A Framework for Adversarially Robust Streaming Algorithms

[Extended Abstract]

Omri Ben-Eliezer
Harvard University
omribene@cmsa.fas.harvard.edu

Rajesh Jayaram
Carnegie Mellon University
rkjayara@cs.cmu.edu

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

Eylon Yogev
Boston Univ. & Tel Aviv Univ.
eylony@gmail.com

ABSTRACT

We investigate the adversarial robustness of streaming algorithms. In this context, an algorithm is considered robust if its performance guarantees hold even if the stream is chosen adaptively by an adversary that observes the outputs of the algorithm along the stream and can react in an online manner. While deterministic streaming algorithms are inherently robust, many central problems in the streaming literature do not admit sublinear-space deterministic algorithms; on the other hand, classical space-efficient randomized algorithms for these problems are generally not adversarially robust. This raises the natural question of whether there exist efficient adversarially robust (randomized) streaming algorithms for these problems.

In this work, we show that the answer is positive for various important streaming problems in the insertion-only model, including distinct elements and more generally F_p-estimation, F_p-heavy hitters, entropy estimation, and others. For all of these problems, we develop adversarially robust $(1+\varepsilon)$-approximation algorithms whose required space matches that of the best known non-robust algorithms up to a $\text{poly}(\log n, 1/\varepsilon)$ multiplicative factor (and in some cases even up to a constant factor). Towards this end, we develop several generic tools allowing one to efficiently transform a non-robust streaming algorithm into a robust one in various scenarios.

1. INTRODUCTION

The streaming model of computation is a central and crucial tool for the analysis of massive datasets, where the sheer size of the input imposes stringent restrictions on the memory, computation time, and other resources available to the algorithms. Examples of practical settings where streaming algorithms are in need are easy to encounter. These include internet routers and traffic logs, databases, sensor networks, financial transaction data, and scientific data streams. Given this wide range of applicability, there has been significant effort devoted to designing and analyzing extremely efficient one-pass algorithms. We recommend the survey of [26] for a comprehensive overview of streaming algorithms and their applications.

Many central problems in the streaming literature do not admit sublinear-space deterministic algorithms, and in these cases randomized solutions are necessary. In other cases, randomized solutions are more efficient and simpler to implement than their deterministic counterparts. While randomized streaming algorithms are well-studied, the vast majority of them are defined and analyzed in the *static* setting, where the stream is worst-case but fixed in advance, and only then the randomness of the algorithm is chosen. However, assuming that the stream sequence is independent of the chosen randomness, and in particular that future elements of the stream do not depend on previous outputs of the streaming algorithm, may not be realistic [1, 4, 13, 14, 16, 25, 27], even in non-adversarial settings. For example, suppose that a user sequentially makes queries to a database, and receives an immediate response after each query. Naturally, future queries made by the user in such a setting may heavily depend on the responses given by the database to previous queries. In other words, the stream updates are chosen adaptively, and cannot be assumed to be fixed in advance.

A streaming algorithm that works even when the stream is adaptively chosen by an adversary (the precise definition given next) is said to be *adversarially robust*. Deterministic algorithms are inherently adversarially robust, since they are guaranteed to be correct on all possible inputs. However, the large gap in performance between deterministic and randomized streaming algorithms for many problems motivates the need for designing adversarially robust randomized algorithms, if they even exist. In particular, we would like to design adversarially robust randomized algorithms which are as space and time efficient as their static counterparts, and yet as robust as deterministic algorithms. The study of

This is a minor revision of the paper entitled "A Framework for Adversarially Robust Streaming Algorithms", published in the Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS'20), June 14–19, 2020, Portland, OR, USA. ACM ISBN 978-1-4503-7108-7/20/06. http://dx.doi.org/10.1145/3375395.3387658

such algorithms is the main focus of our work.

The Adversarial Setting.

There are several ways to define the adversarial setting, which may depend on the information the adversary (who chooses the stream) can observe from the streaming algorithm, as well as other restrictions imposed on the adversary. For the most part, we consider a general model, where the adversary is allowed unbounded computational power and resources, though we do discuss the case later when the adversary is computationally bounded. At each point in time, the streaming algorithm publishes its output to a query for the stream. The adversary observes these outputs one-by-one, and can choose the next update to the stream adaptively, depending on the full history of the outputs and stream updates. The goal of the adversary is to force the streaming algorithm to eventually produce an *incorrect* output to the query, as defined by the specific streaming problem in question.

Formally, a data stream of length m over a domain $[n]$ is a sequence of updates of the form $(a_1, \Delta_1), \ldots, (a_m, \Delta_m)$ where $a_t \in [n]$ is an index and $\Delta_t \in \mathbb{Z}$ is an increment or decrement to that index. The *frequency vector* $f \in \mathbb{R}^n$ of the stream is the vector with i^{th} coordinate $f_i = \sum_{t:a_t=i} \Delta_t$. We write $f^{(t)}$ to denote the frequency vector restricted to the first t updates, namely $f_i^{(t)} = \sum_{j \le t: a_j = i} \Delta_j$. It is assumed at all points t that the maximum coordinate in absolute value, denoted $\|f^{(t)}\|_\infty$, is at most M for some $M > 0$, and that $\log(mM) = O(\log n)$. In the *insertion-only* model, the updates are assumed to be positive, meaning $\Delta_t > 0$, whereas in the *turnstile* model Δ_t can be positive or negative.

The general task in streaming is to respond to some query $\mathcal{Q}$ about the frequency vector $f^{(t)}$ at each point in time $t \in [m]$. Oftentimes, this query is to approximate some function $g : \mathbb{R}^n \to \mathbb{R}$ of $f^{(t)}$ (ideally, one might wish to exactly compute the function g; however, in many cases, and in particular for the problems that we consider here, exact computation cannot be done with sublinear space). For example, counting the number of distinct elements in a data stream is among the most fundamental problems in the streaming literature; here $g(f^{(t)})$ is the number of non-zero entries in $f^{(t)}$. Since exact computation cannot be done in sublinear space [8], the goal is to approximate the value of $g(f^{(t)})$ to within a multiplicative factor of $(1 \pm \varepsilon)$. Another important streaming problem (which is not directly an estimation task) is the *Heavy-Hitters* problem, where the algorithm is tasked with finding all the coordinates in $f^{(t)}$ which are larger than some threshold τ.

Formally, the adversarial setting is modeled by a two-player game between a (randomized) StreamingAlgorithm and an Adversary. At the beginning, a query $\mathcal{Q}$ is fixed, which the StreamingAlgorithm must continually reply to. The game proceeds in rounds, where in the t-th round:

1. Adversary chooses an update $u_t = (a_t, \Delta_t)$ for the stream, which can depend, in particular, on all previous stream updates and outputs of StreamingAlgorithm.

2. StreamingAlgorithm processes the new update u_t and outputs its current response R^t to the query $\mathcal{Q}$.

3. Adversary observes R^t (stores it) and proceeds to the next round.

The goal of the Adversary is to make the StreamingAlgorithm output an incorrect response R^t to $\mathcal{Q}$ at some point t in the stream. For example, in the distinct elements problem, the adversary's goal is that on some step t, the estimate R^t will fail to be a $(1+\varepsilon)$-approximation of the true current number of distinct elements $|\{i \in [n] : f_i^{(t)} \neq 0\}|$.

Streaming algorithms in the adversarial setting.

It was shown by Hardt and Woodruff [16] that linear sketches are inherently *non-robust* in adversarial settings for a large family of problems, thus demonstrating a major limitation of such sketches. In particular, their results imply that no linear sketch can approximate the Euclidean norm of its input to within a polynomial multiplicative factor in the adversarial (turnstile) setting. Here, a linear sketch is an algorithm whose output depends only on values $S \cdot f$ and S, for some (usually randomized) sketching matrix $S \in \mathbb{R}^{k \times n}$. This is quite unfortunate, as the vast majority of turnstile streaming algorithms are in fact linear sketches.

Indeed, the typical guarantee is that for any fixed f, $S \cdot f$ satisfies some property with good probability. If f is allowed to depend on S, this property typically does not hold. For example, if S is a random Gaussian matrix, then the Euclidean norm $\|S \cdot f\|_2$ is close to $\|f\|_2$ with large probability. However, if f is allowed to depend on S, then one can choose f to be a large non-zero vector orthogonal to the rows of S, so that $\|S \cdot f\|_2$ is zero while $\|f\|_2$ is non-zero. One can show that for a number of sketches, answers to previous queries reveal information about S, and consequently an adversary can later construct an f, depending on S, to break them.

On the positive side, recent works of Ben-Eliezer and Yogev [4] and Alon et al. [1] show that *random sampling* is quite robust in the adaptive adversarial setting, albeit with a slightly larger sample size. While uniform sampling is a rather generic and useful tool, it is not sufficient for solving many important streaming tasks, such as estimating frequency moments (F_p-estimation), finding L_2 heavy hitters, and various other data analysis problems. This raises the natural question of whether there exist efficient adversarially robust randomized streaming algorithms for these problems and others, which is the main focus of this work. Perhaps even more importantly, we ask the following.

Is there a generic technique to transform a static streaming algorithm into an adversarially robust streaming algorithm?

This work answers the above questions affirmatively for a large class of algorithms.

1.1 Our Results

We devise adversarially robust algorithms for various fundamental insertion-only streaming problems, including distinct element estimation, F_p moment estimation, heavy hitters, entropy estimation, and several others. In addition, we give adversarially robust streaming algorithms which can handle a bounded number of deletions as well. The required space of our adversarially robust algorithms matches that of the best known non-robust ones up to a small multiplicative factor. Our new algorithmic results are summarized in Table 1.

In contrast, we demonstrate that some classical randomized algorithms for streaming problems in the static setting,

Table 1: A summary of our adversarially robust algorithms (in bold), as compared to the best known upper bounds for randomized algorithms in the static setting and lower bounds for deterministic algorithms. Note that all stated algorithms provide tracking. All results except for the last two (which hold in restricted versions of the turnstile model) are for insertion only streams. We write $\tilde{O}, \tilde{\Omega}$ to hide $\log \varepsilon^{-1}$ and $\log \log n$ factors. The lower bound for deterministic entropy estimation follows from a reduction from estimating F_p for $p = 1 + \tilde{\Theta}(\frac{\varepsilon}{\log^2 n})$.

Problem	Static Randomized	Deterministic	Adversarial	Comments
Distinct elements (F_0-estimation)	$\tilde{O}(\varepsilon^{-2} + \log n)$ [5]	$\Omega(n)$ [8]	$\boldsymbol{\tilde{O}(\varepsilon^{-3} + \varepsilon^{-1} \log n)}$	
			$\boldsymbol{\tilde{O}(\varepsilon^{-2} + \log n)}$	crypto/random oracle
F_p-estimation, $p \in (0,2] \setminus \{1\}$	$O(\varepsilon^{-2} \log n)$ [6]	$\tilde{\Omega}(2^{-1/(1-p)} \cdot n)$ [8]	$\boldsymbol{\tilde{O}(\varepsilon^{-3} \log n)}$	
	$O(\varepsilon^{-3} \log^2 n)$ [22]		$\boldsymbol{\tilde{O}(\varepsilon^{-3} \log^3 n)}$	$\delta = \Theta(n^{-(1/\varepsilon) \log n})$
F_p-estimation, $p > 2$	$O\Big(n^{1-\frac{2}{p}}(\varepsilon^{-3} \log^2 n + \varepsilon^{-\frac{6}{p}} \log^{\frac{4}{p}+1} n)\Big)$ [12]	$\Omega(n)$ [8]	$\boldsymbol{O\Big(n^{1-\frac{2}{p}}(\varepsilon^{-3} \log^2 n + \varepsilon^{-\frac{6}{p}} \log^{\frac{4}{p}+1} n)\Big)}$	$\delta = \Theta(n^{-(1/\varepsilon) \log n})$
ℓ_2 Heavy Hitters	$O(\varepsilon^{-2} \log^2 n)$ [7]	$\Omega(\sqrt{n})$ [21]	$\boldsymbol{\tilde{O}(\varepsilon^{-3} \log^2 n)}$	
Entropy Estimation	$O(\varepsilon^{-2} \log^3 n)$ [9]	$\tilde{\Omega}(n)$ [17]	$\boldsymbol{O(\varepsilon^{-5} \log^6 n)}$	
	$\tilde{O}(\varepsilon^{-2})$ [20]		$\boldsymbol{O(\varepsilon^{-5} \log^4 n)}$	crypto/random oracle
Turnstile F_p-estimation, $p \in (0,2]$	$O(\varepsilon^{-2} \lambda \log^2 n)$ [22]	$\Omega(n)$ [2]	$\boldsymbol{O(\varepsilon^{-2} \lambda \log^2 n)}$	λ-bounded F_p flip number, $\delta = \Theta(n^{-\lambda})$
F_p-estimation, $p \in [1,2]$, α-bounded deletions	$\tilde{O}(\varepsilon^{-2} \log \alpha \log n + \log^2 n)$ [19]	$\tilde{\Omega}(2^{-1/(1-p)} \cdot n)$ [8]	$\boldsymbol{O(\alpha \varepsilon^{-(2+p)} \log^3 n)}$	static only for $p = 1$

such as the celebrated Alon-Matias-Szegedy (AMS) sketch [2] for F_2-estimation, are inherently non-robust to adaptive adversarial attacks in a strong sense.

The Robustification Framework: Flip number, Sketch Switching, and Computation Paths.

Our adversarially robust algorithms make use of two generic robustification frameworks that we develop, allowing one to efficiently transform a non-robust streaming algorithm into a robust one in various settings. Both of the robustification methods rely on the fact that functions of interest do not drastically change their value too many times along the stream. Specifically, the transformed algorithms have space dependency on the *flip-number* of the stream, which is a bound on the number of times the function $g(f^{(t)})$ can change by a factor of $(1 \pm \varepsilon)$ in the stream (see Section 2).

The first method, called *sketch switching*, maintains multiple instances of the non-robust algorithm and switches between them in a way that cannot be exploited by the adversary. The second technique bounds the number of *computation paths* possible in the two-player adversarial game. This technique maintains only one copy of a non-robust algorithm, albeit with an extremely small probability of error δ. We show that a carefully rounded sequence of outputs generates only a small number of possible computation paths, which can then be used to ensure robustness by union bounding over these paths. The framework is described in Section 2.

The two above methods are incomparable: for some streaming problems the former is more efficient, while for others, the latter performs better, and we show examples of each. Specifically, sketch switching can exploit efficiency gains of *strong-tracking*, resulting in particularly good performance for static algorithms that can respond correctly to queries at each step without having to union bound over all m steps. In contrast, the computation paths technique can exploit an algorithm with good dependency on δ (the failure probability). Namely, algorithms that have small dependency in update-time or space on δ will benefit from the computation paths technique.

For each of the problems we consider, we show how to use the framework, in addition to some further techniques which we develop along the way, to solve it. Interestingly, we also demonstrate how cryptographic assumptions (which were not commonly used before in the streaming context) can be applied to obtain an adversarially robust algorithm against computationally bounded adversaries for the distinct elements problem at essentially no extra cost over the space optimal non-robust one. See Table 1 for a summary of our results in the adversarial setting compared to the state-of-the-art in the static setting, as well as to deterministic algorithms.

Distinct elements, F_p-estimation, and more.

Our first suite of results provides robust streaming algorithms for estimating F_p, the p^{th} frequency moment of the frequency vector, defined as $F_p = \|f\|_p^p = \sum_{i=1}^n |f_i|^p$, where we interpret $0^0 = 0$. Estimating frequency moments has a myriad of applications in databases, computer networks, data mining, and other contexts. Efficient algorithms for estimating distinct elements (i.e., estimating F_0) are important for databases, since query optimizers can use them to find the number of unique values of an attribute without having to perform an expensive sort on the values. Efficient algorithms for F_2 are useful for determining the output size of self-joins in databases, and for computing the surprise index of a data sequence [15]. Higher frequency moments are used to determine data skewness, which is important in parallel database applications [10].

We remark that for any fixed $p \neq 1$, including $p = 0$, any deterministic insertion-only algorithm for F_p-estimation requires $\Omega(n)$ space [2, 8] (note that for the case $p = 1$ there is a trivial $O(\log n)$-bit insertion only F_p-estimation algorithm: keeping a counter for $\sum_t \Delta_t$). In contrast, we show that randomized adversarially robust algorithms exist for all p, whose space complexity either matches or has a small multiplicative overhead over the best static randomized algorithms.

We utilize an optimized version of the sketch switching method to derive an upper bound for estimating the number of distinct elements. The result is an adversarially robust F_0 estimation algorithm whose complexity is only a $\Theta(\frac{1}{\varepsilon}\log\varepsilon^{-1})$ factor larger than the optimal static (non-robust) algorithm [5].

THEOREM 1.1. *There is an algorithm which, when run on an adversarial insertion only stream, with probability at least $1-\delta$ produces at every step $t\in[m]$ an estimate R^t such that $R^t = (1\pm\varepsilon)\|f^{(t)}\|_0$. The space used by the algorithm is*

$$O\left(\frac{\log(1/\varepsilon)}{\varepsilon}\left(\frac{\log\varepsilon^{-1}+\log\delta^{-1}+\log\log n}{\varepsilon^2}+\log n\right)\right).$$

A second result applies the computation paths method with a new *static* algorithm for F_0 estimation which has very small update-time dependency on δ, and nearly optimal space complexity. As a result, we obtain an adversarially robust F_0 estimation algorithm with extremely fast update time (note that the update time of the above sketch switching algorithm would be $O(\varepsilon^{-1}\log n)$ to obtain the same result, even for constant δ).

A third result takes a different approach: it shows that under certain standard cryptographic assumptions, there exists an adversarially robust algorithm which asymptotically matches the space complexity of the best non-robust tracking algorithm for distinct elements.

Our next set of results provides adversarially robust algorithms for F_p-estimation with $p>0$. The following result concerns the case $0<p\le 2$. It was previously shown that for p bounded away from one, $\Omega(n)$ space is required to deterministically estimate $\|f\|_p^p$, even in the insertion only model [2, 8]. On the other hand, space-efficient non-robust randomized algorithms for F_p-estimation exist. We leverage these, along with an optimized version of the sketch switching technique to save a $\log n$ factor, and obtain an adversarially robust algorithm for F_p-estimation, where $0<p<2$.

The next result concerns F_p-estimation for $p>2$. Here again, we provide an adversarially robust algorithm which is optimal up to a small multiplicative factor. This result applies the computation paths robustification method as a black box. Notably, a classic lower bound of [3] shows that for $p>2$, $\Omega(n^{1-2/p})$ space is required to estimate $\|f\|_p^p$ up to a constant factor (improved lower bounds have been provided since, e.g., [24, 12]). By using our computation paths technique, we obtain adversarially robust F_p moment estimation algorithms as well for $p>2$. Lastly, we show that our techniques for F_p moment estimation can be extended to data streams with a bounded number of deletions (negative updates).

Additionally, we show how to get adversarial robust streaming algorithms for a range of problems where it is not clear a-priori how to apply our framework. We show how our techniques can be used to solve the popular *heavy-hitters* problem, and we show how to solve the Entropy estimation problem. See Table 1 for a summary of our results.

Attack on AMS Sketch.

As discussed above, many important streaming problems admit efficient adversarially robust algorithms in the insertion model. It is now natural to ask: are classical algorithms for this problem generally adversarially robust?

We prove that the answer is negative: the classic Alon-Matias-Szegedy sketch (AMS sketch) [2], the first and perhaps most well-known F_2 estimation algorithm (which uses sub-polynomial space), is *not* adversarially robust in the insertion-only model. (In the full turnstile model, in which the adversary is more powerful, the fact that the AMS sketch is not robust follows from the linear sketching lower bound of Hardt and Woodruff [16].) Specifically, we demonstrate an adversary which, when run against the AMS sketch, fools the sketch into outputting a value which is not a $(1\pm\varepsilon)$ estimate of the F_2. The non-robustness of standard static streaming algorithms, even under simple attacks, is a further motivation to design adversarially robust algorithms.

In what follows, recall that the AMS sketch computes $S\cdot f$ throughout the stream, where $S\in\mathbb{R}^{t\times n}$ is a matrix of uniform $\{t^{-1/2},-t^{-1/2}\}$ random variables. The F_2-estimate is then the value $\|Sf\|_2^2$.

THEOREM 1.2. *Let $S\in\mathbb{R}^{t\times n}$ be the AMS sketch, $1\le t\le n/c$ for some constant $c>1$. There is an adversary which, with probability $99/100$, succeeds in forcing the estimate $\|Sf\|_2^2$ of the AMS sketch to not be a $(1\pm 1/2)$ approximation of the true norm $\|f\|_2^2$. Moreover, the adversary needs to only make $O(t)$ stream updates before this occurs.*

1.2 Subsequent Work and Open Questions

Based on this paper, a couple of very recent follow-up works have improved upon the space efficiency of our robustification techniques for different settings. Hassidim et al. [18] use techniques from differential privacy to obtain a generic robustification framework in the same mold as ours, where the dependency on the flip number is the improved $\sqrt{\lambda}$ as opposed to linear in λ - the exact bound includes other $\text{poly}((\log n)/\epsilon)$ factors. Similar to our construction, they run multiple independent copies of the static algorithm A with independent randomness and feed the input stream to all of the copies. Unlike our construction, when a query comes, they aggregate the responses from the copies in a way that protects the internal randomness of each of the copies using differential privacy. Using their framework, one may construct an adversarially robust algorithm for F_p-moment estimation that uses $\widetilde{O}(\frac{\log^4 n}{\epsilon^{2.5}})$ bits of memory for any $p\in[0,2]$. This improves over our $\widetilde{O}(\frac{\log n}{\epsilon^3})$ bound for interesting parameter regimes.

Woodruff and Zhou [28] obtain further improvements for a class of problems that have so-called difference estimators which in some cases are (almost) optimal even for the static case. For example, they give an adversarially robust algorithm for F_p-moment estimation that uses $\widetilde{O}(\frac{\log n}{\epsilon^2})$ bits of memory for any $p\in[0,2]$. This improves upon both our work and [18]. Interestingly, difference estimators, which are a new class of algorithms developed in their paper, turn out to be useful also in the sliding windows (classical) model.

Many problems remain open, mainly for achieving optimal bounds for all known streaming problems in the adversarial setting. In particular, one may ask the following:

Do there exist natural streaming tasks that can be solved in the classical setting using small memory, but which require significantly more memory in the adversarial setting?

Very recently, this question was addressed by Kaplan et al. [23] who constructed a streaming problem exhibiting such a separation between the classical setting, where it only requires a polylogarithmic amount of memory, and the adversarial setting, where polynomial memory is required – that is, an exponential separation. Their construction is based on classical results in adaptive data analysis.

One particular question of interest that remains wide open is related to the turnstile streaming model. The large majority of results in this paper (and in subsequent papers) apply in the insertion-only model. The full turnstile model, where arbitrary insertions and deletions are allowed, is much less understood. In particular we ask the following.

Do there exist small memory streaming algorithms in the adversarial turnstile model for the problems in this paper?

2. TOOLS FOR ROBUSTNESS

In this section, we establish two methods, *sketch switching* and *computation paths*, allowing one to convert an approximation algorithm for any sufficiently well-behaved streaming problem to an adversarially robust one for the same problem. The central definition of a *flip number*, bounds the number of major (multiplicative) changes in the algorithm's output along the stream. As we shall see, a small flip number allows for efficient transformation of non-robust algorithms into robust ones. We remark that the notion of flip number we define here also plays a central role in subsequent works ([18], [28]); for example, the main contribution of the former is a generic robustification technique with an improved (square root type instead of linear) dependence in the flip number. The latter improves the poly$(1/\epsilon)$ dependence on the flip number.

2.1 Flip Number

DEFINITION 2.1 (FLIP NUMBER). *Let $\varepsilon \geq 0$ and $m \in \mathbb{N}$, and let $\bar{y} = (y_0, y_1, \ldots, y_m)$ be any sequence of real numbers. The ε-*flip number *$\lambda_\varepsilon(\bar{y})$ of $\bar{y}$ is the maximum $k \in \mathbb{N}$ for which there exist $0 \leq i_1 < \ldots < i_k \leq m$ so that $y_{i_{j-1}} \notin (1 \pm \varepsilon) y_{i_j}$ for every $j = 2, 3, \ldots, k$.*

Fix a function $g\colon \mathbb{R}^n \to \mathbb{R}$ and a class $\mathcal{C} \subseteq ([n] \times \mathbb{Z})^m$ of stream updates. The (ε, m)-flip number $\lambda_{\varepsilon,m}(g)$ of g over $\mathcal{C}$ is the maximum, over all sequences $((a_1, \Delta_1), \ldots, (a_m, \Delta_m)) \in \mathcal{C}$, of the ε-flip number of the sequence $\bar{\bar{}}=(y_0, y_1, \ldots, y_m)$ defined by $y_i = g(f^{(i)})$ for any $0 \leq i \leq m$, where as usual $f^{(i)}$ is the frequency vector after stream updates $(a_1, \Delta_1), \ldots, (a_i, \Delta_i)$ (and $f^{(0)}$ is the n-dimensional zeros vector).

The class $\mathcal{C}$ may represent, for instance, the subset of all insertion only streams, or bounded-deletion streams. For the rest of this section, we shall assume $\mathcal{C}$ to be fixed, and consider the flip number of g with respect to this choice of $\mathcal{C}$. We note that a somewhat reminiscent definition, of an *unvarying algorithm*, was studied by [11] (see Definition 5.2 there) in the context of differential privacy. While their definition also refers to a situation where the output undergoes major changes only a few times, both the motivation and the precise technical details of their definition are different from ours.

Note that the flip number is clearly monotone in ε: namely $\lambda_{\varepsilon',m}(g) \geq \lambda_{\varepsilon,m}(g)$ if $\varepsilon' < \varepsilon$. One useful property of the flip number is that it is nicely preserved under approximations. As we show, this can be used to effectively construct approximating sequences whose 0-*flip number* is bounded as a function of the ε-flip number of the original sequence. This is summarized in the following lemma.

LEMMA 2.2. *Fix $0 < \varepsilon < 1$. Suppose that $\bar{u} = (u_0, \ldots, u_m)$, $\bar{v} = (v_0, \ldots, v_m)$, $\bar{w} = (w_0, \ldots, w_m)$ are three sequences of real numbers, satisfying the following:*

- *For any $0 \leq i \leq m$, $v_i = (1 \pm \varepsilon/8) u_i$.*
- *$w_0 = v_0$, and for any $i > 0$, if $w_{i-1} = (1 \pm \varepsilon/2) v_i$ then $w_i = w_{i-1}$, and otherwise $w_i = v_i$.*

Then $w_i = (1 \pm \varepsilon) u_i$ for any $0 \leq i \leq m$, and moreover, $\lambda_0(\bar{w}) \leq \lambda_{\varepsilon/8}(\bar{u})$.

In particular, if (in the language of Definition 2.1) $u_0 = g(f^{(0)}), u_1 = g(f^{(1)}), \ldots, u_m = g(f^{(m)})$ for a sequence of updates $((a_1, \Delta_1), \ldots, (a_m, \Delta_m)) \in \mathcal{C}$, then $\lambda_0(\bar{w}) \leq \lambda_{\varepsilon/8,m}(g)$.

PROOF. The first statement, that $w_i = (1 \pm \varepsilon) u_i$ for any i, follows immediately since $v_i = (1 \pm \varepsilon/8) u_i$ and $w_i = (1 \pm \varepsilon/2) v_i$ and since $\varepsilon < 1$. The third statement follows by definition from the second one. It thus remains to prove that $\lambda_0(\bar{w}) \leq \lambda_{\varepsilon/8}(\bar{u})$.

Let $i_1 = 0$ and let $i_2, i_3, \ldots, i_k$ be the collection of all values $i \in [m]$ for which $w_{i-1} \neq w_i$. Note that $k = \lambda_0(\bar{w})$ and that $v_{i_{j-1}} = w_{i_{j-1}} = w_{i_{j-1}+1} = \cdots = w_{i_j - 1} \neq v_{i_j}$ for any $j = 2, \ldots, k$. We now claim that for every j in this range, $u_{i_{j-1}} \notin (1 \pm \varepsilon/8) u_{i_j}$. This would show that $k \leq \lambda_{\varepsilon/8}(\bar{u})$ and conclude the proof.

Indeed, fixing any such j, we either have $v_{i_{j-1}} > (1 + \varepsilon/2) v_{i_j}$, or $w_{i_{j-1}} < (1 - \varepsilon/2) v_{i_j}$. In the first case (assuming $u_{i_j} \neq 0$, as the case $u_{i_j} = 0$ is trivial),

$$\frac{u_{i_{j-1}}}{u_{i_j}} \geq \frac{v_{i_j}/(1 + \frac{\varepsilon}{8})}{v_{i_{j-1}}/(1 - \frac{\varepsilon}{8})} \geq \left(1 + \frac{\varepsilon}{2}\right) \cdot \frac{1 - \frac{\varepsilon}{8}}{1 + \frac{\varepsilon}{8}} > 1 + \frac{\varepsilon}{8}\,.$$

In the second case, an analogous computation gives that $u_{i_{j-1}}/u_{i_j} < 1 - \varepsilon/8$. □

Note that the flip number of a function g critically depends on the model in which we work, as the maximum is taken over all sequences of *possible stream updates*; for insertion-only streams, the set of all such sequences is more limited than in the general turnstile model, and correspondingly many streaming problems have much smaller flip number when restricted to the insertion only model. We now give an example of a class of functions with bounded flip number.

PROPOSITION 2.3. *Let $g : \mathbb{R}^n \to \mathbb{R}$ be any monotone function, meaning that $g(x) \geq g(y)$ if $x_i \geq y_i$ for each $i \in [n]$. Assume further that $g(x) \geq T^{-1}$ for all $x > 0$, and $g(M \cdot \vec{1}) \leq T$, where M is a bound on the entries of the frequency vector and $\vec{1}$ is the all 1's vector. Then the flip number of g in the insertion only streaming model is $\lambda_{\varepsilon,m}(g) = O(\frac{1}{\varepsilon} \log T)$.*

PROOF. Observe that $g(f^{(0)}) = 0$, $g(f^{(1)}) \geq T^{-1}$, and $g(f^{(m)}) \leq g(\vec{1} \cdot M) \leq T$. Since the stream has only positive updates, $g(f^{(0)}) \leq g(f^{(1)}) \leq \cdots \leq g(f^{(m)})$. Let $y_1, \ldots, y_k \in [m]$ be any set of points such that $g(f^{(y_i)}) < (1+\varepsilon) g(f^{(y_{i+1})})$ for each i. Since there are at most $O(\frac{1}{\varepsilon} \log T)$ powers of $(1 + \varepsilon)$ between T^{-1} and T, by the pigeonhole principle if $k > \frac{C}{\varepsilon} \log(T)$ for a sufficiently large constant C, then at least two values must satisfy $(1 + \varepsilon)^j \leq g(f^{(y_i)}) \leq g(f^{(y_{i+1})}) \leq (1 + \varepsilon)^{j+1}$ for some j, which is a contradiction. □

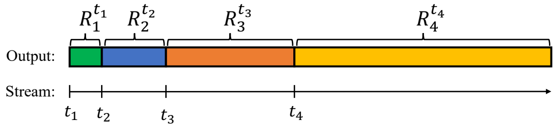

Figure 1: The *sketch switching* method, one of our techniques for transforming a streaming algorithm into an adversarially robust algorithm. As we prove, the following strategy is useful for efficiently robustifying streaming algorithms in a wide range of contexts: maintain several copies $R_1, R_2, \ldots$ of the algorithm, but at any given time t, only communicate to the adaptive adversary the output $R_i^{t_i}$ of a single "active" copy R_i, where $t_i < t$ is the step where R_i became active. We then *switch* the active sketch from R_i to R_{i+1} on the time step t_{i+1} in which the value of $R_i^{t_{i+1}}$ diverges significantly from $R_i^{t_i}$. This will ensure that the algorithm always returns a good approximation, without leaking any meaningful information about its internal state to the adversary.

Note that a special case of the above are the F_p moments of a data stream. Recall here $\|x\|_0 = |\{i : x_i \neq 0\}|$ is the number of non-zero elements in a vector x.

Corollary 2.4. *Let $p > 0$. The (ε, m)-flip number of $\|x\|_p^p$ in the insertion only streaming model is $\lambda_{\varepsilon,m}(\|\cdot\|_p^p) = O(\frac{1+p}{\varepsilon}\log m)$.*

Proof. We have $\|\vec{0}\|_p^p = 0$, $\|z\|_p^p \geq 1$ for any non-zero $z \in \mathbb{Z}$, and $\|f^{(m)}\|_p^p \leq M^p n \leq n^{cp}$ for some constant c, where the second to last inequality holds because $\|f\|_\infty \leq M$ for some $M = \text{poly}(n)$ is assumed at all points in the streaming model. Moreover, for $p = 0$ we have $\|f^{(m)}\|_0 \leq n$. The result then follows from applying Proposition 2.3 with $T = n^{c\cdot\max\{p,1\}}$. ☐

Having a small flip number is very useful for robustness, as our next two robustification techniques demonstrate.

2.2 The Sketch Switching Technique

Our first technique is called *sketch switching*, and is described in Algorithm 1. The technique maintains multiple instances of a static strong tracking algorithm, where each time step only one of the instances is "active". The idea is to change the current output of the algorithm very rarely. Specifically, as long as the current output is a good enough multiplicative approximation of the estimate of the active instance, the estimate we give to the adversary does not change, and the current instance remains active. As soon as this approximation guarantee is not satisfied, we update the output given to the adversary, deactivate our current instance, and activate the next one in line. By carefully exposing the randomness of our multiple instances, we show that the strong tracking guarantee (which a priori holds only in the static setting) can be carried into the robust setting. By Lemma 2.2, the required number of instances, corresponding to the 0-flip number of the outputs provided to the adversary, is controlled by the $(\Theta(\varepsilon), m)$-flip number of the problem.

Lemma 2.5 (Sketch Switching). *Fix any function $g : \mathbb{R}^n \to \mathbb{R}$ and let A be a streaming algorithm that for any $0 < \varepsilon < 1$ and $\delta > 0$ uses space $L(\varepsilon, \delta)$, and satisfies the*

Algorithm 1: Adversarially Robust g-estimation by Sketch Switching

```
λ ← λ_{ε/8,m}(g)
Initialize independent instances A_1, ..., A_λ of
  (ε/8, δ/λ)-strong g-tracking algorithm
ρ ← 1
g̃ ← g(0⃗)
while new stream update (a_k, Δ_k) do
    Insert update (a_k, Δ_k) into each algorithm
      A_1, ..., A_λ
    y ← current output of A_ρ
    if g̃ ∉ (1 ± ε/2)y then
        g̃ ← y
        ρ ← ρ + 1
    Output estimate g̃
end
```

$(\varepsilon/8, \delta)$-strong g-tracking property on the frequency vectors $f^{(1)}, \ldots, f^{(m)}$ of any particular fixed stream. Then Algorithm 1 is an adversarially robust algorithm for $(1+\varepsilon)$-approximating $g(f^{(t)})$ at every step $t \in [m]$ with success probability $1-\delta$, whose space is $O(L(\varepsilon/8, \delta/\lambda)\cdot\lambda)$, where $\lambda = \lambda_{\varepsilon/8,m}(g)$.

Proof. Note that for a fixed randomized algorithm $\mathcal{A}$ we can assume the adversary against $\mathcal{A}$ is deterministic without loss of generality (in our case, $\mathcal{A}$ refers to Algorithm 1). This is because given a randomized adversary and algorithm, if the adversary succeeds with probability greater than δ in fooling the algorithm, then by a simple averaging argument, there must exist a fixing of the random bits of the adversary which fools $\mathcal{A}$ with probability greater than δ over the coin flips of $\mathcal{A}$. Note also here that conditioned on a fixing of the randomness for both the algorithm and adversary, the entire stream and behavior of both parties is fixed.

We thus start by fixing such a string of randomness for the adversary, which makes it deterministic. As a result, suppose that y_i is the output of the streaming algorithm on step i. Then given $y_1, y_2, \ldots, y_k$ and the stream updates $(a_1, \Delta_1), \ldots, (a_k, \Delta_k)$ so far, the next stream update (a_{k+1}, Δ_{k+1}) is deterministically fixed. We stress that the randomness of the algorithm is not fixed at this point; we will gradually reveal it along the proof.

Let $\lambda = \lambda_{\varepsilon/8,m}(g)$ and let $A_1, \ldots, A_\lambda$ be the λ independent instances of an $(\varepsilon/8, \delta/\lambda)$-strong tracking algorithm for g. Since $\delta_0 = \delta/\lambda$, later on we will be able to union bound over the assumption that for all $\rho \in [\lambda]$, A_i satisfies strong tracking on some fixed stream (to be revealed along the proof); the stream corresponding to A_ρ will generally be different than that corresponding to ρ' for $\rho \neq \rho'$.

First, let us fix the randomness of the first instance, A_1. Let $u_1^1, u_2^1, \ldots, u_m^1$ be the updates $u_j^1 = (a_j, \Delta_j)$ that the adversary would make if $\mathcal{A}$ were to output $y_0 = g(\vec{0})$ at every time step, and let $f^{(t),1}$ be the stream vector after updates $u_1^1, \ldots, u_t^1$. Let $A_1(t)$ be the output of algorithm A_1 at time t of the stream $u_1^1, u_2^1, \ldots, u_t^1$. Let $t_1 \in [m]$ be the first time step such that $y_0 \notin (1 \pm \varepsilon/2)A_1(t_1)$, if exists (if not we can set, say, $t_1 = m+1$). At time $t = t_1$, we change our output to $y_1 = A_1(t_1)$. Assuming that A_1 satisfies strong tracking for g with approximation parameter $\varepsilon/8$ with respect to the fixed stream of updates $u_1^1, \ldots, u_m^1$ (which holds with probability at least $1-\delta/\lambda$), we know that $A_1(t) = (1 \pm \varepsilon/8) g(f^{(t)})$ for each $t < t_1$ and that $y_0 = (1 \pm \varepsilon/2) A_1(t)$. Thus, by the first part of Lemma 2.2, $y_0 = (1 \pm \varepsilon) g(f^{(t)})$ for any $0 \le t < t_1$. Furthermore, by the strong tracking, at time $t = t_1$ the output we provide $y_1 = A_1(t_1)$ is a $(1 \pm \varepsilon/8)$-approximation of the desired value $g(f^{(t_1)})$.

At this point, $\mathcal{A}$ "switches" to the instance A_2, and presents y_1 as its output as long as $y_1 = (1 \pm \varepsilon/2) A_2(t)$. Recall that randomness of the adversary is already fixed, and consider the sequence of updates obtained by concatenating $u_1^1, \ldots, u_{t_1}^1$ as defined above (these are the updates already sent by the adversary) with the sequence $u_{t_1+1}^2, \ldots, u_m^2$ to be sent by the adversary if the output from time $t = t_1$ onwards would always be y_1. We condition on the $\varepsilon/8$-strong g-tracking guarantee on A_2 holding for this fixed sequence of updates, noting that this is the point where the randomness of A_2 is revealed. Set $t = t_2$ as the first value of t (if exists) for which $A_2(t) = (1 \pm \varepsilon/2) y_1$ does not hold. We now have, similarly to above, $y_1 = (1 \pm \varepsilon) g(f^{(t)})$ for any $t_1 \le t < t_2$, and $y_2 = (1 \pm \varepsilon/8) g(f^{(t_2)})$.

The same reasoning can be applied inductively for A_ρ, for any $\rho \in [\lambda]$, to get that (provided $\varepsilon/8$-strong g-tracking holds for A_ρ) at any given time, the current output we provide to the adversary y_ρ is within a $(1 \pm \varepsilon)$-multiplicative factor of the correct output for any of the time steps $t = t_\rho, t_\rho + 1, \ldots, \min\{t_{\rho+1} - 1, m\}$. Taking a union bound, we get that with probability at least $1 - \delta$, all instances provide $\varepsilon/8$-tracking (each for its respective fixed sequence), yielding the desired $(1 \pm \varepsilon)$-approximation of our algorithm.

It remains to verify that this strategy will succeed in handling all m elements of the stream (and will not exhaust its pool of algorithm instances before then). Indeed, this follows immediately from Lemma 2.2 applied with $\bar{u} = ((g(f^{(0)}), \ldots, g(f^{(m)}))$, $\bar{v} = (g(f^{(0)}), A_1(1), \ldots, A_1(t_1), A_2(t_1 + 1), \ldots, A_2(t_2), \ldots)$, and $\bar{w}$ being the output that our algorithm $\mathcal{A}$ provides ($y_0 = g(f^{(0)})$ until time $t_1 - 1$, then y_1 until time $t_2 - 1$, and so on). Observe that indeed $\bar{w}$ was generated from v exactly as described in the statement of Lemma 2.2. $\square$

2.3 The Computation Paths Technique

With our sketch switching technique, we showed that maintaining multiple instances of a non-robust algorithm to estimate a function g, and switching between them when the rounded output changes, is a recipe for a robust algorithm to estimate g. We next provide another recipe, which keeps only one instance, whose success probability for any fixed stream is very high; it relies on the fact that if the flip number is small, then the total number of fixed streams that we should need to handle is also relatively small, and we will be able to union bound over all of them. Specifically, we show that any non-robust algorithm for a function with bounded flip number can be modified into an adversarially robust one by setting the failure probability δ small enough.

LEMMA 2.6 (COMPUTATION PATHS). *Fix $g\colon \mathbb{R}^n \to \mathbb{R}$ and suppose that the output of g uses $\log T$ bits of precision. Let A be a streaming algorithm that for any $\varepsilon, \delta > 0$ satisfies the (ε, δ)-strong g-tracking property on the frequency vectors $f^{(1)}, \ldots, f^{(m)}$ of any particular fixed stream. Then there is a streaming algorithm A' satisfying the following.*

1. *A' is an adversarially robust algorithm for $(1 + \varepsilon)$-approximating $g(f^{(t)})$ in all steps $t \in [m]$, with success probability $1 - \delta$.*
2. *The space complexity and running time of A' as above (with parameters ε and δ) are of the same order as the space and time of running A in the static setting with parameters $\varepsilon/8$ and $\delta_0 = \delta / \left(\binom{m}{\lambda} T^{O(\lambda)}\right)$, where $\lambda = \lambda_{\varepsilon/8,m}(g)$.*

PROOF. The algorithm A' that we construct runs by emulating A with the above parameters, and assuming that the output sequence of the emulated A up to the current time t is $v_0, \ldots, v_t$, it generates w_t in exactly the way described in Lemma 2.2: set $w_0 = v_0$, and for any $i > 0$, if $w_{i-1} \in (1 \pm \varepsilon/2) v_i$ then $w_i = w_{i-1}$, and otherwise $w_i = v_i$. The output provided to the adversary at time t would then be w_t.

As in the proof of Lemma 2.5, we may assume the adversary to be deterministic. This means, in particular, that the output sequence we provide to the adversary fully determines its stream of updates $(a_1, \Delta_1), \ldots, (a_m, \Delta_m)$. Take $\lambda = \lambda_{\varepsilon/8,m}(g)$. Consider the collection of all possible output sequences (with $\log T$ bits of precision) whose 0-flip number is at most λ, and note that the number of such sequences is at most $\binom{m}{\lambda} T^{O(\lambda)}$. Each output sequence as above uniquely determines a corresponding stream of updates for the deterministic adversary; let $\mathcal{S}$ be the collection of all such streams.

Pick $\delta_0 = \delta/|\mathcal{S}|$. Taking a union bound, we conclude that with probability $1 - \delta$, A (instantiated with parameters $\varepsilon/8$ and δ_0) provides an $\varepsilon/8$-strong g-tracking guarantee for all streams in $\mathcal{S}$. We fix the randomness of A, and assume this event holds.

At this point, the randomness of both parties has been revealed, which determines an output sequence $v_0, \ldots, v_m$ for the emulated A and the edited output, $w_0, \ldots, w_m$, that our algorithm A' provided to the adversary. The proof now follows by induction over the number t of stream updates that have been seen. The inductive statement is the following:

1. The sequence of outputs that the emulated algorithm A generates in response to the stream updates up to time t, $v_0, \ldots, v_t$, is a $(1 \pm \varepsilon/8)$-approximation of g over the stream up to that time.

2. The sequence of outputs that the adversary receives from A' until time t, $(w_0, \ldots, w_t)$, has 0-flip number at most λ (and is a prefix of a sequence in $\mathcal{S}$).

The base case, $t = 0$, is obvious; and the induction step follows immediately from Lemma 2.2. $\square$

Acknowledgments

The authors wish to thank Arnold Filtser for invaluable feedback. This work was done in part in the Simons Institute for the Theory of Computing. Part of this work was conducted while Omri Ben-Eliezer was at Tel Aviv University. Rajesh Jayaram and David P. Woodruff are supported by the Office of Naval Research (ONR) grant N00014-18-1-2562, and the National Science Foundation (NSF) under Grant No. CCF-1815840. Eylon Yogev is funded by the ISF grants 484/18, 1789/19, Len Blavatnik and the Blavatnik Foundation, The Blavatnik Interdisciplinary Cyber Research Center at Tel Aviv University, and The Raymond and Beverly Sackler Post-Doctoral Scholarship.

3. REFERENCES

[1] N. Alon, O. Ben-Eliezer, Y. Dagan, S. Moran, M. Naor, and E. Yogev. Adversarial laws of large numbers and optimal regret in online classification. *CoRR*, abs/2101.09054, 2021.

[2] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137 – 147, 1999.

[3] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68(4):702–732, 2004.

[4] O. Ben-Eliezer and E. Yogev. The adversarial robustness of sampling. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS, pages 49–62. ACM, 2020.

[5] J. Błasiok. Optimal streaming and tracking distinct elements with high probability. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA, pages 2432–2448. SIAM, 2018.

[6] J. Błasiok, J. Ding, and J. Nelson. Continuous monitoring of l_p norms in data streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, APPROX/RANDOM, pages 32:1–32:13, 2017.

[7] V. Braverman, S. R. Chestnut, N. Ivkin, J. Nelson, Z. Wang, and D. P. Woodruff. Bptree: An ℓ_2 heavy hitters algorithm using constant memory. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS, pages 361–376. ACM, 2017.

[8] A. Chakrabarti and S. Kale. Strong fooling sets for multi-player communication with applications to deterministic estimation of stream statistics. In *IEEE 57th Annual Symposium on Foundations of Computer Science*, FOCS, pages 41–50, 2016.

[9] P. Clifford and I. Cosma. A simple sketching algorithm for entropy estimation over streaming data. In *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics*, AISTATS, pages 196–206, 2013.

[10] D. J. DeWitt, J. F. Naughton, D. A. Schneider, and S. Seshadri. Practical skew handling in parallel joins. In *Proceedings of the 18th International Conference on Very Large Data Bases*, VLDB, pages 27–40, 1992.

[11] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum. Differential privacy under continual observation. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, STOC, page 715–724. ACM, 2010.

[12] S. Ganguly and D. P. Woodruff. High probability frequency moment sketches. In *45th International Colloquium on Automata, Languages, and Programming*, ICALP, pages 58:1–58:15, 2018.

[13] A. C. Gilbert, B. Hemenway, A. Rudra, M. J. Strauss, and M. Wootters. Recovering simple signals. In *2012 Information Theory and Applications Workshop*, pages 382–391. IEEE, 2012.

[14] A. C. Gilbert, B. Hemenway, M. J. Strauss, D. P. Woodruff, and M. Wootters. Reusable low-error compressive sampling schemes through privacy. In *2012 IEEE Statistical Signal Processing Workshop*, SSP, pages 536–539. IEEE, 2012.

[15] I. J. Good. C332. surprise indexes and p-values. *Journal of Statistical Computation and Simulation*, 32(1–2):90–92, 1989.

[16] M. Hardt and D. P. Woodruff. How robust are linear sketches to adaptive inputs? In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing*, STOC, pages 121–130, 2013.

[17] N. J. Harvey, J. Nelson, and K. Onak. Sketching and streaming entropy via approximation theory. In *49th Annual IEEE Symposium on Foundations of Computer Science*, FOCS, pages 489–498, 2008.

[18] A. Hassidim, H. Kaplan, Y. Mansour, Y. Matias, and U. Stemmer. Adversarially robust streaming algorithms via differential privacy. In *Advances in Neural Information Processing Systems 33*, NeurIPS, 2020.

[19] R. Jayaram and D. P. Woodruff. Data streams with bounded deletions. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS, pages 341–354. ACM, 2018.

[20] R. Jayaram and D. P. Woodruff. Towards optimal moment estimation in streaming and distributed models. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, APPROX/RANDOM, pages 29:1–29:21, 2019.

[21] A. Kamath, E. Price, and D. P. Woodruff. A simple proof of a new set disjointness with applications to data streams, 2020.

[22] D. M. Kane, J. Nelson, and D. P. Woodruff. On the exact space complexity of sketching and streaming small norms. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, SODA, pages 1161–1178, 2010.

[23] H. Kaplan, Y. Mansour, K. Nissim, and U. Stemmer. Separating adaptive streaming from oblivious streaming. *CoRR*, abs/2101.10836, 2021.

[24] Y. Li and D. P. Woodruff. A tight lower bound for high frequency moment estimation with small error. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, APPROX/RANDOM, pages 623–638. Springer, 2013.

[25] I. Mironov, M. Naor, and G. Segev. Sketching in adversarial environments. *SIAM Journal on Computing*, 40(6):1845–1870, 2011.

[26] S. Muthukrishnan. Data Streams: Algorithms and Applications. *Foundations and Trends in Theoretical Computer Science*, 1(2):117–236, 2005.

[27] M. Naor and E. Yogev. Bloom filters in adversarial environments. In *Advances in Cryptology - CRYPTO - 35th Annual Cryptology Conference*, pages 565–584, 2015.

[28] D. P. Woodruff and S. Zhou. Tight bounds for adversarially robust streams and sliding windows via difference estimators. *CoRR*, abs/2011.07471, 2020.