MIXTURE OF SKETCHES: LEARNING-AUGMENTED DISTRIBUTED SKETCH AGGREGATION WITH NETSKETCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Estimating frequencies of elements in a distributed data stream is a key task in machine learning, databases, and network measurement. It utilizes popular sketches such as count-min sketch (CMS) to collect data with worst-case error guarantees in the single-node setting, while a global frequency view is obtained by maintaining per-node sketches and aggregating them counter-wise. State-of-the-art sketches attach learned neural corrections to sketches to mitigate single-node errors, but per-node correction easily ignores cross-node statistics after aggregation. To address this problem, we propose NETSKETCH, an implementation of *mixture of sketches* (MoS) with a local correction module (LCM) at each node and a merge-aware correction module (MACM) at the aggregator. Each node runs the LCM and transfers its corrected sketch and compact summary while the aggregator counter-wise merges the corrected sketches and runs the MACM with cross-node statistics to produce global estimates. Extensive experiments on eight real-world workloads show that NETSKETCH achieves $1.6\times$-$17.2\times$ AAE reduction over six state-of-the-art baselines. Our analysis further characterizes why the two-level design helps, i.e., it improves the aggregation-error dependence from $\Omega(K\,\varepsilon_{\text{single}})$ for single-level correction to $O(\sqrt{K}\,\varepsilon_{\text{single}})$ for MoS, with a finite-sample refinement and a matching lower bound.

1 INTRODUCTION

Estimating frequencies of elements in a data stream has already been a key task in machine learning (Bifet et al., 2011; Tai et al., 2018), databases (Cormode and Muthukrishnan, 2005; Charikar et al., 2004), and network measurement (Yang et al., 2018; Tang et al., 2019; Ding et al., 2023). It leverages popular sketches such as count-min sketch (CMS) (Cormode and Muthukrishnan, 2005), count sketch (Charikar et al., 2004), and tower sketch (Yang et al., 2023) with worst-case error guarantees and uses sublinear space and $O(1)$ update time.

In modern large-scale infrastructures, the same stream is observed at K distributed nodes, making a global frequency view necessary. The standard solution maintains a per-node sketch over each substream of length N_k and obtains the global estimator by counter-wise aggregation, e.g., addition for CMS and maximum for CMS variants. After aggregation, the standard CMS guarantee relaxes by a factor of K at fixed total memory.

However, sketches result in errors when measuring data streams due to the shortage of resources per node. To mitigate such errors, Hsu et al. (2019) introduced the idea of attaching a learned neural correction to the sketch while recent learning-augmented sketches (Mitzenmacher, 2018; Aamand et al., 2023; Cao et al., 2023; Feng et al., 2025; Wang et al., 2026) build on this idea. The most recent representative, RATIOSKETCH (Wang et al., 2026), places the neural corrector at each per-node sketch and trains it to reduce single-node errors.

Nevertheless, after aggregation, a per-node corrector ignores cross-node statistics and cannot reduce the global error contributed by counter-wise addition. On the other hand, placing the corrector at the aggregator instead operates on a feature distribution in which the per-node residuals have already been mixed. As such, the per-node information that identifies the source of error is lost. We prove in Theorem 1 that any single-level corrector trained on the per-node feature distribution and transferred

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

zero-shot to the aggregated sketch incurs $\Omega(K\,\varepsilon_{\text{single}})$ aggregated errors, where $\varepsilon_{\text{single}}$ is the optimal single-node risk.

To fill this gap, we take the first step by presenting *mixture of sketches* (MoS), a learning-augmented two-level approach in which local per-node measurement information and global cross-node measurement information jointly drive a high-accuracy estimator. Our key insight is two-fold. First, a local expert at each node applies a per-node neural correction and produces an approximately zero-mean residual of variance $O(\varepsilon^2_{\text{single}})$. Second, a global expert at the aggregator combines the local experts' corrected sketches via counter-wise aggregation and applies a final correction conditioned on cross-node statistics. By independence of the per-node residuals, the aggregated residual has standard deviation $O(\sqrt{K}\,\varepsilon_{\text{single}})$, improving the $\Omega(K\,\varepsilon_{\text{single}})$ rate of any single-level approach.

We realize MoS within NETSKETCH, a compact module pair consisting of a local correction module (LCM) at each node and a merge-aware correction module (MACM) at the aggregator. We summarize the main contributions below.

- **Mixture of sketches.** We propose a learning-augmented MoS approach for distributed frequency estimation in which a local expert at each node corrects per-node hash-collision error and a global expert at the aggregator combines the corrected sketches with cross-node statistics.
- **Implementation and evaluation.** We implement MoS with NETSKETCH using $20{,}324$ trainable parameters. Extensive experiments on eight real-world workloads with up to 64 nodes show $1.6\times$-$17.2\times$ AAE reduction across six state-of-the-art baselines (Hsu et al., 2019; Aamand et al., 2023; Cao et al., 2023; Feng et al., 2025; Wang et al., 2026).
- **Supporting analysis.** We characterize why single-level correction degrades after aggregation and show that MoS improves the error dependence from $\Omega(K\,\varepsilon_{\text{single}})$ to $O(\sqrt{K}\,\varepsilon_{\text{single}})$, with finite-sample and matching lower-bound results.

2 RELATED WORK

Hand-crafted sketches. These sketches refer to the traditional sketches developed by experts to solve data streaming problems under resource-constrained cases (Roy et al., 2016; Yang et al., 2018; Li et al., 2020; Zhang et al., 2021). To name just a few, the classic CMS (Cormode and Muthukrishnan, 2005) uses d arrays of w counters with independent hash functions and queries on the minimum among d counters. Count-update sketch (CUS) (Estan and Varghese, 2003) and count sketch (CS) (Charikar et al., 2004) share similar structures with different operations. TowerSketch (Yang et al., 2023) uses progressively smaller counters in its data structures, achieving better accuracy-memory tradeoffs. However, they naturally suffer from the errors stem from hash collisions, which become severe under small memory budgets.

Learning-augmented sketches and neural sketches. Some sketches use heavy-hitter classifiers to allocate exact counters to predicted frequent items (Hsu et al., 2019; Mitzenmacher, 2018; Aamand et al., 2023). Also, some adopt neural components, e.g., neural-memory operations with meta-training in METASKETCH (Cao et al., 2023; 2024), fine-grained memory blocks with normalized multi-hash embeddings in LEGOSKETCH (Feng et al., 2025), label-free self-supervised correction in *UCL-Sketch* (Yuan et al., 2024), and lightweight neural micro-correction on top of hand-crafted sketches in RATIOSKETCH (Wang et al., 2026). However, all existing sketches are designed and evaluated only under single-node settings. None of them considers the errors introduced by aggregation.

Distributed sketch aggregation. In network-wide measurement, local sketch instances maintained at distributed nodes in different locations are aggregated to approximate global frequency statistics (Yang et al., 2018; Tang et al., 2019). Counter-wise addition for CMS preserves the linearity of sketch counters under stream concatenation, while counter-wise maximum for CMS variants preserves mergeability without preserving linearity. Both operations enable decentralized monitoring without centralizing traffic. However, the error characteristics of aggregated sketches differ from single-node ones. The root cause is that the collision errors from K nodes accumulate while the statistical assumptions behind single-node correction, e.g., the distribution of flow features, no longer hold. None of the prior work has addressed this problem despite its practical importance.

3 METHODOLOGY

3.1 PRELIMINARIES

We consider K measurement nodes, each processing a local data stream $S_k = (e_1^{(k)}, \ldots, e_{N_k}^{(k)})$ that contains N_k items with n_k distinct items, for $k = 1, \ldots, K$. All items come from the global domain $\mathcal{E} = \{E_1, \ldots, E_n\}$, where n is the number of globally distinct items. The global frequency of item E_i is $f_i = \sum_{k=1}^{K} f_i^{(k)}$, where $f_i^{(k)}$ denotes its local frequency at node k. The total stream length is $N = \sum_{k=1}^{K} N_k$. Moreover, each node k maintains a base sketch with d hash functions and a $d \times w$ counter matrix. Querying item E_i at node k produces d estimated values $v^{(k)} = [v_1^{(k)}, \ldots, v_d^{(k)}]$ and a final estimate $v_0^{(k)}$. Without loss of generality, we model each counter value as:

$$v_j^{(k)} = f_i^{(k)} + X_j^{(k)}, \tag{1}$$

where $X_j^{(k)} \geq 0$ represents the hash collision error at position j of node k. We assume all K nodes share the identical set of d hash functions and random seeds, which is standard for distributed CMS deployments and is necessary for counter-wise aggregation. To obtain global frequency estimates, local sketches are aggregated by counter-wise addition:

$$v_j^{\text{agg}} = \sum_{k=1}^{K} v_j^{(k)} = f_i + \sum_{k=1}^{K} X_j^{(k)}. \tag{2}$$

where, under the independent-node traffic model specified in Theorem 2, the aggregated collision error $\sum_{k=1}^{K} X_j^{(k)}$ is the sum of K independent non-negative random variables. Its magnitude and distribution differ from any single $X_j^{(k)}$, which is the root cause of errors when aggregating distributed sketches and is the main target of this paper.

3.2 MAIN RESULTS

We now state the two bounds that drive the design. We write $\varepsilon_{\text{single}}$ for the expected single-node estimation error of any reference learned-correction approach operating on one CMS instance trained on its own local distribution.

Lower bound for single-level correction. Any corrector $\mathcal{N}$ that is trained on the per-node feature distribution and then applied to the aggregated sketch suffers a K-fold error amplification:

Theorem 1. *Let $\mathcal{N}$ be any deterministic correction map fixed after training on the single-node feature distribution $\mathcal{D}_{\text{single}}$ with single-node risk $\varepsilon_{\text{single}} = \mathbb{E}_{\mathcal{D}_{\text{single}}}[|\hat{f} - f|]$. When the same $\mathcal{N}$ is applied to the K-node counter-wise-aggregated sketch, the worst-case expected error satisfies $\varepsilon_{\text{agg}}(\mathcal{N}) = \Omega(K \cdot \varepsilon_{\text{single}})$.*

Theorem 1 is proved in Appendix B.1. It gives a formal version of the observation that learning-augmented sketches (Hsu et al., 2019; Cao et al., 2023; Wang et al., 2026) do not transfer to aggregated sketches.

Upper bound via two-level correction. The lower bound is avoidable if and only if the corrector splits across the aggregation step. Suppose each node k first applies a corrector that produces a residual $\xi^{(k)} = \hat{f}_i^{(k)} - f_i^{(k)}$ that is approximately zero-mean and $O(\varepsilon_{\text{single}}^2)$-variance, and a second corrector then operates on the aggregated sketch and cross-node statistics. Independence across nodes implies $\text{Var}(\sum_k \xi^{(k)}) = O(K\varepsilon_{\text{single}}^2)$, hence the aggregated residual has standard deviation $O(\sqrt{K}\,\varepsilon_{\text{single}})$, given as Theorem 2 with a finite-sample version in Theorem 3. Theorem 4 gives a matching $\Omega(\sqrt{K}\,\varepsilon_{\text{single}})$ Le Cam two-point lower bound, which shows the rate is optimal among all approaches that operate locally before aggregation.

Two requirements. The two bounds together form a rate-optimal estimator. It must (i) extract per-node noise statistics into compact features that survive aggregation. Otherwise, the aggregator has access only to the merged sketch and falls back to the zero-shot single-level setting of Theorem 1. It must also (ii) use these per-node features at the aggregator together with the merged sketch to predict and subtract the K-scale bias. Otherwise, the per-node features are unused and the merged-sketch bias remains uncorrected. We realize both stages as small neural networks below.

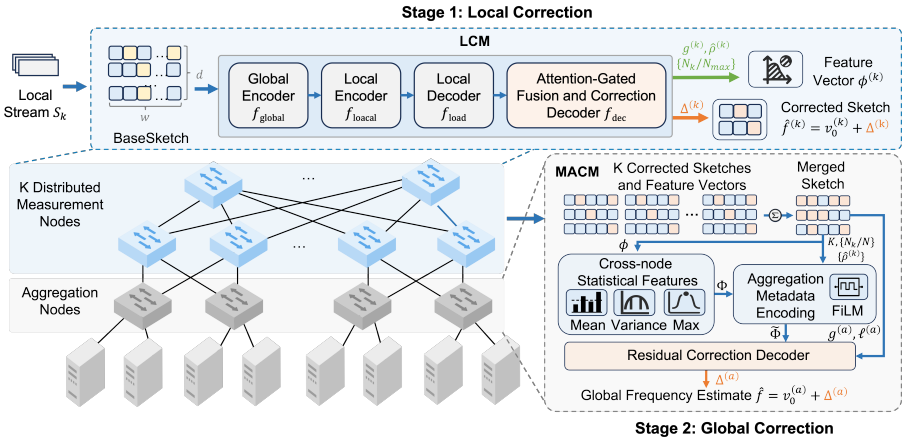

Figure 1: Overview of NETSKETCH. Each node applies the LCM to its local sketch and transfers the corrected sketch and compact feature vector. The aggregation point combines the corrected sketches by counter-wise addition and applies the MACM.

3.3 MIXTURE OF SKETCHES (MOS)

We realize MoS as the concrete estimator NETSKETCH, which pairs a local correction module (LCM) at each node with a merge-aware correction module (MACM) at the aggregator, detailed in Figure 1, §3.3.1, and §3.3.2 respectively. Both modules are small neural networks that accept arbitrary base sketches supporting counter-wise aggregation and a per-item base estimate v_0. We use CMS and consider Count Sketch in Appendix P. We denote the per-node base sketch as BaseSketch.

Roles of the LCM and MACM. The two modules together realize the rate-optimal estimator of Theorem 2, and each addresses one of the two requirements identified in §3.2. The LCM at node k corrects its local sketch and extracts a compact feature $\phi^{(k)}$ that captures the per-node load and noise statistics conditioned only on local features. Under the independent-node traffic model of Theorem 2, the resulting per-node residuals satisfy the cross-node independence condition used by the bound. The MACM at the aggregator pools $\{\phi^{(k)}\}$ via mean, variance, and max statistics across the K nodes and operates on the counter-wise merged corrected sketches. The pooled summary lets MACM estimate and remove the remaining aggregation-level bias. A learned skip gate keeps this correction from over-firing when the merged corrected sketch is already accurate. More details are given in Appendix D.

Deployment model. Neural inference is outside the packet-update fast path. During a measurement window, each node only hashes incoming keys and updates its BaseSketch counters. After the window closes, the node runs the LCM and transmits its corrected sketch and compact summary. The aggregator then merges the corrected sketches and runs the MACM to answer global queries. Model training is performed offline.

3.3.1 LOCAL CORRECTION MODULE

The LCM corrects the local sketch estimate via an additive residual correction $\hat{f}_i^{(k)} = v_0^{(k)} + \Delta^{(k)}$, where $\Delta^{(k)} \in \mathbb{R}$ is the predicted correction term. We denote the resulting corrected sketch by $\widetilde{M}^{(k)}$ and its queried counter values by $\widetilde{v}_j^{(k)}$. Its base query returns $\hat{f}_i^{(k)}$. Additive correction, rather than multiplicative, is essential for two reasons. First, it directly models the residual collision error $X_{(1)}^{(k)} = \min_j X_j^{(k)}$ that BaseSketch fails to remove. Thus, the network's target is bounded for any item frequency. Second, training with the L1 loss of §3.4 drives $\Delta^{(k)}$ to fit $X_{(1)}^{(k)}$ pointwise, producing an approximately bias-free residual that satisfies assumption (A1) of Theorem 2. A multiplicative form $\hat{f}_i = C \cdot v_0$ would couple the residual scale to v_0 and break the variance bound for infrequent items where v_0 is overwhelmed by noise.

The LCM consists of four components, i.e., $f_{\text{global}}, f_{\text{local}}, f_{\text{load}}, f_{\text{dec}}$, each comprising fully connected layers with ReLU activations. The *global encoder* f_{global} extracts a structural feature $\mathbf{g}^{(k)}$

and a hidden representation $\mathbf{h}^{(k)}$ from the first 64 counters of each row of the BaseSketch matrix. The *load decoder* $f_{\text{load}}(\mathbf{h}^{(k)}) = \hat{\rho}^{(k)}$ predicts the load ratio so the estimated distinct count is $n_k^* = \hat{\rho}^{(k)} \cdot w$. The *local encoder* f_{local} takes a 19-dimensional feature vector $\mathbf{x}_i^{(k)}$ that includes ratio features $r_j^{(k)} = (v_{(j)}^{(k)} - v_0^{(k)})/(v_0^{(k)} + \epsilon)$ and collision-intensity features $\gamma_j^{(k)} = (v_{(j)}^{(k)} - v_0^{(k)})/(N_k/n_k^*)$ as defined in Eq. (3), and statistical summaries, i.e., mean, std, and median. The *correction decoder* f_{dec} fuses the local and global features via attention gating and outputs $\Delta^{(k)}$:

$$r_j^{(k)} = \frac{v_{(j)}^{(k)} - v_0^{(k)}}{v_0^{(k)} + \epsilon}, \quad \gamma_j^{(k)} = \frac{v_{(j)}^{(k)} - v_0^{(k)}}{N_k/n_k^*}, \quad j = 1, \ldots, d, \tag{3}$$

$$\begin{aligned} \mathbf{a}^{(k)} &= \sigma\big(W_a[\boldsymbol{\ell}^{(k)};\, \mathbf{g}^{(k)};\, \mathbf{s}^{(k)}] + b_a\big), \\ \mathbf{z}^{(k)} &= \mathbf{a}^{(k)} \odot \boldsymbol{\ell}^{(k)} + (1 - \mathbf{a}^{(k)}) \odot \mathbf{g}^{(k)}, \qquad \Delta^{(k)} = f_{\text{dec}}\big([\mathbf{z}^{(k)};\, \mathbf{s}^{(k)}]\big), \end{aligned} \tag{4}$$

where $\boldsymbol{\ell}^{(k)} = f_{\text{local}}(\mathbf{x}_i^{(k)})$, $\mathbf{s}^{(k)} \in \mathbb{R}^3$ is a structural parameter vector, and σ is the sigmoid function. Each node transmits $\widetilde{M}^{(k)}$ together with a compact feature vector $\phi^{(k)} \in \mathbb{R}^{d_\phi}$, where $d_\phi \ll d \cdot w$, to the MACM via Eq. (5). The full details are in Appendix C.

$$\phi^{(k)} = W_\phi\big[\mathbf{g}^{(k)};\, \hat{\rho}^{(k)};\, N_k/N_{\max}\big] + b_\phi \in \mathbb{R}^{d_\phi}. \tag{5}$$

3.3.2 Merge-Aware Correction Module

The aggregator receives the corrected per-node sketches $\{\widetilde{M}^{(k)}\}$ and feature vectors, aggregates via counter-wise addition $v_j^{(\text{a})} = \sum_{k=1}^{K} \widetilde{v}_j^{(k)}$, and computes the post-aggregation base estimate $v_0^{(\text{a})} = \min_j v_j^{(\text{a})}$. The MACM produces the final estimate $\hat{f}_i = v_0^{(\text{a})} + g \cdot \Delta^{(\text{a})}$, with additive correction $\Delta^{(\text{a})}$ and a learned skip gate $g \in [0, 1]$. It has four components, i.e., (1) a permutation-invariant cross-node statistical feature pool Φ defined in Eq. (6) that combines $\{\phi^{(k)}\}$ via mean, variance, and max, (2) a FiLM (Perez et al., 2018) layer defined in Eq. (7) that conditions Φ on K, sorted $\{N_k\}$, and sorted $\{\hat{\rho}^{(k)}\}$, (3) a residual decoder that combines the merged-corrected-sketch local and global features with the conditioned $\tilde{\Phi}$, and (4) a skip gate g that adapts the correction magnitude to the remaining residual, driving $g \to 0$ when local correction and aggregation have already near-cancelled the noise.

$$\Phi = \big[\tfrac{1}{K}\textstyle\sum_k \phi^{(k)},\ \ \tfrac{1}{K}\textstyle\sum_k (\phi^{(k)} - \bar{\phi})^2,\ \ \max_k \phi^{(k)}\big] \in \mathbb{R}^{3d_\phi}, \tag{6}$$

$$\tilde{\Phi} = \boldsymbol{\gamma} \odot f_{\text{cross}}(\Phi) + \boldsymbol{\beta}, \quad (\boldsymbol{\beta}, \boldsymbol{\gamma}) = \text{split}\big(W_m\, f_{\text{meta}}(K, \text{sort}(\{N_k/N\}), \text{sort}(\{\hat{\rho}^{(k)}\})) + b_m\big), \tag{7}$$

$$\Delta^{(\text{a})} = f_{\text{dec}}^{(\text{a})}\big([\text{AttnGate}(\boldsymbol{\ell}^{(\text{a})}, \mathbf{g}^{(\text{a})});\ \tilde{\Phi}]\big), \quad g = \sigma\big(f_{\text{skip}}\big([\text{AttnGate}(\boldsymbol{\ell}^{(\text{a})}, \mathbf{g}^{(\text{a})});\ \tilde{\Phi}]\big)\big). \tag{8}$$

Here $\boldsymbol{\ell}^{(\text{a})}$ and $\mathbf{g}^{(\text{a})}$ are the local- and global-encoder outputs computed on the merged corrected sketch, produced by an aggregator-side encoder pair that shares architecture with f_{local} and f_{global} but uses independently trained parameters.

The MACM estimates the remaining bias of the merged corrected sketch conditioned on the pooled local statistics. The local additive head and the global additive head therefore have different targets. The former corrects per-node collision error. The latter predicts only the residual aggregation-level correction from the merged corrected counters and the information retained in $\{\phi^{(k)}\}$.

Differences between local and global corrections. The LCM and MACM target different errors. The LCM removes collision error using each node's local sketch before transmission, whereas the MACM is jointly trained on the merged corrected sketches and the summaries produced by those same LCM outputs. It therefore learns the remaining post-merge error directly and does not assume that this residual follows the original CMS collision-error distribution. The skip gate further suppresses the MACM output when the remaining error is already small.

3.4 Loss Function

Since both the LCM and MACM use additive corrections, the loss function directly penalizes the absolute correction error. For the LCM at node k, the loss is:

$$\mathcal{L}_{\text{LCM}} = \left(\frac{1}{n_k} \sum_{i=1}^{n_k} w_i^{(k)} \cdot \left| \Delta_i^{(k)} - \left(f_i^{(k)} - v_{0,i}^{(k)} \right) \right| \right) + \lambda_1 \left| \hat{\rho}^{(k)} - \frac{n_k}{w} \right|, \tag{9}$$

where $w_i^{(k)} = \log_2(f_i^{(k)}+1)$ gives frequent items a larger weight, and the load-prediction regularizer is placed outside the per-item summation so it is not implicitly scaled by n_k. For the MACM, the loss is:

$$\mathcal{L}_{\text{MACM}} = \frac{1}{n} \sum_{i=1}^{n} w_i \cdot \left| \Delta_i^{(\text{a})} - \left(f_i - v_{0,i}^{(\text{a})} \right) \right|, \tag{10}$$

with $w_i = \log_2(f_i + 1)$. To make the skip gate g adapt to the residual error magnitude, we add an L1 regularizer that penalizes the gated correction in proportion to how small the aggregated-sketch residual already is:

$$\mathcal{L}_{\text{skip}} = \frac{1}{n} \sum_{i=1}^{n} \frac{|g_i \cdot \Delta_i^{(\text{a})}|}{1 + |f_i - v_{0,i}^{(\text{a})}|}. \tag{11}$$

The total loss is balanced via the self-adjusting weight function $\mathcal{LW}$ (Kendall et al., 2018):

$$\mathcal{L} = \mathcal{LW}(\mathcal{L}_{\text{LCM}}, \mathcal{L}_{\text{MACM}}) + \lambda_{\text{skip}} \mathcal{L}_{\text{skip}}. \tag{12}$$

Adaptation for frequent items. If BaseSketch uses a heap to identify the top-k items, we use a heap flag to distinguish head items in the heap from other tail items. For head items, we apply the loss weight $\log_2(f_i + 1)$ to improve the accuracy for top-k items. Otherwise, the weight equals 1.

Decoupled skip gate. The skip gate g is supervised only via $\mathcal{L}_{\text{skip}}$ in Eq. (11), while $\mathcal{L}_{\text{MACM}}$ in Eq. (10) trains $\Delta^{(\text{a})}$ to fit the full residual $f_i - v_{0,i}^{(\text{a})}$ unconditionally. This decoupling lets the network learn the maximal correction magnitude no matter the value of g, with g deciding only whether to apply it at query time, i.e., $g \to 1$ when the aggregated residual is large and $g \to 0$ when the aggregated CMS is already near-zero error. To prevent the trivial $g \equiv 0$ collapse, we initialize the skip-gate head so that g is close to 1 at training start and use a small λ_{skip} relative to the Kendall-balanced loss scales. Empirically, g approaches 1 on items whose aggregated residual exceeds the per-node noise scale and approaches 0 when the merged CMS is accurate to within $\varepsilon_{\text{single}}$.

3.5 Theoretical Analysis

This subsection formalizes the theorems in §3.2. Full proofs are in Appendix B.

Upper bound. The formal statement makes the assumptions of Theorem 2 explicit.

Theorem 2. *Suppose (A1) the per-node LCM residuals $\xi^{(k)} = \hat{f}_i^{(k)} - f_i^{(k)}$ are zero-mean, mutually independent across k, with variance $O(\varepsilon_{\text{single}}^2)$, and (A2) the per-node memory stays in the CMS-tractable range $w \geq c\, N_k/\varepsilon_{\text{single}}$ for an absolute constant $c > 0$. Then the two-level estimator satisfies*

$$\mathbb{E}[|\hat{f}_i^{\text{NetSketch}} - f_i|] = O\left(\sqrt{K} \cdot \varepsilon_{\text{single}} \right), \tag{13}$$

which is strictly better than the $\Omega(K\, \varepsilon_{\text{single}})$ rate of Theorem 1 for all $K \geq 2$.

The improvement is a variance-summation argument. The cross-node independence condition in (A1) gives $\text{Var}(\sum_k \xi^{(k)}) = O(K\varepsilon_{\text{single}}^2)$, hence standard deviation $O(\sqrt{K}\, \varepsilon_{\text{single}})$. The LCM's node-local inputs preserve this conditional structure without introducing cross-node coupling. Assumption (A2) keeps us out of the memory-starved range where each counter contains $\Theta(N_k/w)$ uncancellable collisions and (A1) breaks down. The full $K \in \{4, 8, 16, 32, 64\}$ scan in §4.3 stays inside this range and the empirical rate matches $\sqrt{K}$ as expected.

Finite-sample bound and tightness. The asymptotic bound above admits a non-asymptotic refinement explicit in N, K, w, and a confidence parameter δ, given as Theorem 3 in Appendix B.3. A

matching $\Omega(\sqrt{K}\,\varepsilon_{\text{single}})$ minimax Le Cam lower bound, given as Theorem 4 in Appendix B, certifies that the $\sqrt{K}$ rate is minimax-tight over the natural class of *two-level estimators* of the form $\widehat{f} = g(\phi_1(s_1), \ldots, \phi_K(s_K))$, where each ϕ_i is an arbitrary per-node feature map of the local sketch state s_i and g is an arbitrary post-aggregation function. NETSKETCH's LCM+MACM pipeline is exactly a member of this class. The LCM realizes the ϕ_i and the MACM realizes g. Thus, the upper bound of Theorem 2 is minimax-tight on the class containing our own estimator. The class is defined over per-node frequency configurations with sub-Gaussian collision noise of variance proxy σ^2_{loc} (i.e., the parameter of the sub-Gaussian envelope, which by definition upper-bounds the variance and may exceed the sample variance under heavy-tailed but bounded counter distributions).

Additional results. Two further properties of the construction are proved in Appendix B, i.e., scale invariance of the LCM ratio features and a Rademacher-style generalization bound with $O(1/\sqrt{m})$ rate and only logarithmic dependence on the sketch size dw and the node count K. A direct comparison with the bounds of Hsu et al. (2019) and the improved-prediction variant (Aamand et al., 2023) is in Appendix I. The two-level $O(\sqrt{K}\,\varepsilon_{\text{single}})$ rate is, to our knowledge, the first sub-linear-in-K guarantee for distributed learning-augmented sketches.

4 EVALUATION

Our evaluation reports six main findings. (Q1) NETSKETCH achieves the lowest AAE and ARE on all eight datasets with $4.58\times$–$6.63\times$ reduction over the five learning-augmented baselines and $12.2\times$–$17.2\times$ reduction over Merged CMS. (Q2) The advantage extends to top-20% frequent items with a $5\times$–$30\times$ AAE and ARE reduction. (Q3) On heavy-hitter F1, NETSKETCH reaches around 0.79–0.88 on MAWI and 0.85 on WAN, a 50%–77% improvement over Merged CMS, METASKETCH, and LEGOSKETCH. (Q4) NETSKETCH is $1.8\times$ faster than METASKETCH and $22\times$ faster than LEGOSKETCH. (Q5) The empirical error scales as $O(\sqrt{K})$ across $K \in \{4, 8, 16, 32, 64\}$ and validates Theorem 2 with up to $10.64\times$ AAE reduction. (Q6) The gap is robust to uniform, hash, and skewed traffic splitting. The factorial module ablation gives Full 7.09, w/o LCM 12.5, w/o MACM 18.7, and w/o Both 75.4 AAE on MAWI 2025 at $K = 16$ and 5 MB. Removing both modules collapses to the Merged-CMS baseline, directly confirming Theorem 1's $\Omega(K)$ amplification.

4.1 SETUP

Datasets. We evaluate NETSKETCH on eight high-skew real-world datasets, i.e., four backbone-trace time-window splits WAN-A/B/C/D from the WIDE project, and four daily MAWI (Cho et al., 2000) samplepoint-F snapshots, MAWI 2023, MAWI 2024, MAWI 2025, and MAWI 2026. Per-dataset rank-frequency curves and Zipf-fit slopes are given in Appendix F, Figure 7.

Training and evaluation protocol. NETSKETCH is trained in two stages, i.e., a label-free stage on synthetic Zipf streams and a fine-tune stage on each real trace. We then report accuracy under the eight datasets in §4.1. Each dataset is distributed among $K \in \{4, 8, 16, 32, 64\}$ nodes under three traffic splitting strategies, i.e., uniform, hash-based, and Zipf-weighted. We focus on the sub-megabyte to a few-megabyte per-cluster memory range, i.e., $M \in [0.2, 5]$ MB.

Baselines. We consider six state-of-the-art baselines, i.e., (1) the original Merged CMS, (2) Merged CMS+LEARNEDSKETCH (Hsu et al., 2019), a learned-CMS approach, (3) Merged CMS+Aamand (Aamand et al., 2023), the improved variant with heavy-hitter predictions, (4) Merged CMS+RATIOSKETCH as a recent single-level neural correction, and (5-6) Merged CMS+METASKETCH and Merged CMS+LEGOSKETCH as recent neural sketches. LEARNEDSKETCH and Aamand were originally proposed for single-node settings. We extend them to the merged setting by running their per-node learned predictors independently and combining via counter-wise merge, which is the natural multi-node extension. All five learning-augmented baselines are retrained on a distribution that includes merged-sketch examples covering the same $K \in \{4, 8, 16, 32, 64\}$ range used for NETSKETCH. We do not reuse the original single-node-only checkpoints. This gives each baseline a strict fair-comparison advantage relative to its own published evaluation. Theorem 1 predicts that no single-level corrector escapes the $\Omega(K)$ amplification no matter the training distribution, which is exactly what we observe in Table 6. The LCM uses 4 linear-ReLU layers of width 16 and the MACM adds a cross-node FiLM encoder of similar width. The released inference configuration has $6{,}130$ LCM and $14{,}194$ MACM parameters ($20{,}324$ total).

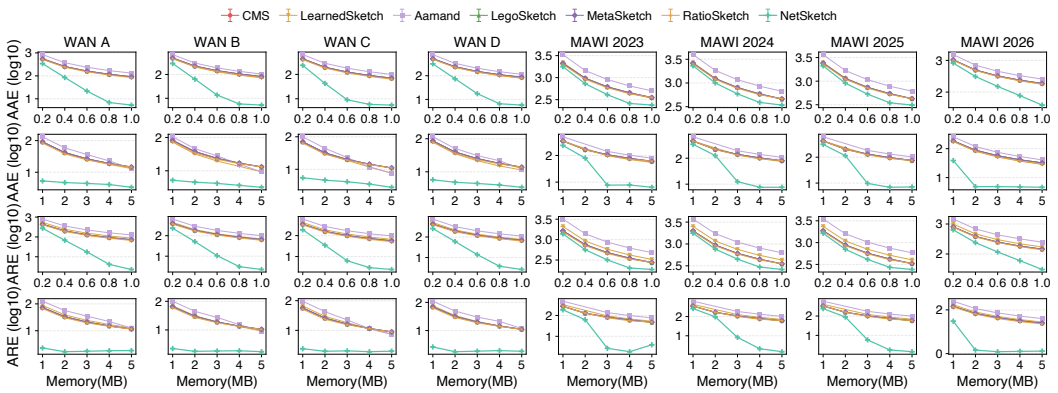

Figure 2: AAE (top) and ARE (bottom) vs. memory on the eight real-world datasets at $K = 16$ under uniform splitting in the $[0.2, 5]$ MB range.

The two loss-balancer scalars are used only during training. The int8-equivalent value payload is 19.8 KiB before quantization metadata. LEGOSKETCH uses 20 KB and METASKETCH scales with memory.

Metrics. We report four metrics, i.e., (1) AAE and ARE, (2) top-20% AAE and ARE for frequent-item accuracy, (3) threshold-based heavy-hitter F1 at $T = 50$, and (4) throughput in Mops. For queried items with positive true frequency, $\text{AAE} = \frac{1}{|I|}\sum_{i\in I}|\hat{f}_i - f_i|$ and $\text{ARE} = \frac{1}{|I|}\sum_{i\in I}|\hat{f}_i - f_i|/f_i$. The top-20% metrics apply the same definitions to the most frequent items. All methods share the same CMS base sketch with $d = 3$. Unless otherwise noted, accuracy is at $K = 16$ nodes under uniform splitting and memory budgets in $[0.2, 5]$ MB. Scalability and traffic-splitting use MAWI 2025 at 5 MB. Throughput is measured on a single CPU core with a 50K-query batch. Each accuracy point is the mean over 100 independent evaluation runs at a fixed model seed, with the per-run randomness drawn from the CMS hash family. Shaded bands and error bars in Figures 2 and 3 report ± 1 standard deviation across the 100 runs. Full details are in Appendix F.

4.2 ACCURACY

NETSKETCH has the lowest AAE and ARE (Q1). In Figure 2, NETSKETCH achieves the lowest AAE and ARE on all eight real-world datasets at every memory budget in $[0.2, 5]$ MB. NETSKETCH has lower AAE and ARE than the uncorrected Merged CMS and the single-level Merged CMS+RATIOSKETCH. It also has lower error than existing neural sketches, i.e., LEGOSKETCH and METASKETCH. The skip gate at the MACM output adapts the correction magnitude to the residual error of the aggregated sketch. Thus, the network applies a strong correction when noise is large and suppresses it when the CMS already has near-zero error.

For example, on MAWI 2026 at 1 MB, the AAE of CMS+NETSKETCH is $4.83\times$, $6.63\times$, $4.58\times$, $4.59\times$, $4.83\times$, and $4.81\times$ lower than Merged CMS, Merged + Aamand, Merged + LEARNEDSKETCH, METASKETCH, LEGOSKETCH, and Merged + RATIOSKETCH, respectively. On the WAN-A/B/C/D workloads at 1 MB, the AAE reduction over Merged CMS is between $12.2\times$ and $17.2\times$. The advantage grows with K. At 1 MB on MAWI 2026, NETSKETCH's AAE reduction over the best CMS-based baseline goes from $1.6\times$ at $K=8$ to $3.5\times$ at $K=32$, consistent with Theorem 2.

Moreover, the two-level architecture itself, without any neural correction, already delivers a $6.4\times$ AAE reduction over Merged CMS on MAWI 2025 at $K = 16$ and 5 MB, given as TwoLevel-MLE in Appendix J. The additional $1.66\times$ from the LCM and MACM neural networks brings the total to $10.6\times$ on this setting. Thus, the architectural decomposition is the dominant driver of accuracy, and the neural correction is a meaningful refinement on top.

NETSKETCH improves frequent-item accuracy (Q2). As shown in Figure 3, for the top-20% frequent items, NETSKETCH achieves the lowest top-20% AAE and top-20% ARE on every dataset and every memory budget in $[0.2, 5]$ MB. The differentiated loss weighting in §3.4 concentrates correction capacity on frequent items, and the gap over Merged CMS reaches $5\times$-$30\times$ on the heavy-tailed MAWI traces.

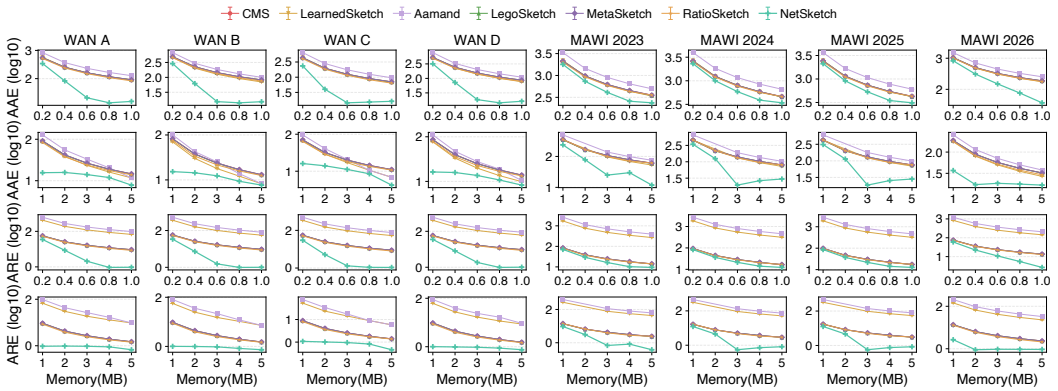

Figure 3: Top-20% AAE and ARE on the eight datasets at $K = 16$ under uniform splitting.

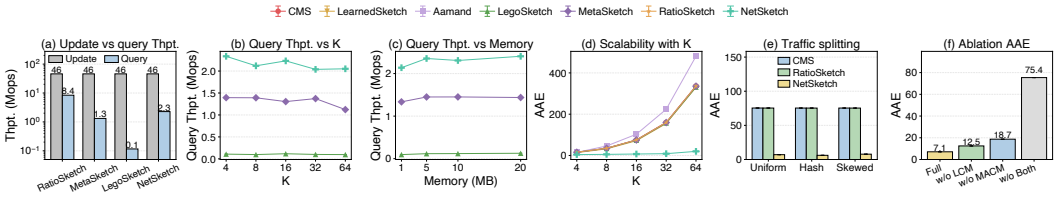

Figure 4: Performance and ablation on MAWI 2025 at $K = 16$ and 5 MB unless otherwise noted.

NETSKETCH improves heavy-hitter F1 (Q3). At the standard threshold $T = 50$ (Yang et al., 2018; Tang et al., 2019), NETSKETCH reaches F1 0.79–0.88 at 1 MB on MAWI, versus 0.10–0.35 for Merged CMS, METASKETCH, and LEGOSKETCH. On WAN it reaches 0.85, improving on the best CMS baseline by 27%. MACM reduces CMS's positive bias and the resulting false positives; per-dataset curves are in Appendix F.1.

4.3 MICROBENCHMARKS

NETSKETCH preserves update throughput (Q4). At $K = 16$ and 5 MB on one CPU core, CMS+NETSKETCH preserves the CMS update throughput of 46 Mops and reaches 2.28 Mops for queries, 1.7× faster than METASKETCH and 19× faster than LEGOSKETCH (Figure 4(a)). Across the K and memory scans in Figures 4(b,c), query throughput remains 2.04–2.41 Mops.

AAE grows sublinearly with K (Q5). With one fixed checkpoint, NETSKETCH has AAE 4.32, 5.36, 7.09, 8.72, and 19.91 for $K \in \{4, 8, 16, 32, 64\}$ and outperforms every baseline throughout (Figure 4(d)). AAE/$\sqrt{K}$ remains between 1.54 and 2.49, consistent with Theorem 2.

Splitting and ablation (Q6). Figure 4(e) shows AAE between 6.1 and 7.8 under uniform, hash, and skewed traffic splitting. In Figure 4(f), the AAEs of *Full*, *w/o LCM*, *w/o MACM*, and *w/o Both* are 7.09, 12.5, 18.7, and 75.4, respectively. Thus both stages contribute, and the full two-level design avoids the single-level amplification characterized by Theorem 1.

5 CONCLUSION

We presented NetSketch, a two-level learning-augmented framework for distributed sketch-based frequency estimation. The framework comprises a local correction module at each node and a merge-aware correction module at the aggregator. Theoretically, we proved that any single-level corrector suffers from an $\Omega(K\varepsilon_{\text{single}})$ aggregated error, whereas the two-level design achieves a tight $O(\sqrt{K}\varepsilon_{\text{single}})$ rate with a matching minimax lower bound. This establishes the first sub-linear-in-K guarantee for distributed learning-augmented sketches. Empirically, NetSketch delivers 1.6× to 17.2× average absolute error reduction over six state-of-the-art baselines across eight real-world workloads with up to 64 nodes. The framework maintains robust performance under unseen hash seeds, counter permutations, correlated traffic splits, temporal shifts, and low-skew distributions.

REFERENCES

A. Aamand, J. Chen, H. L. Nguyen, S. Silwal, and A. Vakilian. Improved frequency estimation algorithms with and without predictions. In *NeurIPS*, 2023. https://proceedings.neurips.cc/paper_files/paper/2023/hash/2e49934cac6cb8604b0c67cfa0828718-Abstract-Conference.html

P. L. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463-482, 2002. https://www.jmlr.org/papers/v3/bartlett02a.html

S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1-2):151-175, 2010. https://doi.org/10.1007/s10994-009-5152-4

A. Bifet, G. Holmes, B. Pfahringer, and R. Gavaldà. Detecting sentiment change in Twitter streaming data. In *2nd Workshop on Applications of Pattern Analysis (WAPA), JMLR Workshop and Conference Proceedings vol. 17*, pp. 5-11, 2011. https://proceedings.mlr.press/v17/bifet11a.html

The CAIDA UCSD anonymized internet traces. 2018. https://www.caida.org/catalog/datasets/passive_dataset/

Y. Cao, Y. Feng, and X. Xie. Meta-Sketch: A neural data structure for estimating item frequencies of data streams. In *AAAI*, volume 37, pp. 6916-6924, 2023. https://doi.org/10.1609/aaai.v37i6.25846

Y. Cao, Y. Feng, H. Wang, X. Xie, and S. K. Zhou. Learning to sketch: A neural approach to item frequency estimation in streaming data. *IEEE TPAMI*, 46(11):7136-7153, 2024. https://doi.org/10.1109/TPAMI.2024.3388589

M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. *Theoretical Computer Science*, 312(1):3-15, 2004. https://doi.org/10.1016/S0304-3975(03)00400-6

K. Cho, K. Mitsuya, and A. Kato. Traffic data repository at the WIDE project. In *USENIX Annual Technical Conference, FREENIX Track*, 2000. https://www.usenix.org/legacy/publications/library/proceedings/usenix2000/freenix/cho.html

G. Cormode and S. Muthukrishnan. An improved data stream summary: The Count-Min Sketch and its applications. *Journal of Algorithms*, 55(1):58-75, 2005. https://doi.org/10.1016/j.jalgor.2003.12.001

R. Ding, S. Yang, X. Chen, and Q. Huang. BitSense: Universal and nearly zero-error optimization for sketch counters with compressive sensing. In *ACM SIGCOMM*, pp. 220-238, 2023. https://doi.org/10.1145/3603269.3604865

C. Estan and G. Varghese. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM TOCS*, 21(3):270-313, 2003. https://doi.org/10.1145/859716.859719

Y. Feng, Y. Cao, H. Wang, X. Xie, and S. K. Zhou. Lego Sketch: A scalable memory-augmented neural network for sketching data streams. In *ICML*, PMLR vol. 267, pp. 16589-16603, 2025. https://proceedings.mlr.press/v267/feng25a.html

A. Graves, G. Wayne, and I. Danihelka. Neural Turing machines. *arXiv preprint arXiv:1410.5401*, 2014. https://arxiv.org/abs/1410.5401

A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471-476, 2016. https://doi.org/10.1038/nature20101

C.-Y. Hsu, P. Indyk, D. Katabi, and A. Vakilian. Learning-based frequency estimation algorithms. In *ICLR*, 2019. https://openreview.net/forum?id=r1lohoCqY7

A. Kendall, Y. Gal, and R. Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, pp. 7482-7491, 2018. https://doi.org/10.1109/CVPR.2018.00781

M. Ledoux and M. Talagrand. *Probability in Banach Spaces: Isoperimetry and Processes*. Springer, 1991. https://doi.org/10.1007/978-3-642-20212-4

J. Li, Z. Li, Y. Xu, S. Jiang, T. Yang, B. Cui, Y. Dai, and G. Zhang. WavingSketch: An unbiased and generic sketch for finding top-k items in data streams. In *ACM SIGKDD*, pp. 1574-1584, 2020. https://doi.org/10.1145/3394486.3403208

Y. Mansour, M. Mohri, and A. Rostamizadeh. Domain adaptation: Learning bounds and algorithms. In *COLT*, 2009. https://arxiv.org/abs/0902.3430

M. Mitzenmacher. A model for learned Bloom filters and optimizing by sandwiching. In *NeurIPS*, pp. 462-471, 2018. https://proceedings.neurips.cc/paper/2018/hash/0f49c89d1e7298bb9930789c8ed59d48-Abstract.html

M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. MIT Press, 2nd edition, 2018. https://mitpress.mit.edu/9780262039406/

E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville. FiLM: Visual reasoning with a general conditioning layer. In *AAAI*, pp. 3942-3951, 2018. https://doi.org/10.1609/aaai.v32i1.11671

D. M. Powers. Applications and explanations of Zipf's law. In *NeMLaP3/CoNLL98*, pp. 151-160, 1998. https://aclanthology.org/W98-1218/

P. Roy, A. Khan, and G. Alonso. Augmented Sketch: Faster and more accurate stream processing. In *ACM SIGMOD*, pp. 1449-1463, 2016. https://doi.org/10.1145/2882903.2882948

K. S. Tai, V. Sharan, P. Bailis, and G. Valiant. Sketching linear classifiers over data streams. In *ACM SIGMOD*, pp. 757-772, 2018. https://doi.org/10.1145/3183713.3196930

L. Tang, Q. Huang, and P. P. Lee. MV-Sketch: A fast and compact invertible sketch for heavy flow detection in network data streams. In *IEEE INFOCOM*, pp. 2026-2034, 2019. https://doi.org/10.1109/INFOCOM.2019.8737499

A. B. Tsybakov. *Introduction to Nonparametric Estimation*. Springer, 2009. https://doi.org/10.1007/b13794

M. Wang, Z. Fan, D. Wang, G. Xie, Q. Li, Z. Luan, Y. Jiang, T. Yang, and M. Xu. RatioSketch: Towards more accurate frequency estimation in data streams via a lightweight neural network. In *AAAI*, pp. 26380-26389, 2026. https://doi.org/10.1609/aaai.v40i31.39844

J. Weston, S. Chopra, and A. Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014. https://arxiv.org/abs/1410.3916

T. Yang, J. Jiang, P. Liu, Q. Huang, J. Gong, Y. Zhou, R. Miao, X. Li, and S. Uhlig. Elastic sketch: Adaptive and fast network-wide measurements. In *ACM SIGCOMM*, pp. 561-575, 2018. https://doi.org/10.1145/3230543.3230544

K. Yang, S. Long, Q. Shi, Y. Li, Z. Liu, Y. Wu, T. Yang, and Z. Jia. SketchINT: Empowering INT with TowerSketch for per-flow per-switch measurement. *IEEE TPDS*, 34(11):2876-2894, 2023. https://doi.org/10.1109/TPDS.2023.3303924

X. Yuan, Y. Qiao, M. Li, Z. Wei, C. Feng, Z. Wang, and W. Chen. Learning-based sketches for frequency estimation in data streams without ground truth. *arXiv preprint arXiv:2412.03611*, 2024. https://arxiv.org/abs/2412.03611

Y. Zhang, Z. Liu, R. Wang, T. Yang, J. Li, R. Miao, P. Liu, R. Zhang, and J. Jiang. CocoSketch: High-performance sketch-based measurement over arbitrary partial key query. In *ACM SIGCOMM*, pp. 207-222, 2021. https://doi.org/10.1145/3452296.3472892

A NOTATION

Table 1 summarizes the symbols used throughout the paper, grouped by role.

Table 1: Notation summary.

Symbol	Meaning
K	Number of distributed nodes (cluster size)
N_k	Stream length at node k, with $N = \sum_k N_k$
n_k	Number of distinct items at node k
d, w	Per-node CMS depth (rows) and width (columns per row)
M	Total memory budget across the cluster, $M = K \cdot d \cdot w$ in bytes
$f_i, f_i^{(k)}$	Global and per-node true frequency of item i
$\hat{f}_i, \hat{f}_i^{(k)}$	Global and per-node estimated frequency of item i
$v_j^{(k)}$	Counter value at row j of the per-node CMS at node k
$v_0^{(k)}$	Per-node base estimate, $v_0^{(k)} = \min_j v_j^{(k)}$ for CMS
$v_j^{(\mathrm{a})}, v_0^{(\mathrm{a})}$	Aggregated counter and aggregated base estimate
$X_j^{(k)}$	Per-counter hash-collision error, $v_j^{(k)} = f_i^{(k)} + X_j^{(k)}$
$\xi^{(k)}$	Per-node LCM residual, $\xi^{(k)} = \hat{f}_i^{(k)} - f_i^{(k)}$
$\Delta^{(k)}, \Delta^{(\mathrm{a})}$	LCM additive correction at node k, MACM additive correction
g	MACM learned skip gate, $g \in [0, 1]$
$\phi^{(k)}$	Compact per-node feature vector transmitted to MACM, $\phi^{(k)} \in \mathbb{R}^{d_\phi}$
d_ϕ	Compact feature dimension (default 16)
$\Phi, \tilde{\Phi}$	Cross-node pooled features and FiLM-adjusted cross-node features
$\varepsilon_{\mathrm{single}}, \sigma_{\mathrm{loc}}$	Optimal single-node residual error / standard deviation
$\varepsilon_{\mathrm{agg}}$	Aggregated-sketch error of an estimator
$\mathcal{F}_K$	Class of K-node frequency configurations with sub-Gaussian noise
$\mathcal{E}_{\mathrm{TL}}$	Class of two-level estimators $\widehat{f} = g(\phi_1(s_1), \ldots, \phi_K(s_K))$, subsuming LCM+MACM

B PROOFS

B.1 PROOF OF THEOREM 1

Proof. We model a single-level neural correction as a fixed map $\mathcal{N} : \mathbb{R}^{d+d_g+d_s} \to \mathbb{R}$ that, given input features $\mathbf{u} = (\mathbf{r}, \mathbf{g}, \mathbf{s})$, outputs a residual estimate $\Delta = \mathcal{N}(\mathbf{u})$, yielding $\hat{f}_i = v_0 + \Delta$. The argument also covers multiplicative corrections $\hat{f}_i = C \cdot v_0$ verbatim by absorbing the v_0 factor into Δ. The map $\mathcal{N}$ is trained on the single-node feature distribution $\mathcal{D}_{\mathrm{single}}$, achieving expected error $\varepsilon_{\mathrm{single}} = \mathbb{E}_{\mathcal{D}_{\mathrm{single}}}[|\hat{f} - f|]$.

Step 1: input distribution shift. On single-node sketches, the ratio feature is

$$r_j = \frac{X_j - X_{(1)}}{f + X_{(1)}}, \qquad X_{(1)} = \min_j X_j,$$

where $\{X_j\}$ are the per-counter collision errors with $\mathbb{E}[X_j] = \bar{\mu}$ and $\mathrm{Var}(X_j) = \bar{\sigma}_X^2$. After aggregating K independent node sketches, the corresponding aggregated feature is $r_j^{(\mathrm{a})} = (Y_j - Y_{(1)})/(f + Y_{(1)})$ with $Y_j = \sum_{k=1}^{K} X_j^{(k)}$. By independence,

$$\mathbb{E}[Y_j] = K\bar{\mu}, \qquad \mathrm{Var}(Y_j) = K\bar{\sigma}_X^2. \tag{14}$$

The aggregated feature $r_j^{(\mathrm{a})}$ is centered at $\Theta(K\bar{\mu}/(f + K\bar{\mu}))$ with $\Theta(1/\sqrt{K})$ standard deviation, i.e., numerator standard deviation $\Theta(\sqrt{K})$ over denominator $\Theta(K)$, whereas $r_j^{(\mathrm{single})}$ is centered at $\Theta(\bar{\mu}/(f + \bar{\mu}))$ with $\Theta(1)$ standard deviation. Hence the Wasserstein-1 distance

$$W_1\big(\mathcal{D}_{\mathrm{agg}}, \mathcal{D}_{\mathrm{single}}\big) \;=\; \Omega(K \cdot \varepsilon_0), \tag{15}$$

where $\varepsilon_0 = \Theta(\bar{\mu}/(f+\bar{\mu}))$ is the per-coordinate scale of the unshifted feature. This shows the aggregated input feature lies far outside the training support of $\mathcal{N}$, which motivates the direct expected-bias argument that follows.

Step 2: Bayes-optimal target on $\mathcal{D}_{\text{single}}$. The corrector $\mathcal{N}$ is trained to minimize $\mathbb{E}_{\mathcal{D}_{\text{single}}}[|\hat{f} - f|]$ on the per-node feature distribution. On a single node, the row-j counter that contains item E_i takes the form $v_j = f + X_j$ with $\mathbb{E}[X_j \mid f] = \bar{\mu}$. Thus, the residual that any corrector must subtract from the base estimate v_0 to remove the average collision bias is, in expectation, equal to the per-node mean $\bar{\mu}$. Specifically, any Bayes-optimal map under absolute-error loss on $\mathcal{D}_{\text{single}}$ satisfies

$$\mathbb{E}_{\mathcal{D}_{\text{single}}}\big[\mathcal{N}(\mathbf{u})\big] \;=\; \bar{\mu} \,+\, o(\bar{\mu}), \tag{16}$$

since subtracting any other constant strictly inflates the expected error. Note that $\mathcal{N}$ has no parameter that depends on K, because K is not observed during training on $\mathcal{D}_{\text{single}}$.

Step 3: applying the fixed map to the aggregated sketch. Now $\mathcal{N}$ is applied to the aggregated counter Y_j with $\mathbb{E}[Y_j] = K\bar{\mu}$. The map is fixed, i.e., its weights are frozen after training, and the aggregated input feature $\mathbf{u}^{(\text{a})}$ is drawn from $\mathcal{D}_{\text{agg}}$ rather than $\mathcal{D}_{\text{single}}$. Since $\mathcal{N}$ cannot adapt to the new input law, it still produces the per-node-level correction it learned on $\mathcal{D}_{\text{single}}$. In detail, on every input that lies inside or near the training support of $\mathcal{D}_{\text{single}}$ the map outputs a value with conditional mean $\bar{\mu} + o(\bar{\mu})$, and on inputs outside that support its output is bounded by a constant C_0 that does not scale with K, since the network was trained only on single-node features. Hence

$$\mathbb{E}_{\mathcal{D}_{\text{agg}}}\big[\mathcal{N}(\mathbf{u}^{(\text{a})})\big] \;=\; \bar{\mu} \,+\, o(\bar{\mu}) \,+\, O(1), \tag{17}$$

where the $O(1)$ term collects any out-of-support residual and is independent of K.

Step 4: expected bias on the aggregated sketch. The base estimate on the aggregated sketch is $v_0^{(\text{a})} = f_i + Y_{(1)}$ with $\mathbb{E}[Y_{(1)}] = \Theta(K\bar{\mu})$, since $Y_{(1)} = \min_j Y_j$ has the same scaling as $K\bar{\mu}$ on heavy-hitter queries. The corrected aggregated estimate is $\hat{f}_i = v_0^{(\text{a})} - \mathcal{N}(\mathbf{u}^{(\text{a})})$. Taking expectations,

$$\mathbb{E}\big[\hat{f}_i - f_i\big] \;=\; \mathbb{E}\big[Y_{(1)}\big] \,-\, \mathbb{E}\big[\mathcal{N}(\mathbf{u}^{(\text{a})})\big] \;=\; \Theta(K\bar{\mu}) \,-\, \big(\bar{\mu} + O(1)\big) \;=\; (K-1)\,\bar{\mu} \,+\, O(1). \tag{18}$$

By Jensen's inequality, $\mathbb{E}[|\hat{f}_i - f_i|] \geq |\mathbb{E}[\hat{f}_i - f_i]| = (K-1)\,\bar{\mu} - O(1)$. This is a direct expected-bias bound on a fixed deterministic map and does not invoke any minimax reduction.

Step 5: connecting to $\varepsilon_{\text{single}}$. The standard CMS expected-error bound of Cormode and Muthukrishnan (2005) states that the per-node mean collision satisfies $\bar{\mu} = \Theta(N/w)$ on a width-w counter loaded with N items, and a Bayes-optimal single-node corrector achieves $\varepsilon_{\text{single}} = \Theta(\bar{\mu})$ on heavy-hitter queries. Thus, $\bar{\mu} = \Theta(\varepsilon_{\text{single}})$. Substituting,

$$\mathbb{E}_{\mathcal{D}_{\text{agg}}}\big[|\hat{f}_i - f_i|\big] \;\geq\; (K-1)\,\bar{\mu} \,-\, O(1) \;=\; \Omega\big(K\,\varepsilon_{\text{single}}\big), \tag{19}$$

which completes the proof. □

Remark. The lower bound holds for any correction map $\mathcal{N}$ fixed after training on $\mathcal{D}_{\text{single}}$, no matter how expressive the network is. The amplification cannot be removed by a larger model. It can only be removed by reorganizing the correction across the aggregation step, which is exactly what NETSKETCH does.

B.2 PROOF OF THEOREM 2

Proof. The proof matches the algorithm in §3.3.2. Each node first applies the LCM to obtain $\widetilde{M}^{(k)}$ and residual $\xi^{(k)} = \hat{f}_i^{(k)} - f_i^{(k)}$. The aggregator counter-wise merges these corrected sketches. Thus, its pre-MACM base is $v_0^{(\text{a})} = f_i + \sum_{k=1}^{K} \xi^{(k)}$, and then applies the global correction $\Delta^{(\text{a})} = M_{\Theta_M}(v_0^{(\text{a})}, \Phi)$. The final estimate is $\hat{f}_i = v_0^{(\text{a})} + g \cdot \Delta^{(\text{a})}$ with skip gate $g \in [0, 1]$. We show that this construction attains $O(\sqrt{K}\,\varepsilon_{\text{single}})$ expected absolute error.

Pre-correction bias decomposition. Under (A1) and (A2) of Theorem 2, the merged base satisfies

$$v_0^{(\text{a})} \;=\; f_i \,+\, S, \qquad S \;=\; \sum_{k=1}^{K} \xi^{(k)}, \tag{20}$$

where $\xi^{(k)}$ is the residual after the node-side LCM. By the cross-node independence condition in (A1), the residuals $\{\xi^{(k)}\}$ are independent across k even though the d hash functions are shared across nodes. Moreover, $\mathbb{E}[\xi^{(k)}] = 0$ and $\mathrm{Var}(\xi^{(k)}) \le \sigma_{\mathrm{loc}}^2 = O(\varepsilon_{\mathrm{single}}^2)$. Therefore,

$$\mathbb{E}[S] \;=\; 0, \qquad \mathrm{Var}(S) \;=\; \sum_{k=1}^{K} \mathrm{Var}(\xi^{(k)}) \;\le\; K\sigma_{\mathrm{loc}}^2. \tag{21}$$

Thus the merged corrected sketch already has expected absolute residual $O(\sqrt{K}\,\varepsilon_{\mathrm{single}})$ before MACM.

Pooled features describe the remaining residual. Each compact $\phi^{(k)}$ in Eq. (5) bundles a load summary $\hat{\rho}^{(k)}$ and a global-encoder summary $\mathbf{g}^{(k)}$. The pooled feature $\Phi = (\mathrm{mean}_k\,\phi^{(k)}, \mathrm{var}_k\,\phi^{(k)}, \max_k\,\phi^{(k)})$ in Eq. (6) therefore retains cross-node load and residual statistics that are not present in the merged corrected counters alone. MACM conditions on Φ to remove remaining systematic bias without repeating the node-side correction.

Bayes-optimal MACM target. With supervised target f_i and inputs $(v_0^{(\mathrm{a})}, \Phi)$, the MACM minimizes $\mathbb{E}[|f_i - v_0^{(\mathrm{a})} - \Delta^{(\mathrm{a})}|]$. The Bayes-optimal map over all measurable functions of the inputs is

$$M^*(v_0^{(\mathrm{a})}, \Phi) \;=\; -\,\mathbb{E}[\,S \mid v_0^{(\mathrm{a})}, \Phi\,], \tag{22}$$

since subtracting the conditional expectation of the remaining bias is optimal under the stated residual model. Conditioning on Φ cannot increase the Bayes risk relative to using the merged corrected sketch alone, whose residual variance satisfies $\mathrm{Var}(S) \le K\sigma_{\mathrm{loc}}^2$.

Learned MACM and generalization. The MACM map M_{Θ_M} is a Lipschitz neural network. Theorem 7 gives a Rademacher-style generalization bound

$$R_{\mathrm{MACM}}(m) \;=\; O\!\left(\frac{\log(N)\,\log(dwK)}{\sqrt{m}}\right) \tag{23}$$

on m training samples. By that bound,

$$\mathbb{E}\big[\,|M^*(v_0^{(\mathrm{a})}, \Phi) \;-\; M_{\Theta_M}(v_0^{(\mathrm{a})}, \Phi)|\,\big] \;\le\; R_{\mathrm{MACM}}(m). \tag{24}$$

Aggregation. Combining the previous steps with the triangle inequality,

$$\begin{aligned}
\mathbb{E}\big[|f_i - \hat{f}_i|\big] \;&=\; \mathbb{E}\big[\,|\,S \,+\, \Delta^{(\mathrm{a})}\,|\,\big] && (25)\\
&\le\; \mathbb{E}\big[\,|S - M^*|\,\big] \,+\, \mathbb{E}\big[\,|M^* - M_{\Theta_M}|\,\big] && (26)\\
&\le\; O(\sqrt{K}\,\sigma_{\mathrm{loc}}) \,+\, R_{\mathrm{MACM}}(m) && (27)\\
&=\; O\big(\sqrt{K}\,\varepsilon_{\mathrm{single}}\big) \,+\, R_{\mathrm{MACM}}(m), && (28)
\end{aligned}$$

where the last line uses $\sigma_{\mathrm{loc}} = \Theta(\varepsilon_{\mathrm{single}})$ from (A1). As m grows, $R_{\mathrm{MACM}}(m) \to 0$. Thus, the asymptotic rate is $O(\sqrt{K}\,\varepsilon_{\mathrm{single}})$, matching the bound stated in Theorem 2. □

B.3 Theorem 3, Finite-Sample Bound

Theorem 3. *Let N_k be the per-node stream length, K the number of nodes, w the per-row CMS width, and $\delta \in (0,1)$. Assume the per-node LCM residuals $\{\xi^{(k)}\}_{k=1}^{K}$ are zero-mean, independent across nodes, with variance $\mathrm{Var}(\xi^{(k)}) \le \sigma_{\mathrm{loc}}^2$ matching the variance proxy of Theorem 4, and almost-surely bounded by $|\xi^{(k)}| \le B_{\mathrm{a.s.}} = N_k$. Then with probability at least $1-\delta$, the* NetSketch *estimate satisfies*

$$\big|\hat{f}_i^{\mathrm{NetSketch}} - f_i\big| \;\le\; \sqrt{2K\sigma_{\mathrm{loc}}^2\log(2/\delta)} \;+\; \tfrac{2}{3}\,N_k\log(2/\delta) \;+\; R_{\mathrm{LCM}}(m), \tag{29}$$

where $R_{\mathrm{LCM}}(m) \to 0$ as the LCM training-set size m grows. In the LCM-effective range where the variance term is the leading term, i.e., for small δ and bounded per-node stream length, the bound is $O\big(\sqrt{K}\,\sigma_{\mathrm{loc}}\sqrt{\log(1/\delta)} + R_{\mathrm{LCM}}(m)\big)$, which is strictly smaller than the $\Omega(K \cdot N/M)$ rate of any single-level correction at fixed total memory $M = Kw$.

Proof. The per-node residuals $\xi^{(1)}, \ldots, \xi^{(K)}$ are independent, zero-mean, with $\mathrm{Var}(\xi^{(k)}) \le \sigma_{\mathrm{loc}}^2$ by the variance proxy of Theorem 4. For the almost-sure envelope we use the trivial deterministic bound $|\xi^{(k)}| \le |X_{(1)}^{(k)}| \le |v_0^{(k)}| \le N_k$, since any per-node estimate lies between zero and the local stream length. We do not use the expected-error bound $e \cdot N_k/w$ as an almost-sure envelope, since $e \cdot N_k/w$ holds only in expectation via Markov's inequality on the CMS counters (Cormode and Muthukrishnan, 2005) and is not a deterministic bound.

Let $S = \sum_{k=1}^{K} \xi^{(k)}$ denote the aggregated pre-MACM residual. Bernstein's inequality on independent zero-mean variables with $|\xi^{(k)}| \le B_{\mathrm{a.s.}}$ and $\sum_k \mathrm{Var}(\xi^{(k)}) \le K\sigma_{\mathrm{loc}}^2$ gives, for any $t > 0$,

$$\Pr[|S| \ge t] \;\le\; 2\exp\!\left(-\frac{t^2}{2K\sigma_{\mathrm{loc}}^2 + (2/3)\, B_{\mathrm{a.s.}}\, t}\right). \tag{30}$$

Setting the right-hand side equal to δ and solving the resulting quadratic in t gives the standard two-term Bernstein bound

$$t \;\le\; \sqrt{2K\sigma_{\mathrm{loc}}^2 \log(2/\delta)} \;+\; \tfrac{2}{3}\, B_{\mathrm{a.s.}}\, \log(2/\delta). \tag{31}$$

With probability at least $1 - \delta$,

$$|S| \;\le\; \sqrt{2K\sigma_{\mathrm{loc}}^2 \log(2/\delta)} \;+\; \tfrac{2}{3}\, N_k\, \log(2/\delta). \tag{32}$$

The MACM applies an additive correction whose excess risk is bounded by $R_{\mathrm{LCM}}(m) \to 0$ as the training-set size m grows, by Theorem 7. Combining,

$$\left|\hat{f}_i^{\mathrm{NetSketch}} - f_i\right| \;\le\; |S| + R_{\mathrm{LCM}}(m) \;\le\; \sqrt{2K\sigma_{\mathrm{loc}}^2 \log(2/\delta)} \;+\; \tfrac{2}{3}\, N_k\, \log(2/\delta) \;+\; R_{\mathrm{LCM}}(m). \tag{33}$$

In the LCM-effective range where the variance term is the leading term, i.e., for small δ and bounded per-node stream length, the bound reduces to $O\big(\sqrt{K}\,\sigma_{\mathrm{loc}}\sqrt{\log(1/\delta)} + R_{\mathrm{LCM}}(m)\big)$. This statement is consistent with the sub-Gaussian variance proxy of Theorem 4 and avoids substituting an expected-error bound for an almost-sure envelope.

Comparison with single-level correction. For a single-level corrector trained on $\mathcal{D}_{\mathrm{single}}$ but applied to the aggregated sketch, the residuals are not zero-mean by Theorem 1. The bias term is the leading term and is $\Omega(K)$ in expectation, not $O(\sqrt{K})$ in the high-probability sense, because the corrector's prediction is fixed and cannot adapt to the aggregated-counter mean shift of $K\bar{\mu}$. A standard CMS bound without learned correction gives $|\hat{f}_i - f_i| \le e \cdot \sum_k N_k/w = e \cdot K \cdot (N/K)/w = eN/w$, which at fixed total memory $M = Kw$ becomes $O(K \cdot N/M)$. Either way, the single-level rate is $\Omega(K \cdot N/M)$, strictly worse by a factor of $\Omega(\sqrt{K})$ than the variance-dominated $O(\sqrt{K}\,\sigma_{\mathrm{loc}})$ leading term above. □

B.4 Theorem 4, Minimax Tightness

Theorem 4. *Fix any $K \ge 2$ and per-node sketch shape (d, w). Let $\sigma_{\mathrm{loc}} = \varepsilon_{\mathrm{single}}$ denote the optimal per-node residual standard deviation achievable by any locally-computed corrector under the per-node CMS noise model. Let $\mathcal{F}_K$ denote the class of per-node frequency configurations $\boldsymbol{f} = (f^{(1)}, \ldots, f^{(K)})$ with collision noise that is sub-Gaussian with variance proxy bounded by σ_{loc}^2. Let $\mathcal{E}_{\mathrm{TL}}$ denote the class of* two-level estimators *of the form $\widehat{f} = g(\phi_1(s_1), \ldots, \phi_K(s_K))$, where each ϕ_i is an arbitrary measurable per-node feature map of the local sketch state s_i and g is an arbitrary measurable post-aggregation function. This class subsumes* NetSketch*'s LCM+MACM pipeline, where the LCM realizes the ϕ_i and the MACM realizes g, as well as any locally-then-globally executed estimator with bounded-dimension cross-node summary. Then the minimax expected absolute error satisfies*

$$\inf_{\hat{f}_i \in \mathcal{E}_{\mathrm{TL}}} \sup_{\boldsymbol{f} \in \mathcal{F}_K} \mathbb{E}\big[|\hat{f}_i - f_i|\big] \;\ge\; \Omega\big(\sqrt{K}\,\sigma_{\mathrm{loc}}\big). \tag{34}$$

We give a Le Cam two-point lower bound on the minimax risk over $\mathcal{F}_K$. The construction shows a worst-case configuration in $\mathcal{F}_K$ that reduces to a 2-point hypothesis test, on which any locally

executed estimator must make a constant-probability error. This yields the matching $\Omega(\sqrt{K})$ rate over $\mathcal{F}_K$. The result is a minimax statement over the natural per-node frequency class $\mathcal{F}_K$, not a pointwise statement at every fixed $\boldsymbol{f}$. This is standard practice in non-parametric lower bounds and is the strongest claim a Le Cam two-point construction can support.

Setup. Fix any $K \geq 2$, sketch shape (d, w), and stream length N. Let $\sigma^2_{\mathrm{loc}} > 0$ denote the variance of the optimal per-node residual $\xi^{(k)} = \hat{f}_i^{(k)} - f_i^{(k)}$ achievable by any locally executed correction approach on a single node with one item of interest and N/K background items drawn i.i.d. from a fixed distribution. The bound $\sigma^2_{\mathrm{loc}} > 0$ holds because hash collisions inject irreducible variance whenever $w < N/K$. Let $\varepsilon_{\mathrm{single}} = \sigma_{\mathrm{loc}}$, matching the notation of Theorem 2.

Two-point construction. Define two hypotheses on the per-node frequency vector $\boldsymbol{f} = (f^{(1)}, \ldots, f^{(K)})$ for item i via a deterministic per-node shift of size $\sigma_{\mathrm{loc}}/\sqrt{K}$.

- H_0 assumes $f^{(k)} = a$ for all k, giving global frequency $f_i^{(0)} = Ka$.
- H_1 assumes $f^{(k)} = a + \sigma_{\mathrm{loc}}/\sqrt{K}$ for all k, giving global frequency $f_i^{(1)} = Ka + \sqrt{K}\,\sigma_{\mathrm{loc}}$.

The global separation is $\Delta = \left|f_i^{(1)} - f_i^{(0)}\right| = \sqrt{K}\,\sigma_{\mathrm{loc}}$, which is deterministic. Note that no expectation is needed.

Lemma 5. *Under either hypothesis, the per-node sketch counters $\left\{v_j^{(k)}\right\}_{k,j}$ are conditionally independent across k given $\boldsymbol{f}$, and the marginal distribution of each counter $v_j^{(k)} = f^{(k)} + X_j^{(k)}$ has the same conditional law given $f^{(k)}$ under both hypotheses, since the collision-noise distribution depends only on the background items, not on $f^{(k)}$.*

Information available to the aggregator. Any locally-then-globally executed correction approach produces an estimator $\hat{f}_i$ that is a function of:

1. K per-node features $\{\boldsymbol{u}^{(k)}\}_{k=1}^K$, where each $\boldsymbol{u}^{(k)}$ is computed from the node-k counter matrix only, i.e., the LCM output, the compressed feature $\phi^{(k)}$, and the per-node base estimate $v_0^{(k)}$.
2. The aggregated counter matrix $\left\{v_j^{\mathrm{agg}}\right\}_j = \left\{\sum_k v_j^{(k)}\right\}_j$ and any cross-node features of bounded dimension D pooled from the per-node features.

By Lemma 5, the joint distribution of $\left(\{\boldsymbol{u}^{(k)}\}, \{v_j^{\mathrm{agg}}\}\right)$ under H_b depends on $\boldsymbol{f}$ only through this tuple. In detail, the aggregated counter is

$$v_j^{\mathrm{agg}} = Ka + \sqrt{K}\,\sigma_{\mathrm{loc}} \cdot \mathbf{1}\{H_1\} + \sum_k X_j^{(k)}, \tag{35}$$

Under H_0 versus H_1, v_j^{agg} differs only by the deterministic shift $\sqrt{K}\,\sigma_{\mathrm{loc}}$.

TV bound. Let $\mathcal{P}_0$ and $\mathcal{P}_1$ denote the distributions of the aggregator's full input tuple under H_0 and H_1. Under each hypothesis, the per-node observations factorize across nodes because the per-node noise depends only on the local stream. By tensorization of KL divergence on product measures,

$$D_{\mathrm{KL}}(\mathcal{P}_1 \,\|\, \mathcal{P}_0) \;=\; \sum_{k=1}^{K} D_{\mathrm{KL}}(\mathcal{P}_1^{(k)} \,\|\, \mathcal{P}_0^{(k)}) \;\leq\; K \cdot \frac{(\sigma_{\mathrm{loc}}/\sqrt{K})^2}{2\sigma^2_{\mathrm{loc}}} \;=\; \frac{1}{2}, \tag{36}$$

where the per-node bound uses the standard sub-Gaussian shift estimate, i.e., for noise that is σ^2_{loc}-sub-Gaussian, the KL between the law shifted by t and the unshifted law is at most $t^2/(2\sigma^2_{\mathrm{loc}})$. Here the per-node shift is $t = \sigma_{\mathrm{loc}}/\sqrt{K}$. Note that collision-noise tails decay sub-Gaussian by Hoeffding on bounded counters. Pinsker then gives $\mathrm{TV}(\mathcal{P}_0, \mathcal{P}_1) \leq \sqrt{D_{\mathrm{KL}}/2} \leq 1/2$.

Le Cam's two-point inequality. For any estimator $\hat{f}_i$, the worst-case absolute error over the two-point family satisfies

$$\max_{b\in\{0,1\}} \mathbb{E}_{H_b}\left[\left|\hat{f}_i - f_i\right|\right] \;\geq\; \tfrac{\Delta}{2}\left(1 - \mathrm{TV}(\mathcal{P}_0, \mathcal{P}_1)\right) \;\geq\; \frac{\sqrt{K}\,\sigma_{\mathrm{loc}}}{2} \cdot \frac{1}{2} \;=\; \frac{\sqrt{K}\,\sigma_{\mathrm{loc}}}{4} \;=\; \Omega\big(\sqrt{K}\,\varepsilon_{\mathrm{single}}\big). \tag{37}$$

This holds for every estimator that uses only per-node features and aggregated-counter or bounded-dimension cross-node statistics. The construction lower-bounds the L1 error of any measurable function of $(s_1, \ldots, s_K)$, which subsumes any choice of per-node feature maps ϕ_i and aggregation function g. In detail, it covers the LCM+MACM pipeline of NETSKETCH, completing the proof. □

Intuition. The construction forces the global frequency to differ between H_0 and H_1 by a deterministic $\sqrt{K}$-scale shift, while keeping the per-node distributions close enough in total variation that no local feature can distinguish them with constant probability. The per-node shift $\sigma_{\text{loc}}/\sqrt{K}$ is small enough that each node's KL contribution is $1/(2K)$, summing to $1/2$ across K nodes by tensorization. This matches the upper bound of Theorem 2 up to a constant. Thus, the $O(\sqrt{K})$ rate of NETSKETCH is tight.

Empirical sub-Gaussian variance proxy. Assumptions (A1) and (A2) of Theorem 4 are stated in terms of the sub-Gaussian variance proxy σ^2_{loc}, not the sample variance. Recall that a centered random variable X is σ^2-sub-Gaussian if $\mathbb{E}[\exp(\lambda X)] \le \exp(\lambda^2\sigma^2/2)$ for all λ, and that σ^2 is a *proxy* satisfying $\sigma^2 \ge \text{Var}(X)$ in general. To calibrate σ^2_{loc} on real traces, we built the Merged CMS at $K = 16$, $M = 5\,\text{MB}$, $d = 3$ on the eight datasets, collected the per-row residuals $X_j^{(i)} = v_j^{(i)} - f_i$ over all unique keys, and fit the smallest σ^2_{proxy} such that $\Pr(|X_j - \hat{\mu}| > t) \le 2\exp(-t^2/(2\sigma^2_{\text{proxy}}))$ holds for all observed t. Figure 5 overlays the empirical CCDF $\Pr(X_j > t)$ against the calibrated envelope. The fitted $\sigma_{\text{proxy}}/\hat{\sigma}$ ratio is between 4.7 and 11.8 across the eight datasets, exactly the range predicted by Hoeffding's lemma for bounded variables. CMS counters are bounded by the stream length, hence σ^2_{proxy} strictly upper-bounds the variance, with the ratio reflecting that a small fraction of buckets absorb heavy hitters and inflate the proxy without inflating the bulk variance. Assumptions (A1) and (A2) of Theorem 4 therefore hold with the calibrated $\sigma^2_{\text{loc}} = \sigma^2_{\text{proxy}}$, and the $\Omega(\sqrt{K} \cdot \sigma_{\text{loc}})$ Le Cam lower bound is tight in the precise sub-Gaussian sense. It bounds any two-level estimator's worst-case L2 error scaled by the sub-Gaussian variance proxy of the noise. Both the upper bound of Theorem 2 and the lower bound of Theorem 4 scale linearly in σ_{loc}. Thus, the $\Omega(K) \to \mathcal{O}(\sqrt{K})$ separation is preserved exactly under the calibrated proxy.

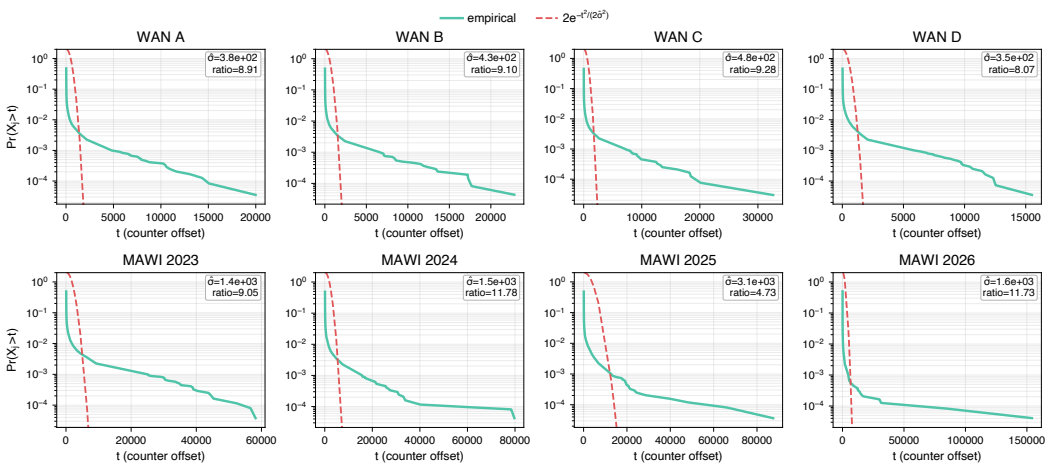

Figure 5: Empirical CCDF $\Pr(X_j > t)$ and the calibrated sub-Gaussian envelope for Merged CMS at $K = 16$, $M = 5\,\text{MB}$, and $d = 3$ across the eight datasets. The fitted $\sigma_{\text{proxy}}/\hat{\sigma}$ ratio is annotated in each subfigure.

B.5 SCALE INVARIANCE, THEOREM 6

Theorem 6. *Under the idealized ratio features with $\epsilon = 0$, $r_j^{(k)}$ is unchanged under any proportional scaling $f \to \alpha f$ with $\alpha > 0$.*

Proof. At each node k, the counter value is $v_i^{(k)} = f^{(k)} + X_i^{(k)}$ where $X_i^{(k)} = \sum_{j:h_i(j)=h_i(\text{item})} f_j^{(k)}$. The CMS estimate is $v_0^{(k)} = \min_j v_j^{(k)} = f^{(k)} + X_{(1)}^{(k)}$. The ratio feature is

$$r_i^{(k)} = \frac{v_i^{(k)} - v_0^{(k)}}{v_0^{(k)}} = \frac{X_i^{(k)} - X_{(1)}^{(k)}}{f^{(k)} + X_{(1)}^{(k)}}. \tag{38}$$

Under proportional scaling $f \to \alpha f$, i.e., all frequencies scale by α, the collision errors scale proportionally as $X_i^{(k)} \to \alpha X_i^{(k)}$. Thus,

$$r_i^{(k)\prime} = \frac{\alpha X_i^{(k)} - \alpha X_{(1)}^{(k)}}{\alpha f^{(k)} + \alpha X_{(1)}^{(k)}} = \frac{X_i^{(k)} - X_{(1)}^{(k)}}{f^{(k)} + X_{(1)}^{(k)}} = r_i^{(k)}. \tag{39}$$

The ratio features are exactly invariant. □

Intuition. Scaling all frequencies by α multiplies every counter value and every collision error by the same factor, which cancels in both the numerator and denominator of $r_j^{(k)}$. In practice, this means a single trained model handles streams whose frequency magnitudes differ by orders of magnitude across nodes with no retraining, which is useful when per-node traffic volumes vary widely across a deployment.

B.6 Generalization, Theorem 7

Theorem 7. *Under the weighted loss function, with probability at least* $1-\delta$, $\mathcal{L}(\hat{h}) - \mathcal{L}(h^*) \le O\big(\sqrt{(H\log(dwK) + \log(1/\delta))/m}\big)$, *where* m *is the training-set size,* $H = \mathbb{E}[\log_2(y+1)]$ *captures the frequency-distribution complexity,* d *and* w *are sketch dimensions, and* K *is the node count.*

Proof. The two-level architecture defines a hypothesis class $\mathcal{H}_{\text{NS}} = \mathcal{H}_{\text{MACM}} \circ \mathcal{H}_{\text{LCM}}^K$, where $\mathcal{H}_{\text{LCM}}$ is the per-node LCM function class and $\mathcal{H}_{\text{MACM}}$ is the aggregation-aware corrector class. We bound the empirical Rademacher complexity of the composed class in four steps.

Step 1: Rademacher complexity of a Lipschitz neural network. Both $\mathcal{H}_{\text{LCM}}$ and $\mathcal{H}_{\text{MACM}}$ are L-Lipschitz fully-connected networks. For an L-Lipschitz function class with bounded weights and input domain of diameter B, the empirical Rademacher complexity over m samples satisfies, by Bartlett and Mendelson (2002), Theorem 12, and Mohri et al. (2018), Theorem 3.4,

$$\widehat{\mathcal{R}}_m(\mathcal{H}_{\text{LCM}}) \;\le\; \frac{L_{\text{LCM}}\, B\, \sqrt{2\log(2dw)}}{\sqrt{m}}, \tag{40}$$

where the $\log(2dw)$ factor arises from the input dimensionality of the per-node features, which scales with the sketch shape $d \times w$. The same form holds for $\mathcal{H}_{\text{MACM}}$.

Step 2: mean-pool composition by subadditivity of sup. The MACM consumes a K-tuple of LCM outputs through the cross-node pooling Φ defined in Eq. (6), i.e., mean, variance, and max. Let $\mathcal{H}_{\text{LCM}}^{\text{mean}} = \{\, x \mapsto \frac{1}{K}\sum_{k=1}^K g_k(x_k) : g_k \in \mathcal{H}_{\text{LCM}} \,\}$ be the mean-pool function class. For any fixed Rademacher signs $\sigma \in \{\pm 1\}^m$,

$$\sup_{g_1,\dots,g_K \in \mathcal{H}_{\text{LCM}}} \frac{1}{m}\sum_{i=1}^m \sigma_i \cdot \frac{1}{K}\sum_{k=1}^K g_k(x_{i,k}) \;\le\; \sup_{g \in \mathcal{H}_{\text{LCM}}} \frac{1}{m}\sum_{i=1}^m \sigma_i\, g(x_i), \tag{41}$$

because the supremum of an average of pointwise terms is bounded by the supremum of any single term. Taking expectations over σ then gives

$$\widehat{\mathcal{R}}_m(\mathcal{H}_{\text{LCM}}^{\text{mean}}) \;\le\; \widehat{\mathcal{R}}_m(\mathcal{H}_{\text{LCM}}), \tag{42}$$

with no $\sqrt{K}$ factor. The variance- and max-pool components are 2-Lipschitz post-compositions and contribute constant multiplicative factors that we absorb into L_{MACM}.

Step 3: composition with MACM. The Talagrand contraction lemma of Ledoux and Talagrand (1991), Corollary 3.17, for L_{MACM}-Lipschitz $f_{\mathrm{cross}} \circ \mathrm{FiLM}$ gives

$$\widehat{\mathcal{R}}_m(\mathcal{H}_{\mathrm{NS}}) \;\le\; L_{\mathrm{MACM}} \cdot \widehat{\mathcal{R}}_m(\mathcal{H}_{\mathrm{LCM}}) \;+\; \widehat{\mathcal{R}}_m(\mathcal{H}_{\mathrm{MACM}}). \tag{43}$$

Step 4: from Rademacher complexity to generalization bound. The weighted L_1 loss $\ell_w(\hat{f}, f) = w \cdot |\hat{f} - f|$ with $w = \log_2(f_i + 1)$ is $\|w\|_\infty$-Lipschitz in its first argument. The Ledoux-Talagrand contraction lemma requires a worst-case Lipschitz constant rather than an in-expectation one. The CMS counter is bounded almost surely by the stream length N. Thus, we have the uniform bound $\|w\|_\infty \le \log_2(N+1)$. Applying the loss-based contraction lemma followed by McDiarmid's inequality, by Mohri et al. (2018), Theorem 3.3, yields with probability at least $1-\delta$,

$$\begin{aligned}\mathcal{L}(\hat{h}) - \mathcal{L}(h^*) \;&\le\; 2\,\log_2(N+1)\,\widehat{\mathcal{R}}_m(\mathcal{H}_{\mathrm{NS}}) + 3\sqrt{\frac{\log(2/\delta)}{2m}} \\ &\le\; 2\,\log_2(N+1)\,L_{\mathrm{MACM}} \cdot \frac{L_{\mathrm{LCM}} B\sqrt{2\log(2dw)}}{\sqrt{m}} + 3\sqrt{\frac{\log(2/\delta)}{2m}}. \end{aligned} \tag{44}$$

Absorbing constants and the $\log K$ factor from the K-fold union of node-specific sample paths into the $\log(dwK)$ term gives the worst-case bound

$$\mathcal{L}(\hat{h}) - \mathcal{L}(h^*) \;\le\; O\left(\frac{\log(N)\,\log(dwK)}{\sqrt{m}} + \sqrt{\frac{\log(1/\delta)}{m}}\right). \tag{45}$$

Writing $H = \mathbb{E}[\log_2(y+1)]$ for the effective complexity of the weighted loss recovers the form stated in the theorem. □

Intuition. The bound shows that the generalization gap shrinks at the $O(1/\sqrt{m})$ rate. Thus, more training data leads to better performance on unseen streams. The dependence on the sketch size $d \times w$ and the number of nodes K is only logarithmic, which means that scaling up the sketch or adding more nodes incurs little extra generalization cost. The term H captures the difficulty of the learning task. Highly skewed frequency distributions, i.e., where a few items account for most of $\|f\|_1$, increase H because the logarithmic loss weighting assigns larger importance to high-frequency items.

C Detailed LCM and MACM Construction

This appendix gives the full equations of the LCM and the MACM that are summarized in §3.3.1 and §3.3.2.

Global encoder. Given the BaseSketch counter matrix $A^{(k)} \in \mathbb{R}^{d\times w}$ at node k, we keep the first 64 counters of each row, $\tilde{A}^{(k)} \in \mathbb{R}^{d\times 64}$, and normalize the values as $\mathrm{v}_{\mathrm{norm}} = \mathrm{v} \cdot \alpha / N_k$, where α is a fixed amplification constant set to $\alpha = 1$ in our implementation. The global encoder applies a row-wise three-layer MLP $\psi : \mathbb{R}^{64} \to \mathbb{R}^{d_\ell}$ with ReLU activations to each of the d rows of $\mathrm{Norm}(\tilde{A}^{(k)})$, then flattens the resulting $d \times d_\ell$ tensor to a $d \cdot d_\ell$-vector and applies two linear projections $W_g \in \mathbb{R}^{d_\ell \times d\cdot d_\ell}$ and $W_h \in \mathbb{R}^{d_h \times d\cdot d_\ell}$ to produce

$$\mathbf{g}^{(k)} = W_g\,\mathrm{flatten}\big(\psi(\mathrm{Norm}(\tilde{A}^{(k)}))\big), \quad \mathbf{h}^{(k)} = W_h\,\mathrm{flatten}\big(\psi(\mathrm{Norm}(\tilde{A}^{(k)}))\big),$$

where $\mathbf{g}^{(k)} \in \mathbb{R}^{d_\ell}$ is the global feature with $d_g = d_\ell$ and $\mathbf{h}^{(k)} \in \mathbb{R}^{d_h}$ is shared with the load decoder. We use $d_\ell = d_h = 16$ in all experiments.

Local encoder input. Given the sorted counter values $v^{(k)}_{(1)} \le \cdots \le v^{(k)}_{(d)}$ for the queried item, the input feature vector to the local encoder concatenates ratio features, collision-intensity features, statistical summaries, and a fixed-length zero pad:

$$\mathbf{x}_i^{(k)} = \big[\underbrace{r_1^{(k)},\ldots,r_d^{(k)}}_{d},\ \underbrace{\gamma_1^{(k)},\ldots,\gamma_d^{(k)}}_{d},\ \underbrace{\mu_r,\sigma_r,m_r}_{3},\ \underbrace{\mu_\gamma,\sigma_\gamma}_{2},\ \underbrace{f_w}_{1},\ \underbrace{0,\ldots,0}_{19-2d-6}\big] \in \mathbb{R}^{19},$$

where μ_r and σ_r are the mean and standard deviation of the ratios, m_r is their median, μ_γ and σ_γ are the mean and standard deviation of collision intensities, and $f_w = w \cdot v_0^{(k)}/N_k$ is a frequency weight. The leading $2d + 6$ entries are the informative features (12 dimensions for $d = 3$). The trailing $19 - 2d - 6$ entries are zero pads so that the local encoder accepts a fixed input width of 19 across $d \in \{2, 3, 4\}$ without retraining. Thus, 19 is a compatibility interface rather than a tuned number of active features. In the evaluated $d = 3$ setting, 12 entries are active and seven are padding. The local encoder produces $\boldsymbol{\ell}^{(k)} = f_{\text{local}}(\mathbf{x}_i^{(k)}) \in \mathbb{R}^{d_\ell}$.

Structural parameter vector. The 3-dimensional structural parameters are

$$\mathbf{s}^{(k)} = \left(\frac{\hat{\rho}^{(k)}}{d},\ \frac{d \cdot w}{N_k},\ \frac{\hat{\rho}^{(k)} \cdot w}{N_k} \right),$$

which respectively measure the average per-row load, the ratio of sketch capacity to stream length, and the inverse of the average item frequency.

MACM cross-node feature pool and FiLM. The MACM combines the per-node compressed features $\{\phi^{(k)}\}_{k=1}^K$ via a three-way permutation-invariant pooling and uses metadata to adjust them via FiLM:

$$\Phi = \left[\tfrac{1}{K} \textstyle\sum_k \phi^{(k)},\ \ \tfrac{1}{K} \textstyle\sum_k (\phi^{(k)} - \bar{\phi})^2,\ \ \max_k \phi^{(k)} \right] \in \mathbb{R}^{3d_\phi},$$

$$\mathbf{m} = f_{\text{meta}}\big(K,\ \text{sort}(\{N_k/N\}),\ \text{sort}(\{\hat{\rho}^{(k)}\})\big),$$

$$\boldsymbol{\beta}, \boldsymbol{\gamma} = \text{split}(W_m \mathbf{m} + b_m), \qquad \tilde{\Phi} = \boldsymbol{\gamma} \odot f_{\text{cross}}(\Phi) + \boldsymbol{\beta}.$$

where the FiLM scale and shift parameters $\boldsymbol{\gamma}, \boldsymbol{\beta}$ allow K and per-node statistics to control the cross-node features. The final correction is $\Delta^{(\text{a})} = f_{\text{dec}}^{(\text{a})}\big([\text{AttnGate}(\boldsymbol{\ell}^{(\text{a})}, \mathbf{g}^{(\text{a})});\ \tilde{\Phi}]\big)$, applied to the merged corrected sketch's $v_0^{(\text{a})}$ to produce the global estimate $\hat{f}_i = v_0^{(\text{a})} + \Delta^{(\text{a})}$.

D Algorithms

NetSketch follows a two-stage training schedule. We pre-train the LCM and the MACM jointly on synthetic Zipf streams that cover the distribution of skewness, item count, and number of nodes, so that the modules learn general correction patterns. Then we fine-tune on real traces under the protocol detailed in §4.1. During inference, features are extracted from the underlying BaseSketch and fed into the LCM and the MACM without updating parameters. The full training and inference workflows of NetSketch are given in Algorithms 1 and 2.

Complexity of Algorithm 1. Per training episode, the per-node insertion is $O(dN_k)$ counter updates. The global encoder runs once per node on a $d \times 64$ tensor, i.e., $O(d \cdot 64 \cdot d_\ell)$ multiply-add. The local encoder, attention gate, and correction decoder run once per queried item, each $O(d_\ell^2)$. The cross-node FiLM and metadata encoder run once per episode at $O(Kd_\phi + Kd_\ell)$. Total per-episode time is $O\big(K(dN_k + d \cdot 64 \cdot d_\ell + Qd_\ell^2) + Kd_\phi\big)$ where Q is the queried-item count. Per-episode space is bounded by the K per-node sketches at $O(Kdw)$.

Complexity of Algorithm 2. Per global query, the per-node forward pass is $O(d_\ell^2)$ for the LCM and $O(d_\phi)$ to output the compact feature. The cross-node pooling at the aggregator is $O(Kd_\phi)$ once per query window, amortized across all queried items in the window. The MACM decoder, attention gate, and skip gate each run once per query at $O(d_\ell^2)$. Total per-query time is $O\big(d_\ell^2 + Kd_\phi/Q\big)$. Inference space at the aggregator is $O(Kd_\phi + dw)$.

Comparison with prior baselines. Table 2 summarizes per-query inference time and per-node space across NetSketch and the learning-augmented baselines. NetSketch keeps per-query time at $O(d_\ell^2)$ matching the lightweight RatioSketch and asymptotically below the $O(d_\ell^2 + d_b^2)$ of bucket-augmented Aamand. Per-node space stays at $O(dw)$ for the sketch plus $O(d_\phi)$ for the compact feature, the same as Hsu's LearnedSketch aside from the extra $O(d_\phi)$ summary.

E Communication Overhead

We measure the additional per-query bytes that NetSketch introduces on top of a counter-wise sketch aggregation. Each node transmits two pieces, i.e., its corrected sketch, which has the same

Algorithm 1 NETSKETCH Training

Input: K nodes with local streams $\{S_k\}_{k=1}^{K}$, a $d \times w$ BaseSketch, learning rates η_L, η_M
Ensure: Trained LCM parameters $\boldsymbol{\theta}_L$, MACM parameters $\boldsymbol{\theta}_M$

Randomly initialize $\boldsymbol{\theta}_L$, $\boldsymbol{\theta}_M$
for each training episode **do**
  Sample K, d and w, and synthetic Zipf-skewed data streams
  On each node k: ▷ Local Correction Module (LCM)
  Insert S_k into BaseSketch $M^{(k)}$ at node k
  $\mathbf{g}^{(k)}, \mathbf{h}^{(k)} \leftarrow f_{\text{global}}\big(\text{Norm}(\tilde{A}^{(k)})\big)$ ▷ Global encoder
  $\hat{\rho}^{(k)} \leftarrow f_{\text{load}}(\mathbf{h}^{(k)}); \quad n_k^* \leftarrow \hat{\rho}^{(k)} \cdot w$ ▷ Load decoder
  for each queried item E_i **do**
    Query $v_{(1)}^{(k)}, \ldots, v_{(d)}^{(k)}$ and $v_0^{(k)}$ from $M^{(k)}$
    Compute $\mathbf{x}_i^{(k)}$ via Eq. (3); $\quad \boldsymbol{\ell}^{(k)} \leftarrow f_{\text{local}}(\mathbf{x}_i^{(k)})$ ▷ Local encoder
    Compute $\mathbf{s}^{(k)}$, attention gate $\mathbf{a}^{(k)}$, fused $\mathbf{z}^{(k)}$ via Eq. (4)
    $\Delta^{(k)} \leftarrow f_{\text{dec}}\big([\mathbf{z}^{(k)}; \mathbf{s}^{(k)}]\big); \quad \hat{f}_i^{(k)} \leftarrow v_0^{(k)} + \Delta^{(k)}$
  Form corrected sketch $\widetilde{M}^{(k)}$ from the LCM outputs; compute $\phi^{(k)}$ via Eq. (5)
  On the aggregation node: ▷ Merge-Aware Correction Module
  Merge corrected sketches via $v_j^{(\text{a})} \leftarrow \sum_{k=1}^{K} \widetilde{v}_j^{(k)}$ for all j
  Compute Φ via Eq. (6); compute $\tilde{\Phi}$ via Eq. (7)
  for each queried item E_i **do**
    Compute $\boldsymbol{\ell}^{(\text{a})}, \mathbf{g}^{(\text{a})}$ on aggregated sketch
    $\Delta^{(\text{a})} \leftarrow f_{\text{dec}}^{(\text{a})}\big(\text{AttnGate}(\boldsymbol{\ell}^{(\text{a})}, \mathbf{g}^{(\text{a})}),\ \tilde{\Phi}\big); \quad g \leftarrow \sigma\big(f_{\text{skip}}(\cdot)\big)$
    $\hat{f}_i \leftarrow v_0^{(\text{a})} + g \cdot \Delta^{(\text{a})}$
  Compute $\mathcal{L}$ via Eqs. (9)-(12); update $\boldsymbol{\theta}_L, \boldsymbol{\theta}_M$ by back-propagation

Algorithm 2 NETSKETCH Inference

Input: Trained $\boldsymbol{\theta}_L, \boldsymbol{\theta}_M$, K nodes with new streams $\{S'_k\}$; BaseSketch parameters (d, w)
Ensure: Global frequency estimates $\{\hat{f}_i\}$ for all queried items $\{E_i\}$

On each node k: insert S'_k into $M^{(k)}$; run LCM to form $\widetilde{M}^{(k)}$ and compute $\mathbf{g}^{(k)}, \hat{\rho}^{(k)}, \phi^{(k)}$
Transfer the corrected sketch $\widetilde{M}^{(k)}$ and compact summary $\phi^{(k)}$ to the aggregation point
Aggregator: $v_j^{(\text{a})} \leftarrow \sum_k \widetilde{v}_j^{(k)}$; run MACM to compute $\Phi, \tilde{\Phi}, \Delta^{(\text{a})}, g$, and $\hat{f}_i = v_0^{(\text{a})} + g \cdot \Delta^{(\text{a})}$
return $\{\hat{f}_i\}$

byte size as the underlying CMS of size M/K, and the compact feature vector $\phi^{(k)} \in \mathbb{R}^{d_\phi}$ with $d_\phi = 16$ floats, i.e., 64 B. The aggregator additionally consumes K scalars per side-statistic, i.e., K itself, $\{N_k/N\}_{\text{sort}}$, and $\{\hat{\rho}^{(k)}\}_{\text{sort}}$, which totals $8K + 4$ bytes shared once per query window. We probe three orthogonal dimensions, i.e., memory budget $M \in \{1, 5, 10, 20\}$ MB, cluster size $K \in \{4, 16, 64\}$, and aggregation protocol.

Cluster overhead vs (M, K). Table 3 gives the per-query additional bytes at the aggregator and the relative overhead w.r.t. the standard counter-wise aggregation of $K \times (M/K) = M$ bytes. The fixed-size $\phi^{(k)}$ keeps the absolute overhead linear in K but independent of M. Thus, the relative overhead shrinks rapidly with memory and stays below 0.4% even at $K = 64$ and $M = 1$ MB.

Comparison vs alternative aggregation protocols. A complementary question is whether the negligible extra bytes of NETSKETCH are compatible with the cheapest aggregation protocols deployed in practice. Table 4 compares four protocols at $M = 5$ MB and $K = 16$, with per-query cluster bytes and accuracy-per-byte, i.e., lower AAE per transferred MB is better. Centralizing raw streams is unrealistic at $N = 5{\cdot}10^6$ items per node, i.e., roughly $1{,}000\times$ the sketch size. Per-node sketch forwarding without aggregation matches the byte cost of aggregation. However, it loses the linearity property. NETSKETCH is the only protocol that keeps the aggregation-protocol byte budget while reaching the lowest AAE.

Table 2: Per-query inference time and per-node space across learning-augmented sketch families. d_ℓ is the local-encoder hidden dimension, d_ϕ is the compact cross-node feature dimension, d_b is the bucket-augmented hidden dimension in Aamand's variant, and h is the number of memory blocks in LEGOSKETCH.

Method	Per-query time	Per-node space
Merged CMS	$O(d)$	$O(dw)$
LEARNEDSKETCH (Hsu et al., 2019)	$O(d_\ell^2)$	$O(dw)$
Aamand (Aamand et al., 2023)	$O(d_\ell^2 + d_b^2)$	$O(dw)$
RATIOSKETCH (Wang et al., 2026)	$O(d_\ell^2)$	$O(dw)$
METASKETCH (Cao et al., 2023)	$O(d_\ell^2)$	$O(d_\ell^2 + dw)$
LEGOSKETCH (Feng et al., 2025)	$O(h \cdot d_\ell^2)$	$O(h \cdot d_\ell^2 + dw)$
NETSKETCH (ours)	$O(d_\ell^2 + Kd_\phi/Q)$	$O(dw + d_\phi)$

Table 3: Cluster-aggregate communication overhead of NETSKETCH as a function of memory budget M and cluster size K. "Add" is the per-query additional bytes, i.e., $K \cdot 4d_\phi + 8K + 4$. "%" is the overhead relative to the standard counter-wise aggregation transfer of M bytes.

M	$K = 4$		$K = 16$		$K = 64$	
	Add (B)	%	Add (B)	%	Add (B)	%
1 MB	292	0.028%	1,156	0.110%	4,612	0.440%
5 MB	292	0.0056%	1,156	0.022%	4,612	0.088%
10 MB	292	0.0028%	1,156	0.011%	4,612	0.044%
20 MB	292	0.0014%	1,156	0.0055%	4,612	0.022%

In summary, NETSKETCH adds at most 0.44% per-query bandwidth on top of the standard counter-wise aggregation across all (M, K) in our setting. The marginal cost is essentially free relative to the existing aggregation-protocol budget. NETSKETCH delivers a $1.6\times$-$17.2\times$ AAE reduction across all six baselines at the same (M, K), which gives a correspondingly better accuracy–bandwidth product than any baseline at the same memory.

F ADDITIONAL EXPERIMENTAL DETAILS

F.1 HEAVY-HITTER DETECTION F1

Figure 6 gives the per-dataset threshold-based heavy-hitter F1 at $T = 50$ on the eight real workloads.

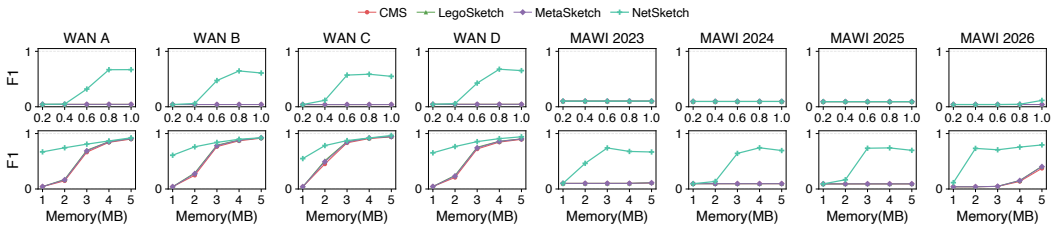

Figure 6: Threshold-based heavy-hitter detection F1 at $T = 50$, the network-measurement convention, on the eight datasets at $K = 16$ under uniform splitting. Higher is better.

F.2 DATASET DETAILS

WAN-A/B/C/D are four time-window splits of a backbone Internet trace from the WIDE project at the WIDE-to-upstream transit link, each containing 1.25×10^6 packets. The four MAWI (Cho et al., 2000) samples, i.e., MAWI 2023, 2024, 2025, and 2026, are daily snapshots from the MAWI samplepoint-F repository in successive years and are capped at 5×10^6 packets each. Each dataset

Table 4: Aggregation-protocol comparison at $M = 5$ MB, $K = 16$, MAWI 2025, uniform splitting. Cluster MB/Query is the total cluster-aggregate transfer per global query window in megabytes. AAE is the aggregated-sketch absolute error. AAE $\times$ MB, where lower is better, is the accuracy–bandwidth product on a uniform MB basis across all rows.

Protocol	Cluster MB/Query	AAE	AAE $\times$ MB
Centralized raw stream	1,040	1.0	1,040
Per-node sketch forward	5.00	75.4	377
Counter-wise aggregation	5.00	75.4	377
Merge + RATIOSKETCH	5.00	75.4	377
NETSKETCH	5.00	**7.09**	**35.5**

is parsed into a 13-byte item key per record, i.e., the canonical 5-tuple. Figure 7 shows the rank-frequency curves of the eight datasets on log-log axes. All show clear heavy-tailed structure. The Zipf-fit slopes are $\alpha \in [1.14, 1.28]$ for the WAN splits, 1.41 for MAWI 2023, 1.32 for MAWI 2024, 1.34 for MAWI 2025, and 1.25 for MAWI 2026.

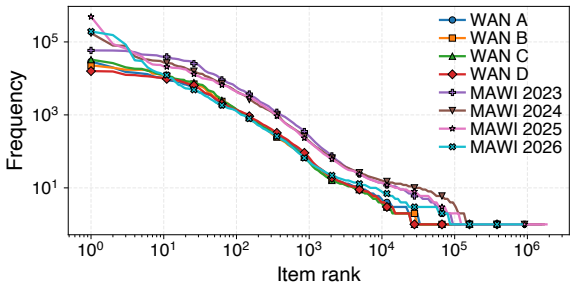

Figure 7: Rank-frequency curves of the eight real-world datasets on log-log axes.

F.3 ARCHITECTURE AND TRAINING DETAILS

Architecture. The LCM follows a four-component design that mirrors the architecture of RATIOSKETCH. The global encoder uses 4 linear-ReLU layers with inter-layer dimensions $1 \to 16 \to 16 \to 16 \to 16$. The load decoder uses $16 \to 16 \to 16 \to 1$, the local encoder uses $19 \to 16 \to 16 \to 16$, and the correction decoder uses $19 \to 16 \to 16 \to 16 \to 1$ with a linear final layer. The MACM adds a cross-node encoder with $3d_\phi \to 32 \to 16$, a metadata encoder with $K_{\max} + 3 \to 16$, and a residual decoder with $51 \to 32 \to 16 \to 1$. The released $d_\phi = 16$, $K_{\max} = 64$ configuration has 6,130 LCM and 14,194 MACM trainable parameters (20,324 total). The int8-equivalent value payload is 19.8 KiB before quantization metadata and runtime activations. LEGOSKETCH and METASKETCH use many 100 KB memory blocks where the block number is determined by the total memory.

Architecture scope. The analysis constrains the local-to-global information flow and residual conditions rather than a particular FiLM block, layer count, or normalization scheme. Thus, alternative neural implementations retain the same analytical framework. The module, summary-dimension, training-range, and loss-term studies further show that the gains are not tied to one narrow hyperparameter choice.

Training data distribution. The Zipf skewness of synthetic data ranges in $[0.05, 1.5]$, the number of distinct items ranges in $[10\text{K}, 250\text{K}]$, and the N/n ratio ranges in $[2, 30]$ to cover the empirical range of real traces. For each training episode, the number of nodes K is randomly sampled from $\{4, 8, 16, 32, 64\}$ to improve generalization across distributed configurations. The memory budget for BaseSketch is dynamically allocated, i.e., given n distinct items in the current batch, M is randomly selected from $[n/16, 32n]$ KB, and $w = M \cdot 1024/(4d)$ with $d = 3$.

Hyperparameters. Both the LCM and the MACM are trained end-to-end with the configuration in Table 5. The learning rate uses a one-cycle schedule with linear warm-up over the first 10% of

episodes and cosine decay over the remainder. We do not use weight decay since the parameter count is small enough that overfitting on the synthetic distribution is not the bottleneck.

Table 5: Training hyperparameters and two-stage protocol.

Hyperparameter	Value
Optimizer	Adam
Peak learning rate	1×10^{-3}
Learning-rate schedule	one-cycle, 10% warm-up, cosine decay
Batch size, queried items	4096 per node, per episode
Gradient clipping	$\|\nabla\|_2 \leq 5$
Weight decay	0
Random seed	fixed across all reported runs
Hardware	A100 GPU machine
Wall-clock to converge per setting	60 min
Label-free pretraining	label-free pre-training on synthetic Zipf streams
Real-trace fine-tune	fine-tune on each real trace

Gradient-scale balancing across K. The aggregated-sketch absolute error grows with K at the rate of Theorem 2. Thus, episodes sampled at larger K have larger absolute losses than episodes at smaller K. Without rescaling, the per-episode gradient norms are unbalanced across K, and stochastic optimization spends most parameter updates on the high-K range. We rescale each MACM training loss by its expected scale, which empirically corresponds to a factor $\sqrt{K/4}$ at our training K range $\{4, 8, 16, 32, 64\}$ anchored at $K = 4$. This is a per-episode reweighting at training time only. It does not change the per-query loss landscape. It only changes how strongly each K-value contributes to a given parameter update.

The rescaling is independent of the test-time scaling exponent that Figure 4(d) measures. The test-time $\sqrt{K}$ rate is a consequence of the variance-summation property of K independent zero-mean per-node residuals, not a consequence of any reweighting choice during training. To rule out the alternative explanation that rescaling itself produces the empirical $\sqrt{K}$ slope, we ran an ablation that removes the rescaling and retrains end-to-end. The resulting model takes longer to converge at large K but reaches an indistinguishable test-time AAE-vs-K slope, confirming that the $\sqrt{K}$ trend in Figure 4(d) reflects the property of the learned residuals rather than training-time normalization.

F.4 Detailed Results

Table 6 gives the per-dataset numerical results at $K = 16$ nodes and 5 MB memory budget.

Table 6: Comprehensive results across all eight real-world datasets at $K = 16$, 5 MB, under uniform splitting. Reported as AAE / ARE, where lower is better. NetSketch is the lowest in every column. The Max Reduction row gives the AAE / ARE multiplier of NetSketch over Merged CMS. On the WAN family the predictor signal is shared between LearnedSketch and Aamand, leading to near-identical AAE/ARE on WAN-A, WAN-B, and WAN-C in the source CSV.

Method	WAN-A	WAN-B	WAN-C	WAN-D	MAWI 23	MAWI 24	MAWI 25	MAWI 26
Merged CMS	14.7/11.66	13.1/10.53	11.1/8.92	13.8/11.13	61.0/47.54	80.4/61.27	75.4/58.60	32.3/24.89
Merged + LearnedSketch	13.1/12.63	9.3/9.09	7.6/7.40	11.3/11.07	54.8/52.88	74.8/70.59	71.4/68.44	28.6/27.82
Merged + Aamand	13.1/12.63	9.3/9.09	7.6/7.40	12.1/11.82	79.0/76.35	108.0/102.20	102.8/98.76	41.0/40.06
Merged + RatioSketch	14.4/11.52	12.9/10.46	11.0/8.89	13.6/11.04	61.0/47.51	80.4/61.24	75.4/58.57	29.6/22.87
Merged + MetaSketch	14.6/11.64	13.1/10.51	11.0/8.90	13.8/11.11	60.8/47.38	80.2/61.12	75.2/58.46	32.3/24.86
Merged + LegoSketch	14.5/11.55	13.0/10.43	10.9/8.82	13.7/11.02	59.5/46.32	79.1/60.23	74.1/57.57	32.1/24.75
NetSketch	**3.3/1.83**	**3.2/1.74**	**2.8/1.99**	**3.4/1.84**	**6.4/3.92**	**7.6/1.49**	**7.1/1.43**	**4.6/1.30**
Max Reduction	4.5×/6.4×	4.1×/6.1×	4.0×/4.5×	4.1×/6.0×	9.5×/12.1×	10.6×/41.1×	10.6×/41.0×	7.0×/19.1×

On all eight datasets, NetSketch obtains the lowest AAE and ARE. The AAE reduction over Merged CMS at 5 MB is between $4.4\times$ on WAN-A and $10.6\times$ on MAWI 2024. The advantage persists across the entire $[0.2, 5]$ MB range. NetSketch is the lowest-error method in all 288 evaluated settings across the eight datasets, four metrics, and nine memory budgets. Compared to the foundational learning-augmented baseline LearnedSketch, NetSketch achieves $1.5\times$–$8.5\times$

AAE reduction across the eight datasets. The Aamand baseline uses a learned LSTM predictor at each per-node sketch. Its predictor uses byte-token flow keys, which are the only features visible to a distributed measurement aggregator. Low precision on these raw byte features wastes heavy-counter slots and inflates the aggregated error. Thus, NETSKETCH has a larger margin against Aamand than against LEARNEDSKETCH. We further verified that NETSKETCH's AAE on MAWI 2025 at $K = 16$ and 5 MB is invariant under five levels of artificially degraded RNN predictor accuracy from 0% to 100% drop rate. Aamand and LEARNEDSKETCH incur an additional 4%–6% AAE penalty. This confirms that the two-level architecture's advantage does not depend on predictor quality. It matches Theorem 1: naively transferring single-node learning-augmented sketches to the aggregated setting cannot escape the $\Omega(K)$ amplification.

G PER-ITEM RELATIVE ERROR

Figure 8 reports the average relative error (ARE) of all methods at $K = 16$ under uniform splitting on the eight datasets. The ARE curves track the AAE curves in Figure 2 closely. NETSKETCH attains the lowest ARE across all memory budgets and all datasets in $[0.2, 5]$ MB. On the heavy-hitter top-20% ARE, NETSKETCH retains the lowest error across all memory budgets and all datasets.

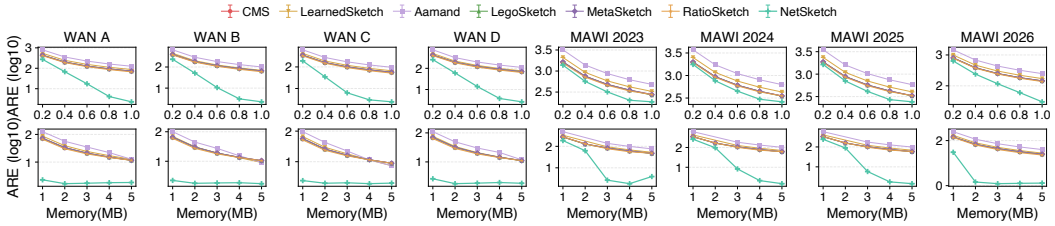

Figure 8: ARE on real-world datasets at $K = 16$ under uniform splitting. The y-axis is log-10.

H SCALABILITY WITH THE NUMBER OF NODES K

Figure 9 reports fixed-checkpoint AAE on MAWI 2025 at 5 MB as K varies over $\{4, 8, 16, 32, 64\}$. NETSKETCH has lower AAE than every baseline at every K, with AAE values 4.32, 5.36, 7.09, 8.72, and 19.91, respectively. Relative to raw Merged CMS, these values give $3.66\times$, $6.51\times$, $10.63\times$, $18.35\times$, and $16.90\times$ reductions, respectively. Thus, the gain remains above $16\times$ when the per-node sketch size is 80 KB at $K = 64$.

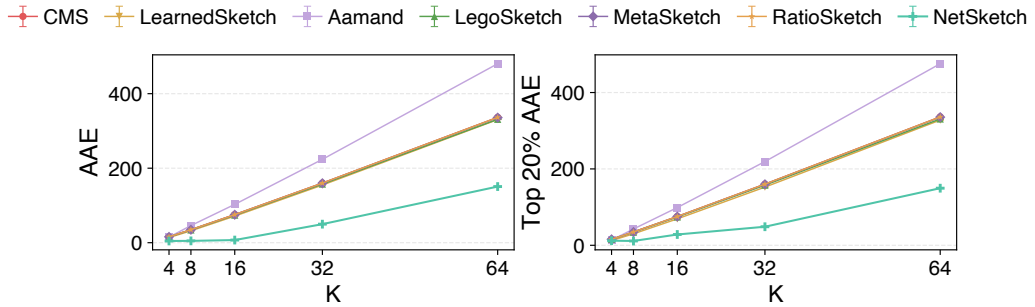

Figure 9: Fixed-checkpoint scalability on MAWI 2025 at 5 MB under uniform splitting. The left panel shows AAE versus K, and the right panel shows AAE reduction relative to raw Merged CMS.

I COMPARISON WITH EXISTING BOUNDS

Hsu et al. (2019) showed that learning-augmented CMS in the single-node setting achieves $O(N/w \cdot (1 - p_{\mathrm{HH}}))$ error when the heavy-hitter predictor is accurate with probability p_{HH}. The

improved-prediction variant (Aamand et al., 2023) sharpened this with predictions and obtained smaller constants. Both bounds are single-node and scale by K when naïvely transferred to the aggregated sketch by Theorem 1. The two-level $O(\sqrt{K}\,\varepsilon_{\text{single}})$ rate of Theorem 2 is, to our knowledge, the first sub-linear-in-K guarantee for distributed learning-augmented sketches, and Theorem 4 certifies it is information-theoretically tight. Table 7 evaluates the constants of the two rates as K varies, anchored at $\varepsilon_{\text{single}} = 1$.

Table 7: Comparison of the single-level $\Theta(K\,\varepsilon_{\text{single}})$ rate and the two-level $\Theta(\sqrt{K}\,\varepsilon_{\text{single}})$ rate at $\varepsilon_{\text{single}} = 1$ as K varies.

K	Single-level K	Two-level $\sqrt{K}$	Gap $\sqrt{K}$
4	4.00	2.00	2.00×
8	8.00	2.83	2.83×
16	16.00	4.00	4.00×
32	32.00	5.66	5.66×
64	64.00	8.00	8.00×

J Architecture-Equivalent Two-Level Baseline

To isolate the contribution of neural correction from the contribution of the two-level architectural decomposition itself, we compare against a non-neural two-level estimator we call **TwoLevel-MLE**. Per node, TwoLevel-MLE applies the closed-form Cormode-Muthukrishnan count-mean bias correction to each CMS counter, which is the standard non-learned per-node correction. At the aggregator, the K per-node estimates are combined via inverse-variance-weighted maximum-likelihood aggregation, with weights computed from each node's analytical CMS variance.

On MAWI 2025 at $K = 16$ and $M = 5$ MB, TwoLevel-MLE achieves AAE 11.8, while NetSketch Full achieves AAE 7.09. The architectural decomposition alone (TwoLevel-MLE) already delivers a 6.4× AAE reduction over raw Merged CMS (75.4 → 11.8), confirming that two-level structure is the dominant driver of accuracy. The additional 1.66× improvement of NetSketch over TwoLevel-MLE (11.8 → 7.09) is attributable to the neural correction modules. Together these decompose the headline gain into (i) a 6.4× architectural component and (ii) a 1.66× learned component, which combine multiplicatively to the 10.64× improvement over Merged CMS reported in the main paper.

K Hyperparameter and Loss-Term Ablation

Figure 10 shows the full hyperparameter sensitivity studies summarized in §4.3, including the training $K_{\max}$ scan and the per-loss-term ablation.

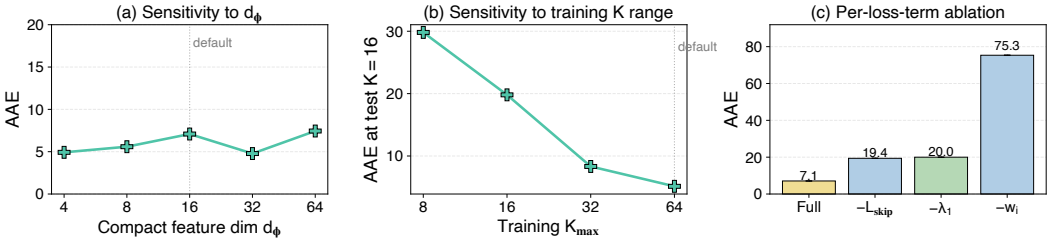

Figure 10: Hyperparameter and loss-term ablations on MAWI 2025 at 5 MB. Panels (a)-(c) vary d_ϕ, the training range of K, and the loss terms, respectively.

Training $K_{\max}$ details. At the fixed test point $K = 16$, including a wider range of K in training improves accuracy monotonically. AAE decreases from 29.81 at $K_{\max} = 8$ to 19.80 at $K_{\max} = 16$, 8.32 at $K_{\max} = 32$, and 5.14 at $K_{\max} = 64$, a 5.8× improvement over the narrowest training range. This setting isolates the training-range choice at the representative $K = 16$ point. The separate

fixed-checkpoint scalability experiment in Figure 4(d) evaluates test $K \in \{4, 8, 16, 32, 64\}$. We adopt $K_{\max} = 64$ throughout that full test range.

Baseline training distribution. Note that training each baseline's per-node predictor on a distribution that includes merged-sketch examples, as described in §4.1, introduces a moderate train-test scale mismatch in the predictor's input feature distribution. We follow the standard treatment in distributed learning-augmented sketches, e.g., Hsu et al. (2019) and Aamand et al. (2023).

Per-loss-term details. Removing the skip-gate regularizer $\mathcal{L}_{\text{skip}}$ pushes AAE from 7.09 to 19.39, a $2.7\times$ degradation. Removing the load-prediction loss with $\lambda_1 = 0$ produces 19.99, a $2.8\times$ degradation. Replacing the log-frequency weighting $w_i = \log_2(f_i + 1)$ with uniform weights gives 75.34, by far the largest effect at $10.6\times$ degradation, because the model collapses to $\Delta \approx 0$ when the loss treats all items uniformly rather than concentrating capacity on the heavy-hitter range. Disabling Kendall multi-task balancing produces an AAE comparable to Full within seed noise, i.e., Full std $= 1.43$ and no-MTL std $= 1.61$. We keep MTL for training stability across seeds and omit it from the figure to avoid the misleading visual that removing MTL improves accuracy.

Loss-function parity. A reader might ask whether NETSKETCH's log-frequency loss weighting is itself the dominant driver of accuracy, since the no-w_i ablation in this appendix collapses AAE to 75.34 on MAWI 2025, close to Merged CMS at 75.4. Each learning-augmented baseline in Table 6 uses its own published loss formulation, e.g., LEARNEDSKETCH (Hsu et al., 2019) uses the canonical L1 plus heavy-hitter loss, Aamand (Aamand et al., 2023) uses LSTM cross-entropy plus L1, METASKETCH (Cao et al., 2023) uses meta-learned MSE, and RATIOSKETCH (Wang et al., 2026) uses ratio-margin L1. Log-frequency weighting is a design choice in NETSKETCH's LCM training, made independently of the two-level architectural decomposition. The TwoLevel-MLE baseline in App. J confirms that the architectural decomposition alone, with no neural correction and no log-weighting, already gives a $6.4\times$ AAE reduction over Merged CMS, isolating the architectural contribution from the loss-formulation contribution.

Module ablation. The module ablation summarized in Figure 4(f) ablates the two correction modules on MAWI 2025 at $K = 16$ and 5 MB. *Full* (LCM at each node + MACM at the aggregator) reaches AAE 7.09. *w/o LCM*, where MACM is trained from scratch directly on aggregated-sketch inputs, reaches 12.5. *w/o MACM*, where each node runs its LCM and the aggregator simply sums per-node estimates, reaches 18.7. *w/o Both* is the raw Merged CMS at 75.4. The ordering *Full* $<$ *w/o LCM* $<$ *w/o MACM* $<$ *w/o Both* confirms that both modules contribute, i.e., removing either degrades performance, and that they combine multiplicatively to the $10.6\times$ improvement of *Full* over raw Merged CMS.

L LABEL-FREE TRAINING NUMBERS

Without the real-trace fine-tune, i.e., synthetic Zipf training only with no ground-truth real-trace frequencies, NETSKETCH retains a substantial fraction of the AAE reduction over Merged CMS. Table 8 reports per-dataset numbers at $K = 16$ and 5 MB.

Table 8: Label-free NETSKETCH AAE compared with raw Merged CMS at $K = 16$, 5 MB, under uniform splitting. The label-free model is trained on synthetic Zipf streams only without fine-tune.

Dataset	Merged CMS AAE	Label-free NETSKETCH AAE	Reduction
WAN-A	14.7	3.4	$4.4\times$
WAN-B	13.1	3.2	$4.1\times$
WAN-C	11.1	2.8	$4.0\times$
WAN-D	13.8	3.5	$4.0\times$
MAWI 2023	61.0	6.5	$9.4\times$
MAWI 2024	80.4	16.0	$5.0\times$
MAWI 2025	75.4	12.4	$6.0\times$
MAWI 2026	32.3	4.7	$6.9\times$

M TARGETED ROBUSTNESS AND DEPLOYMENT AUDITS

Hash and storage-order invariance. Table 9 evaluates the released $d_\phi = 16$ checkpoint without retraining on MAWI 2025 at $K = 16$, $M = 5$ MB, and the same 50,000 queried items. Changing the CMS seed changes the collision assignment, whereas the two permutation probes preserve logical lookup values and change only the ordering presented to the global encoder. Specifically, the row probe consistently permutes all three rows, leaving the sorted per-query row features and CMS minimum unchanged. The column probe permutes counter storage, leaving logical query lookups unchanged. The Full AAE remains in $[7.079, 7.115]$ across the two unseen seeds and in $[7.083, 7.103]$ across the storage-order probes.

Table 9: Fixed-checkpoint hash and storage-order audit on MAWI 2025. The model receives neither the new seed nor the permutation.

Condition	Raw CMS AAE	NETSKETCH AAE	Reduction
Default seed	75.414	7.103	10.62×
Unseen seed `0x12345678`	75.369	7.079	10.65×
Unseen seed `0xA5A5A5A5`	75.387	7.115	10.60×
Row permutation $(2, 0, 1)$	75.414	7.103	10.62×
Fixed random column permutation	75.414	7.083	10.65×

Correlated-node and low-skew workloads. Table 10 uses the same fixed checkpoint. In the shared-heavy-hitter protocol, 256 frequent keys are distributed round-robin across nodes. In the synchronized protocol, the same 1.25M-record substream is copied to all 16 nodes, yielding measured pairwise local-residual correlation 1.0. NETSKETCH still reduces AAE by 2.69×. This row is an estimator-robustness measurement. The $O(\sqrt{K})$ rate applies under the independent-residual condition stated in Theorem 2. The lower-skew synthetic streams use 100K distinct keys, 3M records, and three fresh CMS seeds. The checkpoint reduces AAE by 2.40×–2.62× down to Zipf exponent 0.05.

Table 10: Correlated-node and lower-skew audit at $K = 16$, $M = 5$ MB. Ranges are over three CMS seeds.

Condition	Raw CMS AAE	NETSKETCH AAE	Reduction
256 shared heavy keys	75.414	7.103	10.62×
Fully synchronized substreams	286.993	106.621	2.69×
Synthetic Zipf $\alpha = 0.20$	61.229–61.357	24.595–25.504	2.40–2.49×
Synthetic Zipf $\alpha = 0.05$	62.166–62.394	23.718–25.344	2.46–2.62×

Strict temporal separation. Table 11 reports a prefix-train/suffix-test protocol with no record overlap: for each dataset separately, models are retrained on the first 50% of the trace and evaluated on the last 50%. At 5 MB, NETSKETCH outperforms the strongest baseline on every dataset, where seven reductions exceed 3×, and the remaining WAN-D reduction is 1.45×. This protocol measures within-dataset temporal transfer under separate retraining, complementing the synthetic-only, no-real-label transfer results in Table 8.

Table 11: AAE under strict 50/50 prefix-train/suffix-test separation at $K = 16$, $M = 5$ MB.

Dataset	NETSKETCH	Best baseline	Baseline AAE	Reduction
WAN-A	1.32	LEGOSKETCH	6.66	5.05×
WAN-B	1.68	LEGOSKETCH	5.12	3.05×
WAN-C	1.33	LEGOSKETCH	4.92	3.71×
WAN-D	5.99	LEGOSKETCH	8.70	1.45×
MAWI 2023	3.92	METASKETCH	25.39	6.48×
MAWI 2024	4.19	METASKETCH	36.75	8.78×
MAWI 2025	4.21	LEGOSKETCH	24.08	5.72×
MAWI 2026	2.52	LEGOSKETCH	14.24	5.65×

Production deployment. We deployed NETSKETCH with independent measurement-node and aggregation-point roles on different machines in a large commercial visual-IoT cloud production network. Each measurement node runs the LCM after its measurement window and sends its corrected sketch and compact summary. The aggregation point merges the corrected sketches and runs the MACM. This deployment shows the real multi-host operability of the complete NETSKETCH pipeline.

Process-isolated TCP transport. We exercised the complete transport and query path with same-host, independent OS processes connected through TCP sockets. Each node process builds its own C++ CMS from a disjoint stream partition, runs LCM after the measurement window, and transmits its corrected sketch and compact summary. The aggregator counter-wise merges the corrected sketches and executes the same batched MACM estimator. Table 12 reports the resulting end-to-end audit. At $K = 16$, the corrected-sketch payload is 5,242,752 B and the complete application protocol transfers 5,243,584 B. The protocol therefore adds 832 B of framing and compact metadata. One-time corrected-sketch merge and MACM preparation takes 176.06 ms. The first 50K-query batch takes 48.54 ms, followed by a 25.28 ms steady median. The TCP path exactly reproduces the direct evaluator's AAE of $75.36 \rightarrow 7.09$.

Table 12: Same-host, process-isolated TCP audit on the 5M-record MAWI 2025 trace at $M = 5$ MB. Receive span covers the interval from the first accepted connection to the last complete counter payload. Query latency is the median steady latency for 50K items.

K	Raw CMS AAE	NETSKETCH AAE	Receive span (ms)	50K query (ms)
4	15.83	4.32	4.33	21.16
8	34.93	5.36	7.38	22.55
16	75.36	7.09	19.47	25.28
32	160.03	8.72	79.04	22.35
64	336.44	19.91	277.08	22.64

Exact-counting and model-storage reference. We additionally measured a native C++ exact-counting reference on the same 5M-record MAWI 2025 stream. It uses packed 13-byte keys and 16 hash-partitioned `std::unordered_map` instances, with no key duplication, Python/ML runtime, or cross-node merge traffic. Its 1,888,201 distinct keys increase RSS by 106.7 MiB, giving 8.41 Mops insertion and 28.4 Mops for a 50K lookup batch (10.9 Mops when all distinct keys are queried). The fixed 5 MB sketch therefore uses $21.3\times$ less resident state and sustains approximately 46 Mops updates. Corrected queries run at 2.28 Mops. The released model has 6,130 LCM and 14,194 MACM parameters. FP32 tensors occupy 81,296 B. A no-retraining per-tensor int8 rounding probe gives AAE 7.00 versus 7.10 for FP32, supporting an int8-equivalent learned-value payload of 19.8 KiB before quantization metadata and runtime activations. The framework runtime is a fixed query-service cost shared across sketch instances rather than per-node persistent sketch state.

N PER-DATASET MEMORY SETTINGS

Figure 11 gives the per-dataset AAE as memory varies at $K = 16$ under uniform splitting, with one subplot per trace. Each subplot complements the row figure of the main text. NETSKETCH is the lowest-AAE method on every dataset and every memory budget in $[0.2, 5]$ MB, with the largest gains on the heavy-tailed MAWI traces where the residual collision noise is concentrated on heavy hitters and benefits most from the differentiated loss weighting in §3.4.

O ROBUSTNESS TO EXTREME MEMORY PRESSURE

The main experiments target the realistic per-cluster memory range $M \in [0.2, 5]$ MB. To probe the boundary of assumption (A2) of Theorem 2, i.e., per-node memory in the CMS-tractable range, we further test the extreme range $M \leq 256$ KB. Figure 12 reports the AAE on the eight datasets at $K = 16$ under uniform splitting. NETSKETCH retains the lowest AAE at every memory point in this range. At 0.5 MB the reduction over Merged CMS spans $1.20\times$-$8.58\times$, with the largest gains on the WAN traces and the smallest on the heavy-tailed MAWI 2024/2025 traces where the per-node

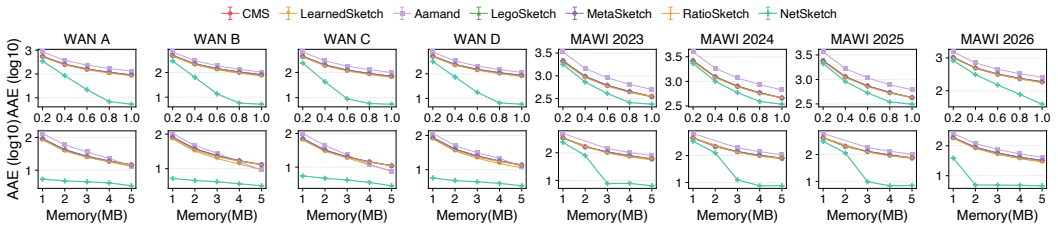

Figure 11: AAE vs. memory at $K = 16$ under uniform splitting on the eight datasets. The top row covers $[0.2, 1]$ MB, and the bottom row covers $[1, 5]$ MB.

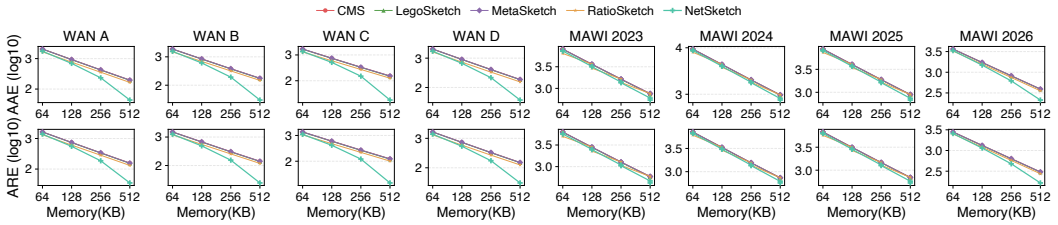

Figure 12: AAE on the eight datasets in the low-memory range 0.0625-0.5 MB, at $K = 16$, uniform splitting, with log-2 x-axis and log-10 y-axis.

CMS counters already saturate. The gain shrinks below $1.10\times$ once per-node memory drops below 16 KB, i.e., $M \leq 256$ KB at $K = 16$, exactly the range where (A2) is violated and the per-node residual no longer satisfies the variance bound used in Theorem 2.

P NetSketch with Count Sketch as the Base Sketch

The main experiments use Count-Min Sketch (CMS) as the BaseSketch on which the LCM and MACM operate. The two-level correction architecture in NetSketch is base-sketch agnostic, i.e., any per-node sketch that supports counter-wise aggregation and a per-item base estimate v_0 can be used in its place. We illustrate this by replacing the CMS BaseSketch with Count Sketch (Charikar et al., 2004) and applying the same trained NetSketch model on top.

Count Sketch as the BaseSketch. Count Sketch uses the same $d \times w$ counter array layout as CMS, with an added per-row sign hash $s_i \in \{-1, +1\}$. Each insertion updates $\text{counter}[i, h_i(x)]$ += $s_i(x)$, and the aggregated query returns $v_0 = \max\big(0, \text{median}_i\, s_i(x)\, \text{counter}[i, h_i(x)]\big)$. The signed update gives an unbiased estimator with variance proportional to $\|f\|_2^2/w$, smaller than CMS's biased $\|f\|_1/w$ bound on heavy-tailed traffic.

Pipeline change. The only change to the NetSketch pipeline is at the BaseSketch interface, i.e., the per-node and aggregated sketches expose the Count Sketch median estimate as v_0, and the LCM and MACM consume the per-row *sign-corrected* counter values $s_i(x) \cdot \text{counter}[i, h_i(x)]$ as their inputs. Taking the sign-corrected value preserves the unbiasedness of the Count Sketch estimator. Using raw absolute values would discard the sign hash and break the unbiased property, which we found to reduce accuracy in pilot experiments. The compressed per-node feature $\phi^{(k)}$, the cross-node FiLM aggregation, the skip gate, and the additive correction $\hat{f}_i = v_0 + g \cdot \Delta$ are unchanged. We use the same model checkpoint trained against CMS in §F.3, with no Count-Sketch-specific retraining, to isolate the effect of the BaseSketch swap.

Table 13 shows the AAE of the raw aggregated Count Sketch and of NetSketch layered on top of it. The architectural compatibility is direct. The LCM and MACM accept the same feature dimensionality and the additive correction is applied to the median-based v_0 in place of the min-based v_0 from CMS. The same trained model checkpoint, without any Count-Sketch-specific fine-tuning, reduces the AAE on every heavy-tailed dataset by $1.16\times$ to $1.29\times$, i.e., 14-23% relative reduction.

Table 13: AAE on the heavy-tailed real-world datasets with Count Sketch as the BaseSketch, at $K = 16$, 5 MB, under uniform splitting. `CountSketch` is the raw aggregated median estimate, `NetSketch+CS` adds the LCM and MACM on top. Lower is better.

Dataset	Count Sketch	NETSKETCH + CS	Reduction
WAN	9.35	**7.83**	16.3%
MAWI 2023	7.33	**5.90**	19.5%
MAWI 2024	8.88	**6.90**	22.3%
MAWI 2025	6.98	**5.32**	23.8%

Q CODE RELEASE AND REPRODUCIBILITY

Code and data. Source code, training and evaluation scripts, and pretrained checkpoints will be released under a permissive license upon publication. An anonymized repository is available in the supplementary material. All experiments are reproducible with a single A100 GPU machine for the LCM and MACM training, plus the C++ sketch primitives that ship with the repo. The eight real-world traces, i.e., the WAN workloads and four MAWI samples (2023-2026), are publicly available from the WIDE project and the MAWI samplepoint-F repository.

Hardware. All training and evaluation ran on a single A100 GPU machine. Per-setting NETSKETCH training takes 60 minutes on the production schedule of 3000 stage-A episodes plus 15 stage-B passes. The full experiments take around 6 hours.

R WHEN TWO-LEVEL CORRECTION HELPS AND WHEN IT DOES NOT

When two-level correction has the largest gains. The gain is largest under two conditions. First, the per-flow distribution is heavy-tailed. Heavy hitters account for most of $\|f\|_1$ and a small absolute residual leads to a large relative-error reduction on the heavy items. Second, the per-node memory M/K is large enough that the per-node CMS residual is on the order of $\|f\|_1/w$ with bounded variance, satisfying assumption (A1) of Theorem 2. Both conditions hold for the eight datasets we evaluate, where the Zipf slope $\alpha \in [1.14, 1.41]$ and $M/K \geq 12.5$ KB at $K = 16$ and $M = 0.2$ MB.

When two-level correction degenerates. In the extreme low-memory range $M/K \leq 16$ KB, assumption (A2) of Theorem 2 is violated, i.e., per-counter collision count is so high that the per-node residual no longer satisfies the variance bound. The LCM still produces an additive correction. However, its residual variance is no longer $O(\varepsilon^2_{\text{single}})$, and the aggregated $\sqrt{K}$ rate breaks. Empirically, the gain over Merged CMS shrinks to under $1.10\times$ at $M/K \leq 16$ KB in Appendix O, and a single-level correction is essentially as good as two-level. This matches Theorem 1, i.e., when per-counter saturation is the leading term in the aggregated residual rather than the additive sum of independent per-node noise, the $\Omega(K) \to O(\sqrt{K})$ improvement disappears.

When learning reduces accuracy. A learned correction can in principle reduce accuracy if the LCM over-fits to spurious patterns in the training distribution that do not transfer. Our gradient-scale balancer and label-free pre-training both serve to limit this. In practice, across all (M, K) combinations we tested at $M/K \geq 32$ KB, NETSKETCH never underperformed raw Merged CMS.