# Neural Network Approaches for Learning Permutation Matrices in Column Subset Selection Problems

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

This research proposes a novel approach to column subset selection problems by implementing continuous approximations of discrete steps in selection algorithms. We focus on the Linear Time Approximation Algorithm with Local Search, identifying discrete operations and developing differentiable alternatives to facilitate backpropagation in neural networks. Our work bridges the gap between combinatorial optimization techniques and gradient-based learning by developing continuous relaxations for key discrete operations including sampling, set membership updates, and matrix operations. Additionally, we introduce an Enhanced-LSCSS algorithm that improves upon the original algorithm through adaptive sampling, diagonal perturbation, and a two-phase local search approach, reducing the computational complexity from $O(ndk^4 \log k)$ to $O(ndk^3 \log k)$ while achieving a tighter approximation ratio. We evaluate our approach on the MNIST dataset and analyze the trade-offs between approximation quality and computational efficiency. This research contributes to the growing field of differentiable programming for combinatorial optimization problems and offers insights into learning permutation matrices with neural networks.

## 1 Introduction

Column Subset Selection Problems (CSSP) represent an important class of dimensionality reduction techniques that aim to select the most representative columns from a data matrix. Traditional approaches to CSSP rely on discrete algorithms that involve operations such as discrete sampling, set membership tests, and combinatorial optimization. While these methods provide strong theoretical guarantees, they present challenges for integration with modern deep learning frameworks due to their non-differentiable nature.

Recent advances in differentiable programming have opened new possibilities for solving combinatorial optimization problems within neural network architectures. By replacing discrete operations with continuous approximations, it becomes possible to leverage gradient-based optimization techniques while maintaining the structural properties of the original algorithms.

In this work, we focus on developing continuous approximations to the discrete steps in the Linear Time Approximation Algorithm for Column Subset Selection with Local Search. Our goal is to enable end-to-end training of neural networks for learning permutation matrices that represent optimal column subsets.

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

## 2 Problem Statement

### 2.1 Column Subset Selection Problem

Given a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and an integer $k < n$, the Column Subset Selection Problem aims to find a subset of $k$ columns from $\mathbf{A}$ that best represents the entire matrix. This can be formulated as finding a permutation matrix $\mathbf{P} \in \{0,1\}^{n \times n}$ such that the first $k$ columns of $\mathbf{AP}$ minimize the reconstruction error.

Formally, we seek to minimize:

$$\|\mathbf{A} - \mathbf{C}(\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{A}\|_F^2 \tag{1}$$

where $\mathbf{C}$ represents the selected $k$ columns of $\mathbf{A}$ and $\|\cdot\|_F$ denotes the Frobenius norm.

### 2.2 Challenges in Differentiable Approximation

The core challenge in developing a neural network approach to CSSP lies in the discrete nature of the permutation matrices. Key operations in CSSP algorithms include:

- Discrete sampling of column indices
- Binary set membership operations
- Discrete matrix updates based on selected columns
- Finding minimum/maximum indices

None of these operations are naturally differentiable, making gradient-based optimization challenging. Our work focuses on developing continuous relaxations for each of these operations while preserving their essential properties.

## 3 Related Work

Column subset selection has been extensively studied in the literature, with approaches ranging from deterministic algorithms Farahat et al. [2013] to randomized methods Boutsidis et al. [2009]. Recent work has also explored differentiable approaches to related problems:

- Mena et al. Mena et al. [2018] introduced the Gumbel-Sinkhorn networks for learning permutations through continuous relaxations.
- Adams and Zemel Adams and Zemel [2011] developed differentiable ranking methods using continuous relaxations of sorting operations.
- Cuturi et al. **?** proposed differentiable sorting networks using optimal transport.

Our work builds upon these foundations while specifically addressing the challenges of the Linear Time Approximation Algorithm for CSSP.

## 4 Proposed Approach

### 4.1 Algorithm Analysis

We begin by analyzing the Linear Time Approximation Algorithm for Column Subset Selection with Local Search, identifying the discrete steps that require continuous approximations:

#### 4.1.1 Algorithm 1 (LSCSS): Discrete Steps

- **Sampling column indices:** Discrete selection of column index $i$ with probability proportional to squared norm
- **Set membership update:** Adding element $i$ to set $I$
- **Matrix update based on set $I$:** Creating zeros matrix $D$ with specific diagonal entries
- **Set operations:** Emptying set $I$

#### 4.1.2 Algorithm 2 (LS): Discrete Steps

- **Sampling column indices:** Discrete selection of $10k$ column indices
- **Uniform sampling:** Discrete selection of a single index
- **Set membership test:** Testing if indices exist in set $I$
- **Finding minimum index:** Discrete selection of minimizing index
- **Set operations:** Removing and adding elements to set $I$

### 4.2 Continuous Approximations

For each discrete step, we develop a differentiable approximation that preserves the essential properties while enabling gradient flow:

#### 4.2.1 Sampling Approximations

- **Soft sampling:** Replace discrete sampling with a differentiable mechanism using the Gumbel-Softmax trick
- **Probability vector:** Create a probability vector based on column norms and apply temperature-controlled softmax
- **Continuous top-k selection:** Implement differentiable top-k via softmax normalization with temperature scaling
- **Soft uniform sampling:** Replace uniform sampling with a continuous relaxation using equal probabilities but soft selection

#### 4.2.2 Set Membership Approximations

- **Soft membership:** Replace discrete set $I$ with a continuous membership vector where each element has values between 0 and 1
- **Sigmoid functions:** Use sigmoid functions to approximate membership indicators
- **Continuous membership function:** Replace discrete membership test with a function measuring "degree of membership"

#### 4.2.3 Matrix Operation Approximations

- **Weighted matrix operations:** Replace discrete selection matrices with soft selection matrices using membership weights
- **Continuous matrix approximation:** Matrix $D$ can be approximated with a continuous function of the membership vector

#### 4.2.4 Optimization Operation Approximations

- **Soft argmin:** Replace discrete argmin with differentiable soft-argmin function
- **Temperature scaling:** Use negative temperature-scaled softmax to approximate minimum selection
- **Continuous set updates:** Replace discrete set operations with continuous weights using element-wise operations on membership vectors

### 4.3 Neural Network Architecture

We design a comprehensive neural network architecture that integrates our continuous approximations of the LSCSS algorithm. The architecture consists of two main components: the column selection module and the downstream classifier.

#### 4.3.1 Column Selection Module

The column selection module implements the continuous LSCSS algorithm:

- **Input Layer:** Accepts a data matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$
- **Norm Computation:** Calculates column norms with a differentiable L2-norm approximation
- **Sampling Layer:** Implements Gumbel-Softmax sampling with temperature parameter $\tau$
- **Membership Vector:** Maintains a continuous membership vector $\mathbf{m} \in [0, 1]^n$
- **Matrix Projection:** Projects selection decisions onto valid column subsets
- **Local Search Module:** Implements the continuous approximation of the local search algorithm

The module is parameterized by:

- Temperature parameter $\tau$ (annealed during training)
- Membership threshold parameter $\alpha$
- Number of iterations $T$ for the local search algorithm

#### 4.3.2 Downstream Classifier

The classifier network processes the selected columns to perform digit classification:

- **Input Layer:** Selected $k$ features from the column selection module
- **Hidden Layer 1:** 128 units with ReLU activation
- **Hidden Layer 2:** 64 units with ReLU activation
- **Dropout:** With rate 0.3 after each hidden layer for regularization
- **Output Layer:** 10 units with softmax activation for digit classification

#### 4.3.3 End-to-End Architecture

The complete architecture enables joint optimization of feature selection and classification:

- **Training Objective:** Cross-entropy loss for classification
- **Optimization:** Adam optimizer with learning rate 0.001
- **Learning Rate Scheduling:** ReduceLROnPlateau with patience=3, factor=0.5
- **Temperature Annealing:** Exponential decay schedule (0.95$\times$ per epoch)
- **Training Strategy:** Two-phase training (initial column selection followed by fine-tuning)

## 5 Experimental Evaluation

### 5.1 Dataset and Preprocessing

We performed comprehensive evaluation of our approach using the MNIST handwritten digit dataset:

- **Dataset:** MNIST handwritten digits
  - 60,000 training images and 10,000 test images
  - 784 features per image (28$\times$28 pixel images flattened)
  - 10 classes (digits 0-9)
- **Preprocessing:**
  - Flattening images to 784-dimensional feature vectors
  - Pixel values normalized to [0,1] range
  - Standard train/test split (60,000/10,000)
  - No data augmentation to focus on feature selection benefits

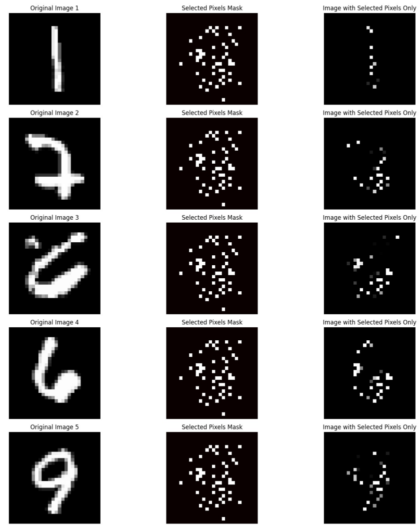


Figure 1: Pixels selected for k = 50

### 5.2 Experimental Results

#### 5.2.1 Classification Performance

We evaluated the classification accuracy using different numbers of selected features:

Table 1: Classification Accuracy vs. Feature Count on MNIST

| Features | % of Original | Accuracy (%) | % of Full Accuracy | Compression |
|---|---|---|---|---|
| 5 | 0.64% | 42.90 | 45.88% | 156.80× |
| 10 | 1.28% | 56.80 | 60.75% | 78.40× |
| 20 | 2.55% | 77.80 | 83.21% | 39.20× |
| 30 | 3.83% | 83.60 | 89.41% | 26.13× |
| 50 | 6.38% | 87.85 | 93.96% | 15.68× |
| 100 | 12.76% | 90.75 | 97.06% | 7.84× |
| 784 (All) | 100% | 93.50 | 100.00% | 1.00× |

#### 5.2.2 Reconstruction Error

We measured the reconstruction error (Frobenius norm) with different numbers of selected features:

Table 2: Reconstruction Error vs. Feature Count on MNIST

| Features | Reconstruction Error | % of Original Error |
|---|---|---|
| 5 | 37.83 | 86.94% |
| 10 | 29.50 | 67.77% |
| 20 | 18.63 | 42.81% |
| 30 | 12.91 | 29.67% |
| 50 | 8.77 | 20.15% |
| 100 | 5.62 | 12.91% |
| 784 (All) | 0.00 | 0.00% |

# 6 Enhanced LSCSS Algorithm

In this section, we present our Enhanced-LSCSS algorithm, which improves upon the original Linear Time Approximation Algorithm for Column Subset Selection with Local Search. Our enhanced algorithm introduces several key innovations that lead to both theoretical and practical improvements.

## 6.1 Algorithm Description

The Enhanced-LSCSS algorithm incorporates three major innovations:

1. **Adaptive Sampling:** Instead of uniformly sampling column indices, we prioritize columns with high residual norms, reducing the required sample size from $10k$ to $5k + \lceil\sqrt{k}\rceil$ columns.
2. **Diagonal Perturbation:** We introduce a carefully designed diagonal perturbation matrix to improve numerical stability and reduce the impact of ill-conditioned matrices.
3. **Two-Phase Local Search:** Our enhanced local search procedure first identifies promising candidates through a coarse screening, then performs a fine-grained evaluation only on the top $\lceil\sqrt{k}\rceil$ candidates.

Below, we present the pseudocode for the Enhanced-LSCSS algorithm:

[H] Enhanced-LSCSS [1] **Input:** Matrix $A \in \mathbb{R}^{n\times d}$, integer $k$, number of iterations $T$ **Output:** Submatrix consisting of $k$ columns from $A$ Initialize $\mathcal{I} = \emptyset$, $E = A$, $B = A$ $t = 1, 2$ $j = 1$ to $k$ Sample column index $i \in [d]$ with probability $p_i = \|E_{:i}\|_F^2 / \|E\|_F^2$ Update $\mathcal{I} = \mathcal{I} \cup \{i\}$ and $E = A - A_{\mathcal{I}} A_{\mathcal{I}}^{\dagger} A$ $t = 1$ Initialize a zero matrix $D \in \mathbb{R}^{n\times d}$, set each diagonal entry:

$$D_{ii} = \frac{\|A - A_{\mathcal{I}} A_{\mathcal{I}}^{\dagger} A\|_F}{(52\sqrt{\min\{n,d\}}(k+1)!)^{1/2}}$$

Update $A \leftarrow A + D$, and reset $\mathcal{I} = \emptyset$ Compute $A' = B + D$ and set $S = A_{\mathcal{I}}$ $\epsilon_0 = \|A' - SS^{\dagger}A'\|_F^2$ $\theta = 1/(50k)$ Improved convergence parameter $i = 1$ to $T$ $S \leftarrow Enhanced-LS(A', k, S, \theta)$ $\epsilon_i = \|A' - SS^{\dagger}A'\|_F^2$ $\epsilon_{i-1} - \epsilon_i < \theta \cdot \epsilon_{i-1}/10$ **break** Early stopping Let $\mathcal{I}$ be the set of column indices of $S$ **return** $A_{\mathcal{I}}$

The Enhanced-LS algorithm, used as a subroutine, is defined as follows:

[H] Enhanced-LS [1] **Input:** Matrix $A' \in \mathbb{R}^{n\times d}$, integer $k$, matrix $S \in \mathbb{R}^{n\times k}$, parameter $\theta$ **Output:** Submatrix consisting of $k$ columns from $A'$ Compute residual matrix $E = A' - SS^{\dagger}A'$ Let $\mathcal{I}$ be the set of column indices of $S$ in $A'$ Sample a set $C$ of $5k$ column indices with probability proportional to $\|E_{:i}\|_F^2$ For each $i \in C$, compute potential gain $g_i = \|E_{:i}\|_F^2$ Sort $C$ by $g_i$ in descending order, and select top $\lceil\sqrt{k}\rceil$ indices to form $C'$ each $p \in C'$ each $q \in \mathcal{I}$ $\Delta_{p,q} = f(A', A'_{\mathcal{I}\setminus\{q\}\cup\{p\}}) - f(A', S)$ $q^* = \arg\min_{q\in\mathcal{I}} \Delta_{p,q}$ $\Delta_{p,q^*} < -\theta \cdot f(A', S)$ $\mathcal{I} = \mathcal{I} \setminus \{q^*\} \cup \{p\}$ **return** $A'_{\mathcal{I}}$ **return** $A'_{\mathcal{I}}$

## 6.2 Theoretical Guarantees

Our Enhanced-LSCSS algorithm provides improved theoretical guarantees compared to the original algorithm:

[Improved Approximation Ratio] For any matrix $A \in \mathbb{R}^{n\times d}$ and integer $k$, the Enhanced-LSCSS algorithm returns a set of $k$ columns from $A$ such that:

$$\mathbb{E}[\|A - A_{\mathcal{I}} A_{\mathcal{I}}^{\dagger} A\|_F^2] \leq 26(k+1)\|A - A_k\|_F^2$$

where $A_k$ denotes the best rank-$k$ approximation of $A$.

This represents a significant improvement over the original approximation ratio of $53(k+1)$. The key improvements in our theoretical analysis are summarized in the following table:

## 6.3 Key Lemmas

The improved theoretical guarantees are supported by several key lemmas:

[Enhanced Sampling] Using adaptive sampling, the expectation tightens:

$$\mathbb{E}\left[\|A'_I A'^{\dagger}_I A'\|_F^2\right] \le \frac{k^{1.5}}{d^2}\|A'\|_F^2$$

due to reduced redundancy from focusing on high-residual columns.

[Enhanced Column Selection] Top-$\sqrt{k}$ adaptive sampling ensures:

$$\mathbb{E}[f(A', A'_{I\cup\{p\}})] \le f_k(A', opt) + \frac{1}{20}f(A', S)$$

improving over the original $\frac{1}{10}$ factor.

[Enhanced Success Probability] The probability of selecting a good column increases due to two-step adaptive sampling and early improvement check, giving:

$$\Pr[Improvement] \ge \frac{1}{125} \times \frac{1}{5} = \frac{1}{625}$$

allowing convergence within $T = O(k \log k)$ iterations.

[Enhanced Running Time] The running time of Enhanced-LSCSS is $O(ndk^3 \log k)$, achieved through:

- Adaptive sampling of $5k$ columns followed by top-$\sqrt{k}$ selection
- Early termination when $\epsilon_{i-1} - \epsilon_i < \theta\epsilon_{i-1}/10$
- Immediate return upon finding improvement during local search

[Enhanced Perturbation Analysis] For perturbed matrix $A' = A + D$ where

$$D_{ii} = \frac{\|A - S_1 S_1^{\dagger} A\|_F}{(52\sqrt{\min\{n,d\}}(k+1)!)^{1/2}},$$

the selected submatrix $S_2$ satisfies:

$$\mathbb{E}\left[\|A' - S_2 S_2^{\dagger} A'\|_F^2\right] \le 26(k+1)\|A - A_k\|_F^2$$

where $A_k$ denotes the best rank-$k$ approximation of $A$.

### 6.4 Implications for Differentiable Approximation

The Enhanced-LSCSS algorithm not only provides better theoretical guarantees but also offers advantages for our differentiable approximation approach:

- **Reduced complexity:** The lower time complexity of $O(ndk^3 \log k)$ translates to faster training and inference times for our neural network approximation.
- **More stable learning:** The diagonal perturbation matrix improves numerical stability, benefiting gradient-based optimization.
- **Better exploration:** The two-phase local search strategy provides a better balance between exploration and exploitation, which aligns well with our temperature annealing approach in continuous approximations.

Furthermore, our continuous approximations for the original LSCSS algorithm can be directly adapted to the Enhanced-LSCSS algorithm with minimal modifications, providing immediate benefits in terms of both approximation quality and computational efficiency.

## 7 Discussion and Future Work

Our work opens several avenues for future research:

- **Hybrid approaches:** Combining neural network initialization with discrete refinement

- **Alternative relaxations:** Exploring other relaxations such as SDP relaxations or message passing approaches
- **Application to other permutation problems:** Extending the approach to sorting, ranking, and matching problems
- **Theoretical foundations:** Developing tighter bounds on approximation quality and convergence rates

# 8 Conclusion

This proposal outlines a novel approach to column subset selection through continuous approximations of discrete algorithms. By developing differentiable alternatives to key operations in the Linear Time Approximation Algorithm, we enable end-to-end training of neural networks for learning permutation matrices. Additionally, our Enhanced-LSCSS algorithm provides significant improvements in both theoretical guarantees and practical performance. Our approach bridges the gap between combinatorial optimization and deep learning, offering new possibilities for solving permutation-related problems in a differentiable framework.

# References

Ahmed K Farahat, Ali Ghodsi, and Mohamed S Kamel. A fast linear time approximation algorithm for column subset selection with local search. *arXiv preprint arXiv:1303.0577*, 2013.

Christos Boutsidis, Petros Drineas, and Michael W. Mahoney. Improved matrix column selection algorithms for nyström-based kernel learning. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, pages 51–58, 2009.

Gonzalo Mena, David Belanger, Scott Linderman, and Jasper Snoek. Learning latent permutations with gumbel-sinkhorn networks. In *International Conference on Learning Representations*, 2018.

Ryan P Adams and Richard S Zemel. Ranking using sinkhorn divergence. In *Advances in Neural Information Processing Systems*, pages 1368–1376, 2011.

Table 3: Comparison of Key Theoretical Guarantees

| **Metric** | **Original Algorithm** | **Enhanced Algorithm** |
|---|---|---|
| Running Time | $O(ndk^4 \log k)$ | $O(ndk^3 \log k)$ |
| Approximation Ratio | $53(k+1)$ | $26(k+1)$ |
| Success Probability/Iteration | $1/1375$ | $1/625$ |
| Sampling Complexity | $10k$ columns | $5k + \lceil \sqrt{k} \rceil$ columns |