# Frequent Direction Algorithms for Approximate Matrix Multiplication with Applications in CCA

**Qiaomin Ye, Luo Luo, Zhihua Zhang**
Department of Computer Science and Engineering
Shanghai Jiao Tong University
{lilianye,ricky,zhihua}@sjtu.edu.cn

## Abstract

Approximate matrix multiplication (AMM) becomes increasingly popular because it makes matrix computation suitable for large-scale datasets. Most previous AMM methods are based on the idea of random selection or random projection. In this paper, we propose a deterministic algorithm FD-AMM for computing an approximation to the product of two given matrices. Moreover, the algorithm works in a streaming manner. In particular, our approach is inspired by a recently proposed matrix sketching algorithm called Frequent Directions (FD). FD-AMM has stronger error bound than both random selection and random projection algorithms with respect to the same space complexity. Our approach also leads to an algorithm for computing the Canonical Correlation Analysis (CCA) of two matrices exactly in a streaming way, which takes less space than the classical method. Experimental results validate the effectiveness of our method.

## 1 Introduction

Modern large data sets are often expressed as large matrices. For example, in the bag-of-words model, each row of the matrix corresponds to one document; in image analysis, feature values are represented by a matrix whose rows correspond to images. Therefore, many computational problems on such data are reduced to standard matrix operations including matrix multiplication, $\ell_2$-regression, and low-rank matrix approximation.

Some data are huge in size and often generated sequentially, e.g., computer network traffic and sensor data. Thus, it is important to handle them in a streaming fashion. A streaming algorithm only makes a single pass over the data and uses a small amount of memory. The limited space constraint is critical when the full data set cannot fit in memory or even disk. Typically, there is a trade-off between the amount of space required and the performance of the algorithm.

Approximate matrix multiplication gains significant increases in popularity as the data sets becoming larger and larger. It is an important approach to fast matrix multiplication. Moreover, it can be used in information retrieval [Eriksson-Bique *et al.*, 2011], image processing [Madrid *et al.*, 2012], large scale $k$-means clustering [Cohen *et al.*, 2015], and approximate leverage scores [Drineas *et al.*, 2012]. The best approximation to the product can be achieved by truncated SVD under any unitarily invariant norms. Although SVD methods are polynomial, they are computation intense when performed exactly.

There are two main approaches for approximate matrix multiplication. The first approach is referred as random selection since it randomly picks rows, columns, or individual entries of the matrices. There are two kinds of sampling methods: uniform sampling and non-uniform sampling. Most studies focus on the latter which obtains more generality. One important non-uniform sampling method is called importance sampling which works with probability proportional to the row's squared Euclidean length [Drineas and Kannan, 2001; Drineas *et al.*, 2006; Eriksson-Bique *et al.*, 2011]. Madrid *et al.* (2012) compared different sampling strategies on ill-conditioned matrices empirically. Pagh (2013) focused on sparse matrices.

The second approach is referred as random projection or "sketching." Many studies are based on the Johnson-Lindenstrauss (JL) Lemma [Dasgupta and Gupta, 2003]. For example, Sarlos (2006) achieved a Frobenius norm error guarantee using the JL transform. Clarkson and Woodruff (2009) further refined the results, and showed that their space usage is nearly optimal in a streaming setting. Kane and Nelson (2014) gained more speed up by introducing a sparse JL transform. Clarkson and Woodruff (2013) came up with a very sparse subspace embedding matrix with modifications by Nelson and Nguyên (2013). Their work reduced time complexity to $\mathcal{O}(nnz(\mathbf{A}) + nnz(\mathbf{B}))$. However, random projection may fail with relatively high probability when sketch size is small.

Besides these two main kinds, there are also some other methods. For example, Cohen *et al.* (2015) proved that using subspace embedding achieves optimal approximate matrix multiplication in terms of stable rank. Bhojanapalli *et al.* (2015) proposed a new method which utilizes sampling and alternating minimization to directly compute a low-rank approximation of matrix product. However, those methods can not be implemented in a streaming way.

In this paper we investigate the streaming memory-limited matrix multiplication problem. Our work is inspired by a recently proposed matrix sketching algorithm called the Frequent-Directions (FD) [Liberty, 2013] which is easy to implement and deterministic. We propose a new method for approximate matrix multiplication (AMM) that we call FD-AMM. To our best knowledge, our method is the first deterministic approach to approximate matrix multiplication. An error bound in terms of spectral norm is given, and experimental results demonstrate its effectiveness and efficiency.

We also apply our approach to computing Canonical Correlation Analysis (CCA) of two matrices in a streaming way. CCA is a fundamental statistical tool to capture the relationship between two multidimensional variables [Hotelling, 1936]. It is analogous to Principal Component Analysis (PCA). But instead of analyzing a single matrix, it aims to analyze the relation between a pair of datasets. It has found its usage in a wide range of applications, e.g., dimensionality reduction [McWilliams *et al.*, 2013], clustering [Chaudhuri *et al.*, 2009], and speech recognition [Wang *et al.*, 2015].

The remainder of the paper is organized as follows. We first define the notation used in this paper and introduce the background of CCA and FD algorithm. Then we describe our proposed FD-AMM method and provide theoretical analysis in section 3. Finally, we provide empirical comparisons with three most used AMM algorithms to demonstrate the superiority of FD-AMM.

## 2 Notation and Preliminaries

In this section we give the notation and preliminaries which will be used in this paper. We use $i : j$ to denote the integer set $\{i, \ldots, j\}$, and define $[n] = 1 : n$. Let $\mathbf{I}_n$ be the $n \times n$ identity matrix, and $\mathbf{0}$ be the matrix of zeros with appropriate size. We are given an $n \times m$ matrix $\mathbf{A} = [\mathbf{a}_1; \mathbf{a}_2; \ldots; \mathbf{a}_n]$ where $\mathbf{a}_i \in \mathbb{R}^{1\times m}$ is the $i$-th row of $\mathbf{A}$. Typically the matrix is assumed to be thin so $n \geq m$. When matrices $\mathbf{A}$ and $\mathbf{B}$ have the same number of rows, we use $[\mathbf{A}\ \mathbf{B}]$ to denote the matrix obtained by concatenating the columns of $\mathbf{B}$ next to the columns of $\mathbf{A}$. Two vectors $\mathbf{x}$ and $\mathbf{y}$ are orthogonal is denoted by $\mathbf{x} \perp \mathbf{y}$.

The squared Frobenius norm of the matrix $\mathbf{A}$ is defined as $\|\mathbf{A}\|_F^2 = \sum_{i=1}^n \|\mathbf{a}_i\|^2$ where $\|\mathbf{a}_i\|$ is the Euclidean norm of row $\mathbf{a}_i$. The spectral norm is $\|\mathbf{A}\|_2 = \max_{\mathbf{x}:\|\mathbf{x}\|=1} \|\mathbf{A}\mathbf{x}\|$.

The singular value decomposition of $\mathbf{A} \in \mathbb{R}^{m\times n}$, written svd($\mathbf{A}$), produces three matrices $\{\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}\}$ so that $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ where $\mathbf{U} \in \mathbb{R}^{n\times n}$ and $\mathbf{V} \in \mathbb{R}^{m\times n}$ satisfy $\mathbf{U}^T\mathbf{U} = \mathbf{U}\mathbf{U}^T = \mathbf{V}^T\mathbf{V} = \mathbf{I}_n$, and $\mathbf{\Sigma}$ is an $n \times n$ diagonal matrix with singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$. Let $r$ be the rank of $\mathbf{A}$. Assume $\widetilde{\mathbf{U}}$ and $\widetilde{\mathbf{V}}$ consist of the first $r$ columns of $\mathbf{U}$ and $\mathbf{V}$ respectively, and $\widetilde{\mathbf{\Sigma}} = \mathbf{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$, then $\widetilde{\mathbf{U}}\widetilde{\mathbf{\Sigma}}\widetilde{\mathbf{V}} = \mathbf{A}$ is the condensed SVD of $\mathbf{A}$. The SVD of an $n \times m$ matrix costs time $\mathcal{O}(mn \min\{n, m\})$.

Then we formally introduce the background of CCA. There are several equivalent ways to define the canonical correlations of a pair of matrices [Golub and Zha, 1995]. Here we use the linear algebraic formulation described in Definition 2.1, which captures the very essence of the procedure, pursuing the directions of maximal correlations between two data matrices.

**Definition 2.1.** *[Avron* et al.*, 2014] Let $\mathbf{A} \in \mathbb{R}^{n\times m_1}$ and $\mathbf{B} \in \mathbb{R}^{n\times m_2}$, and assume that $p = rank(\mathbf{A}) \geq rank(\mathbf{B}) = q$. The canonical correlations $\pi_1(\mathbf{A}, \mathbf{B}) \geq \pi_2(\mathbf{A}, \mathbf{B}) \geq \cdots \geq \pi_q(\mathbf{A}, \mathbf{B})$ of the matrix pencil ($\mathbf{A}$,$\mathbf{B}$) are defined recursively by the following formula:*

$$\pi_i(\mathbf{A}, \mathbf{B}) = \max_{\mathbf{x}\in\mathcal{A}_i, \mathbf{y}\in\mathcal{B}_i} \pi(\mathbf{A}\mathbf{x}, \mathbf{B}\mathbf{y}) = \pi(\mathbf{A}\mathbf{x}_i, \mathbf{B}\mathbf{y}_i),$$

*where*

- $i = 1, \ldots, q,$
- $\pi(\mathbf{u}, \mathbf{v}) = |\mathbf{u}^T\mathbf{v}|/(\|\mathbf{u}\|\|\mathbf{v}\|),$
- $\mathcal{A}_i = \{\mathbf{x} : \mathbf{A}\mathbf{x} \neq \mathbf{0}, \mathbf{A}\mathbf{x} \perp \{\mathbf{A}\mathbf{x}_1, \ldots, \mathbf{A}\mathbf{x}_{i-1}\}\},$
- $\mathcal{B}_i = \{\mathbf{y} : \mathbf{B}\mathbf{y} \neq \mathbf{0}, \mathbf{B}\mathbf{y} \perp \{\mathbf{B}\mathbf{y}_1, \ldots, \mathbf{B}\mathbf{y}_{i-1}\}\}.$

*The $n$-dimensional unit vectors*

$$\mathbf{A}\mathbf{x}_1/\|\mathbf{A}\mathbf{x}_1\|, \ldots, \mathbf{A}\mathbf{x}_q/\|\mathbf{A}\mathbf{x}_q\|$$

$$\mathbf{B}\mathbf{y}_1/\|\mathbf{B}\mathbf{y}_1\|, \ldots, \mathbf{B}\mathbf{y}_q/\|\mathbf{B}\mathbf{y}_q\|$$

*are called canonical vectors.*

There are quite a few methods to compute the canonical correlations [Golub and Zha, 1995]. For example, Björck and Golub (1973) proposed an algorithm based on Theorem 1, which needs exactly QR decomposition of original matrices $\mathbf{A}$ and $\mathbf{B}$.

Theorem 2 can be proved directly by the definition of CCA [Lu and Foster, 2014]. The approach based on Theorem 2 needs matrix multiplication $\mathbf{A}^T\mathbf{A}$ and $\mathbf{B}^T\mathbf{B}$ with $\mathcal{O}(n(m_1^2 + m_2^2))$ computational complexity and matrix decomposition to compute $\mathbf{S}_\mathbf{a}^{-\frac{1}{2}}$ and $\mathbf{S}_\mathbf{b}^{-\frac{1}{2}}$ with $\mathcal{O}(m_1^3 + m_2^3)$ computational complexity. As a result, both two methods require $\mathcal{O}(n(m_1^2 + m_2^2))$ time and $\mathcal{O}(n(m_1 + m_2))$ space. These classical methods are feasible and accurate when the data matrices are small but they can be slow and unstable for large-scale datasets.

**Theorem 1.** *[Björck and Golub, 1973] Assume that the columns of $\mathbf{Q} \in \mathbb{R}^{n\times p}$ and $\mathbf{W} \in \mathbb{R}^{n\times q}$ form an orthonormal basis for the range of $\mathbf{A}$ and $\mathbf{B}$, respectively. Let $\mathbf{Q}^T\mathbf{W} = \widetilde{\mathbf{U}}\widetilde{\mathbf{\Sigma}}\widetilde{\mathbf{V}}^T$ be its condensed SVD. The diagonal elements of $\widetilde{\mathbf{\Sigma}}$ are the canonical correlations of $(\mathbf{A}, \mathbf{B})$. The canonical vectors are given by the first q columns of $\mathbf{Q}\widetilde{\mathbf{U}}$ for $\mathbf{A}$ and of $\mathbf{W}\widetilde{\mathbf{V}}$ for $\mathbf{B}$.*

**Theorem 2.** *[Lu and Foster, 2014] Let $\mathbf{S}_\mathbf{a} = \mathbf{A}^T\mathbf{A}/n$, $\mathbf{S}_\mathbf{b} = \mathbf{B}^T\mathbf{B}/n$ and $\mathbf{S}_\mathbf{ab} = \mathbf{A}^T\mathbf{B}/n$. Let $\mathbf{S}_\mathbf{a}^{-\frac{1}{2}}\mathbf{S}_\mathbf{ab}\mathbf{S}_\mathbf{b}^{-\frac{1}{2}} = \widetilde{\mathbf{U}}\widetilde{\mathbf{\Sigma}}\widetilde{\mathbf{V}}^T$ be its condensed SVD. Then the diagonal elements of $\widetilde{\mathbf{\Sigma}}$ are the canonical correlations of $(\mathbf{A}, \mathbf{B})$. The canonical vectors are given by the first q columns of $\mathbf{S}_\mathbf{a}^{-\frac{1}{2}}\widetilde{\mathbf{U}}$ for $\mathbf{A}$ and of $\mathbf{S}_\mathbf{b}^{-\frac{1}{2}}\widetilde{\mathbf{V}}$ for $\mathbf{B}$.*

## 3 Methodology

In this section, we first review the Frequent Directions algorithm. Then we present the main contribution of the paper: the FD-AMM algorithm. Specifically, given an $n\times m_1$ matrix $\mathbf{A}$ and an $n \times m_2$ matrix $\mathbf{B}$, FD-AMM computes an approximation $\mathbf{X}$ to the product $\mathbf{A}^T\mathbf{B}$ so that $\mathbf{X} \approx \mathbf{A}^T\mathbf{B}$. Moreover, the algorithm is based on a streaming setting which only requires one pass over the input. Finally, we give a new way to compute the CCA of two matrices also in a streaming fashion which is based on FD-AMM.

We describe the procedure of FD algorithm [Liberty, 2013; Ghashami and Phillips, 2014] in Algorithm 1. FD computes a sketch of a given matrix. Specifically, it keeps an $\ell \times d$ ($\ell \leq n$) sketch matrix $\mathbf{H}$ that is updated every time a new row from the $n \times d$ input matrix $\mathbf{G}$ is added. For any unit vector $\mathbf{x} \in \mathbb{R}^d$, it has

$$\|\mathbf{Gx}\|^2 - \|\mathbf{Hx}\|^2 \leq \|\mathbf{G}\|_F^2/\ell.$$

The algorithm maintains an invariant that the last row of the sketch $\mathbf{H}$ is always all-zero valued. During the execution of the algorithm, rows from $\mathbf{G}$ simply replace the all-zero valued row in $\mathbf{H}$. Then, the last row is nullified by a two-stage process. First, we rotate the sketch matrix by SVD so that its rows are in descending order of significance.Then, we "shrink" all the rows so that at least one of them becomes zero.

**Algorithm 1** Frequent Directions (FD)

```
Input: $\ell, \mathbf{G} \in \mathbb{R}^{n\times d}$
  $\mathbf{H} = \mathbf{0}_{\ell\times d}$
  for $i \in 1, \dots, n$ do
    $\mathbf{h}_\ell = \mathbf{g}_i$
    $[\mathbf{Z}, \mathbf{S}, \mathbf{Y}]$ =svd($\mathbf{H}$)
    $\mathbf{M}^{(i)} = \mathbf{SY}^T$ [only for notation]
    $\delta_i = s_\ell^2$ [the $\ell$th entry of $\mathbf{S}$, squared]
    $\mathbf{S}' = \sqrt{\mathbf{S}^2 - \delta_i \mathbf{I}_\ell}$
    $\mathbf{H}^{(i)} = \mathbf{S}'\mathbf{Y}^T$
  end for
  return $\mathbf{H} = \mathbf{H}^{(n)}$
```

The computational cost of each iteration comes from the SVD of an $\ell \times d$ matrix, which is $\mathcal{O}(d\ell^2)$ when $\ell < d$. The total time complexity of FD algorithm is therefore bounded by $\mathcal{O}(nd\ell^2)$.

## 3.1 FD-AMM

**Algorithm 2** FD-AMM

```
Input: $\ell, \mathbf{A} \in R^{n\times m_1}, \mathbf{B} \in \mathbb{R}^{n\times m_2}$
  $\mathbf{G} = [\mathbf{A}\ \mathbf{B}]$
  $\mathbf{H} = \mathrm{FD}(\ell, \mathbf{G})$
  $\mathbf{C} = \mathbf{H}_{[\ell],1:m_1}$
  $\mathbf{D} = \mathbf{H}_{[\ell],m_1+1:m_1+m_2}$
  return $\mathbf{X} = \mathbf{C}^T\mathbf{D}$
```

We presents the proposed method FD-AMM in Algorithm 2. It is built upon the FD algorithm, but it focuses on the approximate product of two given matrices. We provide the theoretical guarantee of FD-AMM algorithm in Theorem 3. The proof of the theorem is presented in the appendix.

**Theorem 3.** *If $\mathbf{X}$ is the result of applying the FD-AMM algorithm to matrices $\mathbf{A}$, $\mathbf{B}$, and sketch size $\ell$, then*

$$\|\mathbf{A}^T\mathbf{B} - \mathbf{X}\|_2 \leq (\|\mathbf{A}\|_F^2 + \|\mathbf{B}\|_F^2)/\ell.$$

Liberty (2013) observed that the time complexity of FD algorithm can be reduced to $\mathcal{O}(nd\ell)$ at the expense of more space. More specifically, increasing $\ell$ by a constant $c > 1$ and then processing every $\ell(c-1)$ elements in a batch setting (each round results in a $c\ell$ rows matrix $\mathbf{H}$) can reduce the overall time to $\mathcal{O}(\frac{c^2}{c-1}nd\ell)$. This trick can be applied to our algorithm. Let $\ell_{new} = 2\ell_{old}$ then the time complexity becomes $\mathcal{O}(2nm\ell_{new})$. Thus our FD-AMM algorithm can run in $\mathcal{O}(nm\ell)$, where $m = m_1 + m_2$.

Using FD-AMM for approximate matrix multiplication has several advantages. First, it is easy to implement. Second, the error bound given by Theorem 3 is stronger than existing state-of-art algorithms with the same sketch size $\ell$. It is worth pointing out that both sampling and random projection based algorithms' error bounds are proportional to $1/\sqrt{\ell}$. Third, our algorithm is deterministic. Hence it is more stable than other random algorithms.

## 3.2 Streaming CCA

Based on the FD-AMM algorithm, we come up with a new way to calculate the canonical correlations of two matrices. The new method is inspired by a special case of the FD-AMM algorithm. When $\ell \geq 2m$, we have $\delta_i = 0$ for all $i = 1, \dots, n$, which makes the "shrink" step does nothing at all thus can be omitted. As a result we have $\mathbf{H}^{(i)} = \mathbf{M}^{(i)}$ and ${\mathbf{H}^{(i-1)}}^T\mathbf{H}^{(i-1)} = {\mathbf{H}^{(i)}}^T\mathbf{H}^{(i)}$. Therefore, the total error incurred by FD-AMM is still zero. Corollary 4 gives the formal observation. The proof of Corollary 4 is similar to that of Theorem 3.

**Corollary 4.** *In Theorem 3, when $\ell \geq 2m$, we will have $\delta_i = 0$ for all $i = 1, \dots, n$. Thus the total error incurred by the FD-AMM algorithm is zero. More specifically,*

$$\mathbf{A}^T\mathbf{B} = \mathbf{C}^T\mathbf{D}.$$

*With the same reason, we have $\mathbf{A}^T\mathbf{A} = \mathbf{C}^T\mathbf{C}$ and $\mathbf{B}^T\mathbf{B} = \mathbf{D}^T\mathbf{D}$ in this case.*

The streaming nature of our FD-AMM algorithm helps especially when the matrices are too large to fit in memory. To be specific, when given a pair of matrices $(\mathbf{A}, \mathbf{B})$, we transform the pair to a new pair $(\mathbf{C}, \mathbf{D})$ that has much fewer rows, and then compute the canonical correlations of the new pair exactly, e.g., using the Björck and Golub algorithm. We present the detailed procedure in Algorithm 3. Theoretical analysis is given in Theorem 5.

**Theorem 5.** *If the two matrices $\mathbf{A}$ and $\mathbf{B}$ are tall-and-thin, that is, $n \gg m_1$, then Algorithm 3 computes the exact canonical correlations of $(\mathbf{A}, \mathbf{B})$. That is,*

$$\pi_i(\mathbf{A}, \mathbf{B}) = \pi_i(\mathbf{C}, \mathbf{D})$$

*for $i = 1, \dots, q$. And the canonical vectors returned by Algorithm 3 are the same as the results computed by the Björck and Golub algorithm.*

Combining Theorem 2 and Corollary 4 can get Theorem 5 directly. Algorithm 3 has a time complexity of $\mathcal{O}(nm^2)$ and space complexity of $\mathcal{O}(m^2)$, while the Björck and Golub algorithm needs $\mathcal{O}(nm)$ space.

**Algorithm 3** Streaming CCA

```
Input: $\mathbf{A} \in \mathbb{R}^{n \times m_1}$ of rank $p$ and $\mathbf{B} \in \mathbb{R}^{n \times m_2}$ of rank $q$
  $(m_1 \geq m_2, p \geq q)$. $m = m_1 + m_2$, $\ell = 2m$. $\mathbf{G} = [\mathbf{A}\ \mathbf{B}]$,
  $\mathbf{H} = \mathbf{0}_{\ell \times m}$.
  for $i \in 1, \ldots, n$ do
    Insert $\mathbf{g}_i$ into a zero valued row of $\mathbf{H}$
    if $\mathbf{H}$ has no zero valued rows then
      $[\mathbf{Z}, \mathbf{S}, \mathbf{Y}]$ =svd($\mathbf{H}$)
      $\mathbf{H} = \mathbf{S}\mathbf{Y}^T$
    end if
  end for
  return $\mathbf{H} = [\mathbf{C}\ \mathbf{D}]$
Compute and return the canonical correlations and the canonical vectors of $(\mathbf{C}, \mathbf{D})$ (using classical algorithm).
```

## 4 Experiments

We compare FD-AMM with baselines. The first one is the brute force approach. The other three are common algorithms that are used in practice: sampling, random projection, and sparse random projection. All methods receive the rows of an $n \times m_1$ matrix $\mathbf{A}$ and an $n \times m_2$ matrix $\mathbf{B}$ one by one during the experiments. The brute force method outputs the product directly, while other methods first compute smaller matrices $\mathbf{C}$ and $\mathbf{D}$ for $\mathbf{A}$ and $\mathbf{B}$ respectively, then compute $\mathbf{C}^T\mathbf{D}$.

**Brute Force (BF):** The brute force approach produces the exact result of $\mathbf{A}^T\mathbf{B}$. It explicitly computes the matrix $\mathbf{A}^T\mathbf{B} = \sum_i^n \mathbf{a}_i^T \mathbf{b}_i$ by aggregating the outer products of the rows of $\mathbf{A}$ and $\mathbf{B}$. The update time of Brute Forces is $\mathcal{O}(m_1 m_2)$ and its space complexity is $\mathcal{O}(m_1 m_2)$.

**Sampling:** Let $p_i = \|\mathbf{a}_i\|\|\mathbf{b}_i\|/S$ be a probability distribution over $[n]$ where $S = \sum_{i=1}^n \|\mathbf{a}_i\|\|\mathbf{b}_i\|$. We form an $\ell \times m_1$ matrix $\mathbf{C}$ and an $\ell \times m_2$ matrix $\mathbf{D}$ by taking $\ell$ i.i.d. (row indices) samples with the $p_i$. Since the value of $S$ is not known *a priori*, we use $\ell$ independent reservoir samplers to implement the streaming algorithm. Therefore, the update for each row requires $\mathcal{O}(m_1 + m_2)$ time and $\mathcal{O}(\ell(m_1 + m_2))$ space. The reader can refer to [Drineas *et al.*, 2006; Eriksson-Bique *et al.*, 2011] for a theoretical analysis of this algorithm.

**Random projection (RP):** The matrices $\mathbf{C}$ and $\mathbf{D}$ have the form of $\mathbf{RA}$ and $\mathbf{RB}$ respectively, where $\mathbf{R}$ is an $\ell \times n$ matrix and $R_{ij} \in \{-1/\sqrt{\ell}, 1/\sqrt{\ell}\}$ uniformly. This is easily computed in a streaming fashion, while requiring at most $\mathcal{O}(\ell(m_1 + m_2))$ space and $\mathcal{O}(\ell(m_1 + m_2))$ time per row. For proofs of effectiveness and usage see [Clarkson and Woodruff, 2009; Cohen *et al.*, 2015].

**Sparse Random Projection (SRP):** Clarkson and Woodruff (2013) proposed a very sparse projection matrix $\mathbf{S}$, which has exactly 1 non-zero element per column. The element can be -1 or +1 uniformly. To implement this algorithm in streaming manner, matrices $\mathbf{C}$ and $\mathbf{D}$ are generated by adding or subtracting the rows of $\mathbf{A}$ and $\mathbf{B}$ randomly. More specifically, $\mathbf{C}$ and $\mathbf{D}$ are initialized to be $\ell \times m_1$ and $\ell \times m_2$ all zeros matrices. When processing rows of $\mathbf{A}$ and $\mathbf{B}$, we perform $\mathbf{c}_{h(i)} \leftarrow \mathbf{c}_{h(i)} + s(i)\mathbf{a}_i$ and $\mathbf{d}_{h(i)} \leftarrow \mathbf{d}_{h(i)} + s(i)\mathbf{b}_i$, where $h\colon [n] \to [\ell]$ and $s\colon [n] \to \{-1, 1\}$ are perfect hash functions.

### 4.1 Approximate Matrix Multiplication

In this section, both $\mathbf{A}$ and $\mathbf{B}$ are random matrices. Each entry of $\mathbf{A}$ and of $\mathbf{B}$ is independent and uniformly distributed on $[0, 1]$. For random algorithms, we execute 50 times for each sketch size and report the corresponding means and variances. The experiments are conducted in Matlab and run on a PC with Intel Core i5 CPU,4GB RAM, and Windows 7 64bit system. The Matlab is set in single thread mode when evaluating time. The performance is measured in terms of accuracy, stability, and speed.

**Accuracy**

Accuracy is calculated by $\|\mathbf{A}^T\mathbf{B} - \mathbf{C}^T\mathbf{D}\|_2$. In the first experiment, the product of two moderately sized matrices ($10000 \times 1000$ and $10000 \times 2000$) is approximated by the algorithms described above. Figure 1 shows the average accuracy with standard deviation over 50 trails when $\ell$ is set to different values. We shift the plots denoted by "Random Projection" and "Sparse Random Projection" a little bit right so they would not overlap with each other. Our results are similar with [Liberty, 2013]. Some key findings are listed below. First, the plot denoted by "FD-AMM Bound" is the theoretical worst case guaranteed by FD-AMM. It illustrates that our algorithm performs significantly better than expected by its worst case analysis. Second, the "FD-AMM Bound" is lower than the random methods all the time. It suggests that even the guaranteed worst case is lower than those of competing algorithms. Third, the three random algorithms perform equally well when considering the best results in 50 trials. The sampling method achieves better average accuracy compared to random projection and sparse random projection.

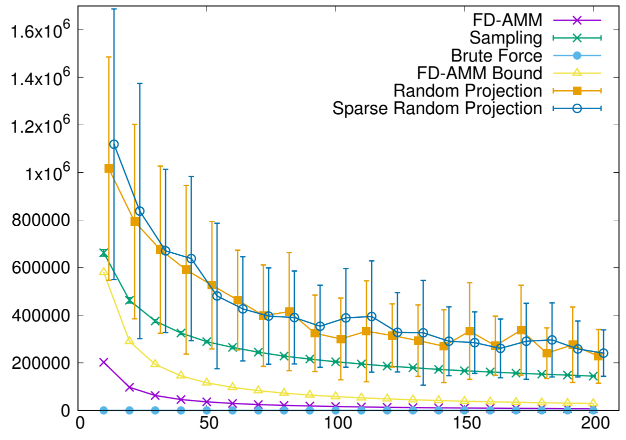


Figure 1: Accuracy of different algorithms. The $x$-axis indicates the value of $\ell$. And The $y$-axis is the accuracy which is measured by $\|\mathbf{A}^T\mathbf{B} - \mathbf{C}^T\mathbf{D}\|_2$.

**Stability**

Now we compare the stability of different algorithms. As mentioned earlier, FD-AMM is a deterministic method which returns the same output every time, while the other three methods are random algorithms. As Figure 1 shown, the accuracy achieved by the random projection and sparse random projection algorithms fluctuates widely between different trials, while the sampling method is more stable. Also, we can find the better stability can be achieved with large sketch size $\ell$, which is consistent with our intuition. Furthermore, we plot all the computation results of the sampling and FD-AMM methods in Figure 2. Even though the row sampling method yields a better approximation to the product compared to random projection and sparse random projection, it is still not deterministic.

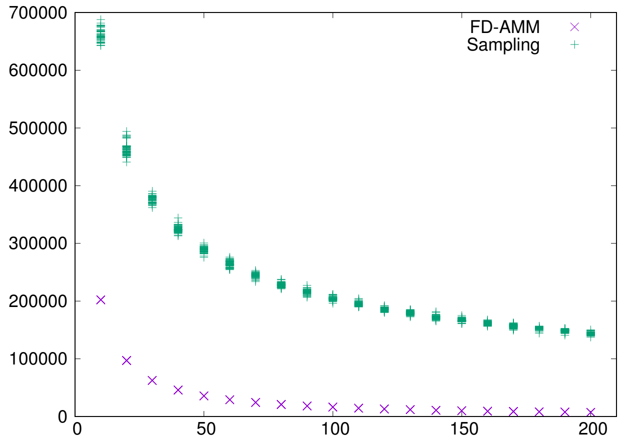


Figure 2: Stability of FD-AMM and Sampling.

**Speed**

The running time of our algorithm is between that of random projection and sampling. In Figure 3, the running time of different AMM algorithms are plotted as a function of their sketch sizes. We have some interesting observations here. First, the larger the sketch size, the longer the running time. However, the Brute Force method computes $\mathbf{A}^T\mathbf{B}$ directly, so its running time is independent of $\ell$. Second, the Sparse Random Projection method is fast and independent of $\ell$, but it's only slightly better than the Sampling method. Third, although FD-AMM shares the same theoretical time complexity ($\mathcal{O}(n\ell(m_1+m_2))$) with random projection, our method scales better.

## 4.2 Streaming CCA on Real-life Data

In this section, we test our streaming CCA algorithm on the annotated video dataset from the Mediamill Challenge [Snoek *et al.*, 2006]. There are 43907 images in total. Each image is a representative key frame of a video shot with 120 features and the whole set is annotated with 101 labels. We perform CCA to analyze the correlation structure between the features and labels. Figure 4(a) shows the canonical correlations and 4(b) shows the relative error. From the Figure 4(a), we can see that a few high correlations exist between the features and labels, with strong decay afterward. Figure 4(b) shows that the results of Streaming CCA are exactly the same as the results computed by the traditional method [Björck and Golub, 1973]. We compare the running time with the traditional method and the approximation algorithm in [Avron *et al.*, 2014]. It turns out that the Streaming

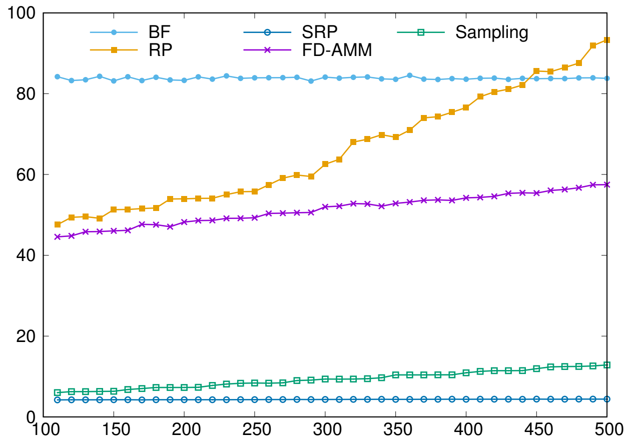
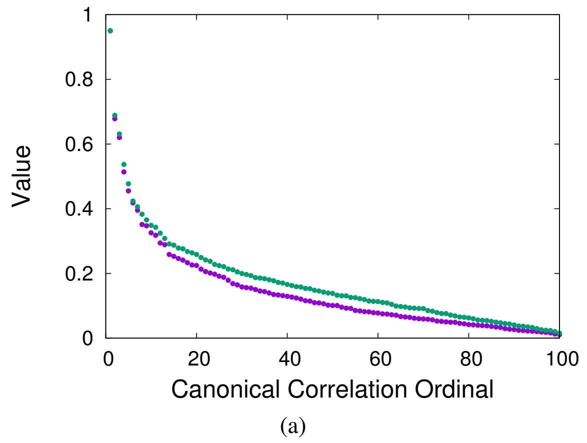


Figure 3: Running time in seconds of different algorithms. The $x$-axis indicates the value of $\ell$, and the $y$-axis is the running time in seconds.

CCA (taking 4.69 sec) is a little faster than traditional method (taking 5.93 sec) and slower than the approximation method (taking 2.51 sec).



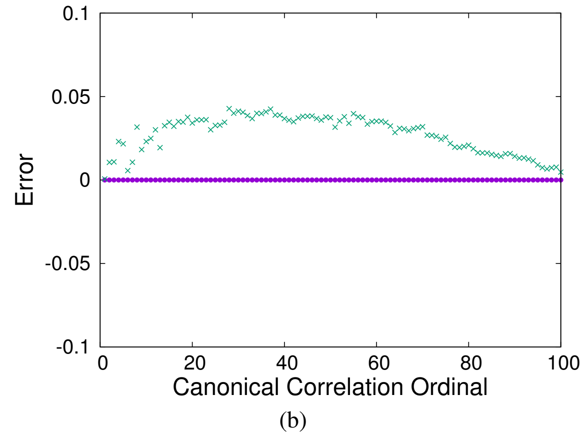


Figure 4: Comparison of streaming CCA and approximate CCA on the Mediamill data set.

## 5 Conclusions

In this paper, we have proposed a deterministic algorithm to compute the approximation of matrix product in a streaming way and provided theoretical analysis for the approximation performance. The experiments have shown that our algorithm is much more accurate than the other three random methods with an acceptable speed. We have applied our approach to computing CCA, giving rise to a fast streaming method. The experiments on real-world dataset have also validated the effectiveness of the method.

## Acknowledgments

This work has been supported by the National Natural Science Foundation of China (No. 61572017), Natural Science Foundation of Shanghai City (No. 15ZR1424200), and Microsoft Research Asia Collaborative Research Award.

## Appendix: Proof of Theorem 3

Let $\mathbf{H}^{(i)} = [\mathbf{C}^{(i)}\ \mathbf{D}^{(i)}]$ and $\mathbf{M}^{(i)} = [\mathbf{P}^{(i)}\ \mathbf{Q}^{(i)}]$. Let $\mathbf{Y} = [\mathbf{Y}_1; \mathbf{Y}_2]$, where $\mathbf{Y}_1$ is the first $m_1$ rows of $\mathbf{Y}$, and $\mathbf{Y}_2$ is the remaining $m_2$ rows of $\mathbf{Y}$. Let $\mathbf{a}_i$ and $\mathbf{b}_i$ denote the $i$-th row of $\mathbf{A}$ and $\mathbf{B}$. In what follows, we denote by $\delta_i$, $\mathbf{P}^{(i)}$, $\mathbf{Q}^{(i)}$, $\mathbf{C}^{(i)}$, $\mathbf{D}^{(i)}$ the values of $\delta$, $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{C}$ and $\mathbf{D}$ respectively after the main loop in the algorithm has been executed $i$ times. Let $\triangle = \sum_{i=1}^n \delta_i$ be the total mass we subtract from the stream during the algorithm. We start by proving three useful lemmas.

**Lemma 6.** *(Corollary 3.1.3 of [Horn and Johnson, 2012]) Let $\mathbf{A} \in \mathbb{R}^{n\times m}$ be given, and let $\mathbf{A}_r$ denote a submatrix of $\mathbf{A}$ obtained by deleting a total of r rows and/or columns from $\mathbf{A}$. Then*

$$\|\mathbf{A}\|_2 \geq \|\mathbf{A}_r\|_2$$

**Lemma 7.** *For any unit vector $\mathbf{x}_1 \in \mathbb{R}^{m_1}$ and $\mathbf{x}_2 \in \mathbb{R}^{m_2}$, we have $\mathbf{x}_1^T\mathbf{P}^{(i)^T}\mathbf{Q}^{(i)}\mathbf{x}_2 - \mathbf{x}_1^T\mathbf{C}^{(i)^T}\mathbf{D}^{(i)}\mathbf{x}_2 \leq \delta_i$.*

*Proof.* As $\mathbf{P}^{(i)} = \mathbf{S}\mathbf{Y}_1^T$, $\mathbf{Q}^{(i)} = \mathbf{S}\mathbf{Y}_2^T$, $\mathbf{C}^{(i)} = \mathbf{S}'\mathbf{Y}_1^T$, and $\mathbf{D}^{(i)} = \mathbf{S}'\mathbf{Y}_2^T$, then

$$\begin{aligned}
& \mathbf{x}_1^T\mathbf{P}^{(i)^T}\mathbf{Q}^{(i)}\mathbf{x}_2 - \mathbf{x}_1^T\mathbf{C}^{(i)^T}\mathbf{D}^{(i)}\mathbf{x}_2 \\
= \ & \mathbf{x}_1^T\mathbf{Y}_1\mathbf{S}^T\mathbf{S}\mathbf{Y}_2^T\mathbf{x}_2 - \mathbf{x}_1^T\mathbf{Y}_1\mathbf{S}'^T\mathbf{S}'\mathbf{Y}_2^T\mathbf{x}_2 \\
= \ & \mathbf{x}_1^T\mathbf{Y}_1\delta_i\mathbf{I}_\ell\mathbf{Y}_2^T\mathbf{x}_2 \\
= \ & \delta_i\mathbf{x}_1^T\mathbf{Y}_1\mathbf{Y}_2^T\mathbf{x}_2 \\
\leq \ & \delta_i\|\mathbf{Y}_1\mathbf{Y}_2^T\|_2 \\
\leq \ & \delta_i\|\mathbf{Y}_1\|_2\|\mathbf{Y}_2\|_2 \\
\leq \ & \delta_i.
\end{aligned}$$

In the last inequality, we use Lemma 6. □

**Lemma 8.** *For any unit vector $\mathbf{x}_1 \in \mathbb{R}^{m_1}$ and $\mathbf{x}_2 \in \mathbb{R}^{m_2}$, we have $0 \leq \mathbf{x}_1^T\mathbf{A}^T\mathbf{B}\mathbf{x}_2 - \mathbf{x}_1^T\mathbf{C}^T\mathbf{D}\mathbf{x}_2 \leq \triangle$.*

*Proof.* Notice that for all $2 \leq i \leq n$, we have

$$\mathbf{x}_1^T\mathbf{P}^{(i)^T}\mathbf{Q}^{(i)}\mathbf{x}_2 = \mathbf{x}_1^T\mathbf{C}^{(i-1)^T}\mathbf{D}^{(i-1)}\mathbf{x}_2 + \mathbf{x}_1^T\mathbf{a}_i^T\mathbf{b}_i\mathbf{x}_2$$

By substituting this into inequality from Lemma 7, we get

$$\mathbf{x}_1^T\mathbf{C}^{(i-1)^T}\mathbf{D}^{(i-1)}\mathbf{x}_2 + \mathbf{x}_1^T\mathbf{a}_i^T\mathbf{b}_i\mathbf{x}_2 \leq \mathbf{x}_1^T\mathbf{C}^{(i)^T}\mathbf{D}^{(i)}\mathbf{x}_2 + \delta_i$$

Subtracting $\mathbf{x}_1^T\mathbf{C}^{(i-1)^T}\mathbf{D}^{(i-1)}\mathbf{x}_2$ from both sides and summing over $i$ reveals

$$\begin{aligned}
& \mathbf{x}_1^T\mathbf{A}^T\mathbf{B}\mathbf{x}_2 \\
= \ & \sum_{i=1}^n \mathbf{x}_1^T\mathbf{a}_i^T\mathbf{b}_i\mathbf{x}_2 \\
\leq \ & \sum_{i=1}^n (\mathbf{x}_1^T\mathbf{C}^{(i)^T}\mathbf{D}^{(i)}\mathbf{x}_2 - \mathbf{x}_1^T\mathbf{C}^{(i-1)^T}\mathbf{D}^{(i-1)}\mathbf{x}_2 + \delta_i) \\
= \ & \mathbf{x}_1^T\mathbf{C}^{(n)^T}\mathbf{D}^{(n)}\mathbf{x}_2 - \mathbf{x}_1^T\mathbf{C}^{(0)^T}\mathbf{D}^{(0)}\mathbf{x}_2 + \sum_{i=1}^n \delta_i \\
= \ & \mathbf{x}_1^T\mathbf{C}^T\mathbf{D}\mathbf{x}_2 + \triangle
\end{aligned}$$

To see the first inequality observe $\mathbf{x}_1^T\mathbf{C}^{(i-1)^T}\mathbf{D}^{(i-1)}\mathbf{x}_2 + \mathbf{x}_1^T\mathbf{a}_i^T\mathbf{b}_i\mathbf{x}_2 = \mathbf{x}_1^T\mathbf{P}^{(i)^T}\mathbf{Q}^{(i)}\mathbf{x}_2 \geq \mathbf{x}_1^T\mathbf{C}^{(i)^T}\mathbf{D}^{(i)}\mathbf{x}_2$ for all $1 \leq i \leq n$. Then we can expand

$$\begin{aligned}
& \mathbf{x}_1^T\mathbf{A}^T\mathbf{B}\mathbf{x}_2 \\
= \ & \sum_{i=1}^n (\mathbf{x}_1^T\mathbf{P}^{(i)^T}\mathbf{Q}^{(i)}\mathbf{x}_2 - \mathbf{x}_1^T\mathbf{C}^{(i-1)^T}\mathbf{D}^{(i-1)}\mathbf{x}_2) \\
\geq \ & \sum_{i=1}^n (\mathbf{x}_1^T\mathbf{C}^{(i)^T}\mathbf{D}^{(i)}\mathbf{x}_2 - \mathbf{x}_1^T\mathbf{C}^{(i-1)^T}\mathbf{D}^{(i-1)}\mathbf{x}_2) \\
= \ & \mathbf{x}_1^T\mathbf{C}^T\mathbf{D}\mathbf{x}_2.
\end{aligned}$$

□

With Lemma 8, we are ready to prove the main theorem.

**Theorem 3.**

$$\|\mathbf{A}^T\mathbf{B} - \mathbf{C}^T\mathbf{D}\|_2 \leq (\|\mathbf{A}\|_F^2 + \|\mathbf{B}\|_F^2)/\ell.$$

*Proof.* In the $i$-th round of the algorithm, $\|\mathbf{P}^{(i)}\|_F^2 + \|\mathbf{Q}^{(i)}\|_F^2 = \|\mathbf{C}^{(i)}\|_F^2 + \|\mathbf{D}^{(i)}\|_F^2 + \ell\delta_i$ and $\|\mathbf{P}^{(i)}\|_F^2 + \|\mathbf{Q}^{(i)}\|_F^2 = \|\mathbf{C}^{(i-1)}\|_F^2 + \|\mathbf{D}^{(i-1)}\|_F^2 + \|\mathbf{a}_i\|^2 + \|\mathbf{b}_i\|^2$. By solving for $\|\mathbf{a}_i\|^2$ and $\|\mathbf{b}_i\|^2$, and summing over $i$ we get

$$\begin{aligned}
& \|\mathbf{A}\|_F^2 + \|\mathbf{B}\|_F^2 \\
= \ & \sum_{i=1}^n (\|\mathbf{a}_i\|^2 + \|\mathbf{b}_i\|^2) \\
= \ & \sum_{i=1}^n (\|\mathbf{C}^{(i)}\|_F^2 + \|\mathbf{D}^{(i)}\|_F^2 \\
& -\|\mathbf{C}^{(i-1)}\|_F^2 - \|\mathbf{D}^{(i-1)}\|_F^2 + \ell\delta_i) \\
= \ & \|\mathbf{C}\|_F^2 + \|\mathbf{D}\|_F^2 + \ell\triangle.
\end{aligned}$$

Using that $\|\mathbf{C}\|_F^2 + \|\mathbf{D}\|_F^2 \geq 0$ we obtain $\triangle \leq (\|\mathbf{A}\|_F^2 + \|\mathbf{B}\|_F^2)/\ell$. Recall that for any unit vector $\mathbf{x}_1 \in \mathbb{R}^{m_1}$ and $\mathbf{x}_2 \in \mathbb{R}^{m_2}$,

$$\|\mathbf{A}^T\mathbf{B} - \mathbf{C}^T\mathbf{D}\|_2 = \sup_{\mathbf{x}_1,\mathbf{x}_2} |\mathbf{x}_1^T(\mathbf{A}^T\mathbf{B} - \mathbf{C}^T\mathbf{D})\mathbf{x}_2|.$$

Combining Lemma 8, we get

$$0 \leq \|\mathbf{A}^T\mathbf{B} - \mathbf{C}^T\mathbf{D}\|_2 \leq \triangle \leq (\|\mathbf{A}\|_F^2 + \|\mathbf{B}\|_F^2)/\ell.$$

□

## References

[Avron *et al.*, 2014] Haim Avron, Christos Boutsidis, Sivan Toledo, and Anastasios Zouzias. Efficient dimensionality reduction for canonical correlation analysis. *SIAM Journal on Scientific Computing*, 36(5):S111–S131, 2014.

[Bhojanapalli *et al.*, 2015] Srinadh Bhojanapalli, Prateek Jain, and Sujay Sanghavi. Tighter low-rank approximation via sampling the leveraged element. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 902–920. SIAM, 2015.

[Björck and Golub, 1973] Åke Björck and Gene H Golub. Numerical methods for computing angles between linear subspaces. *Mathematics of computation*, 27(123):579–594, 1973.

[Chaudhuri *et al.*, 2009] Kamalika Chaudhuri, Sham M Kakade, Karen Livescu, and Karthik Sridharan. Multi-view clustering via canonical correlation analysis. In *Proceedings of the 26th annual international conference on machine learning*, pages 129–136. ACM, 2009.

[Clarkson and Woodruff, 2009] Kenneth L Clarkson and David P Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 205–214. ACM, 2009.

[Clarkson and Woodruff, 2013] Kenneth L Clarkson and David P Woodruff. Low rank approximation and regression in input sparsity time. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 81–90. ACM, 2013.

[Cohen *et al.*, 2015] Michael B Cohen, Jelani Nelson, and David P Woodruff. Optimal approximate matrix product in terms of stable rank. *arXiv preprint arXiv:1507.02268*, 2015.

[Dasgupta and Gupta, 2003] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random structures and algorithms*, 22(1):60–65, 2003.

[Drineas and Kannan, 2001] Petros Drineas and Ravi Kannan. Fast monte-carlo algorithms for approximate matrix multiplication. In *focs*, page 452. IEEE, 2001.

[Drineas *et al.*, 2006] Petros Drineas, Ravi Kannan, and Michael W Mahoney. Fast monte carlo algorithms for matrices i: Approximating matrix multiplication. *SIAM Journal on Computing*, 36(1):132–157, 2006.

[Drineas *et al.*, 2012] Petros Drineas, Malik Magdon-Ismail, Michael W Mahoney, and David P Woodruff. Fast approximation of matrix coherence and statistical leverage. *The Journal of Machine Learning Research*, 13(1):3475–3506, 2012.

[Eriksson-Bique *et al.*, 2011] Sylvester Eriksson-Bique, Mary Solbrig, Michael Stefanelli, Sarah Warkentin, Ralph Abbey, and Ilse CF Ipsen. Importance sampling for a monte carlo matrix multiplication algorithm, with application to information retrieval. *SIAM Journal on Scientific Computing*, 33(4):1689–1706, 2011.

[Ghashami and Phillips, 2014] Mina Ghashami and Jeff M Phillips. Relative errors for deterministic low-rank matrix approximations. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 707–717. SIAM, 2014.

[Golub and Zha, 1995] Gene H Golub and Hongyuan Zha. *The canonical correlations of matrix pairs and their numerical computation*. Springer, 1995.

[Horn and Johnson, 2012] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

[Hotelling, 1936] Harold Hotelling. Relations between two sets of variates. *Biometrika*, pages 321–377, 1936.

[Kane and Nelson, 2014] Daniel M Kane and Jelani Nelson. Sparser johnson-lindenstrauss transforms. *Journal of the ACM (JACM)*, 61(1):4, 2014.

[Liberty, 2013] Edo Liberty. Simple and deterministic matrix sketching. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 581–588. ACM, 2013.

[Lu and Foster, 2014] Yichao Lu and Dean P Foster. Large scale canonical correlation analysis with iterative least squares. In *Advances in Neural Information Processing Systems*, pages 91–99, 2014.

[Madrid *et al.*, 2012] Humberto Madrid, Valia Guerra, and Marielba Rojas. Sampling techniques for monte carlo matrix multiplication with applications to image processing. In *Pattern Recognition*, pages 45–54. Springer, 2012.

[McWilliams *et al.*, 2013] Brian McWilliams, David Balduzzi, and Joachim M Buhmann. Correlated random features for fast semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 440–448, 2013.

[Nelson and Nguyên, 2013] John Nelson and Huy L Nguyên. Osnap: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 117–126. IEEE, 2013.

[Pagh, 2013] Rasmus Pagh. Compressed matrix multiplication. *ACM Transactions on Computation Theory (TOCT)*, 5(3):9, 2013.

[Sarlos, 2006] Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 143–152. IEEE, 2006.

[Snoek *et al.*, 2006] Cees GM Snoek, Marcel Worring, Jan C Van Gemert, Jan-Mark Geusebroek, and Arnold WM Smeulders. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *Proceedings of the 14th annual ACM international conference on Multimedia*, pages 421–430. ACM, 2006.

[Wang *et al.*, 2015] Weiran Wang, Raman Arora, Karen Livescu, and Jeff A Bilmes. Unsupervised learning of acoustic features via deep canonical correlation analysis. In *Proceedings of ICASSP*, 2015.