Optimal Query Complexities for Dynamic Trace Estimation

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

Fred Zhang
UC Berkeley
z0@berkeley.edu

Qiuyi (Richard) Zhang
Google Brain
qiuyiz@google.com

Abstract

We consider the problem of minimizing the number of matrix-vector queries needed for accurate trace estimation in the dynamic setting where our underlying matrix is changing slowly, such as during an optimization process. Specifically, for any m matrices $\mathbf{A}_1, ..., \mathbf{A}_m$ with consecutive differences bounded in Schatten-1 norm by α, we provide a novel binary tree summation procedure that simultaneously estimates all m traces up to ε error with δ failure probability with an optimal query complexity of $\widetilde{O}(m\alpha\sqrt{\log(1/\delta)}/\varepsilon + m\log(1/\delta))$, improving the dependence on both α and δ from Dharangutte and Musco (NeurIPS, 2021). Our procedure works without additional norm bounds on $\boldsymbol{A}_i$ and can be generalized to a bound for the p-th Schatten norm for $p \in [1, 2]$, giving a complexity of $\widetilde{O}(m\alpha(\sqrt{\log(1/\delta)}/\varepsilon)^p + m\log(1/\delta))$. By using novel reductions to communication complexity and information-theoretic analyses of Gaussian matrices, we provide matching lower bounds for static and dynamic trace estimation in all relevant parameters, including the failure probability. Our lower bounds (1) give the first tight bounds for Hutchinson's estimator in the matrix-vector product model with Frobenius norm error *even in the static setting*, and (2) are the first unconditional lower bounds for dynamic trace estimation, resolving open questions of prior work.

1 Introduction

Implicit matrix trace estimation is ubiquitous in numerical linear algebra and arises naturally in a wide range of applications, see, e.g., [25]. In this problem, we are given an oracle which gives us matrix-vector products $\boldsymbol{A}x_1, \boldsymbol{A}x_2, \cdots, \boldsymbol{A}x_m$ for an unknown $n \times n$ square matrix $\boldsymbol{A}$ and queries $x_1, \ldots, x_m$ of our choice, that may be chosen adaptively. In typical applications, one cannot afford to compute the diagonal entries of $\boldsymbol{A}$ explicitly, due to $\boldsymbol{A}$ being implicitly represented and computational constraints. The goal is to efficiently estimate $\mathrm{Tr}\,\boldsymbol{A}$ using only matrix-vector products.

In machine learning and data science, applications of trace estimation include training Gaussian Processes [8, 11], triangle counting [1], computing the Estrada Index [10, 9], and studying optimization landscapes of deep neural networks from Hessian matrices [12, 26]. In these applications, it is common that $\boldsymbol{A}$ is represented implicitly due to its large memory footprint. For example, while it is possible to compute Hessian-vector products via Pearlmutter's trick [20], it is prohibitive to compute or store the Hessian matrix $\boldsymbol{H}$, see, e.g., [12].

Moreover, $\boldsymbol{A}$ may be a matrix function f of another matrix $\boldsymbol{B}$ in some applications. Since computing $f(\boldsymbol{B})$ is expensive, it is desirable to apply implicit trace estimation. For example, during the training of Gaussian Processes, the marginal log-likelihood contains a heavy-computation term, i.e., the log of the determinant of the covariance matrix, $\log(\det(\mathbf{K}))$, where $\mathbf{K} \in \mathbb{R}^{n\times n}$ and n is the number of data points. The canonical way of computing $\log(\det(\mathbf{K}))$ is via a Cholesky factorization on $\mathbf{K}$, which takes $O(n^3)$ time. Instead, implicit trace estimation methods provide fast algorithms for

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

approximating $\log(\det(\mathbf{K})) = \sum_{i=1}^{n} \log(\lambda_i) = \mathrm{tr}(\log(\mathbf{K}))$ on large-scale data. Therefore, it is important to understand the fundamental limits of implicit trace estimation as the *query complexity*, i.e., the minimum number of matrix-vector multiplications required to achieve a desired accuracy and success rate.

Static trace estimation and Hutchinson's method. On the algorithmic side, Hutchinson's method [14] is a simple and widely used method for trace estimation. Let $\boldsymbol{Q} = [q_1, \ldots, q_\ell] \in \mathbb{R}^{n \times \ell}$ be ℓ vectors with i.i.d. standard Gaussian or Rademacher random variables. Given matrix-vector multiplication access to $\boldsymbol{A}$, Hutchinson's method estimates $\mathrm{tr}(\boldsymbol{A})$ by $t = \frac{1}{q}\sum_{i=1}^{q} q_i^T \boldsymbol{A} q_i = \frac{1}{q}\mathrm{tr}(\boldsymbol{Q}^T \boldsymbol{A} \boldsymbol{Q})$. It is known [2] that the estimator satisfies that for any $\varepsilon, \delta \in (0,1)$,

$$|t - \mathrm{Tr}\,\boldsymbol{A}| \leq \varepsilon \|\boldsymbol{A}\|_F, \text{ with probability at least } 1-\delta, \tag{1}$$

provided the number ℓ of queries satisfies $\ell \geq C \log(1/\delta)/\varepsilon^2$ for some fixed constant C.

For Hutchinson's method, there is also previous work which showed for queries of the form $x^\top \boldsymbol{A} x$, $\Omega(1/\varepsilon^2)$ queries are required [21]; however, this does not imply even a lower bound for non-adaptive algorithms that use matrix-vector queries. Though stronger algorithmic results and matching lower bounds are known for the important case of PSD matrices in the non-adaptive setting [18, 16], the optimality of Hutchinson's estimator as an trace estimator for general square matrices in the matrix-vector product model still remains an open problem. Notably, Hutchinson's method chooses the query vectors non-adaptively and it is furthermore unclear whether adaptivity could help.

More generally, there has been a flurry of recent work that gives trace estimators with $o(1/\varepsilon^2)$ query complexity but with a different error guarantee. Specifically, let us consider a Schatten-p norm error guarantee, where the goal is to provide an estimate t such that

$$|t - \mathrm{Tr}\,\boldsymbol{A}| \leq \varepsilon \|\boldsymbol{A}\|_p, \text{ with probability at least } 1-\delta, \tag{2}$$

where $\|\boldsymbol{A}\|_p$ denotes the Schatten-p norm.

For $p = 1$, a previous work [18] proposes a variance-reduced version of Hutchinson's method that uses only $O(1/\varepsilon)$ matrix-vector product queries to achieve a nuclear norm error of $\varepsilon\|\boldsymbol{A}\|_*$, in contrast to the $O(1/\varepsilon^2)$ queries used when the error is in the Frobenius norm. When the matrix is positive semidefinite (PSD), the nuclear norm error is equivalent to a $(1+\varepsilon)$ multiplicative approximation to the trace. Their work, along with a subsequent work [16], shows that $\Omega(1/\varepsilon)$ queries are therefore sufficient and necessary to achieve a $(1+\varepsilon)$ multiplicative trace approximation in this setting. While this line of work mainly focuses on PSD matrices and nuclear norm error, we consider trace estimation on general square matrices with Schatten-p norm error for any $p \in [1,2]$.

Furthermore, we note that the variance-reduced Hutchinson's method splits the queries between approximating the top $O(1/\varepsilon)$ eigenvalues, i.e., by computing a rank-$O(1/\varepsilon)$ approximation to $\boldsymbol{A}$, and performing Hutchinsons's method on the remainder. Due to the low rank approximation subroutine, the query complexity's dependence on the failure probability is more concretely $O(\sqrt{\log(1/\delta)}/\varepsilon + \log(1/\delta))$ for additive $\varepsilon\|\boldsymbol{A}\|_*$ error. The additive $\log(1/\delta)$ rate is shown to be necessary when non-adaptive queries are used, but it is an open problem whether adaptive queries can remove the additive $\log(1/\delta)$ term for trace estimation with Schatten-p norm error [16].

This motivates the natural question:

> *Question 1: Is Hutchinson's method optimal in terms of ε and δ for static trace estimation of general square matrices, even when adaptivity is allowed? How do we generalize Hutchinson's method for error in general Schatten-p norms?*

Dynamic trace estimation. In various applications the input matrix is not fixed. For example, during model training, we need to estimate the trace of a dynamically changing Hessian matrix with respect to some loss function. One may assume that the change at each step is not very large. Motivated by such a scenario, a recent work by Dharangutte and Musco [6] studies dynamic trace estimation.

Formally, let $p \in [1,2]$ and $\boldsymbol{A}_1, \boldsymbol{A}_2, \cdots, \boldsymbol{A}_m$ be $n \times n$ matrices in a stream such that (1) $\|\boldsymbol{A}_i\|_p \leq 1$ for all $i > 1$, where $\|\cdot\|_p$ denotes the Schatten-p norm, and (2) $\|\boldsymbol{A}_{i+1} - \boldsymbol{A}_i\|_p \leq \alpha < 1$ for all $i \leq m-1$. The goal is to output a sequence of estimates $t_1, \cdots, t_m$ such that for each $i \in [m]$,

$$|t_i - \mathrm{Tr}\,\boldsymbol{A}_i| \leq \varepsilon, \text{ with probability at least } 1-\delta, \tag{3}$$

Upper Bounds				
Prior Work	Query Complexity	Matrix Type	Failure Rate	Algorithm Type
[2, 22]	$O(\log(1/\delta)/\varepsilon^2)$	general square	δ	non-adaptive, $p=2$
[18]	$O(\sqrt{\log(1/\delta)}/\varepsilon + \log(1/\delta))$	PSD	δ	adaptive, $p=1$
[18]	$O(\log(1/\delta)/\varepsilon)$	PSD	δ	non-adaptive, $p=1$
[16]	$O(\sqrt{\log(1/\delta)}/\varepsilon + \log(1/\delta))$	PSD	δ	non-adaptive, $p=1$
This work [1]	$O((\sqrt{\log(1/\delta)}/\varepsilon)^p + \log(1/\delta))$	PSD	δ	non-adaptive, general p
Lower Bounds (Adaptive)				
[18]	$\Omega(1/(b+\varepsilon\log(1/\varepsilon)))$	general square, bit	constant	adaptive, $p=1$
This work [2]	$\Omega\left(\frac{1}{\varepsilon^p(b+\log(1/\varepsilon))} + \frac{\log(1/\delta)}{(b+\log\log(1/\delta))}\right)$	general square, bit	δ	adaptive, general p
This work [3]	$\Omega\left((\sqrt{\log(1/\delta)}/\varepsilon)^p\right)$	general square, ram	δ	adaptive, general p
Lower Bounds (Non-Adaptive)				
[18]	$\Omega(1/\varepsilon)$	PSD, ram	constant	non-adaptive, $p=1$
[16]	$\Omega(\sqrt{\log(1/\delta)}/\varepsilon + \frac{\log(1/\delta)}{\log\log(1/\delta)})$	PSD, ram	δ	non-adaptive, $p=1$
This work [4]	$\Omega\left(\log^{p/2}(1/\delta)/(\varepsilon^p(b+\log(1/\varepsilon)))\right)$	general square, bit	δ	non-adaptive, general p
This work [5]	$\Omega\left((\sqrt{\log(1/\delta)}/\varepsilon)^p + \frac{\log(1/\delta)}{\log\log(1/\delta)}\right)$	general square, ram	δ	non-adaptive, general p

Table 1: Upper and lower bounds on the query complexity for static trace estimation. In the bit complexity model, each entry of the query vector is specified by b bits, and the dependence on b is necessary.
[1]: A static upper bound generalizing Hutch++ [18] to Schatten-p norm error (Theorem C.1).
[2]: An adaptive lower bound via communication complexity of the Gap Equality and Approximate Orthogonality problem (Theorem 4.1), which combines Theorem D.2 and Theorem D.3, resolving an open problem that $\log(1/\delta)$ queries are required in the adaptive setting.
[3]: An adaptive lower bound via information-theoretic analysis of Gaussian Wigner matrices (Theorem 4.2), showing optimal dependence on $\log(1/\delta)$.
[4]: A non-adaptive lower bound via communication complexity of Augmented Indexing (Theorem F.1), optimal in all parameters up to the bit complexity term.
[5]: A non-adaptive lower bound combining our Theorem 4.2 and the prior result from [16].

via matrix-vector multiplication query access to the first i matrices $(\boldsymbol{A}_j)_{j=1}^i$. Naïvely, one could estimate each $\operatorname{Tr}\boldsymbol{A}_i$ independently using Hutchinson's method. This, however, does not exploit that the changes are bounded at each step. Alternatively, one can rewrite $\operatorname{Tr}\boldsymbol{A}_i$ as $\operatorname{Tr}\boldsymbol{A}_1 + \sum_{i=2}^i \operatorname{Tr}(\boldsymbol{\Delta}_i)$, where $\boldsymbol{\Delta}_i = \boldsymbol{A}_i - \boldsymbol{A}_{i-1}$, by linearity of the trace, and apply Hutchinson's method on each term. Unfortunately, this scheme suffers from an accumulation of errors over the steps.

The prior work [6] is focused on $p = \{1, 2\}$ and improves upon the naïve ideas above. For $p = 1$, the authors give a method that uses $O\left(m\sqrt{\alpha/\delta}/\varepsilon + \sqrt{1/\delta}/\varepsilon\right)$ queries. For $p = 2$, they provide an algorithm with query complexity $O(m\alpha\log(1/\delta)/\varepsilon^2 + \log(1/\delta)/\varepsilon^2)$ and a *conditional* lower bound showing that this is tight. This leaves open the question:

> *Question 2: Can we design improved algorithms for dynamic trace estimation under a general Schatten norm assumption? Can we prove an unconditionally optimal lower bound?*

1.1 Our Results

Our work resolves the proposed questions (nearly) optimally, and we next discuss our main results.

Static trace estimation: For Question 1, we prove query complexity lower bounds for implicit trace estimation in both bit complexity and real RAM models of computation, resolving the open problem of establishing unconditional lower bounds for the optimality of Hutchinson's method even in the adaptive setting.

To do so, we provide new reductions from classic communication complexity problems, including GAP-EQUALITY and APPROXIMATE-ORTHOGONALITY, to matrix trace estimation. Our main lower bounds demonstrate that $\log(1/\delta)$ queries are always needed even with adaptivity and for general p, there is an additional $1/\varepsilon^p$ dependence. A key idea is a communication protocol simulation

using the product of two matrices rather than the sum, as was used in prior work on PSD lower bounds [18].

Theorem 1.1 (Informal; see Theorem 4.1)**.** *In the bit complexity model, where each entry of each query vector is specified using b bits,*

$$\Omega\left(\frac{1}{\varepsilon^p(b+\log(1/\varepsilon))}+\frac{\log(1/\delta)}{b+\log\log(1/\delta)}\right)$$

number of adaptive queries is necessary to achieve $\varepsilon\|\boldsymbol{A}\|_p$ error with probability at least $1-\delta$.

When adaptivity is not allowed, we give a stronger lower bound (Theorem F.1) of $\Omega(\log^{p/2}(1/\delta)/\varepsilon^p)$. This matches the guarantee of Hutchinson's non-adaptive estimator up to a constant factor, for which random sign vectors suffice and so one can take $b = O(1)$.

We also provide a query complexity lower bound in the real RAM model for general Schatten p-norms with $p \in [1,2]$ by using Gaussian ensembles and controlling the remaining entropy of the distribution conditioned on prior queries. In the special case of $p = 2$ (i.e., Frobenius norm error guarantee), our bound again matches the classic Hutchinson's method up to a constant factor for $p = 2$, and an additive $\log(1/\delta)$ factor for $p < 2$. Note that in the non-adaptive setting, our lower bound in the RAM model can also be improved for $p < 2$ to include a $\log(1/\delta)/\log\log(1/\delta)$ factor. Therefore, this lower bound emphasizes that our dependence on $\log(1/\delta)$ in the ε-dependent term is tight, even in the adaptive setting.

Theorem 1.2 (Informal; see Theorem 4.2)**.** *In the real RAM model, where the queries are real-valued, for sufficiently small ε and any $p \in [1,2]$, $\Omega\left(\left(\sqrt{\log(1/\delta)}/\varepsilon\right)^p\right)$ number of adaptive queries is necessary to achieve $\varepsilon\|\boldsymbol{A}\|_p$ error with probability at least $1-\delta$.*

On the algorithmic front, we give a matching upper bound for static trace estimation for general Schatten-p norm error for $p \in [1,2]$. The argument requires a careful balancing of the ε and δ parameters in the low rank approximation of the Hutch++ procedure from [18]. See Theorem C.1 for a full statement.

Dynamic trace estimation: To answer Question 2, we first give an improved algorithm for dynamic trace estimation that uses a binary tree-based decomposition to estimate all matrix traces with only a small logarithmic overhead. The algorithm improves upon the previous work [6] and gets an optimal dependence on $0 < \alpha, \delta < 1$, up to logarithmic factors. Specifically, for $p = 1$, the prior work gives a method that uses $O\left(m\sqrt{\alpha/\delta}/\varepsilon\right)$ queries for small ε, while our algorithm gives an improved $O(m\alpha\sqrt{\log(1/\delta)}/\varepsilon)$ bound with a linear dependence on α and square root dependence on $\log(1/\delta)$. For $p = 2$, our algorithm matches the query complexity of $O(m\alpha\log(1/\delta)/\varepsilon^2)$ given by previous work. Furthermore, our algorithm works under a general Schatten p-norm assumption for any $p \in [1,2]$:

Theorem 1.3 (Informal; see Theorem 3.1 and Theorem C.2)**.** *For any $p \in [1,2]$, there is a dynamic trace estimation algorithm that achieves error ε and failure rate δ at each step. The algorithm uses a total of*

$$\widetilde{O}\left((m\alpha+1)\left(\sqrt{\log(1/(\alpha\delta))}/\varepsilon\right)^p + m\log(1/(\alpha\delta))\right) \tag{4}$$

matrix-vector product queries. Furthermore, for $p = 1$, it can be improved to

$$\widetilde{O}\left((m\alpha+1)\left(\sqrt{\log(1/(\alpha\delta))}/\varepsilon\right) + m\min(1,\alpha/\varepsilon)\log(1/(\alpha\delta))\right) \tag{5}$$

Furthermore, since our algorithm avoids the variance reduction technique from [6], we may relax the assumptions of dynamic trace estimation and require only the first matrix to have norm $\|\boldsymbol{A}_1\| \le 1$, instead of asking the entire sequence $\boldsymbol{A}_i$ to be bounded in such a way. While the norm bound on all $\boldsymbol{A}_i$ is crucial for the algorithm in [6] (rerunning the analysis naïvely would give a worse query complexity of $O(m^3\alpha^3/\varepsilon)$), our tree-based algorithm achieves a nearly optimal query complexity even when the norm of $\boldsymbol{A}_i$ grows, and we suffer only a $\log m$ overhead in that case. Moreover, in our experiments, we find that our algorithm significantly outperforms previous algorithms on real and synthetic datasets. See Section 6 for our experimental results.

To complement our algorithms, we give unconditional lower bounds showing that our algorithm is nearly optimal. Our lower bounds rely on a reduction from dynamic trace estimation to static matrix trace estimation from [6] and make use of our new lower bounds in the static setting. In particular, the reduction shows that if for a fixed set of parameters ε, δ, p, a static trace estimation scheme requires $\Omega(r)$ queries, then $\Omega(m\alpha r)$ queries are necessary for any dynamic algorithm. Combining this observation with our static trace estimation lower bounds, we get:

Theorem 1.4 (Informal; see Theorem 5.2 and Theorem 5.3)**.** *For any $p = [1, 2)$, our algorithm attains the optimal query complexity, up to bit complexity and logarithmic terms.*

More specifically, we prove lower bounds that match the first term in our upper bound (4) for all $p \in [1, 2]$. For $p = 1$, we give a lower bound (Theorem 5.4) matching the the second term in (5) as well, showing that the $m(\log(1/\delta))$ additive dependence is necessary.

For $p = 2$, the prior work [6] gives a upper bound of $O(m\alpha \log(1/\delta)/\varepsilon^2 + \log(1/\delta)/\varepsilon^2)$. Our lower bounds are unconditional and show that the first term is tight. Moreover, the second term is necessary due to the static lower bound when $m = 1$. This result is not contradicted by the claim of Theorem 5.4. In particular, when $\alpha \geq \varepsilon^2$, Theorem 5.4 is weaker than the $\Omega(m\alpha \log(1/\delta)/\varepsilon^2)$ lower bound; and when $\alpha < \varepsilon^2$, the construction by itself requires ε/α update steps to change the trace by ε, which leads to a lower bound of $\Omega(m\alpha \log(1/\delta)/\varepsilon)$, again weaker than $\Omega(m\alpha \log(1/\delta)/\varepsilon^2)$.

2 Preliminaries

A matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is symmetric positive semi-definite (PSD) if it is real, symmetric and has non-negative eigenvalues. Hence, $x^\top Ax \geq 0$ for all $x \in \mathbb{R}^n$. Let $\operatorname{tr}(\boldsymbol{A}) = \sum_{i=1}^n \boldsymbol{A}_{ii}$ denote the trace of $\boldsymbol{A}$. Let $\|\boldsymbol{A}\|_F = (\sum_{i=1}^n \sum_{j=1}^n \boldsymbol{A}_{ij}^2)^{1/2}$ denote the Frobenius norm and $\|\boldsymbol{A}\|_{op} = \sup_{\|\mathbf{v}\|_2=1} \|\boldsymbol{A}\mathbf{v}\|_2$ denote the operator norm of $\boldsymbol{A}$. We let $\|\boldsymbol{A}\|_p = (\sum_i \sigma_i^p)^{1/p}$ be the Schatten-p norm, where σ_i are the singular values of $\boldsymbol{A}$. Two special cases are the Frobenius norm, which equals the Schatten-2 norm ($\|\boldsymbol{A}\|_F = \|\boldsymbol{A}\|_2$) and the nuclear norm, equals the Schatten-1 norm ($\|\boldsymbol{A}\|_\star = \|\boldsymbol{A}\|_1$).

3 Algorithm for Dynamic Trace Estimation

We give an algorithm for dynamic trace estimation under a general Schatten-p norm assumption, for $p \in [1, 2]$. For $p = 1$, our algorithm provides an improved guarantee upon the DeltaShift++ procedure from [6]. In a later section we complement the result by showing that it is indeed near-optimal. Specifically, we give an algorithm that achieves the following guarantees:

Theorem 3.1 (Improved dynamic trace estimation)**.** *Let $\boldsymbol{A}_1, \boldsymbol{A}_2, \cdots, \boldsymbol{A}_m$ be $n \times n$ matrices such that (1) $\|\boldsymbol{A}_i\|_\star \leq 1$ for all i, and (2) $\|\boldsymbol{A}_{i+1} - \boldsymbol{A}_i\|_\star \leq \alpha$ for all $i \leq m - 1$. Given matrix-vector multiplication access to the matrices, a failure rate $\delta > 0$ and error bound ε, there is an algorithm that outputs a sequence of estimates $t_1, \cdots, t_m$ such that for each $i \in [m]$,*

$$|t_i - \operatorname{Tr} \boldsymbol{A}_i| \leq \varepsilon, \text{ with probability at least } 1 - \delta. \tag{6}$$

The algorithm uses a total of

$$O\left((m\alpha + 1)\log^2(1/\alpha)\sqrt{\log(1/(\alpha\delta))}/\varepsilon + m\min(1, \alpha/\varepsilon)\log(1/(\alpha\delta))\right) \tag{7}$$

matrix-vector multiplication queries to $\boldsymbol{A}_1, \boldsymbol{A}_2, \cdots, \boldsymbol{A}_m$.

Compared with DeltaShift++ in [6], this guarantee provides an exponential improvement in δ and a polynomial improvement in α for $p \neq 2$, while maintaining the optimal dependence on m and ε.

3.1 Algorithm

We now describe our algorithm. The first idea is to partition the m updates into groups of size $s = \lceil 1/(2\alpha) \rceil$. Each group will be treated independently, and we will use

$$O\left(\log^2(1/\alpha)\sqrt{\log(1/(\alpha\delta))}/\varepsilon + \frac{1}{\alpha}\log(1/(\alpha\delta))\right). \tag{8}$$

queries on each group. This leads to our claimed query complexity, as there are $O(m\alpha)$ groups. Note that if $\alpha < \varepsilon$, since $|\operatorname{Tr}(\boldsymbol{A}_j - \boldsymbol{A}_{j-1})| \le \|\boldsymbol{A}_j - \boldsymbol{A}_{j-1}\|_* \le \alpha$, the trace can change by at most an additive α, so we can simply ignore every subsequence of length ε/α. Therefore, we only need to apply our estimators to $m\alpha/\varepsilon$ matrices.

Without loss of generality, consider a group of matrices $\boldsymbol{A}_1, \cdots, \boldsymbol{A}_{1/2\alpha}$. As the first step, we estimate $\operatorname{Tr}(\boldsymbol{A}_j - \boldsymbol{A}_{j-1})$ for each $j \ge 2$ by using the Hutch++ static trace estimator [18] as a black box. Then, for each even integer $j = 2k$ (for an integer $2 \le k \le s/2$), we also estimate $\operatorname{Tr}(\boldsymbol{A}_{2k} - \boldsymbol{A}_{2(k-1)})$ in the same way. More generally, for each integer $j = 2^\ell k$, for $0 \le \ell < \log_2 s$, we use Hutch++ to approximate $\operatorname{Tr}\left(\boldsymbol{A}_{2^\ell k} - \boldsymbol{A}_{2^\ell(k-1)}\right)$. We view this scheme as a binary tree: the bottom level consists of leaves corresponding to the trace difference of neighboring matrices, and nodes at level ℓ correspond to the trace difference of matrices that are 2^ℓ apart in their indices.

To output an estimate of $\operatorname{Tr} \boldsymbol{A}_i$, we will write i in its binary representation and approximate it by $\operatorname{Tr}(\boldsymbol{A}_1)$ plus a sequence of $O(\log(1/\alpha))$ differences, at most one for each level in the binary tree. By setting the success rates and errors bounds at each level carefully, we can achieve the desired error guarantee of Equation (6).

To formalize the construction, we first cite the following guarantee of the Hutch++ algorithm:

Lemma 3.2 (Hutch++, nuclear norm, Theorem 5 of [18])**.** *The Hutch++ estimator uses*

$$N = O\left(\sqrt{\log(1/\delta')}/\varepsilon' + \log(1/\delta')\right)$$

matrix-vector multiplication queries such that given any square matrix $\boldsymbol{A}$ and parameters ε', δ', with probability at least $1 - \delta'$, the algorithm's output t satisfies

$$|t - \operatorname{Tr} \boldsymbol{A}| \le \sqrt{\varepsilon'} \left\|\boldsymbol{A} - \boldsymbol{A}_{1/\varepsilon'}\right\|_F \le \varepsilon' \|\boldsymbol{A}\|_*. \tag{9}$$

Let $\texttt{Hutch++}(\boldsymbol{A}, \varepsilon', \delta')$ denote the output of Hutch++ on matrix $\boldsymbol{A}$ with parameters ε', δ'. It will be invoked with different parameters at different levels of the binary tree construction. A description of the algorithm is given by the pseudocode Algorithm 1, with a helper function Algorithm 2.

For simplicity of analysis, note that since we can add dummy matrices (say, extra copies of $\boldsymbol{A}_1$), we assume that each group has size exactly $s = \lceil 1/(2\alpha) \rceil$ and s is a power of two. This blows up the total number of matrices by at most a constant factor.

Algorithm 1: Improved Dynamic Trace Estimation

Input : A sequence of square matrices $(\boldsymbol{A}_i)_{i=0}^m \in \mathbb{R}^{n \times n}$, failure rate δ, error bound ε
Ouput: Trace estimate t_i for each matrix

Partition the matrices into groups of size $s = \lceil 1/(2\alpha) \rceil$.
For every $g \ge 0$ and $i \in \{0, 1, \cdots, s-1\}$, let $\boldsymbol{A}_i^{(g)} = \boldsymbol{A}_{gs+i+1}$ denote the i-th matrix in the g-th group.
for *each group* $\boldsymbol{A}_0^{(g)}, \cdots, \boldsymbol{A}_{s-1}^{(g)}$ *independently* **do**
    Let $t_0 = \texttt{Hutch++}(\boldsymbol{A}_0^{(g)}, \varepsilon/2, \delta/2))$
    for *each level* ℓ *from* 0 *to* $\log_2 s - 1$ **do**
        $\mathsf{gap} = 2^\ell$
        for k *from* 1 *to* $(s-1)/\mathsf{gap}$ **do**
            Compute $t_{k,\ell} = \texttt{Hutch++}\left(\boldsymbol{A}_{k\cdot\mathsf{gap}} - \boldsymbol{A}_{(k-1)\cdot\mathsf{gap}}, \varepsilon'(\ell), \delta'\right)$, with $\varepsilon'(\ell) = \varepsilon/(2^{\ell+1}\alpha \log_2 s)$ and $\delta' = \alpha\delta$.
    Output $t_{gs+i+1} = t_0 + \textsc{SumTree}(1, i, \log_2 s - 1, t)$ for each $i \in [0, s-1]$.

3.2 Analysis

The analysis of the algorithm is rather lengthy and is delayed to Appendix C.1. In addition, we give a general analysis of the algorithm under Schatten-p norm assumption and the specific improved bounds for $p = 1$ in Appendix C.2 and show how to relax the bounded norm assumption in Appendix C.3.

Algorithm 2: SUMTREE: Helper Function for Tracing the Binary Tree

Input: Indices i, j, level ℓ, binary tree node values t

$\mathsf{gap} = 2^\ell$
if $j \le i$ **then**
  return 0.
if $\mathit{gap} = 1$ **then**
  return $t_{\ell,i}$.
if $j - i \ge \mathit{gap}$ **then**
  return $t_{\ell,\lfloor (j-1)/\mathsf{gap} \rfloor} + \textsc{SumTree}(i + \mathsf{gap}, j, \ell - 1, t)$.
else
  return $\textsc{SumTree}(i, j, \ell - 1, \mathsf{gap})$.

4 Lower Bounds for Adaptive Trace Estimation

In this section, we provide (nearly) optimal lower bounds for trace estimation with adaptive matrix-vector multiplication queries, under general square matrices and Schatten-p norm error.

4.1 Adaptive Lower Bound, Bit Complexity

First, we show two separate lower bounds under bit complexity model, both proven via reductions from communication complexity problems. One shows an $\Omega(1/\varepsilon^p)$ lower bound (Theorem D.2) and the other $\Omega(\log(1/\delta))$ (Theorem D.3), up to bit complexity terms. Combined together, they yield:

Theorem 4.1 (Adaptive query lower bound, bit complexity)**.** *Any algorithm that accesses a square matrix $\boldsymbol{A}$ via matrix-vector multiplication queries requires at least*

$$\Omega\left(\frac{1}{\varepsilon^p(k+\log(1/\varepsilon))} + \frac{\log(1/\delta)}{k+\log\log(1/\delta)}\right)$$

queries to output an estimate t such that with probability at least $1-\delta$, $|t - \operatorname{Tr}\boldsymbol{A}| \le \varepsilon\|\boldsymbol{A}\|_p$, for any $p \in [1,2]$, where the query vectors may be adaptively chosen with entries specified by k bits.

The proofs of the theorems can be found in Appendix D.1.

4.2 Adaptive Lower Bound, RAM

Next, we prove a tight lower bound under the real RAM model (Theorem 4.2). The bounds hold for any Schatten-p norm error. Our proof is via information-theoretic analysis of random Gaussian matrices and is delayed to Appendix D.2.

Theorem 4.2 (Lower Bound for Any Schatten Norm)**.** *For all $p \in [1,2]$, $\delta > 0$ and $0 < \varepsilon < (\log(1/\delta))^{1/2-1/p}$, any algorithm that takes in any input matrix $\boldsymbol{A}$ and succeeds with probability at least $1-\delta$ in outputting an estimate t such that $|t - \operatorname{tr}(\boldsymbol{A})| \le \varepsilon\|\boldsymbol{A}\|_p$ requires*

$$m = \Omega\left(\left(\frac{\sqrt{\log(1/\delta)}}{\varepsilon}\right)^p\right)$$

matrix-vector multiplication queries.

5 Lower Bounds for Dynamic Trace Estimation

Using the query complexity lower bounds for adaptive trace estimation, we can now prove tight lower bounds for dynamic trace estimation. The recent work of Dharangutte and Musco [6] only provides a *conditional* lower bound, assuming that Hutchinson's scheme is optimal. We remove this assumption and make the lower bound unconditional. We additionally prove a lower bound by constructing an explicit hard instance in the dynamic setting. Our lower bounds hold under a general Shatten norm assumption and nearly matches the guarantee of our algorithm.

5.1 Lower Bounds via Static-to-Dynamic Reduction

We first show a lower bound for dynamic trace estimation under a Frobenius norm assumption. This immediately implies that the DeltShift algorithm due to [6] is optimal for $p = 2$.

First, we cite a static-to-dynamic reduction from [6] and its implication. The reduction shows how to solve a static instance using a dynamic trace estimation scheme, and therefore any hardness on the static problem translates to the dynamic setting as well. It holds generally for an error bound in any Schatten norm. For completeness, we give a proof in Appendix E.1.

Lemma 5.1 (Conditional lower bound for dynamic trace estimation [6])**.** *Suppose that any algorithm that achieves Equation (2) for static trace estimation must use $\Omega(r)$ matrix-vector product queries. Then any dynamic trace estimation algorithm requires $\Omega(r\alpha m)$ matrix-vector product queries under a general Schatten-p norm assumption, when $\alpha = 1/(m-1)$.*

It follows immediately from this lemma and our adaptive query lower bound (Theorem 4.1):

Theorem 5.2 (Unconditional lower bound for dynamic trace estimation, bit)**.** *For all $p \in [1, 2]$ and $\varepsilon, \delta \in (0, 1)$, any algorithm for dynamic trace estimation under a Schatten-p norm assumption must use at least*

$$\Omega\left(\alpha m\left(\frac{1}{\varepsilon^p(k+\log(1/\varepsilon))}+\frac{\log(1/\delta)}{k+\log\log(1/\delta)}\right)\right)$$

matrix-vector multiplication queries, where each entry of the query vectors is specified by k bits.

Combining the same reduction (Theorem D.6) with our previous real RAM lower bound (Theorem 4.2) in the static setting gives:

Theorem 5.3 (Unconditional lower bound for dynamic trace estimation, RAM)**.** *For all $p \in [1, 2]$, $\delta > 0$ and $0 < \varepsilon < (\log(1/\delta))^{1/2-1/p}$, any algorithm for dynamic trace estimation under a Schatten-p norm assumption must use at least $\Omega\left(\alpha m\left(\sqrt{\log(1/\delta)}/\varepsilon\right)^p\right)$ matrix-vector multiplication queries.*

5.2 Lower Bound via Explicit Hard Instance

Using the hard instance based on GAP-EQUALITY in the static setting (from the proof of Theorem D.3), we give an explicit hardness construction against any dynamic trace estimation scheme. This yields the following lower bound, and its proof is in Appendix E.2.

Theorem 5.4. *For all $p \in [1, 2]$ and $\varepsilon, \delta \in (0, 1/4)$, any algorithm for dynamic trace estimation under Schatten-p norm assumption must use at least*

$$\Omega\left(m\min\left(1,\frac{\alpha}{\varepsilon}\right)\frac{\log(1/\delta)}{k+\log\log(1/\delta)}\right)$$

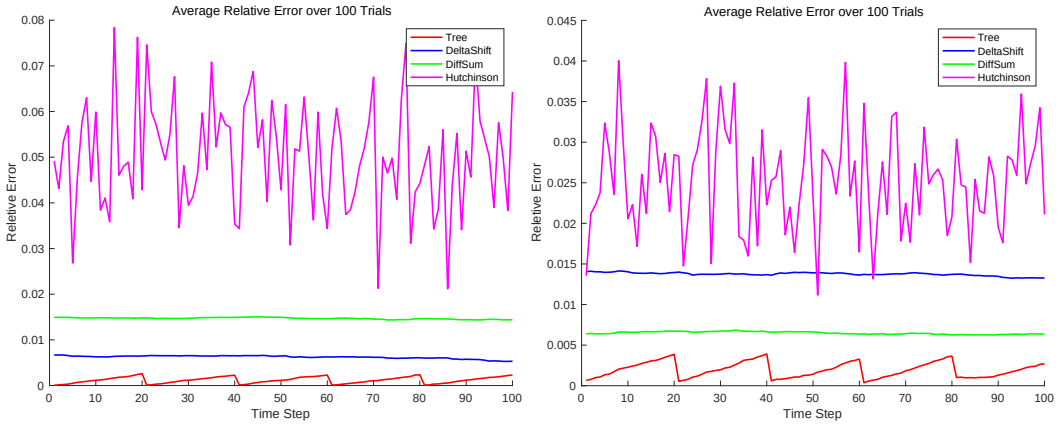
matrix-vector multiplication queries, where each entry of the query vectors is specified by k bits.

6 Experiments

We experimentally validate our algorithmic results. We compare Algorithm 1, with the following procedures on both synthetic and real datasets. More experimental details are in Appendix G.

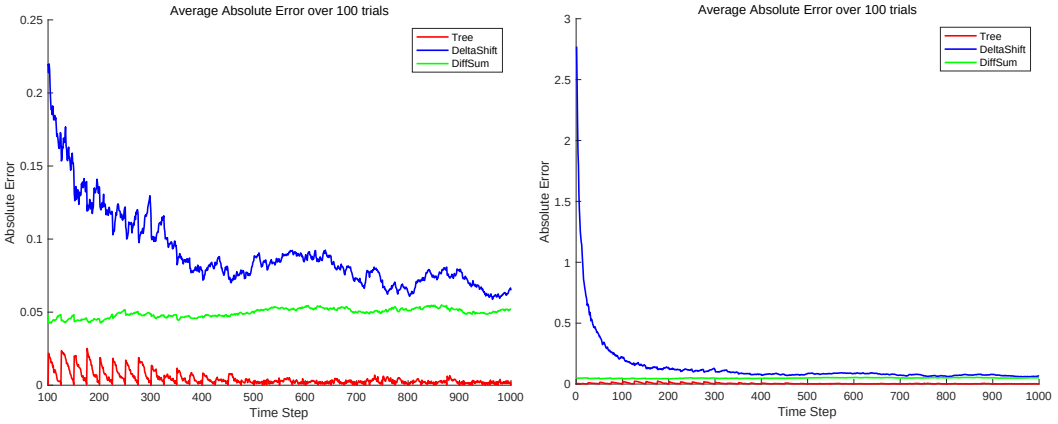
- Hutchinson's: Apply the classic Hutchinson's scheme for each $\mathrm{Tr}(A_i)$ independently.
- DiffSum: Approximate $t_1 \approx \mathrm{Tr}(\boldsymbol{A}_1)$ and each neighboring difference $d_i \approx \mathrm{Tr}(\boldsymbol{A}_i) - \mathrm{Tr}(\boldsymbol{A}_{i-1})$ using Hutchinson's independently. Then output $t_i = t_1 + \sum_{j=2}^{i} d_j$.
- DeltaShift: The main algorithm of [6]. The experiments from [6] demonstrate that DeltaShift outperforms DiffSum and other Hutchinson-based schemes on various datasets.

Synthetic data. We simulate a dynamic trace estimation instance by first generating a (symmetric) random matrix $\boldsymbol{A}^{n\times n}$ and then adding random perturbations over $T = 100$ time steps. The details and results are found in Appendix G.1.



(a) SNAP/ca-GrQc (b) SNAP/ca-HepTh

Figure 1: ArXiv datasets. Query budget is $8,000$. In this experiment, the trace values are large, so we measure the performance of the algorithms by their relative error $|t_i - \operatorname{Tr} \boldsymbol{A}_i^3| / \max_i \operatorname{Tr} \boldsymbol{A}_i^3$.



(a) Error over last 900 time steps (b) Error over entire $1,000$ time steps

Figure 2: MNIST. Query budget is $50,000$.

Counting triangles. Our first experiment on a real-world dataset is on counting triangles in dynamic undirected (simple) graphs. Note that the number of triangles in a graph equals $\frac{1}{6} \operatorname{Tr} \boldsymbol{A}^3$, where $\boldsymbol{A}$ is the adjacency matrix of the graph. Thus, triangle counting reduces to trace estimation.

We use two arXiv collaboration networks with $5,242$ and $9,877$ nodes [17].[1] The nodes represent authors, and edges indicate co-authorships. To simulate a real-world scenario, we add a random clique of size at most 6 to the graph in each step, indicating a group of researchers jointly publishing a paper. We note that our algorithm significantly outperforms other methods (Figure 1).

Neural network weight matrix. We evaluate the performance of the algorithms on a sequence of weight matrices of a neural network, generated during the training process. In particular, we choose a three-layer neural network with a hidden layer of 100×100. We train the network on the MNIST dataset via mini-batch SGD and consider the first $1,000$ steps, when the weights are changing most rapidly. Our algorithm achieves much smaller error than DiffSum and DeltaShift (Figure 2).

[1] The first is the collaboration network of arXiv General Relativity (ca-GrQc) and the second High Energy Physics Theory (ca-HepTh). Both are available at https://sparse.tamu.edu/SNAP.

Acknowledgement

Work done while David P. Woodruff and Fred Zhang were at Google Research in Pittsburgh.

References

[1] Haim Avron. Counting triangles in large graphs using randomized matrix trace estimation. In *Workshop on Large-scale Data Mining: Theory and Applications*, volume 10, page 9, 2010.

[2] Haim Avron and Sivan Toledo. Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *J. ACM*, 58(2), 2011.

[3] Kai Bergermann and Martin Stoll. Fast computation of matrix function-based centrality measures for layer-coupled multiplex networks. *Physical Review E*, 105(3):034305, 2022.

[4] Harry Buhrman, Richard Cleve, and Avi Wigderson. Quantum vs. classical communication and computation. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing (STOC)*, 1998.

[5] Amit Chakrabarti, Ranganath Kondapally, and Zhenghui Wang. Information complexity versus corruption and applications to orthogonality and gap-hamming. In *Approximation, Randomization, and Combinatorial Optimization (APPROX-RANDOM)*, 2012.

[6] Prathamesh Dharangutte and Christopher Musco. Dynamic trace estimation. *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021.

[7] Edoardo Di Napoli, Eric Polizzi, and Yousef Saad. Efficient estimation of eigenvalue counts in an interval. *Numerical Linear Algebra with Applications*, 23(4):674–692, 2016.

[8] Kun Dong, David Eriksson, Hannes Nickisch, David Bindel, and Andrew G Wilson. Scalable log determinants for gaussian process kernel learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[9] Ernesto Estrada. Characterization of 3d molecular structure. *Chemical Physics Letters*, 319(5-6):713–718, 2000.

[10] Ernesto Estrada and Naomichi Hatano. Communicability in complex networks. *Phys. Rev. E*, 77:036111, 2008.

[11] Jack K. Fitzsimons, Diego Granziol, Kurt Cutajar, Michael A. Osborne, Maurizio Filippone, and Stephen J. Roberts. Entropic trace estimates for log determinants. In *ECML/PKDD*, 2017.

[12] Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

[13] Insu Han, Dmitry Malioutov, Haim Avron, and Jinwoo Shin. Approximating the spectral sums of large-scale matrices using chebyshev approximations. *SIAM Journal on Scientific Computing*, 39, 06 2016.

[14] Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

[15] Thathachar S Jayram and David P Woodruff. Optimal bounds for johnson-lindenstrauss transforms and streaming problems with subconstant error. *ACM Transactions on Algorithms (TALG)*, 9(3):1–17, 2013.

[16] Shuli Jiang, Hai Pham, David Woodruff, and Richard Zhang. Optimal sketching for trace estimation. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[17] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1), 2007.

[18] Raphael A Meyer, Cameron Musco, Christopher Musco, and David P Woodruff. Hutch++: Optimal stochastic trace estimation. In *Symposium on Simplicity in Algorithms (SOSA)*, 2021.

[19] Cameron Musco, Praneeth Netrapalli, Aaron Sidford, Shashanka Ubaru, and David P Woodruff. Spectrum approximation beyond fast matrix multiplication: Algorithms and hardness. In *9th Innovations in Theoretical Computer Science Conference (ITCS))*, 2018.

[20] Barak A. Pearlmutter. Fast exact multiplication by the hessian. *Neural Computation*, 6:147–160, 1994.

[21] Farbod Roosta-Khorasani and Uri Ascher. Improved bounds on sample size for implicit matrix trace estimators. *Foundations of Computational Mathematics*, 15(5):1187–1212, 2015.

[22] Farbod Roosta-Khorasani and Uri Ascher. Improved bounds on sample size for implicit matrix trace estimators. *Found. Comput. Math.*, 15(5):1187–1212, October 2015.

[23] Mark Rudelson and Roman Vershynin. Non-asymptotic theory of random matrices: extreme singular values. In *Proceedings of the International Congress of Mathematicians 2010 (ICM 2010) (In 4 Volumes) Vol. I: Plenary Lectures and Ceremonies Vols. II–IV: Invited Lectures*, pages 1576–1602. World Scientific, 2010.

[24] Max Simchowitz, Ahmed El Alaoui, and Benjamin Recht. Tight query complexity lower bounds for pca via finite sample deformed wigner law. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, 2018.

[25] Shashanka Ubaru and Yousef Saad. Applications of trace estimation techniques. In *High Performance Computing in Science and Engineering*, 2018.

[26] Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael W Mahoney. Pyhessian: Neural networks through the lens of the hessian. In *2020 IEEE International Conference on Big Data (BigData)*, 2020.

[27] Yuchen Zhang, Martin Wainwright, and Michael Jordan. Distributed estimation of generalized matrix rank: Efficient algorithms and lower bounds. In *International Conference on Machine Learning (ICML)*, 2015.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes]
(b) Did you mention the license of the assets? [No]
(c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A More Related Work

We summarize prior work on static trace estimation in Table 1. The seminal work of [2] gives the first analysis of Huchinson's estimator, which was improved by [22]. For PSD matrices, the query complexity can be sharpened, and this was shown recently in [18, 16]. These two papers also give matching lower bounds. The study of dynamic trace estimation was initiated by [6], and our work improves upon their results.

Other applications of implicit trace estimation include inference of Determinantal Point Processes [8], approximating the generalized rank of a matrix [27], computing network centrality measures [3], matrix spectrum estimation [13, 19], and eigenvalue counting [7]. See [25] for a recent survey.

B Background on Communication Complexity

Our lower bound proofs use communication complexity. In a communication problem, Alice and Bob receive inputs $x \in \{-1,1\}^m$ and $y \in \{-1,1\}^m$, repsectively, and wish to compute a function $f : \{-1,1\}^m \times \{-1,1\}^m \to \{-1,1\}$. The players communicate according to a protocol P and end with an agreed-upon value z. The sequence of binary messages exchanged by the players is called the transcript of P, denoted $P(x,y)$. We say that the protocol computes f with error δ if $\Pr(z \neq f(x,y)) \leq \delta$. Let $\mathrm{CC}(P)$ be the length (in bits) of the transcript $P(x,y)$. The communication complexity of f is defined to be the minimum communication cost of any protocol with error δ:

$$\mathrm{CC}_\delta(f) = \min\{\mathrm{CC}(P) : P \text{ computes } f \text{ with error } \delta\}. \tag{10}$$

C Proof Details of Section 3

C.1 Proof of Theorem 3.1

To give a proof sketch, we consider a fixed group and a constant $\delta = \Theta(1)$. To bound the query complexity, we observe that within the group, each level of the binary tree incurs roughly the same number of matrix-vector product queries. Moreover, at the bottom level, there are s calls (including the one computing t_0) to `Hutch++`, with $\varepsilon' = \widetilde{O}(\varepsilon/\alpha)$ and $\delta' = O(\alpha)$. By Theorem 3.2, each call uses $\widetilde{O}(\alpha/\varepsilon)$ queries. Hence, each level uses $\widetilde{O}(s\alpha/\varepsilon) = \widetilde{O}(1/\varepsilon)$ queries. Since there are $\log(1/\alpha)$ levels per group and $O(m\alpha)$ groups, this gives a bound of $\widetilde{O}(m\alpha/\varepsilon)$ on the total number of queries, as claimed in Equation (7). A similar argument shows that the scheme achieves the desired error bound ε and failure probability δ. We formally prove Theorem 3.1:

Proof of Theorem 3.1. Fix a group g and an index i. We first argue that the output t_{gs+i+1} is an accurate estimate of the trace $\mathrm{Tr}(\boldsymbol{A}_i^{(g)})$, namely, one which satisfies Equation (6). By construction, the SumTree algorithm decomposes the $t_{gs+i+1} - t_0$ into at most $\log_2 s$ terms, one at each level ℓ. Each term is an estimate $t_{k,\ell}$ of $\mathrm{Tr}\left(\boldsymbol{A}_{2^\ell k}^{(g)} - \boldsymbol{A}_{2^\ell(k-1)}^{(g)}\right)$, for some ℓ, k. By assumption, each increment $\boldsymbol{A}_i^{(g)} - \boldsymbol{A}_{i-1}^{(g)}$ has Schatten-1 norm at most α. Hence, by the triangle inequality,

$$\left\|\boldsymbol{A}_{2^\ell k}^{(g)} - \boldsymbol{A}_{2^\ell(k-1)}^{(g)}\right\|_\star = \left\|\sum_{j=2^\ell k+1}^{2^\ell(k-1)} \boldsymbol{A}_j^{(g)} - \boldsymbol{A}_{j-1}^{(g)}\right\|_\star \leq 2^\ell \alpha$$

By the guarantee of the Hutch++ estimator (Theorem 3.2) and the inequality above, we have that for all ℓ, k

$$\begin{aligned}\left|t_{k,\ell} - \mathrm{Tr}\left(\boldsymbol{A}_{2^\ell k}^{(g)} - \boldsymbol{A}_{2^\ell(k-1)}^{(g)}\right)\right| &\leq \varepsilon'(\ell)\left\|\boldsymbol{A}_{2^\ell k}^{(g)} - \boldsymbol{A}_{2^\ell(k-1)}^{(g)}\right\|_\star \\ &\leq \varepsilon'(\ell)\cdot 2^\ell\alpha \\ &= \varepsilon/(2\log_2 s),\end{aligned} \tag{11}$$

with probability at least $1-\delta'$. Again, by the guarantee of Hutch++, t_0 approximates $\mathrm{Tr}\,\boldsymbol{A}_0^{(g)}$ up to an $(\varepsilon/2)\|\boldsymbol{A}_0^{(g)}\|_\star$ additive error. Therefore, conditioned on Equation (11), the total error of the

estimate t_{gs+i+1} for all g, i is bounded by

$$\begin{aligned}\left|t_{gs+i+1} - \operatorname{Tr}\left(\boldsymbol{A}_i^{(g)}\right)\right| &\le (\varepsilon/2)\left\|\boldsymbol{A}_0^{(g)}\right\|_\star + (\log_2 s)\cdot\varepsilon/(2\log_2 s)\\ &\le (\varepsilon/2)\left\|\boldsymbol{A}_0^{(g)}\right\|_\star + \varepsilon/2\\ &\le \varepsilon \end{aligned} \tag{12}$$

where the first line follows since there are $\log_2 s$ levels and the last line since $\|\boldsymbol{A}_0^{(g)}\|_\star \le 1$ by assumption. To bound the failure rate, we note that for a fixed g and i, Equation (12) holds if Equation (11) holds for all $t_{k,\ell}$ that are accessed in computing t_{gs+i+1} (via the SUMTREE procedure). By construction, there are at most $\log_2 s$ of these terms, where the bound follows from the number of levels of the binary tree. A simple union bound yields the desired guarantee Equation (6).

It remains to prove the bound on the query complexity (Equation (7)). Each group is treated identically, so we consider any fixed group. Within each group (of size s) and at each level ℓ, we make $O(s/2^\ell)$ calls to $\texttt{Hutch++}(\boldsymbol{A}, \varepsilon'(\ell), \delta')$. By Theorem 3.2, this leads to

$$O\left(\left(s/2^\ell\right)\cdot\left(\sqrt{\log(1/\delta')}/\varepsilon'(\ell) + \log(1/\delta')\right)\right)$$

queries at level ℓ. Plugging in the values of $\varepsilon'(\ell), \delta'$, this equals

$$O\left(\log(1/\alpha)\sqrt{\log(1/(\alpha\delta))}/\varepsilon + \frac{1}{2^\ell\alpha}\log(1/(\alpha\delta))\right).$$

Summing over $\ell \in \{0, 1, \cdots, \log_2 s - 1\}$, where $s = O(1/\alpha)$, we have that within each group, the number of queries is bounded by

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

There is a total of $\max\{1, O(m\alpha)\}$ groups. Also, we consider the case when $\alpha < \varepsilon$, where the algorithm only needs to provide a fresh estimate every ε/α time steps. Hence, the sequence length is reduced effectively to $m\alpha/\varepsilon$. Therefore, the query complexity of Algorithm 1 is at most

$$O\left((m\alpha+1)\log^2(1/\alpha)\sqrt{\log(1/(\alpha\delta))}/\varepsilon + m\min\{1, \alpha/\varepsilon\}\log(1/(\alpha\delta))\right).$$

This finishes the proof. □

C.2 General Schatten-*p* Norm Analysis

To generalize trace estimation to $\varepsilon\|\boldsymbol{A}\|_p$ error for any $p \in [1, 2]$, we need to revisit the variance reduction technique to achieve $O(1/\varepsilon)$ query complexity for the nuclear norm. The technique rewrites $\boldsymbol{A} = \boldsymbol{B}_k + \Delta_k$, where $\boldsymbol{B}_k$ is a rank-k matrix with a determined trace, and $\|\Delta_k\|_F \le O(1)\|\boldsymbol{A} - \boldsymbol{A}_k\|_F$, where $\boldsymbol{A}_k$ is the best rank-k approximation to $\boldsymbol{A}$ for some k. Then, we can approximate $\operatorname{tr}(\boldsymbol{A})$ by first explicitly calculating $\operatorname{tr}(\boldsymbol{B}_k)$ and then approximating the trace of Δ_k. The two step procedure requires a careful balancing for how queries are spent between the two components to minimize the total estimation error in the Schatten p norm and results in a $O(1/\varepsilon^p)$ query complexity.

Theorem C.1 (general Schatten-p error analysis of Hutch++)**.** *The Hutch++ estimator of rank k generalizes to a matrix $\boldsymbol{A}$ with any Schatten norm p bound for $p \in [1, 2]$, and satisfies $|\operatorname{tr}(\boldsymbol{A}) - \texttt{Hutch++}(\boldsymbol{A})| \le \varepsilon\|\boldsymbol{A}\|_p$ with a total matrix-vector query complexity of*

$$O\left(\left(\frac{\sqrt{\log(1/\delta)}}{\varepsilon}\right)^p + \log(1/\delta)\right)$$

Proof of Theorem C.1. Let $\boldsymbol{A}_k$ be the best rank-k approximation to $\boldsymbol{A}$. Then the Hutch++ estimator allows us to estimate the trace of $\boldsymbol{A}$ by writing $\boldsymbol{A} = \boldsymbol{A}_k + \Delta$, where $\|\Delta\|_F \le 2\|\boldsymbol{A} - \boldsymbol{A}_k\|_F$.

Then, we can directly calculate $\mathrm{Tr}(\boldsymbol{A}_k)$ and use Hutchinson's method [14] with ℓ matrix-vector multiplication queries, which gives a standard additive error guarantee of

$$C\sqrt{\frac{\log(1/\delta)}{\ell}}\|\Delta\|_F,$$

for some fixed constant C. Now, we use the fact that if σ_i are the singular values of $\boldsymbol{A}$, then by the definition of Schatten norms,

$$\|\Delta\|_F \le 2\|\boldsymbol{A}-\boldsymbol{A}_k\|_F \le \sqrt{\sum_{i=k}^{n}\sigma_i^2} \le \sqrt{\sigma_k^{2-p}\sum_{i=k}^{n}\sigma_i^p}.$$

Note that we have the following inequality: $k\sigma_k^p \le \|\boldsymbol{A}\|_p^p$. Therefore, rearranging gives $\sigma_k^{2-p} \le k^{1-2/p}\|\boldsymbol{A}\|_p^{2-p}$. Finally, we conclude that the total error is bounded by

$$\begin{aligned}
\|\mathrm{tr}(\boldsymbol{A}) - \texttt{Hutch++}(\boldsymbol{A})\| &\le C\sqrt{\frac{\log(1/\delta)}{\ell}}\|\Delta\|_F \\
&\le C\sqrt{\frac{\log(1/\delta)}{\ell}}\sqrt{\sigma_k^{2-p}\sum_{i=k}^{n}\sigma_i^p} \\
&\le C\sqrt{\frac{\log(1/\delta)}{\ell}}\sqrt{k^{1-2/p}\|\boldsymbol{A}\|_p^{2-p}\|\boldsymbol{A}\|_p^p} \\
&\le C\sqrt{\frac{\log(1/\delta)}{\ell k^{2/p-1}}}\|\boldsymbol{A}\|_p.
\end{aligned}$$

Since we want to set $k=\ell$ to minimize the query complexity, it follows that to reduce to error to ε, we need a number of queries equal to:

$$k=\ell=\left(\frac{\sqrt{\log(1/\delta)}}{\varepsilon}\right)^p.$$

Finally, by the same analysis of [18], using $O(k+\log(1/\delta))$ matrix-vector products suffices to obtain a constant-factor rank-k approximation of $\boldsymbol{A}$, and further, we want $l \ge \log(1/\delta)$. This concludes the proof. □

With this generalized analysis of Hutch++, one can easily extend Theorem 3.1 and obtain:

Theorem C.2 (general Schatten-p norm analysis of Algorithm 1)**.** *Let $\boldsymbol{A}_1, \boldsymbol{A}_2, \cdots, \boldsymbol{A}_m$ be $n\times n$ be matrices such that (1) $\|\boldsymbol{A}_i\|_p \le 1$ for all i, and (2) $\|\boldsymbol{A}_{i+1}-\boldsymbol{A}_i\|_p \le \alpha$ for all $i \le m-1$ and some $p\in[1,2]$. Given matrix-vector multiplication access to the matrices, a failure rate $\delta>0$, and an error bound $\varepsilon>0$, there is an algorithm that outputs a sequence of estimates $t_1,\cdots,t_m$ such that for each $i\in[m]$,*

$$|t_i - \mathrm{Tr}\,\boldsymbol{A}_i| \le \varepsilon, \text{ with probability at least } 1-\delta. \tag{13}$$

The algorithm uses a total of

$$O\left((m\alpha+1)\log^{1+p}(1/\alpha)\left(\frac{\sqrt{\log(1/(\alpha\delta))}}{\varepsilon}\right)^p + m\log(1/(\alpha\delta))\right). \tag{14}$$

matrix-vector multiplication queries to $\boldsymbol{A}_1, \boldsymbol{A}_2, \cdots, \boldsymbol{A}_m$.

The proof is via counting the number of queries differently using the general Hutch++ analysis Theorem C.1.

Proof of Theorem C.2. Our error analysis is almost identical to the nuclear norm case. We sketch it here for completeness. Consider any fixed group g. By assumption every increment $\boldsymbol{A}_i^{(g)} - \boldsymbol{A}_{i-1}^{(g)}$

has Schatten-p norm at most α. Hence, by the triangle inequality,

$$\left\|\boldsymbol{A}^{(g)}_{2^\ell k} - \boldsymbol{A}^{(g)}_{2^\ell(k-1)}\right\|_p = \left\|\sum_{j=2^\ell k+1}^{2^\ell(k-1)} \boldsymbol{A}^{(g)}_j - \boldsymbol{A}^{(g)}_{j-1}\right\|_p \le 2^\ell \alpha$$

By the general Schatten-p norm analysis of the Hutch++ estimator (Theorem C.1) and the inequality above, we get that for all ℓ, k:

$$\begin{aligned}\left|t_{k,\ell} - \operatorname{Tr}\left(\boldsymbol{A}^{(g)}_{2^\ell k} - \boldsymbol{A}^{(g)}_{2^\ell(k-1)}\right)\right| &\le \varepsilon'(\ell)\left\|\boldsymbol{A}^{(g)}_{2^\ell k} - \boldsymbol{A}^{(g)}_{2^\ell(k-1)}\right\|_p \\ &\le \varepsilon'(\ell)\cdot 2^\ell\alpha \\ &= \varepsilon/(2\log_2 s), \end{aligned} \tag{15}$$

with probability at least $1-\delta'$. Again, by the guarantees of Hutch++, t_0 approximates $\operatorname{Tr}\boldsymbol{A}^{(g)}_0$ up to an $(\varepsilon/2)\|\boldsymbol{A}^{(g)}_0\|_p$ additive error. Therefore, conditioned on Equation (15), the total error of the estimate t_{gs+i+1} for all g, i is bounded by

$$\begin{aligned}\left|t_{gs+i+1} - \operatorname{Tr}\left(\boldsymbol{A}^{(g)}_i\right)\right| &\le (\varepsilon/2)\left\|\boldsymbol{A}^{(g)}_0\right\|_p + (\log_2 s)\cdot\varepsilon/(2\log_2 s) \\ &\le (\varepsilon/2)\left\|\boldsymbol{A}^{(g)}_0\right\|_p + \varepsilon/2 \\ &\le \varepsilon. \end{aligned} \tag{16}$$

A union bound thus proves the accuracy guarantee (Equation 13).

We now count the query complexity differently using Theorem C.1. As before, within each group (of size s) and in each level ℓ, we make $O(s/2^\ell)$ calls to $\texttt{Hutch++}(\boldsymbol{A}, \varepsilon'(\ell), \delta')$. This leads to a total number of

$$O\left(\left(s/2^\ell\right)\cdot\left(\left(\frac{\sqrt{\log(1/\delta')}}{\varepsilon'(\ell)}\right)^p + \log(1/\delta')\right)\right) \tag{17}$$

matrix-vector multiplication queries by Theorem C.1. Substituting $\varepsilon'(\ell) = \varepsilon/2^{\ell+1}\alpha\log_2 s$ and $\delta' = \alpha\delta$, we have

$$O\left(\log^p(1/\alpha)\left(\frac{\sqrt{\log(1/(\alpha\delta))}}{\varepsilon}\right)^p + + \frac{1}{2^\ell\alpha}\log(1/(\alpha\delta))\right).$$

Note that we have used the fact that $\alpha 2^{l+1} \le \alpha s \le 1$ to simplify the expression. Summing over $\ell \in \{0, 1, \cdots, \log_2 s - 1\}$, where $s = O(1/\alpha)$, we obtain that within each group, the number of queries is bounded by

$$O\left(\log^{1+p}(1/\alpha)\left(\frac{\sqrt{\log(1/(\alpha\delta))}}{\varepsilon}\right)^p + \frac{1}{\alpha}\log(1/(\alpha\delta))\right).$$

Since there are $\max\{1, O(m\alpha)\}$ groups, the total query complexity is at most

$$O\left((m\alpha+1)\log^{1+p}(1/\alpha)\left(\frac{\sqrt{\log(1/(\alpha\delta))}}{\varepsilon}\right)^p + m\log(1/(\alpha\delta))\right).$$

This completes the proof. □

C.3 Relaxing Assumptions

Recall that for dynamic trace estimation, we generally require all matrices $\boldsymbol{A}_i$ to have unit-bounded Schatten-p norm. While it is often the case that the initial matrix $\boldsymbol{A}_1$ has controlled norm, in practice it is unrealistic to assume a general bound on the matrix norm upon dynamic updates. Of course, note that due to the bounded difference assumption, we can always use a linear bound $\|\boldsymbol{A}_i\|_p \le 1 + \alpha i$. However, using this bound naïvely with the analysis of other algorithms, such as Hutchinson's or its variance-reduced version of [6], introduces additional $\operatorname{poly}(m, \alpha)$ terms in the query complexity. Instead, we show that our tree-based procedure without any initial partitioning still attains an optimal dependence on m and α for the nuclear norm.

Theorem C.3 (general Schatten-p norm analysis of non-partitioned Algorithm 1)**.** *Let $\boldsymbol{A}_1, \boldsymbol{A}_2, \cdots, \boldsymbol{A}_m$ be $n \times n$ matrices such that (1) $\|\boldsymbol{A}_1\|_* \leq 1$ and (2) $\|\boldsymbol{A}_{i+1} - \boldsymbol{A}_i\|_* \leq \alpha$ for all $i \leq m-1$. Given matrix-vector multiplication access to the matrices, a failure rate $\delta > 0$ and error bound ε, there is an algorithm that outputs a sequence of estimates $t_1, \cdots, t_m$ such that for each $i \in [m]$,*

$$|t_i - \operatorname{Tr} \boldsymbol{A}_i| \leq \varepsilon, \text{ with probability at least } 1 - \delta. \tag{18}$$

The algorithm uses a total of

$$O\left(m\alpha \log(m)^2 \left(\frac{\sqrt{\log(m\delta)}}{\varepsilon}\right) + m\log(m\delta)\right). \tag{19}$$

matrix-vector multiplication queries to $\boldsymbol{A}_1, \boldsymbol{A}_2, \cdots, \boldsymbol{A}_m$.

The proof follows by grouping all queries into a group of size m, implying that there is a $\log(m)$ overhead by using the tree technique. Therefore, the main alteration to Algorithm 1 is to 1) avoid partitioning into $1/\alpha$ subgroups and 2) calling `Hutch++` at each level with updated parameters: $\varepsilon'(\ell) = \varepsilon/(2^{\ell+1}\alpha \log_2 m)$ and $\delta' = \delta/m$.

Proof of Theorem C.3. Compared with Theorem 3.1, the error and success rate analysis remains unchanged. We only need to count the query complexity differently using Theorem C.1. Note that in this case, there is only one group of size $s = m$. As before, at each level ℓ, we make $O(s/2^\ell)$ calls to `Hutch++`$(\boldsymbol{A}, \varepsilon'(\ell), \delta')$. This leads to a total number of

$$O\left(\left(s/2^\ell\right) \cdot \left(\left(\frac{\sqrt{\log(1/\delta')}}{\varepsilon'(\ell)}\right)^p + \log(1/\delta')\right)\right) \tag{20}$$

matrix-vector multiplication queries by Theorem C.1. Substituting $\varepsilon'(\ell) = \varepsilon/2^{\ell+1}\alpha \log_2 m$ and $\delta' = \delta/m$, we have

$$O\left(m\log(m)\alpha\left(\frac{\sqrt{\log(m\delta)}}{\varepsilon}\right) + \frac{m}{2^\ell}\log(m\delta)\right).$$

Summing over $\ell \in \{0, 1, \cdots, \log_2 m\}$, we obtain that for this large group, the number of queries is bounded by

$$O\left(m\log^2(m)\left(\frac{\sqrt{\log(m\delta)}}{\varepsilon}\right) + m\log(m\delta)\right).$$

This completes the proof. □

D Proof Details of Section 4

D.1 Proof of Theorem 4.1

We prove the two lower bounds separately. Together they imply Theorem 4.1.

D.1.1 Lower Bound I

Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ be a general square matrix. Recall that the goal is to estimate its trace $\operatorname{Tr} \boldsymbol{A}$ up to an additive $\varepsilon\|\boldsymbol{A}\|_p$. We work under the bit complexity model, where the query vectors $q_1, q_2, \cdots q_r \in \mathbb{R}^n$ have entries specified by k bits. To lower bound r, the number of queries, we reduce the communication problem of the Approximate-Orthogonality to trace estimation.

The Approximate-Orthogonality problem is a two-party communication problem defined on inputs in $\{-1, 1\}^m \times \{-1, 1\}^m$ by the Boolean function

$$\mathrm{ORT}_{b,m}(x, y) = \begin{cases} 1, & \text{if } |\langle x, y\rangle| \leq b\sqrt{m} \\ -1, & \text{otherwise.} \end{cases} \tag{21}$$

The problem is known to have $\Omega(m)$ communication complexity, under the uniform distribution. Let

$$\mathrm{tail}(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-x^2/2} dx. \tag{22}$$

be the tail probability of the standard normal.

Lemma D.1 (Communication complexity of ORT, Theorem 4.2 of [5])**.** *Let $b > 1/5$ be a constant and $\theta = \mathrm{tail}(2.01 \max\{66, b\})$. Then we have $CC_\theta(\mathrm{ORT}_{b,m}) = \Omega(m)$. The lower bound holds even when the inputs are drawn uniformly from $\{-1,1\}^m \times \{-1,1\}^m$.*

We now prove our adaptive trace estimation lower bound for general matrices, by connecting it with the APPROXIMATE-ORTHOGONALITY problem. It implies that the classic Hutchinson's estimator is optimal for constant success probability.

Theorem D.2 (Adaptive query lower bound, I)**.** *Any algorithm that accesses a square matrix $\boldsymbol{A}$ via matrix-vector multiplication queries requires at least $\Omega\left(\frac{1}{\varepsilon^p(k+\log(1/\varepsilon))}\right)$ queries to output an estimate t such that with probability at least $1 - \delta/2$, $|t - \mathrm{Tr}\,\boldsymbol{A}| \le \varepsilon \|\boldsymbol{A}\|_p$, for $p \in [1,2]$ and $\delta = \mathrm{tail}(2.01 \cdot 66) = \Theta(1)$, where the query vectors may be adaptively chosen and their entries are specified by k bits.*

Proof of Theorem D.2. Let $\mathcal{A}$ be a possibly adaptive algorithm for trace estimation using matrix-vector multiplication queries. Suppose it takes at most $r(n)$ queries to solve the problem, on any n-by-n square matrix, with success rate at least $1-\delta = \Omega(1)$. Consider an instance of APPROXIMATE-ORTHOGONALITY with $b = 2$, where (x, y) is drawn uniformly from $\{-1,1\}^m \times \{-1,1\}^m$.

The proof proceeds by reducing the problem of computing $\mathrm{ORT}_{b,m}(x, y)$ to trace estimation via $\mathcal{A}$. Let $n = \frac{\delta^{p/2}}{2^{p/2}\varepsilon^p} = \Theta(1/\varepsilon^p)$ and $m = n^2$. The reduction and its resulting communication protocol are given as follows. First, given $x \in \{-1,1\}^m$, Alice creates a square matrix $\boldsymbol{A}$, where the rows of $\boldsymbol{A}$ correspond to the entries of x in order. Similarly, given $y \in \{-1,1\}^m$, Bob creates a square matrix $\boldsymbol{B}$, where the columns of $\boldsymbol{B}$ correspond to the entries of y in order. Then the protocol repeats the following steps for $r(n)$ rounds.

(i) In the i-th round from $i = 1$, Alice creates the first query q_i, according to $\mathcal{A}$, given all previous query values $\left\{q_j^\top \boldsymbol{AB}\right\}_{j<i}$. She computes $q_i^\top \boldsymbol{A}$ and sends it to Bob.

(ii) Bob computes $q_i^\top \boldsymbol{AB}$ and sends it back to Alice.

At the end of the protocol, with probability at least $1 - \delta/2$, Alice and Bob obtain an estimate t such that

$$|t - \mathrm{Tr}(\boldsymbol{AB})| \le \varepsilon \|\boldsymbol{AB}\|_p, \tag{23}$$

by the guarantee of algorithm $\mathcal{A}$. Finally, they output $z = 1$ if $t \le 3\sqrt{m}$ and $z = -1$ otherwise.

We argue that the above protocol computes $\mathrm{ORT}_{b,m}$ with error at most $\delta = \mathrm{tail}(2.01 \cdot 66)$. First, note that by construction of steps (i) and (ii), we have $\mathrm{Tr}(\boldsymbol{AB}) = \langle x, y \rangle$. Therefore, by Equation (23),

$$\Pr_{\mathcal{A}}(|t - \langle x, y\rangle| \le \varepsilon \|\boldsymbol{AB}\|_p) = \Pr_{\mathcal{A}}(|t - \mathrm{Tr}(\boldsymbol{AB})| \le \varepsilon \|\boldsymbol{AB}\|_p) \ge 1 - \delta/2. \tag{24}$$

It now suffices to show that the error term $\varepsilon \|\boldsymbol{AB}\|_p$ is small. Note that since x, y are drawn uniformly at random, it follows that $\mathbb{E}(\boldsymbol{AB})_{i,j}^2 = n$ for all $i, j \in [n]$. By linearity of expectation, $\mathbb{E}\|\boldsymbol{AB}\|_F^2 = n^3$. By Markov's inequality, $\Pr(\|\boldsymbol{AB}\|_F^2 > tn^3) \le 1/t$ for any $t > 0$, and therefore,

$$\Pr(\varepsilon \|\boldsymbol{AB}\|_F > \varepsilon \sqrt{t} n^{3/2}) \le 1/t.$$

Since $\|\boldsymbol{X}\|^p \le n^{1/p-1/q} \|\boldsymbol{X}\|_q$ for any $n \times n$ matrix $\boldsymbol{X}$, it follows that

$$\Pr(\varepsilon \|\boldsymbol{AB}\|_p > \varepsilon \sqrt{t} n^{1/p-1/2} \cdot n^{3/2}) = \Pr(\varepsilon \|\boldsymbol{AB}\|_p > \varepsilon \sqrt{t} n^{1/p+1}) \le 1/t.$$

Plugging in the value of $\varepsilon = \frac{1}{n^{1/p}} \sqrt{\frac{\delta}{2}}$ and setting $t = 2/\delta$, we get

$$\Pr_{x,y} (\varepsilon \|\boldsymbol{AB}\|_p > n) \le 1/t = \delta/2 \tag{25}$$

Combining Equation (24) and Equation (25) and using a union bound,

$$\Pr(|t - \langle x, y\rangle| \le \sqrt{m}) \ge 1 - \delta. \tag{26}$$

Therefore, whenever $\langle x, y\rangle \le 2\sqrt{m}$, we have $t \le 3\sqrt{m}$, and so the protocol outputs $z = 1$ correctly. This proves that the protocol solves $\mathrm{ORT}_{b,m}$ with error at most δ (for $b = 2$).

To complete the proof, we account for the total communication cost of the protocol. For that, we simply note that each message from Alice or Bob is a vector of n dimensions. It suffices to specify each entry with $k + \log(n/\varepsilon)$ bits. Hence, the protocol solves $\mathrm{ORT}_{b,m}$ with communication cost $r(n) \cdot O(n(k + \log(1/\varepsilon)))$. By the communication lower bound Theorem D.1, it is required that

$$r(n) \cdot O(n(k + \log(1/\varepsilon))) \ge m = n^2.$$

Rearranging and using $n = \Theta(1/\varepsilon^p)$, we have $r(n) \ge \Omega\left(\frac{1}{\varepsilon^p(k+\log(1/\varepsilon))}\right)$, as desired. □

D.1.2 Lower Bound II

We now give a second lower bound that yields the correct dependence on the failure probability δ. The bound holds for any Schatten-p norm error guarantee, so we state it generally. In particular, we show:

Theorem D.3 (Adaptive query lower bound, II)**.** *Any algorithm that accesses a square matrix $\boldsymbol{A}$ via matrix-vector multiplication queries requires at least $\Omega\left(\frac{\log(1/\delta)}{k+\log\log(1/\delta)}\right)$ queries to output an estimate t such that with probability at least $1 - \delta$, $|t - \operatorname{Tr}\boldsymbol{A}| \le 0.1\|\boldsymbol{A}\|_p$, for any p and any $\delta \in (0, 1)$, where the query vectors may be adaptively chosen and their entries are specified by k bits.*

Our proof leverages another communication problem, GAP-EQUALITY. In this problem, Alice holds $x \in \{0, 1\}^n$ and Bob holds $y \in \{0, 1\}^n$, under the promise that either $x = y$ or $\|x - y\|_2^2 = n/2$. They wish to compute

$$\mathrm{EQ}_n(x, y) = \begin{cases} 1, & \text{if } x = y \\ -1, & \text{otherwise.} \end{cases} \tag{27}$$

The problem requires linear communication complexity for any deterministic protocol [4].

Lemma D.4 (Communication complexity of GAP-EQUALITY [4])**.** *Any deterministic protocol for computing EQ_n requires $\Omega(n)$ bits of communication.*

We are now ready to prove Theorem D.3.

Proof of Theorem D.3. We give a reduction from solving GAP-EQUALITY as a two-party communication problem to trace estimation via adaptive matrix-vector multiplication queries. Let $n = \log(1/\delta)$ and $x, y \in \{0, 1\}^n$ be an instance of GAP-EQUALITY. Let $\boldsymbol{A} = (x - y)(x - y)^\top$, which has rank 1. Under the promise, either (i) $\boldsymbol{A} = \boldsymbol{0}$, the all 0 matrix, or (ii) has Schatten-p norm $n/2$ for any p. In case (ii), we have $\operatorname{Tr}\boldsymbol{A} = n/2$. Thus, one can compute $\mathrm{EQ}_n(x, y)$, by estimating $\operatorname{Tr}\boldsymbol{A}$ up to an additive error of $0.1\|\boldsymbol{A}\|_p$, for any p.

We now argue any trace estimation algorithm $\mathcal{A}$ with failure rate δ and error ε yields a deterministic protocol for solving EQ_n. First, by a union bound over all possible x, y under the promise, we have that for all $\boldsymbol{A} = (x - y)(x - y)^\top$, the output t of $\mathcal{A}$ given $\boldsymbol{A}$ always satisfies

$$|t - \operatorname{Tr}\boldsymbol{A}| \le \varepsilon\|\boldsymbol{A}\|_p. \tag{28}$$

Suppose $\mathcal{A}$ uses $r = o(\log(1/\delta))$ adaptive queries $q_1, q_2, \cdots, q_r$. In case (i) when $\boldsymbol{A} = \boldsymbol{0}$, all query answers it receives are the zero vector. The algorithm must always output 0, to satisfy the trace estimation guarantee (Equation (28)). Thus, in order to always be correct in case (ii), it must be that one of its query answers is not 0. But as soon as its first query answer is not 0, it knows that it is in case (ii). It follows that algorithm $\mathcal{A}$ just keeps receiving the all-0 vector until it either decides to stop querying or receives a non-zero output vector and immediately decides to stop querying. Thus, for these inputs, we can assume the query algorithm is in fact non-adaptive, since we can consider what its query sequence would be in advance if it were to repeatedly receive the 0 vector as an answer. Hence, we can think of $\boldsymbol{Q} = (q_1, q_2, \cdots, q_r)$ as an $r \times n$ matrix with entries specified with

k bits, and we have the property that $\boldsymbol{Q}(x-y)=0$ if and only if $x=y$. This gives a protocol for GAP-EQUALITY: Alice simply sends $\boldsymbol{Q}x$ to Bob, who checks if $\boldsymbol{Q}x=\boldsymbol{Q}y$. The communication is $r(k+\log n)=r(k+\log\log(1/\delta))$, which must be $\Omega(\log(1/\delta))$ by Theorem D.4, and so we get an $r=\Omega(\log(1/\delta)/(k+\log\log(1/\delta)))$ adaptive lower bound.

□

D.2 Proof of Theorem 4.2

We start with a standard definition.

Definition D.1 (Gaussian and Wigner Random Matrices)**.** *We let $\boldsymbol{G}\sim\mathcal{N}(n)$ denote an $n\times n$ random Gaussian matrix with i.i.d. $\mathcal{N}(0,1)$ entries. We let $\boldsymbol{W}\sim\mathcal{W}(n)=(\boldsymbol{G}+\boldsymbol{G}^T)/2$ denote an $n\times n$ Wigner matrix, where $\boldsymbol{G}\sim\mathcal{N}(n)$.*

Fact D.5 (Upper and Lower Gaussian Tail Bounds)**.** *Letting $Z\sim\mathcal{N}(0,1)$ be a univariate Gaussian random variable, for any $t>0$, $\Pr[|Z|\geq t]=\Theta(t^{-1}\exp(-\frac{t^2}{2}))$.*

Suppose that we draw a matrix $\boldsymbol{G}\in\mathbb{R}^{n\times n}$ from the Gaussian or related Wigner distribution and try to learn the entries of the matrix via matrix-vector queries. Because the Gaussian is rotationally and subspace invariant, after a few queries, the conditional distribution of the remaining matrix is also Gaussian (or Wigner)-distributed, no matter how the queries are chosen. This property allows us to exactly characterize the remaining uncertainty of the trace estimation procedure, especially with respect to the failure probability δ, even after seeing a few query results.

Lemma D.6. *(Conditional Distribution [Lemma 3.4 of [24]]) Let $\boldsymbol{G}\sim\mathcal{N}(n)$ be as in Definition D.1 and suppose our matrix is $\boldsymbol{W}=(\boldsymbol{G}+\boldsymbol{G}^\top)/2$. Suppose we have any sequence of vector queries, $\boldsymbol{v}_1,\ldots,\boldsymbol{v}_T$, along with responses $\boldsymbol{w}_i=\boldsymbol{W}\boldsymbol{v}_i$. Then, conditioned on our observations, there exists a rotation matrix $\boldsymbol{V}$, independent of $\boldsymbol{w}_i$, such that*

$$\boldsymbol{V}\boldsymbol{W}\boldsymbol{V}^\top=\begin{bmatrix}Y_1 & Y_2^\top\\ Y_2 & \widetilde{\boldsymbol{W}}\end{bmatrix}$$

where Y_1,Y_2 are deterministic and $\widetilde{\boldsymbol{W}}=(\widetilde{\boldsymbol{G}}+\widetilde{\boldsymbol{G}}^\top)/2$, where $\widetilde{\boldsymbol{G}}\sim\mathcal{N}(n-T)$.

Proof of Theorem 4.2. By standard minimax arguments, it suffices to construct a hard distribution for any deterministic algorithm. Consider $\boldsymbol{W}\sim\mathcal{W}(n)$ for some n that we will determine later. From concentration of the singular values of large Gaussian matrices [23], with probability at least $1-\delta/10$, we have $\sigma_{\max}(\boldsymbol{G})\leq Cn^{1/2}$ for some absolute constant C when $n\geq\log(1/\delta)$. Therefore, we conclude that $\|\boldsymbol{G}\|_p\leq Cn^{1/2+1/p}$ for some absolute constant C. Therefore, by the triangle inequality, $\|\boldsymbol{W}\|_p$ can be bounded by the same value.

Let m be the number of matrix-vector queries, and assume that $m\leq n/2$. By Theorem D.6, we see that conditioned on the queries, our matrix $\boldsymbol{W}$ can be decomposed into a determined part and a Gaussian submatrix $\widetilde{\boldsymbol{W}}\sim\mathcal{W}(n-m)$. Therefore, our conditional distribution of the trace of $\boldsymbol{W}$ is, up to a deterministic shift, the same as the distribution of $\widetilde{\boldsymbol{W}}$, which is simply a Gaussian with variance at least $n-m\geq n/2$. We can check this since $\mathrm{tr}(\widetilde{\boldsymbol{W}})=\frac{1}{2}\mathrm{tr}(\widetilde{\boldsymbol{G}})+\frac{1}{2}\mathrm{tr}(\widetilde{\boldsymbol{G}}^\top)=\mathrm{tr}(\widetilde{\boldsymbol{G}})=\sum_i\widetilde{\boldsymbol{G}}_{ii}$, where $\boldsymbol{G}_{ii}\sim N(0,1)$ are independent for $1\leq i\leq n-m$.

Since our algorithm determines a Gaussian of variance at least $n-m\geq n/2$ up to an additive error of $\varepsilon\|\boldsymbol{A}\|_p$ with probability at least $1-\delta$, we conclude that if $\varepsilon\|\boldsymbol{A}\|_p\leq\sqrt{\log(1/\delta)n}$, then we have a contradiction from the anti-concentration of Gaussians (see Theorem D.5). Therefore, whenever $\varepsilon\|\boldsymbol{A}\|_p\leq\sqrt{\log(1/\delta)n}$ holds, we can deduce a lower bound on the number of matrix-vector queries: $m\geq n/2$.

Therefore, solving $\varepsilon\|\boldsymbol{A}\|_p\leq\sqrt{\log(1/\delta)n}$ for the largest possible value of n gives:

$$n=\Omega\left(\left(\frac{\sqrt{\log(1/\delta)}}{\varepsilon}\right)^p\right)$$

Note that this holds for any $\delta, \varepsilon > 0$ such that $n \geq \log(1/\delta)$. Therefore, we need to enforce that $\varepsilon < (\log(1/\delta))^{1/2-1/p}$. □

E Proof Details for Section 5

E.1 Proof of Theorem D.6

Proof of Theorem D.6. Let $\alpha = 1/(m-1)$. Given a square matrix $\boldsymbol{A}$ with $\|\boldsymbol{A}\|_p = 1$, construct a sequence of matrices

$$\boldsymbol{A}_1 = 0, \quad \boldsymbol{A}_2 = \alpha \cdot \boldsymbol{A}, \quad \ldots \quad \boldsymbol{A}_{1/\alpha} = (1-\alpha)\boldsymbol{A}, \quad \boldsymbol{A}_m = \boldsymbol{A}. \tag{29}$$

Suppose that we have a dynamic trace estimation algorithm $\mathcal{A}$ running on the sequence $(\boldsymbol{A}_i)$. By construction, each $\boldsymbol{A}_i$ is a scaling of $\boldsymbol{A}$. Suppose that in the end $\mathcal{A}$ outputs an estimate t_m such that $|t_m - \operatorname{Tr} \boldsymbol{A}| \leq \varepsilon \|\boldsymbol{A}\|_p$ with probability at least $1-\delta$, using matrix-vector multiplies with $\boldsymbol{A}$. This solves the static trace estimation problem with a Schatten-p norm error guarantee. By assumption, it must have used $\Omega(r)$ matrix-vector multiplication queries with respect to $\boldsymbol{A}$. Therefore, if $\mathcal{A}$ uses $o(r\alpha m)$ queries, it would immediately violate our assumption, which is a contradiction. □

E.2 Proof of Theorem 5.4

Proof. Let $x, y \in \{0,1\}^n$ be an instance of GAP-EQUALITY, where $n = \log(1/\delta)$. Recall that GAP-EQUALITY is a promise problem. Under its promise, either $x = y$ or $\|x-y\|_2^2 = n/2$, and the goal is to distinguish the two cases. For any given x, y, let $\boldsymbol{B}_{x,y} = \frac{2}{n}(x-y)(x-y)^\top$. Then since $\boldsymbol{B}_{x,y}$ is rank-1, $\|\boldsymbol{B}_{x,y}\|_p = 0$ if $x = y$ or $\|\boldsymbol{B}_{x,y}\|_p = 1$ otherwise.

To obtain the claimed lower bound, we consider two parameter regimes. First, if $\alpha > \varepsilon$, we construct the following hard instance, which is a sequence of m matrices satisfying the Schatten p norm assumption for dynamic trace estimation. Let $\boldsymbol{A}_0 \in \mathbb{R}^{N \times N}$ be an all 0s matrix, with $N = \min\{m, 1/\alpha\} \log(1/\delta)$. Throughout the updates, $\boldsymbol{A}_i$ will remain a block diagonal matrix, which consists of m block matrices along the diagonal and each of dimension $\log(1/\delta) \times \log(1/\delta)$. In particular, for all steps $i = \{1, 2, \cdots, \min\{m, 1/\alpha\} - 1\}$, we set

$$\boldsymbol{A}_i = \begin{bmatrix} \boldsymbol{B}_1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & \mathbf{B}_2 & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & 0 & \ddots & \cdots & \cdots & \cdots & \vdots \\ \vdots & \vdots & 0 & \boldsymbol{B}_i & 0 & \cdots & \vdots \\ \vdots & \vdots & \vdots & 0 & 0 & \cdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \end{bmatrix} \tag{30}$$

where $\boldsymbol{B}_i = \alpha \boldsymbol{B}_{x_i, y_i}$ with $x_i, y_i \in \{0,1\}^n$ an independent instance of GAP-EQUALITY. In other words, at each step i, we update $\boldsymbol{A}_{i-1}$ by replacing the i-th diagonal block (currently being all 0s) with $\boldsymbol{B}_i$. Each update changes the trace by 0 or α, by the construction of $\boldsymbol{B}_{x,y}$. If $m \leq 1/\alpha$, this completes the construction, and note that the matrices $\{\boldsymbol{A}_i\}$ all have norm bounded by 1. If $m > 1/\alpha$, we continue the construction by deleting one distinct diagonal block at each step until the matrix is the zero matrix. Then we repeat the same rounds of insertion (according to Equation (30)) and deletion until reaching time step m. Observe again that the construction satisfies the Schatten norm assumption for dynamic trace estimation.

We now argue the query complexity as follows:

- In the case of $m \leq 1/\alpha$, each update is either (i) trivially 0 or (ii) increases the trace by $\alpha > \varepsilon$. Hence, any dynamic algorithm for outputting $|t_i - \operatorname{Tr} \boldsymbol{A}_i| \leq \varepsilon < \alpha$, with probability at least $1-\delta$, would distinguish between case (i) and (ii) with probability at least $1-\delta$. However, this requires $\Omega\left(\frac{\log(1/\delta)}{k + \log\log(1/\delta)}\right)$ matrix-vector multiplication queries by Theorem D.3.

- In the case of $m \le 1/\alpha$, note that (almost) half of the update steps are insertions. By the same argument, any dynamic algorithm that gives a good estimate in an insertion step i can solve the hard instance of estimating $\operatorname{Tr} \boldsymbol{B}_i$. Hence, we get the same query complexity lower bound.

To summarize, if $\alpha > \varepsilon$, we get a lower bound of $\Omega\left(m \cdot \frac{\log(1/\delta)}{k+\log\log(1/\delta)}\right)$ queries.

Now we move on to the case of $\alpha \le \varepsilon$. We use the same construction as described by Equation 30, where each $\boldsymbol{A}_i$ consists of multiple updates over $s = \lceil \varepsilon/\alpha \rceil$ steps by setting $\boldsymbol{B}_i = \sum_{j=1}^{s} (1/s) \cdot \boldsymbol{B}_{x_i, y_i}$ with $\boldsymbol{B}_{x_i,y_i}$ an independent instance of GAP-EQUALITY. We repeat the argument earlier and apply the hardness of Theorem D.3 on the sequence of $\boldsymbol{A}_i$. This blows up the sequence length by a factor of s, and hence leads to a lower bound of $\Omega\left(m \cdot \frac{\alpha}{\varepsilon} \frac{\log(1/\delta)}{k+\log\log(1/\delta)}\right)$. □

F Lower Bound for Non-Adaptive Trace Estimation

In the case of non-adaptive queries, we give a stronger lower bound than Theorem 4.1 in the bit complexity model. The bound matches Hutchinson's guarantee for general square matrices up to a bit complexity term.

Theorem F.1 (Non-adaptive query lower bound)**.** *Any algorithm that accesses a square matrix $\boldsymbol{A}$ via non-adaptive matrix-vector multiplication queries requires at least $\Omega\left(\frac{\log^{p/2}(1/\delta)}{\varepsilon^p(k+\log(1/\varepsilon))}\right)$ queries to output an estimate t such that with probability at least $1-\delta$, $|t - \operatorname{Tr} \boldsymbol{A}| \le \varepsilon \|\boldsymbol{A}\|_p$, for any p and $\varepsilon, \delta \in (0,1)$, where each entry of the query vectors is specified by k bits.*

The proof is via a reduction from the Augmented Indexing communication problem with low error [15]. For a sufficiently large universe $\mathcal{U}$ and an element $\perp \notin \mathcal{U}$, the problem $\mathrm{IND}_{n,\mathcal{U}}$ is defined as follows.

- Alice gets $x = (x_1, x_2, \ldots, x_n) \in \mathcal{U}^n$.
- Bob gets $y = (y_1, y_2, \ldots, y_n) \in (\mathcal{U} \cup \{\perp\})^n$ such that for some unique i
  - (i) $y_i \in \mathcal{U}$,
  - (ii) $y_k = x_k$ for all $k < i$,
  - (iii) $y_{i+1} = y_{i+2} = \cdots = y_N = \perp$.

Finally, Bob wishes to output whether $x_i = y_i$. The one-way communication complexity of $\mathrm{IND}_{n,\mathcal{U}}$ is known:

Lemma F.2 (Communication complexity of Augmented Indexing [15])**.** *Any one-way communication protocol for computing $\mathrm{IND}_{n,\mathcal{U}}$ with error $\delta \le \frac{1}{4|\mathcal{U}|}$ requires at least $n \log|\mathcal{U}|/2$ bits of communication.*

We now describe how to solve $\mathrm{IND}_{n,\mathcal{U}}$ in one round of communication via a non-adaptive trace estimation protocol.

Proof of Theorem F.1. Let $\kappa = 1/4\delta^{p/2}$, $n = (\sqrt{\log(3/\delta)}/\varepsilon)^p$, $m = c/(4\delta^{p/2}\varepsilon^p)$ for $c > 0$ a small enough constant, and $\mathcal{U} = [\kappa]$. In the following, we view $\mathcal{U}$ equivalently as the collection of one-hot encodings, i.e., 1-sparse vectors in $\{0,1\}^\kappa$. Let x, y be an instance of $\mathrm{IND}_{n,\mathcal{U}}$ and i be the special index under the promise. Given Alice's input $x \in \{0,1\}^{1/\varepsilon^2 \times \kappa}$ and $\varepsilon, \delta \in (0, 1/4)$, we construct an $n \times n$ real square matrix $\boldsymbol{A}$, as follows.

- Let $\boldsymbol{B} \in \{0,1\}^{m \times m}$ have all rows but the i-th row being the all-zeros vector;
- The i-th row of $\boldsymbol{B}$ is the vector $v = x$ (with precisely c/ε^p non-zero entries).
- Let $\boldsymbol{A} = \frac{1}{n}\boldsymbol{G}\boldsymbol{B}\boldsymbol{G}^\top$, where $\boldsymbol{G} \in \mathbb{R}^{n \times m}$ is a random matrix with i.i.d. standard Gaussian entries.

To solve $\mathrm{IND}_{n,\mathcal{U}}$, it suffices for Bob to recover v_i with probability at least $1-\delta$. By construction, we immediately have that $\operatorname{Tr} \boldsymbol{B} = v_i$ and $\|\boldsymbol{B}\|_F = \sqrt{c}/\varepsilon^{p/2}$. Moreover, by the guarantee of Hutchinson's estimator (see, e.g., Lemma 2 of [18]),

$$|\operatorname{Tr} \boldsymbol{A} - \operatorname{Tr} \boldsymbol{B}| \le \varepsilon^{p/2}(\log(1/\delta))^{1/2-p/2}\|\boldsymbol{B}\|_F = \sqrt{c}(\log(1/\delta))^{1/2-p/2} \le \sqrt{c}$$

with probability at least $1-\delta/3$. By the Johnson-Lindenstrauss lemma, $\|\boldsymbol{A}\|_F \le \sqrt{c}$ with probability $1-\delta/3$. By construction, $\boldsymbol{B}$ has rank one and so $\boldsymbol{A}$ has rank one. It follows that $\|\boldsymbol{A}\|_p = \|\boldsymbol{A}\|_F \le \sqrt{c}$ for any p.

Now suppose that there is a non-adaptive trace estimation protocol that has ε approximation error and $\delta/3$ failure rate, using r queries $\boldsymbol{Q} = (q_1, q_2, \cdots, q_r)$. To finish the reduction, Alice sends matrix $\boldsymbol{AQ}$ to Bob. Bob can obtain an estimate t such that with probability at least $1-\delta/3$, $|t - \operatorname{Tr} \boldsymbol{A}| \le \varepsilon\|\boldsymbol{A}\|_p$. Now taking a union bound and applying the triangle inequality, we have that with probability $1-\delta$,

$$\begin{aligned} |t - v_i| = |t - \operatorname{Tr} \boldsymbol{B}| &\le |t - \operatorname{Tr} \boldsymbol{A}| + |\operatorname{Tr} \boldsymbol{A} - \operatorname{Tr} \boldsymbol{B}| \\ &\le \sqrt{c} + \sqrt{c}\varepsilon \\ &< 1/2, \end{aligned}$$

for $\varepsilon < 1/4$ and a sufficiently small c (say, $c < 0.01$). Hence, Bob can recover v_i and compute $\mathrm{IND}_{n,\mathcal{U}}$.

On the other hand, by Theorem F.2, there is a communication lower bound of $\Omega((\log(1/\delta)/\varepsilon^2)$ bits for the problem. Each entry of $v\boldsymbol{Q}$ is specified by $O(\log(1/\varepsilon) + k)$ bits, so the total communication of sending $\boldsymbol{AQ}$ is $O(\log(1/\varepsilon) + k) \cdot r$. This leads to a query lower bound of $r \ge \Omega((\log(1/\delta))/(\varepsilon^2(\log(1/\varepsilon) + k)))$, as claimed. □

G Experimental Details

G.1 Experimental Results on Synthetic Data

We follow a similar experimental set-up as in [6] and consider small and large perturbations. We also report the average absolute error over all time steps and all trials. In the small perturbation regime, our algorithm achieves errors (average error: 0.0104) that are negligible in comparison with DeltaShift (average error: 1.9804) and other procedures. In the high perturbation regime, our algorithm (average error: 1.6607) outperforms Hutchinson's and Diffsum and is comparable with Deltashift (average error: 1.5868). We notice, across a variety of regimes, that Hutchinson's estimator and Diffsum tend to accumulate estimation error over the dynamic updates, whereas our algorithm and DeltaShift remain stable.

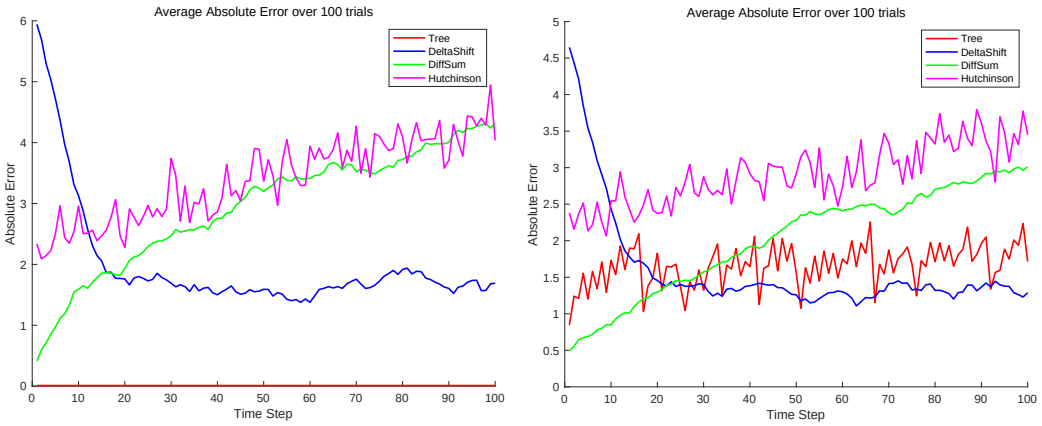
G.2 Experimental Setup

Allocation of query budget. We allocate the same query budget in each time step of DeltaShift and in Hutchinson's estimator. For DiffSum, we allocate $1/5$ of the budget for estimating $\operatorname{Tr} \boldsymbol{A}_1$ and an equal number of queries among the remaining steps. To optimize performance, the number of groups in our algorithm is tuned.

Experiments on synthetic data. On both small and large perturbation experiments, we choose the dimension to be $n = 1000$. The first matrix $\boldsymbol{A}_1$ in the sequence is a symmetric matrix with random (unit-norm) eigenvectors and eigenvalues drawn uniformly from $[-1, 1]$. In the small perturbation regime, a random rank-1 matrix $\Delta_j = 5e^{-5}rgg^\top$ is added in each time step, where r is a random sign and g is a standard Gaussian in n dimensions. In the large perturbation regime, each update is a random rank-20 positive semidefinite matrix.

Neural network weight matrices. The network consists of two hidden layers of the same size, with standard ReLU activations. The mini-batch size is set to 60 and learning rate is set to 0.01.

We optimized the performance of our trace estimation algorithm by choosing its number of groups to be 20.



(a) Low perturbation

(b) High perturbation

Figure 3: Synthetic data, Tree refers to our algorithm (Algorithm 1). Query budget is $8,000$. We measure the error at step i simply by absolute error $|t_i - \mathrm{Tr}\, \boldsymbol{A}_i|$