Space-Optimal Streaming Algorithms via Efficient Encodings

Liudeng Wang
Texas A&M University
eureka@tamu.edu

David P. Woodruff
Carnegie Mellon University and Google Research
dwoodruf@cs.cmu.edu

Shenghao Xie
Carnegie Mellon University
shenghax@cs.cmu.edu

Samson Zhou
Texas A&M University
samsonzhou@gmail.com

Abstract

We study the problem of constructing coresets on data streams: Given a dataset of n points in $\mathbb{R}^d$ arriving sequentially, the goal is to maintain a small weighted subset that accurately approximates the objective values of the entire dataset for a given class of query functions. Previous methods like merge-and-reduce and online sensitivity sampling required more space than the size of an optimal such subset in the offline setting. In this work, we show that such overheads are not necessary by presenting space-optimal streaming algorithms for central high-dimensional optimization problems such as Euclidean (k, z)-clustering and L_p subspace embeddings, as well as applications to projection-cost preserving sketches.

For (k, z)-clustering, our streaming algorithm uses memory independent of the number n of input points (in words of memory) and the aspect ratio Δ, yielding a coreset with an optimal $\tilde{\mathcal{O}}\left(\frac{dk}{\min(\varepsilon^4, \varepsilon^{z+2})}\right)$ words of memory for accuracy parameter $\varepsilon \in (0, 1)$. For L_p subspace embeddings, where the rows of matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$ arrive in an insertion-only stream, we achieve an efficient algorithm that uses optimal space for different ranges of p: $\tilde{\mathcal{O}}\left(\frac{d^2}{\varepsilon^2}\right)$ words for $p \leq 2$ and $\tilde{\mathcal{O}}\left(\frac{d^{p/2+1}}{\varepsilon^2}\right)$ words for $p > 2$, both independent of the number of rows n. Thus, our work shows that streaming algorithms can match offline algorithms in space complexity.

1 Introduction

A core challenge in large-scale machine learning systems is processing datasets that far exceed standard computer memory capacities. To solve this issue, the *streaming model* has emerged as a fundamental paradigm, in which elements of a dataset arrive sequentially in a data stream and an algorithm must process each arriving element immediately and irrevocably as it cannot revisit discarded data as the stream moves on. The primary goal is to design algorithms that can efficiently approximate, compute, or detect properties of the underlying dataset with minimal memory. Thus, streaming algorithms would ideally excel at summarizing large-scale, evolving datasets under strict memory constraints. Despite significant progress, it remains open whether such fundamental tasks inherently require more space in the streaming model compared to their offline counterparts.

Coresets. One of the most common data summarization approaches in streaming algorithms is to collect a *coreset* [HM04, FL11], which is a small, weighted subset that approximately preserves a target cost function. Formally, given a set of queries $\mathcal{Q}$ induced by a streaming problem P, a dataset X of n elements, and a cost function $\mathrm{cost}(\cdot, \cdot)$ defined by the problem context, a (strong)

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

$(1+\varepsilon)$-coreset for (X, P) is a weighted subset Y of X such that for all queries $q \in \mathcal{Q}$, $\mathrm{cost}(q, Y) \in (1 \pm \varepsilon) \cdot \mathrm{cost}(q, X)$. The appeal of coresets lies in their ability to provide nearly identical cost values with reduced space complexity, enabling streaming algorithms to compute accurate approximations under memory constraints.

Perhaps the most important application of coresets is to Euclidean (k, z)-clustering, in which given a dataset $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, a metric $\mathrm{dist}(\cdot, \cdot)$, a cluster parameter $k > 0$, and a positive integer exponent $z > 0$, and the goal is to minimize the clustering cost across all sets $\mathcal{C}$ of at most k centers:

$$\min_{\mathcal{C} \subset \mathbb{R}^d, |\mathcal{C}| \le k} \mathrm{cost}(X, \mathcal{C}) := \min_{\mathcal{C} \subset \mathbb{R}^d, |\mathcal{C}| \le k} \sum_{i=1}^{n} \min_{c \in \mathcal{C}} \mathrm{dist}(x_i, c)^z.$$

In the streaming model, the points in X arrive sequentially and a coreset Z_t at time t satisfies $\mathrm{cost}(X_t, \mathcal{C}) \in (1 \pm \varepsilon) \cdot \mathrm{cost}(Z_t, \mathcal{C})$. We can then perform an efficient (k, z)-clustering algorithm on each coreset Z_t to identify a near-optimal set of cluster centers, instead of operating on the full dataset arrived at time t. Naturally, smaller coresets correspond to more space-efficient algorithms. Another core usage of coresets is to L_p subspace embeddings [Woo14, CP15], which compact a large-scale matrix $\mathbf{A}$ while preserving all vector norms under projections, i.e., $\|\mathbf{Mx}\|_p \approx \|\mathbf{Ax}\|_p$ for all $\mathbf{x}$, enabling efficient dimensionality reduction and accelerating down-stream tasks like least-squares regression.

When constructed offline, many coreset schemes are known to require only $\mathrm{poly}_{\mathcal{Q}}\left(d, \frac{1}{\varepsilon}\right)$ samples [Sar06, CP15, CLSS22]. As is often standard in the streaming literature, we allow each word of space to use $\Theta(\log(nd))$ bits of storage, so that we can store the weight and each coordinate of each point using $\mathcal{O}(1)$ words of space, and thus their space complexities depend only on structural parameters but not on n. However, constructing and maintaining such coresets in the streaming model is far more challenging, since we do not have access to the entire dataset and require immediate decisions. Existing approaches, merge-and-reduce [BS80, HM04] and online sampling [CWZ23, CMP20, WY23], often incur logarithmic overhead in n in space, which hinders their ability to handle large-scale datasets and puts forward the question of whether there is a separation in space between the offline and the streaming settings. Thus it is natural to ask:

Can we construct streaming algorithms for data summarization problems that achieve space usage (in words) independent of input size n?

1.1 Our Contributions

In this paper, we resolve the above question by providing a framework that produces space-efficient streaming algorithms via so called "efficient encodings". Specifically, our framework efficiently encodes the coresets instead of storing them with high precision, reducing the memory consumption while enabling efficient reconstruction for future usage. We then apply the framework to provide algorithms for (k, z)-clustering, L_p subspace embeddings, and projection-cost preserving sketches with space independent of the input size n, without losing additional factors that depend on the structural properties, e.g., those related to $\mathcal{Q}$, d, or ε. Therefore, our results match or improve on the space complexity of previous streaming algorithms in *all* factors and match the *offline* setting in the words of space. In essence, our framework demonstrates that streaming overhead is *not* an inherent inevitability for data summarization problems, as long as they are amenable to an efficient encoding.

(k, z)-clustering. As mentioned earlier, (k, z)-clustering minimizes the clustering cost across all sets of at most k centers, where the special cases $z = 1$ and $z = 2$ recover the classical k-median and k-means problems, respectively. Intuitively, clustering partitions a dataset into groups so that points within the same group are similar, while points in different groups are dissimilar. Since its early study in the 1950s [S$^+$56, Mac67], clustering has become a core unsupervised learning task in many domains, including bioinformatics, combinatorial optimization, computational geometry, computer graphics, data analysis, and machine learning.

In the (insertion-only) streaming model, the points $x_1, \ldots, x_n$ of X arrive sequentially and for each time $t \in [n]$, the goal is to find a clustering $\mathcal{C}_t$ on the prefix subset $X_t := \{x_1, \ldots, x_t\}$, inducing a cost that is within a $(1+\varepsilon)$-fraction of the optimal clustering cost on X_t, for an accuracy parameter $\varepsilon \in (0, 1)$. We remark that due to the space constraints of the streaming model, we cannot represent input points in arbitrary precision and thus we generally assume the points in X to lie within the

grid $\{1, \ldots, \Delta\}^d$. Utilizing our efficient encoding framework, our first main result provides space-optimal coresets construction algorithms for (k, z)-clustering. In the following, we use the notation $\tilde{\mathcal{O}}(f) := \text{polylog}(f)$.

Theorem 1.1 (Informal version of Theorem D.11)**.** *Given a set X of n points on $[\Delta]^d$ and an accuracy parameter $\varepsilon \in (0, 1)$, there is a one-pass insertion-only streaming algorithm that uses $\tilde{\mathcal{O}}\left(\frac{dk}{\min(\varepsilon^4, \varepsilon^{2+z})}\right)$ words of space and outputs a $(1+\varepsilon)$-coreset of X.*

Theorem 1.1 is the first result for (k, z)-clustering coreset construction on insertion-only streams using $\mathcal{O}_{k,d,\varepsilon}(1)$ words of space, i.e., space usage independent of n. Despite a long line of work on clustering in the streaming model [HM04, HK07, Che09, FL11, BFLR19, BBC+19, WZZ23], the majority of previous results achieved space overheads that were polylogarithmic in n, compared to the offline setting. In a recent breakthrough work, [CWZ23] gave a streaming algorithm for (k, z)-clustering that uses $o_{k,d,\varepsilon}(\log(n\Delta))$ words of space, but their result still fails to have full independence of n. On the contrary, Theorem 1.1 matches the best known *offline* coreset constructions [CSS21, CLSS22] in terms of dependencies on d, k, and $\frac{1}{\varepsilon}$ in the space, and thus matches or improves upon the space usage of all previous streaming algorithms for insertion-only streams across all factors. Notably, our approach in Theorem 1.1 supports an efficient encoding for any coreset construction. For instance, if $k < \frac{1}{\varepsilon^2}$, it may be desirable to use existing coreset constructions of size $\tilde{\mathcal{O}}\left(\frac{k^2}{\varepsilon^2}\right)$ [CLSS22] rather than $\tilde{\mathcal{O}}\left(\frac{k}{\varepsilon^{z+2}}\right)$; our algorithm transitions to this case smoothly. In fact, our algorithm actually uses $\tilde{\mathcal{O}}(dk \log(n\Delta)) + \frac{dk}{\min(\varepsilon^4, \varepsilon^{2+z})} \cdot \text{polylog}\left(k, \frac{1}{\varepsilon}, \log(n\Delta)\right)$ bits of space, i.e., the $\frac{1}{\varepsilon}$ factors do not multiply the $\log(n\Delta)$ factors. Therefore, our results are near-optimal with a recent lower bound on the communication complexity of clustering [ZTHH24], which proved an $\Omega(dk/\varepsilon^2)$-bit lower bound for arbitrary clustering sketches under appropriate parameter regimes. See Figure 1 for a summary of the space complexity of recent results in the streaming model.

Streaming algorithm	Words of Memory
Online sensitivity sampling [CWZ23]	$\tilde{\mathcal{O}}\left(\frac{d^2k^2}{\varepsilon^2}\log^2 n\right)$
Merge-and-reduce with coreset of [CLSS22]	$\tilde{\mathcal{O}}\left(\frac{dk}{\min(\varepsilon^4, \varepsilon^{2+z})}\log^4 n\right)$
[CWZ23]	$\tilde{\mathcal{O}}\left(\frac{dk}{\min(\varepsilon^4, \varepsilon^{2+z})}\right) \cdot \text{polylog}(\log n)$
Theorem 1.1 (this work)	$\tilde{\mathcal{O}}\left(\frac{dk}{\min(\varepsilon^4, \varepsilon^{2+z})}\right)$

Fig. 1: Table of state-of-the-art (k, z)-clustering algorithms on insertion-only streams. We have $z = \mathcal{O}(1)$, $\Delta = \text{poly}(n)$, and the assumption that $k > \frac{1}{\varepsilon^z}$ for the purpose of presentation.

Subspace embeddings. Given a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ and an accuracy parameter $\varepsilon \in (0, 1)$, L_p subspace embeddings seek for a matrix $\mathbf{M} \in \mathbb{R}^{o(n) \times d}$ with fewer rows that preserves the projection of $\mathbf{A}$ on every vector $\mathbf{x} \in \mathbb{R}^d$ with respect to the ℓ_p norm, i.e., $\|\mathbf{Mx}\|_p \in (1 \pm \varepsilon)\|\mathbf{Ax}\|_p$. Such an embedding reduces the dimension of $\mathbf{A}$ while enabling nearly identical linear computations, including regression, low-rank approximation, and principal component analysis. Since their development in randomized linear algebra [FKV04, Woo14], subspace embeddings have become a standard tool for large-scale data processing, supporting efficient algorithms in optimization, statistics, and machine learning. We remark that the choice of the norm parameter p determines which geometric structure is preserved. For instance, $p = 2$ preserves Euclidean geometry, $p = 1$ yields greater resilience to anomalous rows, and larger values of p interpolate smoothly toward the behavior of ℓ_∞ norm.

In insertion-only streams, the rows of $\mathbf{A}$ are given one by one in the stream. At each time step $t \in [n]$, the algorithm must maintain a correct $(1+\varepsilon)$-embedding for the prefix matrix $\mathbf{A}_t$ consisting of the first t rows $\mathbf{a}_1, \ldots, \mathbf{a}_t$. Our second result gives the construction of ℓ_p subspace embeddings:

Theorem 1.2 (Informal version of Theorem E.15)**.** *Given a matrix $\mathbf{A}$ of n rows on $[-M, \ldots, -1, 0, 1, \ldots, M]^d$, with $M = \text{poly}(n)$, there exists a one-pass streaming algorithm in the row arrival model that outputs a $(1+\varepsilon)$-coreset. The algorithm uses $\tilde{\mathcal{O}}\left(\frac{d^2}{\varepsilon^2}\right)$ words of space for $p \in [1, 2]$ and $\tilde{\mathcal{O}}\left(\frac{d^{p/2+1}}{\varepsilon^2}\right)$ words of space for $p > 2$.*

We note that existing algorithms based on using merge-and-reduce with the optimal coreset constructions for L_p subspace embeddings require $\mathrm{poly}(d, \log n)$ bits of space, e.g., $\tilde{\mathcal{O}}\left(\frac{d^2}{\varepsilon^2}\right) \cdot \mathrm{polylog}(n)$ for $p \leq 2$ [DMM06a, DMM06b, Sar06] and $\tilde{\mathcal{O}}\left(\frac{d^{p/2+1}}{\varepsilon^2}\right) \cdot \mathrm{polylog}(n)$ for $p > 2$ [CP15, WY23]. Similarly, schemes based on online sampling also require $\mathrm{poly}(d, \log n)$ bits of space, e.g., $\tilde{\mathcal{O}}\left(\frac{d^2}{\varepsilon^2}\right) \cdot \mathrm{polylog}(n)$ for $p \leq 2$ [CMP20] and $\tilde{\mathcal{O}}\left(\frac{d^{p/2+1}}{\varepsilon^2}\right) \cdot \mathrm{polylog}(n)$ for $p > 2$ [CP15, BDM+20, WY23]. Hence, it was unknown whether there is an inherent separation between the offline setting and the streaming model for coreset constructions for L_p subspace embeddings. Theorem 1.2 resolves this question, showing that there is no overhead for the streaming model, i.e., we can perform L_p subspace embedding in $\mathcal{O}_{d,\varepsilon}(1)$ words of space. We next remark on the differing behaviors across the ranges of p. This is not a coincidence, as [LWW21] showed that any constant-factor L_p subspace embedding for $p \geq 2$ requires $\Omega(d^{\max(p/2,1)} + 1)$ bits. Thus, our result has tight dependencies in terms of the dimension d. See Figure 2 and Theorem E.15 for more details.

Streaming algorithm	Words of Memory
Merge-and-reduce with coreset of [DMM06a, DMM06b, Sar06], $p \leq 2$	$\tilde{\mathcal{O}}\left(\frac{d^2}{\varepsilon^2}\right) \cdot \mathrm{polylog}(n)$
Online leverage score/Lewis weight sampling [CMP20, WY23], $p \leq 2$	$\tilde{\mathcal{O}}\left(\frac{d^2}{\varepsilon^2} \log n\right)$
Theorem 1.2 (this work), $p \leq 2$	$\tilde{\mathcal{O}}\left(\frac{d^2}{\varepsilon^2}\right)$
Merge-and-reduce with coreset of [DDH+09], $p \geq 1$	$\tilde{\mathcal{O}}\left(\frac{d^{\max(p/2+1,p)+2}}{\varepsilon^2} \log^3 n\right)$
Merge-and-reduce with coreset of [CP15], $p \geq 2$	$\tilde{\mathcal{O}}\left(\frac{d^{p/2+1}}{\mathrm{poly}(\varepsilon)}\right) \cdot \mathrm{polylog}(n)$
Online Lewis weight sampling [WY23], $p > 2$	$\tilde{\mathcal{O}}\left(\frac{d^{p/2+1}}{\varepsilon^2}\right) \cdot \mathrm{polylog}(n)$
Theorem 1.2 (this work), $p > 2$	$\tilde{\mathcal{O}}\left(\frac{d^{p/2+1}}{\varepsilon^2}\right)$

Fig. 2: Table of L_p subspace embedding algorithms on insertion-only streams. We summarize existing results with $\kappa = \mathrm{poly}(n)$ for the purpose of presentation.

Projection-cost preserving sketch. Finally, we apply our efficient encoding framework to obtain projection-cost preserving sketches. Given a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ and an accuracy parameter $\varepsilon \in (0, 1)$, the goal is to construct a matrix $\mathbf{B} \in \mathbb{R}^{o(n) \times d}$ of fewer rows such that for all rank-k orthogonal projection matrices $\mathbf{P} \in \mathbb{R}^{d \times d}$, $\|\mathbf{B} - \mathbf{BP}\|_F^2 \in (1 \pm \varepsilon)\|\mathbf{A} - \mathbf{AP}\|_F^2$, where $\|\cdot\|_F$ denotes the Frobenius norm. Introduced by [CEM+15], projection-cost preserving sketches relax the requirements in subspace embeddings and instead preserve the residual cost under low-rank projections. Importantly, these sketches can be used to compute the low-rank approximation of $\mathbf{A}$, which is essential in random matrix algorithms and dimensionality reductions. We next present our result:

Theorem 1.3 (Informal version of Theorem F.14)**.** *Given a matrix $\mathbf{A}$ of n rows on $[-M, \ldots, -1, 0, 1, \ldots, M]^d$ with $M = \mathrm{poly}(n)$, there exists a one-pass streaming algorithm in the row arrival model that uses $\frac{k^2}{\varepsilon^2} \cdot \mathrm{polylog}\left(\frac{1}{\varepsilon}, \log(nd\kappa)\right) + \tilde{\mathcal{O}}\left(\frac{dk}{\varepsilon^2}\right)$ words of space and outputs a $(1+\varepsilon)$ rank-k projection-cost preservating sketch of $\mathbf{A}$.*

We note that [BDM+20] obtains a sketch with $\frac{dk}{\varepsilon^2} \log n \log^2 \kappa$ words of space in the online setting, which naturally generalizes a streaming algorithm. This complexity can be further reduced to $\frac{dk}{\varepsilon^2} \mathrm{polylog}\left(\frac{1}{\varepsilon}, \log(nd\kappa)\right)$ words of space by passing to a merge-and-reduce structure with the optimal (offline) coreset construction in [CMM17] (see e.g., the framework in [CWZ23]). On the contrary, our streaming algorithm in Theorem 1.3 achieves $\frac{k^2}{\varepsilon^2} \cdot \mathrm{polylog}\left(\frac{1}{\varepsilon}, \log(nd\kappa)\right) + \tilde{\mathcal{O}}\left(\frac{dk}{\varepsilon^2}\right)$ words of space. By comparison, our result avoids d multiplying the factors involving n, significantly improving the space complexity in previous algorithms. Furthermore, assuming $d \gg k\,\mathrm{poly}(\log \log n)$, e.g., when the projection matrix has high dimension but low rank. the space complexity (in words) is entirely independent of n, matching the best known *offline* coreset construction [CMM17]. In addition, for $\varepsilon = \Omega(k^{-1/2})$, PCP sketches formed from rescaled input rows require $\Omega(k/\varepsilon^2)$ rows in the worst

case, as follows from the spectral-sparsification lower bound of [BSS12]. Storing such a sketch explicitly as a dense matrix uses $\Omega(dk/\varepsilon^2)$ entries. When the dk/ε^2 term dominates our space bound, our space in words nearly-matches this offline lower bound.

Technical novelties. Our work introduces several technical innovations that may be of independent interest. Recall that two standard approaches to streaming coresets are online sensitivity sampling and merge-and-reduce. Online sensitivity sampling [CMP20, CWZ23, WY23] samples and reweights each arriving point according to its importance relative to the current prefix, but can retain substantially more points than offline sampling. Merge-and-reduce [BS80, HM04] constructs a binary tree on the data stream, stores a block of points in each node, and merges two nodes via offline coreset construction methods. Since approximation errors accumulate over $O(\log n)$ levels, each reduction requires accuracy $O(\varepsilon/\log n)$. For the problems considered here, both approaches incur additional $\mathrm{polylog}(n)$ factors in space compared to offline constructions (see more detail in Appendix A).

The streaming framework in [CWZ23] filters the input stream by online sensitivity sampling and passes the resulting stream to a merge-and-reduce structure, which reduces the dependency on n to poly-iterative logarithmic factors, i.e., $\mathrm{poly}(\log\log(n))$. However, this gap remains unavoidable if we use high precision encoding to store the coreset maintained by each active node in the merge-and-reduce structure. Instead, we efficiently encode each coreset by rounding the high-accuracy approximation to a constant-factor approximation using $\mathcal{O}_{\mathcal{Q},d,\varepsilon}(1)$ words of space. Unfortunately, separately encoding each coreset does not match our ultimate goal of getting rid of n, since we need to store the constant-factor approximation with high precision and there are $\mathrm{poly}(\log\log(n))$ coresets in the merge-and-reduce structure. To solve this issue, we store a global constant-factor approximation of the arrived dataset and round each coreset to this sketch, using only $\mathcal{O}_{\mathcal{Q},d,\varepsilon}(1)$ words in total. Note that the reconstructed coresets are no longer valid for the local dataset on each node in the merge-and-reduce tree, but together they give a $(1+\varepsilon)$-coreset of the entire dataset, which is sufficient for our purpose.

Despite our general framework, designing an efficient encoding remains highly nontrivial and fundamentally problem-dependent. For (k,z)-clustering, suppose we have a set C' of $\mathcal{O}(k)$ centers that is a constant-factor approximation to the optimal clustering on X. We can then write each point as $x = \pi_{C'}(x) + (x - \pi_{C'}(x))$, where $\pi_{C'}(x)$ is the closest center of C' to x, and round each coordinate of the offset $x - \pi_{C'}(x)$ to a power of $(1+\varepsilon')$ for $\varepsilon' = \mathrm{poly}\left(\varepsilon, \frac{1}{d}, \frac{1}{\log(n\Delta)}\right)$. Let y' be the rounded offset and set $x' = \pi_{C'}(x) + y'$. Then the encoded points x' form a $(1+\varepsilon)$-coreset for X using only $\mathcal{O}\left(\log k + d\log\frac{1}{\varepsilon'}\right)$ bits.

For L_p subspace embeddings, directly quantizing projected rows of $\mathbf{A}$ on a constant-factor approximation $\mathbf{M}$ can introduce large errors. Our key insight is to first precondition the stream using a well-conditioned basis derived from $\mathbf{M}$, so that no individual row contributes too much to the embedding error. This makes the rounding error proportional to row sensitivities, which is in total at most $\varepsilon'\,\mathrm{poly}(d)$, thus the overall error is bounded.

For projection-cost preserving sketches, we instead round the rows to the optimal rank-k approximation $\mathbf{A}_k$ and write the matrix as $\mathbf{A} = \mathbf{A}_k + (\mathbf{A} - \mathbf{A}_k)$. The intuition is that projection-cost preservation depends primarily on accurately capturing the dominant k-dimensional structure of the data, while the residual component contributes only to the bounded tail error. We therefore store the head component $\mathbf{A}_k$ with high precision and quantize only the residual $\mathbf{A} - \mathbf{A}_k$. We observe that $\mathbf{A}_k$ is a projection of the rows of $\mathbf{A}$ onto the span of the top-k right singular vectors. Thus for global encoding, we first obtain a constant-factor projection-cost preserving sketch $\mathbf{B}$ of the entire stream, and then project the sketch matrix on each node to the top-k right singular vectors of $\mathbf{B}$.

1.2 Preliminaries

For a positive integer n, we write $[n] = \{1, 2, \ldots, n\}$. We use $\mathrm{poly}(n)$ to denote an arbitrary fixed polynomial in n, and $\mathrm{polylog}(n)$ to denote a polynomial in $\log n$. We use the notation $\tilde{\mathcal{O}}(f) := \mathrm{polylog}(f)$. An event is said to hold with high probability if it occurs with probability at least $1 - \frac{1}{\mathrm{poly}(n)}$. Throughout the paper, we restrict our attention to Euclidean spaces. For points $x, y \in \mathbb{R}^d$, we write $\mathrm{dist}(x,y) = \|x-y\|_2$. For a point x and a subset $S \subset \mathbb{R}^d$, we further define $\mathrm{dist}(x,S) = \min_{s\in S}\mathrm{dist}(x,s)$. For $z \geq 1$, the ℓ_z norm of a vector x is denoted by $\|x\|_z$, where $\|x\|_z^z = \sum_{i=1}^d |x_i|^z$. Given a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$, its Frobenius norm is $\|\mathbf{A}\|_F^2 = \sum_{i\in[n]}\sum_{j\in[d]} A_{i,j}^2$.

For a fixed $z \geq 1$ and sets $X, C \subset \mathbb{R}^d$ with $X = \{x_1, \ldots, x_n\}$, we define the clustering objective $\mathrm{cost}(X, C) = \sum_{i=1}^{n} \mathrm{dist}(x_i, C)^z$.

2 Frameworks for Space-Optimal Streaming Algorithms

In this section, we formally describe our efficient encoding framework. Suppose that we have the following subroutines, each with failure probability at most δ:

- An online sampling scheme that returns a $(1+\varepsilon)$-coreset of size $\mathcal{S}_{\text{ol}}(\varepsilon, \delta)$.
- An algorithm that merges two coresets to obtain a $(1+\varepsilon)$-coreset of size $\mathcal{S}_{\text{off}}(\varepsilon, \delta)$.
- A global efficient encoding scheme that encodes a stream with m partitions $\mathcal{K} = \mathcal{K}_1 \circ \ldots \circ \mathcal{K}_m$, where each part $\mathcal{K}_i$ has size at most $\mathcal{S}_{\text{off}}(\varepsilon, \delta)$, using $\mathcal{S}_{\text{gl-encode}}(\varepsilon, \delta) + m \cdot \mathcal{S}_{\text{loc-encode}}(\varepsilon, \delta)$ bits of memory, where $\mathcal{S}_{\text{gl-encode}}(\varepsilon, \delta)$ denotes the space to store the global constant-factor approximation and $\mathcal{S}_{\text{loc-encode}}(\varepsilon, \delta)$ denotes the space to store the offset of each active coreset.

Note that we do not show the dependency of n and d in the above space complexities for simplicity of presentation. We then assemble the above subroutines. Given a stream $\mathcal{S}$ of length n, we use the online sampling scheme to generate a stream $\mathcal{S}'$ and pass it on to a merge-and-reduce tree to produce a coreset $\mathcal{C}$. We store the active nodes in the tree using global efficient encoding. Upon each new arrival, we reconstruct the coreset from the global encoding, which defines the online sampling probability. We visualize this process in Figure 3 and present the formal algorithm in Algorithm 4.

(1) Given a stream $\mathcal{S}$ of length n, use an online sampling scheme to generate a stream $\mathcal{S}'$.
(2) Run merge-and-reduce on $\mathcal{S}'$ to produce a coreset $\mathcal{C}$.
(3) Given a constant-factor approximation $\mathcal{C}'$ of $\mathcal{S}$, round the active nodes in the merge-and-reduce data structure to $\mathcal{C}'$ with global efficient encoding.
(4) Whenever a new item arrives in the stream, reconstruct the coreset from the efficient encoding to compute the online sampling probability in Step (1).

Fig. 3: High-level summary of our approach

Note that there are at most $\ell = \frac{\mathcal{S}_{\text{ol}}(\varepsilon,\delta)}{\mathcal{S}_{\text{off}}(\varepsilon,\delta)}$ leaf nodes in the merge-and-reduce tree, so the number of active nodes is bounded by the height $h = \mathcal{O}(\log \ell)$. Since each level in the merge-and-reduce tree at most incurs an additional $(1+\varepsilon)$-factor in the multiplicative error, it suffices to take $\varepsilon' = \mathcal{O}\left(\frac{\varepsilon}{h}\right)$ and $\delta' = \mathcal{O}\left(\frac{\delta}{\ell}\right)$ when we construct each coreset. Therefore, the global encoding scheme encodes h coresets and uses $\mathcal{S}_{\text{gl-encode}}\left(\frac{\varepsilon}{h}, \frac{\delta}{\ell}\right) + h \cdot \mathcal{S}_{\text{loc-encode}}\left(\frac{\varepsilon}{h}, \frac{\delta}{\ell}\right)$ bits of space in total (see the formal statement in Framework C.3). We remark that in the problems we study, the sample complexities of state-of-the-art online algorithms and their offline counterparts are off by a factor of $\mathrm{polylog}_{\mathcal{Q},d,\varepsilon}(n)$, and so $h = \mathcal{O}\left(\log \frac{\mathcal{S}_{\text{ol}}(\varepsilon,\delta)}{\mathcal{S}_{\text{off}}(\varepsilon,\delta)}\right) = \mathrm{polylog}_{\mathcal{Q},d,\varepsilon}(\log(n))$. In this work, we design efficient encoding scheme to store the offset of each coreset with $\mathcal{S}_{\text{loc-encode}}\left(\frac{\varepsilon}{h}, \frac{\delta}{\ell}\right) = \mathrm{polylog}_{\mathcal{Q},d,\varepsilon}(\log(n)) \cdot \mathcal{S}_2(\varepsilon, \delta)$ bits of space, i.e., it only loses $\mathrm{polylog}_{\mathcal{Q},d,\varepsilon}(\log(n))$ factors. Therefore, the second term in our space complexity only loses $\mathrm{polylog}_{\mathcal{Q},d,\varepsilon}(\log(n))$ factors, which becomes an $o(1)$ term in words of space as we assume that each word stores $\log(n)$ bits. In addition, the first term does not multiply h since we only store a global constant-factor approximation. Thus, our framework matches the words of space of the best offline algorithm, giving $\mathcal{O}_{\mathcal{Q},d,\varepsilon}(1)$ words of space as we desire.

3 Efficient Encodings for Space-Optimal Streaming Algorithms

Now, we present efficient encoding schemes for the three essential data summarization problems we study. Combining with our framework in Section 2, we obtain space-optimal streaming algorithms.

3.1 (k, z)-Clustering

We now give our efficient encoding for a given coreset for (k, z)-clustering. Given a dataset X, which can be viewed as either the original input or a set of weighted points that forms a coreset of some underlying dataset, we first acquire a constant-factor approximation C' for (k, z)-clustering on X. For each $x \in X$, let $\pi_{C'}(X)$ be the closest center of C' to x. We then write each $x \in X$ as $x = \pi_{C'}(x) + (x - \pi_{C'}(x))$, thereby decomposing x into the closest center $\pi_{C'}(x)$ and its offset $x - \pi_{C'}(x)$ from the center. We show that for the purposes of $(1 + \varepsilon)$-approximation for (k, z)-clustering, we can afford to round each coordinate of $x - \pi_{C'}(x)$ to a power of $(1 + \varepsilon')$, where $\varepsilon' = \text{poly}\left(\varepsilon, \frac{1}{d}, \frac{1}{\log(n\Delta)}\right)$, forming a vector y'. Finally, we can store C' and thus encode the vector $x' = \pi_{C'}(x) + y'$ using $\mathcal{O}\left(\log k + d \log\left(\frac{1}{\varepsilon}, d, \log(n\Delta)\right)\right)$ bits, by storing the identity of $\pi_{C'}(x)$ and the exponent of the offset for each of the d coordinates. We give the algorithm in full in Algorithm 1.

Algorithm 1 Efficient Encoding for Coreset Construction for (k, z)-Clustering

Require: Data set $X \subset [\Delta]^d$ with weight $w(\cdot)$, accuracy parameter $\varepsilon \in (0, 1)$, number of clusters k, parameter $z \geq 1$, failure probability $\delta \in (0, 1)$
Ensure: $(1 + \varepsilon)$-coreset for (k, z)-clustering

$\varepsilon' \leftarrow \frac{\text{poly}(\varepsilon^z)}{\text{poly}(k, \log(nd\Delta)}$
Find a set C' of k centers that is a constant-factor approximation to (k, z)-clustering on X
for each $x \in X$ **do**
  Let $c'(x)$ be the closest center of C' to x
  Let y' be the offset $x - c'(x)$
  Let y be y' with coordinates rounded to a power of $(1 + \varepsilon')$
  Let $x' = (c'(x), y)$, storing the *exponent* for each coordinate of y
  $X' \leftarrow X' \cup \{x'\}$
end for

return (C', X')

We state the guarantees of Algorithm 1 and postpone the analysis to Appendix D.

Theorem 3.1. *Given a constant-factor approximation C' and let X be a coreset construction with weights bounded by $[1, \text{poly}(nd\Delta)]$, then X' is a $(1 + \varepsilon)$-coreset for X that uses $\mathcal{O}\left(dk \log(n\Delta)\right) + |X| \cdot \text{polylog}\left(k, \frac{1}{\varepsilon}, \log(nd\Delta), \log\frac{1}{\delta}\right)$ bits of space.*

Thus, we encode a given coreset using $\mathcal{O}\left(dk + |X|\right)$ words of space. Now recall that directly implementing the local efficient encoding does not give our desirable result. Instead, given a partition $X_1 \sqcup \ldots \sqcup X_m$ of X, we can apply Algorithm 5 to round each segment X_i on the global constant-factor approximation C' of X, and although the output X_i' might not be a strong coreset for each corresponding X_i, the resulting error will only be an additive $\varepsilon' \cdot \text{cost}(X, C)$, which suffices for our purposes. In particular, we will have $\varepsilon' = \frac{\varepsilon}{\text{poly}(\log\log(nd\Delta))}$ and the total additive error is $\varepsilon \cdot \text{cost}(X, C)$, which eventually leads to a space-optimal streaming algorithm via our framework.

3.2 L_p Subspace Embedding

Next, we describe our approach for L_p subspace embeddings. Given a matrix $\mathbf{A}$, which can be viewed as either the original matrix or a set of reweighted rows that forms a coreset of some underlying matrix, we first acquire a constant-factor approximation $\mathbf{M}$ for L_p subspace embedding on $\mathbf{A}$. Now note that $\mathbf{M}$ has a different dimension so the approach of rounding the offset does not make sense. One possible alternative is to project $\mathbf{A}$ onto $\mathbf{M}$ and round the resulting rows, but this could have large error due to cancellation of rounded entries. For instance, suppose that there is a row $\mathbf{r} = (A, B)$ for some large values of $A, B > 0$. By rounding each coordinate of $\mathbf{r}$ to the nearest power of $(1 + \varepsilon')$ for some $\varepsilon' = \text{poly}\left(\frac{\varepsilon}{d, \log(nM)}\right)$, we obtain a row $\mathbf{r}' = (A', B')$. But for $\mathbf{v} = (B, -A)$, we have $\langle \mathbf{v}, \mathbf{r} \rangle = 0$, whereas $\langle \mathbf{v}, \mathbf{r}' \rangle$ can be as large as $\varepsilon' \cdot (|A| + |B|)$. Moreover, this error can compound across all rows of the matrix $\mathbf{A}$ so that the resulting additive error to the estimate $\|\mathbf{A}\mathbf{v}\|_p^p$ could be as large as $(\varepsilon')^p \cdot \|\mathbf{A}\|_p^p$ and so ε' needs to be scaled by a prohibitively small factor.

We instead observe that the correct procedure is to first multiply the matrix $\mathbf{A}$ with a preconditioner so that none of its rows can contribute a large amount to the error. We then use the existence of a well-conditioned basis to compute a deterministic preconditioner $\mathbf{P} \in \mathbb{R}^{d\times d}$, so that $\mathbf{MP}^{-1}$ has condition number $\mathrm{poly}(d)$. Now for each row $\mathbf{a}_t$ of $\mathbf{A}$, let $\mathbf{b}'_t$ be $\mathbf{a}_t\mathbf{P}$ with each coordinate rounded to a power of $(1+\varepsilon')$ for $\varepsilon' = \frac{\mathrm{poly}(\varepsilon)}{\mathrm{poly}(d^p, \log(nd))}$. We then store the exponent of each rounded coordinate of $\mathbf{B}'$, as well as $\mathbf{M}$. The algorithm appears in full in Algorithm 2.

Algorithm 2 Efficient Encoding for Coreset Construction for L_p Subspace Embedding

Require: Matrix $\mathbf{A} \subset \mathbb{R}^{r\times d}$, accuracy parameter $\varepsilon \in (0,1)$, failure probability $\delta \in (0,1)$
Ensure: $(1+\varepsilon)$-coreset for L_p subspace embedding

$\varepsilon' \leftarrow \frac{\mathrm{poly}(\varepsilon)}{\mathrm{poly}(d^p, \log(nd\Delta))}$
Find a matrix $\mathbf{M}$ that is a constant-factor L_p subspace embedding on $\mathbf{A}$
Compute a deterministic preconditioner $\mathbf{P}$ of $\mathbf{M}$ using well-conditioned bases, such that $\mathbf{MP}^{-1}$ has condition number $\mathrm{poly}(d)$
for each row $\mathbf{a} \in \mathbf{A}$ **do**
  Let $\mathbf{b}'$ be $\mathbf{aP}$ with each coordinate rounded to a power of $(1+\varepsilon')$
  $\mathbf{B}' \leftarrow \mathbf{B}' \circ \mathbf{b}'$, storing the *exponent* for each entry of $\mathbf{b}'$
end for

return $(\mathbf{M}, \mathbf{B}')$, from which estimate $\mathbf{A}'$ can be constructed

We state the guarantees of Algorithm 2 and postpone the analysis to Appendix E.

Theorem 3.2. *Let $\mathbf{A} \in \mathbb{R}^{r\times d}$ be a coreset construction with entries in $\{0\} \cup \left[\frac{1}{\mathrm{poly}(n)}, \mathrm{poly}(n)\right]$ and let $\mathbf{M} \in \mathbb{R}^{m\times d}$ be a constant-factor approximation. Then $\mathbf{A}'$ is a $(1+\varepsilon)$-coreset for $\mathbf{A}$ for L_p subspace embedding that uses $md\log n + rd \cdot \mathrm{polylog}\left(d, \frac{1}{\varepsilon}, \log(nd\kappa), \log\frac{1}{\delta}\right)$ bits of space.*

Thus, we encode a given subspace embedding using $\mathcal{O}\left((m+r)d\right)$ words of space. Similarly, we obtain the global encoding by first deriving the preconditioner $\mathbf{P}$ based on a global constant-factor approximation $\mathbf{M}$, and then projecting each row of each local L_p subspace embedding to $\mathbf{P}$.

3.3 Projection-Cost Preserving Sketches

In this section, we describe the efficient encoding for projection-cost preserving (PCP) sketches. Recall that we round a PCP sketch to its optimal rank-k approximation: given a matrix $\mathbf{A}$, let $\mathbf{A}_k$ be the optimal rank-k approximation to $\mathbf{B}$, we store $\mathbf{A}_k$ with high precision and round the offset $\mathbf{D} = \mathbf{A} - \mathbf{A}_k$ using entry-wise relative error quantization with precision parameter ε' to obtain $\mathbf{D}'$. Then it can be shown that the reconstruction matrix is still a $(1+\varepsilon)$-PCP sketch $\mathbf{A}' := \mathbf{A}_k + \mathbf{D}'$ for $\varepsilon' \le \mathcal{O}(\varepsilon)$, i.e., $\|\mathbf{A}' - \mathbf{A}'\mathbf{P}\|_F \in (1+\varepsilon)\|\mathbf{A} - \mathbf{AP}\|_F$ for any rank-k projection matrix $\mathbf{P}$.

Similar to (k,z)-clustering and L_p subspace embedding, we need a global efficient encoding since we cannot afford to store the low-rank matrix $\mathbf{A}_k$ of each active node, but the construction is more complicated. We observe that $\mathbf{A}_k$ is a projection of the rows of $\mathbf{A}$ onto the span of its top-k right singular vectors. Based on this intuition, we project the PCP-sketch $\mathbf{Q}_i$ on each active node to the top-k right singular vectors of $\mathbf{A}$, represented by $\mathbf{V}_k$. To obtain the efficient encoding, for each node i, we compute the head coefficients $\mathbf{Q}_i\mathbf{V}_k$ and store with high precision; and then we compute the tail residual $\mathbf{T}_i = \mathbf{Q}_i(\mathbb{I}_d - \mathbf{V}_k\mathbf{V}_k^\top)$ and store with quantization parameter ε'. Note that in reconstruction, we write the matrix as $\mathbf{Q}_i\mathbf{V}_k\mathbf{V}_k^\top + \mathbf{Q}_i(\mathbb{I}_d - \mathbf{V}_k\mathbf{V}_k^\top)$, where the first part corresponds to $\mathbf{A}_k$ and the second part corresponds to $\mathbf{A} - \mathbf{A}_k$. We present the formal description in Algorithm 3.

We state the guarantees of Algorithm 3 and postpone the analysis to Appendix F.

Theorem 3.3. *Let $\mathbf{A} = \mathbf{Q}_1 \circ \ldots \circ \mathbf{Q}_m \in \mathbb{R}^{r\times d}$ be a coreset construction with entries in $\{0\} \cup \left[\frac{1}{\mathrm{poly}(n)}, \mathrm{poly}(n)\right]$, then $\mathbf{Q}'$ is a $(1+\varepsilon)$-PCP sketch for $\mathbf{A}$ that uses $\mathcal{O}\left(kd\log n + mrk\log n + mrd\left(\log\frac{1}{\varepsilon'} + \log\log n\right)\right)$ bits of space.*

Thus, we encode a given PCP sketch using $\mathcal{O}\left(kd + mrd\right)$ words of space, which leads to the space-optimal algorithm in Theorem 1.3 through our framework. Here, another technical challenge is that,

Algorithm 3 Global efficient encoding for low-rank approximation

Require: Accuracy parameter $\varepsilon < \frac{1}{2}$, matrix $\mathbf{A} = \mathbf{Q}_1 \circ \ldots \circ \mathbf{Q}_m \in \mathbb{R}^{r \times d}$, rank parameter k
Ensure: Encodings $\mathbf{Q}'_1, \ldots, \mathbf{Q}'_m$.

Let ε' denote the quantization precision to be determined later. Let $\varepsilon' \leq \frac{\varepsilon}{4\sqrt{C_\varepsilon}}$, where $C_\varepsilon = \frac{1+\varepsilon}{1-\varepsilon}$
Compute the SVD of $\mathbf{A}$ and let $\mathbf{V}_k \in \mathbb{R}^{d \times k}$ be the top-k right singular vectors. Define projections $\mathbf{P}_\mathbf{A} = \mathbf{V}_k \mathbf{V}_k^\top$ and $\mathbf{P}_\mathbf{A}^\perp = \mathbb{I}_d - \mathbf{P}_\mathbf{A}$
Store $\mathbf{V}_k$ using high precision (e.g., $\mathcal{O}(\log n)$ bits per entry)
for $i \in [m]$ **do**
    Compute head coefficients $\mathbf{H}_i = \mathbf{Q}_i \mathbf{V}_k$ and store with high precision
    Compute tail residual $\mathbf{T}_i = \mathbf{Q}_i \mathbf{P}_\mathbf{A}^\perp$
    Round $\mathbf{T}_i$ entry-wise with relative precision ε' to obtain $\mathbf{T}'_i$. Store $\mathbf{T}_i$
end for
In reconstruction, compute $\mathbf{Q}'_i = \mathbf{H}_i \mathbf{V}_k^\top + \mathbf{T}'_i$ for each $i \in [m]$

return $\mathbf{Q}' = \mathbf{Q}'_1 \circ \ldots \circ \mathbf{Q}'_m$

unlike (k, z)-clustering and L_p subspace embedding, where the reconstructed coreset can be directly used to define the next online sampling probability, a PCP sketch does not necessarily provide a valid estimation of the online sensitivity, as this would require subspace embedding properties. Fortunately, we show that by properly choosing the quantization factor, the reconstructed PCP sketch also gives a constant-factor L_2-subspace embedding, which suffices to give a good approximation.

Remark on running time. On every arrival, we decode the current sketch and use it to estimate the new item's online sampling score. When a leaf is full, we apply the offline coreset construction to each pair of summaries that must be merged, then re-encode the active coresets using a common reference. A tree with ℓ leaves performs $O(\ell)$ such merges and encoding rounds, so their cost is amortized over the n arrivals and is negligible for $n \gg \mathrm{poly}(d, 1/\varepsilon)$. For L_p subspace embeddings with fixed p and for PCP sketches, the resulting amortized update time is $\mathrm{poly}(d)$ (see Theorem E.15 and Theorem F.14). The exponent of d depends in part on the matrix multiplication exponent ω through the computation of online Lewis weights for embeddings and ridge leverage scores for PCP sketches. For clustering, decoding adds each stored offset to its reference center, while estimating the online sensitivity by enumerating candidate sets of k centers can take exponential time in k. We leave improving these update times while preserving our space bounds to future work.

4 Experimental Results

In this section, we provide empirical validation of our novel techniques. Our goal is to demonstrate that our efficient encoding approach not only provides theoretical space guarantees but also yields practical improvements in memory efficiency without sacrificing approximation quality. The experiments are conducted using an Apple M2 CPU, with 16 GB RAM and 8 cores.

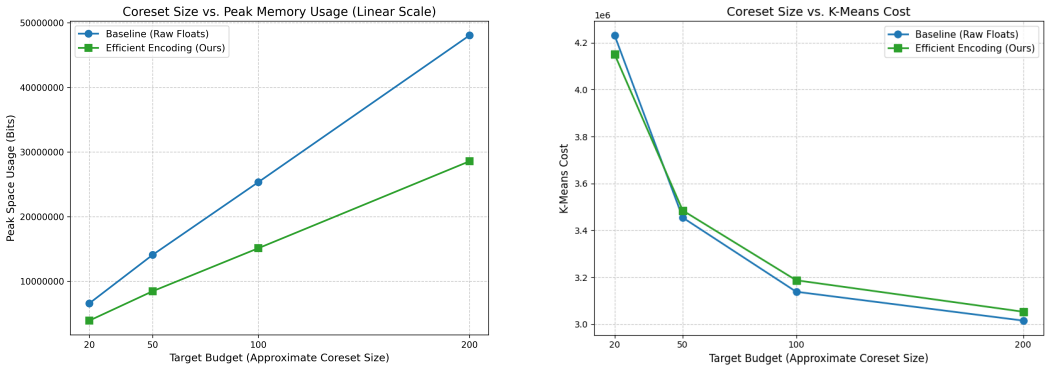
Real-world dataset. We conducted our k-means clustering algorithms on the MNIST dataset [LBBH98], which consists of $n = 70{,}000$ images of handwritten digits 0–9, and each image has a dimension of 28×28, represented by $d = 784$ features corresponding to pixel intensities. We set $k = 10$, corresponding to the ten digit classes. MNIST serves as a large-scale dataset to evaluate the memory efficiency in streaming coreset construction. The dataset is publicly available through the MNIST website and the UCI repository.

Baseline. Our primary baseline is the k-means coreset construction algorithm through the framework in [CWZ23], where the algorithm first conducts online sensitivity sampling on the data stream, and then passes the selected points to a merge-and-reduce structure. This baseline achieves the state-of-the-art worst-case upper bounds on space complexity among prior streaming algorithms (see Figure 1). We compare our algorithm in Figure 3, which efficiently encodes each coreset obtained by the merge-and-reduce structure, with the baseline and demonstrate our ability in space reduction.

Evaluation Metrics. We compare the performance of the two methods on *clustering cost* and *space consumption*. Once the algorithms return the k-means coresets at the end of the stream, we run the same k-means heuristic in SKLEARN.CLUSTER on the reported coresets and calculate the clustering

cost with respect to the entire dataset, which is the main evaluation of the quality of the constructed coresets. We then compute the peak space consumption (in bits) of the two methods required to maintain the data structure during the stream. We vary the sampling coefficients of the two methods to obtain target sample budgets of approximately 20, 50, 100, and 200, and compare the three metrics under each budget.

Results and Discussion. We summarize our empirical results on the MNIST dataset in Figure 4, which plots the average clustering costs and peak space consumption over 5 repetitions. The full result table is deferred to Appendix G.



(a) Space consumption vs. target budget

(b) Clustering cost vs. target budget

Fig. 4: Empirical comparison on the MNIST dataset. Target budgets indicate approximate final coreset sizes.

As shown in the "Space consumption vs. target budget" comparison in Figure 4, our efficient encoding scheme consistently outperforms the baseline on space usage, achieving a reduction of approximately 40% across all tested budgets. Thus by storing each sampled point using efficient quantization instead of high precision, our algorithm requires significantly fewer bits to store the data structure than the baseline. Despite the reduced space, as shown in the "Clustering cost vs. target budget" comparison in Figure 4, our algorithm achieves clustering costs within 2% of the baseline across all tested budgets, indicating that efficient encoding and reconstruction preserve comparable empirical coreset quality. These results demonstrate the effectiveness of our efficient encoding framework in reducing memory consumption while maintaining comparable clustering quality.

Acknowledgments

David P. Woodruff is supported in part by Office of Naval Research award number N000142112647 and a Simons Investigator Award. Shenghao Xie is supported in part by NSF CCF-2335411, the work was completed when the author was affiliated with Texas A&M University. Samson Zhou is supported in part by NSF CCF-2335411 and gratefully acknowledges funding provided by the Oak Ridge Associated Universities (ORAU) Ralph E. Powe Junior Faculty Enhancement Award.

References

[BBC+19] Luca Becchetti, Marc Bury, Vincent Cohen-Addad, Fabrizio Grandoni, and Chris Schwiegelshohn. Oblivious dimension reduction for *k*-means: beyond subspaces and the johnson-lindenstrauss lemma. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1039–1050. ACM, 2019. 3

[BDM+20] Vladimir Braverman, Petros Drineas, Cameron Musco, Christopher Musco, Jalaj Upadhyay, David P. Woodruff, and Samson Zhou. Near optimal linear algebra in the online and sliding window models. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 517–528, 2020. 4, 24, 30, 31

[BFLR19] Vladimir Braverman, Dan Feldman, Harry Lang, and Daniela Rus. Streaming coreset constructions for m-estimators. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pages 62:1–62:15, 2019. 3

[BS80] Jon Louis Bentley and James B. Saxe. Decomposable searching problems I: static-to-dynamic transformation. *J. Algorithms*, 1(4):301–358, 1980. 2, 5, 14

[BSS12] Joshua Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-Ramanujan sparsifiers. *SIAM Journal on Computing*, 41(6):1704–1721, 2012. 5

[CD21] Xue Chen and Michal Derezinski. Query complexity of least absolute deviation regression via robust uniform convergence. In *Conference on Learning Theory, COLT*, volume 134, pages 1144–1179, 2021. 24, 26

[CEM+15] Michael B. Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC*, pages 163–172, 2015. 4, 30

[Che09] Ke Chen. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM J. Comput.*, 39(3):923–947, 2009. 3

[CLS+22] Vincent Cohen-Addad, Kasper Green Larsen, David Saulpic, Chris Schwiegelshohn, and Omar Ali Sheikh-Omar. Improved coresets for euclidean k-means. In *NeurIPS*, 2022. 19

[CLSS22] Vincent Cohen-Addad, Kasper Green Larsen, David Saulpic, and Chris Schwiegelshohn. Towards optimal lower bounds for k-median and k-means coresets. In *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1038–1051, 2022. 2, 3, 19

[CMM17] Michael B. Cohen, Cameron Musco, and Christopher Musco. Input sparsity time low-rank approximation via ridge leverage score sampling. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1758–1777, 2017. 4, 30, 35

[CMP20] Michael B. Cohen, Cameron Musco, and Jakub Pachocki. Online row sampling. *Theory Comput.*, 16:1–25, 2020. 2, 4, 5, 24, 28, 30, 37

[CP15] Michael B. Cohen and Richard Peng. l_p row sampling by lewis weights. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC*, pages 183–192, 2015. 2, 4, 24, 26

[CSS21] Vincent Cohen-Addad, David Saulpic, and Chris Schwiegelshohn. A new coreset framework for clustering. In *STOC: 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 169–182, 2021. 3

[CSWZ23] Yeshwanth Cherapanamjeri, Sandeep Silwal, David P. Woodruff, and Samson Zhou. Optimal algorithms for linear algebra in the current matrix multiplication time. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 4026–4049, 2023. 24

[CW09] Kenneth L. Clarkson and David P. Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC*, pages 205–214, 2009. 24, 25

[CWZ23] Vincent Cohen-Addad, David P. Woodruff, and Samson Zhou. Streaming euclidean k-median and k-means with $o(\log n)$ space. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 883–908, 2023. 2, 3, 4, 5, 9, 14, 15, 19

[DDH+09] Anirban Dasgupta, Petros Drineas, Boulos Harb, Ravi Kumar, and Michael W. Mahoney. Sampling algorithms and coresets for l_p regression. *SIAM J. Comput.*, 38(5):2060–2078, 2009. 4, 24, 25

[DMM06a] Petros Drineas, Michael W. Mahoney, and S. Muthukrishnan. Subspace sampling and relative-error matrix approximation: Column-based methods. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX and 10th International Workshop on Randomization and Computation, RANDOM, Proceedings*, pages 316–326, 2006. 4, 24

[DMM06b] Petros Drineas, Michael W. Mahoney, and S. Muthukrishnan. Subspace sampling and relative-error matrix approximation: Column-row-based methods. In *Algorithms - ESA 2006, 14th Annual European Symposium, Proceedings*, pages 304–314, 2006. 4, 24

[FKV04] Alan Frieze, Ravi Kannan, and Santosh Vempala. Fast monte carlo algorithms for finding low-rank approximations. *J. ACM*, 51(6):1025–1041, 2004. 3

[FL11] Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC*, pages 569–578, 2011. 1, 3

[HK07] Sariel Har-Peled and Akash Kushal. Smaller coresets for k-median and k-means clustering. *Discret. Comput. Geom.*, 37(1):3–19, 2007. 3

[HLW23] Lingxiao Huang, Jian Li, and Xuan Wu. On optimal coreset construction for euclidean (k, z)-clustering, 2023. 19

[HM04] Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 291–300, 2004. 1, 2, 3, 5, 14

[JL84] William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. conference in modern analysis and probability (new haven, conn., 1982), 189–206. In *Contemp. Math*, volume 26, 1984. 16

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 9

[LWW21] Yi Li, Ruosong Wang, and David P. Woodruff. Tight bounds for the subspace sketch problem with applications. *SIAM J. Comput.*, 50(4):1287–1335, 2021. 4

[Mac67] J MacQueen. Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*, pages 281–297, 1967. 2

[MM20] Cameron Musco and Christopher Musco. Projection-cost-preserving sketches: Proof strategies and constructions. *CoRR*, abs/2004.08434, 2020. 30

[MMM+22] Raphael A. Meyer, Cameron Musco, Christopher Musco, David P. Woodruff, and Samson Zhou. Fast regression for structured inputs. In *The Tenth International Conference on Learning Representations, ICLR, 2022*, 2022. 24

[MMM+23] Raphael A. Meyer, Cameron Musco, Christopher Musco, David P. Woodruff, and Samson Zhou. Near-linear sample complexity for L_p polynomial regression. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 3959–4025, 2023. 24

[MMWY22] Cameron Musco, Christopher Musco, David P. Woodruff, and Taisuke Yasuda. Active linear regression for l_p norms and beyond. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 744–753, 2022. 24

[MWZ22] Sepideh Mahabadi, David P. Woodruff, and Samson Zhou. Adaptive sketches for robust regression with importance sampling. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pages 31:1–31:21, 2022. 24

[PPP21] Aditya Parulekar, Advait Parulekar, and Eric Price. L1 regression with lewis weights subsampling. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pages 49:1–49:21, 2021. 24

[S+56] Hugo Steinhaus et al. Sur la division des corps matériels en parties. *Bull. Acad. Polon. Sci*, 1(804):801, 1956. 2

[Sar06] Tamás Sarlós. Improved approximation algorithms for large matrices via random projections. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), Proceedings*, pages 143–152, 2006. 2, 4, 24

[SW18] Christian Sohler and David P. Woodruff. Strong coresets for k-median and subspace approximation: Goodbye dimension. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 802–813, 2018. 20

[Woo14] David P. Woodruff. Sketching as a tool for numerical linear algebra. *Found. Trends Theor. Comput. Sci.*, 10(1-2):1–157, 2014. 2, 3

[WY23] David P. Woodruff and Taisuke Yasuda. Online lewis weight sampling. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 4622–4666, 2023. 2, 4, 5, 24, 25, 28, 30

[WZZ23] David P. Woodruff, Peilin Zhong, and Samson Zhou. Near-optimal k-clustering in the sliding window model. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems, NeurIPS 2023*, 2023. 3

[ZTHH24] Xiaoyi Zhu, Yuxiang Tian, Lingxiao Huang, and Zengfeng Huang. Space complexity of euclidean clustering. *arXiv preprint arXiv:2403.02971*, 2024. 3, 15

A Technical Overview

In this section, we provide a high-level intuition behind our framework in Figure 3 and the efficient encoding scheme for each problem.

A.1 Streaming Framework for Optimal Space

We begin with constructing the coreset in the offline setting, where the dataset X is fixed and known by the algorithm. For X and a family of queries $\mathcal{Q}$, the standard method is to sample each point $x \in X$ with probability proportional to its *sensitivity*, i.e., $p(x) \propto s(x)$, and re-weight the point by the inverse $\frac{1}{p(x)}$ if sampled. Here, the sensitivity of x is defined by $s(x) = \max_{q \in \mathcal{Q}} \frac{\mathrm{cost}_q(x,C)}{\mathrm{cost}_q(X,C)}$, where $\mathrm{cost}_q(X, C)$ is the weighted sum of all elements in X; and captures the important elements in the dataset with respect to the queries $\mathcal{Q}$, often leading to a good summarization of the entire dataset.

In the streaming setting, it is challenging to compute these sensitivities since we do not have access to the entire dataset. The commonly-used online sensitivity approach samples each item x_t of the stream with probability proportional to its *online sensitivity*, which measures the importance of x_t with respect to the points of the stream that have already arrived, i.e., its sensitivity with respect to $X_t = \{x_1, \ldots, x_t\}$. To compute the online sensitivities, we use the sensitivity of an element with respect to the previously sampled elements to give a close approximation, since we cannot afford to store the entire prefix X_t. However, a crucial drawback of this approach is that future sampling probabilities now depend on whether the past elements are sampled, and so standard concentration bounds that require independence do not apply. Although the correctness can be taken care of by martingale or decoupling analysis, these limitations lose poly-logarithmic factors in the number of samples, which is prohibitive for our purpose.

On the other hand, merge-and-reduce [BS80, HM04] constructs a binary tree on the data stream, where each node contains a block of points, and then "merges" two child nodes by offline coreset methods. Formally, the approach framework first partitions the stream into consecutive blocks of size $f(\mathcal{Q}, n, d, \varepsilon', \delta')$, and then constructs a coreset with accuracy $(1+\varepsilon')$ and failure probability δ' for each block of the stream. For each node in the tree at depth ℓ, the framework takes the coresets of its two children at depth $\ell+1$ and applies the same offline summarization algorithm to their union, producing a new coreset with accuracy $(1+\varepsilon')$. Observe that the coreset at the root of the tree represents a valid $(1+\varepsilon)$-approximation for the entire stream for sufficiently small ε'. However, the merge-and-reduce tree has height roughly $\log_{\mathcal{Q},d,\varepsilon}(n)$ and the error in each level accumulates, so we need to construct offline coreset with finer accuracy, i.e., $\varepsilon' = \frac{\varepsilon}{\log_{\mathcal{Q},d,\varepsilon}(n)}$, losing additional $\mathrm{polylog}_{\mathcal{Q},d,\varepsilon}(n)$ factors, and thus, it cannot immediately be used to achieve our goal of streaming algorithms with space (in words) independent of n.

Instead of directly implementing the merge-and-reduce method on the input stream, a recent framework proposed by [CWZ23] first down-sample the stream via the online sampling scheme and then passes the resulted stream $\mathcal{S}'$ on to a merge-and-reduce method. Now, the height of the merge-and-reduce coresets becomes $\mathcal{O}(\log(|\mathcal{S}'|)) = \mathrm{polylog}_{\mathcal{Q},d,\varepsilon}(\log(n))$, and so it suffices to have $\varepsilon' = \frac{\varepsilon}{\mathcal{O}(\log|\mathcal{S}'|)} = \frac{\varepsilon}{\mathrm{polylog}_{\mathcal{Q},d,\varepsilon}(\log(n))}$, using less space compared to the previous choice of $\varepsilon' = \frac{\varepsilon}{\log_{\mathcal{Q},d,\varepsilon}(n)}$. Although this method reduces $\mathrm{polylog}_{\mathcal{Q},d,\varepsilon}(n)$ factors to $\mathrm{polylog}_{\mathcal{Q},d,\varepsilon}(\log(n))$ factors, it still does not suffice for our goal of having space usage comparable to offline settings.

Efficient encoding framework. To solve the above problem, we propose a framework via efficient encoding schemes. Trivially, one can store the solution with high precision, but this would require $\mathcal{O}(\log(n))$ bits for each item. To achieve our goal of $\mathcal{O}_{\mathcal{Q},d,\varepsilon}(1)$ words of space, we design the efficient encoding scheme to store each item with less space. Consider an "active" node in the merge-and-reduce tree (i.e., a node that is not merged to a parent node), our strategy is to store an inaccurate sketch, e.g., a constant-factor approximation coreset of the points in the node, with high precision. Then we store the offset of the $(1+\varepsilon)$-coreset obtained by the merge-and-reduce tree using quantization with precision parameter ϵ, e.g., rounding the value to the nearest power of $(1+\epsilon)$. Note that the size of the inaccurate sketch does not contain the additional $\mathrm{polylog}_{\mathcal{Q},d,\varepsilon}(\log(n))$ factors, and so storing it with high precision does not incur factors involving n in the number of words. On the other hand, due to the efficient encoding scheme that stores the offsets, we avoid using $\log(n)$ bits to store each point. Importantly, when we need to conduct some operations using the coresets, e.g., merging two coresets or computing the next online sensitivity, we can reconstruct the coresets from the efficient encoding effectively. By appropriately choosing the precision parameter ϵ, we ensure the accuracy of the reconstructed coresets while achieving $\mathcal{O}_{\mathcal{Q},d,\varepsilon}(1)$ bits of words.

Unfortunately, the merge-and-reduce tree implementing on $\mathcal{S}'$ has height $\mathcal{O}\left(\log_{\mathcal{Q},d,\varepsilon}(|S'|)\right) = \mathrm{polylog}_{\mathcal{Q},d,\varepsilon}(\log(n))$ and therefore we need to efficiently encode $\mathrm{polylog}_{\mathcal{Q},d,\varepsilon}(\log(n))$ active coresets simultaneously. If we apply the efficient encoding scheme separately on each node, it requires storing the inaccurate sketch in each of them, and so the total space remains $\mathrm{polylog}_{\mathcal{Q},d,\varepsilon}(\log(n))$, which does not meet our requirement. To solve this, we implement a global efficient encoding scheme across all nodes. Namely, we store a single inaccurate sketch of the entire stream with high precision, rather than storing an inaccurate sketch for each coreset, and then encode the offset of each simultaneously stored coreset to the sketch. We observe that whenever we need to do a merge operation at some time t, we use the stored coreset to recompute a constant-factor approximation C' to X_t, which is the entire prefix dataset arrived before t. For each coreset for a subset S_v of data points representing an active node v in the tree, we can then perform the efficient encoding with respect to C' instead of a separate constant-factor approximation. Although each encoding no longer guarantees $(1+\varepsilon)$-multiplicative approximation to $\mathrm{cost}_q(S_v)$ for a query $q \in \mathcal{Q}$, it still guarantees $\varepsilon \cdot \mathrm{cost}_q(X_t)$ additive error to $\mathrm{cost}_q(S_v)$, which suffices for a $(1+\varepsilon)$-approximation to $\mathrm{cost}_q(X_t)$ after summing across all subsets S_v at time t. In addition, if the recomputed sketch C' suffices to approximate the online sensitivity of the next insertion, then the efficient encoding ensures the correctness of the entire framework.

We then show that our framework provides space-optimal streaming algorithms for various high-dimensional optimization problems; we believe that it might be applied to other problems, e.g., graph sparsification, and so it is of independent interest. However, the efficient encoding schemes are not easy to obtain, especially, although the reconstructed sketch still gives a $(1+\varepsilon)$-coreset, it might

break the structural properties required to estimate the next online sampling probability. For instance, being a $(1+\varepsilon)$-projection-cost preservation sketch does not guarantee such properties directly, which needs further analysis.

A.2 Efficient Encoding Schemes for Streaming Problems

Given the framework that gives us optimal space, it remains to develop an efficient encoding scheme for each problem.

(k, z)-clustering. We first present the efficient encoding of the coresets in (k, z)-clustering. Suppose we have a set C' of $\mathcal{O}(k)$ centers that is a constant-factor approximation to the optimal (k, z)-clustering on X. We can noisily encode X by writing each $x \in X$ as $x = \pi_{C'}(x) + (x - \pi_{C'}(x))$, where $\pi_{C'}(x)$ is the closest center of C' to x. We then round each coordinate of $x - \pi_{C'}(x)$ by rounding each coordinate to a power of $(1+\varepsilon')$ for $\varepsilon' = \text{poly}\left(\varepsilon, \frac{1}{d}, \frac{1}{\log(n\Delta)}\right)$ to form a vector y'. Hence, given C', each vector $x' = \pi_{C'}(x) + y'$ can be encoded using $\mathcal{O}\left(\log k + d \log \frac{1}{\varepsilon'}\right)$ bits, since we can store the *exponents* of the offsets. It is known [CWZ23, ZTHH24] that if X' is the set of all points of X rounded in this manner, then X' is a $(1+\varepsilon)$-coreset for X. Moreover, we can round the weights of X' to powers of $(1+\varepsilon')$, thereby efficiently encoding each coreset in the merge-and-reduce framework to improve the overall space complexity of the problem. Recall that a local encoding scheme is insufficient to achieve optimal space to store the merge-and-reduce tree. To overcome this issue and obtain a global efficient encoding, we provide a single constant-factor approximation to the global dataset, rather than providing a constant-factor approximation for each level of the merge-and-reduce tree.

Subspace embeddings. Next, we describe our approach for L_p subspace embeddings. It is not immediately obvious how to utilize a constant-factor subspace embedding $\mathbf{M} \in \mathbb{R}^{m \times d}$ of the input matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ to achieve an efficient encoding, especially since the dimensions are different. One possible approach is to project $\mathbf{A}$ onto $\mathbf{M}$ and round the resulting rows, but the rounding process could result in large error due to cancellation of rounded entries. For example, suppose that after the projection, there is a row $\mathbf{r} = (A, B)$ for some large values of $A, B > 0$. By rounding each coordinate of $\mathbf{r}$ to the nearest power of $(1+\varepsilon')$ for some $\varepsilon' = \text{poly}\left(\frac{\varepsilon}{d, \log(nM)}\right)$, we obtain a row $\mathbf{r}' = (A', B')$. Now for $\mathbf{v} = (B, -A)$, we have $\langle \mathbf{v}, \mathbf{r} \rangle = 0$, but $\langle \mathbf{v}, \mathbf{r}' \rangle$ can be as large as $\varepsilon' \cdot (|A| + |B|)$. Moreover, this error can compound across all rows of the matrix $\mathbf{A}$ so that the resulting additive error to the estimate $\|\mathbf{A}\mathbf{v}\|_p^p$ could be as large as $(\varepsilon')^p \cdot \|\mathbf{A}\|_p^p$.

We instead observe that the correct procedure is to first multiply the matrix $\mathbf{A}$ with a preconditioner so that none of its rows can contribute a large amount to the error. To that end, we first use a well-conditioned basis to compute a preconditioner $\mathbf{P} \in \mathbb{R}^{d \times d}$ of the constant-factor approximation $\mathbf{M}$ and multiply each row of $\mathbf{A}$, so that for all vectors $\mathbf{x} \in \mathbb{R}^d$ with $\|\mathbf{A}\mathbf{x}\|_p = 1$, we have that $\frac{1}{\text{poly}(d)} \leq \|\mathbf{A}\mathbf{P}\mathbf{x}\|_p \leq \text{poly}(d)$. In particular, the maximum possible additive error $\langle \mathbf{v}, \mathbf{r}' \rangle$ for each row $\mathbf{r}$ can be charged to the sensitivity of $\mathbf{r}$, and it follows that the sum of the errors due to the rounding can be at most $\varepsilon' \cdot \text{poly}(d)$. Since $\varepsilon' = \text{poly}\left(\frac{\varepsilon}{d, \log(nM)}\right)$, then the overall error is $\mathcal{O}(\varepsilon)$, which achieves a subspace embedding because $\|\mathbf{A}\mathbf{x}\|_p = 1$. For a row $\mathbf{b}_t = \mathbf{a}_t\mathbf{P}$, we then round each entry of $\mathbf{b}_t$ to the nearest power of $(1+\varepsilon')$ and store the exponent as before. We can then also achieve a streaming algorithm by compressing the entire merge-and-reduce tree, as before.

Projection-cost preservation. To efficiently encode a matrix $\mathbf{A}$ for the purpose of obtaining projection-cost preserving sketches, we consider its optimal rank-k approximation $\mathbf{A}_k$ and write the matrix as $\mathbf{A} = \mathbf{A}_k + (\mathbf{A} - \mathbf{A}_k)$. We then store $\mathbf{A}_k$ with high precision, e.g., $\mathcal{O}(\log n)$ bits per entry, and round each entry of the residual matrix $\mathbf{D} = \mathbf{A} - \mathbf{A}_k$ to powers of $(1+\varepsilon')$ to form a matrix $\mathbf{D}'$. We note that this is slightly different from our previous approach to round to a constant-factor approximation. The intuition for rounding to $\mathbf{A}_k$ is that projection-cost preservation depends only on how well the sketch represents distances in k-dimensional subspaces. Decomposing $\mathbf{A} = \mathbf{A}_k + \mathbf{D}$ isolates the structured component $\mathbf{A}_k$ that captures the directions contributing most to the residual cost $\|\mathbf{A} - \mathbf{A}\mathbf{P}\|_F^2$ for any rank-k projection matrix $\mathbf{P}$. In contrast, the contribution of the residual matrix $\mathbf{D}$ to the projection-cost is inherently bounded by $\|\mathbf{D}\|_F^2 \leq \|\mathbf{A} - \mathbf{A}\mathbf{P}\|_F^2$ for any projection matrix $\mathbf{P}$, and so quantizing $\mathbf{D}$ with precision ε' gives us a desirable $(1+\varepsilon)$-projection-cost preserving sketch in reconstruction.

We still need a global efficient encoding since we cannot afford to store the low-rank matrix $\mathbf{A}_k$ for each node in the merge-and-reduce tree. We observe that in the local efficient encoding, $\mathbf{A}_k$ is a projection of the rows of $\mathbf{A}$ onto the span of the top-k right singular vectors. Similarly, for global encoding, we project the PCP sketch matrix $\mathbf{Q}_i$ on each node to the top-k right singular vectors of $\mathbf{A}$. To obtain the efficient encoding, for each node i, we compute the head coefficients (corresponding to $\mathbf{A}_k$) and store with high precision; and then we compute the tail residual (corresponding to $\mathbf{A} - \mathbf{A}_k$) and store with quantization (see the formal description in Appendix F). As we mentioned earlier, being a projection-cost preserving sketch does not necessarily ensure that it can be used to provide a valid estimation of the online sampling probability, as the latter would require subspace embedding properties. Fortunately, we show that by properly choosing the quantization factor, we are able to satisfy this requirement, giving the correctness of the overall algorithm.

B Additional Preliminaries

For a positive integer n, we write $[n] = \{1, 2, \ldots, n\}$. We use $\mathrm{poly}(n)$ to denote an arbitrary fixed polynomial in n, and $\mathrm{polylog}(n)$ to denote a polynomial in $\log n$. An event is said to hold with high probability if it occurs with probability at least $1 - \frac{1}{\mathrm{poly}(n)}$.

For (k, z)-clustering, we assume the dataset $X \subset [\Delta]^d$. Following standard assumptions in the streaming model, we assume $\Delta = \mathrm{poly}(n)$ and allow each memory word to store $\Theta(\log(nd\Delta))$ bits, i.e., $\Theta(\log(ndM))$ bits for for subspace embedding and projection-cost preservation. This ensures that the weight and each coordinate of a point in $\mathbb{R}^d$ can be represented using $\mathcal{O}(1)$ words.

Throughout the paper, we restrict our attention to Euclidean spaces rather than general metric spaces. For points $x, y \in \mathbb{R}^d$, we write $\mathrm{dist}(x, y) = \|x - y\|_2$. For a point x and a subset $S \subset \mathbb{R}^d$, we further define $\mathrm{dist}(x, S) = \min_{s \in S} \mathrm{dist}(x, s)$. For $z \geq 1$, the ℓ_z norm of a vector x is denoted by $\|x\|_z$, where $\|x\|_z^z = \sum_{i=1}^d |x_i|^z$. Given a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$, its Frobenius norm is

$$\|\mathbf{A}\|_F^2 = \sum_{i \in [n]} \sum_{j \in [d]} A_{i,j}^2.$$

For a fixed $z \geq 1$ and sets $X, C \subset \mathbb{R}^d$ with $X = \{x_1, \ldots, x_n\}$, we define the clustering objective

$$\mathrm{cost}(X, C) = \sum_{i=1}^n \mathrm{dist}(x_i, C)^z.$$

We will make use of the following inequality, sometimes referred to as a generalized triangle inequality.

Fact B.1 (Generalized triangle inequality)**.** *For any $z \geq 1$ and any $x, y, w \in \mathbb{R}^d$,*

$$\mathrm{dist}(x, y)^z \leq 2^{z-1} \big(\mathrm{dist}(x, w)^z + \mathrm{dist}(w, y)^z \big).$$

We also utilize the classical Johnson-Lindenstrauss dimensionality reduction result.

Theorem B.2 (Johnson-Lindenstrauss lemma)**.** *[JL84] Given an accuracy parameter $\varepsilon \in \big(0, \frac{1}{2}\big)$ and set $m = \mathcal{O}\left(\frac{1}{\varepsilon^2} \log n\right)$. For any set $X \subset \mathbb{R}^d$ of n points, there exists a distribution over linear maps $\Pi : \mathbb{R}^d \to \mathbb{R}^m$ such that, with high probability over $\pi \sim \Pi$,*

$$(1 - \varepsilon)\|x - y\|_2 \leq \|\pi x - \pi y\|_2 \leq (1 + \varepsilon)\|x - y\|_2$$

holds simultaneously for all $x, y \in X$.

We further recall the standard Hoeffding's inequality.

Theorem B.3 (Hoeffding's inequality)**.** *Let $X_1, \ldots, X_n$ be independent random variables with $a_i \leq X_i \leq b_i$, and let $S_n = \sum_{i=1}^n X_i$. Then for any $t > 0$,*

$$\mathbf{Pr}\left[|S_n - \mathbb{E}[S_n]| > t\right] \leq 2 \exp\left(- \frac{t^2}{\sum_{i=1}^n (b_i - a_i)^2} \right).$$

Next we formalize the notion of a coreset, which serves as a compact weighted proxy for a dataset while approximately preserving a class of query costs.

Definition B.4 (Coreset)**.** *Let $\mathcal{Q}$ be a family of query functions over a weighted point set $X \subseteq [\Delta]^d$ with weights $w : X \to \mathbb{R}_{>0}$. A weighted subset $Y \subseteq X$ is a $(1+\varepsilon)$-coreset for $(X, \mathcal{Q})$ if, for every $q \in \mathcal{Q}$,*

$$(1-\varepsilon)\operatorname{cost}(q, X) \leq \operatorname*{cost}_q(Y) \leq (1+\varepsilon)\operatorname{cost}(q, X),$$

where $\operatorname{cost}_q(Y) = \sum_{x \in Y} w(x)\, q(x)$.

This definition subsumes the specific coresets considered in this work. For (k, z)-clustering, a query q corresponds to a set of k centers, and $q(x)$ equals the z-th power of the distance from x to its closest center. For subspace embeddings, q is a vector $\mathbf{v} \in \mathbb{R}^d$ and $q(x) = |\mathbf{x}^\top \mathbf{v}|^p$, so that $\operatorname{cost}_q(Y) = \|\mathbf{Y}\mathbf{v}\|_p^p$ for the matrix $\mathbf{Y}$ formed by rows of Y. For projection-cost preservation (PCP), q is a projection matrix $\mathbf{P}$, and the cost is the residual Frobenius norm $\|\mathbf{Y} - \mathbf{Y}\mathbf{P}\|_F^2$.

C Frameworks for Space-Optimal Streaming Algorithms

In this section, we formally describe our framework in Figure 3, which is applied in the following sections to achieve space-efficient streaming algorithms for (k, z)-clustering, subspace embedding, and projection-cost preservation. To get rid of the $\log(n)$ bits to store each sampled item, we introduce the following efficient encoding scheme.

Definition C.1 (Efficient encoding scheme)**.** *Given a dataset X and a family of query functions $\mathcal{Q}$, an efficient encoding scheme stores a space-efficient structured representation $C(X)$ such that one can recover a $(1+\varepsilon)$-coreset X' of X from $C(X)$.*

The encoding $C(X)$ can be viewed as a rounded representation of the location of the points in (k, z)-clustering or a low-dimensional matrix from which a $(1+\varepsilon)$-coreset can be reconstructed efficiently. In addition, as discussed earlier, storing each active coreset separately with the above efficient encoding scheme loses extra $\operatorname{polylog}_{\mathcal{Q},d,\varepsilon}(\log(n))$ factors. To avoid this, we store the active coresets in the tree using a global efficient encoding scheme, as defined below.

Definition C.2 (Global efficient encoding)**.** *Given a stream $\mathcal{S} = \mathcal{S}_1 \circ \mathcal{S}_2 \circ, \ldots, \circ \mathcal{S}_m$, a global efficient encoding scheme stores a sketch $C(\mathcal{S}_i)$ for each $\mathcal{S}_i$ such that one can recover a $(1+\varepsilon)$-coreset $\mathcal{S}'$ of $\mathcal{S}$ from $\cup_i C(\mathcal{S}_i)$.*

The global encoding no longer gives a $(1+\varepsilon)$-coreset for each node in the tree by reconstruction, however, it gives a $(1+\varepsilon)$-coreset for the entire stream, which is sufficient for our purpose.

We assembly all the subroutines in Algorithm 4. Given a stream $\mathcal{S}$ of length n, we use the online sampling scheme to generate a stream $\mathcal{S}'$ and pass it on to a merge-and-reduce tree to produce a coreset $\mathcal{C}$, which we store using global efficient encoding. Upon each new arrival, we reconstruct the coreset from the global encoding, which gives an accurate estimate of the online sensitivity and suffices to define the online sampling probability.

We provide guarantees of Algorithm 4 in Framework C.3.

Framework C.3. *Given a stream $\mathcal{S}$ of length n, a family of query functions $\mathcal{Q}$, and an accuracy parameter ε, suppose that we have access to the following subroutines:*

- *An online sampling scheme that returns a $(1+\varepsilon)$-coreset of size $\mathcal{S}_{ol}(\mathcal{Q}, n, d, \varepsilon, \delta)$ with probability $1-\delta$.*
- *An (offline) algorithm that merges two coresets of size $\mathcal{S}_{off}(\mathcal{Q}, n, d, \varepsilon, \delta)$ to obtain a $(1+\varepsilon)$-coreset of size $\mathcal{S}_{off}(\mathcal{Q}, n, d, \varepsilon, \delta)$ with probability $1-\delta$.*
- *A global efficient encoding scheme that encodes a stream with m partitions $\mathcal{K} = \mathcal{K}_1 \circ \ldots \circ \mathcal{K}_m$, where each part $\mathcal{K}_i$ has size at most $\mathcal{S}_{off}(\mathcal{Q}, n, d, \varepsilon, \delta)$, using $\mathcal{S}_{gl\text{-}encode}(\mathcal{Q}, n, d, \varepsilon, \delta) + m \cdot \mathcal{S}_{loc\text{-}encode}(\mathcal{Q}, n, d, \varepsilon, \delta)$ bits of memory, where $\mathcal{S}_{gl\text{-}encode}(\mathcal{Q}, n, d, \varepsilon, \delta)$ denotes the space to store the global constant-factor approximation and $\mathcal{S}_{loc\text{-}encode}(\mathcal{Q}, n, d, \varepsilon, \delta)$ denotes the space to store the offset of each active coreset.*

Then there is an algorithm that returns a $(1+\varepsilon)$-coreset at all times with probability $1-\delta$. The algorithm uses

$$\mathcal{S}_{gl\text{-}encode}\left(\mathcal{Q}, n, d, \frac{\varepsilon}{h}, \frac{\delta}{\ell}\right) + h \cdot \mathcal{S}_{loc\text{-}encode}\left(\mathcal{Q}, n, d, \frac{\varepsilon}{h}, \frac{\delta}{\ell}\right)$$

Algorithm 4 Space-Optimal Streaming Framework

Require: Stream $\mathcal{S}$ of n objects in $\mathbb{R}^d$ (points, rows, or vectors), target query function family $\mathcal{Q}$ defining $\mathrm{cost}_{\mathcal{Q}}(\cdot)$, accuracy parameter $\varepsilon \in (0,1)$, failure probability δ, budget B for the merge-and-reduce coreset

Ensure: $(1+\varepsilon)$-coreset of $\mathcal{S}$

for each element $x_t \in \mathcal{S}$ **do**
  Reconstruct the current coreset from the global encoding
  Sample x_t with the online sampling scheme and pass on a sub-stream $\mathcal{S}'$. {Compute the sampling probability using the reconstructed coreset}
  if x_t is sampled to $\mathcal{S}'$ **then**
    Input x_t to the on-going leaf node in the merge-and-reduce tree
    Store x_t with high precision, (e.g., $\mathcal{O}(\log n)$ bits for each entry)
    if the on-going leaf node has size B **then**
      Let $\mathcal{C}_1, \ldots, \mathcal{C}_\ell$ be the active coresets. Reconstruct $\mathcal{C}_1, \ldots, \mathcal{C}_\ell$ from the global encoding {$\mathcal{C}_1, \ldots, \mathcal{C}_\ell$ have not been merged to a parent node. $\bigcup_{i\in\ell} \mathcal{C}_i$ is a coreset of the previous stream}
      Let $\mathcal{C}_{\ell+1}$ be the coreset on the on-going leaf node, merge $\mathcal{C}_{\ell+1}$ with $\mathcal{C}_1, \ldots, \mathcal{C}_\ell$ {Merge two child nodes using the offline algorithm}
      Store the resulted coresets with global efficient encoding
    end if
  end if
end for

return the items in the active coresets

bits of memory, where $\ell = \frac{\mathcal{S}_{ol}(\mathcal{Q},n,d,\varepsilon,\delta)}{\mathcal{S}_{off}(\mathcal{Q},n,d,\varepsilon,\delta)}$ *is the number of leaf nodes in the merge-and-reduce tree and* $h = \mathcal{O}(\log \ell)$ *is the height. Here, we assume that the space complexities scale at most polynomially with* ε *and* δ*, and so we omit the constant terms.*

Proof. We analyze the correctness and the complexity of Algorithm 4. Let $B = \mathcal{S}_{\text{off}}(\mathcal{Q}, n, d, \varepsilon, \delta)$ be the budget for the merge-and-reduce coreset. First, we apply the online sampling scheme which succeeds upon each arrival with probability $1 - \frac{\delta}{\mathrm{poly}(n)}$ so that we can union bound across the entire stream. Note that we reconstruct the coreset from the global encoding scheme at each step, which ensures the validity of the online sampling probability.

Then we implement the merge-and-reduce method on the online sampled sub-stream with coreset size $\mathcal{S}_{\text{off}}(\mathcal{Q}, n, d, \varepsilon, \delta)$. Since the online sampling scheme returns a stream of size $\mathcal{S}_{\text{ol}}(\mathcal{Q}, n, d, \varepsilon, \delta)$ (which we do not store), the number of leaf nodes is at most $\ell = \frac{\mathcal{S}_{\text{ol}}(\mathcal{Q},n,d,\varepsilon,\delta)}{\mathcal{S}_{\text{off}}(\mathcal{Q},n,d,\varepsilon,\delta)}$ and the height of the merge-and-reduce tree is $h = \mathcal{O}(\log \ell)$. Since our definition of the coreset attains mergeability, i.e., if A_1 is a $(1+\varepsilon)$-coreset of B_1 and A_2 is a $(1+\varepsilon)$-coreset of B_2, then $A_1 \circ A_2$ is a $(1+\varepsilon)$-coreset of $B_1 \circ B_2$, and transitivity, i.e., if A is a $(1+\varepsilon)$-coreset of B and B is a $(1+\varepsilon)$-coreset of C, then A is a $(1+\varepsilon)^2$-coreset of C (see Definition B.4). Thus, it suffices to apply the offline algorithm to merge the coresets with parameters $\varepsilon' = \theta\left(\frac{\varepsilon}{h}\right)$ and $\delta' = \theta\left(\frac{\delta}{\ell}\right)$. Then the total distortion is at most $(1+\varepsilon')^h = 1 + \mathcal{O}(\varepsilon' h) = 1 + \varepsilon$, and the failure probability can be union bounded since there are at most $\mathcal{O}(\ell)$ times of merging coresets. This gives the correctness of our algorithm.

Next, we compute the bits of memory storage. Note that when we apply the global efficient encoding scheme, there are at most h coresets to encode (which is the number of coresets that are not merged to a parent node in the tree) and each of size $\mathcal{S}_{\text{off}}\left(\mathcal{Q}, n, d, \frac{\varepsilon}{h}, \frac{\delta}{\ell}\right)$. Therefore, the total number of bits we store is

$$\mathcal{S}_{\text{gl-encode}}\left(\mathcal{Q}, n, d, \frac{\varepsilon}{h}, \frac{\delta}{\ell}\right) + h \cdot \mathcal{S}_{\text{loc-encode}}\left(\mathcal{Q}, n, d, \frac{\varepsilon}{h}, \frac{\delta}{\ell}\right).$$

□

We remark that in the high-dimensional optimization problems we study, the state-of-the-art online algorithms and their offline counterparts are often off by $\mathrm{polylog}_{\mathcal{Q},d,\varepsilon}(n)$, and so in the above formula,

$h = \log \frac{\mathcal{S}_{\text{ol}}(\mathcal{Q},n,d,\varepsilon,\delta)}{\mathcal{S}_{\text{off}}(\mathcal{Q},n,d,\varepsilon,\delta)} = \text{polylog}_{\mathcal{Q},d,\varepsilon}(\log(n))$. Recall that $\mathcal{S}_{\text{gl-encode}}\left(\mathcal{Q}, n, d, \frac{\varepsilon}{h}, \frac{\delta}{\ell}\right)$ is the space to store the global constant-factor approximation, which is often tight in words of space compared to the offline setting. Moreover, $\mathcal{S}_{\text{loc-encode}}\left(\mathcal{Q}, n, d, \frac{\varepsilon}{h}, \frac{\delta}{\ell}\right)$ is the space to store the offset of each active coreset, which only loses $\text{polylog}_{\mathcal{Q},d,\varepsilon}(\log(n))$ factors by our efficient quantization methods. Therefore, the second term in our space complexity only loses $\text{polylog}_{\mathcal{Q},d,\varepsilon}(\log(n))$ factors, which becomes a lower-order term, i.e., $o(1)$, in words of space as we assume each word stores $\log(n)$ bits. Thus, our framework gives $\mathcal{O}_{\mathcal{Q},d,\varepsilon}(1)$ words of space as we desire.

D (k, z)-Clustering

In this section, we apply our framework in Appendix B to achieve optimal $\mathcal{O}_{k,d,\varepsilon}(1)$ words of space. Particularly, we provide the global efficient encoding for coresets for (k, z)-clustering and show its usage in the framework.

We begin by reviewing the formal definition of a strong coreset for (k, z)-clustering.

Definition D.1 (Coreset)**.** *Given a set $X = \{x_1, \ldots, x_n\}$ of points in $\mathbb{R}^d$, a distance function* dist*, and an accuracy parameter $\varepsilon > 0$, a* coreset *for (k, z) clustering is a weighted subset S such that for any set $C \subset \mathbb{R}^d$ of k points, we have*

$$(1-\varepsilon)\sum_{t=1}^{n} \text{dist}(x_t, C)^z \le \sum_{q\in S} w(q)\,\text{dist}(q, S)^z \le (1+\varepsilon)\sum_{t=1}^{n} \text{dist}(x_t, C)^z,$$

where w is the weight function.

We next define the (offline) sensitivity for (k, z)-clustering.

Definition D.2 (Sensitivities for (k, z)-clustering)**.** *The* sensitivity *of a point $x \in X$ for (k, z)-clustering in a metric space with metric* dist *is*

$$\max_{C:|C|\le k} \frac{\text{cost}(x, C)}{\text{cost}(X, C)} = \max_{C:|C|\le k} \frac{\text{dist}(x, C)^z}{\sum_{x\in X} \text{dist}(x, C)^z}.$$

We use the following (offline) algorithm for (k, z)-clustering:

Theorem D.3. *[CLSS22, HLW23, CLS+22] Given an accuracy parameter $\varepsilon \in (0, 1)$, there exists an algorithm that samples $\tilde{\mathcal{O}}\left(\min\left(\frac{1}{\varepsilon^2}\cdot k^{2-\frac{z}{z+2}}, \frac{1}{\min(\varepsilon^4,\varepsilon^{2+z})}\cdot k\right)\right)$ points and constructs a $(1+\varepsilon)$-coreset.*

We next define the online sensitivity for (k, z)-clustering.

Definition D.4 (Online sensitivity for (k, z)-clustering)**.** *Given a sequence of points $x_1, \ldots, x_n$, the* online sensitivity *of x_t for (k, z)-clustering is*

$$\max_{C:|C|\le k} \frac{\text{cost}(x_t, C)}{\text{cost}(X_t, C)} = \max_{C:|C|\le k} \frac{\text{dist}(x_t, C)^z}{\sum_{i=1}^{t} \text{dist}(x_i, C)^z},$$

where $X_t := \{x_1, \ldots, x_t\}$ is the prefix dataset at time t.

We next recall the following upper bound on the sum of the online sensitivities, which is used to bound the number of samples.

Theorem D.5. *[CWZ23] Given a sequence of points $x_1, \ldots, x_n \subset [\Delta]^d$ and let $\sigma(x_t)$ denote the online sensitivity of x_t for each $t \in [n]$ for (k, z)-clustering, where $z \ge 1$, then we have*

$$\sum_{t=1}^{n} \sigma(x_t) = \mathcal{O}\left(2^{2z} k \log^2(nd\Delta)\right).$$

We next state the formal guarantees of online sensitivity sampling.

Theorem D.6 (Online sensitivity sampling)**.** *[CWZ23] Given a sequence of points $x_1, \ldots, x_n \subset [\Delta]^d$, we sample each point x_t with probability $p_t \ge \min(1, \gamma \cdot \sigma(x_t))$, where σ_t is the online sensitivity of x_t and $\gamma = \mathcal{O}\left(\frac{dk}{\varepsilon^2}\log\frac{n\Delta}{\varepsilon}\right)$ and weight x_t by $\frac{1}{p_t}$ if sampled. Then with high probability, we obtain a weighted subset that is a $(1+\varepsilon)$-coreset for (k, z)-clustering containing at most $\mathcal{O}\left(\frac{dk^2}{\varepsilon^2}\log^3\frac{n\Delta}{\varepsilon}\right)$ points.*

D.1 Efficient Encoding for Coreset Construction for (k, z)-Clustering

We next analyze our efficient encoding algorithm for a given coreset. We first restate the algorithm in Algorithm 1 for completeness.

Algorithm 5 Efficient Encoding for Coreset Construction for (k, z)-Clustering, Restatement of Algorithm 1

Require: Data set $X \subset [\Delta]^d$ with weight $w(\cdot)$, accuracy parameter $\varepsilon \in (0, 1)$, number of clusters k, parameter $z \geq 1$, failure probability $\delta \in (0, 1)$
Ensure: $(1 + \varepsilon)$-coreset for (k, z)-clustering

$\varepsilon' \leftarrow \frac{\mathrm{poly}(\varepsilon^z)}{\mathrm{poly}(k \log(nd\Delta))}$
Find a set C' of k centers that is a constant-factor approximation to (k, z)-clustering on X
for each $x \in X$ **do**
  Let $c'(x)$ be the closest center of C' to x
  Let y' be the offset $x - c'(x)$
  Let y be y' with coordinates rounded to a power of $(1 + \varepsilon')$
  Let $x' = (c'(x), y)$, storing the *exponent* for each coordinate of y
  $X' \leftarrow X' \cup \{x'\}$
end for

return (C', X')

To show correctness of our encoding, we first recall the following fact.

Fact D.7 (Claim 5 in [SW18])**.** *Suppose $z \geq 1$, $x, y \geq 0$, and $\varepsilon \in (0, 1]$. Then*

$$(x + y)^z \leq (1 + \varepsilon) \cdot x^z + \left(1 + \frac{2z}{\varepsilon}\right)^z \cdot y^z.$$

Given a constant-factor approximation C', we now show that the rounding X' of the dataset X by representing each point as a rounded offset from its closest center in C' is a strong coreset of X.

Lemma D.8. *Let $\varepsilon \in \left(0, \frac{1}{2}\right)$ and let X' be the weighted dataset S defined by their offsets from the set C' of centers from Algorithm 5. Then for all $C \subset [\Delta]^d$ with $|C| \leq k$,*

$$(1 - \varepsilon) \cdot \mathrm{cost}(C, X) \leq \mathrm{cost}(C, X') \leq (1 + \varepsilon) \cdot \mathrm{cost}(C, X).$$

Proof. Let C be any fixed set of at most k centers. We abuse notation so that for each $x \in X$, we use x' to denote the corresponding point $c(x) + y$, where $y' = x - c'(x)$ is the offset, and y is y' with its coordinates rounded to a power of $(1 + \varepsilon')$. Similarly, we use c' to denote $c(x) \in C'$. Thus, we have $\|x - x'\|_2 \leq \varepsilon' \cdot \|c' - x\|_2$ for all $x \in X$.

For ease of presentation, we first write X' so that the weights are not rounded to a power of $(1 + \varepsilon')$. By the triangle inequality,

$$\mathrm{cost}(C, X') = \sum_{x' \in X'} (\mathrm{dist}(x', C))^z \leq \sum_{x' \in X'} (\mathrm{dist}(x', x) + \mathrm{dist}(x, C))^z.$$

Since $\mathrm{dist}(x, x') = \|x - x'\|_2 \leq \varepsilon' \cdot \|c' - x\|_2 = \varepsilon' \cdot \mathrm{dist}(x, C')$, then

$$\mathrm{cost}(C, X') \leq \sum_{x' \in X'} \left(\varepsilon' \cdot \mathrm{dist}(x, C') + \mathrm{dist}(x, C)\right)^z.$$

By Fact D.7,

$$\mathrm{cost}(C, X') \leq \sum_{x' \in X'} \left(\left(1 + \frac{\varepsilon}{2}\right) \cdot (\mathrm{dist}(x, C))^z + \left(1 + \frac{4z}{\varepsilon}\right)^z \cdot (\varepsilon' \cdot \mathrm{dist}(x, C'))^z\right).$$

Thus,

$$\mathrm{cost}(C, X') \leq \left(1 + \frac{\varepsilon}{2}\right) \cdot \mathrm{cost}(C, X) + \left(1 + \frac{4z}{\varepsilon}\right)^z (\varepsilon')^z \cdot \mathrm{cost}(C', X).$$

Since C' is a constant-factor approximation to the optimal (k,z)-clustering of X, then there exists a constant $\gamma \geq 1$ such that $\text{cost}(C, X') \leq \gamma \cdot \text{cost}(C, X)$. Hence,

$$\text{cost}(C, X') \leq \left(\left(1 + \frac{\varepsilon}{2}\right) + \left(1 + \frac{4z}{\varepsilon}\right)^z (\varepsilon')^z \cdot \gamma\right) \cdot \text{cost}(C, X).$$

Then for $\varepsilon' = \frac{\text{poly}(\varepsilon^z)}{\text{poly}(k, \log(nd\Delta))}$, we have that

$$\text{cost}(C, X') \leq \left(1 + \frac{2}{3}\varepsilon\right) \cdot \text{cost}(C, X),$$

which almost gives the right hand side of the inequality (due to the notation X' representing the output prior to the rounding of the weights).

Similarly, by triangle inequality and $\text{dist}(x, x') \leq \varepsilon' \cdot \text{dist}(x, C')$,

$$\begin{aligned}
\text{cost}(C, X) &= \sum_{x \in X} (\text{dist}(x, C))^z \\
&\leq \sum_{x \in X} (\text{dist}(x, x') + \text{dist}(x', C))^z \\
&\leq \sum_{x' \in X'} \left(\varepsilon' \cdot \text{dist}(x, C') + \text{dist}(x', C)\right)^z.
\end{aligned}$$

By Fact D.7,

$$\begin{aligned}
\text{cost}(C, X) &\leq \left(1 + \frac{\varepsilon}{2}\right) \cdot \text{cost}(C, X') + \left(1 + \frac{4z}{\varepsilon}\right)^z (\varepsilon')^z \cdot \text{cost}(C', X) \\
&\leq \left(1 + \frac{\varepsilon}{2}\right) \cdot \text{cost}(C, X') + \left(1 + \frac{4z}{\varepsilon}\right)^z (\varepsilon')^z \cdot \gamma \cdot \text{cost}(C, X).
\end{aligned}$$

Then for $\varepsilon' = \frac{\text{poly}(\varepsilon^z)}{\text{poly}(k, \log(nd\Delta))}$, we have that

$$\text{cost}(C, X) \leq \left(1 + \frac{\varepsilon}{2}\right) \cdot \text{cost}(C, X') + \frac{\varepsilon}{100} \cdot \text{cost}(C, X),$$

so that $\left(1 - \frac{2}{3} \cdot \varepsilon\right) \text{cost}(C, X) \leq \text{cost}(C, X')$, which almost gives the left-hand side of the inequality (due to the notation X' representing the output prior to the rounding of the weights).

Finally, since the weights are rounded to a power of $(1+\varepsilon')$, then the cost can only change by $(1+\varepsilon')$. Since $\varepsilon' = \frac{\text{poly}(\varepsilon^z)}{\text{poly}(k, \log(nd\Delta))}$, then it follows that

$$(1 - \varepsilon) \cdot \text{cost}(C, X) \leq \text{cost}(C, X') \leq (1 + \varepsilon) \cdot \text{cost}(C, X).$$

□

We now give the full guarantees of Algorithm 5, which gives an efficient encoding of an input set X.

Lemma D.9. *Let X be a coreset construction with weights bounded by $[1, \text{poly}(nd\Delta)]$. Then X' is a $(1+\varepsilon)$-coreset for X that uses $\mathcal{O}\left(dk \log(n\Delta)\right) + |X| \cdot \text{polylog}\left(k, \frac{1}{\varepsilon}, \log(nd\Delta), \log\frac{1}{\delta}\right)$ bits of space.*

Proof. Algorithm 5 then outputs a set (C', X') that encodes X' using C'. By Lemma D.8, we have that for all $C \subset [\Delta]^d$ with $|C| \leq k$,

$$(1 - \varepsilon) \cdot \text{cost}(C, X) \leq \text{cost}(C, X') \leq (1 + \varepsilon) \cdot \text{cost}(C, X).$$

Thus, X' is a strong coreset for X.

It remains to analyze the space complexity of (C', X'). We first require C' to encode S. Since C' is a set of k centers, then C' can be represented using $\mathcal{O}\left(dk \log(n\Delta)\right)$ bits of space. Additionally, each point x' in X' is encoded using $\mathcal{O}\left(d \log\left(\frac{1}{\varepsilon} + \log n + \log \Delta\right)\right)$ bits of space due to storing the d coordinates of the rounded offset y. In particular, we do not store the explicit coordinates, but rather their exponents. Moreover, we use $\log k$ bits to store the closest center $c'(x)$. Finally, each weight has weight $\text{poly}(nd\Delta)$ and thus can be approximated to $(1+\varepsilon)$-multiplicative factor by storing the exponent, using $\mathcal{O}\left(\log\log(nd\Delta)\right)$ bits of space. Therefore, the output of the algorithm is encoded using $\mathcal{O}\left(dk \log(n\Delta)\right) + |X| \cdot \text{polylog}\left(k, \frac{1}{\varepsilon}, \log(nd\Delta), \log\frac{1}{\delta}\right)$ total bits of space. □

D.2 Space-Efficient Clustering in the Streaming Model

In this section, we show how to obtain a global encoding of the coresets inspired by the efficient encoding from the previous section. Then, we implement our framework in Appendix B and achieve our main algorithm for space-efficient (k, z)-clustering on insertion-only data streams.

The main shortcoming of immediately applying the previous efficient encoding to each node of a merge-and-reduce tree is that the constant-factor approximation requires $\mathcal{O}(kd \log(n\Delta))$ bits of storage per efficient encoding, and there can be $\text{polylog}(\log(n\Delta))$ such efficient encodings due to the height of the merge-and-reduce tree on a data stream produced by online sensitivity sampling. Instead, we provide a single constant-factor approximation to the global dataset and show that the error of maintaining such a global constant-factor approximation does not compound too much over the course of the stream.

The algorithm appears in full in Algorithm 6, which implements our space-efficient framework in Algorithm 4 using the global encoding scheme.

Algorithm 6 (k, z)-Clustering on Insertion-Only Streams

Require: Data set $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ that arrives as a data stream, $\varepsilon \in (0, 1)$, number of clusters k, parameter $z \geq 1$
Ensure: $(1 + \varepsilon)$-coreset for (k, z)-clustering

$Z \leftarrow \emptyset$, $\lambda \leftarrow \mathcal{O}\left(\frac{k}{\varepsilon^2} \cdot \log k \log n\right)$
for each $t \in [n]$ **do**
  Let $\widehat{x_t}$ be the image of x_t after applying a JL transform into $\mathcal{O}(\log n)$ dimensions
  Use $Z \cup \{\widehat{x_t}\}$ to compute a $\mathcal{O}(1)$-approximation $\widehat{\sigma(x_t)}$ to the online sensitivity of x_t
  $p(x_t) \leftarrow \min(1, \lambda \cdot \widehat{\sigma(x_t)})$
  With probability $p(x_t)$, sample x_t into a stream $\mathcal{S}'$ with weight $\frac{1}{p(x_t)}$
  Let Z be the running output of merge-and-reduce on $\mathcal{S}'$ using the efficient encoding for coreset construction in Algorithm 5 with a *global* constant-factor approximation, for accuracy $\frac{\varepsilon}{\text{poly}(\log \log n\Delta)}$ and failure probability $\frac{1}{\text{poly}\left(\frac{1}{\varepsilon}, \log(n\Delta)\right)}$ {Detailed implementation of Algorithm 4}
end for

return Z

We first show that given a partition $X_1 \sqcup \ldots \sqcup X_m$ of X, we can apply Algorithm 5 and although the output X_i' from our efficient encoding will no longer necessarily be a strong coreset for each corresponding X_i, the resulting error will only be an additive $\varepsilon' \cdot \text{cost}(X, C)$, which suffices for our purposes. Specifically, we will have $\varepsilon' = \frac{\varepsilon}{\text{poly}(\log \log(nd\Delta))}$ and ultimately have $m = \text{poly}(\log \log(nd\Delta))$.

Lemma D.10. *Let $X_1 \sqcup \ldots \sqcup X_m = X$ be a partition of X for $m = \text{poly}(\log \log(nd\Delta))$. Let C' be a set of k centers that is a set C' of k centers that is a constant-factor approximation to (k, z)-clustering on X. For each $i \in [m]$, let X_i' be the set of rounded points corresponding to X_i from Algorithm 5. Then for all $i \in [m]$ and all $C \subset [\Delta]^d$ with $|C| \leq k$,*

$$|\,\text{cost}(X_i', C) - \text{cost}(X_i, C)| \leq \frac{\varepsilon}{\text{poly}(\log \log nd\Delta)} \cdot \text{cost}(X, C).$$

Proof. Note that by Lemma D.8 with accuracy $\frac{\varepsilon}{\text{poly}(\log \log nd\Delta)}$, we have

$$|\,\text{cost}(X', C) - \text{cost}(X, C)| \leq \frac{\varepsilon}{\text{poly}(\log \log nd\Delta)} \cdot \text{cost}(X, C).$$

Since

$$|\,\text{cost}(X_i', C) - \text{cost}(X_i, C)| \leq \sum_{i=1}^{m} |\,\text{cost}(X_i', C) - \text{cost}(X_i, C)| \leq \varepsilon \cdot \text{cost}(X, C),$$

then the claim follows. □

It remains to give the full guarantees of Algorithm 6 by showing correctness of the global encoding and in particular, that there is no compounding error across the $\mathrm{poly}(\log\log(nd\Delta))$ iterations.

Theorem D.11. *Given a set X of n points on $[\Delta]^d$, let $f\left(n, d, \Delta, k, \frac{1}{\varepsilon}, z\right)$ be the number of points of a coreset construction with weights $[1, \mathrm{poly}(nd\Delta)]$ for (k, z)-clustering. Then Algorithm 6 outputs a $(1+\varepsilon)$-coreset of X and uses $\mathcal{O}\left(dk\log(n\Delta)\right) + f\left(n, d, \Delta, k, \frac{\mathrm{polylog}(\log(nd\Delta))}{\varepsilon}, z\right) \cdot \mathrm{polylog}\left(\frac{1}{\varepsilon}, \log(nd\Delta)\right)$ bits of space.*

Proof. We first show that with high probability, the following invariants are satisfied at each time:

(1) $\widehat{\sigma(x_1)}, \ldots, \widehat{\sigma(x_t)}$ are constant-factor approximations to the online sensitivities $\sigma(x_1), \ldots, \sigma(x_t)$ with respect to (k, z)-clustering

(2) At most $\mathrm{poly}\left(k, \frac{1}{\varepsilon}, \log(nd\Delta)\right)$ points have been sampled into $\mathcal{S}'$

(3) Z at time t is a $(1+\varepsilon)^2$-coreset to $X_t := \{x_1, \ldots x_t\}$

We prove correctness by induction on t. The online sensitivity of the first point is 1 and it will be correctly computed by the data structure. Thus we have that after the first step, we have $\mathcal{S}' = Z = \{x_1\}$ and the base case is complete.

Let $\mathcal{E}_t$ be the event that the invariants hold at time t. Conditioning on $\mathcal{E}_{t-1}$ and in particular the correctness of Z at time $t-1$ being a $(1+\varepsilon)$-coreset of $X_{t-1} := \{x_1, \ldots, x_{t-1}\}$, we have that $Z \cup \{x_t\}$ is a strong coreset of $X_t := \{x_1, \ldots, x_t\}$. Thus we can use $Z \cup \{x_t\}$ to compute a constant-factor approximation to the online sensitivity $\sigma(x_t)$ of x_t. By Theorem B.2, it follows that with high probability, we can instead use $Z \cup \{\widehat{x_t}\}$ to compute a constant-factor approximation $\widehat{\sigma(x_t)}$ to the online sensitivity $\sigma(x_t)$. Then by Theorem D.6, $\mathcal{S}'$ after time t will be a $(1+\varepsilon)$-coreset for X_t with high probability.

By Theorem D.6 through Theorem D.5, we have that by sampling points with probabilities that are constant-factor approximations to their online sensitivities, then the total number of sampled points into $\mathcal{S}'$ is $\mathcal{O}\left(\frac{dk^2}{\varepsilon^2}\log^3\frac{n\Delta}{\varepsilon}\right)$ with high probability. Thus, $\mathbf{Pr}\left[\mathcal{E}_t\right] \geq 1 - \frac{1}{\mathrm{poly}(n)}$.

Observe that merge-and-reduce will be correct if $\mathcal{S}'$ is a stream with length at most $k \cdot \mathrm{poly}\left(\frac{1}{\varepsilon}, \log(nd\Delta)\right)$, since we set the failure probability to be $\frac{1}{\mathrm{poly}\left(\frac{1}{\varepsilon}, \log(n\Delta)\right)}$ and merge-and-reduce will consider at least k stream updates before applying a new coreset construction. Thus conditioned on (1) the correctness of $\mathcal{S}'$ after time t being a $(1+\varepsilon)$-coreset for X_t and (2) the correctness of merge-and-reduce on $\mathcal{S}'$, we have that Z after time t will be a $(1+\varepsilon)$-coreset for $\mathcal{S}'$, and thus a $(1+\varepsilon)^2$-strong coreset for X_t, which completes the induction. That is, $\mathbf{Pr}\left[\mathcal{E}_t \mid \mathcal{E}_1, \ldots, \mathcal{E}_{t-1}\right] \geq 1 - \frac{1}{\mathrm{poly}(n)}$. The correctness guarantee then follows by rescaling ε and a union bound over all $t \in [n]$.

To analyze the space complexity, note that by the guarantees of the invariants, the total number of sampled points into $\mathcal{S}'$ is $\mathcal{O}\left(\frac{dk^2}{\varepsilon^2}\log\frac{nd\Delta}{\varepsilon}\right)$. However, $\mathcal{S}'$ is not maintained explicitly, but instead given as input to the merge-and-reduce subroutine, which uses the efficient encoding for coreset construction given in Algorithm 5 with accuracy $\frac{\varepsilon}{\mathrm{poly}(\log\log(nd\Delta))}$ and failure probability $\frac{1}{\mathrm{poly}\left(\frac{1}{\varepsilon}, \log(nd\Delta)\right)}$. Thus by Lemma D.9, the total representation of Z uses $\mathcal{O}\left(dk\log(n\Delta)\right) + \frac{dk^2}{\varepsilon^2} \cdot 2^{2z} \cdot \mathrm{polylog}\left(\frac{1}{\varepsilon}, \log(n\Delta)\right)$ bits of space. □

E Subspace Embeddings

In this section, we provide our one-pass streaming algorithm on insertion-only streams for L_p subspace embeddings using $\mathcal{O}_{k,d,\varepsilon}(1)$ words of space. Recall that in te problem of constructing L_p subspace embeddings, the goal is to produce a matrix $\mathbf{M} \in \mathbb{R}^{m \times d}$ such that for an accuracy parameter $\varepsilon \in (0, 1)$ and all vectors $\mathbf{x} \in \mathbb{R}^d$, we have

$$(1-\varepsilon)\|\mathbf{A}\mathbf{x}\|_p \leq \|\mathbf{M}\mathbf{x}\|_p \leq (1+\varepsilon)\|\mathbf{A}\mathbf{x}\|_p.$$

L_p subspace embeddings serve as a foundational tool for data compression by maintaining the essential geometric structure of high-dimensional datasets. This preservation is vital for various downstream tasks, including dimensionality reduction, compressed sensing, and linear regressions. Hence, there is a long line of work studying L_p subspace embeddings [DMM06a, DMM06b, Sar06, DDH+09, CW09, CP15, CMP20, BDM+20, PPP21, MWZ22, MMM+22, MMWY22, CSWZ23, MMM+23, WY23].

We first define the leverage scores, which describes the importance of a row for L_2 subspace embeddings.

Definition E.1 (Leverage scores)**.** *The* leverage score *of a row* $\mathbf{a}_i \in \mathbb{R}^d$ *of a matrix* $\mathbf{A} \in \mathbb{R}^{n\times d}$ *is*

$$\max_{\mathbf{x}:\|\mathbf{A}\mathbf{x}\|_2=1} \langle \mathbf{a}, \mathbf{x}\rangle^2 = \mathbf{a}_i^\top (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{a}_i.$$

For the purpose of constructing L_p subspace embedding, [CP15] generalized the above definition to L_p Lewis weights, defined as follows:

Definition E.2 (L_p Lewis weights)**.** *The* Lewis weight *of a row* $\mathbf{a}_i \in \mathbb{R}^d$ *of a matrix* $\mathbf{A} \in \mathbb{R}^{n\times d}$ *is the* i*-th diagonal entry of a diagonal weight matrix* $\mathbf{W}$ *such that for all* $i \in [n]$

$$W_{i,i} = \ell_i(\mathbf{W}^{\frac{1}{2}-\frac{1}{p}}\mathbf{A}),$$

where $\ell_i(\mathbf{W}^{\frac{1}{2}-\frac{1}{p}}\mathbf{A})$ *denotes the leverage score of the* i*-th row of the matrix* $\mathbf{W}^{\frac{1}{2}-\frac{1}{p}}\mathbf{A}$.

L_p Lewis weights may be viewed as leverage scores computed after a density reweighting that balances the rows so that preserving $\|\mathbf{A}\mathbf{x}\|_p$ reduces to preserving the Euclidean norm $\|\mathbf{A}'\mathbf{x}\|_2$ for a suitably rescaled matrix $\mathbf{A}'$ [CP15]. Compared with alternative row-importance measures such as L_p leverage scores [DDH+09] or L_p sensitivities [BDM+20, CD21], Lewis weights achieve optimal row-sampling complexity for L_p embeddings. For the special case $p = 2$, it is known that oblivious subspace embeddings with $\mathcal{O}\left(\frac{d}{\varepsilon^2}\right)$ rows exist [Sar06]. By combining these methods,we have the following guarantee.

Theorem E.3. *[Sar06, CP15, WY23] Let* $\mathbf{A} \in \mathbb{R}^{n\times d}$, $\varepsilon \in (0,1)$, *and* $p \geq 1$. *Define*

$$f\left(d, \frac{1}{\varepsilon}, p\right) = \begin{cases} \frac{d}{\varepsilon^2}\log\frac{d}{\varepsilon}\,\mathrm{polylog}\left(\frac{d}{\varepsilon}\right), & p \in [1,2), \\ \mathcal{O}\left(\frac{d}{\varepsilon^2}\right), & p = 2, \\ \frac{d^{p/2}}{\varepsilon^2}\cdot\mathrm{polylog}\left(\frac{d}{\varepsilon}\right), & p > 2. \end{cases}$$

There exists a coreset construction for L_p *subspace embedding consisting of* $f\left(d, \frac{1}{\varepsilon}, p\right)$ *rows, where each entry is bounded by* $\|\mathbf{A}\|_\infty \cdot \mathrm{poly}(n)$.

We now extend the notion of Lewis weights to the online setting.

Definition E.4 (Online L_p Lewis weights)**.** *For each* $t \in [n]$, *the online* L_p Lewis weight *of a row* $\mathbf{a}_t \in \mathbb{R}^d$ *of a matrix* $\mathbf{A} \in \mathbb{R}^{n\times d}$ *is the Lewis weight of* $\mathbf{a}_i$ *with respect to the prefix matrix* $\mathbf{A}_t = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_t$, *i.e., the submatrix of* $\mathbf{A}$ *consisting of the first* t *rows of* $\mathbf{A}$.

We recall bounds on the total mass of online Lewis weights.

Theorem E.5. *[BDM+20, WY23] Let* $\mathbf{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\} \in \mathbb{R}^{n\times d}$ *and let* $\sigma(\mathbf{a}_t)$ *denote the online* L_p *Lewis weight of row* $\mathbf{a}_t$ *for* $t \in [n]$, *with* $p \geq 1$. *Then, for* $p \in [1,2]$,

$$\sum_{t=1}^{n} \sigma(\mathbf{a}_t) = d \cdot \mathrm{polylog}(n\kappa),$$

and for $p > 2$,

$$\sum_{t=1}^{n} \sigma(\mathbf{a}_t) = d^{p/2} \cdot \mathrm{polylog}(n\kappa),$$

where κ *denotes the online condition number.*

Finally, we state the guarantees provided by sampling according to online Lewis weights.

Theorem E.6 (Online Lewis weight sampling). *[WY23] Given a stream of rows $\mathbf{a}_1, \ldots, \mathbf{a}_n \in \mathbb{R}^{n\times d}$, we sample row $\mathbf{a}_t$ with probability $p_t \geq \min(1, \gamma \cdot \sigma(\mathbf{a}_t))$, where $\sigma(\mathbf{a}_t)$ is its online Lewis weight and $\gamma = \mathcal{O}\left(\frac{\log n}{\varepsilon^2}\right)$, and rescale $\mathbf{a}_t$ by a factor $\frac{1}{p_t}$ if sampled. Then, with high probability, the resulting weighted sample forms a $(1+\varepsilon)$-coreset for L_p subspace embedding whose size is at most*

$$g\left(n, d, \frac{1}{\varepsilon}, \kappa, p\right) = \begin{cases} \frac{d}{\varepsilon^2} \cdot \text{polylog}(n\kappa), & p \in [1, 2], \\ \frac{d^{p/2}}{\varepsilon^2} \cdot \text{poly}(\log^{p/2}(n\kappa)), & p > 2. \end{cases}$$

We use standard approaches, e.g., Lemma 4.1 in [CW09] to upper bound the magnitude of the logarithm of any nonzero singular value of a matrix with bounded integer entries. This in turn upper bounds the logarithm of the online condition number.

Lemma E.7. *Suppose $\mathbf{A} \in \mathbb{R}^{n\times d}$ has integer entries bounded in magnitude by M. Then for any nonzero singular value σ_i of $\mathbf{A}$, we have $|\log \sigma_i| = \mathcal{O}(d \log(nM))$.*

Proof. We have that the characteristic polynomial of $\mathbf{A}^\top\mathbf{A}$ is $p(x) = \det(x \cdot \mathbb{I}_d - \mathbf{A}^\top\mathbf{A})$. Note that if $\mathbf{A}$ has rank r, then $p(x) = x^{d-r}\prod_{i=1}^r (x - \lambda_i)$, where λ_i is the i-th eigenvalue of $\mathbf{A}^\top\mathbf{A}$, writing $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_r$. Since all entries of $\mathbf{A}^\top\mathbf{A}$ are integers, then the coefficients of $p(x)$ are also integers. Moreover, since the eigenvalues of $\mathbf{A}^\top\mathbf{A}$ are non-negative, then $\prod_{i=1}^r \lambda_i \geq 1$. Because all entries of $\mathbf{A}^\top\mathbf{A}$ are at most $\text{poly}(ndM)$ in magnitude, then

$$\lambda_i \leq \lambda_1 \leq \|\mathbf{A}^\top\mathbf{A}\|_F \leq \text{poly}(ndM).$$

Since $\prod_{i=1}^r \lambda_i \geq 1$, then we have that $\lambda_i \geq \frac{1}{\text{poly}(ndM)^d}$. Since $\sigma_i^2 = \lambda_i$, then $|\log \sigma_i| = \mathcal{O}(d \log(nM))$. □

We next recall the following definition of a well-conditioned basis.

Definition E.8 (Well-conditioned basis). *Given a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$ of rank r, let $p \in [1, \infty)$ and q be its dual norm, so that $\frac{1}{p} + \frac{1}{q} = 1$. Then matrix $\mathbf{U} \in \mathbb{R}^{n\times d}$ is an (α, β, p)-well-conditioned basis for the column space of $\mathbf{A}$ if:*

(1) The column space of $\mathbf{A}$ is the column space of $\mathbf{U}$

(2) $\sum_{i\in[n], j\in[d]} U_{i,j}^p \leq \alpha^p$

(3) For all $\mathbf{x} \in \mathbb{R}^d$, we have $\|\mathbf{z}\|_q \leq \beta\|\mathbf{U}\mathbf{z}\|_p$

One such construction of a well-conditioned basis gives the following properties:

Theorem E.9. *[DDH+09] Given a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$ of rank r, let $p \in [1, \infty)$ and q be its dual norm, so that $\frac{1}{p} + \frac{1}{q} = 1$. There exists an (α, β, p)-well-conditioned basis $\mathbf{U}$ for the column space of $\mathbf{A}$ such that:*

- *If $p < 2$, then $\alpha = r^{\frac{1}{p}+\frac{1}{2}}$ and $\beta = 1$*
- *If $p = 2$, then $\alpha = \sqrt{r}$ and $\beta = 1$*
- *If $p > 2$, then $\alpha = r^{\frac{1}{p}+\frac{1}{2}}$ and $\beta = r^{\frac{1}{q}-\frac{1}{2}}$*

Moreover, $\mathbf{U}$ can be computed in time $\mathcal{O}\left(ndr + nr^2 \log n\right)$.

For completeness, we briefly describe the construction of the well-conditioned basis of Theorem E.9, given by [DDH+09]. Given a QR decomposition $\mathbf{A} = \mathbf{Q}\mathbf{R}$, so that $\mathbf{Q}$ is any $n \times r$ matrix that is an orthonormal basis for the span of $\mathbf{A}$ and $\mathbf{R}$ is an $r \times d$ matrix, define the set $S = \{\mathbf{x} \in \mathbb{R}^d \mid \|\mathbf{Q}\mathbf{x}\|_p \leq 1\}$ and the $r \times r$ matrix $\mathbf{F}$ so that $\mathcal{E}_{LJ} = \{\mathbf{x} \in \mathbb{R}^r \mid \mathbf{x}^\top\mathbf{F}\mathbf{x} \leq 1\}$ is the Löwner-John ellipsoid of S, let $\mathbf{G} \in \mathbb{R}^{d\times r}$ be the full rank and upper triangular matrix such that $\mathbf{F} = \mathbf{G}^\top\mathbf{G}$. Then $\mathbf{U} := \mathbf{Q}\mathbf{G}^{-1}$ is the desired (α, β, p)-well-conditioned basis for the column span of $\mathbf{A}$.

We now define the L_p sensitivity of a row of a matrix, for the purposes of L_p subspace embeddings.

Definition E.10 (L_p sensitivity)**.** *For a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$, the L_p* sensitivity *of row $\mathbf{a}_t$ is the quantity*

$$\max_{\mathbf{y}\in\mathbb{R}^d:\|\mathbf{A}\mathbf{y}\|_p=1} |\langle \mathbf{a}_t, \mathbf{y}\rangle|^p .$$

Finally, we recall the following upper bound on the total L_p sensitivity for L_p subspace embedding.

Lemma E.11 (Upper bounds on the sum of the L_p sensitivities)**.** *[CP15, CD21] For a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$, let $s(\mathbf{a}_t)$ denote the L_p sensitivity of the t-th row of $\mathbf{A}$ for L_p subspace embedding. Then $\sum_{t=1}^n s(\mathbf{a}_t) \le d$ for $p \le 2$ and $\sum_{t=1}^n s(\mathbf{a}_t) \le d^{p/2}$ for $p > 2$.*

E.1 Efficient Encoding for Coreset Construction for L_p Subspace Embedding

We now analyze our efficient encoding for a given coreset for L_p subspace embeddings. We first restate the algorithm in Algorithm 2 for completeness.

Algorithm 7 Efficient Encoding for Coreset Construction for L_p Subspace Embedding, Restatement of Algorithm 2

Require: Matrix $\mathbf{A} \subset \{-M, \dots, -1, 0, 1, \dots, M\}^{n\times d}$, accuracy parameter $\varepsilon \in (0,1)$, failure probability $\delta \in (0,1)$
Ensure: $(1+\varepsilon)$-coreset for L_p subspace embedding

1: $\varepsilon' \leftarrow \frac{\varepsilon}{\mathrm{poly}(d^p, \log(nd\Delta))}$
2: Find a matrix $\mathbf{M}$ that is a constant-factor L_p subspace embedding on $\mathbf{A}$ {Theorem E.3}
3: Compute a deterministic preconditioner $\mathbf{P}$ of $\mathbf{M}$ using well-conditioned bases, such that $\mathbf{M}\mathbf{P}^{-1}$ has condition number $\mathrm{poly}(d)$
4: **for** each row $\mathbf{a} \in \mathbf{A}$ **do**
5: Let $\mathbf{b}'$ be $\mathbf{a}\mathbf{P}$ with each coordinate rounded to a power of $(1+\varepsilon')$
6: $\mathbf{B}' \leftarrow \mathbf{B}' \circ \mathbf{b}'$, storing the *exponent* for each entry of $\mathbf{b}'$
7: **end for**
8:
9: **return** $(\mathbf{M}, \mathbf{B}')$, from which estimate $\mathbf{A}'$ can be constructed

Given a constant-factor subspace embedding $\mathbf{M}$, we now show that the matrix $\mathbf{B}'$ acquired by rounding each row of $\mathbf{A}\mathbf{P}$ can be used to construct a matrix $\mathbf{A}'$ that is a strong coreset of $\mathbf{A}$.

Lemma E.12. *Let $\varepsilon \in \left(0, \frac{1}{2}\right)$ and let $\mathbf{A}' = \mathbf{B}'\mathbf{P}^{-1}$, where $\mathbf{B}'$ is the output of Algorithm 7 and $\mathbf{P}$ is the deterministic preconditioner of $\mathbf{M}$. Then for all $\mathbf{x} \in \mathbb{R}^d$,*

$$(1-\varepsilon)\|\mathbf{A}\mathbf{x}\|_p^p \le \|\mathbf{A}'\mathbf{x}\|_p \le (1+\varepsilon)\|\mathbf{A}\mathbf{x}\|_p^p.$$

Proof. Without loss of generality, we assume that $\mathbf{A}$ is full rank. Note that we can write $\mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{P}\mathbf{P}^{-1}\mathbf{x}$ and thus by rewriting $\mathbf{y} = \mathbf{P}^{-1}\mathbf{x}$, it suffices to show that for all $\mathbf{y} \in \mathbb{R}^d$

$$(1-\varepsilon)\|\mathbf{A}\mathbf{P}\mathbf{y}\|_p^p \le \|\mathbf{A}'\mathbf{P}\mathbf{y}\|_p^p \le (1+\varepsilon)\|\mathbf{A}\mathbf{P}\mathbf{y}\|_p^p.$$

We remark that if $\mathbf{A}$ is not full rank, the task would be to show claim for all $\mathbf{y}$ in the column span of $\mathbf{P}^{-1}$.

Let $\mathbf{B} = \mathbf{A}\mathbf{P}$ and observe that Algorithm 7 rounds all coordinates of each row $\mathbf{b}_t = \mathbf{a}_t\mathbf{P}$ to their closest power of $(1+\varepsilon')$. Then we have

$$|\langle \mathbf{b}_t, \mathbf{y}\rangle^p - \langle \mathbf{b}'_t, \mathbf{y}\rangle^p| \le \varepsilon' \cdot \max_{\mathbf{y}\in\mathbb{R}^d, \|\mathbf{y}\|_p=1} |\langle \mathbf{b}_t, \mathbf{y}\rangle|^p .$$

For each $t \in [n]$, let $s(\mathbf{b}_t)$ denote the sensitivity of $\mathbf{b}_t$ with respect to $\mathbf{B}$, so that

$$s(\mathbf{b}_t) = \max_{\mathbf{y}\in\mathbb{R}^d} \frac{|\langle \mathbf{b}_t, \mathbf{y}\rangle|^p}{\|\mathbf{B}\mathbf{y}\|_p^p}.$$

Recall that $\mathbf{B} = \mathbf{A}\mathbf{P}$ and $\mathbf{P}$ is a preconditioner for $\mathbf{M}$, so that $\mathbf{M}\mathbf{P}$ is well-conditioned. That is, for any unit vector $\mathbf{y} \in \mathbb{R}^d$, we have

$$\frac{1}{d^{pC}} \cdot \|\mathbf{M}\mathbf{P}\mathbf{y}\|_p^p \le \|\mathbf{y}\|_p^p \le d^{pC} \cdot \|\mathbf{M}\mathbf{P}\mathbf{y}|_p^p,$$

for a fixed constant $C > 0$. Since $\mathbf{M}$ is a constant-factor subspace embedding for $\mathbf{A}$, then we certainly have

$$\frac{1}{d^{p(C+1)}} \cdot \|\mathbf{APy}\|_p^p \leq \|\mathbf{y}\|_p^p \leq d^{p(C+1)} \cdot \|\mathbf{APy}|_p^p.$$

Thus it follows that

$$\frac{1}{d^{p(C+1)}} \cdot \|\mathbf{By}\|_p^p \leq \|\mathbf{y}\|_p^p \leq d^{p(C+1)} \cdot \|\mathbf{By}|_p^p.$$

Therefore,

$$\max_{\mathbf{y}\in\mathbb{R}^d} |\langle \mathbf{b}_t, \mathbf{y}\rangle|^p \leq s(\mathbf{b}_t) \cdot d^{p(C+1)},$$

so that

$$|\|\langle \mathbf{b}_t, \mathbf{y}\rangle^p - \langle \mathbf{b}'_t, \mathbf{y}\rangle^p| \leq \varepsilon' \cdot s(\mathbf{b}_t) \cdot d^{p(C+1)}.$$

Then by triangle inequality, we have

$$\left|\|\mathbf{By}\|_p^p - \|\mathbf{B}'\mathbf{y}\|_p^p\right| \leq \sum_{t=1}^n |\|\langle \mathbf{b}_t, \mathbf{y}\rangle - \langle \mathbf{b}'_t, \mathbf{y}\rangle|^p \leq \varepsilon' \cdot \sum_{t=1}^n s(\mathbf{b}_t) \cdot d^{p(C+1)}.$$

Since the sum of L_p sensitivities is at most d by Lemma E.11, then

$$\left|\|\mathbf{By}\|_p^p - \|\mathbf{B}'\mathbf{y}\|_p^p\right| \leq \varepsilon' \cdot d^{p(C+2)}.$$

The desired claim then follows by the setting of $\varepsilon' = \frac{\text{poly}(\varepsilon)}{\text{poly}(d^p, \log(nd\Delta))}$, recalling that $\mathbf{By} = \mathbf{APP}^{-1}\mathbf{x}$ and $\mathbf{B}'\mathbf{y} = \mathbf{A}'\mathbf{PP}^{-1}\mathbf{x}$. □

We now give the full guarantees of Algorithm 7, which gives an efficient encoding of an input matrix $\mathbf{A}$.

Lemma E.13. *Let $\mathbf{A} \in \mathbb{R}^{m\times d}$ be a coreset construction with entries in $\{0\} \cup \left[\frac{1}{\text{poly}(n)}, \text{poly}(n)\right]$. Let*

$$f\left(d, \frac{1}{\varepsilon}, p\right) = \begin{cases} \frac{d}{\varepsilon^2} \log \frac{d}{\varepsilon} \text{ polylog}\left(\frac{d}{\varepsilon}\right), & p \in [1, 2) \\ \mathcal{O}\left(\frac{d}{\varepsilon^2}\right), & p = 2 \\ \frac{d^{p/2}}{\varepsilon^2} \cdot \text{polylog}\left(\frac{d}{\varepsilon}\right), & p > 2 \end{cases}.$$

Then $\mathbf{A}'$ is a $(1+\varepsilon)$-coreset for $\mathbf{A}$ for L_p subspace embedding that uses $\mathcal{O}\left(f\left(d, 1, p\right)\right) \cdot (d \log n) + md \cdot \text{polylog}\left(d, \frac{1}{\varepsilon}, \log(nd\kappa), \log\frac{1}{\delta}\right)$ bits of space.

Proof. Firstly, note that Algorithm 7 outputs a pair $(\mathbf{M}, \mathbf{B}')$ that encodes $\mathbf{B}'$ using $\mathbf{M}$. By Lemma E.12, we have that for all $\mathbf{x} \in \mathbb{R}^d$,

$$(1-\varepsilon)\|\mathbf{Ax}\|_1 \leq \|\mathbf{A}'\mathbf{x}\|_1 \leq (1+\varepsilon)\|\mathbf{Ax}\|_1,$$

where $\mathbf{A}' = \mathbf{B}'\mathbf{P}^{-1}$. Thus, $\mathbf{A}'$ is a coreset for $\mathbf{A}$.

It remains to analyze the space complexity of $(\mathbf{M}, \mathbf{B}')$. Since $\mathbf{M}$ is a constant-factor L_p subspace embedding, then by Theorem E.3, $\mathbf{M}$ can be represented using $\mathcal{O}\left(f(d, 1, p)\right) \cdot d \log(n)$ bits of space. Furthermore, each offset $\mathbf{b}'$ in $\mathbf{B}'$ is encoded using $\mathcal{O}\left(d \log\left(\frac{1}{\varepsilon} + \log(nd\kappa)\right)\right)$ bits of space due to storing the d coordinates of the offset rounded to the power of $(1+\varepsilon')$. Specifically, we store the exponent of each offset after the rounding, rather than the explicit coordinates. Therefore, the output of the algorithm is encoded using $\mathcal{O}\left(f\left(d, 1, p\right)\right) \cdot (d \log n) + md \cdot \text{polylog}\left(d, \frac{1}{\varepsilon}, \log(nd\kappa), \log\frac{1}{\delta}\right)$. □

E.2 Subspace Embedding in the Streaming Model

In this section, we show how to use a global encoding combined with the efficient encoding from the previous section to achieve our main algorithm for L_p subspace embeddings in the row-arrival model, using our space-efficient framework in Appendix B.

Similar to (k, z)-clustering, the main downfall of immediately applying the previous efficient encoding to each node of a merge-and-reduce tree is that the constant-factor approximation requires $\mathcal{O}\left(d^2 \log(nd)\right)$ bits of storage per efficient encoding, and there can be $\text{polylog}(\log(nd))$ such efficient encodings due to the height of the merge-and-reduce tree on a data stream produced by

online Lewis weight sampling. As before, we circumvent this issue by showing that we can instead consider a single constant-factor approximation to the global dataset. We then show that the error of maintaining such a global constant-factor approximation does not compound too much over the course of the stream.

We remark that in our algorithm, we apply the online coreset construction for L_p subspace embedding as a black box [WY23]. As we mentioned earlier, in online coreset sampling, if we use the previous sketch maintained by the algorithm to compute the future sampling probabilities, the estimation will depend on whether the previous rows are sampled, and so standard concentration inequalities do not apply. To avoid this dependency, [WY23] kept a separate constant-factor L_2 subspace embedding sketch and used it to compute the future online sampling probabilities of the L_p subspace embedding sketch, decoupling the randomness of estimating the online sampling probabilities and that of sampling the rows. We observe that by passing an online L_2 subspace embedding sketch (e.g., by online row sampling [CMP20]) to the merge-and-reduce structure, we obtain a constant-factor approximation with $d\,\mathrm{poly}(\log\log(n\kappa))$ rows. Thus, we can store this sketch by our global efficient encoding in $\mathcal{O}(d)$ words, which does not exceed our space budget. We give the algorithm in full in Algorithm 8, which applies our space-efficient framework in Algorithm 4.

Algorithm 8 L_p-subspace embedding on insertion-only streams

Require: Matrix $\mathbf{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\} \in \mathbb{R}^{n\times d}$ that arrives as a data stream, $\varepsilon \in (0,1)$, parameter $p \geq 1$

Ensure: $(1+\varepsilon)$-coreset for L_p-subspace embedding

$\mathbf{Z}, \mathbf{Z}' \leftarrow \emptyset$, $\lambda \leftarrow \mathcal{O}\left(\frac{d}{\varepsilon^2}\cdot \log d \log n\right)$

for each $t \in [n]$ **do**

  Use $\mathbf{Z}' \cup \{\widehat{\mathbf{a}_t}\}$ to compute a $\mathcal{O}(1)$-approximation $\widehat{\sigma(\mathbf{a}_t)}$ to the online Lewis weight of $\mathbf{a}_t$

  $p(\mathbf{a}_t) \leftarrow \min(1, \lambda \cdot \widehat{\sigma(\mathbf{a}_t)})$

  With probability $p(\mathbf{a}_t)$, sample $\frac{1}{p(\mathbf{a}_t)}\cdot \mathbf{a}_t$ into a stream $\mathcal{S}'$

  Let $\mathbf{Z}$ be the running output of merge-and-reduce on $\mathcal{S}'$ using the efficient encoding for coreset construction in Algorithm 7 with a *global* constant-factor embedding, for accuracy $\frac{\varepsilon}{\mathrm{poly}(\log\log n\kappa)}$ and failure probability $\frac{1}{\mathrm{poly}\left(\frac{1}{\varepsilon}, \log(n\kappa)\right)}$ {Detailed implementation of Algorithm 4}

  Maintain a separate constant-factor L_2 subspace embedding sketch $\mathbf{Z}'$ with efficient encoding

end for

return $\mathbf{Z}$

We first show that given disjoint matrices $\mathbf{Q}_1, \ldots, \mathbf{Q}_m$ of $\mathbf{A}$, we can apply Algorithm 7 and although the output $\mathbf{Q}_i'$ from our efficient encoding will no longer necessarily be a strong coreset for each corresponding $\mathbf{Q}_i$, the resulting error will only be an additive $\varepsilon' \cdot \|\mathbf{A}\mathbf{x}\|_p^p$ for a fixed $\mathbf{x} \in \mathbb{R}^d$, which suffices for our purposes. Specifically, we will set $\varepsilon' = \frac{\varepsilon}{\mathrm{poly}(\log\log(n\kappa))}$ and $m = \mathrm{poly}(\log\log(n\kappa))$, provided the entries of $\mathbf{A}$ are integers bounded in magnitude by $\mathrm{poly}(nd)$.

Lemma E.14. *Let $\mathbf{Q}_1 \circ \ldots \circ \mathbf{Q}_m = \mathbf{A}$ be a partition of $\mathbf{A}$. Let $\mathbf{M}$ be a constant-factor L_p subspace embedding on $\mathbf{A}$. For each $i \in [m]$, let $\mathbf{Q}_i'$ be the resulting matrix after applying the inverse of the preconditioner to the rounded matrix resulting from $\mathbf{Q}_i$. Then for all $i \in [m]$ and all $\mathbf{x} \in \mathbb{R}^d$,*

$$\left|\|\mathbf{Q}_i'\mathbf{x}\|_p^p - \|\mathbf{Q}_i\mathbf{x}\|_p^p\right| \leq \frac{\varepsilon}{\mathrm{poly}(\log\log n\kappa)}\cdot \|\mathbf{A}\mathbf{x}\|_p^p.$$

Proof. Note that by Lemma E.12 with accuracy $\frac{\varepsilon}{\mathrm{poly}(\log\log n\kappa)}$, we have

$$\left|\|\mathbf{A}'\mathbf{x}\|_p^p - \|\mathbf{A}\mathbf{x}\|_p^p\right| \leq \frac{\varepsilon}{\mathrm{poly}(\log\log n\kappa)}\cdot \|\mathbf{A}\mathbf{x}\|_p^p.$$

Since

$$\left|\|\mathbf{Q}_i'\mathbf{x}\|_p^p - \|\mathbf{Q}_i\mathbf{x}\|_p^p\right| \leq \sum_{i=1}^m \left|\|\mathbf{Q}_i'\mathbf{x}\|_p^p - \|\mathbf{Q}_i\mathbf{x}\|_p^p\right| = \left|\|\mathbf{A}'\mathbf{x}\|_p^p - \|\mathbf{A}\mathbf{x}\|_p^p\right|,$$

then the claim follows. □

We now give the full guarantees of Algorithm 8 by showing correctness of the global encoding and specifically, that there is no compounding error across the $\text{poly}(\log\log(nd))$ iterations. We remark that our update time in each step is dominated by reconstructing the sketch and computing the online sampling probability, and so the update time is $\text{poly}(d)$, where the power of d depends on the matrix-multiplication exponential.

Theorem E.15. *Given a matrix $\mathbf{A}$ of n rows on $[-M, \ldots, -1, 0, 1, \ldots, M]^d$, with $M = \text{poly}(n)$ and online condition number κ defined as above, there exists a one-pass streaming algorithm in the row arrival model that outputs a $(1+\varepsilon)$-coreset of $\mathbf{A}$ and uses $\mathcal{O}(d) \cdot f\left(d, \frac{1}{\varepsilon}, p\right)$ words of space, where*

$$f\left(d, \frac{1}{\varepsilon}, p\right) = \begin{cases} \frac{d}{\varepsilon^2} \log \frac{d}{\varepsilon} \operatorname{polylog}\left(\frac{d}{\varepsilon}\right), & p \in [1,2) \\ \mathcal{O}\left(\frac{d}{\varepsilon^2}\right), & p = 2 \\ \frac{d^{p/2}}{\varepsilon^2} \cdot \operatorname{polylog}\left(\frac{d}{\varepsilon}\right), & p > 2 \end{cases}.$$

The (amortized) update time of this algorithm is $\text{poly}(d)$.

Proof. Consider Algorithm 8. We prove correctness by induction on t, claiming that after $j \cdot f\left(d, \frac{1}{\varepsilon}, p\right)$ samples into $\mathcal{S}'$, then $\mathbf{Z}$ is a $(1+\mathcal{O}(\varepsilon))^{j+1}$ coreset of $\mathbf{A}_t$. The online Lewis weight of the first row is always 1 and thus $\mathbf{Z} = \mathbf{a}_1$ after the first step, and the base case is complete. For the inductive hypothesis, we condition on the correctness of $\mathbf{Z}$ being a $(1+\mathcal{O}(\varepsilon))^j$-coreset of $\mathbf{A}_{t-1} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_{t-1}$ after $j-1$ merges have occured. Then $\mathbf{Z} \circ \mathbf{a}_t$ is a $(1+\mathcal{O}(\varepsilon))^j$-coreset of $\mathbf{A}_t = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_t$, so we can use $\mathbf{Z} \circ \mathbf{a}_t$ to compute a constant-factor approximation to the online Lewis weight $\sigma(\mathbf{a}_t)$ of $\mathbf{a}_t$ for a sufficiently small ε to be specified later. Note that merge-and-reduce will be correct if $\mathcal{S}'$ is a stream with length at most $\mathcal{O}\left(g\left(n, d, \frac{1}{\varepsilon}, \kappa, p\right)\right)$, since the failure probability is set to be $\frac{1}{\text{poly}\left(\frac{1}{\varepsilon}, \log(n\kappa)\right)}$ and merge-and-reduce will consider at least $f\left(d, \frac{1}{\varepsilon}, p\right)$ stream updates before applying a new coreset construction. Thus by Lemma E.14 the correctness of merge-and-reduce on $\mathcal{S}'$, we have that $\mathbf{Z}$ after time t will be a $(1+\mathcal{O}(\varepsilon))^{j+1}$-strong coreset for $\mathbf{A}_t$ after j merges, which completes the induction. Finally, we have that by Theorem E.6 through Theorem E.5, at most $\mathcal{O}\left(g\left(n, d, \frac{1}{\varepsilon}, \kappa, p\right)\right)$ samples will be inserted into $\mathcal{S}$, so that $j \leq \mathcal{O}\left(\log \frac{g\left(n, d, \frac{1}{\varepsilon}, \kappa, p\right)}{f\left(d, \frac{1}{\varepsilon}, p\right)}\right) \leq \text{poly}(\log\log(n\kappa))$. The correctness guarantee then follows by rescaling ε with $\text{poly}(\log\log(n\kappa))$.

To analyze the space complexity, recall that the total number of sampled rows into $\mathcal{S}'$ is $n' = \mathcal{O}\left(g\left(n, d, \frac{1}{\varepsilon}, \kappa, p\right)\right)$, with high probability. We do not maintain $\mathcal{S}'$, but instead feed $\mathcal{S}'$ as input to the merge-and-reduce subroutine. Hence, the input to merge-and-reduce has size n' and so note that each coreset in the merge-and-reduce tree has size $f\left(d, \frac{\text{poly}(\log\log(n\kappa))}{\varepsilon}, p\right)$. We use the efficient encoding for coreset construction given in Algorithm 7 with accuracy $\frac{\varepsilon}{\text{poly}(\log\log(n\kappa))}$ and failure probability $\frac{1}{\text{poly}\left(\frac{1}{\varepsilon}, \log(n\kappa)\right)}$. Similar to the analysis of Lemma E.13, we recall that total representation of $\mathbf{Z}$ requires a constant-factor L_p subspace embedding $\mathbf{M}$ that uses $\mathcal{O}(f(d,1,p)) \cdot (d \log n)$ bits of space. We then use Lemma E.14 to encode each of the coresets of size $f\left(d, \frac{\text{poly}(\log\log(n\kappa))}{\varepsilon}, p\right)$, using $d \cdot \operatorname{polylog}\left(d, \frac{1}{\varepsilon}, \log(nd\kappa), \log\frac{1}{\delta}\right)$ bits per row. Since the height of the merge-and-reduce tree is at most $\text{poly}(\log\log(n\kappa))$, it follows that there are at most $\text{poly}(\log\log(n\kappa))$ such coresets. Therefore, the total space required is

$$\mathcal{O}(d \log n) \cdot f(d,1,p) + f\left(d, \frac{1}{\varepsilon}, p\right) \cdot \operatorname{polylog}(n') \cdot d \cdot \operatorname{polylog}\left(\frac{1}{\varepsilon}, \log(nd\kappa)\right).$$

in bits, which is equivalent to $\mathcal{O}(d) \cdot f\left(d, \frac{1}{\varepsilon}, p\right)$ words of space, as desired. In addition, we maintain a constant-factor L_2 subspace embedding $\mathbf{Z}'$, which is used to estimate the online Lewis weights. The construction of $\mathbf{Z}'$ follows the same procedure, which passes the online sub-stream to a merge-and-reduce structure. Note that each coreset in the merge-and-reduce tree now has size $f(d, \text{poly}(\log\log(n\kappa)), 2)$. We again use the efficient encoding for coreset construction given in Algorithm 7 with accuracy $\frac{1}{\text{poly}(\log\log(n\kappa))}$ and failure probability $\frac{1}{\text{poly}\left(\frac{1}{\varepsilon}, \log(n\kappa)\right)}$. Thus, the total space required is

$$\mathcal{O}(d \log n) \cdot f(d,1,2) + f(d,1,2) \cdot \operatorname{polylog}(n') \cdot d \cdot \operatorname{polylog}(\log(nd\kappa)).$$

in bits, which is dominated by the space to store the L_p subspace embedding sketch.

To analyze the update time, at each step, we need to reconstruct the constant-factor L_2 subspace embedding from our global encoding, using $\mathrm{poly}(d, \mathrm{polylog}(\log n\kappa)$ update time by Theorem E.9. Then, we apply the reconstructed sketch to estimate the online Lewis weight of $\mathbf{a}_t$, using $\mathrm{poly}(d)$ update time [CMP20, WY23]. We note that, whenever we merge some coreset, we need $\mathrm{poly}\left(d, \frac{1}{\varepsilon}, \log\log(n\kappa)\right)$ update time to compute the global efficient encoding. However, since we only conduct $\mathrm{polylog}(n\kappa)$ times such operation in the entire stream, they are amortized to lower order terms and hence are dominated by the time to compute the online Lewis weights. □

F Projection-Cost Preserving Sketches

In this section, we describe our one-pass streaming algorithm on insertion-only streams for constructing projection-cost preserving (PCP) sketches that uses $\mathcal{O}_{k,\varepsilon}\left(\mathrm{poly}(\log\log(n\kappa))\right) + \mathcal{O}_{d,\varepsilon}(1)$ words of space. Recall that in the problem of constructing (PCP) sketches in the streaming setting, the rows of a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$ are revealed sequentially, and the goal is to maintain a much smaller matrix $\mathbf{B} \in \mathbb{R}^{m\times d}$ such that for every rank-k orthogonal projection matrix $\mathbf{P} \in \mathbb{R}^{d\times d}$,

$$(1-\varepsilon)\|\mathbf{A}-\mathbf{AP}\|_F^2 \le \|\mathbf{B}-\mathbf{BP}\|_F^2 \le (1+\varepsilon)\|\mathbf{A}-\mathbf{AP}\|_F^2,$$

where $\|\cdot\|_F$ denotes the Frobenius norm. A PCP sketch approximately preserves the total squared distance of the data to any k-dimensional subspace, thereby maintaining the geometric structure relevant to dimensionality reduction. As a result, PCP sketches play a central role in randomized numerical linear algebra, including applications to low-rank approximation, matrix sketching, and principal component analysis [CEM+15, CMM17, BDM+20, MM20].

We next recall the notion of ridge leverage scores, which generalize classical leverage scores and are fundamental for defining the sampling probability in constructing PCP sketches.

Definition F.1 (λ-ridge leverage score)**.** *For a parameter $\lambda > 0$, the* λ-ridge leverage score *of a row $\mathbf{a}_i \in \mathbb{R}^d$ of a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$ is defined as*

$$\mathbf{a}_i^\top(\mathbf{A}^\top\mathbf{A} + \lambda\mathbb{I}_d)^{-1}\mathbf{a}_i.$$

In our setting, we choose $\lambda_{\mathbf{A}} := \frac{\|\mathbf{A}-\mathbf{A}_k\|_F^2}{k}$, where $\mathbf{A}_k$ denotes the best rank-k approximation to $\mathbf{A}$.

Sampling rows proportionally to their $\lambda_{\mathbf{A}}$-ridge leverage scores produces a $(1+\varepsilon)$-PCP sketch. We summarize the standard offline guarantee below.

Theorem F.2 (Offline rank-k PCP sketch)**.** *[CMM17] Given a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$, there is an algorithm which, with probability at least $1-\delta$, outputs a PCP sketch matrix $\mathbf{M}$ consisting of*

$$f\left(k, \frac{1}{\varepsilon}, \frac{1}{\delta}\right) = \mathcal{O}\left(\frac{k}{\varepsilon^2}\log\frac{k}{\delta}\right)$$

rescaled rows of $\mathbf{A}$.

Similarly, we extend the ridge leverage score to the online setting.

Definition F.3 (Online λ-ridge leverage score)**.** *For a parameter $\lambda > 0$, the* online λ-ridge leverage score *of a row $\mathbf{a}_i \in \mathbb{R}^d$ in a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$ is*

$$\mathbf{a}_i^\top(\mathbf{A}_{i-1}^\top\mathbf{A}_{i-1} + \lambda\mathbb{I}_d)^{-1}\mathbf{a}_i,$$

where $\mathbf{A}_{i-1}$ denotes the prefix matrix formed by the first $i-1$ rows of $\mathbf{A}$.

In the streaming setting, the value $\lambda_{\mathbf{A}} = \frac{\|\mathbf{A}-\mathbf{A}_k\|_F^2}{k}$ is not known in advance. Instead, for each incoming row $\mathbf{a}_i$, one sets $\lambda_i = \Theta\left(\frac{\|\mathbf{M}_{i-1}-(\mathbf{M}_{i-1})_k\|_F^2}{k}\right)$, where $\mathbf{M}_{i-1}$ consists of the previously sampled and rescaled rows. Because this estimate never exceeds the true $\lambda_{\mathbf{A}}$, the resulting procedure oversamples but still guarantees valid PCP preservation.

We now state the performance guarantee of the online PCP construction.

Theorem F.4 (Online rank-k PCP sketch)**.** *[BDM+20] Let $\varepsilon > 0$, $k \geq 1$, and let $\mathbf{A} \in \mathbb{R}^{n\times d}$ be a matrix whose rows arrive sequentially with condition number κ. There exists an online algorithm which, with high probability, outputs a PCP sketch matrix $\mathbf{M}$ consisting of*

$$g\left(n, k, \frac{1}{\varepsilon}, \kappa\right) = \mathcal{O}\left(\frac{k}{\varepsilon^2}\log n \log^2 \kappa\right)$$

rescaled rows of $\mathbf{A}$.

Finally, we recall a technical bound on the cumulative online ridge leverage scores, which controls the number of sampled rows.

Lemma F.5 (Upper bound on the sum of online ridge leverage scores)**.** *[BDM+20] Let $\mathbf{A} \in \mathbb{R}^{n\times d}$ have condition number κ. For each $i \in [n]$, define*

$$\lambda_{\mathbf{a}_i} := \frac{\|\mathbf{A}_{i-1} - (\mathbf{A}_{i-1})_k\|_F^2}{k}, \qquad \sigma(\mathbf{a}_i) := \mathbf{a}_i^\top (\mathbf{A}_{i-1}^\top \mathbf{A}_{i-1} + \lambda_{\mathbf{a}_i}\mathbb{I}_d)^{-1}\mathbf{a}_i.$$

Then the total online ridge leverage satisfies $\sum_{i=1}^n \sigma(\mathbf{a}_i) = \mathcal{O}\left(k\log^2\kappa\right)$.

F.1 Efficient Encoding for Coreset Construction for Projection-Cost Preservation

In this section, we describe the efficient encoding for PCP. The strategy is to round a PCP sketch to its optimal rank-k approximation: we first run the algorithm in Theorem F.2 to obtain a $(1+\varepsilon)$-PCP $\mathbf{B}$. Let $\mathbf{B} \in \mathbb{R}^{r\times d}$. Let $\mathbf{B}_k$ be the optimal rank-k approximation to $\mathbf{B}$, we round $\mathbf{D} = \mathbf{B} - \mathbf{B}_k$ using entry-wise relative error quantization with precision parameter ε' to obtain $\mathbf{D}'$. Then we store $(\mathbf{B}_k, \mathbf{D}')$ in the memory, and the reconstruction matrix is defined as $\mathbf{B}' := \mathbf{B}_k + \mathbf{D}'$. The following lemma shows that the encoding gives a $(1+\varepsilon)$-PCP of matrix $\mathbf{B}$.

Lemma F.6. *For all rank k orthogonal projection matrices $\mathbf{P} \in \mathbb{R}^{d\times d}$,*

$$(1-\varepsilon)\|\mathbf{B} - \mathbf{B}\mathbf{P}\|_F^2 \leq \|\mathbf{B}' - \mathbf{B}'\mathbf{P}\|_F^2 \leq (1+\varepsilon)\|\mathbf{B} - \mathbf{B}\mathbf{P}\|_F^2.$$

Proof. Let $\mathbf{Q} = \mathbb{I}_d - \mathbf{P}$, it suffices to show that

$$\left|\|\mathbf{B}'\mathbf{Q}\|_F^2 - \|\mathbf{B}\mathbf{Q}\|_F^2\right| \leq \varepsilon\|\mathbf{B}\mathbf{Q}\|_F^2.$$

Now, we define the error matrix $\mathbf{E} = \mathbf{B}' - \mathbf{B} = \mathbf{D}' - \mathbf{D}$, then we have

$$\|\mathbf{B}'\mathbf{Q}\|_F^2 = \|(\mathbf{B}+\mathbf{E})\mathbf{Q}\|_F^2 = \|\mathbf{B}\mathbf{Q}\|_F^2 + 2\langle\mathbf{B}\mathbf{Q}, \mathbf{E}\mathbf{Q}\rangle_F + \|\mathbf{E}\mathbf{Q}\|_F^2.$$

Then, we can decompose

$$\left|\|\mathbf{B}'\mathbf{Q}\|_F^2 - \|\mathbf{B}\mathbf{Q}\|_F^2\right| = \left|2\langle\mathbf{B}\mathbf{Q}, \mathbf{E}\mathbf{Q}\rangle_F + \|\mathbf{E}\mathbf{Q}\|_F^2\right| \leq 2\|\mathbf{B}\mathbf{Q}\|_F\|\mathbf{E}\mathbf{Q}\|_F + \|\mathbf{E}\mathbf{Q}\|_F^2,$$

where the second step is by Cauchy-Schwartz inequality. Since $\mathbf{Q}$ is a projection matrix, we have $\|\mathbf{E}\mathbf{Q}\|_F \leq \|\mathbf{E}\|_F\|\mathbf{Q}\|_2 = \|\mathbf{E}\|_F$. Moreover, the entry-wise quantization of $\mathbf{D}$ gives us

$$\|\mathbf{E}\|_F = \|\mathbf{D}' - \mathbf{D}\|_F \leq \varepsilon'\|\mathbf{D}\|_F = \varepsilon'\|\mathbf{B} - \mathbf{B}_k\|_F.$$

Since $\mathbf{B}_k$ is the best low-rank approximation to $\mathbf{B}$ and $\mathbf{P}$ is a rank-k projection matrix, we have

$$\|\mathbf{B} - \mathbf{B}_k\|_F \leq \|\mathbf{B} - \mathbf{B}\mathbf{P}\|_F = \|\mathbf{B}\mathbf{Q}\|_F$$

Then we have

$$\left|\|\mathbf{B}'\mathbf{Q}\|_F^2 - \|\mathbf{B}\mathbf{Q}\|_F^2\right| = 3\varepsilon'\|\mathbf{B}\mathbf{Q}\|_F^2.$$

Taking $\varepsilon' = \frac{\varepsilon}{3}$ gives us the desired bound. □

We next upper bound the space complexity of the encoding.

Lemma F.7. *The efficient encoding scheme stores $\mathcal{O}\left(\left(\frac{k^2}{\varepsilon^2} + kd\right)\log n + dr\cdot\left(\log\frac{1}{\varepsilon} + \log\log n\right)\right)$ bits in memory. In addition, reconstruction takes $\mathcal{O}(rdk)$ time.*

Proof. We first analyze the space required to store the encoding $\mathbf{D}' \in \mathbb{R}^{r\times d}$. We use entry-wise rounding with parameter ε', e.g., by rounding it to the nearest power of $(1+\varepsilon')$. The number of bits required for the base is $\mathcal{O}\left(\log \frac{1}{\varepsilon'}\right) = \mathcal{O}\left(\log \frac{1}{\varepsilon}\right)$. The number of bits required for the exponent depends on the range of the entries in $\mathbf{D}$, which is $\mathrm{poly}(n)$ since we assume that the entry of $\mathbf{A}$ is in $[\frac{1}{\mathrm{poly}(n)}, \mathrm{poly}(n)]$, and so the bits for the exponent are $\mathcal{O}\left(\log \frac{\log n}{\varepsilon'}\right) = \mathcal{O}\left(\log \frac{1}{\varepsilon} + \log\log n\right)$. Thus total bit complexity to store $\mathbf{D}$ is:

$$\mathcal{O}\left(dr \cdot \left(\log \frac{1}{\varepsilon} + \log\log n\right)\right).$$

In addition, we need to store the best rank-k approximation $\mathbf{B}_k$ with high precision, using $\mathcal{O}\left(\left(\frac{k^2}{\varepsilon^2} + kd\right)\log n\right)$ bits, which gives the total space complexity. Moreover, reconstruction computes $\mathbf{B}' = \mathbf{B} + \mathbf{D}'_i$ in $\mathcal{O}\left(rdk\right)$ time. □

Thus, we have the following theorem statement for the local encoding.

Theorem F.8 (Local encoding for PCP)**.** *Let $\mathbf{A} \in \mathbb{R}^{n\times d}$ be a matrix with entries in $\{0\} \cup \left[\frac{1}{\mathrm{poly}(n)}, \mathrm{poly}(n)\right]$ and let $\mathbf{B} \in \mathbb{R}^{r\times d}$ be a $(1+\varepsilon)$-PCP of $\mathbf{A}$. Then there exists a $(1+\varepsilon)$-PCP of $\mathbf{B}$ using $\mathcal{O}\left(\left(\frac{k^2}{\varepsilon^2} + kd\right)\log n + dr \cdot \left(\log \frac{1}{\varepsilon} + \log\log n\right)\right)$ bits of space to store and $\mathcal{O}\left(rdk\right)$ time to reconstruct.*

Global efficient encoding.

Similar to (k,z)-clustering and L_p subspace embedding, immediately applying the previous efficient encoding to each node of a merge-and-reduce tree loses a $\mathrm{polylog}(\log(nd))$ factor due to the height of the merge-and-reduce tree. Again, we bypass this issue by rounding each local PCP to the top-k right singular vectors of a global constant-factor approximation. We then show that the error of maintaining such a global encoding does not compound too much over the entire stream. We present the formal description in Algorithm 9.

Algorithm 9 Global efficient encoding for low-rank approximation

Require: Accuracy parameter $\varepsilon < \frac{1}{2}$, matrix $\mathbf{A} = \mathbf{Q}_1 \circ \ldots \circ \mathbf{Q}_m$, matrix $\mathbf{B} \in \mathbb{R}^{r\times d}$ that is a $(1+\varepsilon)$-PCP of $\mathbf{A}$, matrices $\mathbf{W}_1, \ldots, \mathbf{W}_m \in \mathbb{R}^{r\times d}$ where each $\mathbf{W}_i$ is a $(1+\varepsilon)$-PCP of $\mathbf{Q}_i$, rank k

Ensure: Encodings $\mathbf{W}'_1, \ldots, \mathbf{W}'_m$.

1: Let ε' denote the quantization precision to be determined later. Let $\varepsilon' \le \frac{\varepsilon}{4\sqrt{C_\varepsilon}}$, where $C_\varepsilon = \frac{1+\varepsilon}{1-\varepsilon}$
2: Compute the SVD of $\mathbf{B}$ and let $\mathbf{V}_k \in \mathbb{R}^{d\times k}$ be the top-k right singular vectors. Define projections $\mathbf{P}_\mathbf{B} = \mathbf{V}_k\mathbf{V}_k^\top$ and $\mathbf{P}_\mathbf{B}^\perp = \mathbb{I}_d - \mathbf{P}_\mathbf{B}$
3: Store $\mathbf{V}_k$ using high precision (e.g., $\mathcal{O}\left(\log n\right)$ bits per entry)
4: **for** $i \in [m]$ **do**
5: Compute head coefficients $\mathbf{H}_i = \mathbf{W}_i\mathbf{V}_k$ and store with high precision
6: Compute tail residual $\mathbf{T}_i = \mathbf{W}_i\mathbf{P}_\mathbf{B}^\perp$
7: Round $\mathbf{T}_i$ entry-wise with relative precision ε' to obtain $\mathbf{T}'_i$. Store $\mathbf{T}_i$
8: **end for**
9: In reconstruction, compute $\mathbf{W}'_i = \mathbf{H}_i\mathbf{V}_k^\top + \mathbf{T}'_i$ for each $i \in [m]$
10:
11: **return** $\mathbf{W}' = \mathbf{W}'_1 \circ \ldots \circ \mathbf{W}'_m$

First, we prove a useful lemma that bounds the Frobenius norm of the projected matrix.

Lemma F.9. *For any projection $\mathbf{Q}$, we have $\|\mathbf{W}_i\mathbf{Q}\|_F^2 \le C_\varepsilon\|\mathbf{B}\mathbf{Q}\|_F^2$, where $C_\varepsilon = \frac{1+\varepsilon}{1-\varepsilon}$.*

Proof. Since $\mathbf{W}_i$ is a $(1+\varepsilon)$-PCP of $\mathbf{Q}_i$, $\|\mathbf{W}_i\mathbf{Q}\|_F^2 \le (1+\varepsilon)\|\mathbf{Q}_i\mathbf{Q}\|_F^2$. As $\mathbf{A}$ is the concatenation of $\mathbf{Q}_j$, we have

$$\|\mathbf{A}\mathbf{Q}\|_F^2 = \sum_{j=1}^{m}\|\mathbf{Q}_j\mathbf{Q}\|_F^2 \ge \|\mathbf{Q}_i\mathbf{Q}\|_F^2.$$

Now, since $\mathbf{B}$ is a $(1+\varepsilon)$-PCP of $\mathbf{A}$, $\|\mathbf{BQ}\|_F^2 \geq (1-\varepsilon)\|\mathbf{AQ}\|_F^2$. Combining these inequalities yields

$$\|\mathbf{W}_i\mathbf{Q}\|_F^2 \leq \frac{1+\varepsilon}{1-\varepsilon} \cdot \|\mathbf{BQ}\|_F^2 = C_\varepsilon \|\mathbf{BQ}\|_F^2.$$

□

Using the previous property, we can now show that the global efficient encoding gives a PCP with ε-additive error, which can subsequently be transformed into multiplicative error for our downstream purposes.

Lemma F.10. *For all rank k orthogonal projection matrices $\mathbf{P} \in \mathbb{R}^{d\times d}$ and each $i \in [m]$,*

$$\|\mathbf{W}_i - \mathbf{W}_i\mathbf{P}\|_F^2 - \varepsilon\|\mathbf{B} - \mathbf{BP}\|_F^2 \leq \|\mathbf{W}_i' - \mathbf{W}_i'\mathbf{P}\|_F^2 \leq \|\mathbf{W}_i - \mathbf{W}_i\mathbf{P}\|_F^2 + \varepsilon\|\mathbf{B} - \mathbf{BP}\|_F^2.$$

Proof. Let $\mathbf{Q} = \mathbb{I}_d - \mathbf{P}$ and fix an arbitrary $i \in [m]$. It suffices to show that

$$\left|\|\mathbf{W}_i'\mathbf{Q}\|_F^2 - \|\mathbf{W}_i\mathbf{Q}\|_F^2\right| \leq \varepsilon\|\mathbf{BQ}\|_F^2.$$

Let $\mathbf{E}_i = \mathbf{W}_i' - \mathbf{W}_i = \mathbf{T}_i' - \mathbf{T}_i$, then by the analysis in Lemma F.6, the projection-cost error satisfies

$$\left|\|\mathbf{W}_i'\mathbf{Q}\|_F^2 - \|\mathbf{W}_i\mathbf{Q}\|_F^2\right| \leq 2\|\mathbf{W}_i\mathbf{Q}\|_F\|\mathbf{E}_i\mathbf{Q}\|_F + \|\mathbf{E}_i\mathbf{Q}\|_F^2.$$

Since $\mathbf{Q}$ is a projection, $\|\mathbf{E}_i\mathbf{Q}\|_F \leq \|\mathbf{E}_i\|_F \leq \varepsilon'\|\mathbf{T}_i\|_F$, and therefore,

$$\left|\|\mathbf{W}_i'\mathbf{Q}\|_F^2 - \|\mathbf{W}_i\mathbf{Q}\|_F^2\right| \leq 2\varepsilon'\|\mathbf{W}_i\mathbf{Q}\|_F\|\mathbf{T}_i\|_F + (\varepsilon')^2\|\mathbf{T}_i\|_F^2.$$

By Lemma F.9, $\|\mathbf{W}_i\mathbf{Q}\|_F^2 \leq C_\varepsilon\|\mathbf{BQ}\|_F^2$. In addition, since $\mathbf{P}_\mathbf{B}$ is the projection matrix onto the top-k singular vectors, we have

$$\|\mathbf{T}_i\|_F^2 = \|\mathbf{W}_i\mathbf{P}_\mathbf{B}^\perp\|_F^2 \leq C_\varepsilon\|\mathbf{BP}_\mathbf{B}^\perp\|_F^2 \leq C_\varepsilon\|\mathbf{BQ}\|_F^2.$$

Substituting these bounds gives

$$\left|\|\mathbf{W}_i'\mathbf{Q}\|_F^2 - \|\mathbf{W}_i\mathbf{Q}\|_F^2\right| \leq C_\varepsilon(2\varepsilon' + (\varepsilon')^2)\|\mathbf{BQ}\|_F^2.$$

Setting $\varepsilon' \leq \frac{\varepsilon}{4\sqrt{C_\varepsilon}}$ gives the desired result of the global efficient encoding scheme. □

Unfortunately, the previous lemma is not sufficient for our purpose, as we need to show that the reconstructed matrix gives a good approximation to the online ridge leverage score of the next row in the stream. To ensure this, we provide the following Loewner order bound of the global encoding. This statement shows that the reconstructed sketch preserve the subspace embedding properties up to a small additive error, which suffices for our use the estimate the next online ridge leverage score.

Lemma F.11. *Let $\lambda_\mathbf{B} := \frac{\|\mathbf{B}-\mathbf{B}_k\|_F^2}{k}$.Then the following additive Loewner bound holds for each $i \in [m]$:*

$$\frac{1}{2} \cdot \mathbf{W}_i^\top\mathbf{W}_i - 3(\varepsilon')^2 k\lambda_\mathbf{B} \cdot \mathbb{I}_d \preceq \mathbf{W}_i'^\top\mathbf{W}_i' \preceq \frac{3}{2} \cdot \mathbf{W}_i^\top\mathbf{W}_i + 9(\varepsilon')^2 k\lambda_\mathbf{B} \cdot \mathbb{I}_d.$$

Proof. We recall that Algorithm 9 writes $\mathbf{W}_i = \mathbf{H}_i\mathbf{V}_k^\top + \mathbf{T}_i$ and $\mathbf{W}_i' = \mathbf{H}_i\mathbf{V}_k^\top + \mathbf{T}_i'$, and we use the definition of the error matrix in Lemma F.10: let $\mathbf{E}_i := \mathbf{W}_i' - \mathbf{W}_i = \mathbf{T}_i' - \mathbf{T}_i$. Entrywise relative quantization with precision ε' ensures that $\|\mathbf{E}_i\|_F \leq \varepsilon'\|\mathbf{T}_i\|_F$. Since $\mathbf{T}_i = \mathbf{W}_i\mathbf{P}_B^\perp$ and $\mathbf{P}_B$ is a projection matrix, Lemma F.9 gives

$$\|\mathbf{T}_i\|_F^2 = \|\mathbf{W}_i\mathbf{P}_B^\perp\|_F^2 \leq C_\varepsilon \cdot \|\mathbf{BP}_B^\perp\|_F^2.$$

Because $\mathbf{BP}_B^\perp = \mathbf{B} - \mathbf{B}_k$, we have

$$\|\mathbf{BP}_B^\perp\|_F^2 = \|\mathbf{B} - \mathbf{B}_k\|_F^2 = k\lambda_\mathbf{B},$$

and hence $\|\mathbf{T}_i\|_F^2 \leq C_\varepsilon k\lambda_\mathbf{B}$. Then we obtain

$$\|\mathbf{E}_i\|_F^2 \leq (\varepsilon')^2 \cdot \|\mathbf{T}_i\|_F^2 \leq (\varepsilon')^2 C_\varepsilon k\lambda_\mathbf{B}.$$

Thus, we bound the Gram matrix of the error matrix:

$$\mathbf{E}_i^\top \mathbf{E}_i \preceq \|\mathbf{E}_i\|_F^2 \cdot \mathbb{I}_d \preceq (\varepsilon')^2 C_\varepsilon k \lambda_{\mathbf{B}} \cdot \mathbb{I}_d.$$

Next, we prove the Loewner bound. We first expand the Gram matrix of $\mathbf{W}_i'$:

$$\mathbf{W}_i'^\top \mathbf{W}_i' = \mathbf{W}_i^\top \mathbf{W}_i + \mathbf{W}_i^\top \mathbf{E}_i + \mathbf{E}_i^\top \mathbf{W}_i + \mathbf{E}_i^\top \mathbf{E}_i.$$

For any $\eta > 0$, applying operator Young's inequality, we have

$$-(\eta\, \mathbf{W}_i^\top \mathbf{W}_i + \eta^{-1} \mathbf{E}_i^\top \mathbf{E}_i) \preceq \mathbf{W}_i^\top \mathbf{E}_i + \mathbf{E}_i^\top \mathbf{W}_i \preceq \eta\, \mathbf{W}_i^\top \mathbf{W}_i + \eta^{-1} \mathbf{E}_i^\top \mathbf{E}_i.$$

Taking $\eta = \frac{1}{2}$, since $\mathbf{E}_i^\top \mathbf{E}_i \preceq (\varepsilon')^2 C_\varepsilon k \lambda_{\mathbf{B}} \cdot \mathbb{I}_d$, we have the following upper bound:

$$\begin{aligned}\mathbf{W}_i'^\top \mathbf{W}_i' &\preceq \mathbf{W}_i^\top \mathbf{W}_i + \frac{1}{2}\mathbf{W}_i^\top \mathbf{W}_i + 2\mathbf{E}_i^\top \mathbf{E}_i + \mathbf{E}_i^\top \mathbf{E}_i \\ &\preceq \frac{3}{2}\mathbf{W}_i^\top \mathbf{W}_i + 3(\varepsilon')^2 C_\varepsilon k \lambda_{\mathbf{B}} \cdot \mathbb{I}_d.\end{aligned}$$

Similarly, we have the following lower bound:

$$\begin{aligned}\mathbf{W}_i'^\top \mathbf{W}_i' &\succeq \mathbf{W}_i^\top \mathbf{W}_i - \left(\frac{1}{2}\mathbf{W}_i^\top \mathbf{W}_i + 2\mathbf{E}_i^\top \mathbf{E}_i\right) + \mathbf{E}_i^\top \mathbf{E}_i \\ &\succeq \frac{1}{2}\mathbf{W}_i^\top \mathbf{W}_i - (\varepsilon')^2 C_\varepsilon k \lambda_{\mathbf{B}} \cdot \mathbb{I}_d.\end{aligned}$$

Moreover since we assume that $\varepsilon < \frac{1}{2}$, we have $C_\varepsilon \leq 3$, and then combining these two bounds gives our desired result. □

We now analyze the space to store the global encoding.

Lemma F.12. *The global efficient encoding scheme stores*

$$\mathcal{O}\left(kd \log n + mrk \log n + mrd\left(\log \frac{1}{\varepsilon'} + \log\log n\right)\right)$$

bits in memory. In addition, the reconstruction time is $\mathcal{O}(mrdk)$.

Proof. First, we store $\mathbf{V}_k \in \mathbb{R}^{d\times k}$ with high precision, using $\mathcal{O}(kd \log n)$ bits of space. Second, we store each head coefficient $\mathbf{H}_i \in \mathbb{R}^{r\times k}$ with high precision, using $\mathcal{O}(mrk \log n)$ bits of space. Lastly, we store each tail residual $\mathbf{T}_i'$ using entry-wise quantization with precision parameter ε'. By Lemma F.7, the quantization stores $\mathcal{O}\left(\log \frac{1}{\varepsilon'} + \log\log n\right)$ bits per entry, and so the space usage is $\mathcal{O}\left(mrd\left(\log \frac{1}{\varepsilon'} + \log\log n\right)\right)$. Summing up the three parts gives the total space complexity. In addition, reconstruction computes $\mathbf{W}_i' = \mathbf{H}_i \mathbf{V}_k^\top + \mathbf{T}_i'$ in $\mathcal{O}(rdk)$ time for each $i \in [m]$, giving $\mathcal{O}(mrdk)$ overall. □

We now give the full guarantee of the global encoding scheme.

Theorem F.13 (Global encoding for PCP)**.** *Let* $\mathbf{A} = \mathbf{Q}_1 \circ \ldots \circ \mathbf{Q}_m \in \mathbb{R}^{n\times d}$ *be a matrix with entries in* $\{0\} \cup \left[\frac{1}{\mathrm{poly}(n)}, \mathrm{poly}(n)\right]$*, let* $\mathbf{B} \in \mathbb{R}^{r\times d}$ *be a* $(1+\varepsilon)$*-PCP of* $\mathbf{A}$*, and let* $\mathbf{W}_i \in \mathbb{R}^{r\times d}$ *be a* $(1+\varepsilon)$*-PCP of* $\mathbf{Q}_i$ *for each* $i \in [m]$*, where* $\mathbf{B}$ *and* $\mathbf{W}_i$*'s are obtained from ridge-leverage score sampling. Let* $\lambda_{\mathbf{A}} := \frac{\|\mathbf{A}-\mathbf{A}_k\|_F^2}{k}$*, there is an algorithm that encodes* $\mathbf{W}_i$ *to* $\mathbf{W}_i'$ *with precision* $\varepsilon' \leq \mathcal{O}(\varepsilon)$ *for each* $i \in [m]$ *such that it satisfies: let* $\mathbf{W} = \mathbf{W}_1' \circ \ldots \circ \mathbf{W}_m'$*, then for all rank* k *orthogonal projection matrices* $\mathbf{P} \in \mathbb{R}^{d\times d}$ *and each* $i \in [m]$*,*

$$(1 - \mathcal{O}(m\varepsilon))\|\mathbf{A} - \mathbf{AP}\|_F^2 \leq \|\mathbf{W} - \mathbf{WP}\|_F^2 \leq (1 + \mathcal{O}(m\varepsilon))\|\mathbf{A} - \mathbf{AP}\|_F^2.$$

In addition, we have the Loewner bound:

$$\frac{1}{4} \cdot \mathbf{A}^\top \mathbf{A} - \mathcal{O}\left((\varepsilon')^2 mk\lambda_{\mathbf{A}}\right) \cdot \mathbb{I}_d \preceq \mathbf{W}^\top \mathbf{W} \preceq 2 \cdot \mathbf{A}^\top \mathbf{A} + \mathcal{O}\left((\varepsilon')^2 mk\lambda_{\mathbf{A}}\right) \cdot \mathbb{I}_d.$$

The algorithm uses

$$\mathcal{O}\left(kd \log n + mrk \log n + mrd\left(\log \frac{1}{\varepsilon'} + \log\log n\right)\right)$$

bits of space to store and $\mathcal{O}(mrdk)$ *time to reconstruct.*

Proof. Fix an arbitrary rank k orthogonal projection matrices $\mathbf{P} \in \mathbb{R}^{d\times d}$. First, we prove the PCP property By Lemma F.10, we have for each $i \in [m]$,

$$\|\mathbf{W}_i - \mathbf{W}_i\mathbf{P}\|_F^2 - \varepsilon\|\mathbf{B} - \mathbf{B}\mathbf{P}\|_F^2 \le \|\mathbf{W}'_i - \mathbf{W}'_i\mathbf{P}\|_F^2 \le \|\mathbf{W}_i - \mathbf{W}_i\mathbf{P}\|_F^2 + \varepsilon\|\mathbf{B} - \mathbf{B}\mathbf{P}\|_F^2.$$

Since $\mathbf{B}$ is a $(1+\varepsilon)$-PCP of $\mathbf{A}$, we have $\|\mathbf{B} - \mathbf{B}\mathbf{P}\|_F^2 = \Theta(1)\|\mathbf{A} - \mathbf{B}\mathbf{A}\|_F^2$, which gives

$$\|\mathbf{W}_i - \mathbf{W}_i\mathbf{P}\|_F^2 - \mathcal{O}(\varepsilon)\|\mathbf{A} - \mathbf{A}\mathbf{P}\|_F^2 \le \|\mathbf{W}'_i - \mathbf{W}'_i\mathbf{P}\|_F^2 \le \|\mathbf{W}_i - \mathbf{W}_i\mathbf{P}\|_F^2 + \mathcal{O}(\varepsilon)\|\mathbf{A} - \mathbf{A}\mathbf{P}\|_F^2.$$

Moreover, since each $\mathbf{W}_i$ is a $(1+\varepsilon)$-PCP of $\mathbf{Q}_i$, we have

$$|\mathbf{W}'_i - \mathbf{W}'_i\mathbf{P}\|_F^2 \ge (1-\varepsilon)\|\mathbf{Q}_i - \mathbf{Q}_i\mathbf{P}\|_F^2 - \mathcal{O}(\varepsilon)\|\mathbf{A} - \mathbf{A}\mathbf{P}\|_F^2$$
$$|\mathbf{W}'_i - \mathbf{W}'_i\mathbf{P}\|_F^2 \le (1+\varepsilon)\|\mathbf{Q}_i - \mathbf{Q}_i\mathbf{P}\|_F^2 + \mathcal{O}(\varepsilon)\|\mathbf{A} - \mathbf{A}\mathbf{P}\|_F^2.$$

Now, because $\mathbf{A} = \mathbf{Q}_1 \circ \ldots \circ \mathbf{Q}_m$ we have $\|\mathbf{A} - \mathbf{A}\mathbf{P}\|_F^2 = \sum_{i\in[m]}\|\mathbf{Q}_i - \mathbf{Q}_i\mathbf{P}\|_F^2$, and the same holds for $\mathbf{W} = \mathbf{W}'_1 \circ \ldots \circ \mathbf{W}'_m$. Therefore, summing across all $i \in [m]$, we have

$$(1 - \mathcal{O}(m\varepsilon))\|\mathbf{A} - \mathbf{A}\mathbf{P}\|_F^2 \le \|\mathbf{W} - \mathbf{W}\mathbf{P}\|_F^2 \le (1 + \mathcal{O}(m\varepsilon))\|\mathbf{A} - \mathbf{A}\mathbf{P}\|_F^2.$$

Next, we show the Loewner bound. From Lemma F.11, we have for each $i \in [m]$,

$$\frac{1}{2}\cdot\mathbf{W}_i^\top\mathbf{W}_i - \mathcal{O}\left((\varepsilon')^2 k\lambda_{\mathbf{B}}\right)\cdot\mathbb{I}_d \preceq \mathbf{W}_i'^\top\mathbf{W}'_i \preceq \frac{3}{2}\cdot\mathbf{W}_i^\top\mathbf{W}_i + \mathcal{O}\left((\varepsilon')^2 k\lambda_{\mathbf{B}}\right)\cdot\mathbb{I}_d,$$

where $\lambda_{\mathbf{B}} := \frac{\|\mathbf{B}-\mathbf{B}_k\|_F}{k} = \Theta(\lambda_{\mathbf{A}})$. Thus,

$$\frac{1}{2}\cdot\mathbf{W}_i^\top\mathbf{W}_i - \mathcal{O}\left((\varepsilon')^2 k\lambda_{\mathbf{A}}\right)\cdot\mathbb{I}_d \preceq \mathbf{W}_i'^\top\mathbf{W}'_i \preceq \frac{3}{2}\cdot\mathbf{W}_i^\top\mathbf{W}_i + \mathcal{O}\left((\varepsilon')^2 k\lambda_{\mathbf{A}}\right)\cdot\mathbb{I}_d.$$

Note that $\mathbf{W}_i$ is obtained from ridge-leverage score sampling, then we have the following Loewner order property (c.f., Theorem 5 in [CMM17])

$$(1-\varepsilon)\mathbf{Q}_i^\top\mathbf{Q}_i - \varepsilon\lambda_{\mathbf{A}}\cdot\mathbb{I}_d \preceq \mathbf{W}_i^\top\mathbf{W}_i \preceq (1+\varepsilon)\mathbf{Q}_i^\top\mathbf{Q}_i + \varepsilon\lambda_{\mathbf{A}}\cdot\mathbb{I}_d.$$

Therefore, we have

$$\frac{1}{4}\cdot\mathbf{Q}_i^\top\mathbf{Q}_i - \mathcal{O}\left((\varepsilon')^2 k\lambda_{\mathbf{A}}\right)\cdot\mathbb{I}_d \preceq \mathbf{W}_i'^\top\mathbf{W}'_i \preceq 2\cdot\mathbf{W}_i^\top\mathbf{W}_i + \mathcal{O}\left((\varepsilon')^2 k\lambda_{\mathbf{A}}\right)\cdot\mathbb{I}_d.$$

Again, since $\mathbf{A}^\top\mathbf{A} = \sum_{i\in[m]}\mathbf{Q}_i^\top\mathbf{Q}_i$, summing across each $i \in [m]$, we have

$$\frac{1}{4}\cdot\mathbf{A}_i^\top\mathbf{A}_i - \mathcal{O}\left((\varepsilon')^2 mk\lambda_{\mathbf{A}}\right)\cdot\mathbb{I}_d \preceq \mathbf{W}^\top\mathbf{W} \preceq 2\cdot\mathbf{W}_i^\top\mathbf{W}_i + \mathcal{O}\left((\varepsilon')^2 mk\lambda_{\mathbf{A}}\right)\cdot\mathbb{I}_d.$$

Lastly, the space usage and the reconstruction time follow from Lemma F.12. □

F.2 Projection-Cost Preservation in the Streaming Model

In this section, we show how to use the global encoding from the previous section to achieve our streaming algorithm for projection-cost preservation with optimal space. We give the algorithm in full in Algorithm 10, which applies our space-efficient framework in Algorithm 4.

We now give the full guarantees of Algorithm 10. We remark that our update time in each step is dominated by reconstructing the sketch and computing the online sampling probability, and so the update time is $\mathrm{poly}(d)$, where the power of d depends on the matrix-multiplication exponential.

Theorem F.14. *Given a matrix $\mathbf{A}$ of n rows on $[-M, \ldots, -1, 0, 1, \ldots, M]^d$, with $M = \mathrm{poly}(n)$ and online condition number κ, there exists a one-pass streaming algorithm in the row arrival model that outputs a $(1+\varepsilon)$-coreset of $\mathbf{A}$ and uses $\mathcal{O}\left(k\,\mathrm{polylog}\left(\frac{1}{\varepsilon}, \log(nd\kappa)\right) + d\right)\cdot f\left(d, \frac{1}{\varepsilon}, \frac{1}{0.01}\right)$ words of space, where $f\left(k, \frac{1}{\varepsilon}, \frac{1}{\delta}\right) = \mathcal{O}\left(\frac{k}{\varepsilon^2}\log\frac{k}{\delta}\right)$. The (amortized) update time of this algorithm is $\mathrm{poly}(d)$.*

Proof. The proof mainly follows from the proof of L_p subspace embedding in Theorem E.15. However, the main difference is that the PCP property does not necessarily imply the subspace embedding property, which is essential in showing the validity of the online sampling probability. Thus, we need to use a finer precision parameter ε' and apply the Loewner order bound in Theorem F.13.

Algorithm 10 Projection-Cost preservation on insertion-only streams

Require: Matrix $\mathbf{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\} \in \mathbb{R}^{n \times d}$ that arrives as a data stream, $\varepsilon \in (0,1)$, parameter $p \geq 1$

Ensure: $(1+\varepsilon)$-coreset for projection-cost preservation

$\mathbf{Z} \leftarrow \emptyset$, $\alpha \leftarrow \mathcal{O}\left(\frac{\log n}{\varepsilon^2}\right)$

for each $t \in [n]$ **do**

  Use $\mathbf{Z} \cup \{\widehat{\mathbf{a}_t}\}$ to compute the online λ-ridge leverage score $\tau(\mathbf{a}_t)$ of $\mathbf{a}_t$, where $\lambda = \frac{\|\mathbf{Z}-\mathbf{Z}_k\|_F^2}{k}$

  $p(\mathbf{a}_t) \leftarrow \min(1, \alpha \cdot \tau(\mathbf{a}_t))$

  With probability $p(\mathbf{a}_t)$, sample $\frac{1}{p(\mathbf{a}_t)} \cdot \mathbf{a}_t$ into a stream $\mathcal{S}'$

  Let $\mathbf{Z}$ be the running output of merge-and-reduce on $\mathcal{S}'$ using the global encoding for coreset in Algorithm 9, for accuracy $\frac{\varepsilon}{\mathrm{poly}(\log\log n\kappa)}$, failure probability $\frac{1}{\mathrm{poly}\left(\frac{1}{\varepsilon}, \log(n\kappa)\right)}$, and precision $\varepsilon' = \frac{1}{\sqrt{k}} \cdot \mathrm{poly}(\log k, \log\frac{1}{\varepsilon}, \log\log n)$ {Detailed implementation of Algorithm 4}

end for

return $\mathbf{Z}$

Consider Algorithm 10. Again, we prove correctness by induction on t, claiming that after $j \cdot f\left(k, \frac{1}{\varepsilon}, \frac{1}{\delta}\right)$ samples into $\mathcal{S}'$, then $\mathbf{Z}$ is a $(1+\mathcal{O}(\varepsilon))^{j+1}$ coreset of $\mathbf{A}_t$. The online ridge leverage score of the first row is always 1 and thus $\mathbf{Z} = \mathbf{a}_1$ after the first step, and the base case is complete. For the inductive hypothesis, we condition on the correctness of $\mathbf{Z}$ being a $(1+\mathcal{O}(\varepsilon))^j$-coreset of $\mathbf{A}_{t-1} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_{t-1}$ after $j-1$ merges have occured, such that $\lambda_{\mathbf{Z}}$ is within the range of $(1 \pm \mathcal{O}(j\varepsilon)) \cdot \lambda_{\mathbf{A}_{t-1}}$. Moreover, we condition on that $\mathbf{Z}$ is a constant-factor subspace embedding of $\mathbf{A}_{t-1}$. Therefore, we can use $\mathbf{Z}$, $\lambda_{\mathbf{z}}$, and $\mathbf{a}_t$ to compute a constant-factor approximation to the desired online λ-ridge leverage score $\mathbf{a}_t^\top(\mathbf{A}_{t-1}^\top\mathbf{A}_{t-1} + \lambda_{\mathbf{A}_t} \cdot \mathbb{I}_d)^{-1}\mathbf{a}_t$, which suffices to give a valid sampling probability. Then by Theorem F.4 through Lemma F.5, sub-stream $\mathcal{S}'$ has $g\left(n, k, \frac{1}{\varepsilon}, \kappa\right) = \mathcal{O}\left(\frac{k}{\varepsilon^2}\log n \log^2\kappa\right)$ edges. Note that merge-and-reduce will consider at least $f\left(d, \frac{1}{\varepsilon}, \frac{1}{\delta}\right)$ stream updates before applying a new coreset construction and the failure probability is set to $\frac{1}{\mathrm{poly}\left(\frac{1}{\varepsilon}, \log(n\kappa)\right)}$, so that $j \leq \mathcal{O}\left(\log\frac{g\left(n,d,\frac{1}{\varepsilon},\kappa,p\right)}{f\left(d,\frac{1}{\varepsilon},p\right)}\right) \leq \mathrm{poly}(\log\log(n\kappa))$ and the failure probability can be union bound over the j merges. Thus by Theorem F.13, we have that $\mathbf{Z}$ after time t will be a $(1+\mathcal{O}(\varepsilon))^{j+1}$-strong coreset for $\mathbf{A}_t$ after j merges; and moreover, by our choice of precision $\varepsilon' = \frac{1}{\sqrt{k}} \cdot \mathrm{poly}(\log k, \log\frac{1}{\varepsilon}, \log\log n)$ in Theorem F.13, $\mathbf{Z}$ still gives a constant-factor subspace embedding for $\mathbf{A}_t$, which completes the induction. The correctness guarantee then follows by rescaling ε with $\mathrm{poly}(\log\log(n\kappa))$.

To analyze the space complexity, note that each coreset in the merge-and-reduce tree has size $f\left(d, \frac{\mathrm{poly}(\log\log(n\kappa))}{\varepsilon}, \mathrm{poly}\left(\frac{1}{\varepsilon}, \log(n\kappa)\right)\right)$. We use the global encoding for coreset construction given in Algorithm 9 with accuracy $\frac{\varepsilon}{\mathrm{poly}(\log\log(n\kappa))}$, failure probability $\frac{1}{\mathrm{poly}\left(\frac{1}{\varepsilon}, \log(n\kappa)\right)}$, and precision parameter $\varepsilon' = \frac{1}{\sqrt{k}} \cdot \mathrm{poly}(\log k, \log\frac{1}{\varepsilon}, \log\log n)$. By Theorem F.13, the global encoding takes

$$\mathcal{O}\left(kd\log n + f\left(d, \frac{\mathrm{poly}(\log\log(n\kappa))}{\varepsilon}, \mathrm{poly}\left(\frac{1}{\varepsilon}, \log(n\kappa)\right)\right) \cdot jk\log n\right) +$$
$$\mathcal{O}\left(f\left(d, \frac{\mathrm{poly}(\log\log(n\kappa))}{\varepsilon}, \mathrm{poly}\left(\frac{1}{\varepsilon}, \log(n\kappa)\right)\right) \cdot jd\left(\log\frac{1}{\varepsilon'} + \log\log n\right)\right)$$

bits of space to store. Since the height of the merge-and-reduce tree is at most $\mathrm{poly}(\log\log(n\kappa))$, it follows that there are $j \leq \mathrm{poly}(\log\log(n\kappa))$ such coresets. In addition, by Theorem F.2 we have $f\left(k, \frac{1}{\varepsilon}, \frac{1}{\delta}\right) = \mathcal{O}\left(\frac{k}{\varepsilon^2}\log\frac{k}{\delta}\right)$, which scales polynomially with $\frac{1}{\varepsilon}$ and $\log\frac{1}{\delta}$. Therefore, the total space required is

$$\mathcal{O}(kd\log n) + (k\log n + d) \cdot f\left(d, \frac{1}{\varepsilon}, \frac{1}{0.01}\right) \cdot \mathrm{polylog}\left(\frac{1}{\varepsilon}, \log(nd\kappa)\right).$$

in bits, which is equivalent to $\mathcal{O}\left(k\,\mathrm{polylog}\left(\frac{1}{\varepsilon}, \log(nd\kappa)\right) + d\right) \cdot f\left(d, \frac{1}{\varepsilon}, \frac{1}{0.01}\right)$ words of space, as desired.

To analyze the update time, in each step, we reconstruct the PCP sketch $\mathbf{Z}$ from our global encoding using $\mathrm{poly}\left(k, \frac{1}{\varepsilon}, \log\log(n\kappa)\right)$ update time by Lemma F.12. Note that we can instead store a constant-factor PCP sketch to define the online sampling probability, which suffices to give a constant-approximation to the ridge leverage score, so we can improve the update time from $\mathrm{poly}\left(k, \frac{1}{\varepsilon}, \log\log(n\kappa)\right)$ to $\mathrm{poly}\left(k, \log\log(n\kappa)\right)$. Then we apply $\mathbf{Z}$ to estimate the online ridge leverage score of row $\mathbf{a}_t$, using $\mathrm{poly}(d)$ time [CMP20]. Similar to L_p subspace embedding, whenever we merge some coreset, we need $\mathrm{poly}\left(d, \frac{1}{\varepsilon}, \log\log(n\kappa)\right)$ update time to compute the global efficient encoding. However, since we only conduct $\mathrm{polylog}(n\kappa)$ times such operation in the entire stream, they are amortized to lower order terms and hence are dominated by the time to compute the online ridge leverage scores. □

G Additional Experimental Results on MNIST

Table Table 1 reports the full experimental results in Section 4, averaging across 5 repetitions. We also include the amortized time, which is the total update time divided by the number of images.

Target budget	Method	Mean cost	Peak space (bits)	Time/image (ms)
20	Baseline	4,231,656	6,581,440	7.15
	Ours	4,150,784	3,911,787	8.77
50	Baseline	3,454,396	14,092,320	10.26
	Ours	3,484,771	8,462,245	15.66
100	Baseline	3,138,230	25,337,707	13.40
	Ours	3,187,218	15,114,096	24.41
200	Baseline	3,015,100	48,054,560	14.91
	Ours	3,052,912	28,572,587	32.61

Table 1: Clustering cost, peak space, and amortized update time on MNIST.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

(1) **Claims**

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [Yes]

Justification: The main contributions and scope of this paper are clearly stated in the abstract and introduction.

Guidelines:

- The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

(2) **Limitations**

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: The paper presents and clarifies the implications of all the assumptions made and discusses the computational efficiency of the proposed algorithms.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

(3) **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: For each theoretical statement, the paper provide the full set of assumptions and a complete (and correct) proof.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

(4) **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper provides all details regarding the datasets and algorithms used in the experiment.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

(5) **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The paper provides the code in the supplementary materials and detailed information about the tested dataset.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.

- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

(6) **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: The paper provides all training and test details and includes all parameters used in the experiment in the supplemental material.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

(7) **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: The paper plots the average of 5 repetitions, and the performance gains are substantial (in each repetition), making the empirical improvements clear without the need for reporting error bars.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

(8) **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: The paper provide detailed information on the computer resources.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

(9) **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: We have reviewed the NeurIPS Code of Ethics and believe that the research conforms to all aspects.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

(10) **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The paper discusses the practical significance and potential applications of our proposed streaming algorithms. We do not foresee any negative societal impacts.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

(11) **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: We believe the paper poses no such risks.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

(12) **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The paper clearly cited the datasets and baselines.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

(13) **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The codes released by the paper include detailed instructions and documentation.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.

- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

(14) **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

(15) **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

(16) **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.