FROSH: FasteR Online Sketching Hashing

Xixian Chen[1,2], **Irwin King**[1,2], **Michael R. Lyu**[1,2]
[1]Shenzhen Research Institute, The Chinese University of Hong Kong, Shenzhen, China
[2]Department of Computer Science and Engineering,
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
{xxchen, king, lyu}@cse.cuhk.edu.hk

Abstract

Many hashing methods, especially those that are in the data-dependent category with good learning accuracy, are still inefficient when dealing with three critical problems in modern data analysis. First, data usually come in a streaming fashion, but most of the existing hashing methods are batch-based models. Second, when data become huge, the extensive computational time, large space requirement, and multiple passes to load the data into memory will be prohibitive. Third, data often lack sufficient label information. Although the recently proposed Online Sketching Hashing (OSH) is promising to alleviate all three issues mentioned above, its training procedure still suffers from a high time complexity. In this paper, we propose a FasteR Online Sketching Hashing (FROSH) method to make the training process faster. Compared with OSH, we leverage fast transform to sketch data more compactly. Particularly, we derive independent transformations to guarantee the sketching accuracy, and design a novel implementation to make such transformations applicable to online data sketching without increasing the space cost. We rigorously prove that our method can yield a comparable learning accuracy with a lower time complexity and an equal space cost compared with OSH. Finally, extensive experiments on synthetic and real-world datasets demonstrate the excellent performance of our method.

1 INTRODUCTION

Hashing is an efficient method to conduct an approximate nearest neighbor search, which is critical for machine learning and applications such as clustering, retrieval and matching [43]. It transforms data into a low-dimensional representation, i.e., a short hash code consisting of a sequence of bits in Hamming space. The data distance in original space then is approximated by the Hamming distance that can be calculated extremely fast in modern CPU. Thus, the approximate nearest neighbor search can be accomplished with less time and space costs. Current hashing methods can be categorized broadly as data-independent and data-dependent techniques. Locality Sensitive Hashing (LSH) methods [3, 8, 12, 15, 39] are prime examples of data-independent techniques, which are also unsupervised. They construct hash functions based on random projection, which are typically very fast and theoretically guaranteed, but are developed only for certain distance functions and often require long code length to achieve acceptable search accuracy because of ignoring the data distribution. Compared with data-independent methods, data-dependent hashing techniques achieve better accuracy performance with shorter binary codes, while usually incurring a larger computational cost to train the hashing functions. These data-dependent methods can be categorized as unsupervised and (semi-)supervised techniques. In unsupervised studies [13, 17, 22, 27–29, 33, 37, 44, 49], hash functions are learned from data distribution rather than being randomly generated by preserving a metric induced distance in the Hamming space. Supervised methods [19, 23, 24, 26, 35, 38, 42] additionally leverage the label information, and thus often outperform those unsupervised ones.

Although data-dependent hashing methods have achieved a promising learning accuracy, yet they suffer from some critical problems when confronted with the data such as web images, videos, stocks, genomes and web documents in the big data era. First, data often become available continuously in a streaming fashion, and each data point or data chunk can be processed only once [22, 25]. Hence, batch learning strategies

are not allowed. Moreover, with batch-learners, it is unclear how to adapt the hash functions as the dataset continues to grow and new variations appear over time. Second, the data size n and dimension d can be large with $1 \ll d \ll n$ [10], so that the high time complexity and $O(nd)$ space cost [22, 25] are prohibitive. In particular, advanced batch-based unsupervised hashing solutions such as SGH [17] and OCH [27] can avoid the $O(nd)$ space cost merely by performing multiple passes over the data, while the disk IO overhead for loading all data into memory multiple times will be the major performance bottleneck [47]. Third, labels are commonly missing, noisy, and scarce in today's big data situation, and labeling streaming data is also expensive and infeasible [40, 46, 48].

To tackle the above challenges, we focus on the most recently proposed online hashing, i.e., online sketching hashing (OSH) [22], which is data-dependent, space-efficient, unsupervised, and in a single pass. Note that there have been several other investigations in online hashing. Such hashing methods include online kernel-based hashing (OKH) [14], adaptive online hashing (AOH) [5], and [6] but they all belong to the supervised category, which require extra label information.

However, given $O(d\ell)$ space to store and perform calculation on the streaming data, the OSH method retains a training time of $O(nd\ell + d\ell^2)$ to yield $r \leq O(\ell)$ hashing bits, where n is the data size, d is the data dimension, and ℓ denotes the sketching size satisfying $\ell < d \ll n$. Such training time is still expensive since $1 \ll d \ll n$ [10]. In this paper, we attempt to reduce the running time further. Our contributions are summarized as follows:

- First, we propose a FasteR Online Sketching Hashing (FROSH) method by improving the data sketching procedure in the OSH method. Specifically, we employ a fast transform to reduce the data size, in which independent transformations are applied for different small data chunks to make the sketching compact and accurate.
- Second, we design a more efficient way to implement the fast transform in our FROSH. Compared with the standard way, our strategy reduces the space burden incurred by the fast transform for sketching streaming data while maintaining the same time efficiency.
- Third, we analyze the accuracy of our FROSH, and give an error bound comparable with OSH. Moreover, our FROSH has a smaller time complexity of $\widetilde{O}(n\ell^2 + nd)$ with an equal space usage. Extensive experiments demonstrate the computational efficiency with a competitive learning accuracy.

The remainder of this paper is organized as follows. In Section 2, we review the prior work and techniques. In Section 3, we present our method along with theoretical analysis, and emphasize its advantages from various aspects. In Section 4, we provide extensive empirical results, and Section 5 concludes the whole work.

Notation. Let $[k]$ denote a set of integers $\{1, 2, \ldots, k\}$. Given a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$, for $i \in [n]$, $j \in [d]$, we let $\mathbf{a}^i \in \mathbb{R}^{1\times d}$ denote the i-th row of $\mathbf{A}$, $\mathbf{a}_j \in \mathbb{R}^n$ denote the j-th column of $\mathbf{A}$, and a_{ij} denote the (i,j)-th element of $\mathbf{A}$ or j-th element of $\mathbf{a}^i$. Let $\{\mathbf{A}_t\}_{t=1}^k$ denote a set of matrices $\{\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_k\}$. Let $a_{t,ij}$ and $\mathbf{a}_{t,i}$ denote the (i,j)-th element and i-th column of matrix $\mathbf{A}_t$, respectively. Let $\mathbf{A}^T$ denote the transpose of $\mathbf{A}$, and $\mathrm{Tr}(\mathbf{A})$ denote its trace. Let $|a|$ denote the absolute value of a. Let $\|\mathbf{A}\|_2$ and $\|\mathbf{A}\|_F$ denote the spectral norm and Frobenius norm of $\mathbf{A}$, respectively. Let $\|\mathbf{a}\|_q = (\sum_{j=1}^d |a_j|^q)^{1/q}$ for $q \geq 1$ be the ℓ_q-norm of $\mathbf{a} \in \mathbb{R}^d$. We denote the SVD of $\mathbf{A}$ by $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \sum_{i=1}^{\rho} \sigma_i \mathbf{u}_i \mathbf{v}_i^T = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^T + \mathbf{U}_k^{\perp}\mathbf{\Sigma}_k^{\perp}{\mathbf{V}_k^{\perp}}^T$, where $\mathrm{rank}(\mathbf{A}) = \rho$, $\mathbf{A}_k = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^T$ represents the best rank k approximation to $\mathbf{A}$, and $\sigma_i(\mathbf{A})$ denotes the i-th largest singular value of $\mathbf{A}$. Finally, $\mathbf{A} \preceq \mathbf{B}$ means that $\mathbf{B} - \mathbf{A}$ is positive semi-definite.

2 RELATED WORK

In this section, we describe the properties of the OSH method. Then, we outline and discuss the fast transforms, especially the Subsampled Randomized Hadamard Transform (SRHT).

2.1 ONLINE SKETCHING HASHING

We rephrase the background and details of the OSH method [22]. Given data $\mathbf{A} \in \mathbb{R}^{n\times d}$ and the unknown projection matrix $\mathbf{W} \in \mathbb{R}^{d\times r}$, the k-th hashing function for a data point $\mathbf{a}^i \in \mathbb{R}^{1\times d}$ is defined as

$$h_k(\mathbf{a}^i) = \mathrm{sgn}((\mathbf{a}^i - \boldsymbol{\mu})\mathbf{w}_k), \tag{1}$$

where $\boldsymbol{\mu} = \bar{\mathbf{A}} = \frac{1}{n}\sum_{i=1}^n \mathbf{a}^i$. By dropping the non-differentiable function $\mathrm{sgn}(\cdot)$ for the binary codes [42], the objective can be reformulated as the same as that of Principal Component Analysis (PCA)

$$\begin{aligned} \max_{\mathbf{W}\in\mathbb{R}^{d\times r}} \quad & \mathrm{Tr}(\mathbf{W}^T(\mathbf{A}-\boldsymbol{\mu})^T(\mathbf{A}-\boldsymbol{\mu})\mathbf{W}) \\ \text{s.t.} \quad & \mathbf{W}^T\mathbf{W} = \mathbf{I}_r, \end{aligned} \tag{2}$$

where $(\mathbf{A}-\boldsymbol{\mu})$ denotes a matrix $[\mathbf{a}^1-\boldsymbol{\mu}; \mathbf{a}^2-\boldsymbol{\mu}; \ldots; \mathbf{a}^n-\boldsymbol{\mu}]$.

The solution to $\mathbf{W} \in \mathbb{R}^{d\times r}$ in Eq. (2) is the top r right eigenvectors of the covariance matrix $(\mathbf{A}-\boldsymbol{\mu})^T(\mathbf{A}-\boldsymbol{\mu})$.

Algorithm 1 Online Sketching Hashing (OSH) [22]

Input: Data $\mathbf{A} = \{\mathbf{A}_j \in \mathbb{R}^{h_j \times d}\}_{j=1}^{s}$, sketching size $\ell < d$, positive integer η, hashing bits r

1: Initialize sketching matrix by $\mathbf{B} = \mathbf{0}^{\ell \times d}$
2: Set $\boldsymbol{\mu}_1$ as the row mean vector of $\mathbf{A}_1$
3: Let $\boldsymbol{\varphi} = \boldsymbol{\mu}_1$, $\tau = h_1$, $\xi = 0$
4: Sketch $\mathbf{G}_1 = (\mathbf{A}_1 - \boldsymbol{\mu}_1) \in \mathbb{R}^{h_1 \times d}$ into $\mathbf{B} \in \mathbb{R}^{\ell \times d}$
5: **for** $j = 2 : s$ **do**
6: Set $\boldsymbol{\mu}_j$ as the row mean vector of $\mathbf{A}_j$
7: Set $\boldsymbol{\varsigma} = \sqrt{\frac{\tau h_j}{\tau + h_j}}(\boldsymbol{\mu}_j - \boldsymbol{\varphi}) \in \mathbb{R}^{1 \times d}$
8: Sketch $\mathbf{G}_j = [(\mathbf{A}_j - \boldsymbol{\mu}_j); \boldsymbol{\varsigma}] \in \mathbb{R}^{(h_j+1) \times d}$ into $\mathbf{B} \in \mathbb{R}^{\ell \times d}$
9: Set $\boldsymbol{\varphi} = \frac{\tau \boldsymbol{\varphi}}{\tau + h_j} + \frac{h_j \boldsymbol{\mu}_j}{\tau + h_j}$ # Update the mean vector
10: Set $\tau = \tau + h_j$ # Update the data size
11: Set $\xi = \xi + 1$
12: **if** $\xi == \eta$ **then**
13: Compute the SVD of $\mathbf{B} \in \mathbb{R}^{\ell \times d}$, and assign the top r right singular vectors as $\mathbf{W}^T \in \mathbb{R}^{r \times d}$
14: Set $\xi = 0$
15: **return** $\mathbf{W}$

For $d < n$, it takes $O(nd^2)$ time and $O(nd)$ space, which is infeasible for large n and d [18].

To tackle above computational issues, sketching the data before the training is a promising way. Specifically, a sketch of a data matrix is another matrix that is significantly smaller than the original but still approximates it well and preserves the properties of interest. It implies that the storing and the computing on the sketch will be much easier than with the original large matrix, and the downstream learning algorithms on the sketch can still guarantee the learning accuracy [4, 9, 25, 32, 45].

Leveraging the effectiveness of sketching in the learning, OSH is proposed, and its details are presented in Algorithm 1. It aims at efficiently achieving a small matrix $\mathbf{B} \in \mathbb{R}^{\ell \times d}$ for the centered data $(\mathbf{A} - \boldsymbol{\mu})$, then computing the SVD on $\mathbf{B}$ only takes $O(d\ell^2)$ time and $O(d\ell)$ space for $\ell < d$. The difficulty lies in that sketching cannot be directly applied in each centered data chunk $(\mathbf{A}_j - \boldsymbol{\mu})$, since $\boldsymbol{\mu}$ can only be obtained after observing all data points in $\mathbf{A}$.

Then, the *online centering procedure* in steps 3 to 10 are to ensure that $(\mathbf{A}_{[j]} - \hat{\boldsymbol{\mu}}_j)^T(\mathbf{A}_{[j]} - \hat{\boldsymbol{\mu}}_j) = \mathbf{G}_{[j]}^T\mathbf{G}_{[j]}$ after the j-th iteration, where $\mathbf{A}_{[j]} = [\mathbf{A}_1; \mathbf{A}_2; \cdots; \mathbf{A}_j]$ by stacking all observed $\mathbf{A}_j$ vertically, $\mathbf{G}_{[j]}$ follows a similar definition, and $\hat{\boldsymbol{\mu}}_j$ is the row mean vector of $\mathbf{A}_{[j]}$. Steps 4 and 8 are to get a smaller matrix $\mathbf{B}$ via an efficient sketching such that $\mathbf{B}^T\mathbf{B} \approx \mathbf{G}_{[j]}^T\mathbf{G}_{[j]}$ after the j-th iteration. After all iterations, OSH guarantees that $\mathbf{B}^T\mathbf{B} \approx \mathbf{G}_{[s]}^T\mathbf{G}_{[s]} = (\mathbf{A}_{[s]} - \hat{\boldsymbol{\mu}}_s)^T(\mathbf{A}_{[s]} - \hat{\boldsymbol{\mu}}_s) = (\mathbf{A} - \boldsymbol{\mu})^T(\mathbf{A} - \boldsymbol{\mu})$.

If $\sum_{j=1}^{s} h_j = O(n)$, the sketching steps in OSH will take $O(nd\ell)$ time and $O(d\ell)$ space. The SVD in step 13 runs for about ξ/η times. As ξ/η can be manually set to be a small constant, step 13 takes $O(d\ell^2)$ time and $O(d\ell)$ space in total. The computational cost of the online data centering procedure is negligible. In conclusion, OSH consumes $O(nd\ell + d\ell^2)$ time and $O(d\ell)$ space.

Remark 1. To address the problem that most of the information can be contained by only a small number of significant singular vectors in $\mathbf{W} \in \mathbb{R}^{d \times r}$, OSH also empirically applies a random orthogonal rotation $\boldsymbol{\Upsilon} \in \mathbb{R}^{r \times r}$ (the orthonormal bases of an $r \times r$ random Gaussian matrix) to all singular vectors $\mathbf{W} \in \mathbb{R}^{d \times r}$ in Algorithm 1 via $\mathbf{W}\boldsymbol{\Upsilon}$. This step resembles Iterative Quantization [13] but runs much more efficiently with streaming settings maintained and negligible computational cost incurred.

2.2 FREQUENT DIRECTIONS FOR SKETCHING

Algorithm 2 Frequent Directions (FD) [25]

Input: Data $\mathbf{A} \in \mathbb{R}^{n \times d}$, sketching matrix $\mathbf{B} \in \mathbb{R}^{\ell \times d}$

1: **if** $\mathbf{B}$ not exists **then**
2: Set $\mathbf{B} = \mathbf{0}^{\ell \times d}$
3: **for** $i \in [n]$ **do**
4: Insert $\mathbf{a}^i \in \mathbb{R}^{1 \times d}$ into a zero valued row of $\mathbf{B}$
5: **if** $\mathbf{B}$ has no zero valued rows **then**
6: $[\mathbf{U}, \boldsymbol{\Sigma}, \mathbf{V}] = \mathrm{SVD}(\mathbf{B})$
7: $\widehat{\boldsymbol{\Sigma}} = \sqrt{\max(\boldsymbol{\Sigma}^2 - \sigma_{\ell/2}^2\mathbf{I}_l, 0)}$
8: Set $\mathbf{B} = \widehat{\boldsymbol{\Sigma}}\mathbf{V}^T$

OSH employs the Frequent Directions (FD) as a black-box to sketch streaming data, which is summarized in Algorithm 2. It operates by keeping collecting row vectors from the data source. Once the sketching matrix $\mathbf{B} \in \mathbb{R}^{\ell \times d}$ has ℓ non-zero rows, the shrinking procedure as shown in steps 5 to 8 will reduce the size of the matrix $\mathbf{B} \in \mathbb{R}^{\ell \times d}$ by a half, *which repeats throughout the entire streaming data*.

In this part, we assume that FD will process n data points. The computational cost of the FD algorithm is dominated by the shrinking procedure that involves computing an exact SVD on $\mathbf{B} \in \mathbb{R}^{\ell \times d}$, which takes $O(d\ell^2)$ time and $O(d\ell)$ space for $\ell < d$. Since the shrinking procedure is operated once nearly every $\ell/2$ iterations of the main loop, the time complexity is $O(d\ell^2 \times n/\ell) = O(nd\ell)$ with $O(d\ell)$ space burden.

Recently, the Sparse FD (SFD) algorithm [11] is proposed to take advantage of the sparsity of $\mathbf{A}$ by applying a powerful randomized SVD [34] instead of an exact SVD in each shrinking procedure. SFD still takes $O(d\ell)$

space but only requires $\widetilde{O}(\text{nnz}(\mathbf{A})\ell + n\ell^2)$ running time, where $\text{nnz}(\mathbf{A})$ means the number of non-zero entries in $\mathbf{A}$. This time cost is much smaller than $O(nd\ell)$ if $\ell \ll d$ and $\mathbf{A} \in \mathbb{R}^{n\times d}$ is extremely sparse.

Remark 2. FD works well for streaming data. For instance, given two (or more) data chunks like $\mathbf{G}_1 \in \mathbb{R}^{h_1\times d}$ and $\mathbf{G}_2 \in \mathbb{R}^{(h_2+1)\times d}$ in Algorithm 1, we invoke FD and run it on $\mathbf{G}_1$ to yield a sketching matrix $\mathbf{B} = \mathbf{B}_1 \in \mathbb{R}^{\ell\times d}$. Based on the current $\mathbf{B} = \mathbf{B}_1$, we then run FD on $\mathbf{G}_2$ to update $\mathbf{B}$ and get $\mathbf{B} = \mathbf{B}_2 \in \mathbb{R}^{\ell\times d}$. Such procedures are obviously equivalent to that we directly invoke FD for once and run it on $[\mathbf{G}_1; \mathbf{G}_2] \in \mathbb{R}^{(h_1+h_2+1)\times d}$, which also gets an identical $\mathbf{B} = \mathbf{B}_2$.

2.3 FAST TRANSFORM

Given a vector $\mathbf{a} \in \mathbb{R}^m$, its compressed representation $\mathbf{b} \in \mathbb{R}^q$ can be obtained by performing the matrix-vector multiplication like $\mathbf{\Phi a}$, where $q < m$ and $\mathbf{\Phi} \in \mathbb{R}^{q\times m}$ is a random projection matrix (e.g., a Gaussian random matrix). Generally, computing $\mathbf{\Phi a}$ takes $O(qm)$ time. It can be a computational bottleneck of fast implementing numerous learning tasks that involve data compression.

Fast transforms, which are based on the structured projection matrix like Hadamard matrix or Fourier matrix, then are employed to overcome the shortcomings of the classical transformation methods. For Hadamard matrix $\mathbf{H}_m \in \mathbb{R}^{m\times m}$, its entry is defined as

$$\mathbf{H}_{ij} = (-1)^{\langle i-1, j-1\rangle}, \tag{3}$$

where $\langle i-1, j-1\rangle$ is the dot-product of the b-bit vectors of integers $i-1$ and $j-1$ expressed in binary, and $b \leq \max(\lceil \log(i+1)\rceil, \lceil \log(j+1)\rceil)$. It can also be defined recursively as

$$\mathbf{H}_m = \begin{bmatrix} \mathbf{H}_{m/2} & \mathbf{H}_{m/2} \\ \mathbf{H}_{m/2} & -\mathbf{H}_{m/2} \end{bmatrix} \quad \text{and} \quad \mathbf{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

The normalized Hadamard matrix is defined by $\mathbf{H} = \sqrt{\frac{1}{m}}\mathbf{H}_m$, and by exploring the recursive structure of $\mathbf{H}$, computing $\mathbf{Ha}$ only takes $O(m \log m)$ time with $O(m)$ space [1].

To compress data, we perform the Subsampled Randomized Hadamard Transform (SRHT), i.e., data is transformed via a randomized Hadamard matrix and then uniformly sampled in the resulting entries. For a $q \times m$ SRHT matrix, it is defined as the form $\mathbf{\Phi} = \mathbf{SHD}$, where $\mathbf{D} \in \mathbb{R}^{m\times m}$ is a diagonal matrix with its diagonal elements being i.i.d. Rademacher random variables (i.e., 1 or -1 with equal probability), $\mathbf{S} \in \mathbb{R}^{q\times m}$ is a scaled sampling matrix with each row uniformly sampled *without replacement* from m rows of the identity matrix $\mathbf{I}_m \in \mathbb{R}^{m\times m}$ multiplied by $\sqrt{\frac{m}{q}}$.

Differently, computing $\mathbf{\Phi a}$ only takes $O(m \log q)$ time with $O(m)$ space [1, 31].

3 FASTER ONLINE SKETCHING HASHING

In this section, we motivate and present FROSH, whose sketching relies on our Faster FD (FFD). To make FROSH maintain an equal space efficiency with that of OSH, we also derive a new implementation of fast transform in FFD for sketching streaming data. The analysis guarantees the performance of our solution.

3.1 MOTIVATION, METHOD AND RESULTS

The computational cost of OSH is dominated by its sketching method FD. This sketching requires $O(nd\ell)$ time to obtain $\mathbf{B} \in \mathbb{R}^{\ell\times d}$ from the data $\mathbf{A} \in \mathbb{R}^{n\times d}$, which is computationally expensive for $1 \ll d \ll n$ [10]. Even regarding SFD, its $\widetilde{O}(\text{nnz}(\mathbf{A})\ell + nl^2)$ time cost still tends to be $\widetilde{O}(nd\ell + n\ell^2)$ since the centered data $\mathbf{A} - \boldsymbol{\mu}$ are dense.

We first present an FFD sketching method as stated in Algorithm 3 to tackle above issues. In steps 5 to 8, after collecting m data points into $\mathbf{F} \in \mathbb{R}^{m\times d}$, a new SRHT matrix is generated to compress $\mathbf{F}$ with its size reduced to $\ell/2$. Then, the newly compressed data and previously shrunken data form the $\mathbf{B} \in \mathbb{R}^{\ell\times d}$. Steps 9 to 10 employ a shrinking procedure that is similar to that in FD.

Algorithm 3 Faster Frequent Directions (FFD)

Input: Data $\mathbf{A} \in \mathbb{R}^{n\times d}$, sketching matrix $\mathbf{B} \in \mathbb{R}^{\ell\times d}$

1: **if** $\mathbf{B}$ not exists **then**
2: Set $t = 1$, $\mathbf{B} = \mathbf{0}^{\ell\times d}$
3: Set $\mathbf{F} = \mathbf{0}^{m\times d}$ with $m = \Theta(d)$ # Only needed for notation
4: **for** $i \in [n]$ **do**
5: Insert $\mathbf{a}^i$ into a zero valued row of $\mathbf{F}$
6: **if** $\mathbf{F}$ has no zero valued rows **then**
7: Generate a fast transform $\mathbf{\Phi}_t = \mathbf{S}_t\mathbf{H}\mathbf{D}_t \in \mathbb{R}^{(\ell/2)\times m}$ in the t-th trial
8: Insert $\mathbf{\Phi}_t\mathbf{F} \in \mathbb{R}^{(\ell/2)\times d}$ into the last $\frac{\ell}{2}$ rows of $\mathbf{B}$
9: $[\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}] = \text{SVD}(\mathbf{B})$
10: $\widehat{\mathbf{\Sigma}} = \sqrt{\max(\mathbf{\Sigma}^2 - \sigma^2_{\ell/2}\mathbf{I}_l, 0)}$
11: Set $\mathbf{B} = \widehat{\mathbf{\Sigma}}\mathbf{V}^T$
12: Let $\mathbf{B}_t = \mathbf{B}$, $\mathbf{C}_t = \mathbf{\Phi}_t\mathbf{F}$ # Only needed for proof notations
13: Set $\mathbf{F} = \mathbf{0}^{m\times d}$, $t = t + 1$

However, there may exist one problem in each iteration of step 8, i.e., computing $\mathbf{\Phi}_t\mathbf{F}$ in the standard way has

to take $O(md\log\ell)$ time but with $O(md)$ space to collect all m data points. For a big $m=\Theta(d)$, the space cost of $O(md)=O(d^2)$ is practically prohibitive since space can be severely limited [25], which is also inferior to FD method whose advantages includes that only $O(d\ell)$ space is employed.

Fortunately, we do not have to keep a matrix $\mathbf{F}\in\mathbb{R}^{m\times d}$ explicitly to store m data points. Instead, $\mathbf{F}\in\mathbb{R}^{m\times d}$ here is simply used for easily presenting the algorithm. Furthermore, as data come sequentially, we only take $O(d\ell)$ space to handle $O(\ell)$ data points, save and combine intermediate results. We repeat such procedures until that we have sequentially processed m data points denoted by $\mathbf{F}\in\mathbb{R}^{m\times d}$, which finally still only takes $O(md\log\ell)$ time in total to achieve $\mathbf{\Phi}_t\mathbf{F}$. The detailed implementation is deferred to the subsequent section.

Moreover, we adopt an independent and distinct $\mathbf{\Phi}_t$ in each iteration of step 8, which benefits the sketching accuracy by controlling the variance in all sketchings. Below, we formally characterize the properties of our proposed FFD.

Theorem 1 (FFD)**.** *Given data $\mathbf{A}\in\mathbb{R}^{n\times d}$ and the sketching size $\ell\le k=\min(m,d)$, let the small sketch $\mathbf{B}\in\mathbb{R}^{\ell\times d}$ be constructed by FFD. Then, with probability at least $1-p\beta-(2p+1)\delta-\frac{2n}{e^k}$ we have*

$$\|\mathbf{A}^T\mathbf{A}-\mathbf{B}^T\mathbf{B}\|_2\le\widetilde{O}\Big(\frac{1}{\ell}+\Gamma(\ell,p,k)\Big)\|\mathbf{A}\|_F^2 \quad (4)$$

where $\Gamma(\ell,p,k)=\sqrt{\frac{k}{\ell p^2}}+\sqrt{\frac{1+\sqrt{k/\ell}}{p}}$ with $p=\frac{n}{m}$, and $\widetilde{O}(\cdot)$ hides logarithmic factors on (β,δ,k,d,m) as given in Eqs. (12)(16)(30).

The running time of the algorithm is $\widetilde{O}(nl^2\frac{d}{m}+nd)$ and its space cost is $O(d\ell)$ before taking $m=\Theta(d)$.

Remark 3. For a fixed n, a smaller m improves the sketching accuracy while taking more time burden. We choose $m=\Theta(d)$ such that the time consumption of FFD is $\widetilde{O}(n\ell^2+nd)$. If $\ell\ll d$, it is superior to $O(nd\ell)$ time cost in FD and $\widetilde{O}(nd\ell+n\ell^2)$ time usage in SFD for dense centered data.

When data keep coming (i.e., n increases) and/or m decreases, p will get larger, which makes the bound in Eq. (4) tighter. If $p=\Omega(\ell^{3/2}k^{1/2})$, i.e., $n=\Omega(\ell^{3/2}d^{3/2})$ when $m=\Theta(d)$, then the error bound of FFD becomes $\widetilde{O}(\frac{1}{\ell}\|\mathbf{A}\|_F^2)$, which is asymptotically comparable with the bounds $\frac{2}{\ell}\|\mathbf{A}\|_F^2$ of FD and $\widetilde{O}(\frac{1}{\ell}\|\mathbf{A}\|_F^2)$ of SFD.

Thus, FFD is more applicable to the big data situation $1\ll d\ll n$ with computational cost reduced and accuracy guaranteed. Competitive empirical results suggest there is an opportunity to further tighten our error bound.

Algorithm 4 FasteR Online Sketching Hashing (FROSH)

Input: Data $\mathbf{A}=\{\mathbf{A}_j\in\mathbb{R}^{h_j\times d}\}_{j=1}^s$, sketching size $\ell<d$, positive integer η, hashing bits r

1: Run steps 1 to 3 of Algorithm 1
2: Run steps 4 to 10 of Algorithm 1 with FFD to sketch the centered data into $\mathbf{B}\in\mathbb{R}^{\ell\times d}$
3: Run steps 11 to 15 of Algorithm 1 and return $\mathbf{W}\in\mathbb{R}^{d\times r}$

Based on FFD and OSH, our FROSH is straightforward given in Algorithm 4. Its step 2 contains the online data centering procedure, which is equivalent to that in OSH. Then, combining Theorem 1 and the properties of the online data centering procedure yields the next result.

Theorem 2 (FROSH)**.** *Given data $\mathbf{A}\in\mathbb{R}^{n\times d}$ with its row mean vector $\boldsymbol{\mu}\in\mathbb{R}^{1\times d}$, let the sketch $\mathbf{B}^{\ell\times d}$ be generated by FROSH in Algorithm 4. Then, with probability defined in Theorem 1 we have*

$$\|(\mathbf{A}-\boldsymbol{\mu})^T(\mathbf{A}-\boldsymbol{\mu})-\mathbf{B}^T\mathbf{B}\|_2\le\widetilde{O}\Big(\frac{1}{\ell}+\Gamma(\ell,p,k)\Big)\|\mathbf{A}-\boldsymbol{\mu}\|_F^2, \quad (5)$$

where $(\mathbf{A}-\boldsymbol{\mu})\in\mathbb{R}^{n\times d}$ means subtracting each row of $\mathbf{A}$ by $\boldsymbol{\mu}$, $\Gamma(\ell,p,k)$ is from Theorem 1, and the top r right singular vectors of $\mathbf{B}^{\ell\times d}$ are used for hashing projections $\mathbf{W}^T\in\mathbb{R}^{r\times d}$ in Remark 1.

The algorithm takes $\widetilde{O}(n\ell^2+nd+d\ell^2)$ time and $O(d\ell)$ space cost after taking $m=\Theta(d)$ in the FFD of FROSH.

Remark 4. This primary analysis resembles that of OSH [22], which follows the conception that accurate sketching does not harm the learning accuracy [4, 9, 22, 25, 32, 45]. Hence, the $\mathbf{W}$ from $\mathbf{B}$ is expected to well approximate that from $\mathbf{A}-\boldsymbol{\mu}$ (more illustrations on the approximation for $\mathbf{W}$ are deferred into the appendix), and we use this sketching error for the centered data $\mathbf{A}-\boldsymbol{\mu}$ to justify if the sketching-based hashing works properly [22]. Thus, the accuracy comparisons between OSH and FROSH resemble Remark 3. Compared with $\widetilde{O}(nd\ell+d\ell^2)$ time and $O(d\ell)$ space cost in OSH, we improve a lot if $1<\ell\ll d\ll n$.

3.2 IMPLEMENTATION OF THE FAST TRANSFORM IN FFD

Define $q=\ell/2$, then we show how to simply adopt $O(qd)$ space and perform a single pass through $\mathbf{F}\in\mathbb{R}^{m\times d}$ to achieve $\mathbf{\Phi F}$ with $O(md\log q)$ time, given $\mathbf{\Phi}=\mathbf{SHD}\in\mathbb{R}^{q\times m}$, $q\le m$, and $\log m\le O(d\log q)$.

To clarify, we choose an m such that $m=2^b$ for a positive integer b. We can always find a $q/2\le p\le q$ to define positive integers c and p such that $m=2^cp$.

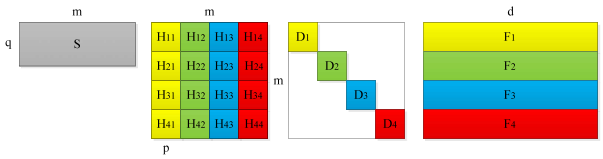

Figure 1: Space-efficient implementation of $\mathbf{\Phi F} = \mathbf{SHDF}$. We set $m/p = 2^c$ with $c = 2$ for clarity.

As shown in Figure 1, we divide data matrix $\mathbf{F} \in \mathbb{R}^{m\times d}$ into 2^c blocks as $\{\mathbf{F}_i \in \mathbb{R}^{p\times d}\}_{i=1}^{2^c}$, and the diagonal matrix $\mathbf{D} \in \mathbb{R}^{m\times m}$ into 2^c square blocks as $\{\mathbf{D}_i \in \mathbb{R}^{p\times d}\}_{i=1}^{2^c}$. Hadamard matrix $\mathbf{H} \in \mathbb{R}^{m\times m}$ can also be divided into $(2^c)^2 = 4^c$ square blocks $\{\mathbf{H}_{ij} \in \mathbb{R}^{p\times p}\}_{i,j=1}^{2^c}$.

We take $O(pd)$ space to receive streaming data from $\mathbf{F}_1 \in \mathbb{R}^{p\times d}$. Then, perform the matrix multiplication through $\mathbf{S}[\mathbf{H}_{11}; \mathbf{H}_{21}; \cdots; \mathbf{H}_{2^c1}]\mathbf{D}_1\mathbf{F}_1 \in \mathbb{R}^{q\times d}$. We first run $\mathbf{H}_{11}\mathbf{D}_1\mathbf{F}_1 \in \mathbb{R}^{p\times d}$, taking $O(pd\log p)$ time and $O(pd)$ space. We then only have to get at most q resulting rows from $\{\mathbf{H}_{i1}\mathbf{F}_1\}_{i\geq 2}^{2^c}$ since $\mathbf{S} \in \mathbb{R}^{q\times m}$ only selects q rows. Fortunately, $\mathbf{H}_{11} = +\mathbf{H}_{i1}$ or $\mathbf{H}_{11} = -\mathbf{H}_{i1}$ holds for any $i \in [2^c]$, and the $\pm$ sign can be determined by calculating the first entry of each $\mathbf{H}_{i1}$ based on Eq. (3) whose running time is at most $O(\log m)$. Hence, the ultimate time complexity is $O(pd\log p) + O(qd) + O(q\log m)$. It is also straightforward to check that $O(pd + qd)$ space suffices for all calculations and saving the compressed data.

Then, we remove $\mathbf{F}_1$, receive data from $\mathbf{F}_2$, do the similar calculations, and update current result based on the previous compressed data from $\mathbf{F}_1$. The time and space costs on $\mathbf{F}_2$ still keep asymptotically unchanged.

Finally, the space usage of computing $\mathbf{\Phi F}$ is $O(pd) = O(qd)$, and its time cost is $2^c[O(pd\log p) + O(qd) + O(q\log m)] = O(md\log q)$ given $q/2 \leq p \leq q$, $m = 2^c p$, and $\log m \leq O(d\log q)$.

3.3 ERROR ANALYSIS

In our analysis, we turn to series of existing theoretical tools. We use the Matrix Bernstein inequality on the sum of zero-mean random matrices given as below.

Theorem 3 ([41]). *Let $\{\mathbf{A}_i\}_{i=1}^L \in \mathbb{R}^{n\times d}$ be independent random matrices with $\mathbb{E}[\mathbf{A}_i] = \mathbf{0}^{n\times d}$ and $\|\mathbf{A}_i\|_2 \leq R$ for all $i \in [L]$. Define a variance parameter as $\sigma^2 = \max\{\|\sum_{i=1}^L \mathbb{E}\left[\mathbf{A}_i\mathbf{A}_i^T\right]\|_2, \|\sum_{i=1}^L \mathbb{E}\left[\mathbf{A}_i^T\mathbf{A}_i\right]\|_2\}$. Then, for all $\epsilon \geq 0$ we have*

$$\mathbb{P}(\|\sum_{i=1}^L \mathbf{A}_i\|_2 \geq \epsilon) \leq (d+n)\exp(\frac{-\epsilon^2/2}{\sigma^2 + R\epsilon/3}). \quad (6)$$

It is also helpful to provide the next result that characterizes the property of compressed data via SRHT matrix.

Theorem 4 ([31]). *Given $\mathbf{A} \in \mathbb{R}^{m\times d}$, let rank$(\mathbf{A}) \leq k \leq \min(m, d)$ and $\mathbf{\Phi} \in \mathbb{R}^{q\times m}$ be the SRHT matrix. Then, with probability at least $1 - (\delta + \frac{m}{e^k})$ we have*

$$(1-\Delta)\mathbf{A}^T\mathbf{A} \preceq \mathbf{A}^T\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{A} \preceq (1+\Delta)\mathbf{A}^T\mathbf{A}, \quad (7)$$

where $\Delta = \Theta(\sqrt{\frac{k\log(2k/\delta)}{q}})$.

With Corollary 3 of [2], it is straightforward to have the next norm bound for compressed data vectors.

Lemma 1. *Give data matrix $\mathbf{A} \in \mathbb{R}^{m\times d}$, and the SRHT matrix $\mathbf{\Phi} \in \mathbb{R}^{q\times m}$. Then, with probability at least $1-\beta$, we have*

$$\|\mathbf{\Phi}\mathbf{a}_i\|_2 \leq \sqrt{2\log(\frac{2md}{\beta})}\|\mathbf{a}_i\|_2, \textit{ for all } i \in [d]. \quad (8)$$

Before proceeding, we first give the following Lemma together with its proof.

Lemma 2. *Given data matrix $\mathbf{X} \in \mathbb{R}^{m\times d}$, and the scaled sampling matrix $\mathbf{S} \in \mathbb{R}^{q\times m}$ in SRHT. Then, we have*

$$\mathbb{E}[\mathbf{X}^T\mathbf{S}^T\mathbf{S}\mathbf{X}] = \mathbf{X}^T\mathbf{X}. \quad (9)$$

Proof of Lemma 2. It is related to sampling without replacement, and we defer the proof in the appendix. □

We are now ready to prove our main results: Theorem 1 and Theorem 2.

Proof of Theorem 1. To clarify, we let $q = \ell/2$, define p as the times that steps 6 to 11 in Algorithm 3 have been executed, and assume $p = \frac{n}{m}$ without loss of generality for the input $\mathbf{A} = [\mathbf{A}_1; \mathbf{A}_2; \cdots; \mathbf{A}_p] \in \mathbb{R}^{n\times d}$ with $\{\mathbf{A}_t \in \mathbb{R}^{m\times d}\}_{t=1}^p$. By the triangle inequality, we have

$$\begin{aligned}\|\mathbf{A}^T\mathbf{A} - \mathbf{B}^T\mathbf{B}\|_2 &\leq \|\mathbf{A}^T\mathbf{A} - \mathbf{C}^T\mathbf{C}\|_2 \\ &+ \|\mathbf{C}^T\mathbf{C} - \mathbf{B}^T\mathbf{B}\|_2, \quad (10)\end{aligned}$$

where $\mathbf{C} = [\mathbf{C}_1; \mathbf{C}_2; \cdots; \mathbf{C}_p] \in \mathbb{R}^{pq\times d}$ is from compressing each $\mathbf{A}_t$ by $\mathbf{\Phi}_t = \mathbf{S}_t\mathbf{H}\mathbf{D}_t$ in Algorithm 3 such that $\mathbf{C}_t = \mathbf{\Phi}_t\mathbf{A}_t$. Since $\mathbf{B}$ results from running standard FD on $\mathbf{C}$, then with probability at least $1 - p\beta$, we have

$$\|\mathbf{C}^T\mathbf{C} - \mathbf{B}^T\mathbf{B}\|_2 \leq \frac{2}{\ell}\|\mathbf{C}\|_F^2 \quad (11)$$

$$\leq \frac{4}{\ell}\log(\frac{2md}{\beta})\|\mathbf{A}\|_F^2, \quad (12)$$

where Eq. (11) directly follows from the error bound of FD [25], and Eq. (12) holds by combing $\|\mathbf{C}\|_F^2 =$

$\sum_{t=1}^{p}\|\mathbf{C}_t\|_F^2=\sum_{t=1}^{p}\sum_{i=1}^{d}\|\mathbf{c}_{t,i}\|_2^2$ with Lemma 1 and the union bound.

We next bound $\|\mathbf{A}^T\mathbf{A}-\mathbf{C}^T\mathbf{C}\|_2$. Define $\mathbf{X}_t=\mathbf{H}\mathbf{D}_t\mathbf{A}_t$, then it follows that

$$\|\mathbf{A}^T\mathbf{A}-\mathbf{C}^T\mathbf{C}\|_2=\|\sum_{t=1}^{p}(\mathbf{A}_t^T\mathbf{A}_t-\mathbf{C}_t^T\mathbf{C}_t)\|_2 \quad (13)$$

$$=\|\sum_{t=1}^{p}(\mathbf{A}_t^T\mathbf{A}_t-\mathbf{X}_t^T\mathbf{S}_t^T\mathbf{S}_t\mathbf{X}_t)\|_2. \quad (14)$$

Let $\mathbf{Z}_t=\mathbf{A}_t^T\mathbf{A}_t-\mathbf{X}_t^T\mathbf{S}_t^T\mathbf{S}_t\mathbf{X}_t$, then obviously $\{\mathbf{Z}_t\}_{t=1}^p$ are independent random variables. By Lemma 2, we perform the expectation w.r.t. $\mathbf{S}_t$ and $\mathbf{D}_t$ to obtain that

$$\mathbb{E}[\mathbf{X}_t^T\mathbf{S}_t^T\mathbf{S}_t\mathbf{X}_t]=\mathbb{E}_{\mathbf{D}_t}\mathbb{E}_{\mathbf{S}_t}[\mathbf{X}_t^T\mathbf{S}_t^T\mathbf{S}_t\mathbf{X}_t|\mathbf{D}_t] \quad (15)$$

$$=\mathbb{E}_{\mathbf{D}_t}[\mathbf{X}_t^T\mathbf{X}_t]=\mathbb{E}_{\mathbf{D}_t}[\mathbf{A}_t^T\mathbf{D}_t^T\mathbf{H}^T\mathbf{H}\mathbf{D}_t\mathbf{A}_t]=\mathbf{A}_t^T\mathbf{A}_t,$$

where the second equality follows from Lemma 2 with $\mathbf{D}_t$ fixed, and the last equality holds because $\mathbf{H}$ and $\mathbf{D}_t$ are the unitary matrices. Thus, $\{\mathbf{Z}_t\}_{t=1}^p$ satisfy the setting of the Matrix Bernstein inequality in Theorem 3.

Hence, due to that $\|\mathbf{Z}_t\|_2\le\Delta_t\|\mathbf{A}_t^T\mathbf{A}_t\|_2=\Delta_t\|\mathbf{A}_t\|_2^2$ resulted from Theorem 4, we first achieve

$$R=\max_{t\in[p]}\Delta_t\|\mathbf{A}_t\|_2^2 \quad (16)$$

with probability at least $1-(p\delta+\sum_{t=1}^{p}\frac{m}{e^{k_t}})$ after applying union bound, where $\Delta_t=\Theta(\sqrt{\frac{k_t\log(2k_t/\delta)}{q}})$ and $\text{rank}(\mathbf{A}_t)\le k_t\le\min(m,d)$.

Regarding σ^2, due to the symmetry of each matrix $\mathbf{Z}_t$, $\sigma^2=\|\sum_{t=1}^{p}\mathbb{E}[(\mathbf{Z}_t)^2]\|_2$ holds. Next, we have

$$\mathbf{0}^{d\times d}\preceq\mathbb{E}[(\mathbf{Z}_t)^2] \quad (17)$$

$$=\mathbb{E}[(\mathbf{X}_t^T\mathbf{S}_t^T\mathbf{S}_t\mathbf{X}_t)^2]-(\mathbf{A}_t^T\mathbf{A}_t)^2 \quad (18)$$

$$\preceq\mathbb{E}[\|\mathbf{S}_t\mathbf{X}_t\|_2^2\mathbf{X}_t^T\mathbf{S}_t^T\mathbf{S}_t\mathbf{X}_t]-(\mathbf{A}_t^T\mathbf{A}_t)^2 \quad (19)$$

$$\preceq\mathbb{E}[(1+\Delta_t)\|\mathbf{A}_t\|_2^2\mathbf{X}_t^T\mathbf{S}_t^T\mathbf{S}_t\mathbf{X}_t]-(\mathbf{A}_t^T\mathbf{A}_t)^2 \quad (20)$$

$$=(1+\Delta_t)\|\mathbf{A}_t\|_2^2\mathbf{A}_t^T\mathbf{A}_t-(\mathbf{A}_t^T\mathbf{A}_t)^2 \quad (21)$$

with probability at least $1-(\delta+\frac{m}{e^{k_t}})$, where Eqs. (18)(21) hold because $\mathbb{E}(\mathbf{X}_t^T\mathbf{S}_t^T\mathbf{S}_t\mathbf{X}_t)=\mathbf{A}_t^T\mathbf{A}_t$, Eq. (20) follows from Theorem 4, and Eq. (19) holds because of

$$\mathbf{0}^{d\times d}\preceq(\mathbf{X}_t^T\mathbf{S}_t^T\mathbf{S}_t\mathbf{X}_t)^2\preceq\|\mathbf{S}_t\mathbf{X}_t\|_2^2\mathbf{X}_t^T\mathbf{S}_t^T\mathbf{S}_t\mathbf{X}_t,$$

which results from that for any $\mathbf{y}\in\mathbb{R}^d$,

$$\mathbf{y}^T(\mathbf{X}_t^T\mathbf{S}_t^T\mathbf{S}_t\mathbf{X}_t)^2\mathbf{y}=\|\mathbf{y}^T\mathbf{X}_t^T\mathbf{S}_t^T\mathbf{S}_t\mathbf{X}_t\|_2^2$$
$$\le\|\mathbf{y}^T\mathbf{X}_t^T\mathbf{S}_t^T\|_2^2\|\mathbf{S}_t\mathbf{X}_t\|_2^2=\|\mathbf{S}_t\mathbf{X}_t\|_2^2\mathbf{y}^T\mathbf{X}_t^T\mathbf{S}_t^T\mathbf{S}_t\mathbf{X}_t\mathbf{y}.$$

Then, we have

$$\|\sum_{t=1}^{p}\mathbb{E}[(\mathbf{Z}_t)^2]\|_2\le\sum_{t=1}^{p}\|\mathbb{E}[(\mathbf{Z}_t)^2]\|_2 \quad (22)$$

$$\le\sum_{t=1}^{p}\Big\|(1+\Delta_t)\|\mathbf{A}_t\|_2^2\mathbf{A}_t^T\mathbf{A}_t-(\mathbf{A}_t^T\mathbf{A}_t)^2\Big\|_2 \quad (23)$$

$$=\sum_{t=1}^{p}\Big\|(1+\Delta_t)\|\mathbf{A}_t\|_2^2\mathbf{U}_t\mathbf{\Sigma}_t^2\mathbf{U}_t-\mathbf{U}_t\mathbf{\Sigma}_t^4\mathbf{U}_t\Big\|_2 \quad (24)$$

$$=\sum_{t=1}^{p}\Big\|(1+\Delta_t)\|\mathbf{A}_t\|_2^2\mathbf{\Sigma}_t^2-\mathbf{\Sigma}_t^4\Big\|_2 \quad (25)$$

$$=\sum_{t=1}^{p}\max_{j\in[d]}|(1+\Delta_t)\sigma_{t1}^2\sigma_{tj}^2-\sigma_{tj}^4| \quad (26)$$

$$\le\sum_{t=1}^{p}(1+\Delta_t)\sigma_{t1}^4=\sum_{t=1}^{p}(1+\Delta_t)\|\mathbf{A}_t\|_2^4 \quad (27)$$

$$\le\max_{t\in[p]}p(1+\Delta_t)\|\mathbf{A}_t\|_2^4. \quad (28)$$

where Eq. (23) establishes due to Eq. (21), and $\mathbf{U}_t$ in Eq. (24) is from the SVD of $\mathbf{A}_t$ with $\mathbf{A}_t=\mathbf{U}_t\mathbf{\Sigma}_t\mathbf{V}_t^T$ and the eigenvalues $\sigma_{tj}\triangleq\sigma_{t,jj}$ listed in the descending order in $\mathbf{\Sigma}_t$. By Theorem 3, we have

$$\mathbb{P}(\|\sum_{t=1}^{p}\mathbf{Z}_t\|_2\ge\epsilon)\le 2d\exp(\frac{-\epsilon^2/2}{\sigma^2+R\epsilon/3}). \quad (29)$$

Denote the RHS of Eq. (29) by δ, we then obtain that

$$\epsilon=\log(\frac{2d}{\delta})\Big(\frac{R}{3}+\sqrt{(\frac{R}{3})^2+\frac{2\sigma^2}{\log(2d/\delta)}}\Big) \quad (30)$$

$$\le\log(\frac{2d}{\delta})\frac{2R}{3}+\sqrt{2\sigma^2\log(\frac{2d}{\delta})} \quad (31)$$

$$\le\max_{t\in[p]}\widetilde{O}\Big(\Delta_t\|\mathbf{A}_t\|_2^2\Big)+\max_{t\in[p]}\widetilde{O}\Big(\sqrt{p(1+\Delta_t)}\|\mathbf{A}_t\|_2^2\Big) \quad (32)$$

$$\le\max_{t\in[p]}\widetilde{O}\Big((\sqrt{\frac{k}{\ell p^2}}+\sqrt{\frac{1+\sqrt{k/\ell}}{p}})\frac{\|\mathbf{A}_t\|_2^2}{\|\mathbf{A}_t\|_F^2}\Big)\|\mathbf{A}\|_F^2 \quad (33)$$

$$\le\widetilde{O}\Big((\sqrt{\frac{k}{\ell p^2}}+\sqrt{\frac{1+\sqrt{k/\ell}}{p}})\Big)\|\mathbf{A}\|_F^2. \quad (34)$$

To derive Eq. (33) from Eq. (32), we first substitute $\Delta_t=\Theta(\sqrt{\frac{k_t\log(2k_t/\delta)}{q}})$ into Eq. (32) and set $k=k_t=\min(m,d)$, which allows Eq. (32) to become the maximum of the sum of two functions. Then, we leverage the definition $q=\ell/2$ and apply a common practical assumption of that $p\lambda_1\le\|\mathbf{A}\|_F^2=\sum_{t=1}^{p}\|\mathbf{A}_t\|_F^2\le p\lambda_2$ with each $\|\mathbf{A}_t\|_F^2$ bounded between λ_1 and λ_2 that are very close to each other [2, 36].

Combing Eq. (34) with Eqs. (10)(12) by the union bound achieves the desired result with probability at least $1-p\beta-(2p+1)\delta-2p\frac{m}{e^k}$.

The computational analysis is straightforward based on Sections 2.2 and 3.2. □

Proof of Theorem 2. This proof directly follows from that in Section 2.1 and OSH [22] since FROSH differs OSH merely in the sketching techniques. Then, leveraging Theorem 1 immediately yields the desired results.

□

4 EXPERIMENTS

In this section, we conduct three sets of experiments to empirically verify the properties and demonstrate the superiority of our proposed FROSH. The experiments are conducted in MATLAB R2015a and run on a standard workstation with Intel CPU@2.90GHz and 128GB RAM. The MATLAB is set in single thread mode to fairly test time. All results are averaged over 10 independent runs.

4.1 NUMERICAL STUDIES ON SKETCHING

First of all, it is necessary to verify the sketching properties of FD and FFD, which dominate the performance of OSH and FROSH, respectively. We aim to demonstrate that FFD can achieve a comparable sketching accuracy with a faster running speed in comparison with FD.

To make an insightful comparison, we run both algorithms on the synthetic data $\mathbf{A} \in \mathbb{R}^{n\times d}$, which is generated as [25] to verify the sketching methods. Specifically, $\mathbf{A} \in \mathbb{R}^{n\times d}$ is formally defined as $\mathbf{A} = \mathbf{GFU} + \mathbf{W}/\gamma$, where $\mathbf{G} \in \mathbb{R}^{n\times k}$ is the signal coefficient with $g_{ij} \sim \mathcal{N}(0,1)$, the square diagonal matrix $\mathbf{F} \in \mathbb{R}^{k\times k}$ contains the diagonal entries $f_{ii} = 1-(i-1)/k$ that gives linearly diminishing signal singular values, $\mathbf{U} \in \mathbb{R}^{k\times d}$ defines the signal row space with $\mathbf{UU}^T = \mathbf{I}_k$ ($k \ll d$), and $\mathbf{W} \in \mathbb{R}^{n\times d}$ consists of noise $w_{ij} \sim \mathcal{N}(0,1)$.The parameter γ determines if the noise can dominate the signal. Following [25], we set $k = 10$ and $\gamma = 10$.

In the experiments, we vary the sketching size ℓ along $(16, 32, 64, 100, 128, 200, 256)$. We also vary the data size n and dimension d of $\mathbf{A} \in \mathbb{R}^{n\times d}$, and the parameter m in FFD. Such variables can impact the sketching performance in different degrees.

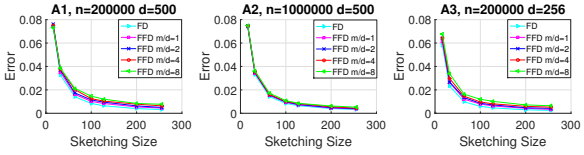

Figure 2: Accuracy comparisons of sketching methods.

We plot the relative error $\|\mathbf{A}^T\mathbf{A} - \mathbf{B}^T\mathbf{B}\|_2/\|\mathbf{A}\|_F^2$ versus the sketching size ℓ in Figure 2 and compare the running time given in Figure 3. It can be seen that, for data $\mathbf{A}_1$, FFD displays comparable accuracy while enjoying a much lower time cost compared with FD. Besides, when m increases, the sketching accuracy of FFD slightly decreases, but its running speed increases especially for a larger ℓ. Data $\mathbf{A}_2$ differs $\mathbf{A}_1$ only on n, and we observe that a larger n improves the sketching accuracy of FFD. Finally, we decrease the dimension d in $\mathbf{A}_3$, and we find that the improvement of running speed for FFD becomes smaller than that in $\mathbf{A}_1$. All these observations are consistent with the proved results in Theorem 1.

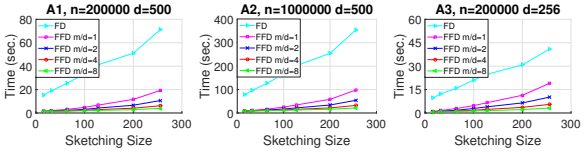

Figure 3: Time comparisons of sketching methods.

All the results indicate that our proposed FFD has the potential to benefit the sketching-based online hashing.

4.2 COMPARISONS WITH LSH AND OSH

In this part, we compare against online unsupervised hashing methods including LSH [8] and OSH. We employ four datasets: CIFAR-10 [21], MNIST [7], GIST-1M [16], and FLICKR-25600 [49]. For CIFAR-10, it consists of 60,000 images in 10 classes with 6000 images per class. We extract 512-dimensional GIST descriptor to represent each image. MNIST contains 70,000 images with 784-dimensional features. GIST-1M consists of one million 960-dimensional GIST descriptors. The FLICKR-25600 dataset contains 100,000 images sampled from a noisy Internet image collection, and each image is represented by a 25,600-dimensional vector normalized to be of unit norm. Following the common setting, we take the top 2% nearest neighbors in Euclidean space as the ground truth for all datasets.

In both OSH and FROSH, we set the sketching size $\ell = 2r$, where r is the code length assigned from $\{32, 64, 128\}$. We train both algorithms in a streaming fashion by evenly dividing each dataset into 10 parts, and evaluate the mean average precision (MAP) score after each round. We also empirically set $m = 4d$ in FROSH during the training. For LSH, it does not require training, and we simply report its MAP at the final round [22].

Figure 4 shows the MAP scores at different rounds on four datasets with 32, 64 and 128 bits codes. On all datasets, it is apparent that our proposed FROSH performs as accurately as OSH and outperforms LSH with

a large margin. In addition, both OSH and FROSH stably improve the MAP scores when receiving more data, which demonstrates that a successful adaption to the data variations has been achieved.

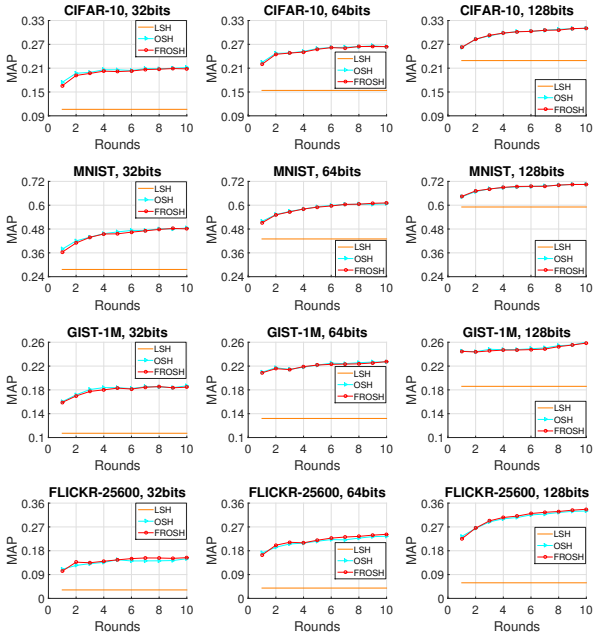

Figure 4: MAP at each round with 32, 64, and 128 bits.

Table 1 provides the accumulated training time of OSH and FROSH with 32, 64, 128 bits codes, respectively. FROSH consistently has the lower training time for each comparison, which achieves about $10 \sim 20$ times speed-up than OSH. Thus, our method is highly efficient.

Table 1: Accumulated training time of OSH and FROSH after all rounds (in sec.).

Dataset	**Method**	32bits	64bits	128bits
CIFAR-10	OSH	7.78	11.88	22.09
	FROSH	**0.63**	**0.94**	**2.11**
MNIST	OSH	13.25	18.93	30.75
	FROSH	**1.17**	**1.49**	**2.56**
GIST-1M	OSH	228	331	520
	FROSH	**21**	**27**	**45**
FLICKR-25600	OSH	679	1283	2570
	FROSH	**72**	**92**	**134**

4.3 COMPARISONS WITH BATCH SOLUTIONS

We also compare OSH and our FROSH against two leading batch methods SGH [17] and OCH [27]. Based on the literature [17, 27, 29], we know that SGH maintains a superior tradeoff between the learning accuracy and the scalability, and OCH is the state-of-the-art regarding the accuracy performance compared with the other unsupervised hashing methods such as SpH [44], AGH [30], IsoH [20], DGH [29], and OEH [28].

In Figure 5, we clearly observe that FROSH achieves comparable or even better accuracy in comparison with the two leading batch solutions, which suggests that not only does the online sketching maintain sufficient information necessary for the hash function training but also it owns a good learning ability. In conclusion, considering the properties that FROSH can adapt the hash functions to the new coming data and enjoy superior training efficiency (single pass, lowest costs of space and time), our proposed method is more appropriate for hashing learning on the big streaming unlabeled data. Note that we defer the time comparisons in the appendix.

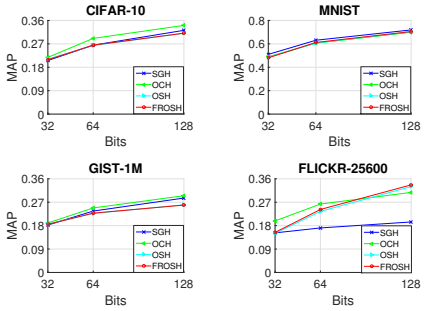

Figure 5: MAP comparisons at different code lengths.

5 CONCLUSION

In this paper, we propose an effective hashing method for streaming unlabeled data. Its basic idea is to reduce the sketching time that is the dominated cost in the OSH method. The analysis shows better time efficiency with accuracy guaranteed. The simulations on both synthetic and real-world datasets support our theoretical findings and demonstrate the practicability of our method.

The sketching designed in this work also provides an insightful view for accelerating general streaming matrix multiplications, which then may be helpful for deriving an online version of OCH. Such extensions can be explored in the future work.

Acknowledgments

We thank the reviewers for their insightful comments to improve the paper quality. We also truly thank Cong Leng for the fruitful discussions and suggestions. This work was fully supported by the National Natural Science Foundation of China (Project No. 61332010), the Research Grants Council of the Hong Kong Special Administrative Region, China ((No. CUHK 14208815 and No. CUHK 14234416 of the General Research Fund), and 2015 Microsoft Research Asia Collaborative Research Program (Project No. FY16-RES-THEME-005).

References

[1] N. Ailon and E. Liberty. Fast dimension reduction using rademacher series on dual bch codes. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, 2008.

[2] F. Anaraki and S. Becker. Preconditioned data sparsification for big data with applications to pca and k-means. *arXiv preprint arXiv:1511.00152*, 2015.

[3] A. Andoni, P. Indyk, T. Laarhoven, I. Razenshteyn, and L. Schmidt. Practical and optimal lsh for angular distance. In *Advances in Neural Information Processing Systems*, 2015.

[4] H. Avron, H. Nguyen, and D. Woodruff. Subspace embeddings for the polynomial kernel. In *Advances in Neural Information Processing Systems*, 2014.

[5] F. Cakir and S. Sclaroff. Adaptive hashing for fast similarity search. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1044–1052, 2015.

[6] F. Cakir and S. Sclaroff. Online supervised hashing. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 2606–2610. IEEE, 2015.

[7] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2011.

[8] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388. ACM, 2002.

[9] A. Choromanska, K. Choromanski, M. Bojarski, T. Jebara, S. Kumar, and Y. LeCun. Binary embeddings with structured hashed projections. In *ICML*, 2016.

[10] P. Dhillon, Y. Lu, D. P. Foster, and L. Ungar. New subsampling algorithms for fast least squares regression. In *Advances in Neural Information Processing Systems*, pages 360–368, 2013.

[11] M. Ghashami, E. Liberty, and J. M. Phillips. Efficient frequent directions algorithm for sparse matrices. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 845–854. ACM, 2016.

[12] A. Gionis, P. Indyk, R. Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB*, 1999.

[13] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

[14] L.-K. Huang, Q. Yang, and W.-S. Zheng. Online hashing. In *IJCAI*. Citeseer, 2013.

[15] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, 1998.

[16] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 2011.

[17] Q.-Y. Jiang and W.-J. Li. Scalable graph hashing with feature transformation. In *IJCAI*, pages 2248–2254, 2015.

[18] I. Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

[19] W.-C. Kang, W.-J. Li, and Z.-H. Zhou. Column sampling based discrete supervised hashing. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[20] W. Kong and W.-J. Li. Isotropic hashing. In *Advances in Neural Information Processing Systems*, 2012.

[21] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[22] C. Leng, J. Wu, J. Cheng, X. Bai, and H. Lu. Online sketching hashing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2503–2511, 2015.

[23] W.-J. Li, S. Wang, and W.-C. Kang. Feature learning based deep supervised hashing with pairwise labels. In *IJCAI*, 2016.

[24] Y. Li, R. Wang, H. Liu, H. Jiang, S. Shan, and X. Chen. Two birds, one stone: Jointly learning binary code for large-scale face image retrieval and attributes prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[25] E. Liberty. Simple and deterministic matrix sketching. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2013.

[26] G. Lin, C. Shen, Q. Shi, A. van den Hengel, and D. Suter. Fast supervised hashing with decision trees for high-dimensional data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[27] H. Liu, R. Ji, Y. Wu, and F. Huang. Ordinal constrained binary code learning for nearest neighbor search. In *AAAI*, 2017.

[28] H. Liu, R. Ji, Y. Wu, and W. Liu. Towards optimal binary code learning via ordinal embedding. In *AAAI*, 2016.

[29] W. Liu, C. Mu, S. Kumar, and S.-F. Chang. Discrete graph hashing. In *Advances in Neural Information Processing Systems*, pages 3419–3427, 2014.

[30] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. In *Proceedings of the 28th international conference on machine learning*, 2011.

[31] Y. Lu, P. Dhillon, D. P. Foster, and L. Ungar. Faster ridge regression via the subsampled randomized hadamard transform. In *Advances in neural information processing systems*, 2013.

[32] H. Luo, A. Agarwal, N. Cesa-Bianchi, and J. Langford. Efficient second order online learning by sketching. In *Advances in Neural Information Processing Systems*, 2016.

[33] L. Mukherjee, S. N. Ravi, V. K. Ithapu, T. Holmes, and V. Singh. An nmf perspective on binary hashing. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[34] C. Musco and C. Musco. Randomized block krylov methods for stronger and faster approximate singular value decomposition. In *Advances in Neural Information Processing Systems*, pages 1396–1404, 2015.

[35] B. Neyshabur, N. Srebro, R. R. Salakhutdinov, Y. Makarychev, and P. Yadollahpour. The power of asymmetry in binary hashing. In *Advances in Neural Information Processing Systems*, 2013.

[36] J. Pennington, F. Yu, and S. Kumar. Spherical random features for polynomial kernels. In *Advances in Neural Information Processing Systems*, pages 1846–1854, 2015.

[37] F. Shen, W. Liu, S. Zhang, Y. Yang, and H. T. Shen. Learning binary codes for maximum inner product search. In *International Conference on Computer Vision*. IEEE, 2015.

[38] F. Shen, C. Shen, W. Liu, and H. Tao Shen. Supervised discrete hashing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[39] A. Shrivastava and P. Li. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In *Advances in Neural Information Processing Systems*, pages 2321–2329, 2014.

[40] J. Song, L. Gao, Y. Yan, D. Zhang, and N. Sebe. Supervised hashing with pseudo labels for scalable multimedia retrieval. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 827–830. ACM, 2015.

[41] J. A. Tropp. An introduction to matrix concentration inequalities. *Foundations and Trends in Machine Learning*, 2015.

[42] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for scalable image retrieval. In *Computer Vision and Pattern Recognition, 2010 IEEE Conference on*, pages 3424–3431. IEEE, 2010.

[43] J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen. A survey on learning to hash. *arXiv preprint arXiv:1606.00185*, 2016.

[44] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Advances in neural information processing systems*, 2009.

[45] D. P. Woodruff et al. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 2014.

[46] C. Woolam, M. M. Masud, and L. Khan. Lacking labels in the stream: classifying evolving stream data with few labels. In *International Symposium on Methodologies for Intelligent Systems*, pages 552–562. Springer, 2009.

[47] S. Wu, S. Bhojanapalli, S. Sanghavi, and A. G. Dimakis. Single pass pca of matrix products. In *Advances in Neural Information Processing Systems*, 2016.

[48] L. Xie, L. Zhu, and G. Chen. Unsupervised multi-graph cross-modal hashing for large-scale multimedia retrieval. *Multimedia Tools and Applications*, pages 1–20, 2016.

[49] F. X. Yu, S. Kumar, Y. Gong, and S.-F. Chang. Circulant binary embedding. In *ICML*, 2014.