Online Interpretable Matrix Decomposition for Large-Scale Streaming Data

Supplementary Material

7. Sampling techniques

7.1. Leverage Score Sampling

Given a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ with rank-k SVD $\mathbf{A} \approx \mathbf{W}_k\mathbf{S}_k\mathbf{V}_k^T$, the leverage score of column j is defined as:

$$\ell_j = \sum_{i=1}^{k} v_{ji}^2 = \|\mathbf{V}_k(j,:)\|_2^2 \tag{12}$$

Algorithm 5 outlines the steps for computing the CUR decomposition using the leverage score-based sampling technique.

Algorithm 5 Leverage Score CUR Decomposition

Require: Matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, target rank k, selection counts c, r
1: Compute SVD: $\mathbf{A} = \mathbf{W}\mathbf{\Sigma}\mathbf{V}^T$ (rank k)
2: Compute column leverage scores: $\ell_j = \sum_{i=1}^{k} \mathbf{v}_{ji}^2$
3: Compute row leverage scores: $\ell_i = \sum_{i=1}^{k} \mathbf{w}_{ij}^2$
4: Sample c columns with probabilities $\propto \ell_j$
5: Sample r rows with probabilities $\propto \ell_i$
6: Construct matrices $\mathbf{C}$ and $\mathbf{R}$ from selected columns/rows
7: Compute $\mathbf{U} = \mathbf{C}^\dagger\mathbf{A}\mathbf{R}^\dagger$
8: **return** $\mathbf{C}, \mathbf{U}, \mathbf{R}$

Drineas et al. [16] proved that sampling columns proportionally to leverage scores yields provable approximation guarantees:

Theorem 1 (Drineas et al. [16]). *Let $\mathbf{A} \approx \mathbf{W}_k\mathbf{S}_k\mathbf{V}_k^T$ be the rank-k SVD. If columns and rows are sampled with probabilities proportional to leverage scores $\{\ell_j\}$ and $\{\ell_i\}$, and $c = O(k/\epsilon^2)$, $r = O(k/\epsilon^2)$, then with probability at least $1-\delta$:*

$$\|\mathbf{A} - \mathbf{CUR}\| \le (1+\epsilon)\|\mathbf{A} - \mathbf{A}_k,\| \tag{13}$$

where $\mathbf{A}_k$ is the best rank-k approximation.

7.2. DEIM for CUR

The DEIM selects a k row indices by processing the k orthonormal vectors of the $\mathbf{W} \in \mathbb{R}^{m\times k}$ matrix, which is denoted as $\mathbf{p} = [p_1, p_2, \ldots, p_k]$. Those indices are selected based on the interpolatory projector, which is given as $\mathcal{K} = \mathbf{W}(\mathbf{PW})^{-1}\mathbf{P}^T$, where $\mathbf{P} = \mathbf{I}_m(:,\mathbf{p}) \in \mathbb{R}^{m\times k}$ is the row selection matrix. In summary, the DEIM procedure consists of two main steps: (1) computing the residuals to eliminate the direction of the previously processed singular vectors and (2) finding the element's index with the maximum absolute value of the residual vector. This procedure is repeated until the last singular vector, $\mathbf{w}_k$, is processed to obtain the index p_k and return the selected row indices $\mathbf{p} = [p_1, p_2 \ldots, p_k]$. A similar procedure is applied to the right singular vectors (i.e., $\mathbf{V} \in \mathbb{R}^{n\times k}$) to get selected column indices, i.e., $\mathbf{q} = [q_1, q_2 \ldots, q_k]$. Algorithms 3 and 6 summarize the steps

Algorithm 6 CUR-DEIM [37]

Require: $\mathbf{A} \in \mathbb{R}^{m\times n}$, target rank k
Ensure: $\mathbf{C} \in \mathbb{R}^{m\times k}$, $\mathbf{U} \in \mathbb{R}^{k\times k}$, $\mathbf{R} \in \mathbb{R}^{k\times n}$ such that $\mathbf{A} \approx \mathbf{CUR}$
1: Compute SVD: $[\mathbf{W}, \mathbf{S}, \mathbf{V}] = \texttt{SVD}(\mathbf{A}, k)$ {Exact or approximated}
2: Compute selected indices:
3: $\mathbf{q} \leftarrow$ DEIM($\mathbf{V}$) {Algorithm 3}
4: $\mathbf{p} \leftarrow$ DEIM($\mathbf{W}$) {Algorithm 3}
5: Construct factor matrices:
6: $\mathbf{C} = \mathbf{A}(:,\mathbf{q})$
7: $\mathbf{R} = \mathbf{A}(\mathbf{p},:)$
8: Compute $\mathbf{U} = \mathbf{C}^\dagger\mathbf{A}\mathbf{R}^\dagger$

of the DEIM sampling method for computing the row indices and the CUR-DEIM method, respectively. As demonstrated in Lemma 3.2 in [13], selecting the indices of the maximal residual element at each iteration limits increasing the $\|\mathcal{K}\mathbf{x} - \mathbf{x}\|_2$ for any vector $\mathbf{x}$. A similar bound is proved in [37] for matrices. Specifically, let a set of distinct indices $\mathbf{p} \in \mathbb{N}^k$ is given, and suppose $\mathbf{P}^T\mathbf{W}$ is invertible. If $\mathbf{W}^T\mathbf{W} = \mathbf{I}_k$, then for any matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, the following bound is satisfied

$$\begin{aligned}\|\mathbf{A} - \mathcal{K}\mathbf{A}\|_2 &\le \|(\mathbf{PW})^{-1}\|_2\|(\mathbf{I}_m - \mathbf{W}\mathbf{W}^T)\mathbf{A}\|_2 \\ &= \|(\mathbf{PW})^{-1}\|_2\sigma_{k+1},\end{aligned}$$

where σ_{k+1} is the first neglected singular value.

7.3. QDEIM

Given the rank-k SVD approximation of $\mathbf{A}$ as $\mathbf{A} \approx \mathbf{W}\mathbf{\Sigma}\mathbf{V}^T$, then the column indices vector $\mathbf{q} \in \mathbb{N}^k$ selected by QDEIM is computed as follows, $\mathbf{V}^T\mathbf{S} = \mathbf{QT}$, where $\mathbf{S} = \mathbf{I}_n(:, [\mathbf{q}_1 \quad \mathbf{q}_2]) \in \mathbb{R}^{n\times n}$ is a permutation matrix, $\mathbf{Q} \in \mathbb{R}^{k\times k}$ and $\mathbf{T} \in \mathbb{R}^{k\times n}$ are the orthogonal and upper-triangular matrices, respectively, resulting from the pivot QR of $\mathbf{V}^T$. The selected column indices are the first k pivot column indices (i.e., $\mathbf{q}_1$) [17]. The QDEIM implementation is straightforward using available software (e.g., MATLAB) and well-optimized QR column-pivoting methods.

8. Analysis of the Proposed Methods

8.1. Complete Incremental CUR Algorithm

In this section, we present the complete steps of the proposed incremental CUR algorithms. To improve computational efficiency in streaming scenarios, we develop an online CUR-type strategy that avoids the need for complete recomputation, as shown in Algorithm 7.

Algorithm 7 Complete Step of Incremental CUR

Require: New batch $\mathbf{B}_t \in \mathbb{R}^{m\times n_t}$, parameters (k, c, r), method $\in \{leverage, DEIM\}$
Require: Previous state: $\mathbf{W}_t, \mathbf{S}_t, \mathbf{V}_t, \mathbf{C}_t, \mathcal{I}_t, \mathcal{J}_t$
SVD Update
$\mathbf{W}_{t+1}, \mathbf{S}_{t+1}, \mathbf{V}_{t+1} = \text{IncSVD}(\mathbf{W}_t, \mathbf{S}_t, \mathbf{V}_t, \mathbf{B}_t, k)$
Column and Row Selection
if method = *leverage* **then**
  $[\mathbf{C}_{t+1}, \mathcal{J}_{t+1}] \leftarrow$ Applying Algorithm 1;
  $[\mathbf{R}_{t+1}, \mathcal{I}_{t+1}] \leftarrow$ Applying Algorithm 2;
else if method = *DEIM* **then**
  $[\mathbf{C}_{t+1}, \mathbf{R}_{t+1}, \mathcal{I}_{t+1}, \mathcal{J}_{t+1}] \leftarrow$ Applying Algorithm 4;
end if
U Matrix Computation
Compute $\mathbf{M}_1 = \mathbf{C}_{t+1}^{\dagger} \cdot \mathbf{W}_{t+1}$
Compute $\mathbf{M}_2 = \mathbf{S}_{t+1} \cdot \mathbf{V}_{t+1}^T \cdot \mathbf{R}_{t+1}^{\dagger}$
$\mathbf{U}_{t+1} = \mathbf{M}_1 \cdot \mathbf{M}_2$
return $\mathbf{W}_{t+1}, \mathbf{S}_{t+1}, \mathbf{V}_{t+1}$
return $\mathbf{C}_{t+1}, \mathbf{U}_{t+1}, \mathbf{R}_{t+1}, \mathcal{I}_{t+1}, \mathcal{J}_{t+1}$

8.2. Error Analysis of the Proposed Methods

Let the cumulative data matrix at streaming time $t+1$ be

$$\mathbf{A}_{t+1} = [\mathbf{A}_t,\ \mathbf{B}_t] \in \mathbb{R}^{m\times n_{1:t}},$$

where each new block $\mathbf{B}_t$ corresponds to newly appended data. Our proposed incremental CUR algorithm based on the leverage score sampling approximates the data matrix $\mathbf{A}_{t+1}$ as,

$$\mathbf{A}_{t+1} \approx \mathbf{C}_{t+1}\,\mathbf{U}_{t+1}\,\mathbf{R}_{t+1},$$

where $\mathbf{C}_{t+1}$ contains selected columns, $\mathbf{R}_{t+1}$ contains the approxiamted selected rows, and $\mathbf{U}_{t+1}$ is a middle matrix.

8.2.1. Error Analysis of Incremental CUR (Inc-LS)

In our first proposed method (Inc-LS), the selections are made deterministically via the leverage scores of the current SVD approximation, as presented in Algorithms 1 and 2. At time step $t+1$, we only have access to an incremental SVD factors of $\mathbf{A}_{t+1}$. Using Brand's algorithm (or any equivalent incremental update), we maintain a low-rank SVD:

$$\mathbf{A}_{t+1} \approx \mathbf{W}_{t+1}\,\mathbf{\Sigma}_{t+1}\,\mathbf{V}_{t+1}^T, \tag{14}$$

where $\mathbf{W}_{t+1} \in \mathbb{R}^{m\times k}$ and $\mathbf{V}_{t+1} \in \mathbb{R}^{n_{1:t}\times k}$ have orthonormal columns, and $\mathbf{\Sigma}_{t+1} \in \mathbb{R}^{k\times k}$ is diagonal. We denote the incremental SVD residual by

$$\begin{aligned}\mathbf{E}_{t+1}^{inc} &:= \mathbf{A}_{t+1} - \mathbf{W}_{t+1}\,\mathbf{\Sigma}_{t+1}\,\mathbf{V}_{t+1}^T,\\ \varepsilon_{t+1}^{inc} &:= \|\mathbf{E}_{t+1}^{inc}\|_F^2.\end{aligned} \tag{15}$$

We compute leverage scores to identify informative columns and rows. For $\mathbf{V}_{t+1} = [\mathbf{v}_1, \ldots, \mathbf{v}_k]$ and $\mathbf{W}_{t+1} = [\mathbf{w}_1, \ldots, \mathbf{w}_k]$:

$$\ell_{t+1}^j = \|\mathbf{V}_{t+1}(j,:)\|_2^2, \qquad \ell_{t+1}^i = \|\mathbf{W}_{t+1}(i,:)\|_2^2.$$

We deterministically choose the c columns of $\mathbf{A}_{t+1}$ with largest ℓ_{t+1}^j values and the r rows with largest ℓ_{t+1}^i values to form

$$\mathbf{C}_{t+1} := \mathbf{A}_{t+1}(:, \mathcal{J}), \qquad \mathbf{R}_{t+1} := \mathbf{A}_{t+1}(\mathcal{I}, :).$$

The selected indices $\mathcal{J}, \mathcal{I}$ depend implicitly on the current singular subspaces. For a given pair $(\mathbf{C}_{t+1}, \mathbf{R}_{t+1})$, the optimal middle factor that minimizes the Frobenius error is given by

$$\mathbf{U}_{t+1}^{\star} = \mathbf{C}_{t+1}^{+}\,\mathbf{A}_{t+1}\,\mathbf{R}_{t+1}^{+},$$

where $(\cdot)^+$ denotes the Moore–Penrose pseudoinverse. The deterministic leverage-score CUR analysis in [16] states that if $\mathbf{C}_{t+1}, \mathbf{R}_{t+1}$ are chosen according to leverage scores of the top-k singular subspaces, then

$$\begin{aligned}\|\mathbf{A}_{t+1} - \mathbf{C}_{t+1}\mathbf{U}_{t+1}^{\star}\mathbf{R}_{t+1}\|_F^2 \ \le\ \\ (1+\epsilon)\,\|\mathbf{A}_{t+1} - \mathbf{A}_{t+1}^k\|_F^2,\end{aligned} \tag{16}$$

where $\mathbf{A}_{t+1}^k$ is the best rank-k approximation of $\mathbf{A}_{t+1}$ and $\epsilon = O(k/c) + O(k/r)$ depends on the number of selected rows/columns. Explicitly forming $\mathbf{A}_{t+1}$ to compute $\mathbf{U}_{t+1}^{\star} = \mathbf{C}_{t+1}^{+}\,\mathbf{A}_{t+1}\,\mathbf{R}_{t+1}^{+}$ is computationally expensive and infeasible in streaming settings. Hence, we use the incremental SVD factors to approximate $\mathbf{U}_{t+1}$ as:

$$\mathbf{U}_{t+1} := \mathbf{C}_{t+1}^{+}\mathbf{W}_{t+1}\mathbf{\Sigma}_{t+1}\mathbf{V}_{t+1}^T\mathbf{R}_{t+1}^{+}. \tag{17}$$

This can be factorized for clarity as

$$\begin{aligned}\mathbf{M}_1 &= \mathbf{C}_{t+1}^{+}\mathbf{W}_{t+1} \in \mathbb{R}^{c\times k},\\ \mathbf{M}_2 &= \mathbf{\Sigma}_{t+1}\mathbf{V}_{t+1}^T\mathbf{R}_{t+1}^{+} \in \mathbb{R}^{k\times r},\end{aligned} \tag{18}$$

such that $\mathbf{U}_{t+1} = \mathbf{M}_1\mathbf{M}_2$. The resulting CUR reconstruction is

$$\widehat{\mathbf{A}}_{t+1} = \mathbf{C}_{t+1}\mathbf{U}_{t+1}\mathbf{R}_{t+1}. \tag{19}$$

If $\mathbf{A}_{t+1} = \mathbf{W}_{t+1}\,\mathbf{\Sigma}_{t+1}\,\mathbf{V}_{t+1}^T$ exactly, then $\mathbf{U}_{t+1} = \mathbf{U}_{t+1}^{\star}$. Due to incremental SVD approximation, $\mathbf{U}_{t+1}$ differs slightly. Let the overall incremental CUR residual as

$$\mathbf{E}_{t+1} := \mathbf{A}_{t+1} - \mathbf{C}_{t+1}\mathbf{U}_{t+1}\mathbf{R}_{t+1}. \tag{20}$$

Substituting (14) and (17) gives

$$\begin{aligned}\mathbf{E}_{t+1} =&(\mathbf{W}_{t+1}\,\mathbf{\Sigma}_{t+1}\,\mathbf{V}_{t+1}^T + \mathbf{E}_{t+1}^{inc}) - \\ &\mathbf{C}_{t+1}(\mathbf{C}_{t+1}^{+}\mathbf{W}_{t+1}\mathbf{\Sigma}_{t+1}\mathbf{V}_{t+1}^T\mathbf{R}_{t+1}^{+})\mathbf{R}_{t+1} \\ =&(\mathbf{W}_{t+1}\,\mathbf{\Sigma}_{t+1}\,\mathbf{V}_{t+1}^T - \mathbf{C}_{t+1}(\mathbf{C}_{t+1}^{+}\mathbf{W}_{t+1}\mathbf{\Sigma}_{t+1} \\ &\mathbf{V}_{t+1}^T\mathbf{R}_{t+1}^{+})\mathbf{R}_{t+1}) + \mathbf{E}_{t+1}^{inc}.\end{aligned} \tag{21}$$

By adding and subtracting the optimal CUR reconstruction $\mathbf{C}_{t+1}\mathbf{U}_{t+1}^{\star}\mathbf{R}_{t+1}$ to separate the errors as:

$$\begin{aligned}\mathbf{E}_{t+1} =& \underbrace{(\mathbf{A}_{t+1} - \mathbf{C}_{t+1}\mathbf{U}_{t+1}^{\star}\mathbf{R}_{t+1})}_{\text{CUR sampling error}} + \\ &\underbrace{(\mathbf{C}_{t+1}\mathbf{U}_{t+1}^{\star}\mathbf{R}_{t+1} - \mathbf{C}_{t+1}\mathbf{U}_{t+1}\mathbf{R}_{t+1})}_{U\text{-matrix error}} + \\ &\underbrace{\mathbf{E}_{t+1}^{inc}}_{\text{incremental SVD error}}.\end{aligned} \tag{22}$$

By the triangle inequality for the Frobenius norm, we have

$$\begin{aligned}\|\mathbf{E}_{t+1}\|_F \le&\|\mathbf{A}_{t+1} - \mathbf{C}_{t+1}\mathbf{U}_{t+1}^{\star}\mathbf{R}_{t+1}\|_F + \\ &\|\mathbf{C}_{t+1}\mathbf{U}_{t+1}^{\star}\mathbf{R}_{t+1} - \mathbf{C}_{t+1}\mathbf{U}_{t+1}\mathbf{R}_{t+1}\|_F + \\ &\|\mathbf{E}_{t+1}^{inc}\|_F.\end{aligned} \tag{23}$$

Let us define the three error magnitudes:

$$\begin{cases}\varepsilon_{t+1}^{cur} := \|\mathbf{A}_{t+1} - \mathbf{C}_{t+1}\mathbf{U}_{t+1}^{\star}\mathbf{R}_{t+1}\|_F^2, \\ \varepsilon_{t+1}^{u} := \|\mathbf{C}_{t+1}\mathbf{U}_{t+1}^{\star}\mathbf{R}_{t+1} - \mathbf{C}_{t+1}\mathbf{U}_{t+1}\mathbf{R}_{t+1}\|_F^2, \\ \varepsilon_{t+1}^{inc} := \|\mathbf{E}_{t+1}^{inc}\|_F^2.\end{cases}$$

Squaring and using $(a+b+c)^2 \le 3(a^2+b^2+c^2)$ gives $\|\mathbf{E}_{t+1}\|_F^2 \le 3\big(\varepsilon_{t+1}^{cur} + \varepsilon_{t+1}^{u} + \varepsilon_{t+1}^{inc}\big)$ This constant factor 3 can be absorbed into $(1+\epsilon)$ notation in expectation. Using the deterministic bound (16) for ε_{t+1}^{cur} yields

$$\mathbb{E}\|\mathbf{E}_{t+1}\|_F^2 \le (1+\epsilon)\,\|\mathbf{A}_{t+1} - \mathbf{A}_{t+1}^{k}\|_F^2 + \underbrace{(\varepsilon_{t+1}^{inc} + \varepsilon_{t+1}^{u})}_{\delta_{t+1}}. \tag{24}$$

Define explicitly $\delta_{t+1} := \varepsilon_{t+1}^{inc} + \varepsilon_{t+1}^{u}$, where the term ε_{t+1}^{inc} represents the error of maintaining the SVD incrementally, and ε_{t+1}^{u} represents the error of approxiamting the $\mathbf{U}_{t+1}$. Both are typically small if the incremental SVD is accurate and $\mathbf{C}_{t+1}, \mathbf{R}_{t+1}$ are well-conditioned. Therfore, at each incremental step when a new batch $\mathbf{B}_t$ arrives, the updated SVD $(\mathbf{W}_{t+1}, \mathbf{\Sigma}_{t+1}, \mathbf{V}_{t+1})$ yields new CUR components $(\mathbf{C}_{t+1}, \mathbf{R}_{t+1}, \mathbf{U}_{t+1})$. Applying the same analysis to $\mathbf{A}_{t+1}$ gives the recursive error bound:

$$\begin{aligned}\mathbb{E}\|\mathbf{A}_{t+1} - &\mathbf{C}_{t+1}\mathbf{U}_{t+1}\mathbf{R}_{t+1}\|_F^2 \le \\ &(1+\epsilon)\,\|\mathbf{A}_{t+1} - \mathbf{A}_{t+1}^{k}\|_F^2 + \delta_{t+1},\end{aligned} \tag{25}$$

where ϵ depends on the number of sampled columns/rows and $\delta_{t+1} = \varepsilon_{t+1}^{inc} + \varepsilon_{t+1}^{u}$ is the total incremental error. Thus, the incremental procedure preserves the desirable CUR guarantee up to a small additive term δ_{t+1}, which captures accumulated incremental and middle matrix approximation errors.

8.2.2. Error Analysis of Inc-DEIM

In this section, we analyze the error bound of the proposed incremental CUR method, which leverages the DEIM sampling technique. Let $\mathbf{A}_{t+1} \in \mathbb{R}^{m \times n_{1:t}}$ denote the data matrix available up to step $t+1$. Suppose that at step $t+1$, we have obtained an approximate rank-k truncated incremental SVD as $\mathbf{W}_{t+1}\,\mathbf{\Sigma}_{t+1}\,\mathbf{V}_{t+1}^T$, and let $\mathcal{I}_{t+1}$ and $\mathcal{J}_{t+1}$ be the row and column indices selected by our proposed incremental CUR based on DEIM sampling, and define the corresponding selection matrices

$$\begin{aligned}\mathbf{P}_{t+1} &= \mathbf{I}(:, \mathcal{I}_{t+1}) \in \mathbb{R}^{m \times k}, \\ \mathbf{S}_{t+1} &= \mathbf{I}(:, \mathcal{J}_{t+1}) \in \mathbb{R}^{n_{1:t} \times k}.\end{aligned} \tag{26}$$

Then the approxiamtion of $\mathbf{A}_{t+1}$ is given as,

$$\mathbf{A}_{t+1} \approx \mathbf{C}_{t+1}\,\mathbf{U}_{t+1}\,\mathbf{R}_{t+1}, \tag{27}$$

where $\mathbf{C}_{t+1} = \mathbf{A}_{t+1}\mathbf{S}_{t+1}$ and $\mathbf{R}_{t+1} = \mathbf{P}_{t+1}^T\mathbf{A}_{t+1}$. Let $\mathbf{U}_{t+1} \in \mathbb{R}^{k \times k}$ denote the middle matrix constructed from the low-rank factors of as

$$\mathbf{U}_{t+1} = (\mathbf{S}_{t+1}^T\mathbf{W}_{t+1})^{-1}\mathbf{\Sigma}_{t+1}(\mathbf{V}_{t+1}^T\mathbf{P}_{t+1})^{-1}. \tag{28}$$

Then the CUR approximation error satisfies the bound

$$\begin{aligned}&\|\mathbf{A}_{t+1} - \mathbf{C}_{t+1}\,\mathbf{U}_{t+1}\,\mathbf{R}_{t+1}\| \le \\ &\|(\mathbf{V}_{t+1}^T\mathbf{S}_{t+1})^{-1}\|\,\|\mathbf{A}_{t+1}(\mathbf{I} - \mathbf{V}_{t+1}\mathbf{V}_{t+1}^T)\| + \\ &\|(\mathbf{P}_{t+1}^T\mathbf{W}_{t+1})^{-1}\|\,\|(I - \mathbf{W}_{t+1}\mathbf{W}_{t+1}^T)\mathbf{A}_{t+1}\|.\end{aligned} \tag{29}$$

8.2.3. Proof

The DEIM algorithm associates to the bases $\mathbf{W}_{t+1}$ and $\mathbf{V}_{t+1}$ two oblique projectors as

$$\begin{aligned}\mathcal{P}_W &= \mathbf{W}_{t+1}(\mathbf{P}_{t+1}^T\mathbf{W}_{t+1})^{-1}\mathbf{P}_{t+1}^T, \\ \mathcal{P}_V &= \mathbf{V}_{t+1}(\mathbf{S}_{t+1}^T\mathbf{V}_{t+1})^{-1}\mathbf{S}_{t+1}^T,\end{aligned} \tag{30}$$

The DEIM index selection guarantees that $(\mathbf{P}_{t+1}^T\mathbf{W}_{t+1})$ and $(\mathbf{S}_{t+1}^T\mathbf{V}_{t+1})$ are nonsingular and not severely ill–conditioned [13, 37]. Note that, by construction, $\mathcal{P}_W$ and $\mathcal{P}_V$ are not orthogonal projectors but satisfy the interpolation conditions

$$\mathbf{P}_{t+1}^T\mathcal{P}_W = \mathbf{P}_{t+1}^T, \qquad \mathcal{P}_V\mathbf{S}_{t+1} = \mathbf{S}_{t+1},$$

ensuring exact reconstruction of the selected rows and columns. The CUR reconstruction can be written as

$$\begin{aligned}\mathbf{C}_{t+1}\,\mathbf{U}_{t+1}\,\mathbf{R}_{t+1} =&\mathbf{A}_{t+1}\mathbf{S}_{t+1}(\mathbf{S}_{t+1}^T\mathbf{W}_{t+1})^{-1}\mathbf{\Sigma}_{t+1} \\ &(\mathbf{V}_{t+1}^T\mathbf{P}_{t+1})^{-1}\mathbf{P}_{t+1}^T\mathbf{A}_{t+1} \\ =&\mathcal{P}_W^T\mathbf{A}_{t+1}\mathcal{P}_V,\end{aligned}$$

which means it projects $\mathbf{A}_{t+1}$ obliquely onto the spaces spanned by the selected columns/rows of $\mathbf{W}_{t+1}$ and $\mathbf{V}_{t+1}$. Let error is deifed as $\mathbf{E}_{t+1} = \mathbf{A}_{t+1} - \mathbf{C}_{t+1}\,\mathbf{U}_{t+1}\,\mathbf{R}_{t+1}$, the we obtain

$$\mathbf{E}_{t+1} = \mathbf{A}_{t+1} - \mathcal{P}_W^T \mathbf{A}_{t+1} \mathcal{P}_V. \tag{31}$$

By adding and subtracting $\mathcal{P}_W^T \mathbf{A}_{t+1}$ gives

$$\begin{aligned} \mathbf{E}_{t+1} &= (\mathbf{A}_{t+1} - \mathcal{P}_W^T \mathbf{A}_{t+1}) + \mathcal{P}_W^T \mathbf{A}_{t+1}(\mathbf{I} - \mathcal{P}_V) \\ &= (\mathbf{I} - \mathcal{P}_W^T)\mathbf{A}_{t+1} + \mathcal{P}_W^T \mathbf{A}_{t+1}(\mathbf{I} - \mathcal{P}_V). \end{aligned} \tag{32}$$

By taking the norm and applying the triangle inequality,

$$\|\mathbf{E}_{t+1}| \leq \|(\mathbf{I} - \mathcal{P}_W^T)\mathbf{A}_{t+1}\| + \|\mathcal{P}_W^T \mathbf{A}_{t+1}(\mathbf{I} - \mathcal{P}_V)\| \tag{33}$$

This separates the total reconstruction error into a row space component and a column space component. The first term can be bounded as

$$(\mathbf{I} - \mathcal{P}_W^T)\mathbf{A}_{t+1} = (\mathbf{I} - \mathbf{W}_{t+1}(\mathbf{P}_{t+1}^T \mathbf{W}_{t+1})^{-1}\mathbf{P}_{t+1}^T)\mathbf{A}_{t+1}$$

By multiplying by $\mathbf{W}_{t+1}\mathbf{W}_{t+1}^T$ and its orthogonal complement $\mathbf{I} - \mathbf{W}_{t+1}\mathbf{W}_{t+1}^T$, we decompose it as

$$\mathbf{A}_{t+1} = \mathbf{W}_{t+1}\mathbf{W}_{t+1}^T \mathbf{A}_{t+1} + (\mathbf{I} - \mathbf{W}_{t+1}\mathbf{W}_{t+1}^T)\mathbf{A}_{t+1}.$$

Since $(\mathbf{I} - \mathcal{P}_W^T)\mathbf{W}_{t+1} = 0$, the first term vanishes. Consequently, $(\mathbf{I} - \mathcal{P}_W^T)\mathbf{A}_{t+1} = (\mathbf{I} - \mathcal{P}_W^T)(\mathbf{I} - \mathbf{W}_{t+1}\mathbf{W}_{t+1}^T)\mathbf{A}_{t+1}$. By taking norms and using submultiplicativity,

$$\begin{aligned} &\|(\mathbf{I} - \mathcal{P}_W^T)\mathbf{A}_{t+1}\| \leq \\ &\|\mathbf{I} - \mathcal{P}_W^T\|\,\|(\mathbf{I} - \mathbf{W}_{t+1}\mathbf{W}_{t+1}^T)\mathbf{A}_{t+1}\| \leq \\ &\|(\mathbf{P}_{t+1}^T \mathbf{W}_{t+1})^{-1}\|\,\|(\mathbf{I} - \mathbf{W}_{t+1}\mathbf{W}_{t+1}^T)\mathbf{A}_{t+1}\|. \end{aligned} \tag{34}$$

Hence, the first term satisfies

$$\|(\mathbf{I} - \mathcal{P}_W^T)\mathbf{A}_{t+1}\| \leq \|\mathbf{I} - \mathcal{P}_W^T\|\,\|(\mathbf{I} - \mathbf{W}_{t+1}\mathbf{W}_{t+1}^T)\mathbf{A}_{t+1}\| \tag{35}$$

For the DEIM projection, a well-known bound ([17]) states $\|\mathbf{I} - \mathcal{P}_W^T\| \leq \|(\mathbf{P}_{t+1}^T \mathbf{W}_{t+1})^{-1}\|$. Thus we obtain

$$\begin{aligned} \|(\mathbf{I} - \mathcal{P}_W^T)\mathbf{A}_{t+1}\| \leq \|(\mathbf{P}_{t+1}^T \mathbf{W}_{t+1})^{-1}\| \\ \|(\mathbf{I} - \mathbf{W}_{t+1}\mathbf{W}_{t+1}^T)\mathbf{A}_{t+1}\|. \end{aligned} \tag{36}$$

The second term in Equation (33) can be bounded as follows. Since, $\mathbf{I} - \mathcal{P}_V = \mathbf{I} - \mathbf{V}_{t+1}(\mathbf{S}_{t+1}^T \mathbf{V}_{t+1})^{-1}\mathbf{S}_{t+1}^T$, then

$$\begin{aligned} \mathbf{A}_{t+1}(\mathbf{I} - \mathcal{P}_V) = \mathbf{A}_{t+1}(\mathbf{I} - \mathbf{V}_{t+1}\mathbf{V}_{t+1}^T) + \\ \mathbf{A}_{t+1}[\mathbf{V}_{t+1}\mathbf{V}_{t+1}^T - \mathbf{V}_{t+1}(\mathbf{S}_{t+1}^T \mathbf{V}_{t+1})^{-1}\mathbf{S}_{t+1}^T]. \end{aligned}$$

Since $\mathbf{V}_{t+1}\mathbf{V}_{t+1}^T$ lies in the subspace spanned by $\mathbf{V}_{t+1}$, the residual term is governed by $\|(\mathbf{V}_{t+1}^T \mathbf{S}_{t+1})^{-1}\|$. Thus,

$$\begin{aligned} \|\mathbf{A}_{t+1}(\mathbf{I} - \mathcal{P}_V)\| \leq \|(\mathbf{V}_{t+1}^T \mathbf{S}_{t+1})^{-1}\| \\ \|\mathbf{A}_{t+1}(\mathbf{I} - \mathbf{V}_{t+1}\mathbf{V}_{t+1}^T)\|. \end{aligned} \tag{37}$$

Consequently,

$$\begin{aligned} \|\mathcal{P}_W^T \mathbf{A}_{t+1}(\mathbf{I} - \mathcal{P}_V)\| \leq \|\mathcal{P}_W^T\|\,\|\mathbf{A}_{t+1}(\mathbf{I} - \mathcal{P}_V)\| \leq \\ \|(\mathbf{V}_{t+1}^T \mathbf{S}_{t+1})^{-1}\|\,\|\mathbf{A}_{t+1}(\mathbf{I} - \mathbf{V}_{t+1}\mathbf{V}_{t+1}^T)\| \end{aligned} \tag{38}$$

By substituting (36) and (38) into (33), we obtain

$$\begin{aligned} &\|\mathbf{A}_{t+1} - \mathbf{C}_{t+1}\,\mathbf{U}_{t+1}\,\mathbf{R}_{t+1}\| \leq \\ &\|(\mathbf{V}_{t+1}^T \mathbf{S}_{t+1})^{-1}\|\,\|\mathbf{A}_{t+1}(\mathbf{I} - \mathbf{V}_{t+1}\mathbf{V}_{t+1}^T)\| + \\ &\|(\mathbf{P}_{t+1}^T \mathbf{W}_{t+1})^{-1}\|\|(\mathbf{I} - \mathbf{W}_{t+1}\mathbf{W}_{t+1}^T)\mathbf{A}_{t+1}\|, \end{aligned} \tag{39}$$

where $\|\mathbf{A}_{t+1}(\mathbf{I} - \mathbf{V}_{t+1}\mathbf{V}_{t+1}^T)\|$ measures the portion of $\mathbf{A}_{t+1}$ lying outside the span of the right singular vectors $\mathbf{V}_{t+1}$ (quantifying column–space truncation error), $\|(\mathbf{I} - \mathbf{W}_{t+1}\mathbf{W}_{t+1}^T)\mathbf{A}_{t+1}\|$ measures the loss in the left singular subspace (quantifying row–space truncation error), and $\|(\mathbf{V}_{t+1}^T \mathbf{S}_{t+1})^{-1}\|$ and $\|(\mathbf{P}_{t+1}^T \mathbf{W}_{t+1})^{-1}\|\|$ are the DEIM conditioning factors, which express the stability of the corresponding pivot selections. In the incremental SVD update, as presented in Equation 15. Because $(\mathbf{I} - \mathbf{W}_{t+1}\mathbf{W}_{t+1}^T)\mathbf{A}_{t+1}$ and $\mathbf{A}_{t+1}(\mathbf{I} - \mathbf{V}_{t+1}\mathbf{V}_{t+1}^T)$ already measure the energy not captured by $(\mathbf{W}_{t+1}, \mathbf{V}_{t+1})$, the residual ε_{t+1}^{inc} is implicitly contained in these projection terms, and thus the same two–term bound (39) holds for incremental SVD updates.

9. Incremental SVD

We utilize Brand's incremental SVD method [8] as the foundation of our proposed online CUR-type algorithms. Given the rank-k SVD of accumulated data at step t as $\mathbf{A}_t = \mathbf{W}_t \mathbf{S}_t \mathbf{V}_t^T$, where $\mathbf{W}_t \in \mathbb{R}^{m \times k}$, $\mathbf{S}_t \in \mathbb{R}^{k \times k}$, $\mathbf{V}_t \in \mathbb{R}^{n_{1:t} \times k}$, and new batch $\mathbf{B}_t \in \mathbb{R}^{m \times n_t}$, then, SVD of $\mathbf{A}_{t+1} \in \mathbb{R}^{m \times n_{1:t}}$ is

$$\mathbf{A}_{t+1} = [\mathbf{A}_t, \mathbf{B}_t] = \mathbf{W}_{t+1}\mathbf{S}_{t+1}\mathbf{V}_{t+1}^T, \tag{40}$$

Brand's algorithm computes the SVD of $\mathbf{A}_{t+1}$ through the following steps in Algorithm 8. It has computational complexity $O(mk\,n_t + k^2 n_t + k^3)$, where k is the target rank and b_{t+1} is the batch size. The complexity breaks down as follows. Step 1 (orthogonalization) requires $O(mk \cdot b_{t+1})$. Step 2 (QR decomposition) requires $O(mk^2 + k^2 b_{t+1})$. Step 3 (matrix construction) requires $O(k^2)$. Step 4 (SVD of K) requires $O((k + b_{t+1})^3)$, which simplifies to $O(k^3)$ when $b_{t+1} \ll k$. Finally, Step 5 (factor updates) requires $O(mk^2 + n_{t+1}k^2)$. The dominant terms give the stated complexity.

10. Experimental Results

10.1. Generating Synthetic Datasets

We conducted a thorough evaluation of the proposed Incremental CUR methods using synthetic datasets that encompass low-rank matrices with additive Gaussian

Algorithm 8 Incremental SVD Update (IncSVD)

Require: $\mathbf{W}_t \in \mathbb{R}^{m\times k}$, $\mathbf{S}_t \in \mathbb{R}^{k\times k}$, $\mathbf{V}_t \in \mathbb{R}^{n_{1:t-1}\times k}$, $\mathbf{B}_t \in \mathbb{R}^{m\times n_t}$

Step 1: Orthogonalize new columns

$\mathbf{P} = \mathbf{B}_t - \mathbf{W}_t(\mathbf{W}_t^T\mathbf{B}_t)$.

Step 2: QR decomposition of residual

$\mathbf{P} = \mathbf{Q}\mathbf{R}_{QR}$ (economy QR)

Step 3: Form augmented matrix

$$\mathbf{K} = \begin{bmatrix} \mathbf{S}_t & \mathbf{W}_t^T\mathbf{B}_t \\ \mathbf{0}_{r'\times k} & \mathbf{R}_{QR} \end{bmatrix}.$$

Step 4: SVD of augmented system

$\mathbf{K}$ as $\mathbf{K} = \tilde{\mathbf{W}}\tilde{\mathbf{S}}\tilde{\mathbf{V}}^T$

Step 5: Update factors

$\mathbf{W}_{t+1} = [\mathbf{W}_t, \mathbf{Q}]\tilde{\mathbf{W}}_k \in \mathbb{R}^{m\times k}$

$\mathbf{S}_{t+1} = \tilde{\mathbf{S}}_k \in \mathbb{R}^{k\times k}$

$$\mathbf{V}_{t+1} = \begin{bmatrix} \mathbf{V}_t & \mathbf{0}_{n_{1:t-1}\times n_t} \\ \mathbf{0}_{n_t\times k} & \mathbf{I}_{n_t} \end{bmatrix} \tilde{\mathbf{V}}$$

Step 6: Truncate to rank k

Keep only the top k singular values and corresponding vectors

return $\mathbf{W}_{t+1}, \mathbf{S}_{t+1}, \mathbf{V}_{t+1}$

noise, exponential decay patterns, and sparse and block-structured matrices, as follows

Low-Rank Gaussian Matrices: Low-rank matrices with additive Gaussian noise were constructed as:$\mathbf{A} = \mathbf{W}\mathbf{V} + \sigma\boldsymbol{\epsilon}$, where $\mathbf{W} \in \mathbb{R}^{m\times r}$ and $\mathbf{V} \in \mathbb{R}^{r\times n}$ are random Gaussian matrices with rank $r = 10$, $\boldsymbol{\epsilon} \sim \mathcal{N}(0,1)$ is white Gaussian noise, and $\sigma = 0.01$ is the noise level.

Exponential Decay Patterns: It model spatially correlated data as: $\mathbf{A}_{i,j} = \exp(-\lambda\sqrt{(i-1)^2 + (j-1)^2}) + \sigma\epsilon_{i,j}$, where $\lambda = 0.1$ is the decay parameter, and $\epsilon_{i,j} \sim \mathcal{N}(0,1)$ represents additive noise with $\sigma = 0.01$.

Random Sparse Matrices: Sparse matrices were generated using: $\mathbf{A} = \text{sprandn}(m, n, \rho) + \sigma\boldsymbol{\epsilon}$, where $\rho = 0.1$ is the sparsity density (10% non-zero elements), and the resulting sparse matrix is densified with additive Gaussian noise.

Block-Structured Matrices: used to simulate data with natural clustering,

$$\mathbf{A} = \begin{bmatrix} \mathbf{B}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{B}_p \end{bmatrix} + \sigma\boldsymbol{\epsilon}$$

where each diagonal block $\mathbf{B}_i \in \mathbb{R}^{m_i\times n_i}$ contains random Gaussian entries, and $p = 3$ blocks were used in both row and column dimensions.

10.2. Results on Synthetic Datasets

The box plot comparisons across four generated matrices are shown in Fig. 3. It demonstrates consistent advantages of the proposed incremental methods. Error distributions show that Inc-LS and Inc-DEIM achieve comparable accuracy to batch methods across all datasets, with tightly clustered medians and minimal variance. For Low-Rank Gaussian data, Inc-DEIM exhibits superior stability with fewer outliers than batch CUR variants. Our incremental methods outperform the batch CUR method and rank-k SVD in terms of running time across various datasets. Incremental methods use only 5–15 MB with minimal variance, versus 50–150 MB for batch CUR. SVD is worst at 100–250 MB with high variability. A 10–20× memory edge makes incremental methods ideal for resource-limited streaming.

10.3. Experimental Results on Real-World Datasets

Figure 4 illustrates the evolution of error as the number of batches increases. Our Inc-DEIM is showing rapid error reduction in early batches (e.g., Reviews, BrainQ, YaleB) and maintains lower error throughout the incremental process. Inc-LS demonstrates consistent performance across all datasets, particularly excelling on USPS, Coil-100, and Traffic. Incremental methods achieve comparable or superior accuracy to batch methods when processing data sequentially, demonstrating their effectiveness for streaming and large-scale applications. Figure 5 shows the error distribution across nine real-world datasets, comparing our proposed incremental methods (Inc-LS and Inc-DEIM) with baseline CUR batch methods and the optimal Rank-k SVD approximation. The boxplots demonstrate that Inc-DEIM consistently achieves the lowest median error across most datasets, including USPS, Traffic, StreamVel, Mnist, Jester, and BrainQ, often approaching or matching the performance of Rank-k SVD. Inc-LS also performs competitively, particularly on USPS, Coil-100, and YaleB datasets. Notably, both incremental methods significantly outperform the batch CUR variants (CUR-LS, CUR-DEIM, CUR-QDEIM) in terms of median error and variance. Figure 6 compares computational time across nine datasets, demonstrating the significant efficiency gains of our proposed incremental methods. It shows that Inc-LS and Inc-DEIM achieve the fastest execution times across all datasets, while maintaining a near-constant, low computational cost as the batch size increases, compared to batch CUR methods. The batch methods show quadratic or superlinear growth, particularly evident in the Coil100, MNIST, and Jester datasets.

Figure 7 illustrates the runtime distribution via boxplots on a logarithmic scale. Inc-DEIM consistently achieves 1-2 orders of magnitude speedup over batch CUR methods across most datasets, with speedups exceeding 100× on the Reviews, Coil100, BrainQ, and YaleB datasets. Inc-LS achieves similar speedups of

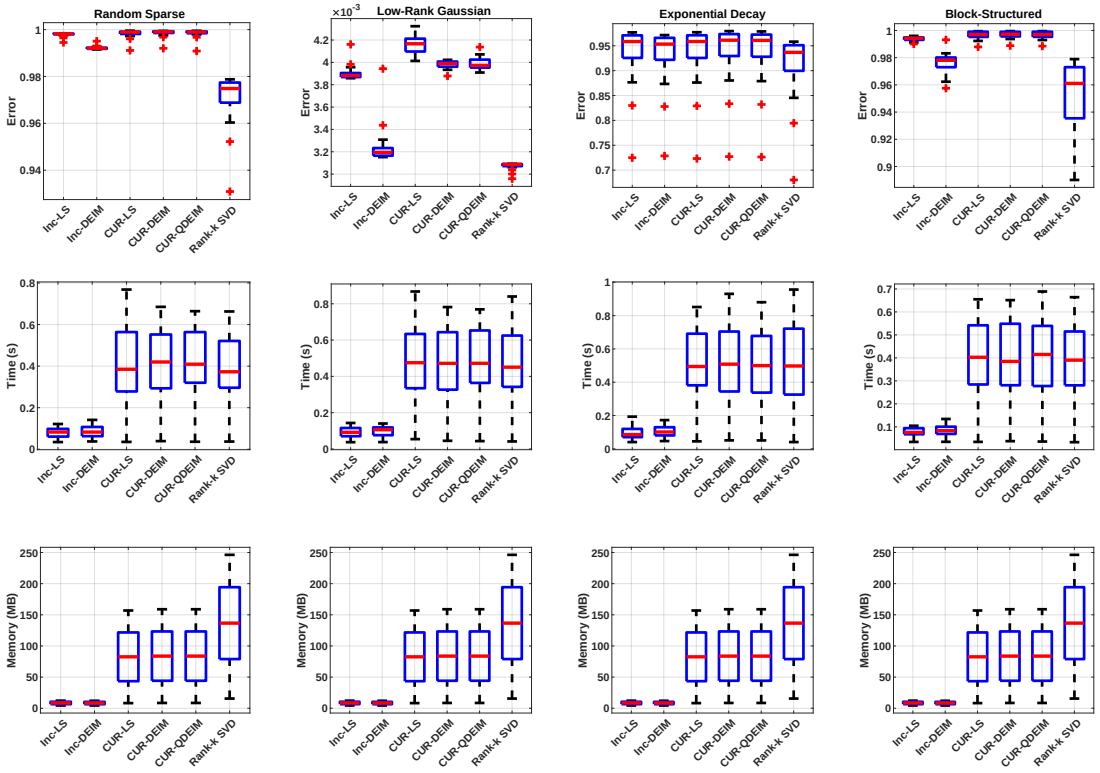

Figure 3. Comparison across four generated matrix structures, illustrating the consistent advantages of the proposed incremental methods.

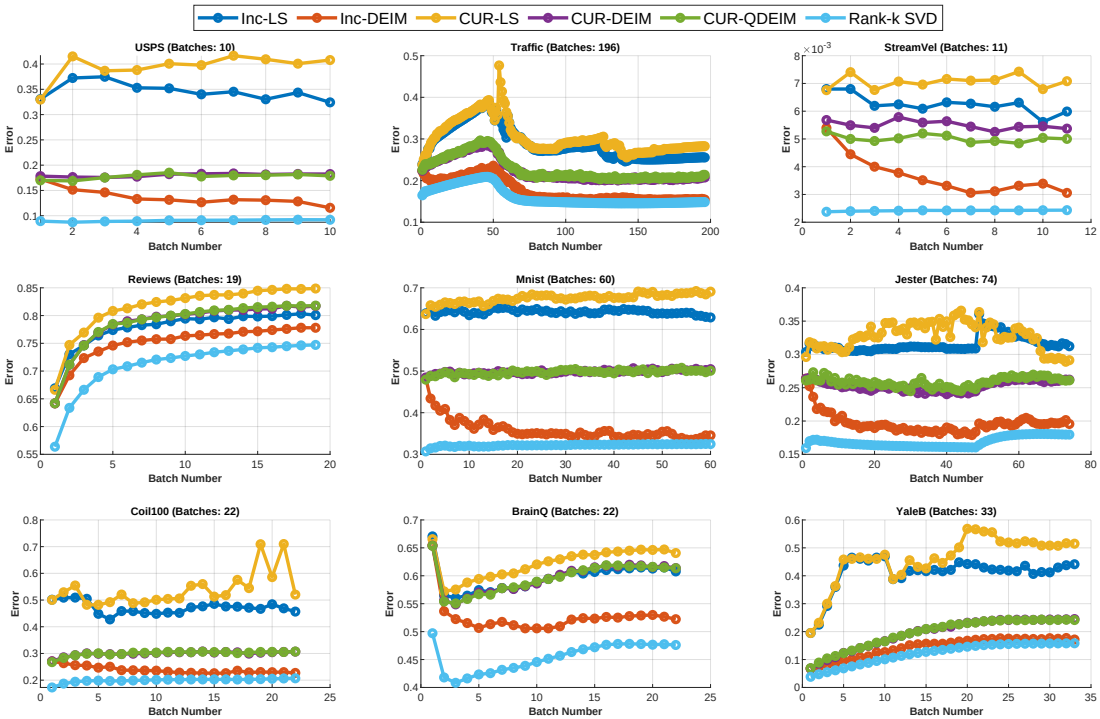

Figure 4. Approximation error with increasing the number of batches across nine real-world datasets, comparing the proposed incremental methods (Inc-LS and Inc-DEIM) with baseline CUR batch methods (CUR-LS, CUR-DEIM, CUR-QDEIM) and the optimal Rank-k SVD approximation.

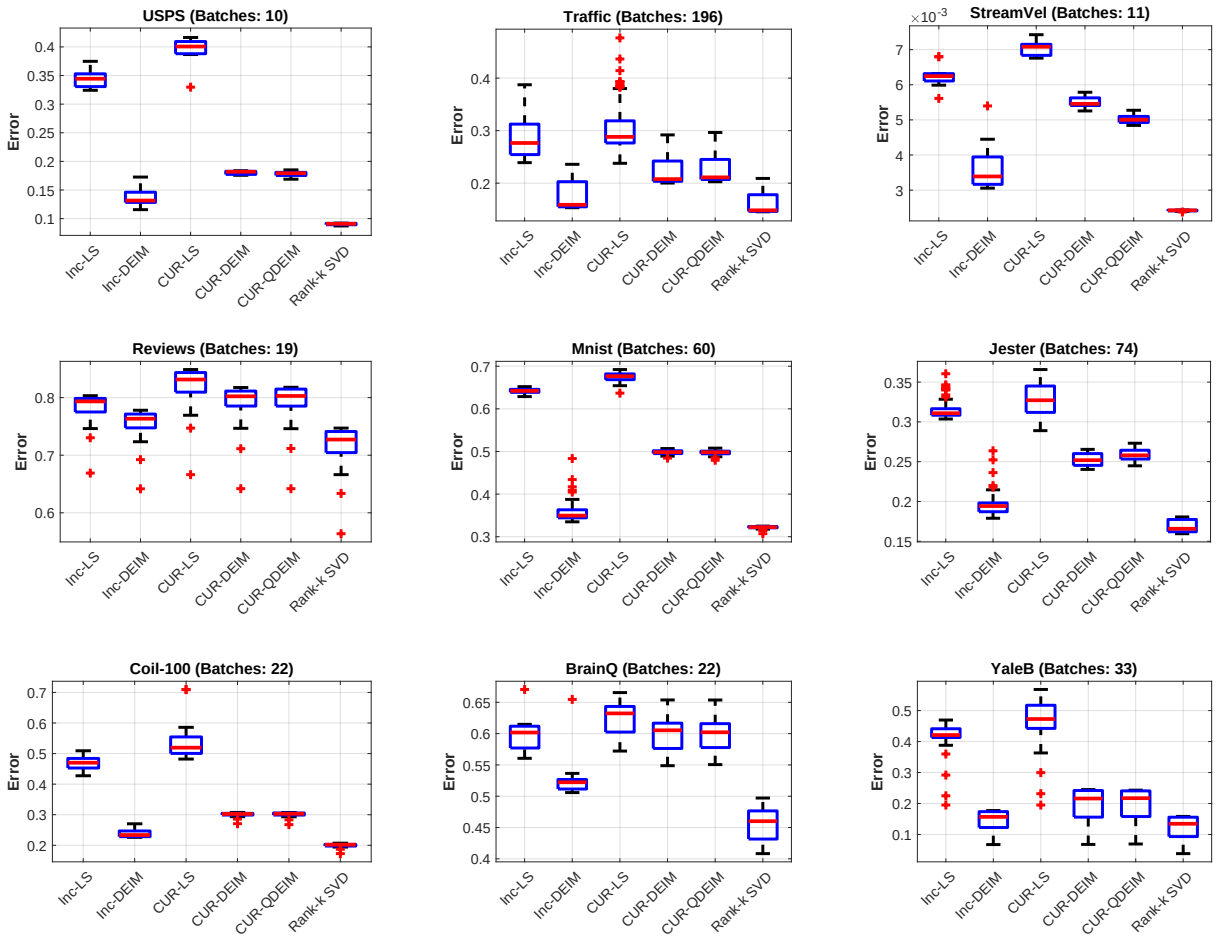

Figure 5. Box plot of error evolution over incremental updates across nine real-world datasets, comparing the proposed incremental methods with baseline CUR batch methods and the optimal Rank-k approximation

10-100× compared to CUR-LS, CUR-DEIM, and CUR-QDEIM. Even Rank-k SVD, despite being optimized, requires significantly more computation time than our incremental approaches. These results demonstrate that our proposed methods achieve substantial computational savings while maintaining competitive accuracy.

Figure 8 demonstrates the memory consumption patterns across nine datasets as batch processing progresses. It reveals that Inc-DEIM maintains the lowest and most stable memory footprint across all datasets, with nearly constant memory usage regardless of the number of batches processed. Inc-LS also exhibits excellent memory efficiency, showing only modest linear growth, in contrast to the steep increases observed in batch methods. Notably, Rank-k SVD and CUR-QDEIM exhibit the highest memory consumption, with a dramatic growth evident in Traffic (reaching 5,000 MB), Reviews (2,000 MB), BrainQ (1,800 MB), and YaleB (1,900 MB).

Figure 9 presents the memory usage distribution via boxplots, highlighting the substantial memory savings achieved by our incremental methods. Inc-LS and Inc-DEIM demonstrate huge memory reductions compared to batch CUR methods across all datasets. On large-scale datasets, such as Traffic and Reviews, incremental methods consume under 200 MB, while batch methods require 2,000-5,000 MB. These results confirm that our proposed incremental approaches achieve substantial memory savings of up to 95% while maintaining competitive accuracy and superior computational efficiency.

10.4. Experimental Results on Video Sequences

We use YUV video sequences[3] (*akiyo-qcif*, *foreman-qcif*, *news-qcif*). For each video sequence with f frames of size $h \times w$, we form $\mathbf{A} = \begin{bmatrix} f_1(:) & f_2(:) & \cdots & f_f(:) \end{bmatrix} \in \mathbb{R}^{hw \times f}$, where each column is a vectorized frame $f_i(:) \in \mathbb{R}^{hw}$. The parameters are $k = 15$, $c = 15$, and $r = 15$, and the batch size is set to 1, allowing for the processing

[3] http://trace.eas.asu.edu/yuv/

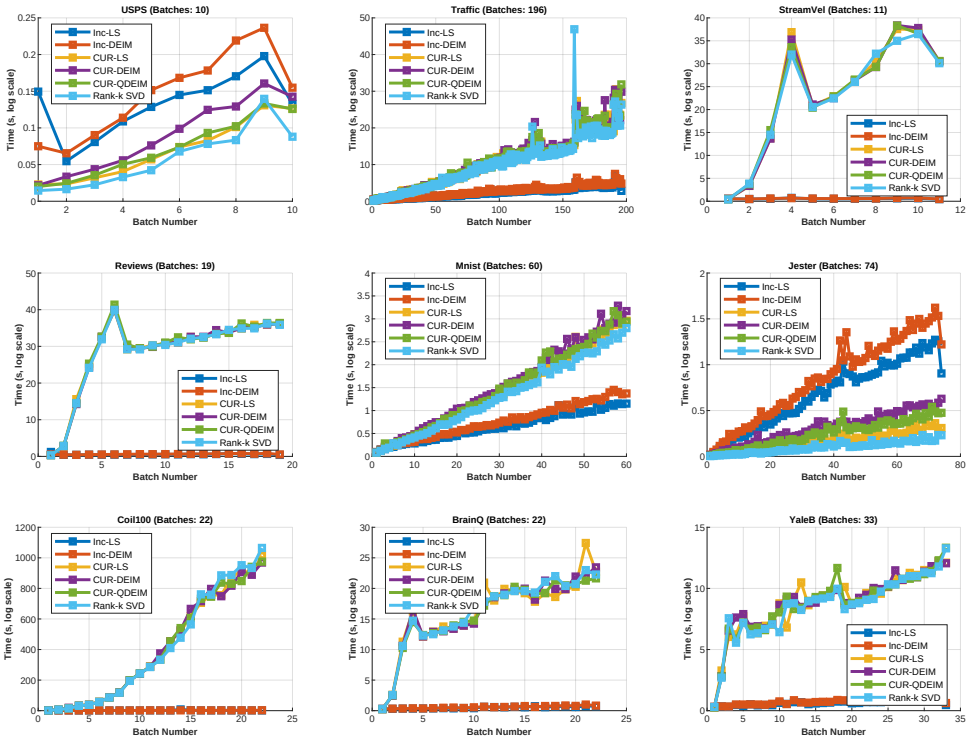

Figure 6. Running time across nine real-world datasets, comparing the proposed incremental methods with baseline CUR batch methods and the optimal Rank-k approximation.

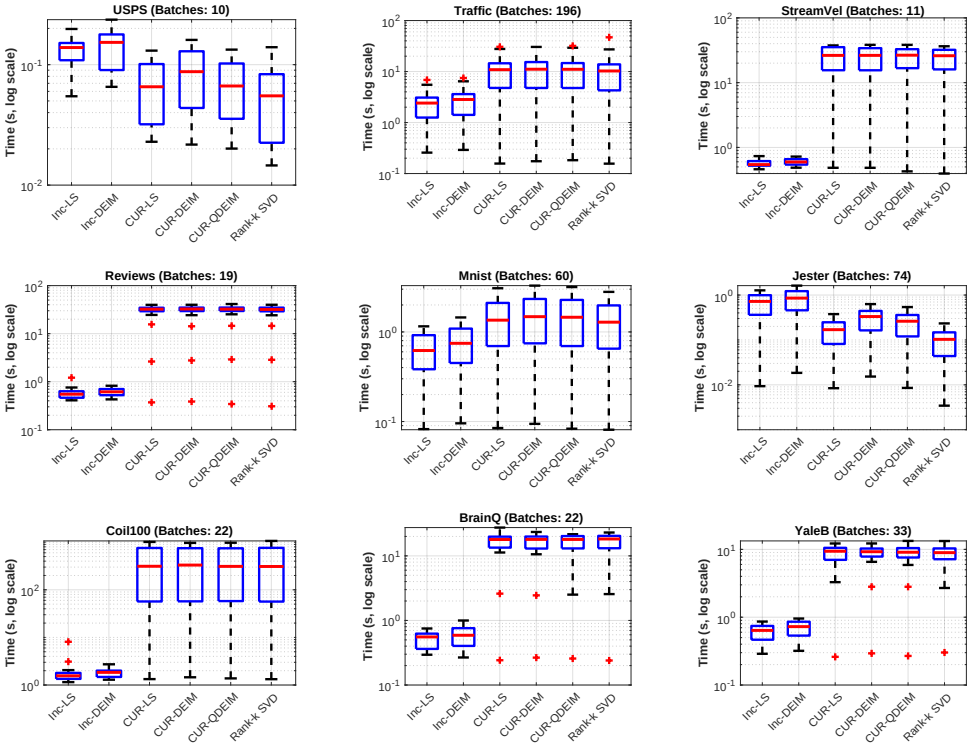

Figure 7. Box plot of running time over incremental updates across nine real-world datasets, comparing the proposed incremental methods with baseline CUR batch methods and the optimal Rank-k approximation

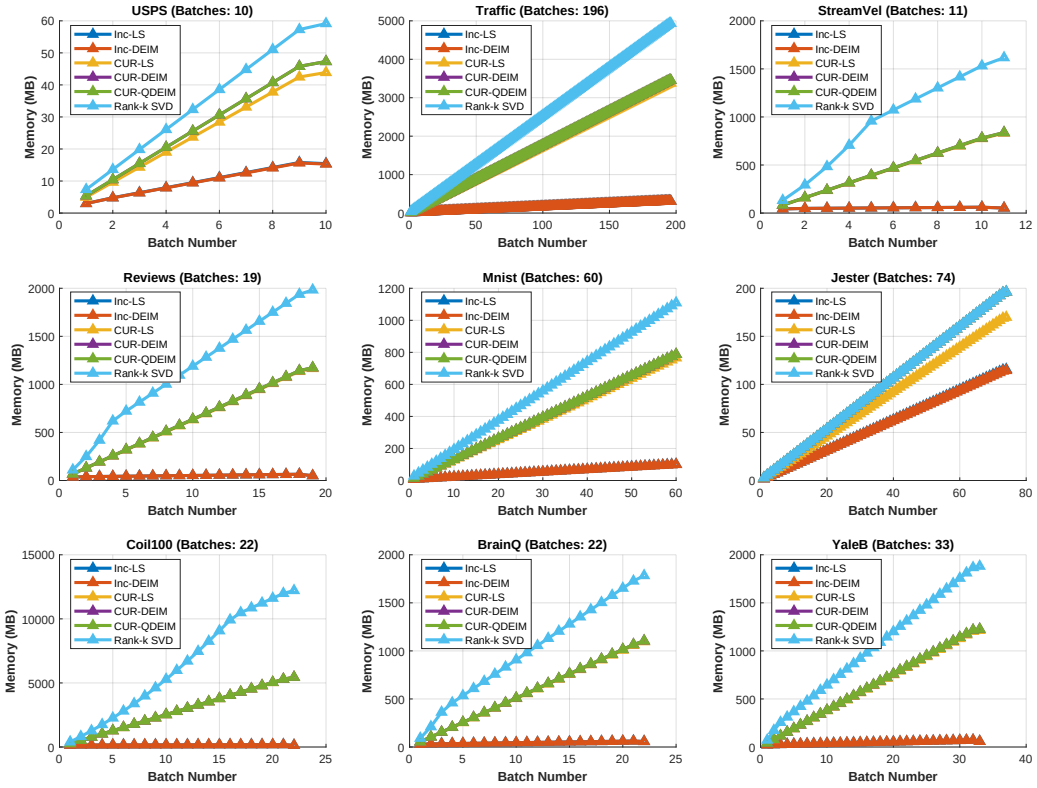

Figure 8. Memory usage (in MB) across nine real-world datasets, comparing the proposed incremental methods with baseline CUR batch methods and the optimal Rank-k approximation.

of a single frame (column) at each step. Fig. 10 shows reconstructed frames of four video sequences using our methods versus baseline algorithms. Our incremental methods, especially Inc-DEIM, showed a strong balance of performance and efficiency, generally outperforming batch CUR methods and performing comparably to optimal low-rank approximation with k-SVD in visualizing frames.

10.5. Parameter Analysis

To evaluate the proposed incremental CUR methods and examine the effects of key parameters, we generated a synthetic low-rank dataset under a controlled configuration. The data matrix $\mathbf{A} \in \mathbb{R}^{1000\times3000}$ was constructed as

$$\mathbf{A} = \mathbf{UV} + \sigma\mathcal{N},$$

where $\mathbf{U} \in \mathbb{R}^{1000\times15}$ and $\mathbf{V} \in \mathbb{R}^{15\times3000}$ are random matrices with entries drawn from a standard normal distribution, and $\mathcal{N}$ represents additive Gaussian noise with variance $\sigma = 0.01$. This configuration ensures that the data possesses an approximate rank of 15 while maintaining a realistic level of noise. For the parameter studies, two sets of experiments were conducted: (1) by varying the SVD rank (k), and the number of selected columns (c) and rows (r) simultaneously with $k = c = r \in \{10, 20, 30, 40, 50\}$ using a fixed batch size of 50, and (2) by fixing $k = c = r = 20$ and varying the batch size, representing the number of incoming columns processed at each incremental step, with batch size $\in \{100, 200, 300, 400, 500\}$. The total number of columns was kept constant at 3000 in all cases, and the matrix was processed in a streaming manner to emulate incremental data arrival. At each setting, three performance metrics were recorded: (i) the cumulative computation time (in seconds), (ii) the average memory usage over all iterations (in MB), and (iii) the final relative reconstruction error. These metrics collectively evaluate the efficiency, scalability, and accuracy of the proposed incremental methods in comparison to their batch counterparts.

Table 6 investigates the joint effect of the SVD rank (k) and the number of selected columns (c) and rows (r). The results show that the proposed incremental meth-

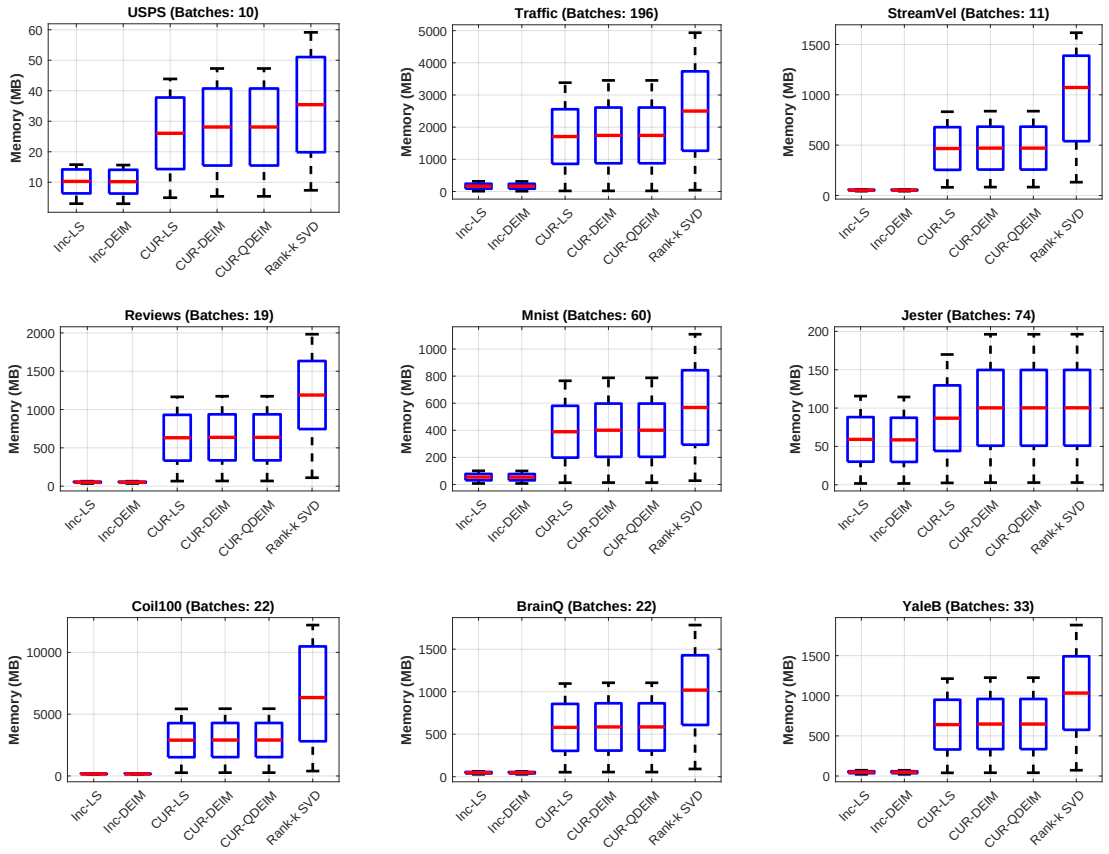

Figure 9. Box plot of memory usage (in MB) over incremental updates across nine real-world datasets, comparing the proposed incremental methods with baseline CUR batch methods and the optimal Rank-k approximation

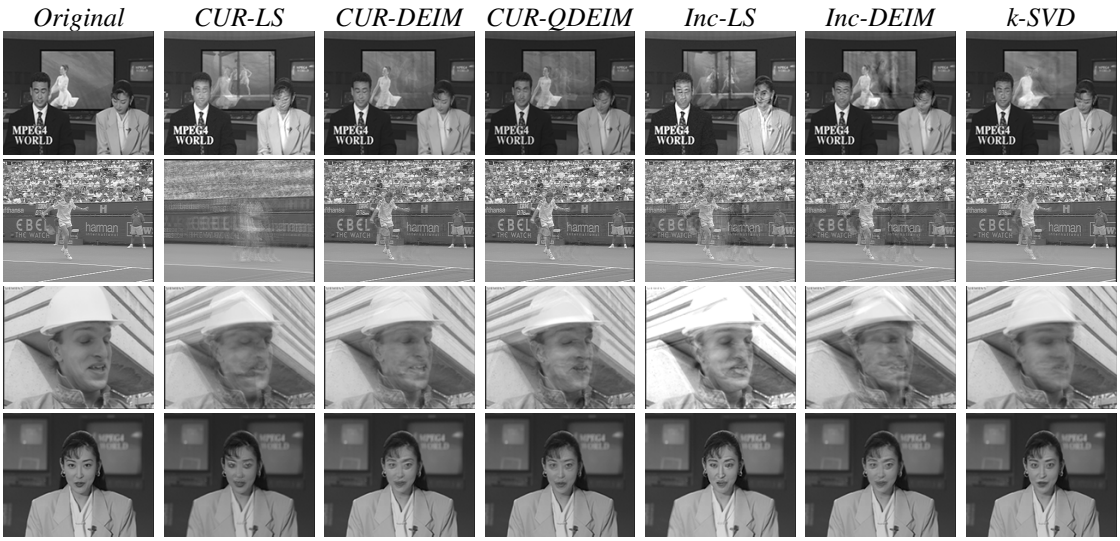

Figure 10. The reconstructed frames from the four video sequences using our proposed methods and baseline algorithms.

ods (Inc-LS and Inc-DEIM) consistently outperform the batch CUR variants (CUR-LS, CUR-DEIM, CUR-QDEIM) in both computational efficiency and memory usage, while maintaining or improving reconstruction accuracy. Specifically, the incremental methods achieve approximately 10 times lower runtime and 6–7 times

Table 6. Parameter Study: SVD rank (k), Number of selected columns (c) and rows (r), with batch size is 50. The cumulative time (in seconds), the average memory usage over iterations (in MB), and the final relative error are reported.

$k/c/r$	CUR-LS			CUR-DEIM			CUR-QDEIM			Inc-LS			Inc-DEIM		
	Time	**Error**	**Memory**	**Time**	**Error**	**Memory**	**Time**	**Error**	**Memory**	**Time**	**Error**	**Memory**	**Time**	**Error**	**Memory**
10	11.102	7.51e-01	23.7	10.738	6.55e-01	23.9	10.885	6.72e-01	23.9	0.530	6.61e-01	1.2	0.452	5.59e-01	1.1
20	11.332	4.81e-03	24.1	11.553	4.18e-03	24.4	11.360	3.94e-03	24.4	0.452	3.67e-03	1.9	0.523	2.63e-03	1.9
30	11.878	3.67e-03	24.5	11.493	3.57e-03	25.0	11.640	3.52e-03	25.0	0.600	3.29e-03	2.7	0.841	2.58e-03	2.7
40	11.233	3.27e-03	24.9	11.413	3.28e-03	25.6	10.889	3.35e-03	25.6	0.683	3.07e-03	3.5	1.114	2.53e-03	3.5
50	11.800	3.14e-03	25.3	12.581	3.14e-03	26.2	12.160	3.15e-03	26.2	1.088	2.97e-03	4.3	1.917	2.51e-03	4.2

Table 7. Parameter Study: Batch Size, where $k = 20$, $c = 20$, $r = 20$. The cumulative time (in seconds), the average memory usage over iterations (in MB), and the final relative error are reported.

Batch Size	CUR-LS			CUR-DEIM			CUR-QDEIM			Inc-LS			Inc-DEIM		
	Time	**Error**	**Memory**	**Time**	**Error**	**Memory**	**Time**	**Error**	**Memory**	**Time**	**Error**	**Memory**	**Time**	**Error**	**Memory**
100	5.688	5.08e-03	24.5	5.708	4.12e-03	24.8	5.716	3.97e-03	24.8	0.266	4.34e-03	2.3	0.317	2.61e-03	2.3
200	2.838	5.08e-03	25.2	2.850	4.12e-03	25.6	2.839	3.97e-03	25.6	0.276	4.06e-03	3.1	0.280	2.66e-03	3.0
300	1.999	5.08e-03	26.0	2.046	4.12e-03	26.4	1.985	3.97e-03	26.4	0.303	4.61e-03	3.8	0.305	2.69e-03	3.8
400	1.822	5.08e-03	28.0	1.770	4.12e-03	28.4	1.782	3.97e-03	28.4	0.334	4.15e-03	4.4	0.386	2.66e-03	4.4
500	1.310	5.08e-03	27.6	1.305	4.12e-03	28.0	1.283	3.97e-03	28.0	0.376	4.43e-03	5.3	0.373	2.81e-03	5.3

lower memory consumption across all tested parameter settings. As k, c, and r increase simultaneously, all methods exhibit a clear reduction in relative error, reflecting improved low-rank approximations. Among the tested approaches, Inc-DEIM yields the best overall performance, achieving the lowest reconstruction error at all parameter settings while retaining significant efficiency benefits. For example, at $k = c = r = 50$, Inc-DEIM attains an error of 2.51×10^{-3} with a runtime of only 1.9 s, compared to over 12 s for the corresponding batch method. Overall, these findings confirm that jointly increasing k, c, and r enhances approximation quality, while the incremental CUR formulations provide substantial computational and memory advantages, making them highly effective for large-scale or dynamically updated matrix problems.

Table 7 examines how varying the incremental update size (batch size) affects performance when the total number of columns is fixed at 3000, with $k = c = r = 20$. The results show clear trade-offs between computational efficiency, memory usage, and accuracy for both the batch and incremental CUR methods. As the batch size increases, the runtime of both batch and incremental methods decreases, since fewer total update steps are required to process the same data. For example, the runtime of the incremental DEIM method drops slightly from 0.32s at a batch size of 100 to 0.37s at 500, while batch CUR methods show a much larger reduction—from roughly 5.7s to 1.3s. However, even at the largest batch size, both incremental variants remain an order of magnitude faster than all batch counterparts. In terms of accuracy, the relative errors remain stable across different batch sizes, indicating that the incremental methods are robust to the choice of update size. Among them, Inc-DEIM consistently achieves the lowest reconstruction error (2.6×10^{-3}), matching the accuracy of batch DEIM while maintaining its computational advantage. Memory usage increases gradually with batch size for both incremental methods (from about 2MB to 5MB), reflecting the larger amount of data processed per update. Still, this remains approximately 5–6 times lower in memory consumption compared to batch CUR methods (25˘28MB). The incremental methods, especially Inc-DEIM, achieve the best overall balance, offering stable accuracy, minimal memory overhead, and significantly faster execution across all batch configurations.