FURTHER TESTING AND COMPARISONS OF THE HYBRID-(L1, L2) SAMPLING ALGORITHM WITH L1 AND L2 SAMPLING

Yogesh Agrawal, Fernando Spadea

ABSTRACT

In this paper, we analyze and test Abhisek Kundu's matrix sparsification algorithm Kundu (2015). This algorithm attempts to combine the benefits of l_1 and l_2 sampling to create an optimal algorithm. We consider the theory behind this and analyze how well it performs with a range of input parameters similar to the original paper, so we can compare our results to Kundu's. However, we use different datasets so as to find new and interesting results. In our experiments, we find that this hybrid algorithm does not perform better than l_1 or l_2 sampling, so it may not be as widely applicable as it was originally presented to be.

1 INTRODUCTION

Matrices are a very useful format for storing and working with data, but they can often become too large or dense to efficiently work with. Often, it is much more efficient to work with sparse matrices, or those with mostly 0 values, but most matrices are not going to be that clean. Thus, it's useful to be able to transform these matrices into sparse matrices while losing as little of the information they hold as possible. There exist several different algorithms to attempt to do just this. The main goal of these algorithms is generally to minimize the norm of the difference between the original and sparse matrix. This paper will look at Abhisek Kundu's algorithm (Kundu, 2015).

To perform element-wise sparsification, various sampling techniques have been used. In sampling, we assign probabilities to each element of a matrix, and then use those probabilities to randomly select the desired number of elements to keep from said matrix. With $l1$ sampling, we assign these probabilities by taking the absolute value of an entry and dividing it by the 1-norm of the matrix (Achlioptas et al., 2013b). As for $l2$ sampling, we use the entry squared dividing by the 2-norm of the matrix instead (Drineas & Zouzias, 2011b). Element-wise sparsification was initially pioneered by Achlioptas (2001) to be done by $l2$ sampling technique. However, they realised $l2$ sampling is not good to prove accurate bounds for $||\mathbf{A} - \tilde{\mathbf{A}}||_2$.Achlioptas & McSherry (2007) found that small entries of the matrix needed to be sampled with probabilities that depended only on absolute values because if a small element is sampled and rescaled using $l2$ sampling, this would result in huge entry in $\tilde{\mathbf{A}}$ because of the rescaling and cause the variance of $l2$ sampling to be very high. $l1$ sampling of these small entries rectified this issue. (Drineas & Zouzias, 2011a) proposed sparsification algorithm that bypassed the need for $l1$ sampling by zeroing out small elements whose absolute values were below a threshold. However, Abhisek Kundu claims to have created an improved algorithm by taking advantage of both $l1$ and $l2$ sampling (Kundu, 2015). The claim made is that their algorithm retains the good properties of $l2$ sampling that bias towards the data elements in the presence of small noise, while regularizing smaller entries using $l1$ sampling. The author tested his algorithm with a few different data sets to compare it with just using $l1$ or $l2$ sampling (Kundu, 2015). Nevertheless, our paper aims to further test his algorithm against $l1$ and $l2$ sampling on several more distinct data sets.

The original paper tested their algorithm on Synthetic, TechTC, Handwritten Digit, and Stock data. However, the handwritten digit data testing was limited to 6, 9, and 1, so we will be testing the algorithm on the MNIST handwritten digit data set with digits 0-9 (Deng, 2012). We will also test the algorithm on a license plate dataset [1]. Lastly, we will test the algorithm on a data set of historical football match data [2]. These additional tests will help support or devalue the algorithm presented by Abhisek Kundu. Even if this algorithm proves to be efficient, it is important to make sure that it actually does a good job of representing the original matrix or else it will fail.

2 INTRODUCTION TO HYBRID L1-L2 SAMPLING TECHNIQUE

The hybrid L1-L2 sampling technique for matrix sparsification is an approach that combines the benefits of L1 sampling and L2 sampling to achieve a more efficient and accurate sparsification process. Matrix sparsification is the process of re-

[1] https://www.kaggle.com/datasets/andrewmvd/car-plate-detection

[2] https://www.kaggle.com/datasets/tobycrabtree/nfl-scores-and-betting-data

ducing the number of non-zero elements in a matrix while preserving its essential properties, such as spectral or structural information. This is particularly useful in large-scale data analysis, machine learning, and numerical optimization, where working with dense matrices can be computationally expensive.

Element-wise sparsification of a matrix using L1 sampling is a technique used to create a sparse representation of a matrix by retaining only a small fraction of its original elements. This can be useful for reducing the storage and computational requirements when working with large matrices, particularly in machine learning and data analysis applications. The probability of each element to be selected is given by $\frac{|\mathbf{A}_{ij}|}{||\mathbf{A}||_1}$ which is basically the ratio of the absolute value of that element and the l_1 norm of the matrix. By using L1 sampling for element-wise sparsification, it is possible to create a sparse representation of the input data while preserving the most significant elements. L1 sampling can be effective in preserving the structure of the input matrix but may have difficulty preserving spectral properties.

L2 sampling, on the other hand, considers the frobenius norm of the matrix to assign the probability of each element being selected. The probabilities are assigned using $\frac{\mathbf{A}_i j^2}{||\mathbf{A}||_F^{2\,2}}$ which is the ratio of squared value of that element and the Frobenius norm of the matrix. L2 sampling is better at preserving spectral properties but may struggle to preserve the structural properties of the input matrix.

The hybrid L1-L2 sampling technique combines these two approaches by taking into account both L1 and L2-norms. This is done by calculating a combined score for each element based on a weighted combination of their L1 and L2-norms. The weight, α, can be adjusted to favor either L1 or L2-norms, depending on the desired balance between preserving structural and spectral properties.

To understand the difference between $l1$ and $l2$ sampling technique, it is key to understand the difference between spectral and structural properties of matrices.

Difference between Spectral and Structural Properties of a Matrix

Spectral and structural properties of a matrix are two distinct aspects that describe its characteristics and behavior. Here, we explain the main differences between these two types of properties:

Spectral properties:

Spectral properties of a matrix are related to its eigenvalues, eigenvectors, and singular values. These properties provide insights into the matrix's behavior under linear transformations and can be used to analyze various aspects of the matrix. Some important spectral properties include:

- Eigenvalues and eigenvectors: Eigenvalues are scalar values associated with a matrix that reveal information about the linear transformation it represents. Eigenvectors are the corresponding vectors that remain unchanged in direction (but may change in length) when multiplied by the matrix. The set of all eigenvalues is called the spectrum of the matrix.
- Singular values: Singular values are the square roots of the non-negative eigenvalues of the matrix product of the original matrix and its transpose (for real matrices) or conjugate transpose (for complex matrices). Singular value decomposition (SVD) is a method to factorize a matrix into three matrices, which can reveal important information about the matrix, such as its rank, range, and null space.
- Spectral radius: The spectral radius is the maximum absolute value of the eigenvalues of a matrix. It provides information about the convergence properties of iterative methods and the stability of dynamical systems.
- Condition number: The condition number of a matrix is the ratio of its largest singular value to its smallest singular value. It provides information about the sensitivity of a linear system to changes in the input data.

Structural properties:

Structural properties of a matrix are related to its shape, pattern of non-zero elements, and other characteristics that describe its structure. These properties can be used to analyze the matrix's symmetry, diagonal dominance, and features that impact the efficiency of algorithms and storage requirements. Some important structural properties include:

- Symmetry: A matrix is symmetric if its transpose is equal to itself ($A = A^T$). Symmetric matrices have useful properties, such as having real eigenvalues and orthogonal eigenvectors, which can be exploited to simplify computations.

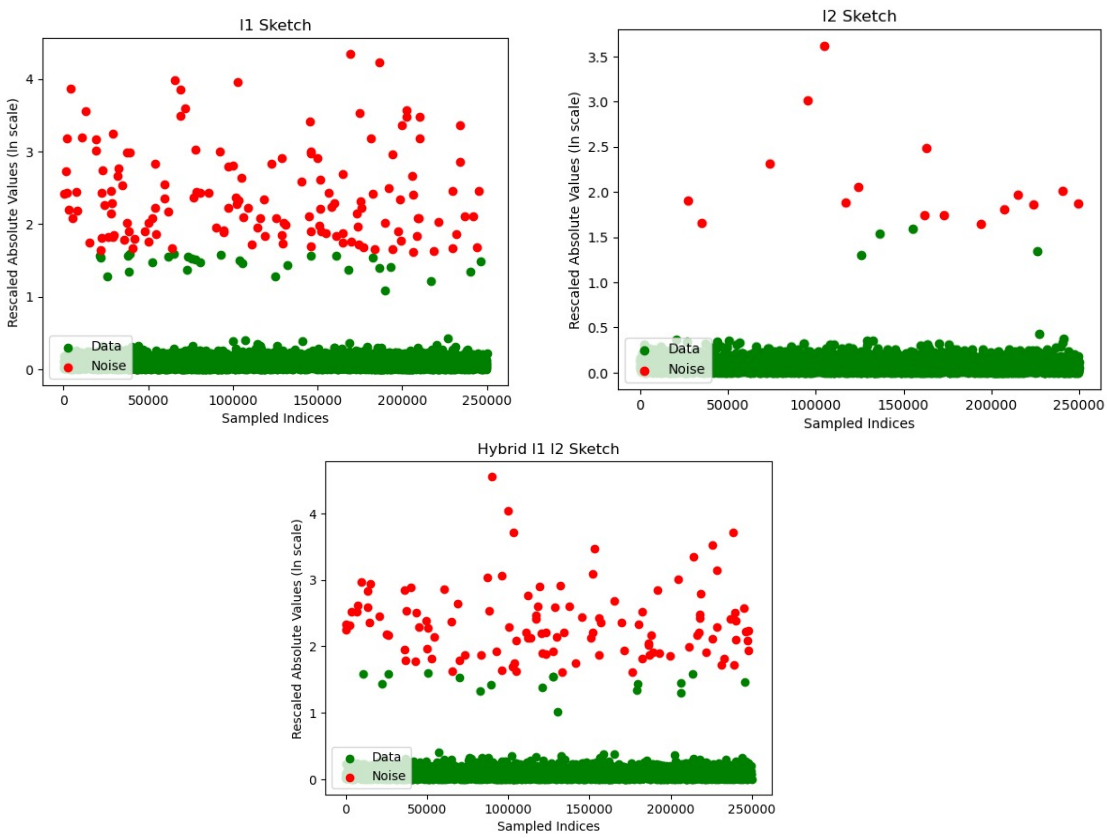


Figure 1: Elements of sparse Sketches$\tilde{\mathbf{A}}$ produced by three sampling techniques. The y-axis plots rescaled absolute values (in $\ln$ scale) of $\tilde{\mathbf{A}}$ corresponding to sampled indices.

- Diagonal dominance: A matrix is diagonally dominant if the absolute value of each diagonal element is greater than or equal to the sum of the absolute values of the other elements in the same row (or column). Diagonally dominant matrices have certain convergence properties that can be utilized in iterative methods.
- Block structure: Some matrices can be divided into smaller submatrices, called blocks. Exploiting block structure can lead to more efficient algorithms for matrix operations, such as multiplication and inversion.

To visually compare the three sampling techniques, the author suggests to construct a $500 * 500$ binary data $\mathbf{D}$ and then perturb it by random Gaussian matrix $\mathbf{N}$ whose elements $\mathbf{N}_{ij}$ follow Gaussian distibution with mean 0 and standard deviation 0.1. This perturbed matrix is denoted by $\mathbf{A}_{0.1}$. Then, sample $s = 5000$ indices in i.i.d. trials according to l_1, l_2 and hybrid probabilities and produce a sketch of sparse matrix $\tilde{\mathbf{A}}$. We followed similar method and ran the experiment on $5000 * 5000$ matrix. The resultant sketches produced are given in figure 1. As claimed by the author, we also find that l_1 sampling have controlled variance but most of them are noise. l_2 sampling is biased towards the data elements and small number of sampled noisy elements create large variance due to rescaling whereas hybrid-(l_1, l_2) sampling benefits from bias of l_2 towards data elements , as well as regularization properties of l_1.

In summary, the hybrid L1-L2 sampling technique for matrix sparsification aims to achieve a better balance between preserving the structural and spectral properties of the input matrix by combining the strengths of L1 and L2 sampling techniques. This can result in more accurate and efficient sparsification, which can be particularly beneficial in large-scale data analysis, machine learning, and numerical optimization applications.

3 Advantages and Disadvantages

The hybrid L1-L2 sampling technique for matrix sparsification comes with its own set of advantages and disadvantages.

Advantages:

1. Balanced sparsification: The hybrid approach allows for a more balanced sparsification by considering both the L1 and L2-norms of the rows or columns. This can result in better preservation of both structural and spectral properties of the input matrix.
2. Flexibility: The method offers flexibility in terms of the weights assigned to the L1 and L2-norms. Depending on the specific problem or application, the weights can be adjusted to favor either L1 or L2-norms, allowing the user to emphasize either structural or spectral properties as needed.
3. Improved accuracy: By considering both L1 and L2-norms, the hybrid approach can often achieve more accurate sparsification compared to using either L1 or L2 sampling alone. This can lead to better performance in downstream tasks, such as data analysis, machine learning, and numerical optimization.
4. Scalability: The hybrid L1-L2 sampling technique is applicable to large-scale matrices, making it suitable for problems where computational resources are limited or where working with dense matrices is computationally expensive.

Disadvantages:

1. complexity: The hybrid approach requires the calculation of both L1 and L2-norms, as well as the assignment of weights for their combination. This can result in increased complexity compared to using either L1 or L2 sampling alone.
2. Tuning parameters: Determining the appropriate weights for combining the L1 and L2-norms may require additional tuning or experimentation, which can be time-consuming and may not always guarantee optimal results.
3. Loss of sparsity: Although the hybrid approach aims to preserve the essential properties of the input matrix, it may still result in a loss of sparsity. This can be an issue if the primary goal is to minimize the number of non-zero elements in the resulting sparse matrix.
4. No universal solution: The hybrid L1-L2 sampling technique may not be the best solution for all problems. Depending on the specific properties of the input matrix and the goals of the sparsification process, other techniques or specialized algorithms might be more appropriate.

In conclusion, the hybrid L1-L2 sampling technique offers a flexible and balanced approach to matrix sparsification, but it comes with some trade-offs, such as increased complexity and the need for parameter tuning. The suitability of this approach depends on the specific problem, application, and desired balance between preserving structural and spectral properties.

4 POSSIBLE STRUGGLES

The hybrid L1-L2 sampling technique for matrix sparsification might struggle with certain types of data inputs or problem characteristics:

1. Highly imbalanced data: If the input matrix has extreme differences in the magnitudes of its elements or in the distribution of its non-zero elements, the hybrid approach may struggle to find an appropriate balance between preserving structural and spectral properties. In such cases, the method might require extensive tuning of the weights assigned to L1 and L2-norms.
2. Noise-dominated data: In the presence of significant noise in the input matrix, the hybrid L1-L2 sampling technique may have difficulty distinguishing between informative elements and noise, which could affect the accuracy of the sparsification process.
3. Strongly structured matrices: If the input matrix exhibits strong structural patterns or specific properties, such as block structure, banded structure, or specific types of rank structure, the hybrid L1-L2 sampling technique may not be the most suitable method. In these cases, specialized sparsification techniques tailored to the specific matrix structure might yield better results.
4. Highly sparse matrices: For matrices that are already very sparse, the hybrid L1-L2 sampling technique might not provide significant benefits over other methods or even the original sparse matrix. In these cases, the added complexity of the hybrid approach may not be justified.
5. Data with specific requirements: In some applications, it may be necessary to prioritize either structural or spectral properties of the input matrix to a greater extent than the hybrid L1-L2 sampling technique allows. In

such cases, using L1 or L2 sampling alone or employing other specialized sparsification methods might be more appropriate.

While the hybrid L1-L2 sampling technique can be an effective method for matrix sparsification in many cases, it is not a one-size-fits-all solution. The suitability of this approach for a given problem depends on the characteristics of the input matrix and the specific goals of the sparsification process.

4.1 EXAMPLE

Suppose we have a large block-diagonal matrix, where the matrix is composed of smaller, dense submatrices along the main diagonal, and the off-diagonal elements are zero. Such a matrix can arise in applications like multi-domain physics simulations or multi-block systems.In this case, the hybrid L1-L2 sampling technique might be inefficient due to the following reasons:

1. The matrix structure is already sparse: Since the off-diagonal blocks of the matrix are filled with zeros, the matrix is already sparse. Applying the hybrid L1-L2 sampling technique might not provide significant benefits in terms of sparsity or computational savings.
2. Loss of block structure: The hybrid L1-L2 sampling technique does not specifically take into account the block structure of the input matrix. By sampling rows and columns based on their L1 and L2-norms, the resulting sparse matrix might lose the original block structure, which could be important for downstream applications.
3. Inefficient sampling: The L1 and L2-norms might not provide meaningful information for selecting rows and columns in this case. Since the matrix is already block-diagonal, it would be more efficient to focus on preserving the structure of the diagonal blocks themselves, rather than considering the norms of the rows and columns across the entire matrix.

In this example, it might be more appropriate to use specialized sparsification techniques tailored to the block structure of the matrix, such as block-wise sparsification or hierarchical sampling methods. These methods can better preserve the block structure and potentially provide more accurate and efficient sparsification.

This example is implemented in our experiments using data from **License plate numbers** to test our theory on the limitation of hybrid algorithm on such matrices.

5 METHODOLOGY AND KEY FORMULAE

Let us define the sampling operator $S_\Omega : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ which will extract the elements from our given matrix $\mathbf{A}$. Here Ω is defined as multi-set of sampled indices (i_t, j_t) where t is an integer between 1 and s. Then:

$$S_\Omega = \frac{1}{s}\sum_{t=1}^{s} \frac{\mathbf{A}_{i_t j_t}}{p_{i_t j_t}} \boldsymbol{e}_{i_t} \boldsymbol{e}_{j_t}^T \quad \text{where}(i_t, j_t) \in \Omega$$

In the Element-wise Matrix Sparsification algorithm, S_Ω is the sparse approximation of the input matrix A. It is constructed by sampling a certain number of non-zero elements from the input matrix, while ensuring that the error between A and S_Ω is within a specified bound. The matrix S_Ω has the same dimensions as A, but most of its elements are set to zero.

The construction of S_Ω is done as follows:

1. First, the optimal α^*, the number of non-zero elements s^*, and the probabilities p_{ij} are calculated.

2. The algorithm then samples s^* indices from the input matrix A, with the probability of selecting each element (i, j) being proportional to p_{ij}.

3. For each sampled index (i, j), the algorithm adds the value of the corresponding element in the input matrix A to the sparse approximation S_Ω, divided by the product of the probability p_{ij} and the number of non-zero elements s^*:

$$S_\Omega(i, j) \mathrel{+}= \frac{A(i, j)}{p_{ij} s^*}$$

This procedure ensures that the sparse approximation S_Ω captures the most important information in the original matrix A while having significantly fewer non-zero elements.

5.1 ALGORITHM

Algorithm 1 Element-wise Matrix Sparsification

Require: $A \in \mathbb{R}^{m\times n}$, accuracy parameter $\epsilon > 0$.
1: Set s as in eq. 8.
2: **for** $t = 1 \dots s$ (i.i.d. trials with replacement) **do**
3: Randomly sample pairs of indices $(i_t, j_t) \in [m] \times [n]$ with $P[(i_t, j_t) = (i, j)] = p_{ij}$, where p_{ij} are as in 1, using α as in 7.
4: **end for**
5: **Output(sparse)**: $S_\Omega(A) = \frac{1}{s}\sum_{t=1}^{s} \frac{A_{i_t j_t}}{p_{i_t j_t}} e_{i_t} e_{j_t}^T$.

To sample s elements, we will use algorithm given in Algorithm 1 based on the probability distribution $\{p_{ij}\}_{i,j=1}^{m,n}$ which we calculate using the formula:

$$p_{ij} = \alpha \frac{|\mathbf{A}_{ij}|}{||\mathbf{A}||_1} + (1-\alpha)\frac{\mathbf{A}_{ij}^2}{||\mathbf{A}||_F^2} \quad \text{where} \quad \alpha \in (0, 1] \tag{1}$$

The p_{ij} function calculates the sampling probabilities for each element of matrix $\mathbf{A}$. It is a normalized version of a weighted combination of matrix A's element-wise absolute values and element-wise squares, with the weighting determined by the parameter α.

Lemma of matrix-Bernstein gives proof for the quality of the approximation produced by the algorithm. The theorem goes as follows:
Theorem 1 Let $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\epsilon > 0$ be an accuracy parameter. Let $\mathbf{S}_\Omega$ be the sampling operator defined above and assume we generate multi-set Ω using the above sampling probabilities. Then, with probability at least $1 - \delta$,

$$||\mathbf{S}_\Omega(\mathbf{A}) - \mathbf{A}||_2 \leq \epsilon ||\mathbf{A}||_2 \tag{2}$$

if

$$s \geq \frac{2}{\epsilon^2 ||\mathbf{A}||_2^2} \left(\rho^2(\alpha) + \gamma(\alpha) \in ||\mathbf{A}||_2/3\right) \ln\left(\frac{m+n}{\delta}\right) \tag{3}$$

Here, Matrix-Bernstein inequality gave us the functional norm in (3). To understand the reasoning behind the above formulae, a brief description of the functions involved is as given below:

5.2 FUNCTIONS DESCRIPTION

5.2.1 XI FUNCTION (ξ)

The ξ function computes an intermediate value representing a weighted combination of matrix A's element-wise absolute values and element-wise squares. The weighting is determined by the parameter α.

$$\xi_{ij} = \frac{||\mathbf{A}||_F^2}{\alpha \frac{||\mathbf{A}||_F^2}{|||\mathbf{A}||_{ij}|||\mathbf{A}||_1} + (1-\alpha)} \tag{4}$$

The ξ function represents the trade-off between the absolute value of matrix elements and the squared Frobenius norm of the matrix. This trade-off is important because it helps balance the contribution of individual elements to the overall approximation error. In the context of the algorithm, ξ is a matrix of the same size as the input matrix $\mathbf{A}$. Each element of ξ is calculated using the formula mentioned earlier. The parameter alpha is used to control the trade-off between these two factors. When alpha is closer to 1, the algorithm prioritizes minimizing the sum of absolute values of the errors, and when alpha is closer to 0, it prioritizes minimizing the sum of squared errors.

5.2.2 RHO SQUARED FUNCTION (ρ^2)

The ρ^2 function calculates the difference between the maximum row-wise and column-wise sums of ξ and the square of the smallest singular value ($\sigma_{\min}^2$) of matrix A.

$$\rho^2(\alpha) = \max\left(\max_i \sum_j \xi_{ij}, \max_j \sum_i \xi_{ij}\right) - \sigma^2_{\min} \tag{5}$$

where σ_{min} refers to the smallest singular value of $\mathbf{A}$.
The ρ^2 function computes an error metric that is used to assess the quality of the sparse approximation. In particular, it measures the maximum deviation of the row and column sums of ξ (which is derived from the input matrix $\mathbf{A}$ and parameter alpha) from the minimum singular value squared of the input matrix A. **A lower ρ^2 value indicates a better approximation, as it means that the sparse approximation closely preserves the row and column sums of the input matrix**.

5.2.3 GAMMA FUNCTION (γ)

γ controls the incoherence of the matrix, i.e, larger values of γ makes data more spiky.

$$\gamma(\alpha) = \max_{i,j} \frac{||\mathbf{A}||_1}{\alpha + (1-\alpha)\frac{|\mathbf{A}_{ij}|||\mathbf{A}||_1}{||\mathbf{A}||_F^2}} + ||\mathbf{A}||_2 \tag{6}$$

The gamma function is used to compute an upper bound on the error of the sparse approximation. The gamma function essentially helps to control the error bound of the matrix approximation, ensuring that the sparse approximation does not deviate too much from the original matrix. The goal of the algorithm is to minimize the error bound while creating a sparse representation of the matrix

The algorithm uses these functions to determine the optimal alpha value that minimizes the error bound of the sparse approximation. It then computes the number of non-zero elements that should be in the sparse approximation and constructs the sparse approximation by sampling the matrix elements according to their probability.

Then, to find the optimal α^* and s^* we can solve following optimization problem:

$$\alpha^* = \min_{\alpha \in (0,1]} f(\alpha) = \rho^2(\alpha) + \gamma(\alpha)\epsilon||\mathbf{A}||_2/3 \tag{7}$$

$$s^* = \frac{2}{\epsilon^2||\mathbf{A}||_2^2}\left(\rho^2(\alpha^*) + \gamma(\alpha^*) \in ||\mathbf{A}||_2/3\right)\ln\left(\frac{m+n}{\delta}\right) \tag{8}$$

This formula ensures that the number of non-zero elements in the sparse approximation S_Ω is sufficient to provide an accurate representation of the input matrix $\mathbf{A}$, while keeping the error between the two matrices within the specified bound.

6 PROOF

The proofs in this paper are referenced from (Kundu, 2015).

6.1 THEOREM 1

To prove the above mentioned **Theorem 1** by following the method used by Kundu in their paper. We use the non-commutative matrix valued Bernstein bound of (Recht, 2011) and the proof outline of (Achlioptas et al., 2013a) and (Drineas & Zouzias, 2011a). Rephrasing the matrix Bernstein bound as per our notations, we get:
Lemma 1 [Theorem 3.2 of (Recht, 2011)]
Let $M_1, M_2, \ldots, M_s$ be independent, zero-mean random matrices in $\mathbb{R}^{m\times n}$. Suppose

$$\max_{t\in[s]}\left\{\mathbb{E}(M_t M_t^T)^2, \mathbb{E}(M_t^T M_t)^2\right\} \leq \rho^2$$

and $\|M_t\|_2 \leq \gamma$ for all $t \in [s]$. Then, for any $\varepsilon > 0$,

$$\left\|\frac{1}{s}\sum_{t=1}^{s} M_t\right\|_2 \leq \varepsilon$$

holds, subject to a failure probability at most

$$(m+n)\exp\left(-\frac{s\varepsilon^2}{2\rho^2+\frac{2}{3}\gamma\varepsilon}\right).$$

For all $t \in [s]$ we define the matrix $M_t \in \mathbb{R}^{m\times n}$ as follows:

$$M_t = \frac{A_{it}j_t}{p_{it}j_t}e_{it}e_j^T - A.$$

It now follows that

$$\frac{1}{s}\sum_{t=1}^{s} M_t = \frac{1}{s}\sum_{t=1}^{s}\left(\frac{A_{it}j_t}{p_{it}j_t}e_{it}e_j^T - A\right) = S\Omega(A) - A.$$

We can bound $\|M_t\|_2$ for all $t \in [s]$. We define the following quantity:

$$\lambda = \frac{\|A\|_1 \cdot |A_{ij}|}{\|A\|_F^2}, \quad \text{for } A_{ij} \neq 0.$$

Lemma 2 Using our notation, and using probabilities from the formula 1, for all $t \in [s]$,

$$\|M_t\|_2 \leq \max_{i,j:A_{ij}\neq 0}\left\{\|A\|_1\frac{\alpha+(1-\alpha)\lambda}{\|A\|_F^2} + \|A\|_2\right\}.$$

Proof: Using probabilities of the form (1.2), and because $A_{ij} = 0$ is never sampled,

$$\|M_t\|_2 = \left\|\frac{A_{it}j_t}{p_{it}j_t}e_{it}e_j^T - A\right\|_2 \leq \max_{i,j:A_{ij}\neq 0}\left\{\frac{\alpha\|A\|_1 + (1-\alpha)|A_{ij}|}{\|A\|_F^2}\right\} + \|A\|_2$$

Using (1.8), we obtain the bound. □

Next, we bound the spectral norm of the expectation of $M_t M_t^T$.

Lemma 3 Using our notation , and using probabilities of the form (1.2), for all $t \in [s]$,

$$\mathbb{E}(M_t M_t^T)^2 \leq \|A\|_F^2\frac{\beta_1 - \sigma^2\min(A)}{\|A\|_F^2},$$

where,

$$\beta_1 = \max_i \sum_{j=1}^{n}\left\{\frac{\alpha\|A\|_F^2}{|A_{ij}|\|A\|_1} + (1-\alpha)\right\}^{-1}, \quad \text{for } A_{ij}\neq 0.$$

Proof: Recall that $A = \sum_{i=1}^{m}\sum_{j=1}^{n} A_{ij}e_ie_j^T$ and $M_t = \frac{A_{it}j_t}{p_{it}j_t}e_{it}e_j^T - A$ to derive

$$\begin{aligned}\mathbb{E}[M_tM_t^T] &= \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}\left(\frac{A_{ij}}{p_{ij}}e_ie_j^T - A\right)\left(\frac{A_{ij}}{p_{ij}}e_je_i^T - A^T\right)\\ &= \sum_{i=1}^{m}\sum_{j=1}^{n}\frac{A_{ij}^2}{p_{ij}}e_ie_i^T - AA^T.\end{aligned}$$

Sampling according to probabilities of eqn. (1.2), and because $A_{ij} = 0$ is never sampled, we get, for $A_{ij}\neq 0$,

$$\begin{aligned}\sum_{i=1}^{m}\sum_{j=1}^{n}\frac{A_{ij}^2}{p_{ij}} &= \|A\|_F^2\sum_{i=1}^{m}\sum_{j=1}^{n}\left\{\frac{\alpha\|A\|_F^2}{|A_{ij}|\|A\|_1} + (1-\alpha)\right\}^{-1},\\ &\leq \|A\|_F^2\sum_{i=1}^{m}\max_i\sum_{j=1}^{n}\left\{\frac{\alpha\|A\|_F^2}{|A_{ij}|\|A\|_1} + (1-\alpha)\right\}^{-1}.\end{aligned}$$

Thus,

$$\mathbb{E}[M_tM_t^T] \preceq \|A\|_F^2\beta_1\sum_{i=1}^{m}e_ie_i^T - AA^T = \|A\|_F^2\beta_1 I_m - AA^T.$$

Note that, $\|A\|_F^2 \beta_1 I_m$ is a diagonal matrix with all entries non-negative, and AA^T is a positive semi-definite matrix. Therefore,

$$(\mathbb{E}[M_t M_t^T])^2 \leq \|A\|_F^2(\beta_1 - \sigma^2 \min(A)).$$

Similarly, we can obtain

$$(\mathbb{E}[M_t^T M_t])^2 \leq \|A\|_F^2(\beta_2 - \sigma^2 \min(A)),$$

where,

$$\beta_2 = \max_j \sum_{i=1}^{m} \left\{ \frac{\alpha\|A\|_F^2}{|A_{ij}|\|A\|_1} + (1-\alpha) \right\}^{-1}, \quad \text{for } A_{ij} \neq 0.$$

We can now apply Theorem 1 with

$$\rho^2(\alpha) = \|A\|_F^2 \max\{\beta_1, \beta_2\} - \sigma^2 \min(A)$$

and

$$\gamma(\alpha) = \|A\|_1 \frac{\alpha + (1-\alpha)\lambda}{\|A\|_F^2} + \|A\|_2$$

to conclude that $\|S\Omega(A) - A\|_2 \leq \varepsilon$ holds subject to a failure probability at most

$$(m+n) \exp\left(-\frac{s\varepsilon^2}{2\rho^2(\alpha) + \frac{2}{3}\gamma(\alpha)\varepsilon} \right).$$

Finally, we can bound the failure probability by δ, and setting $\varepsilon = \epsilon * ||\mathbf{A}||_2$

6.2 Implementation

Our implementation of this algorithm is written in a jupyter notebook using numpy in python [3]. The data sets were downloaded and stored alongside the jupyter notebook so that it could be directly used by the python code for testing with the algorithm.

7 Experiments

7.1 Validation using Synthetic data

Kundu validates the necessity of 1 by creating a plot $f(\alpha)$ in 7 for a perturbed matrix created as in 2. The author finds that $f(\alpha)$ is not minimized at extreme values of α showing the optimal solution is not presented by L_1 or L_2 norm alone but by a combination of both factors. We created various different types of synthetic data in similar way to the author with varying amount of noise and size of matrix to verify if we obtain similar results.

7.2 Data Sets

While Kundu's original paper tested its sparsification and PCA approximation algorithms well, it didn't do so extensively. There are still many different types of data sets to test it with, so our paper has determined three more data sets that are worth testing on. Our choices of data sets each have their own justification for being selected.

We chose to further test the algorithm on a complete handwritten digit data set that includes zero through 9. While the original paper did test the algorithm on handwritten digits, it oddly chose to only include the numbers 1, 6, and 9 (Kundu, 2015). This could have been due to many different reasons such as a constraint on the amount of computing power available to this project. However, the inclusion of 6 and 9 specifically was likely not a coincidence. There is merit in seeing how the results of the algorithm on a complete handwritten digit set will compare to the paper's original results on a limited portion of the set.

As for the license plate image data set, we chose this data set for a few reasons. Firstly, it is simply an image data set which is perfect for being represented as matrices, and the results are able to be easily visualized to be better understood. Secondly, images of license plates form a sort of block matrix due to the license plate being its own section within the image. Block images are notoriously difficult for sparsification algorithms to deal with, so testing Kundu's algorithm against a real world data set of block matrices is important ().

[3] https://github.com/fespadea/Project-Repo

We lastly chose the football match history data set because it is very different from the style of data tested against in the original paper. The results of matches in football can vary significantly, and the increments in which points are awarded creates an odd distribution of scores. It will be interesting to see how the algorithm is able to handle this different type of data and its range of values.

7.3 Experiment Design

We tested the matrix algorithm in the same way as the original paper by apply the algorithm to different matrices with different values of s and α. We then compare the output of the sparsification to the original matrix by taking the norm of the difference between the two and dividing it by the norm of the original matrix. We then also take the log base 2 of the result.

$$Error = \log_2(||A - \tilde{A}||_2/||A||_2)$$

We will then do this repeatedly for the same different values of s and α as in the original paper (Kundu, 2015) while also including results for $\alpha = 0$ which the original paper oddly chose to exclude (Kundu, 2015). We use the equation

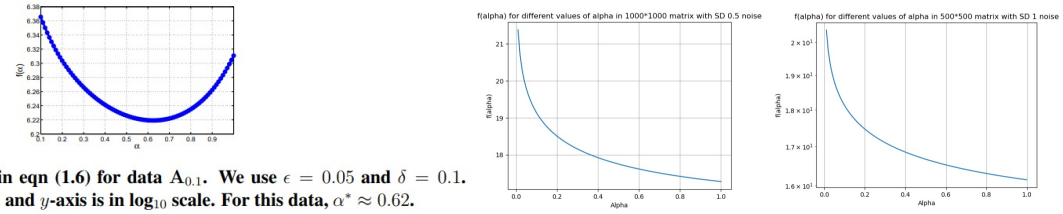
$$\frac{s}{k * (m + n)} = sMult$$

to determine our value of s for each iteration where k is the stable rank of the matrix and m and n are its dimensions. The $sMult$ value will be changed to 1, 3, and 5, so that we test different values of s as was done in Kundu's thesis (Kundu, 2015). For α, we will use the values from 0 to 1 in increments of 0.1. The results for just using l_1 sampling are represented by $\alpha = 1$, and the results for just using l_2 sampling are similarly represented by $\alpha = 0$. We then graph the error results against the different values of α with the different values of s being graphed as different lines.

7.4 Results

7.4.1 Validation using Synthetic data

The author had created 2 by using $500 * 500$ binary data matrix and perturbed it by noise of standard deviation 0.1. We tried similar experiments, but with different size of matrix and different standard deviation for the noise. Our results in comparison to Kundu's are presented in figure 2



Plot of $f(\alpha)$ in eqn (1.6) for data $\mathbf{A}_{0.1}$. We use $\epsilon = 0.05$ and $\delta = 0.1$. x-axis plots α and y-axis is in $\log_{10}$ scale. For this data, $\alpha^* \approx 0.62$.

Figure 2: On the left is result obtained by author compared to our results presented in center and right diagrams

The author found clear minimal value of α between 0 and 1 justifying the need of hybrid sampling whereas our data obtained minimum value when alpha was near 1 showing $L1$ sampling in itself is sufficient in such cases for sampling the data. The author does put a disclaimer in their report by stating "We must point out that, in this example, we control the noise for $\tilde{\mathbf{A}}_{0.1}$, and we know what a good threshold may look like. However, in reality we have no control over the noise. Therefore, choosing the right threshold for l_2, without any prior knowledge, is an improbable task." This statement combined with our findings shows that range of need for hybrid sampling might be very limited.

7.4.2 Handwritten digits

We test the sparsification algorithm on this data set by applying it to 2000 randomly selected elements of the MNIST Handwritten Digit dataset. We also apply the algorithm to exclusively all of the 1, 6, and 9 digits to compare our results to the original paper's graph where they did the same experiment. We also apply the algorithm to just 2000 1, 6, and 9 digits to have a more direct comparison to our general 2000 Handwritten digits results. Note that this is a different digit dataset than the one used in the original paper, so the results may differ. We prepared the data by flattening each digit image and then concatenating them into a matrix where each flattened digit is a row of the matrix. Thus, for the 2000 digit datasets, we end up with 2000 by 768 matrices, and a 18609 by 768 matrix for the 1, 6, and 9 digits matrix.

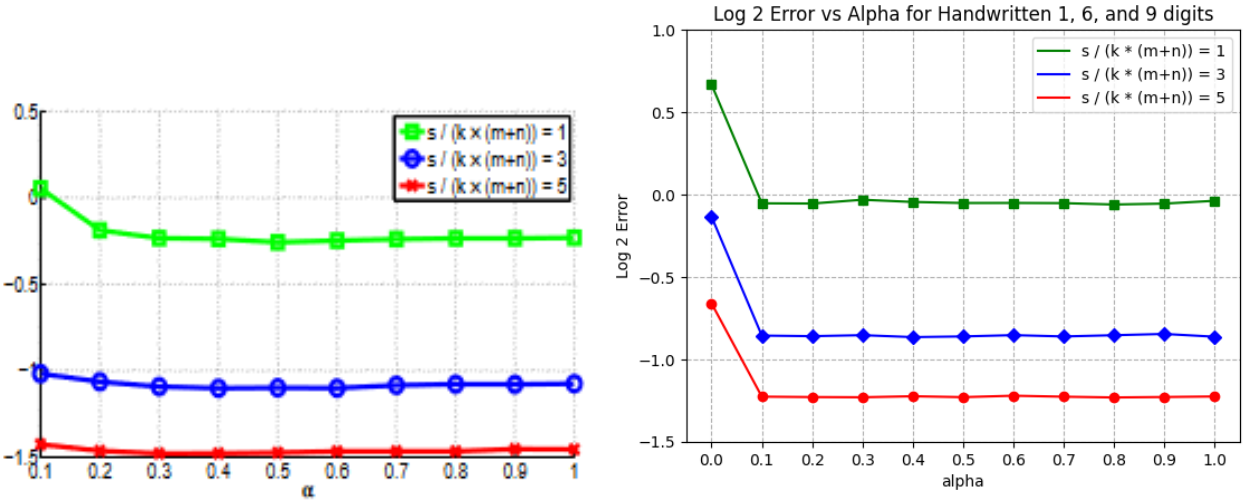


Figure 3: On the left is the original graph from the Kundu's thesis (Kundu, 2015), and on the right is our graph of our calculated error for our sparsified handwritten 1, 6, and 9 digit matrices, as described above, against values of alpha for different values of s.

First, we look at our results for the set of 1, 6, and 9 digits in figure 3. Here we see a similar pattern to that of the original paper (the left graph in figure 3) where the value of α seems to matter very little as the lines for the different values of s are largely flat. In this case, it is clear that hybrid sampling is not much better than pure l_1 or l_2 sampling.

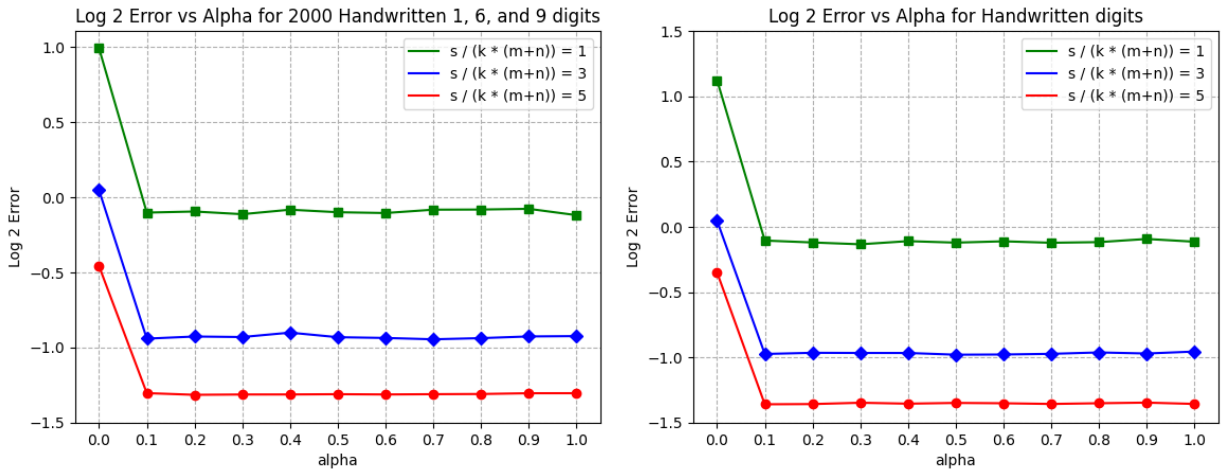


Figure 4: On the left is a graph of our calculated error for our sparsified 2000 handwritten 1, 6, and 9 digit matrices, as described above, against values of alpha for different values of s, and on the right we have the same type of graph but for our 2000 handwritten digit matrix of all digits.

Next, we look at our results for the 2000 random handwritten digits in figure 4. We have also performed the experiments on just 2000 1, 6, and 9 handwritten digits to have a direct comparison between the results when the digits are restricted. We can see that the results are similar between the two graph, but they are slightly better for the unrestricted matrix since they are lower values. However, it is still once again clear that hybrid sampling is not much more effective than pure l_1 or l_2 sampling in this case.

7.4.3 License Plates

We tested the sparsification algorithm on the License plate data set in much the same way that we did with the handwritten digit data set. The major difference with this data set is simply that the individual matrices are much larger due to the

Figure 5: The license plate images corresponding to the graphs in figure 6 (images 0-3 of the dataset).

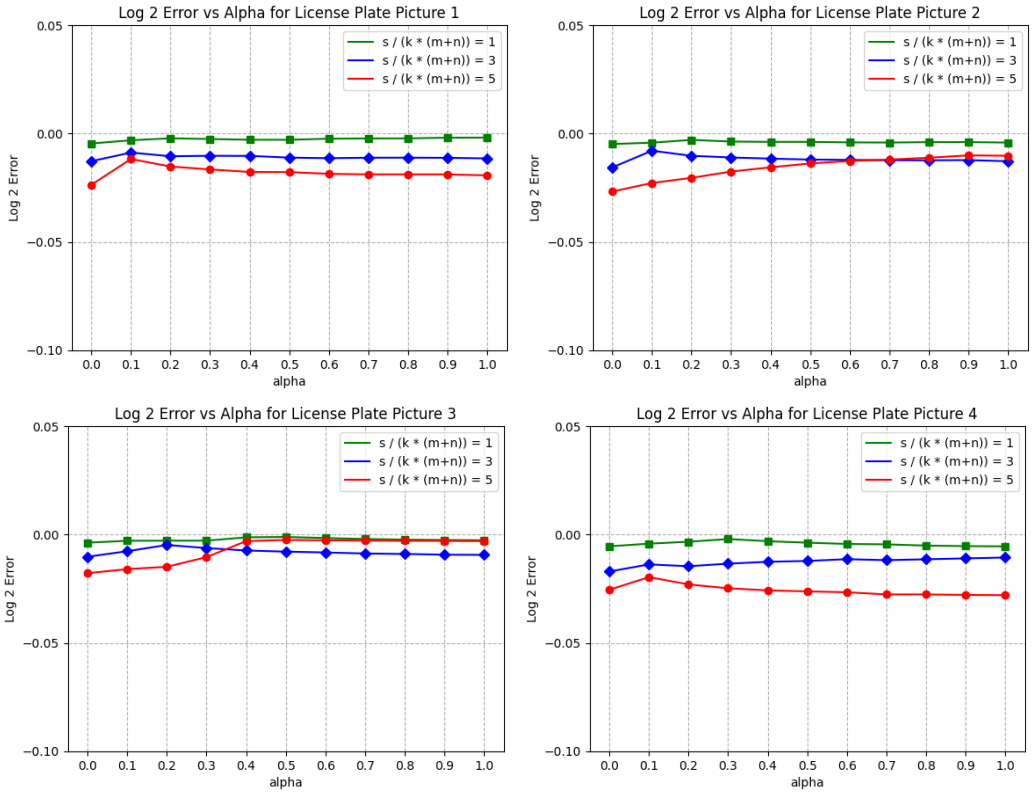


Figure 6: The graphs of our calculated error for our sparsified license plate matrices, as described above, against values of alpha for different values of s.

higher resolution images, so we individually ran the aglorithm on 4 of the images instead of concatenating them into one large matrix. Also, these matrices are examples of block matrices that are generally difficult for sparsification algorithms to deal with, so it will be interesting to see if hybrid sampling can achieve better results. The images used can be seen in figure 5.

We can see our results for the license plate pictures in figure 6. Here, our results are not very good across the board, likely due to the fact that these are block matrices. However, we do end up seeing some interesting results for these images. Most notably, graphs 2 and 3 actually see diminishing results as $sMult$ reaches 5 for high values of α. These, along with graph

1, are also unique in that l_2 sampling performs better than l_1 sampling. In the case of graph 1, l_1 and l_2 sampling both perform better individually than they do in any combination. Graph 4 is a bit closer to what we would expect, but it is still a little odd in that l_1 sampling performs better than hybrid sampling at low α. Overall, hybrid sampling underperforms with these images when compared to l_1 and l_2 sampling, although they all perform poorly.

7.4.4 Football Historical Scores

We also tested the sparsification algorithm on the football historical scores data set in much the same way that we did with the handwritten digit and license plate data sets. This is a simple 2 column matrix of the home and away scores from 13516 games.

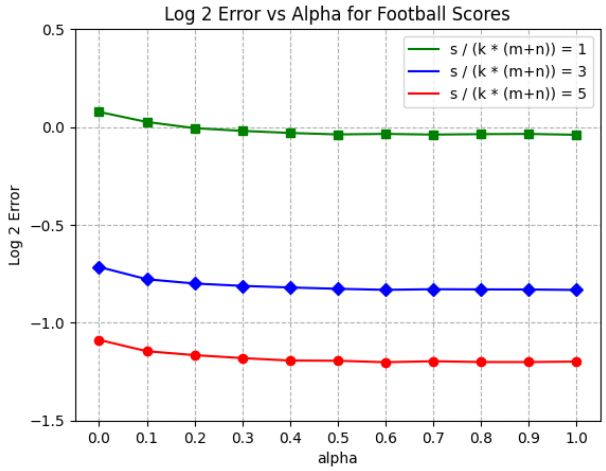


Figure 7: Graph of our calculated error for our sparsified football scores matrices, as described above, against values of alpha for different values of s.

The results for the sparsification of the football scores can be seen in figure 7. These results are once again relatively standard with an increase in s causing a decrease in error, and l_1 performing better than l_2, but hybrid sampling not performing noticeably better. The sparsification algorithm does not care about the shape of the matrix, so the heavy verticality of this matrix does not affect its results. Overall, it does not seem that the odd nature of football scores has made a noticeable impact on the results when compared to those achieved by running the algorithm on handwritten digits.

8 Conclusion

Overall, our results weren't very impressive. We found that hybrid sampling didn't noticeably improve our results with any of the tested datasets, and was actually worse in certain cases. This isn't to say that Kundu's hybrid sparsification algorithm is worthless, seeing as he found cases where it performed well in comparison to pure l_1 and l_2 sampling, but it isn't as widely applicable as one may hope. If computation power or time is not a concern, then there is no harm in performing hybrid sparsification as it will result in the optimal combination of l_1 and l_2 sampling for your use case. However, if computation is a concern, one may be better off just trying out l_1 or l_2 sampling as the benefits of hybrid sampling can often be minimal or nonexistent.

9 Going Forward:

One major advantage of hybrid sampling presented by the author is the fast approximation of PCA of the matrix compared to other sampling method. Going forward, we would like to study the algorithms he performs to modify the hybrid sampling to one-pass hybrid sampling and then study his findings regarding the PCA approximation. Then only we can truly discuss the true need of hybrid sampling as stated by the author.

REFERENCES

D Achlioptas. Fast computation of lowrank matrix approximation. In *Proceedings of the thirty-P third annual ACM symposium on Theory of computing*, pp. 611–618. ACM, 2001.

Dimitris Achlioptas and Frank McSherry. Fast computation of low-rank matrix approximations. *Journal of the ACM (JACM)*, 54(2):9–es, 2007.

Dimitris Achlioptas, Zohar Karnin, and Edo Liberty. Matrix entry-wise sampling: Simple is best. *Submitted to KDD*, 1: 1–4, 2013a.

Dimitris Achlioptas, Zohar Karnin, and Edo Liberty. Matrix entry-wise sampling: Simple is best. *Submitted to KDD*, 1: 1–4, 2013b.

Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

Petros Drineas and Anastasios Zouzias. A note on element-wise matrix sparsification via a matrix-valued bernstein inequality. *Information Processing Letters*, 111(8):385–389, 2011a.

Petros Drineas and Anastasios Zouzias. A note on element-wise matrix sparsification via a matrix-valued bernstein inequality. *Information Processing Letters*, 111(8):385–389, 2011b.

Abhisek Kundu. *Element-wise matrix sparsification and reconstruction*. Rensselaer Polytechnic Institute, 2015.

Benjamin Recht. A simpler approach to matrix completion. *Journal of Machine Learning Research*, 12(12), 2011.

A APPENDIX

Github repository with code used in this paper: https://github.com/fespadea/Project-Repo

Handwritten Digit Dataset: https://www.kaggle.com/datasets/hojjatk/mnist-dataset

License Plate Dataset: https://www.kaggle.com/datasets/andrewmvd/car-plate-detection

Football Scores Dataset: https://www.kaggle.com/datasets/tobycrabtree/nfl-scores-and-betting-data