RatioSketch: Towards More Accurate Frequency Estimation in Data Streams via a Lightweight Neural Network

Mengbo Wang[1, 2], **Zhuochen Fan**[2 ✉*], **Dayu Wang**[3], **Guorui Xie**[2], **Qing Li**[2 ✉], **Zeyu Luan**[2], **Yong Jiang**[1, 2], **Tong Yang**[3], **Mingwei Xu**[4]

[1]Tsinghua Shenzhen International Graduate School, Tsinghua University, China
[2]Department of Strategic and Advanced Interdisciplinary Research, Pengcheng Laboratory, China
[3]State Key Laboratory of Multimedia Information Processing, School of Computer Science, Peking University, China
[4]Institute for Network Sciences and Cyberspace, Tsinghua University, China
wmb25@mails.tsinghua.edu.cn, {fanzhch, xiegr, liq, luanzy}@pcl.ac.cn, victorwang030401@stu.pku.edu.cn, jiangy@sz.tsinghua.edu.cn, yangtong@pku.edu.cn, xumw@tsinghua.edu.cn

Abstract

Sketch-based solutions are widely used to estimate item frequencies in infinite data streams. Traditional hand-crafted sketches face the bottleneck of further eliminating errors because they cannot fully utilize the data stream distribution. Although recent neural sketches represented by MetaSketch and LegoSketch have improved generalization capabilities, they face bottlenecks such as high computational overhead and parameter sensitivity. Meanwhile, they ignore load information, fail to fully utilize the local information in hand-crafted sketches, and do not focus on the frequent items that are usually more important in data streams. In this paper, we propose RatioSketch, a novel lightweight neural network correction framework that synergizes the advantages of hand-crafted sketches and neural sketches in a "micro-correction" paradigm. The key idea is to retain the efficient underlying data structure of the hand-crafted sketch and to build a neural correction layer in its output space. We select multiple representative hand-crafted sketches as use cases to study the correction performance of RatioSketch on them. Extensive experimental evaluations on several real-world datasets show that RatioSketch-corrected sketches achieve consistently higher estimation accuracy than their uncorrected counterparts, as well as outperforming neural baselines such as MetaSketch and LegoSketch under identical memory budgets.

Code — https://github.com/pkufzc/RatioSketch

Introduction

As a basic task in data stream processing, frequency estimation aims to estimate the number of occurrences of individual items in infinite data streams and has been widely used in fields such as network measurement (Yang et al. 2018; Tang, Huang, and Lee 2019; Ding et al. 2023) and machine learning (Bifet et al. 2011; Tai et al. 2018; Zhang et al. 2024). Traditional hand-crafted sketch algorithms have been widely studied and recognized both in academia and industry due to their $O(1)$ time complexity, sublinear space complexity, and small estimation error. Although recent research trend of hand-crafted sketches is to reduce the error, their static architecture does not allow them to adapt to diverse data stream distributions, nor can they directly utilize the overall distributional patterns or temporal dynamics of data streams to further reduce errors. Once they need to be deployed in different data stream scenarios, their errors are likely to rise significantly, otherwise they must undergo complex and time-consuming tuning first, which limits their adaptability in real-world heterogeneous workloads.

Recent neural sketches attempt to address these limitations via learnable structures (Rae, Bartunov, and Lillicrap 2019), with the state-of-the-art (SOTA) solutions being MetaSketch (Cao, Feng, and Xie 2023; Cao et al. 2024) and LegoSketch (Feng et al. 2025). MetaSketch uses memory-augmented neural networks (MANNs) for frequency estimation. Specifically, it formulates sketch operations such as insert and query as differentiable memory accesses, and trains general models on synthetic distributions through meta-learning (Santoro et al. 2016) to achieve cross-distribution adaptation.

The key idea is to abstract item storage as neural write operations, allowing the network to implicitly learn collision-resolution behavior and thereby reduce errors under constrained memory. However, it faces critical limitations: 1) High training cost. Its cross-domain generalization relies on a large number of meta-task pre-training; 2) Poor scalability. Its fixed-size memory blocks hinder dynamic space budget adaptation. 3) High latency. Its inference requires full MANN forwarding.

To enhance the performance of MetaSketch, the recent LegoSketch introduces a modular architecture with composable memory blocks for elastic scaling, enabling flexible reconstruction of memory organization according to available space. It first improves scalability through normalized multi-hashing for zero-shot cross-domain transfer. Then, it improves accuracy by optimizing high space budget scenarios with self-guided loss and reconstructing global stream features with Deepsets-based (Zaheer et al. 2017) memory scanning.

However, its block-based architecture introduces high inference costs and performs poorly on highly skewed data

[*]✉ Corresponding authors.

streams.Its design does not consider the frequent items that are actually more important, nor does it make full use of local information, and its hard threshold is not flexible enough.

Furthermore, a core issue with leading solutions such as MetaSketch and LegoSketch lies in their inability to balance the tension between neural network expressiveness and stream processing efficiency.They clearly prioritize the former, focusing on representational power and flexibility, and as a result, their real-world performance in high-speed stream environments remains suboptimal, often incurring significant computation and latency costs.

In this paper, we propose RatioSketch, a novel lightweight neural network-based sketch correction framework that aims to significantly improve the frequency estimation performance of both traditional hand-crafted sketches and emerging neural sketches, without altering their underlying data structures or compromising update efficiency greatly. RatioSketch synergizes the advantages of hand-crafted sketches and neural sketches through a "micro-correction" paradigm, in which the neural component operates only in the output space of the base hand-crafted sketch to refine its estimates with minimal overhead. In other words, it retains the efficient underlying data structure of the base hand-crafted sketches and builds a neural correction layer in their output space. Compared with reconstructing the entire storage process (such as MANN-based solutions mentioned above), RatioSketch focuses on mining the multi-source information inside hand-crafted sketches, with the key techniques including:

- **Multi-Source Feature Fusion.** It integrates a Global Encoder capturing sketch matrix states, a local analyzer extracting proportional errors from hash collisions, and a structural adapter modeling item-slot relationships, forming a triple feature enhancement mechanism;
- **Ratio-Based Modeling.** It decouples estimation into the output of hand-crafted sketches and neural scaling coefficients via frequency-space transformation;
- **Differentiated Loss Weighting.** It imposes a log-frequency weighted penalty on each item, thereby emphasizing the correction of frequent items.

Contributions. 1) We propose RatioSketch, a novel lightweight neural network-based sketch framework applicable to diverse hand-crafted sketches, consistently correcting their frequency estimation accuracy for both all items and frequent items in arbitrary data streams. 2) RatioSketch addresses a series of problems such as high training overhead, weak scalability, weak generalization, and lack of support for frequent items, which were not addressed by previous neural sketches, through key technical innovations, and can achieve absolute accuracy advantage with only a small memory budget. 3) We also provide a rigorous theoretical analysis for the estimation error of RatioSketch. 4) Extensive experiments on six real-world datasets demonstrate that RatioSketch significantly outperforms SOTA methods with minimal training overhead. The source codes of RatioSketch are available at GitHub.

Related Work

Hand-Crafted Sketches

As a kind of probabilistic data structure, sketches are among the most appropriate and widely adopted solutions for data stream processing tasks such as frequency estimation for all items and frequent items. For per-item frequency estimation, the most classic sketch is the Count-Min Sketch (CMS) (Cormode and Muthukrishnan 2005a), which consists of d arrays $A_i(1 \leq i \leq d)$, each with a hash function $h_i(.)$ and w counters. When inserting an item e, CMS increments the d hashed counter $A_i[h_i(e)]$ by 1. When querying the (estimated) frequency of e, CMS reports the smallest value among the d hashed counters $A_i[h_i(e)]$. Similarly, the basic structure of the classic CUS (Estan and Varghese 2003) and CS (Charikar, Chen, and Farach-Colton 2002) is also $d \cdot w$ counters, but the item operations are different. Their errors are caused by hash collisions, where different items are hashed to the same counter, which is particularly severe with small memory budgets. In addition, the skewed distribution of data streams (Cormode and Muthukrishnan 2005b) causes most counters to record counts far less than their capacity. To avoid wasting space, the SOTA TowerSketch (Yang et al. 2023) allocates the same memory for each array but uses progressively smaller counters from higher to lower levels. This hierarchical design ensures that frequent items overflow into larger, high-precision counters at upper levels, while infrequent items remain tracked in smaller, low-cost counters at lower levels, achieving a better balance between accuracy and memory utilization. In addition, there are many SOTA sketches designed specifically for frequent-item frequency estimation[1] (Roy, Khan, and Alonso 2016; Yang et al. 2018; Li et al. 2020; Zhang et al. 2021). CMS, CUS, and CS can also be equipped with a small heap to accomplish this goal, allowing dynamic maintenance of the current top-k items in the stream.

Neural Sketches

Recent advances integrate neural networks with sketches to improve adaptability. One of the SOTA, MetaSketch (Cao, Feng, and Xie 2023; Cao et al. 2024), reformulates stream processing as neural memory operations and implements a memory-augmented neural network (MANN) (Weston, Chopra, and Bordes 2014; Graves, Wayne, and Danihelka 2014; Graves et al. 2016) with four learnable modules: embedding, addressing, memory, and decoding. It employs meta-training on synthetic Zipf (Powers 1998) streams for distributional robustness. However, its fixed-size memory matrix impedes dynamic budget scaling, making it difficult to adapt efficiently when the available memory or stream density changes over time. Another SOTA LegoSketch (Feng et al. 2025) introduces composable memory blocks for elastic scaling, enabling zero-shot domain transfer via normalized multi-hash embeddings. While signifi-

[1]This task is also known as finding top-k items and heavy-hitters, which aims to report the items with the k largest frequencies and the items with frequencies exceeding a given threshold, respectively, and they are often considered the same research problem.

cantly improving accuracy and scalability, it still incurs a large computational cost.

Methodology

Preliminaries

First, we outline the item frequency estimation problem in a standard data stream model. Consider a data stream $\mathcal{S} = (e_1, \dots, e_N)$ consisting of a total of N items with n distinct items. Each item $e_i \in \mathcal{S}$, comes from the item domain set $\mathbb{E} = \{E_1, \dots, E_n\}$, where the items in $\mathbb{E}$ are unique. The frequency f_i of the item E_i represents its number of occurrences in $\mathcal{S}$. Hence, the total frequencies, $i.e.$, the sum of all f_i, are exactly the length N of $\mathcal{S}$.

The design goal of RatioSketch is to make it theoretically possible to correct the frequency estimation results for any base hand-crafted sketch (whether it is the typical CMS or the SOTA TowerSketch). For illustration, we adopt the Count-Min Sketch (CMS) as the default base hand-crafted sketch (denoted BaseSketch) for correction, since it represents the most fundamental and widely studied sketch architecture. For E_i, when using BaseSketch to query its frequency, there are usually multiple estimated values to choose from. We use $\mathbf{v} = [v_1, \dots, v_d]$ to denote the estimated frequency for the d hashed positions ($i.e.$, d optional values) when querying E_i, and v_0 to denote the estimated value finally provided by BaseSketch.

RatioSketch Framework: Frequency Correction Network Based on Multi-Feature Fusion

Overview RatioSketch consists mainly of two parts: the underlying **BaseSketch** that performs standard counting operations, and a lightweight **Correction Module** that refines its output estimates through neural inference. The item insertion and storage operations (Figure 1 Actions) depend on BaseSketch itself. When querying the final estimated frequency, we invoke the Correction Module to make fine-grained corrections based on the intermediate values provided by BaseSketch, effectively performing a post-hoc neural calibration. Correction Module consists mainly of four parts: Global Information Module (Figure 1 Global Encoder), Local Information Module (Figure 1 Local Encoder), Load Information Module (Figure 1 Load Decoder) and Ratio Decoding Module (Figure 1 Ratio Decoder). Each encoder or decoder consists solely of several layers of fully connected neural networks with ReLU activation functions. An overview of the inputs and outputs of the corresponding encoder or decoder is provided in Equations 2-7. It is worth noting that the Global Information Module and the Load Information Module only need to be run once during the final batch query $\hat{f}_i$ and are independent of N, thus achieving high efficiency.

Global Information Module The Global Information Module extracts structural characteristics from the Storage Matrix (size $d \cdot w$) illustrated in Figure 1 to provide auxiliary information for frequency correction. To balance computational efficiency with comprehensive feature extraction, we select the first 64 counters from each row for processing. Our experimental results show that 64 counters offer a stable trade-off: fewer counters fail to capture sufficient global load patterns, while more counters introduce additional overhead without bringing measurable accuracy gains.

The selected counter values are normalized as $value_{\text{norm}} = \frac{value \cdot \alpha}{N}$ (where α is an amplification factor) to ensure effective neural network processing. These normalized values are processed through the Global Encoder to derive hidden info, which are subsequently processed to extract the final global info.

The resulting global information encapsulates critical data stream characteristics, including distribution patterns, load ratios, thereby providing essential information for the Ratio Decoder.

Local Information Module This module focuses on the local values $\mathbf{v} = [v_1, \dots, v_d]$ at the d hashed positions corresponding to a single item, capturing the fine-grained local variations caused by hash collisions and counter saturation. We aim for this module to explore whether the current item is frequent or infrequent, as manifested by the differences among the v_j values, which implicitly denote the degree of interference from other items. Theoretically, the v_0 of frequent items is less affected by other items, leading to smaller fluctuations in r_j (to be defined below). In contrast, the v_0 of infrequent items is more significantly influenced by other items, resulting in larger fluctuations in r_j. We expect the Local Encoder to utilize this information to assist the operation of the Ratio Decoder.

For the data processing DP(.), we first sort $\mathbf{v}$ from small to large to get $v_1 \le v_2 \le \dots \le v_d$. Then, the ratio feature r_j is calculated for each position, with the denominator being the final estimated frequency v_0 of BaseSketch:

$$r_j = \frac{v_j - v_0}{v_0}, \quad j = 1, 2, \dots, d \tag{1}$$

We further extract statistics such as the mean (μ_r), variance (σ_r^2), and median (m_r) of r_j. Meanwhile, the frequency weight $f_w = \frac{w \cdot v_0}{N}$ and other possible features are concatenated into $[r_1, r_2, \dots, r_d, \mu_r, \sigma_r^2, m_r, f_w]$ as the input of Local Encoder. After these features are encoded, they are converted into local feature vectors and input into the Ratio Decoder.

Sorting helps the network learn from inputs of different magnitudes, automatically assigning appropriate weights to positions like maximum/minimum values. Statistical compression through mean, variance, and median helps capture local distribution patterns, ultimately providing fine-grained local information for BaseSketch correction.

Load Information Module In supervised learning, the learning goal of this module is $\frac{n}{w}$, which is called $load_r$ (Figure 1 Load Info) to represent the severity of counter slot conflicts. To achieve this goal, the Load Decoder senses the hidden features output by the Global Encoder (Figure 1 Hidden Info) and outputs $load_r$. Since w of BaseSketch is known, n can be predicted by calculating $load_r \cdot w$. Meanwhile, since we predict a ratio $\frac{n}{w}$ instead of a specific n, we can make predictions on BaseSketch with any different memory size. $\frac{n}{w}$

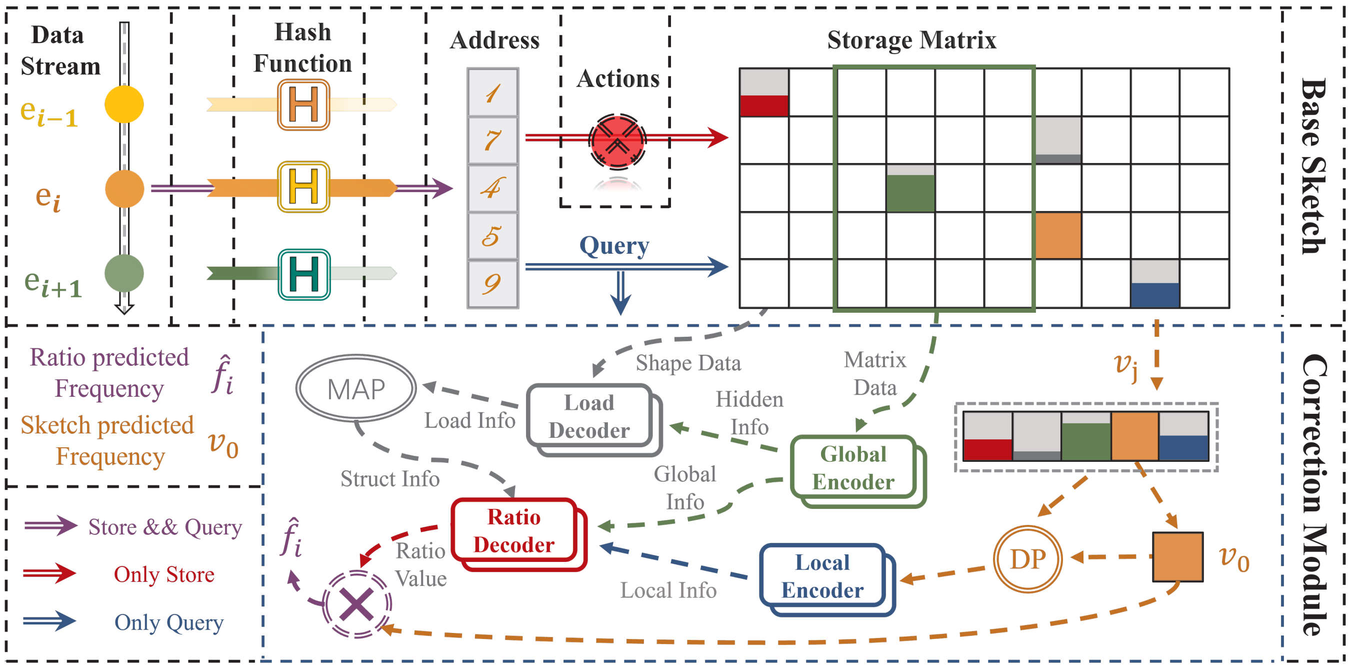

Figure 1: The Framework of RatioSketch.

has a small variation range and is less affected by the range of neural network parameters.

Even if w varies from hundreds of thousands to millions, it still performs well. This is impossible for those solutions (such as LegoSketch) that directly use neural networks to predict n.

Feature Fusion and Ratio Decoding Module Ultimately, the model splices global information, local sorting enhancement features, and structural parameters through the Ratio Decoder, which outputs the ratio correction coefficient C with a decoding frequency of:

$$\hat{f} = C \cdot v_0 \tag{2}$$

Where $C = \text{RatioDec}(\cdot)$ is output by the above feature fusion network. The whole process is shown in Fig. 1:

$$G, G_{hidden} = \text{GlobalEnc}\left(\text{Norm}\left(memory\right)\right) \tag{3}$$

$$L = \text{LocalEnc}\left(\text{DP}\left(\mathbf{v}, v_0\right)\right) \tag{4}$$

$$load_r = \text{LoadDec}\left(G_{hidden}\right) \tag{5}$$

$$S = \left(\frac{load_r}{d}, \frac{d \cdot w}{N}, \frac{load_r \cdot w}{N}\right) \tag{6}$$

$$C = \text{RatioDec}\left(L \odot G, S\right) \tag{7}$$

Where: G and G_{hidden} are global and hidden feature information, respectively, which are extracted by Global Encoder GlobalEnc after being scaled by Storage Matrix; DP represents the corresponding operation described in the Local Information Module on $\mathbf{v}$, and finally obtains the vector $[r_1, r_2, \ldots, r_d, \mu_r, \sigma_r^2, m_r, f_w]$; L is local information, which is extracted by the Local Encoder LocalEnc from the vector provided by the DP operation; S is structural information (Figure 1 Struct Info), which is maped (Figure 1 MAP) from $load_r$ and BaseSketch size information; and $\odot$ represents item-by-item multiplication (fusion of global and local information).

Summary of Theoretical Advantages Our RatioSketch greatly improves the generalization, structural adaptability and robustness of the model through global and local information fusion, proportionality and load perception mechanisms, and can achieve high-accuracy frequency correction under any BaseSketch structure and data stream distribution.

Loss Function Design

The loss function of RatioSketch uses a weighted loss based on ratio error[2], as follows: For all items, a logarithmic weighted ratio error is used:

$$\text{RatioLoss} = \frac{1}{n}\sum_{i=1}^{n} \log_2(f_i + 1) \cdot \left| \frac{\hat{f}_i}{v_0^i} - \frac{f_i}{v_0^i} \right| \tag{8}$$

The logarithmic weight $\log_2(f_i + 1)$ makes frequent items account for a larger proportion of the loss, improving the accuracy of their correction.

In addition, in order to ensure the rationality of structural parameter estimation and further improve generalization, the item size structural constraint is introduced:

$$\text{LoadLoss} = \left| \frac{w \cdot load_r}{n} - 1 \right| \tag{9}$$

In addition, we use a self-adjusting weight assignment function $\mathcal{LW}$ (Kendall, Gal, and Cipolla 2018) to balance the

[2]Our ratio loss (instead of the ARE/AAE combined adaptive weighting used by SOTA) is verified to be more stable and effective, with faster training convergence.

relationship between RatioLoss and LoadLoss, which is expressed as follows:

$$\mathcal{LW} = \sum_{i=1}^{\mathcal{N}} \left(\frac{1}{2s_i^2}\mathcal{L}_i + \log(1+s_i^2) \right) \tag{10}$$

where $\mathcal{N}$ is the number of loss items, $\mathcal{L}_i$ is the original loss of the i-th item, s_i is the learnable weight parameter of the i-th item and can be automatically learned through backpropagation. Hence, the final total loss is:

$$\mathcal{L} = \mathcal{LW}(\mathrm{RatioLoss}, \mathrm{LoadLoss}) \tag{11}$$

Adaptation to different BaseSketch Firstly, RatioSketch can further optimize its correction for frequent items estimated by CMS with heap. Specifically, we use $heapflag$ to mark whether an item is in the heap: if it is, the item is at the head of the data stream; otherwise, it is at the tail. In feature processing, we concatenate $heapflag$ with features such as $\mathrm{DP}(v, v_0)$ and input them into the Local Encoder together. In the loss function, we also set different loss functions: for those in the head, the loss function is not changed, and $\log_2(f_i+1)$ is still used to emphasize frequent items; for those in the tail, $\log_2(f_i+1)$ is replaced by 1, so that the final loss function becomes:

$$\mathcal{L} = \mathcal{LW}(\mathrm{RatioLoss_h}, \mathrm{RatioLoss_t}, \mathrm{LoadLoss}) \tag{12}$$

Secondly, each of the d hash functions of TowerSketch corresponds to multiple counters of different sizes. We select some counters from small to large and splice them together in a specified order for better input into the Global Encoder.

Thirdly, RatioSketch allows manual control of ratio ranges as needed. This configurable range lets practitioners set appropriate bounds for different BaseSketch structures and scenarios.

Training and Inference

The training and inference procedures for RatioSketch are described in Algorithm 1. During the training phase, RatioSketch is trained on synthetic datasets and fine-tuned using real-world datasets or past samples from historical data streams. During the inference phase, we only need to extract features from the underlying BaseSketch and input them into the correction model to obtain a more accurate frequency estimation.

Algorithm 1: RatioSketch training and reasoning procedures

```
Require: multiple arbitrary data streams $\mathcal{S} = \{(E_i, f_i)\}$,
    BaseSketch $\mathcal{M}$, RatioSketch $\mathcal{R}$
 1: initialize $\mathcal{M}$
 2: for each $\mathcal{S}$ do
 3:     Insert $\mathcal{S}$ data into $\mathcal{M}$, obtain multiple estimates $v_j^i$
        and final estimates $v_0^i$ for each item $E_i$;
 4:     Extract global information $G$, local information $L_i$,
        structural information $S$;
 5:     Calculate input features $F_i = [L_i \odot G, S]$;
 6:     Calculate correction coefficient $C_i = \mathcal{R}(F_i)$;
 7:     Predict frequency $\hat{f}_i = C_i \cdot v_0^i$;
 8:     Calculate loss $(\hat{f}_i, f_i)$;
 9:     Back propagate and update model parameters;
10: end for
11: Inference phase: Repeat steps 3-7 for the new data
    stream without updating parameters
```

Mathematical Analysis

Problem Setup

We consider the Count-Min Sketch (CMS) framework as our foundational sketch structure. We assume CMS employs d independent hash functions and a $d \cdot w$ counter matrix.

Under this setup, each sketch value can be modeled as:

$$v_j = f + X_j$$

where f is the true frequency of the item and X_j represents the hash collision error at position j. The collision errors $\{X_j\}_{j=1}^d$ are typically non-negative due to the additive nature of hash collisions in CMS.

Analysis of Feature Engineering

The superior performance of RatioSketch stems from its principled feature engineering design. Our framework is built upon three key theoretical insights that ensure both robustness and generalization across diverse data distributions and sketch configurations.

First, our ratio-based feature design achieves scale invariance, enabling the same model to handle frequencies spanning multiple orders of magnitude without retraining. Second, the statistical compression of local features provides distributional stability, ensuring consistent performance regardless of the underlying data characteristics. Third, our load-aware mechanism guarantees structural adaptability, allowing seamless deployment across different sketch sizes and memory constraints.

Theorem 1. ***(Scale Invariance of Ratio Features)*** *Under the Count-Min Sketch framework, the ratio features $r_i = \frac{v_i - v_0}{v_0}$ exhibit exact scale invariance with respect to the true frequency f.*

Proof. In the CMS framework, each counter value is $v_i = f + X_i$, where $X_i = \sum_{k:h_i(k)=h_i(\text{item})} f_k$ represents the collision error from other items hashing to the same position. The CMS estimate is $v_0 = \min_j v_j = f + X_{(1)}$ where $X_{(1)} = \min_j X_j$.

The ratio feature becomes:

$$r_i = \frac{v_i - v_0}{v_0} = \frac{(f + X_i) - (f + X_{(1)})}{f + X_{(1)}} = \frac{X_i - X_{(1)}}{f + X_{(1)}}$$

For any scaling $f \to \alpha f$, the collision errors scale proportionally: $X_i \to \alpha X_i$ (since all frequencies scale). Thus:

$$r_i' = \frac{\alpha X_i - \alpha X_{(1)}}{\alpha f + \alpha X_{(1)}} = \frac{\alpha(X_i - X_{(1)})}{\alpha(f + X_{(1)})} = \frac{X_i - X_{(1)}}{f + X_{(1)}} = r_i$$

Therefore, the ratio features exhibit exact scale invariance under proportional scaling of all frequencies. □

Theorem 2. ***(Stability of Statistical Moments)*** *In the CMS framework, the statistical moments (μ_r, σ_r^2, m_r) of ratio features $\{r_i\}$ are stable with bounded variance for reasonable load factors.*

Proof. In CMS, due to hash function independence, $\mathbb{E}[X_i] = \frac{N-f}{w}$ where N is the stream size. The mean ratio becomes:

$$\mu_r = \frac{1}{d}\sum_{i=1}^{d} r_i = \frac{\bar{X} - X_{(1)}}{f + X_{(1)}}$$

As $d \to \infty$, by the law of large numbers, $\bar{X} \to \frac{N-f}{w}$. For the minimum statistic $X_{(1)}$, it converges to a stable value dependent on the collision distribution rather than zero.

Therefore:

$$\mathbb{E}[\mu_r] \approx \frac{\frac{N-f}{w} - \mathbb{E}[X_{(1)}]}{f + \mathbb{E}[X_{(1)}]}$$

The variance analysis shows $\mathrm{Var}(\mu_r) = O(\frac{1}{d(f+\mathbb{E}[X_{(1)}])^2})$, which decreases with both d and f, ensuring statistical stability. Similar bounds hold for σ_r^2 and m_r. □

Theorem 3. ***(Load Ratio Consistency)*** *In CMS, the empirical load ratio $\hat{\rho} = \frac{\hat{n}}{w}$ converges to the true load ratio $\rho = \frac{n}{w}$ at rate $O(\sqrt{\log w/w})$.*

Proof. In CMS, each counter j is non-empty with probability $p = 1 - (1 - \frac{1}{w})^n \approx 1 - e^{-n/w}$. The number of distinct items can be estimated as:

$$\hat{n} = -w \ln\left(\frac{\text{number of empty counters}}{w}\right)$$

Let Z be the number of empty counters. Then $Z \sim \text{Binomial}(w, e^{-n/w})$. By standard concentration inequalities:

$$P\left(\left|\frac{Z}{w} - e^{-n/w}\right| \geq \epsilon\right) \leq 2e^{-2w\epsilon^2}$$

Applying the delta method to the logarithmic transformation and solving for ϵ yields the convergence rate $O(\sqrt{\log w/w})$ for $\hat{\rho} = \frac{\hat{n}}{w}$. □

Analysis of the Training Process

Our method provides strong theoretical foundations for end-to-end learning with neural networks. First, we rigorously demonstrate that our transformation preserves (nearly) all Fisher information, establishing the fundamental information-theoretic capacity of our approach. Building upon this, we prove that the training process is well-behaved, that is, the optimization landscape is smooth and the model converges efficiently. Furthermore, we show formal generalization bounds for the learned models and illustrate their robustness guarantees under small input perturbations. These results together provide a comprehensive theoretical understanding of why our neural sketch approach is both powerful and widely applicable.

The analysis of Fisher Information is crucial for establishing the information-theoretic capacity of our method. By rigorously demonstrating that our feature transformation preserves a significant amount of the original Fisher Information, we formally validate that the subsequent neural network is provided with nearly all the essential information required to accurately estimate the true frequency. This preservation ensures that the learning process is fundamentally well-posed and has the potential to converge to a high-accuracy solution, confirming that our feature engineering is not only intuitive but also theoretically sound.

Theorem 4. ***(Fisher Information Preservation)*** *Under the CMS framework with reasonable load conditions, sorting hash values and employing ratio features preserve at least $(1 - O(\sqrt{\log d}/d))$ of the Fisher information for estimating the true frequency f. Statistical moments (μ_r, σ_r^2, m_r) retain essential information with only $O(\log d/d)$ loss.*

Proof. In CMS, each counter value is $v_i = f + X_i$ where X_i represents collision errors. Under reasonable independence assumptions for hash functions, the Fisher information for estimating f from the original counters is approximately:

$$I_{\mathbf{v}}(f) \approx \sum_{i=1}^{d} \frac{1}{\mathrm{Var}(v_i)} = \sum_{i=1}^{d} \frac{1}{\mathrm{Var}(X_i)}$$

Sorting preserves the total information content but changes the joint distribution structure. The ratio transformation $r_i = \frac{v_i - v_0}{v_0}$ removes the absolute scale dependency while maintaining relative differences.

By the delta method, the information loss from the ratio transformation is bounded by the inverse of the minimum eigenvalue of the transformation Jacobian. For well-conditioned cases where v_0 is not too small, this loss is $O(\sqrt{\log d}/d)$.

The further compression to statistical moments (μ_r, σ_r^2, m_r) loses additional information. Under the assumption that ratios are approximately Gaussian (by CLT for large d), these three moments capture most of the distributional information, with loss bounded by $O(\log d/d)$. □

Theorem 5. ***(Training Smoothness and Convergence)*** *Under the weighted ratio loss with logarithmic weighting, the training process converges with rate dependent on the frequency distribution. For bounded ratio features, stochastic gradient descent achieves:*

$$\mathbb{E}[\|\nabla\mathcal{L}(\theta_t)\|^2] \leq C \cdot \rho^t$$

where C and $\rho < 1$ depend on the weight distribution $\{\log_2(f_i + 1)\}$.

Proof. The weighted ratio loss has gradients:

$$\nabla\text{RatioLoss} = \frac{1}{n}\sum_{i=1}^{n} \log_2(f_i + 1) \cdot \text{sign}\left(\frac{\hat{f}_i - f_i}{v_0^i}\right) \cdot \nabla\hat{f}_i$$

The logarithmic weights $\log_2(f_i + 1)$ are bounded for practical frequency ranges, ensuring gradient boundedness.

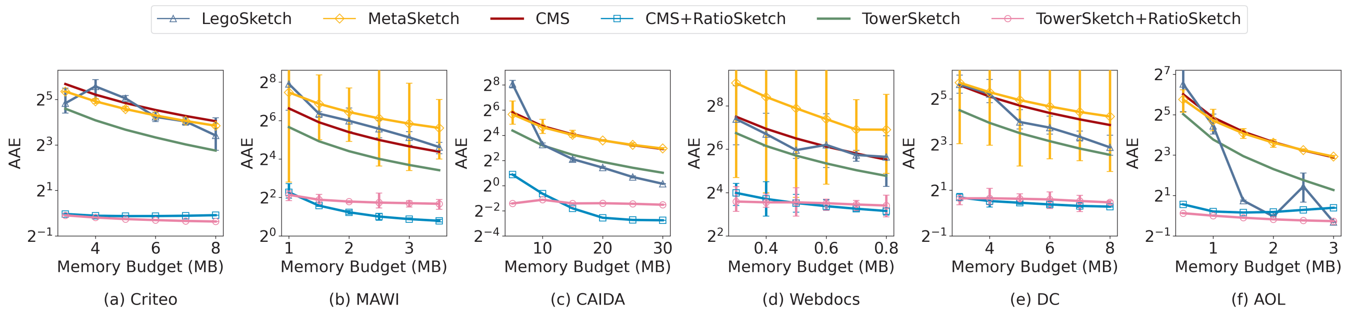

Figure 2: AAE for All Items on Six Real-World Datasets.

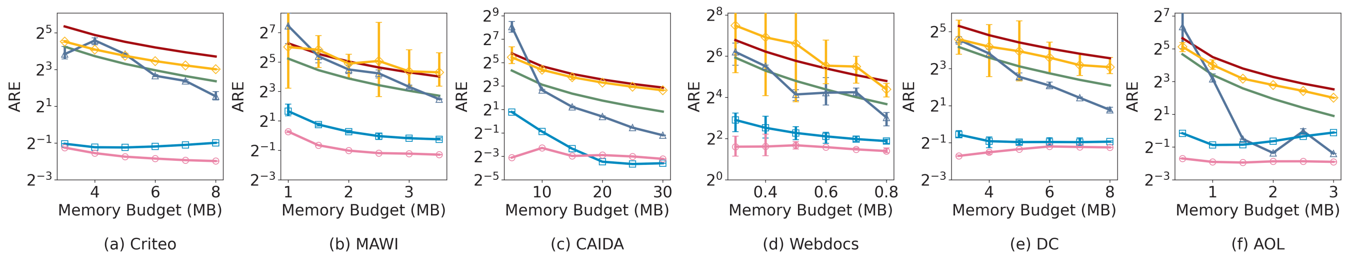

Figure 3: ARE for All Items on Six Real-World Datasets.

The self-adaptive weighting $\mathcal{LW}$ automatically balances the ratio and load losses, providing stable optimization dynamics. The combination of bounded features and adaptive weighting leads to convergence guarantees. □

Theorem 6. ***(Generalization Bound)*** *Under the weighted loss function, with probability at least $1 - \delta$:*

$$\mathcal{L}(\hat{h}) - \mathcal{L}(h^*) \leq O\left(\sqrt{\frac{H\log(dw) + \log(1/\delta)}{m}}\right)$$

where $H = \mathbb{E}[\log_2(y+1)]$ captures the frequency distribution's complexity.

Proof. The logarithmic weighting creates a non-uniform sampling effect, where frequent items contribute more to the loss. This modifies the Rademacher complexity by a factor related to the weight distribution. The effective sample complexity becomes frequency-dependent, with the bound scaling with $H = \mathbb{E}[\log_2(y+1)]$. The multi-task nature (ratio + load losses) adds logarithmic factors to account for the joint optimization. □

Theorem 7. ***(Robustness Bound)*** *Under the weighted ratio loss, for inputs S, S' differing by one item with frequency Δf:*

$$|\hat{f}(S') - \hat{f}(S)| \leq L_{net} \cdot \log_2(\Delta f + 1) \cdot O\left(\frac{\Delta f}{v_0}\right)$$

where the logarithmic weight amplifies the sensitivity for significant frequency changes.

Proof. The change in weighted loss due to inserting one item with frequency Δf is:

$$\Delta\mathcal{L} = \log_2(\Delta f + 1) \cdot \left|\frac{\Delta \hat{f}}{v_0}\right|$$

For frequent items, $\log_2(\Delta f + 1)$ can be large, amplifying the impact on the loss and consequently on the model's output. The final robustness bound includes this logarithmic amplification factor, showing that the method is more sensitive to changes in frequent items, which aligns with the design goal of prioritizing accuracy for frequent items. □

Experiments

Experimental Setup

Datasets Six real-world datasets are used in the experiment: Webdocs (Lucchese et al. 2004), MAWI (Cho, Mitsuya, and Kato 2000), DC (Benson, Akella, and Maltz 2010), AOL (Pass, Chowdhury, and Torgeson 2006), Criteo (Diemert et al. 2017), and CAIDA. The statistical information of each dataset is shown in Table 1. Each dataset is equally divided into a training set and a test set. In the training phase, the main model is first trained on synthetic Zipf distribution datasets. The Zipf parameter (skewness) range of the synthetic data is $[0.1, 1.5]$, and the number of items ranges from $[10{,}000, 100{,}000]$. The generation method and parameter distribution of synthetic data are consistent with real-world data streams to improve the generalization ability of the model. During the training phase, the model is verified on real-world datasets, and historical stream data can be used for fine-tuning to achieve better correction and facilitate actual deployment.

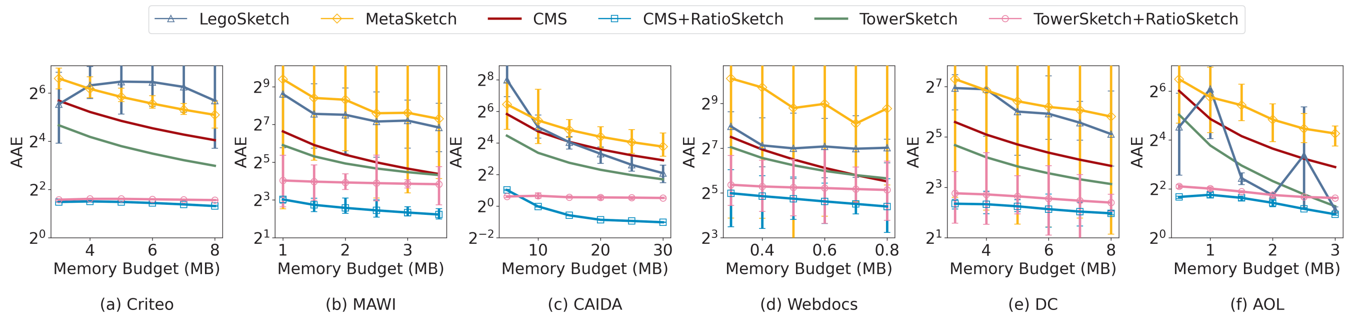

Figure 4: AAE for Frequent Items on Six Real-World Datasets.

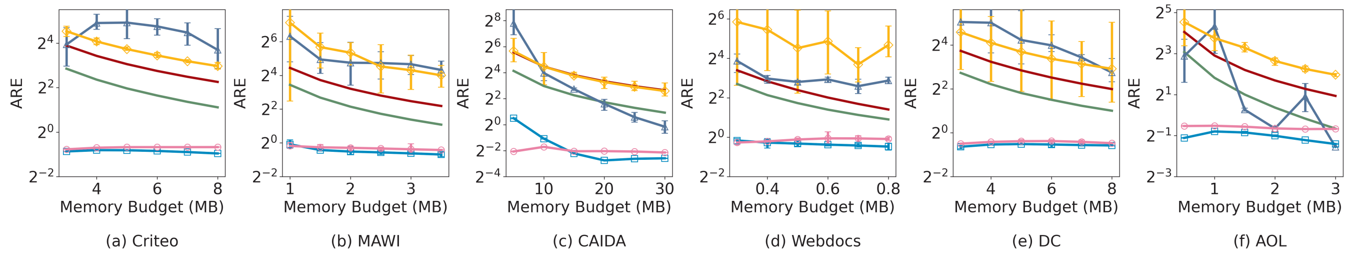

Figure 5: ARE for Frequent Items on Six Real-World Datasets.

Dataset	n	N	Zipf Skewness
Webdocs	812638	10000000	1.29
MAWI	4185350	23352934	0.80
DC	7907345	19855388	0.67
AOL	1717043	3614506	0.63
Criteo	9023599	16468027	0.56
CAIDA	24701616	33503292	0.21

Table 1: Statistics of real-world datasets

Baselines To comprehensively evaluate the correction capability and applicability of RatioSketch, we select multiple sketches for comparison. Firstly, we adopt the classic CMS and the SOTA TowerSketch as the underlying BaseSketch data structures used by RatioSketch, and they are also used as baselines. In other words, RatioSketch is integrated into the above-mentioned BaseSketches as an additional correction module to correct their frequency estimation results. Secondly, MetaSketch and LegoSketch, as the SOTA neural sketches, are naturally used as baselines.

Memory Budget During the training phase, in order to improve the adaptability of the model under different memory budgets, each round of tasks will dynamically allocate the memory size for BaseSketch. Specifically, according to n in the current training batch, a value in the memory range $[\frac{n}{16}, n \cdot 32]$ is randomly selected as the total memory M (in KB); Then, $w = \frac{M \cdot 1024}{4 \cdot d}$ is set, where $d = 3$ (division by 4 for 32-bit counters). This strategy ensures that the model workscan work effectively under various memory constraints. During the evaluation phase, a fixed-size memory is used for testing. LegoSketch and MetaSketch use many memory blocks, each of size 100KB, and the block count is determined by total allocated memory. Each item is randomly mapped to one block for insertion/query.

Parameters The core architecture details are as follows: The Global Encoder contains four Linear-ReLU layers ($1 \rightarrow 16 \rightarrow 16 \rightarrow 16 \rightarrow 16$). The Load Decoder and Local Information Encoder each include three Linear-ReLU layers, with dimensions ($16 \rightarrow 16 \rightarrow 16 \rightarrow 1$) and ($8 \rightarrow 16 \rightarrow 16 \rightarrow 16$), respectively. The Ratio Decoder has four layers ($19 \rightarrow 16 \rightarrow 16 \rightarrow 16 \rightarrow 1$), where the first three use ReLU and the final layer uses a Sigmoid activation to produce the output ratio. As a result of this compact design, RatioSketch achieves a compact model size of only 11KB, significantly smaller than LegoSketch (20KB) and much more efficient than MetaSketch, whose parameter count scales with memory size and requires approximately $2\times$ memory matrix storage due to its slot-dependent mechanism.

Evaluation Metrics We use the two most common error metrics, Average Absolute Error (AAE) and Average Relative Error (ARE), which are defined as follows:

$$AAE(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n}\left|\hat{f}_i - f_i\right|, ARE(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|\hat{f}_i - f_i\right|}{f_i}$$

In addition, we calculate AAE/ARE separately for the items ranked in the top 20% of frequency to evaluate the estimation ability of frequent items, denoted by Top AAE/ARE. Finally, we evaluate speed using throughput, which is defined as millions of operations per second (Mops).

Accuracy and Speed

AAE/ARE for All Items As shown in Figures 2-3, the results show that the two corrected schemes CMS+RatioSketch and TowerSketch+RatioSketch, have the best AAE/ARE on all datasets, which are significantly better than the uncorrected BaseSketches (CMS and TowerSketch), and clearly better than the two SOTA neural sketches (LegoSketch and MetaSketch). It is worth noting that our corrected schemes have a more obvious advantage when the memory budget is small. For example, under the minimum memory budget of 5MB of the CAIDA dataset (the most commonly used dataset for hand-crafted sketches): the AAE of CMS+RatioSketch (TowerSketch+RatioSketch) is 148.2× (726.7×), 27.7× (135.9×), 31.0× (152.2×), and 11.3× (55.6×) lower/better than those of LegoSketch, MetaSketch, CMS, and TowerSketch, respectively; the ARE of CMS+RatioSketch (TowerSketch+RatioSketch) is 155.3× (2301.5×), 25.5× (377.3×), 32.2× (477.1×), and 11.6× (172.4×) lower/better than those of LegoSketch, MetaSketch, CMS, and TowerSketch, respectively. This is significant because in actual deployment, memory resources are often scarce, and memory usage usually needs to be considered first.

AAE/ARE for Frequent Items As shown in Figures 4-5, the results show that the corrected schemes CMS+RatioSketch and TowerSketch+RatioSketch still have the optimal AAE/ARE among all schemes on all datasets, verifying that the correction of RatioSketch specifically for frequent items is effective. Specifically, under the minimum memory budget of 5MB of the CAIDA dataset: the AAE of CMS+RatioSketch (TowerSketch+RatioSketch) is 125.3× (166.6×), 42.8× (56.9×), 28.0× (37.2×), and 10.8× (14.4×) lower/better than those of LegoSketch, MetaSketch, CMS, and TowerSketch, respectively; the ARE of CMS+RatioSketch (TowerSketch+RatioSketch) is 158.6× (934.2×), 36.0× (211.9×), 33.8× (199.0×), and 12.5× (73.6×) lower/better than those of LegoSketch, MetaSketch, CMS, and TowerSketch, respectively.

Throughput We evaluate the throughput of all schemes in terms of store and query operations using the CAIDA dataset on both CPU and GPU platforms. As shown in Figure 6, the results show that the store throughput of the corrected schemes CMS+RatioSketch and TowerSketch+RatioSketch is comparable to that of the uncorrected schemes CMS and TowerSketch, and is clearly better than that of LegoSketch and MetaSketch; While their query throughput is somewhat reduced due to the additional forward propagation calculation, it is still acceptable.

Ablation Study

To evaluate the impact of each feature module in RatioSketch, we design ablation experiments: remove the global information, local information, and structural information (processed load information) branches, respectively, and compare them with the complete model. We conduct tests on three datasets: CAIDA, DC, and Webdocs, with Zipf skewnesses of 0.21, 0.67, and 1.29, respectively, to verify the effectiveness of each module under different Zipf distributions. As shown in Figure 7, removing any feature module leads to performance degradation, confirming that all three are crucial for enhancing the model's generalization ability and correction accuracy.

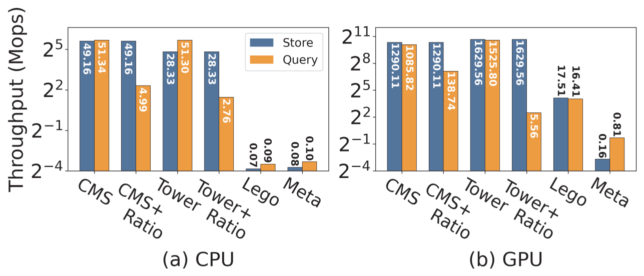

Figure 6: Throughput Comparison.

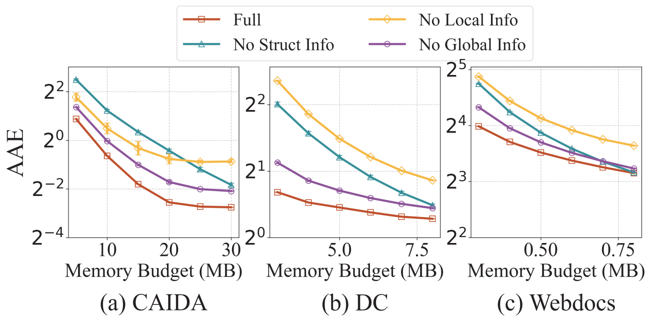

Figure 7: Ablation Study.

Conclusion

In this paper, we propose a novel lightweight neural network-based sketch correction framework called RatioSketch, which aims to overcome the challenges of SOTA neural sketches. RatioSketch establishes itself as a universally compatible corrector for diverse hand-crafted sketches, intrinsically boosting their frequency estimation accuracy. The advantages of RatioSketch are theoretically proven through rigorous mathematical analysis. Our extensive experiments show that RatioSketch-corrected CMS and TowerSketch have significantly better AAE and ARE than their uncorrected versions, as well as better than the SOTA neural sketches such as MetaSketch and LegoSketch, with minimal memory overhead.

Acknowledgments

This work is supported by the Basic and Frontier Research Project of PCL under grant No. 2025QYB004, the National Natural Science Foundation of China (NSFC) under grant No. 62402012, the Major Key Project of PCL under grant No. PCL2025A09, and the Shenzhen Key Lab of Software Defined Networking under grant No. ZDSYS20140509172959998.

References

Benson, T.; Akella, A.; and Maltz, D. A. 2010. Network traffic characteristics of data centers in the wild. In *IMC*, 267–280.

Bifet, A.; Holmes, G.; Pfahringer, B.; and Gavalda, R. 2011. Detecting sentiment change in Twitter streaming data. In *WAPA*, 5–11.

Cao, Y.; Feng, Y.; Wang, H.; Xie, X.; and Zhou, S. K. 2024. Learning to sketch: A neural approach to item frequency estimation in streaming data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(11): 7136–7153.

Cao, Y.; Feng, Y.; and Xie, X. 2023. Meta-sketch: A neural data structure for estimating item frequencies of data streams. In *AAAI*, volume 37, 6916–6924.

Charikar, M.; Chen, K.; and Farach-Colton, M. 2002. Finding frequent items in data streams. *Theoretical Computer Science*, 312(1): 3–15.

Cho, K.; Mitsuya, K.; and Kato, A. 2000. Traffic data repository at the WIDE project. In *USENIX ATC*.

Cormode, G.; and Muthukrishnan, S. 2005a. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1): 58–75.

Cormode, G.; and Muthukrishnan, S. 2005b. Summarizing and mining skewed data streams. In *SDM*, 44–55.

Diemert, E.; Meynet, J.; Galland, P.; and Lefortier, D. 2017. Attribution modeling increases efficiency of bidding in display advertising. In *AdKDD*, 1–6.

Ding, R.; Yang, S.; Chen, X.; and Huang, Q. 2023. Bitsense: Universal and nearly zero-error optimization for sketch counters with compressive sensing. In *SIGCOMM*, 220–238.

Estan, C.; and Varghese, G. 2003. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Transactions on Computer Systems*, 21(3): 270 – 313.

Feng, Y.; Cao, Y.; Wang, H.; Xie, X.; and Zhou, S. K. 2025. Lego Sketch: A Scalable Memory-augmented Neural Network for Sketching Data Streams. In *ICML*.

Graves, A.; Wayne, G.; and Danihelka, I. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Graves, A.; Wayne, G.; Reynolds, M.; Harley, T.; Danihelka, I.; Grabska-Barwińska, A.; Colmenarejo, S. G.; Grefenstette, E.; Ramalho, T.; Agapiou, J.; et al. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626): 471–476.

Kendall, A.; Gal, Y.; and Cipolla, R. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, 7482–7491.

Li, J.; Li, Z.; Xu, Y.; Jiang, S.; Yang, T.; Cui, B.; Dai, Y.; and Zhang, G. 2020. Wavingsketch: An unbiased and generic sketch for finding top-k items in data streams. In *SIGKDD*, 1574–1584.

Lucchese, C.; Orlando, S.; Perego, R.; and Silvestri, F. 2004. WebDocs: a real-life huge transactional dataset. In *FIMI*, volume 126.

Pass, G.; Chowdhury, A.; and Torgeson, C. 2006. A picture of search. In *InfoScale*, 1–es.

Powers, D. M. 1998. Applications and explanations of Zipf's law. In *CoNLL*, 151–160.

Rae, J.; Bartunov, S.; and Lillicrap, T. 2019. Meta-learning neural bloom filters. In *ICML*, 5271–5280.

Roy, P.; Khan, A.; and Alonso, G. 2016. Augmented sketch: Faster and more accurate stream processing. In *SIGMOD*, 1449–1463.

Santoro, A.; Bartunov, S.; Botvinick, M.; Wierstra, D.; and Lillicrap, T. 2016. Meta-learning with memory-augmented neural networks. In *ICML*, 1842–1850.

Tai, K. S.; Sharan, V.; Bailis, P.; and Valiant, G. 2018. Sketching linear classifiers over data streams. In *SIGMOD*, 757–772.

Tang, L.; Huang, Q.; and Lee, P. P. 2019. Mv-sketch: A fast and compact invertible sketch for heavy flow detection in network data streams. In *INFOCOM*, 2026–2034.

Weston, J.; Chopra, S.; and Bordes, A. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Yang, K.; Long, S.; Shi, Q.; Li, Y.; Liu, Z.; Wu, Y.; Yang, T.; and Jia, Z. 2023. SketchINT: Empowering INT with TowerSketch for per-flow per-switch measurement. *IEEE Transactions on Parallel and Distributed Systems*, 34(11): 2876–2894.

Yang, T.; Jiang, J.; Liu, P.; Huang, Q.; Gong, J.; Zhou, Y.; Miao, R.; Li, X.; and Uhlig, S. 2018. Elastic sketch: Adaptive and fast network-wide measurements. In *SIGCOMM*, 561–575.

Zaheer, M.; Kottur, S.; Ravanbakhsh, S.; Poczos, B.; Salakhutdinov, R. R.; and Smola, A. J. 2017. Deep sets. *NeurIPS*, 30.

Zhang, H.; Liu, Z.; Chen, B.; Zhao, Y.; Zhao, T.; Yang, T.; and Cui, B. 2024. Cafe: Towards compact, adaptive, and fast embedding for large-scale recommendation models. *SIGMOD*, 2(1): 1–28.

Zhang, Y.; Liu, Z.; Wang, R.; Yang, T.; Li, J.; Miao, R.; Liu, P.; Zhang, R.; and Jiang, J. 2021. CocoSketch: High-performance sketch-based measurement over arbitrary partial key query. In *SIGCOMM*, 207–222.