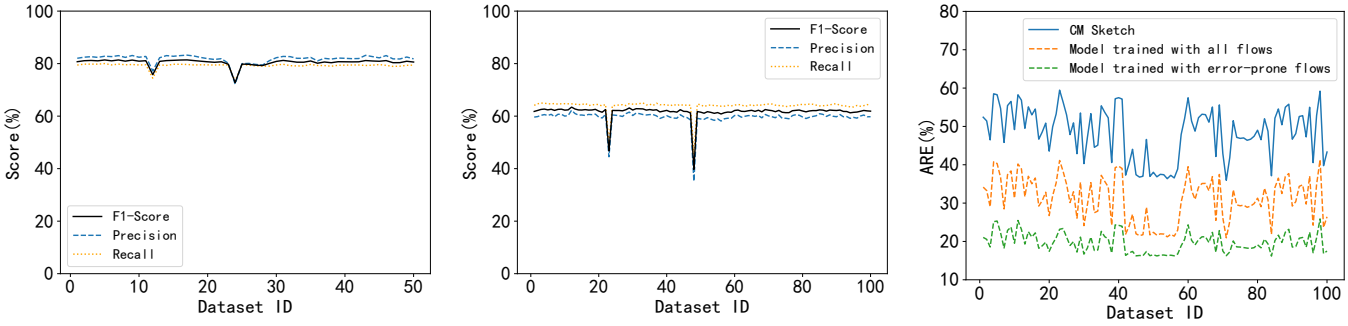
B. Existing works

Sketches require a large amount of memory to accurately estimate each traffic flow. As a summary algorithm, a sketch usually spends much less memory than the input data. Extensive studies have been conducted to effectively store and measure the targeted information by sketches.

Yang et al. put forward the OpenSketch [1], which is the earliest sketch-based measurement architecture for SDN. OpenSketch uses the classification function to select specific flows for statistics, and the interested flows are recorded into the sketch. It achieves good performance and causes extensive discussion on the tradeoff between the sketch resources and the measurement accuracy. With the growth of traffic, achieving accurate and scalable measurements with sketches has become a challenging problem. However, static memory allocation of the sketch causes scalability problems. When the memory is overwhelmed, the performance of the sketch cannot be guaranteed. To solve these problems, some studies use counter-based technologies to dynamically accept or exclude the flow to improve the scalability of sketches [11, 17, 37, 38]. For example, Huang et al. propose LD-sketch for detecting heavy hitters [10]. LD-sketch adjusts the size of sketch memory by counting. It tracks the heavy flows and dynamically records the flows of interest rather than the uninterested flows. It greatly reduces storage space usage and maintains certain accuracy and stability. However, the measurement accuracy will decrease in some special cases, such as the sharp increase of heavy flow frequency. BitCount[39] maintains a set of counters to count the total number of 1-bits in each bit position of the binary representation of IP addresses. However, owing to the additional property of bit counters, this method may involve inevitable false positive errors. IM-SUM[38] achieves constant update time consumption, i.e., an amortized time of O(1), by periodically removing many small flows from the table at once instead of removing the minimum flow upon the arrival of a non-resident flow. However, IM-SUM performs only a small portion of the maintenance procedure to achieve a time complexity of O(1).

There is still much room for improving counter-based technologies. The accuracy of the existing sketches varies significantly with the changing characteristics of network traffic. They insert approximate information about the expected metric into the sketch and then estimate it based on a hand-derived theoretical model. To derive the model manually, it is necessary to simplify the assumption of network traffic because it is impossible to derive a model covering all actual scenarios. Since the network traffic characteristics change significantly in practice, and the simplified assumptions are not always valid, the accuracy of existing sketching techniques used in actual deployment would vary greatly.

Guo et al. have pointed out that there are more than 80 sketch designs and optimization strategies, but only two are related to machine learning algorithms [30]. RL-Sketch [31] uses deep reinforcement learning to replace the manual counting in LD-sketch for dynamic adjustment and realize the intelligent regulation of storage space. However, it is not a universal measurement model because it is only designed to optimize a special sketch. Yang et al. design a measurement model based on machine learning [32]. They use linear regression



(a) Classification results of two-second interval traffic data.

(b) Classification results of one-second interval traffic data.

(c) Results of different regression models.

Fig. 1. Experimental analysis of existing studies.

models to correct the error-prone flow query value in the sketch. It has a certain universality and achieves a certain improvement in query accuracy while maintaining the server resource overhead. However, the process of identifying error-prone flows is excessively dependent on the threshold that is set manually. As a result, the accuracy of identification cannot be guaranteed, which greatly affects the accuracy of linear regression.

C. Motivation

In this section, we take CM sketch as an example and use case studies and experiments to illustrate the shortcomings of identifying error-prone flows by manually setting thresholds and the shortcomings of the regression model trained by all flows, inspiring the thinking of using machine learning to train error-prone flows model and the use of low-precision flows to train regression model. Section IV will introduce all the metrics and parameters.

Identify error-prone flows by manually setting thresholds. We conduct some experiments to illustrate the shortcomings of classifying error-prone flows by manually setting thresholds. In this paper, error-prone flows are defined as the flows with low precision. We adopt a quantitative criterion, that is whether the ratio between the real value of the flow and the query value is less than 90% to define an error-prone flow. We use this criterion to check the classification effect with the manually setting thresholds. We insert the traffic extracted from the dataset every two seconds into the sketch. Then, the error-prone flows are identified through a manually set threshold. We also insert the traffic extracted from the dataset every second into the sketch and conduct the identification process once again.

As shown in Figure 1 (a) and Figure 1 (b), with the two-second interval traffic data, the F1-Score is 0.805, the precision is 0.817, and the recall is 0.792. While the F1-Score is 0.617, the precision is 0.596, and the recall is 0.639 with the one-second interval traffic data. The results of classifying error-prone flows are quite different when the threshold is the same. This is because both the number of packets of the same flow inserted into the sketch and the sketch sequence values of the same flow are different. Therefore, a manually set threshold cannot adapt to diverse traffic environments.

Train regression models with error-prone flows classified by a threshold. In this section, we use an experiment to illustrate the disadvantages of training regression models with error-prone flows classified by a threshold.

Classification by manual threshold classifies a large number of high-precision flows as error-prone flows, which significantly reduces the accuracy of the regression. Because this classification causes many real low-precision flows to be ignored, and many high-precision flows are used for training, the model effect is also poorer than using all low-precision flows for training. To observe the influence of non-error-prone flows on regression model training, under the preset sketch size, we insert the traffic extracted from the dataset per second into the sketch and extract all error-prone flows with less than 90% accuracy, the error-prone flows of the sketch are then optimized by a regression model trained with all flows and a regression model trained with low-precision flows, respectively.

As shown in Figure 1 (c), in our preliminary experiments, under the same traffic environment, the regression model trained with all flows can decrease ARE a lot for CM sketch, but compared to the regression model trained with low-precision flows, the ARE reduction is still less. The poor model performance is due to the large number of high-precision flow features learned when the regression model is training with all flows.

Problem Statement: ML-Sketch [32] classifies error-prone flows only by comparing the difference between the two smallest values in the sketch sequence with its manually set threshold. However, due to the different update and query algorithms of distinct sketches, it isn't easy to find a common threshold to ensure classification accuracy. In practice, due to changes in network traffic, thresholds also need to be adjusted frequently to adapt to different traffic environments, achieving the best balance between generality and accuracy is difficult. In addition, how ML-Sketch classifies error-prone flows also results in many sketch sequence values of high-precision flows being used to train regression models. In this work, we have a more quantitative definition of an error-prone flow, use deep learning models to classify error-prone flows, and improve the training method for regression models.

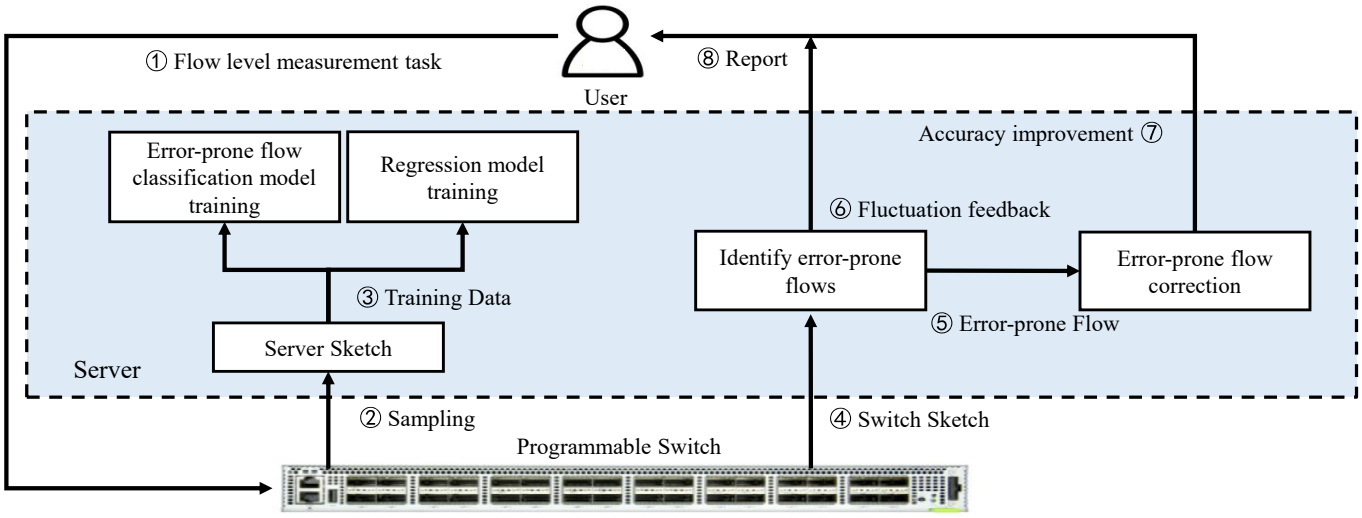


Fig. 2. The framework of TalentSketch.

III. TalentSketch Design

TalentSketch is a generalized measurement framework that does not depend on any specific sketch type or specific query task and is widely applicable to query tasks based on various sketches. It only uses a small percentage of all packets to generate the models which are sampled from the network to be tested, and feeds back the changes in the sketch accuracy caused by network fluctuations in real-time, and corrects the traffic that is not accurately measured.

Figure 2 shows its internal structure. When the user deploys the sketch-based query task in the switch, the server samples the switch traffic to construct a relatively small sketch to learn the characteristics of the current network environment. TalentSketch uses these features to train the models, analyze the accuracy of the sketch measurement deployed in the switch, and make corrections.

A. Sampling

Flow-level network measurement is usually performed with a time window of one second, and the size of the sketch on the switch is also a preset estimated size based on the flow per second. Since network traffic passes through the measurement node very quickly, the sketch will be filled up within one second. At this time, the switch sends the sketch to the server for storage and clears the switch sketch to prepare for the following second measurement. The server extracts the corresponding information according to the sketch sent by the switch and answers the user's measurement task. To reduce the model's dependence on the network environment and improve the effectiveness of the model in a particular network environment, we consider obtaining traffic samples from the application scenarios of the model by sampling and then generate a dataset to train the model with environmental characteristics. The model trained by this method has the characteristics of the network environment and can reach higher accuracy when used.

The sampling process brings additional bandwidth overhead between the server and the switch. To avoid affecting the regular business between the server and the switch, we have designed an adjustable sampling rate, which could be adjusted manually according to the idle bandwidth between the server and the switch. If the sampling rate α is set, the switch will send one of the data packets to the server whenever α packets are passing through the switch. Of course, by using a high sampling rate, the flow data obtained will be more granular, and the trained models will have stronger perceptions, making the result more accurate.

B. Training data generation

Next, we use the sampled traffic to generate the training model data on the server. This requires the establishment of two additional data structures on the server to count the sampled flow information: (1) establish an error-free data structure to count the accurate information of the flow, which is used to assist sample generation and label marking, (2) establish a sketch similar to that on the switch, and generate sample features of the flows passing through the sketch.

First, benefiting from the abundant computing and storage resources on the server and the relatively loose timeliness requirements, it is simple and feasible to use the error-free data structure to count the information we care about in the flows after sampling. Data structures such as red-black trees can be used to count traffic accurately. Then it records the information of the number of flows and the number of packages to guide the establishment of the sketch on the server and records the to-be-tested query task information to mark sketch convection's measurement accuracy.

After that, we use sampled traffic on the server to create the same kind of sketch as on the switch. To facilitate the distinction, we call the sketch built on the switch "Switch Sketch", and the sketch built on the server "Server Sketch". The server sketch uses the sampled flows sent by the switch to generate sketch data with the characteristics of the sampled

traffic, including the sketch sequence values and query values of the flows. It then combines these data with the real values of the corresponding flows recorded in an error-free data structure to generate model training data and uses these data to train the models. Due to the existence of the sampling process, the number of flows and packets has changed significantly. If the server uses the same sketch as that on the switch, this will result in a large difference between the data distribution of the dataset used for model training and model application, which results in a great loss in the accuracy of the model. To reduce the impact of sampling on the characteristics and restore the original characteristics of the traffic, we design two Server Sketch adjustment strategies to maintain the characteristics of Server Sketch.

Relatively small sketch resource configuration. The difference in the number of flows leads to different hash collision probabilities when using the same sketch. The Server Sketch resource configuration determines the sketch hash conflict rate of the training samples. Adjusting the Server Sketch resource is one of the key steps in restoring traffic characteristics. The Switch Sketch size $SwSL$ (Switch Sketch Len) is based on the user's estimation of the number of flows (for example, using FMSketch) in the network environment to obtain the number of flows in the network environment $SwFN$ (Switch Flow Number), and then the switch sketch size can be obtained based on the experience or the theoretical equation of sketch error. The Flow size $SeFN$ (Server Flow Number) sampled by the server is obtained by accurate statistics using an error-free data structure. The hash number of the server, that is, the number of sketch rows, is the same as that of the switch, and the Server Sketch size $SeSL$ (Server Sketch Len) is reduced according to Eqn (1).

$$SeSL = \frac{SwSL * SeFN}{SwFN} \tag{1}$$

Sketch data interval mapping. The traffic before and after sampling has a large difference in the number of packets, which will cause the Server Sketch measurement value of the flow sample to be very different from the measurement value on the Switch Sketch. The accuracy of sketch convection measurement is usually determined by the difference between multiple counters indexed by the Flow ID. So we can take the relative size of the counter values of each flow. In addition, the size of the counter values between different flows also implies certain information. For example, a large flow and a small flow are mapped to the same counter through a hash function, which often leads to a significant decrease in the measurement accuracy of a small flow. So we need to take the relative value of the traffic samples in the same time window according to the same reference. In each time window, according to the maximum value $LMax$ and the minimum value $LMin$ in the sketch, each value V in the sketch is mapped to the interval from zero to one using Eqn (2) to obtain V'. Since $LMin$ tends to have a relatively small value, it can be ensured that the ratio between the counter values of the same flow is basically unchanged. It is worth noting that this equation also needs to map the real value measured by the error-free data structure. When the smart model is applied to Switch Sketch, Switch Sketch adopts this mapping method. The output result of the model is changed inversely according to Eqn (2), and its real value can be restored.

$$V' = \frac{V - LMin}{LMax} \tag{2}$$

C. Model design

As shown in Figure 2, according to the measurement task we deploy in this paper, i.e., the flow size, we train two LSTM-based models with the trained dataset. One is called the error-prone flow model, which is used to judge the accuracy of the flow estimation and identify the flows with low-precision estimation. The other is the regression model for the error-prone flow correction. In fact, after sampling, training data generation, and training machine learning-based theoretical models according to the corresponding measurement task, the framework can also deal with other queries.

LSTM-Based. First, to classify and correct the accuracy of the sketch, we not only consider the linear relationship but also take the nonlinear relationship into account. In our experiments, we found that the linear model of machine learning can also be classified and corrected to a certain degree. However, the accuracy of LSTM is much higher than it. The correction of the flow accuracy of the LSTM can often reach twice that of the linear model (Exp#5). In addition, LSTM has a strong affinity with the sketch. Sketch measurement often uses a sequence composed of counter values indexed by the Flow ID through hash functions to estimate the measured value of the flow. The relative size relationship between counters is mainly used in this process. Take the query of Flow Size using Count-Min sketch as an example. Count-Min sketch puts the five-tuple of the flow into each hash function for query, sorts the query values, and lists them in turn. It finally selects the smallest as the sketch query value, the query results of various sketches have varying degrees of dependence on the interrelationship of the elements in the sequence. Due to the multi-hash function feature of the sketch, the difference between the values in the sequence is often the basis for judging how many hash conflicts occur, and it is also the basis for the reliability of the query. The relative size of the values in the sequence greatly affects the accuracy of the query. LSTM is good at processing serialization features and can make bracelets faster. Finally, the sketch's dimensions can be changed, that is, the number of hash functions can be changed. LSTM can handle variable-length data. So we use LSTM as the basic model.

Special design. In sketch measurement, inaccurate measurement flows only account for a small part. If we only build a single model to detect the accuracy of all traffic, the model will be less affected by a small amount of inaccurate measurement flows during model training. This will make the model converge slowly and have low accuracy. In addition, a complete and accurate accuracy detection model requires a complex neural network structure, and it will have poor efficiency when

it is put into use. Most network measurement tasks do not require utterly accurate measurement, and minor errors are negligible and do not affect the nature of the measurement results. Therefore, a smaller neural network structure can be used to classify the traffic with extremely low measurement accuracy and then corrected by the larger regression model of the network. This not only ensures accuracy but also improves timeliness, so that the data can be processed in linear time.

Error-prone flow model. We define an error-prone flow as a flow that is inaccurately estimated by the sketch and the margin of error exceeds 10%, that is, the estimation accuracy (the definition of accuracy will be introduced in Section IV) of this flow is less than 90%. The error-prone flow model is trained using the sketch generated by the flow after sampling, and the sketch sequence of the flow is used as a feature. Whether the ratio of the real value of the flow obtained from the error-free data structure to the sketch query value exceeds 90% is used as a label for training. There are two primary purposes for designing the error-prone flow model. On the one hand, error-prone flows only account for a part of all flows, so not all flows need to be corrected for query values. We separate the error-prone flow from the non-error-prone flow in the sketch, sending the error-prone flow into the regression model and directly generating the user report for the non-error-prone flow. On the other hand, it could get feedback about network changes' impact on the sketch's accuracy. The error-prone flow model can judge the proportion of current inaccurate flows to the total flow to respond to changes in network traffic in real-time. When the network traffic surges, the proportion of error-prone flows increases. Users can adjust the Switch sketch size.

Regression model. The role of the regression model is to deal with error-prone flows and correct inaccurate query values. In the first part, we observed that a regression model trained with all flows learns many low-precision flow features and cannot guarantee good regression performance on error-prone flows. The regression model adopts the sketch generated by the traffic after sampling and takes the sketch counter sequence of the error-prone flows instead of all flows as the feature, and the real value as the label for training. Since the information in the sketch is not fully utilized, for example, the Count-Min sketch only selects the one-dimensional hash function value with the smallest value as the query value, and the size relationship between the remaining values is not effectively utilized. Assuming that the counter sequences of the two flows are <100, 102, 103> and <100, 153, 199> respectively, they will get the same query value. However, the difference between the counter value sequence of the first flow is slight, which is in line with the characteristics of hash collisions between large and small flows and has little impact on large flows. There are more conflicts in the second flow, from the sequence information, and it can be discovered that this flow needs more adjustments. The difference between the feature sequence of the error-prone flow and the real value will be greater than the difference between the feature sequence of the non-error-prone flow and the real value. Learning the feature sequence of the error-prone flow can also avoid learning too many useless features, so it improves the query accuracy of the error-prone flow more.

TABLE I
SUPER PARAMETER

Type	Layer	Unit	Epoch	LR	Batch Size
E-p flow	1	5	1000	0.003	5000
Regression	1	30	3000	0.0003	30000

D. Response to query

In response user's query, the framework uses the error-prone flow model to process the Switch Sketch sent to the server, separate the error-prone flows, and generate a report of the number and proportion of error-prone flows, and a report of the query value of non-error-prone flows. After that, the error-prone flow is fed into the regression model. The regression model corrects the error-prone flow to a certain extent and generates the error-prone flow query value report. Finally, the error-prone flow percentage report, the non-error-prone flow query value report, and the error-prone flow query value report are combined and sent to the user as the answer to the query in this time window.

IV. IMPLEMENTATION AND EVALUATION

In this section, we conduct extensive experiments to verify the performance of TalentSketch. Our experiments show that with low additional bandwidth overhead, TalentSketch can do the following: (1) Classify traffic and extract error-prone flows that are inaccurately estimated by the sketch (Exp#1). (2) Correct the low-precision flow rate to reduce measurement errors (Exp#2). (3) Reflect changes in the network environment by generating the error-prone flow quantity report (Exp#3). (4) Apply to various sketches to optimize sketch measurement (Exp#4). (5) Get better performance than ML-Sketch[32] (Exp#5). (6) Prolonged use after one training session (Exp#6).

We developed TalentSketch in Python 3.7, and the prototype used Pytorch, which is a deep learning framework. During the model training process, we train the proposed two models at the same time. The regression model has more neural network nodes than the error-prone flow model. We use Pytorch-based LSTM, and all the parameters utilized are shown in Table I.

A. Setup

Telemetry Application. The measurement task of our deployment is to estimate the flow size. Three different types of sketches are used to measure the size of the flows, and TalentSketch is used to optimize it with three different sampling rates. We use the first 10 seconds of data to train the model, and use the next 100 seconds of data for testing.

Testbed. As shown in Table II, our testbed is equipped with 64 2.60GHz CPU cores and 188GB RAM, and it runs Ubuntu 18.04.

TABLE II
EXPERIMENTAL PLATFORM

name	Specifications
CPU	Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz
Memory	188G
System	Ubuntu 18.04

TABLE III
SKETCH CONFIGURATION

Type	Rows	Column
CM sketch	3	40000
CU sketch	3	40000
CSM sketch	3	70000

Trace. We pick a one-hour trace in CAIDA 2018 [33]. We divide the trace into one-second time intervals, and each contains 70K active flows and 640K packets.

Sketch Parameters. The size of the sketch configured on the switch determines the accuracy of the baseline. Combined with the theoretical calculation results of the sketch error and the configuration experience, as shown in Table III, we set the number of rows of the three kinds of sketches to 3 and the number of columns to 40,000 and 70,000. The accuracy of the CM sketch under the network load of this experiment is about 70%, the average accuracy of the CU sketch is about 80%, and the accuracy of the CSM sketch is about 50%.

Sampling Parameters. We sample the packets uniformly and use three sampling rates: high, medium, and low to evaluate TalentSketch. When using a high sampling rate, the server samples every 5 packets. The medium sampling rate is sampling every 10 packets, while the low sampling rate is 50.

Training Consumption. For time overhead, our model training time is limited to 10 minutes. In fact, the model tends to converge enough to be used within 3 to 5 minutes. As shown in Figure 3, we sample the dataset for the first ten seconds to evaluate bandwidth consumption. Due to different sampling rates, additional bandwidth between the server and the switch is consumed between 200K and 1.6M per second.

Methodology. By controlling the same sketch resource overhead, we compare the changes in sketch measurement accuracy before and after the application of TalentSketch. To realize the network change response test, we simulated the rapid changes in the network environment by merging the traffic for several consecutive seconds into a one-time window.

Metrics. The error-prone flow model is a two-classification model: all network flows are classified into error-prone and non-error-prone flows. Precision, recall, and F1-Score are the commonly used evaluation indicators for binary classification tasks. Precision refers to the proportion of true predictions among the predicted positive classes. Recall refers to the proportion of positive classes that are correctly predicted among all positive classes. F1-Score refers to the harmonic average of precision and recall, which can better reflect the performance of the two classification algorithms. The regression model is a model of value estimation. Using the average relative error ARE can better reflect the change in the overall accuracy of the flow.

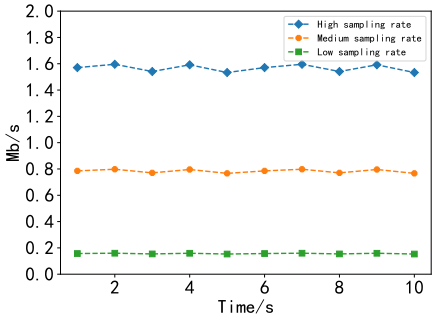


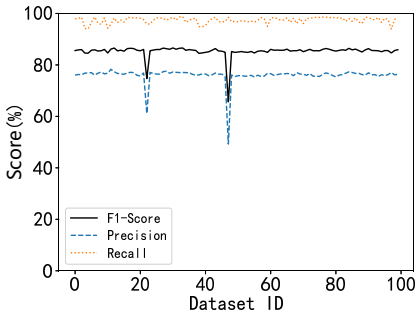
Fig. 3. Bandwidth consumption with different sampling rates.

Since there are both large and small flows in the flow sample if the absolute error is used, the weight obtained by the large flow will be larger, and the overall error change cannot be reflected. Using the average relative error (ARE) can better reflect the change in the accuracy of the overall flow. r_i represents the true value of the i^{th} flow, $\hat{r_i}$ represents the measured value of the i^{th} flow, and n represents the total number of flows, then the average relative error ARE is defined as $\frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{r_i}-r_i|}{r_i}$, and we define $Accuracy = 1 - ARE$.

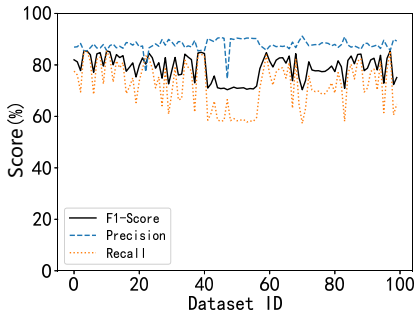
B. Experiment results

(Exp#1) Classification accuracy. Figure 4 shows the test results of the error-prone flow model trained with the high sampling rate, medium sampling rate, and low sampling rate. At the high sampling rate, the F1-Score is 0.858, the precision is 0.759, and the recall is 0.986. At the medium sampling rate, the F1-Score is 0.788, the precision is 0.877, and the recall is 0.721. At the low sampling rate, the F1-Score is 0.540, the precision rate is 0.959, and the recall rate is 0.377. The F1-Score and recall decrease as the sampling rate decreases, and the precision increases as the sampling rate decreases.

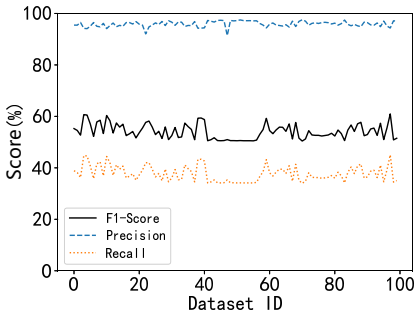
The error-prone flow model does not reach a very high accuracy at a higher sampling rate. This is because error-prone and non-error-prone flows have the same characteristics in some cases. Assuming that there are large and small flows in the network that are indexed to the same counter by the hash function of the sketch, they have the same characteristics. For large flows, the deviation of the counter from the real value is the size of the small flow, and for small flows, the deviation of the counter from the real value is the size of the large flow. Small flows have larger errors and are more likely to become error-prone flows, while large flows have smaller deviations and are more likely to be classified as non-error-prone flows. So there may be cases where the characteristics are the same, but the labels are different. In this case, to obtain a smaller loss, the error-prone flow model will learn in the direction with a larger number of the two labels, so that most of the samples have the correct classification. Thus, there will always be a small part of the classification errors of the sample.



(a) Flow classification with a high sampling rate.

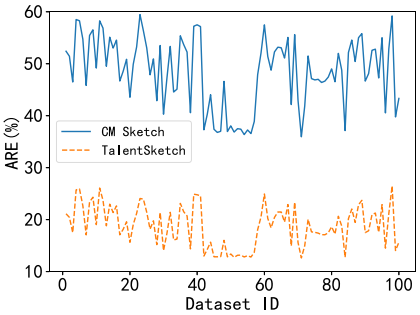


(b) Flow classification with a medium sampling rate.

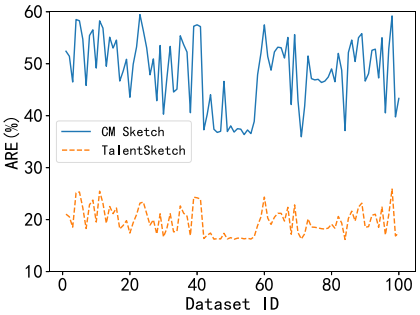


(c) Flow classification with a low sampling rate.

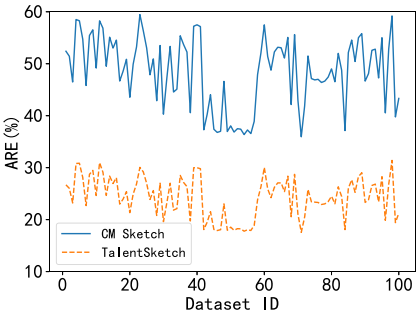
Fig. 4. (Exp#1) Flow classification performance under different sampling rates.



(a) Correction of error-prone flows with a high sampling rate.

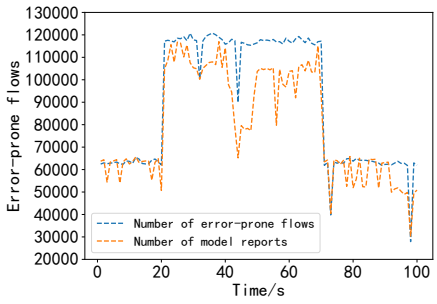


(b) Correction of error-prone flows with a medium sampling rate.

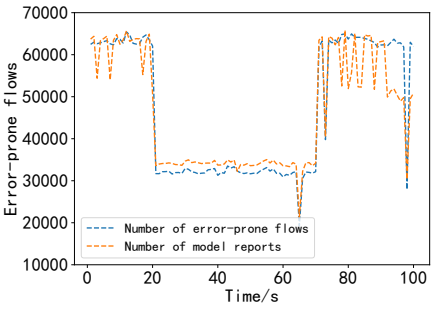


(c) Correction of error-prone flows with a low sampling rate.

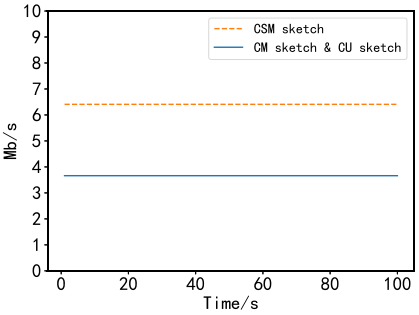
Fig. 5. (Exp#2) Correction of error-prone flows under different sampling rates.



(a) Response to sharply increasing flows.



(b) Response to sharply decreasing flows.



(c) Bandwidth consumption of uploading the switch sketch.

Fig. 6. (Exp#3) TalentSketch responds to network fluctuations.

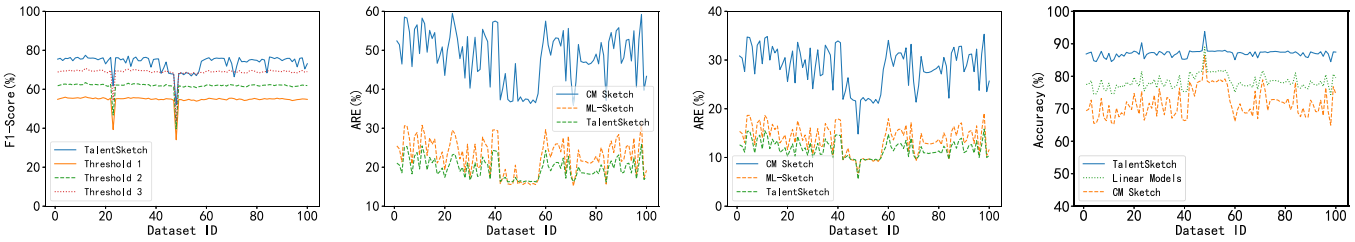
As the sampling rate decreases, the recall keeps dropping, the precision keeps rising, and the F1-Score keeps dropping. This is because the model's ability to detect error-prone flows is weakened at low sampling rates, and only a few flows with obvious error characteristics are detected.

(Exp#2) Correction of error-prone flows. Figure 5 shows the corrections of regression models under three sampling rates: high, medium, and low. Initially, the average relative error of error-prone flows in CM Sketch is 48.45%. When the sampling rate is high, the average relative error of error-prone flows drops to 18.81%, which is a decrease of 29.64%. After correction by the regression model trained by the medium sampling rate, the average relative error of the error-prone flow dropped to 19.70%, a decrease of 28.75%. At a low sampling rate, after the correction of the regression model, the average relative error of the error-prone flow dropped to 24.35%, a decrease of 24.10%.

The regression model can effectively reduce the error of the flow samples whose sketch measurement accuracy is less than 0.9 under the three sampling rates of high, medium, and low. When the sampling rate increases, the reduction accuracy becomes smaller, and the measurement accuracy improves. As the sampling rate decreases, the number of flows decreases, the number of packets decreases, and the difference between the sample characteristics and the original flow becomes larger. The correction value of the convective sample of the

(a) TalentSketch optimizes CM sketch. (b) TalentSketch optimizes CU sketch. (c) TalentSketch optimizes CSM sketch.

Fig. 7. (Exp#4) TalentSketch optimizes different kinds of sketches.



(a) Comparison of F1 scores for different thresholds and model classification.

(b) Correction of error-prone flows with accuracy below 90% by different regression models.

(c) Correction of error-prone flows classified by thresholds by different regression models.

(d) LSTM models vs. Linear models.

Fig. 8. (Exp#5) TalentSketch vs. ML-Sketch.

regression model becomes smaller, and the average relative error increases.

(Exp#3) Response to network fluctuations. Figure 6 shows the error-prone flow model's response to traffic changes and the bandwidth consumption when data is uploaded to the switch. Figure 6 (a) shows error-prone flows changes reported by the error-prone flow model under simulated network traffic. At the 20th second, there was a sudden change in network traffic, and the number of traffic per second increased sharply. At the 70th second, the network traffic recovered, and the number of error-prone flows reported by the error-prone flow model also returned to normal. Similarly, in Figure 6 (b), when the network traffic suddenly decreases, the number of error-prone flows reported by the error-prone flow model decreases, and when the network returns to normal, the number of error-prone flows reported will return to the original value. Figure 6 (c) shows the bandwidth consumption when the switch sketch is uploaded to the server for measurement, CM sketch and CU sketch have the same size, so the bandwidth consumption is the same. The error-prone flow model can respond to the dynamics of network traffic promptly. When the network fluctuates sharply, the reported error-prone flows present significant changes.

(Exp#4) Optimizes different kinds of sketches. As shown in Figure 7 (a), the measurement accuracy of CM sketch before the optimization is 71.73% on average, and the measurement accuracy after optimization by the model trained with all flows is 87.03% on average, which increases the measurement accuracy of CM sketch by 15%.

As shown in Figure 7 (b), the measurement accuracy of CU sketch before the optimization is 81.42% on average, and the measurement accuracy after optimization by models trained by all flows is 93.86% on average, which improves the measurement accuracy of CU sketch by 12%.

As shown in Figure 7 (c), the measurement accuracy of CSM sketch before the optimization is 51.09% on average, the measurement accuracy after optimization by models trained by all flows is 74.38% on average, which increases the measurement accuracy of CSM sketch by 23%.

TalentSketch can optimize a variety of different precision and different types of sketches, which has a certain versatility, and the measurement accuracy of various sketches is improved by more than 12%. Among the three sketches, CU sketch has the highest accuracy, it has the least number of error-prone flows. CSM sketch has the lowest accuracy, it has the smallest number of error-prone flows. Therefore, after improving the accuracy of error-prone flows, TalentSketch improves the overall accuracy least for the CU sketch and most for the CSM sketch.

(Exp#5) TalentSketch vs. ML-Sketch. To respond to the query task of the flow size, TalentSketch and ML-Sketch first classify the flows into two parts, a part of the flows with higher precision, TalentSketch and ML-Sketch use the traditional algorithm of the sketch technique to estimate the size of the flows. The other part of the flows with lower precision, TalentSketch and ML-Sketch estimate their size by

applying the trained regression model on the values of the d hashed counters of this flow in the regular sketch. We compare TalentSketch and ML-Sketch from three aspects, the effect of different flow classification methods, the effect of different regression model training methods, and the effect of different machine learning models.

When classifying flows, ML-Sketch first generates a learning sketch using the packets in the sampled set S. Next, it traverses through all flow IDs in the hash table and identifies the error-prone flows by comparing the absolute difference between the values of the two smallest hashed counters of each flow. TalentSketch restores the flow features by adjusting the size of the learning sketch and performing interval mapping. It uses a more quantitative error-prone flow evaluation criterion, that is, whether the accuracy of the flow exceeds 90%, and uses this as a label to train an error-prone flow model, and use the model to separate the error-prone flow.

In Figure 1 (a) and Figure 1 (b), we have shown that the single thresholds cannot adapt to different network environments, a single threshold will have a large difference in its error-prone flow classification effect in different network environments. As shown in Figure 8 (a), in the preset network environment, we set the thresholds to 1, 2, and 3 respectively. We found that as the threshold is set smaller and smaller, the F1 score gets higher and higher, so adjusting the threshold according to the network environment is necessary. In contrast to the above, the error-prone flow model can quickly adapt to the traffic environment after once training, and the classification effect is better.

When training a regression model, ML-Sketch uses each error-prone flow classified by a threshold as a training sample. For any given error-prone flow, the values of its d hashed counters in the learning sketch serve as the features and the real values of the packet counts of the flow, recorded in the hash table, serve as the target. TalentSketch uses each error-prone flow with less than 90% accuracy as a training sample to train the regression model.

As shown in Figure 8 (b) and Figure 8 (c), compared with the regression model trained by the error-prone flows classified by the threshold, the regression model trained by the error-prone flows with an accuracy of less than 90% has a better regression effect for two kinds of error-prone flows in ML-Sketch and TalentSketch. This is because the proposed regression model learns the features of low-precision error-prone flows, which enables TalentSketch to perform maximum accuracy correction on low-precision flows.

In terms of model selection, ML-Sketch chooses the linear model to train the regression model, while TalentSketch chooses the LSTM to train the error-prone flow model. Finally, we compare the two architectures of LSTM models and linear models. As shown in Figure 8 (d), LSTM models improve sketch accuracy by 15%, and linear models improve accuracy by 7%. The accuracy improvement of the former is more than twice that of the latter, which indicates that LSTM is more suitable for the work related to sketching sequences. The above experiments demonstrate the improvement of TalentSketch in error-prone flow classification, query value regression, and the advantage of machine learning method selection.

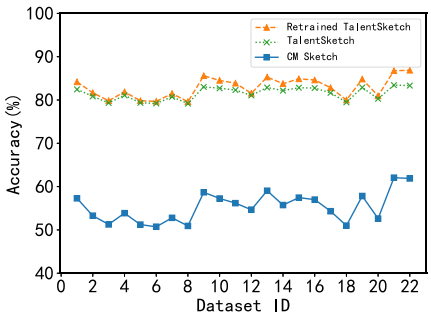


Fig. 9. (Exp#6) TalentSketch vs. Retrained TalentSketch.

(Exp#6) Training frequency. At last, we determine how often the TalentSketch should be retrained to optimize accuracy. As shown in Figure 9, We train TalentSketch under a preset traffic collection time window, then train TalentSketch under the double time window, and compare the effect of different TalentSketches under the double time window. We observed that training on a large amount of data collected under the preset time window is sufficient to provide a good guarantee for improving sketch accuracy under the big traffic environment, which reflects the adaptability of TalentSketch, confirms the experimental results of (Exp#3) and the effect of two Server Sketch adjustment strategies. So, in practice, machine learning-based training will just be an infrequent process that doesn't take up a lot of CPU, bandwidth, and memory on the switch/router.

V. Conclusion

We propose TalentSketch, a hybrid measurement architecture based on LSTM. TalentSketch can give feedback on the sketch accuracy and the network dynamics in real-time by consuming a small number of computing resources of the host. It does not depend on the specific network environment and parameter hypothesis when designing the models, including the error-prone flow model and the regression model. One is used to classify the inaccurately estimated flows, and another is applied to correct them. Experiment results show that TalentSketch improves the measurement accuracy of different sketches by 12%~23% without increasing switch resource overhead.

VI. Acknowledgments

We thank our ICNP shepherd Patrick P. C. Lee for his feedback and encouragement. This work is supported by the National Key Research and Development Program of China under Grant No. 2019YFB1802600; the National Natural Science Foundation of China under Grant Nos. 62032013, 62072091, and 61872073; the LiaoNing Revitalization Talents Program under Grant No. XLYC1902010.

REFERENCES

[1] M. Yu, L. Jose, and R. Miao, "Software defined traffic measurement with opensketch," in *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*. Lombard, IL: USENIX Association, Apr. 2013, pp. 29–42.

[2] S. K. Fayazbakhsh, L. Chiang, V. Sekar, M. Yu, and J. C. Mogul, "Enforcing network-wide policies in the presence of dynamic middlebox actions using flowtags," in *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*. Seattle, WA: USENIX Association, Apr. 2014, pp. 543–546.

[3] Q. Huang, X. Jin, P. P. C. Lee, R. Li, L. Tang, Y.-C. Chen, and G. Zhang, "Sketchvisor: Robust network measurement for software packet processing," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 113–126.

[4] Z. Liu, A. Manousis, G. Vorsanger, V. Sekar, and V. Braverman, "One sketch to rule them all: Rethinking network flow monitoring with univmon," in *Proceedings of the 2016 ACM SIGCOMM Conference*, ser. SIGCOMM '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 101–114.

[5] C. Monsanto, J. Reich, N. Foster, J. Rexford, and D. Walker, "Composing software defined networks," in *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*. Lombard, IL: USENIX Association, Apr. 2013, pp. 1–13.

[6] M. Moshref, M. Yu, R. Govindan, and A. Vahdat, "Trumpet: Timely and precise triggers in data centers," in *Proceedings of the 2016 ACM SIGCOMM Conference*, ser. SIGCOMM '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 129–143.

[7] Y. Zhang, "An adaptive flow counting method for anomaly detection in sdn," in *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 25–30.

[8] R. Ben Basat, G. Einziger, R. Friedman, M. C. Luizelli, and E. Waisbard, "Constant time updates in hierarchical heavy hitters," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 127–140.

[9] C. Callegari, "Advanced statistical approaches for network anomaly detection (3 hours tutorial)," in *Conference on Security of Information and Networks (SIN 2009)*, vol. 6, 2009, p. 10.

[10] Q. Huang and P. P. C. Lee, "Ld-sketch: A distributed sketching design for accurate and scalable anomaly detection in network data streams," in *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, 2014, pp. 1420–1428.

[11] V. Sivaraman, S. Narayana, O. Rottenstreich, S. Muthukrishnan, and J. Rexford, "Heavy-hitter detection entirely in the data plane," in *Proceedings of the Symposium on SDN Research*, ser. SOSR '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 164–176.

[12] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *7th USENIX Symposium on Networked Systems Design and Implementation (NSDI 10)*. San Jose, CA: USENIX Association, Apr. 2010.

[13] A. Kabbani, M. Alizadeh, M. Yasuda, R. Pan, and B. Prabhakar, "Af-qcn: Approximate fairness with quantized congestion notification for multi-tenanted data centers," *2010 18th IEEE Symposium on High Performance Interconnects*, pp. 58–65, 2010.

[14] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Computers & Security*, vol. 28, no. 1-2, pp. 18–28, 2009.

[15] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. R. Ganger, G. A. Gibson, and B. Mueller, "Safe and effective fine-grained tcp retransmissions for datacenter communication," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, p. 303–314, Aug. 2009.

[16] Y. Xie, V. Sekar, D. Maltz, M. Reiter, and H. Zhang, "Worm origin identification using random moonwalks," in *2005 IEEE Symposium on Security and Privacy (S P'05)*, 2005, pp. 242–256.

[17] M. T. Arashloo, Y. Koral, M. Greenberg, J. Rexford, and D. Walker, "Snap: Stateful network-wide abstractions for packet processing," in *Proceedings of the 2016 ACM SIGCOMM Conference*, ser. SIGCOMM '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 29–43.

[18] N. Bandi, A. Metwally, D. Agrawal, and A. El Abbadi, "Fast data stream algorithms using associative memories," in *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 247–256.

[19] R. Schweller, A. Gupta, E. Parsons, and Y. Chen, "Reversible sketches for efficient and accurate change detection over network data streams," in *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 207–212.

[20] A. Kumar, M. Sung, J. J. Xu, and J. Wang, "Data streaming algorithms for efficient and accurate estimation of flow size distribution," *SIGMETRICS Perform. Eval. Rev.*, vol. 32, no. 1, p. 177–188, Jun. 2004.

[21] J. Sanjuàs-Cuxart, P. Barlet-Ros, N. Duffield, and R. R. Kompella, "Sketching the delay: Tracking temporally uncorrelated flow-level latencies," in *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, ser. IMC '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 483–498.

[22] X. Dimitropoulos, P. Hurley, and A. Kind, "Probabilistic lossy counting: An efficient algorithm for finding heavy hitters," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 1, p. 5, Jan. 2008.

[23] K. Cho, "Recursive lattice search: Hierarchical heavy hitters revisited," in *Proceedings of the 2017 Internet Measurement Conference*, ser. IMC '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 283–289.

[24] Y. Lin and H. Liu, "Separator: Sifting hierarchical heavy hitters accurately from data streams," in *Advanced Data Mining and Applications*, R. Alhajj, H. Gao, J. Li, X. Li, and O. R. Zaïane, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 170–182.

[25] G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava, "- finding hierarchical heavy hitters in data streams," in *Proceedings 2003 VLDB Conference*, J.-C. Freytag, P. Lockemann, S. Abiteboul, M. Carey, P. Selinger, and A. Heuer, Eds. San Francisco: Morgan Kaufmann, 2003, pp. 464–475.

[26] M. Mitzenmacher, T. Steinke, and J. Thaler, "Hierarchical heavy hitters with the space saving algorithm," *Society for Industrial and Applied Mathematics*, 2011.

[27] M. Moshref, M. Yu, R. Govindan, and A. Vahdat, "Scream: Sketch resource allocation for software-defined measurement," in *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '15. New York, NY, USA: Association for Computing Machinery, 2015.

[28] A. Gupta, R. Harrison, M. Canini, N. Feamster, J. Rexford, and W. Willinger, "Sonata: Query-driven streaming network telemetry," in *Proceedings of the 2018 Conference of the ACM Special*

Interest Group on Data Communication, ser. SIGCOMM '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 357–371.

[29] S. Narayana, A. Sivaraman, V. Nathan, P. Goyal, V. Arun, M. Alizadeh, V. Jeyakumar, and C. Kim, "Language-directed hardware design for network performance monitoring," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 85–98.

[30] S. Li, L. Luo, and D. Guo, "Sketch for traffic measurement: design, optimization, application and implementation," *CoRR*, vol. abs/2012.07214, 2020.

[31] Z. Zhou, D. Zhang, and X. Hong, "Rl-sketch: Scaling reinforcement learning for adaptive and automate anomaly detection in network data streams," in *2019 IEEE 44th Conference on Local Computer Networks (LCN)*, 2019.

[32] T. Yang, L. Wang, Y. Shen, M. Shahzad, Q. Huang, X. Jiang, K. Tan, and X. Li, "Empowering sketches with machine learning for network measurements," in *Proceedings of the 2018 Workshop on Network Meets AI & ML*, ser. NetAI'18. New York, NY, USA: Association for Computing Machinery, 2018, p. 15–20.

[33] "Caida anonymized internet traces 2018 dataset," http://www.caida.org/.

[34] G. Cormode and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications," *Journal of Algorithms*, vol. 55, no. 1, pp. 58–75, 2005.

[35] C. Estan and G. Varghese, "New directions in traffic measurement and accounting," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 4, p. 323–336, aug 2002.

[36] T. Li, S. Chen, and Y. Ling, "Per-flow traffic measurement through randomized counter sharing," *IEEE/ACM Transactions on Networking*, vol. 20, no. 5, pp. 1622–1634, 2012.

[37] T. Yang, H. Zhang, J. Li, J. Gong, S. Uhlig, S. Chen, and X. Li, "Heavykeeper: An accurate algorithm for finding top-k elephant flows," *IEEE/ACM Transactions on Networking*, vol. 27, no. 5, pp. 1845–1858, 2019.

[38] R. Ben Basat, G. Einziger, R. Friedman, and Y. Kassner, "Optimal elephant flow detection," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017, pp. 1–9.

[39] F. Wang and L. Gao, "Simple and efficient identification of heavy hitters based on bitcount," in *2019 IEEE 20th International Conference on High Performance Switching and Routing (HPSR)*, 2019.