

RESEARCH

Open Access

# SSCNet: learning-based subspace clustering



Xingyu Xie<sup>1</sup>, Jianlong Wu<sup>2</sup>, Guangcan Liu<sup>3</sup> and Zhouchen Lin<sup>1,4,5\*</sup>

## Abstract

Sparse subspace clustering (SSC), a seminal clustering method, has demonstrated remarkable performance by effectively solving the data sparsity problem. However, it is not without its limitations. Key among these is the difficulty of incremental learning with the original SSC, accompanied by a computationally demanding recalculation process that constrains its scalability to large datasets. Moreover, the conventional SSC framework considers dictionary construction, affinity matrix learning and clustering as separate stages, potentially leading to suboptimal dictionaries and affinity matrices for clustering. To address these challenges, we present a novel clustering approach, called SSCNet, which leverages differentiable programming. Specifically, we redefine and generalize the optimization procedure of the linearized alternating direction method of multipliers (ADMM), framing it as a multi-block deep neural network, where each block corresponds to a linearized ADMM iteration step. This reformulation is used to address the SSC problem. We then use a shallow spectral embedding network as an unambiguous and differentiable module to approximate the eigenvalue decomposition. Finally, we incorporate a self-supervised structure to mitigate the non-differentiability inherent in  $k$ -means to achieve the final clustering results. In essence, we assign unique objectives to different modules and jointly optimize all module parameters using stochastic gradient descent. Due to the high efficiency of the optimization process, SSCNet can be easily applied to large-scale datasets. Experimental evaluations on several benchmarks confirm that our method outperforms traditional state-of-the-art approaches.

**Keywords:** Subspace clustering, Learning-based optimization, Linearized alternating direction method of multipliers (ADMM), Differentiable low-rank decomposition

## 1 Introduction

### 1.1 Background and limitation

Clustering constitutes a cornerstone task within the domain of machine learning. Particularly in the digital age, the ease of data acquisition from the Internet results in an abundant supply of unlabeled data. Given the prohibitive cost and time investment required for data labeling, clustering techniques have become instrumental in elucidating correlations within these datasets. Consequently, the development of efficient clustering algorithms for large datasets is highly important.

Sparse subspace clustering (SSC) [1] and low-rank representation (LRR) [2–4] have made significant strides over

the past decade. By decomposing the data matrix into a self-representative sparse (or low-rank) term and a noise term, they are capable of capturing the global sparse (or low-rank) structure of the data. The robustness to noise of these methods has led to wide-scale application in diverse fields, including image clustering, segmentation and denoising. These methods have consistently outperformed traditional clustering methods, such as spectral clustering [5] and  $k$ -means.

However, there are limitations in SSC and LRR, especially when they are applied to large datasets. First, to assign clustering labels to new data, these methods necessitate recalculation, which consumes memory on the order of  $O(n^2)$ , where  $n$  is the data size. This substantial memory requirement can render them ineffective for large-scale datasets. Second, both SSC and LRR rely on the “self-expression” property, presuming that the provided data matrix inherently serves as an effective dictionary for rep-

\*Correspondence: [zlin@pku.edu.cn](mailto:zlin@pku.edu.cn)

<sup>1</sup>National Key Lab of General AI, School of Intelligence Science and Technology, Peking University, Beijing, China

<sup>4</sup>Institute for Artificial Intelligence, Peking University, Beijing, China

Full list of author information is available at the end of the article

resentation. Unfortunately, this property may not hold true in the absence of proper data pre-processing.

Finally, they execute subspace clustering through three isolated procedures: (1) dictionary construction, (2) affinity matrix learning, and (3) spectral clustering on the affinity matrix. Given the lack of integration between these steps, the derived dictionary and affinity matrix may be sub-optimal for subspace clustering.

In response to the aforementioned challenges, recent research has attempted to leverage the benefits of deep learning to address the first two limitations. By developing suitable network architectures and training strategies, various deep clustering methods [6–14] have shown encouraging results.

Existing deep clustering approaches can generally be categorized into two groups. The first encompasses auto-encoder-based methods, where different loss functions are embedded into the coder-layer of the auto-encoder [6, 8–10, 12]. The second group includes task transformation methods that reframe the clustering task as a set of pairwise classification problems. For instance, studies such as Refs. [7, 13] explored pairwise correlations between clusters or samples to guide the parameter learning of deep neural networks. However, these deep learning methods primarily concentrate on loss function design and network training, and they are not without their shortcomings, including a lack of robustness and high demand for data. Furthermore, these methods do not possess the strong generalization ability that traditional methods offer.

## 1.2 Solutions and contributions

To exploit the nonlinear representation capabilities of deep learning while preserving the geometric and theoretical properties of traditional methods, differential programming (DP) has emerged as a compelling alternative. Broadly speaking, DP first integrates learnable parameters into classical numerical solvers, followed by discriminative learning on the training data to derive task-specific optimization schemes. Given the widespread success of deep learning across many applications, many researchers treat deep neural networks as learnable units for integration with the optimization process. For example, Sprechmann et al. [15] employed a deep auto-encoder to address unstructured robust principal component analysis problems. Zhou et al. [16] uncovered the connection between sparse coding and long short-term memory (LSTM), while Peng et al. [17] reimagined the  $k$ -means algorithm as a network with a unique structure.

Through the lens of DP, most hyper-parameters associated with traditional methods can be jointly learned via deep learning optimizers [18–21], with additional learnable parameters supplementing the limited capacity of the original strategies. Generally, DP is considered a by-product of learning-based optimization, aiming to resolve

traditional optimization problems in a differentiable and data-driven manner [22–24].

Although DP offers an efficient and differentiable solution to specific optimization problems, the constraint of differentiability precludes the direct application of DP modules to many operators, such as singular value decomposition (SVD) and matrix inversion. The differentiation of these operators remains an open problem within the machine-learning community. Indyk et al. [25] approximated low-rank decomposition via a differentiable power method, but it assumed large gaps between eigenvalues, which was rarely the case in clustering problems. In subspace clustering, SVD is pivotal. Most low-rank or sparse problems necessitate performing SVDs during the optimization procedure, implying that a differentiable SVD strategy could potentially illuminate the differentiation of most common low-rank or sparse optimization solvers, thereby generating a wealth of derived learning-based optimization methods. Notably, for SSC, the spectral clustering step involves an eigenvalue decomposition that cannot be bypassed.

Aligning with the principles of DP, we integrate all the steps of SSC with differential modules, including a universal differentiable eigenvalue decomposition module, and propose SSCNet to address the issues prevalent in subspace clustering and deep clustering. Specifically, first, we generalize the optimization procedure of the linearized alternating direction method of multipliers (L-ADMM) as a multi-block deep neural network, where each block corresponds to a step of the L-ADMM iteration. We then apply the DP framework to the SSC problem, jointly performing dictionary construction and affinity learning. This component acts as the first differentiable module in our unified network. Second, we reframe the spectral clustering process as two additional differentiable modules composed of an eigenvector mapping module and a  $k$ -means clustering module. To differentiate eigenvalue decomposition, we approximate manifold gradient descent on the Stiefel manifold and generate feasible points via a Cayley transform. Finally, by combining these differentiable modules, we obtain the proposed unified network, SSCNet, which can jointly learn the optimal dictionary, affinity matrix, and clustering parameters. We also introduce a novel re-weighting technique to handle the noise term in SSC. In terms of computational efficiency, SSCNet can be effectively optimized by stochastic gradient descent (SGD). Therefore, it is well-suited for large-scale datasets.

Our main contributions are summarized as follows:

- 1) By applying the DP framework to sparse representation and spectral clustering differentiation, we establish the novel SSCNet, which jointly learns the dictionary, affinity matrix, and clustering parameters. Unlike other deep clustering methods, SSCNet inherits the robustness of SSC and requires less training data.

- 2) We generalize manifold gradient descent as a differentiable multi-layer deep neural network capable of performing SVD or eigenvalue decomposition in a learning-based manner, which can be efficiently optimized by SGD. The proposed layer is highly versatile and may be of independent interest.
- 3) We provide several valuable techniques for effectively training SSCNet and conduct experiments on multiple datasets to evaluate the subspace clustering methods. Compared with state-of-the-art approaches, our SSCNet demonstrates superior performance.

## 2 Related work

### 2.1 Subspace clustering

For subspace clustering, most methods [1–4, 26–29] first need to learn the affinity matrix based on feature representations. Then, spectral clustering [5] is applied to group the samples based on the affinity matrix. Among the popular methods, LRR and SSC are two of the most classic methods. Based on self-representation, LRR imposes the nuclear norm, i.e., the sum of singular values, to constrain the affinity matrix under the low-rank assumption, while SSC utilizes the  $\ell_1$ -norm, i.e., the sum of absolute values of all entries, under the sparse assumption. As we mainly focus on SSC in this paper, we introduce the framework of SSC in detail. The objective function of SSC is

$$\begin{aligned} \min_{\mathbf{Z}, \mathbf{E}} \|\mathbf{Z}\|_1 + \lambda \|\mathbf{E}\|_1, \\ \text{s.t. } \mathbf{X} = \mathbf{XZ} + \mathbf{E}, \text{diag}(\mathbf{Z}) = \mathbf{0}, \end{aligned} \quad (1)$$

where  $\mathbf{0}$  is the all zero matrix,  $\lambda$  is the balance constant,  $\mathbf{Z}$  is the desired sparse affinity matrix,  $\mathbf{X}$  is the data matrix, and  $\mathbf{E}$  is the error term. The  $\ell_1$ -norm  $\|\cdot\|_1$  is defined by the sum of the absolute values of all the entries in the matrix. With the noise under a sparse pattern, SSC assumes that the data points obey an underlying linear structure and aim to sparsely represent each data instance by the linear combination of its neighbors from the same subspace. Based on SSC, Li and Vidal [26] proposed the structured SSC to jointly learn the original affinity matrix of SSC and the spectral clustering mapping function. Based on LRR, Xie et al. [4, 30] proposed an implicit block diagonal LRR. Feng et al. [27] and Lu et al. [28] investigated the block diagonal property of subspace clustering and provided a theoretical guarantee.

### 2.2 Deep clustering

With the label absent, defining a proper loss function for deep clustering is crucial. The existing deep clustering methods can be classified into two categories depending

on whether the auto-encoder is adopted. For the first category, the total loss function is defined by summing the reconstruction loss of the auto-encoder and clustering loss of the latent representation layer. For the clustering loss, Xie et al. [6] and Guo et al. [10] proposed deep embedding clustering to adopt the Kullback–Leibler divergence loss, which used highly confident samples as supervision and then made samples in each cluster distributed more densely. Yang et al. [13] incorporated the  $k$ -means loss. Ji et al. [12] added a self-representation layer in the middle of the traditional auto-encoder. Peng et al. [9] used the subspace clustering loss to regularize the learning of latent representation. For the second category, specific loss functions are directly designed based on the last layer output without auto-encoder. Yang et al. [7] introduced a recurrent-agglomerative framework to merge clusters that were close to each other. Chang et al. [11] investigated the correlation between samples based on the normalized output and used such similarity as supervision. Bojanowski and Joulin [31] directly used fixed targets uniformly sampled from a unit sphere to constrain feature assignment. Shaham et al. [32] performed the spectral clustering using deep neural networks. Caron et al. [33] directly used the results of  $k$ -means as supervised labels to train neural networks.

### 2.3 Differentiable programming

Differentiable programming, called DP, can recast the existing optimization process as a differentiable module, and then the model can be optimized in a data-driven way. For example, Gregor and LeCun [34] unfolded the optimization of  $\ell_1$ -norm regularized sparse coding as a simple recurrent neural network (RNN). The number of RNN layers corresponds to the number of iterations, and the weights correspond to the dictionary. Zhou et al. [16] developed an LSTM formulation to solve the  $\ell_1$ -norm regularized sparse coding. Peng et al. [17] recast the updating rules of  $k$ -means as a fully connected network. Yang et al. [35] defined a deep architecture over the ADMM algorithm pipeline (ADMM-Net) for compressive sensing and magnetic resonance image reconstruction. Recently, Chen et al. [36] and Liu et al. [37] theoretically revealed that DP was not only a generalized optimization, but also benefited parameter learning and even brought the linear convergence when the original optimization was unconstrained, such as compressive sensing. Unlike previous works, Xie et al. [22] provided a unified differentiable framework for problems with linear constraints. This framework is generalized from the L-ADMM [38], named differentiable linearized ADMM (D-LADMM), which is more general and has fewer auxiliary variables than ADMM-Net. Moreover, its analysis shows that, with proper activation functions, the output of D-LADMM can solve the original optimization problem. In other words, D-LADMM can solve an

optimization problem in a learning-based fashion under mild conditions. With such reformulation or transformation, the original problem can be solved by joint learning. The solution of the optimization can be efficiently computed with limited memory.

### 3 Preliminaries and general framework

In this section, we review the classic SSC and its solver—L-ADMM [38, 39] and introduce a general DP framework [22] to differentiate the solver.

#### 3.1 Dictionary-based SSC

SSC contains two steps. The first step is to solve a convex or non-convex optimization problem to obtain the coefficient matrix  $\mathbf{Z}$  and then construct a graph matrix  $\mathbf{W}$  based on  $\mathbf{Z}$ . The graph matrix  $\mathbf{W}$  describes the pair-wise similarity between the training data. The second step is to perform spectral clustering on  $\mathbf{W}$ . In particular, for spectral clustering, one needs an eigenvector decomposition of the Laplacian graph matrix and then utilizes  $k$ -means clustering on the eigenvectors to obtain the final clustering results.

In the first step, with a slight difference from the original SSC model, we consider a more general optimization problem—dictionary-based SSC with the given data sample matrix  $\mathbf{X} \in \mathbb{R}^{d \times n}$ :

$$\min_{\mathbf{Z}, \mathbf{E}} \|\mathbf{Z}\|_1 + \lambda \|\mathbf{E}\|_{2,1}, \quad \text{s.t. } \mathbf{X} = \mathbf{AZ} + \mathbf{E}, \quad (2)$$

where  $\|\mathbf{E}\|_{2,1} = \sum_i \|\mathbf{E}_{:,i}\|_2$ ,  $\mathbf{E}_{:,i}$  is the  $i$ -th column of the error term  $\mathbf{E} \in \mathbb{R}^{d \times n}$ ,  $\mathbf{A} \in \mathbb{R}^{d \times m}$  is the desired basis matrix for the subspace, and  $\mathbf{Z} \in \mathbb{R}^{m \times n}$  is the sparse affinity matrix. Since we want to maintain the robustness of the classic SSC, we model the sample-specific corruptions by  $\ell_{2,1}$ -norm to eliminate outliers. Note that one can learn the dictionary  $\mathbf{A}$  and  $\mathbf{Z}$  simultaneously in the DP framework, which we will show later. After obtaining the coefficient  $\mathbf{Z}$ , we construct the graph matrix  $\mathbf{W} \in \mathbb{R}^{n \times n}$ . One can easily set  $\mathbf{W} = |\mathbf{Z}|^T + |\mathbf{Z}|$  when  $\mathbf{A} = \mathbf{X}$ . However, for a general dictionary  $\mathbf{A}$ , it is more prevalent to set the  $(i, j)$ -th entry of  $\mathbf{W}$  as follows:

$$W_{ij} = \exp\left(-\frac{\|\mathbf{z}_i - \mathbf{z}_j\|_2^2}{\sigma}\right), \quad (3)$$

where  $\mathbf{z}_i = [\mathbf{Z}]_{:,i}$  is the  $i$ -th column of the coefficient matrix and  $\sigma$  is a re-scaling constant. Note that the graph Laplacian matrix  $\mathbf{L} \in \mathbb{R}^{n \times n}$  of  $\mathbf{W}$  is  $\mathbf{L} = \mathbf{D} - \mathbf{W}$ , where  $\mathbf{D}$  is a diagonal matrix with the  $i$ -th diagonal entry being  $\sum_j W_{ij}$ .

For the second step, we can perform spectral clustering. Given the cluster number  $\tilde{k}$ , we first find the  $\tilde{k}$  eigenvectors  $\mathbf{Y} \in \mathbb{R}^{n \times \tilde{k}}$  corresponding to the 2nd to the  $(\tilde{k} + 1)$ -th smallest eigenvalues of the graph Laplacian matrix  $\mathbf{L}$ . Then we treat each row of the  $n \times \tilde{k}$  eigenvectors matrix  $\mathbf{Y}$  as an instance and perform  $k$ -means clustering to obtain the final labels.

#### 3.2 Linearized ADMM solver

The problem in Eq. (2) can be solved by the L-ADMM with the following updating rule:

$$\begin{cases} \mathbf{T}^{(k)} = \mathbf{AZ}^{(k)} + \mathbf{E}^{(k)} - \mathbf{X}, \\ \mathbf{Z}^{(k+1)} = \mathcal{S}_\alpha(\mathbf{Z}^{(k)} - \alpha \mathbf{A}^T(\boldsymbol{\lambda}^{(k)} + \beta \cdot \mathbf{T}^{(k)})), \\ \mathbf{E}^{(k+1)} = \tilde{\mathcal{S}}_{\frac{\lambda}{\beta}}(\mathbf{X} - \mathbf{AZ}^{(k+1)} - \frac{\boldsymbol{\lambda}^{(k)}}{\beta}), \\ \boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)} + \beta(\mathbf{AZ}^{(k+1)} + \mathbf{E}^{(k+1)} - \mathbf{X}), \end{cases} \quad (4)$$

where  $\boldsymbol{\lambda}^{(k)} \in \mathbb{R}^{d \times n}$  is the Lagrange multiplier.  $\alpha$  and  $\beta$  are parameters for the shrink operators. We need  $\alpha > \|\mathbf{A}^T \mathbf{A}\|$ , i.e.,  $\alpha$  is greater than the spectral norm of the matrix  $\mathbf{A}^T \mathbf{A}$ . The shrinkage operator [40] is defined as follows:

$$\mathcal{S}_\lambda(\mathbf{X})_{ij} = (|x_{ij}| - \lambda)_+ \text{sign}(x_{ij}), \quad (5)$$

where  $(x)_+ := \max\{0, x\}$ ,  $x_{ij}$  is the  $(i, j)$ -th entry of  $\mathbf{X}$ , and  $\text{sign}(\cdot)$  is the sign function. Similarly, the column-wise shrinkage operator is defined as follows:

$$[\tilde{\mathcal{S}}_\lambda(\mathbf{X})]_{:,i} = (\|\mathbf{X}_{:,i}\|_2 - \lambda)_+ \frac{\mathbf{X}_{:,i}}{\|\mathbf{X}_{:,i}\|_2}. \quad (6)$$

Notably, we did not utilize the traditional ADMM to solve the problem in Eq. (2), since Liu et al. [41] showed that the linearized step did not make the optimization divergent. Fewer variables can reduce the computational complexity. As shown in the following subsection, we can reduce the number of learning parameters without introducing auxiliary variables, and hence accelerate the training speed. On the other hand, the traditional ADMM for the problem in Eq. (2) inverses the matrix  $(\mathbf{I} + \mathbf{A}^T \mathbf{A})$  in the iteration, where  $\mathbf{I}$  is the identity matrix. However, when we generalize matrix  $\mathbf{A}$  to a learnable and non-linear mapping, it is difficult to define the inverse of a neural network.

#### 3.3 General D-LADMM

In this section, we introduce the general framework D-LADMM [22] to differentiate steps in Eq. (4). D-LADMM treats the iteration in Eq. (4) as one neural network block and converts some fixed parameters to learnable parameters, leading to a  $k$ -layer feed-forward neural network. By setting

$$\mathbf{T}^{(k)} = \mathcal{A}_{\vartheta_1^k}(\mathbf{Z}^{(k)}) + \mathbf{E}^{(k)} - \mathbf{X}, \quad (7)$$

we have the following updating rules:

$$\begin{cases} \mathbf{Z}^{(k+1)} = \eta_{\vartheta_1^k}(\mathbf{Z}^{(k)} - \mathcal{A}_{\vartheta_2^k}^T(\boldsymbol{\lambda}^{(k)} + \beta^{(k)} \circ \mathbf{T}^{(k)})), \\ \mathbf{E}^{(k+1)} = \zeta_{\vartheta_2^k}(\mathbf{X} - \mathcal{A}_{\vartheta_1^k}(\mathbf{Z}^{(k+1)}) - \frac{1}{\beta^{(k)}} \circ \boldsymbol{\lambda}^{(k)}), \\ \boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)} + \beta^{(k)} \circ (\mathcal{A}_{\vartheta_1^k}(\mathbf{Z}^{(k+1)}) + \mathbf{E}^{(k+1)} - \mathbf{X}), \end{cases} \quad (8)$$



where  $\Theta = \{\vartheta_1^k, \vartheta_2^k, \theta_1^k, \theta_2^k, \beta^k\}_{k=0}^{K-1}$  are learnable matrices,  $\circ$  is the Hadamard product, and parameterized functions  $\eta(\cdot)$  and  $\zeta(\cdot)$  are some non-linear activation functions with parameters  $\theta_1$  and  $\theta_2$ , respectively. Here  $\mathcal{A}_{\vartheta_1}(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^m$  and  $\mathcal{A}_{\vartheta_2}^T(\cdot) : \mathbb{R}^m \rightarrow \mathbb{R}^d$ , parameterized by  $\vartheta_1$  and  $\vartheta_2$ , respectively, are non-linear parameterized mappings. The operator  $\mathcal{A}_{\vartheta_1}$  performs the mapping column-wisely if it is applied to the matrix. The two mappings are generalized from the dictionary  $\mathbf{A}$ .  $\mathcal{A}_{\vartheta_2}^T(\cdot) : \mathbb{R}^m \rightarrow \mathbb{R}^d$  is the generalized adjoint mapping of  $\mathcal{A}_{\vartheta_1}(\cdot)$ . In general, we need only the parameterized functions  $\mathcal{A}_{\vartheta_1}$  and  $\mathcal{A}_{\vartheta_1}^T$  to have a similar form, e.g., linear and adjoint mapping, convolution and deconvolution; their parameters  $\vartheta_1$  and  $\vartheta_2$  can be different. Note that  $\beta \in \mathbb{R}^{d \times n}$  in Eq. (4) is now a learnable matrix in contrast to a deterministic penalty parameter  $\beta$ . We expand the dimension of the penalty parameter such that the penalties in different directions are also learnable.

It is obvious that Eq. (8) is the same as Eq. (4) when  $\eta(\cdot)$  and  $\zeta(\cdot)$  are the original shrink operators and  $\mathcal{A}_{\vartheta_1}(\cdot)$  degenerates into the original dictionary  $\mathbf{A}$ . Compared to L-ADMM, D-LADMM first converts the shrinkage operators in Eq. (4) into learnable activation functions, and then replaces the given matrix  $\mathbf{A}$  with a non-linear parameterized mapping while expanding the dimension of the penalty parameter.

Unlike the original L-ADMM solver where no parameter is learnable, Eq. (8) can be treated as a block of a specially structured neural network and trained using SGD over the observation. Many empirical results, e.g., Refs. [17, 22, 34, 35, 42] showed that a trained  $k$ -layer DP model or its variants could obtain a good solution to the same quality within one or two orders of magnitude fewer iterations than the original optimization method. Especially, the results in Ref. [22] implied that, under mild conditions,  $\Theta$  exists such that  $\mathbf{Z}^k$  converges to the optimal solution set exponentially fast in terms of the layer number  $k$ . However, vanilla L-ADMM may struggle to have a linear convergence rate.

## 4 Differentiable SSC

In this section, we first specify each step of the differentiable solver in Eq. (8). Then, we introduce the objective to train this solver in an unsupervised way. Finally, we construct a sparse graph matrix based on the output of the differentiable SSC solver.

### 4.1 Differentiable solver for SSC

Our differentiable SSC consists of an affinity updating layer, a de-noising layer, and a multiplier updating layer. We discuss them in detail.

*Affinity updating layer  $\mathbf{Z}^{(k+1)}$ .* This layer corresponds to the first step in Eq. (8). We merge the dictionary construction step into the mapping  $\mathcal{A}_{\vartheta_1}$ , so we assume that  $d \ll m$ .

Given the variables  $\mathbf{Z}^{(k)}$ ,  $\mathbf{E}^{(k)}$  and  $\lambda^{(k)}$ , the first step in Eq. (4) is decomposed and generalized into two operations:

$$\begin{cases} \mathbf{T}^{(k)} = \mathcal{A}_{\vartheta_1^k}(\mathbf{Z}^{(k)}) + \mathbf{E}^{(k)} - \mathbf{X}, \\ \mathbf{Z}^{(k+1)} = \mathcal{R}(\mathbf{Z}^{(k)} - \mathcal{A}_{\vartheta_2^k}^T(\lambda^{(k)} + \beta^{(k)} \circ \mathbf{T}^{(k)}); \mathbf{B}_1^{(k)}), \end{cases} \quad (9)$$

where weight  $\beta^{(k)} \in \mathbb{R}^{d \times n} \geq \mathbf{0}$ , and the threshold matrix  $\mathbf{B}_1^{(k)} \in \mathbb{R}^{m \times n}$  is learnable. Denote the rectified linear unit (ReLU) function as  $r(\cdot)$ , then  $\mathcal{R}(\mathbf{X}; \mathbf{B}) = r(\mathbf{X} - \mathbf{B}) - r(-\mathbf{X} - \mathbf{B})$ .  $\mathcal{R}(\cdot; \mathbf{B})$  comes from the shrinkage operator  $\mathcal{S}_\lambda(\cdot)$  and  $\mathbf{B}$  is the threshold. We initialize the learnable weight  $\beta^{(k)}$  to all-one matrix  $\mathbf{1}$  and the learnable threshold  $\mathbf{B}_1^{(k)}$  to  $\mathbf{1} \times 0.15$ . When  $k = 0$ , we set  $\mathbf{Z}^1 = \mathcal{A}_{\vartheta_1}^T(\mathbf{A}_{\vartheta_1}(\mathbf{X}))$ .

*De-noising layer  $\mathbf{E}^{(k+1)}$ .* This layer corresponds to the second step in Eq. (8). Given  $\mathbf{Z}^{(k+1)}$  and  $\lambda^k$  as the input, the output of this layer is given by

$$\mathbf{E}^{(k+1)} = \mathcal{R}(\mathbf{X} - \mathcal{A}_{\vartheta_1^k}(\mathbf{Z}^{(k+1)}) - \mathbf{W}_1^{(k)} \circ \lambda^{(k)}; \mathbf{B}_2^{(k)}). \quad (10)$$

Similarly,  $\mathbf{W}_1^{(k)}$  and  $\mathbf{B}_2^{(k)} \in \mathbb{R}^{d \times n}$  are learnable parameters.  $\mathcal{R}(\cdot)$  is the same as that in the affinity updating layer. We set  $\lambda^{(0)} = \mathbf{0}$  when  $k = 0$ .

Note that we still adopt the non-linear function  $\mathcal{R}(\cdot; \mathbf{B})$  here. In practice, the dimension of data is usually large. The element-wise operation is more suitable for DNN training. For implementation convenience, we drop the constraint that  $\mathbf{W}_1^{(k)} = 1/\beta^{(k)}$ .

*Multiplier updating layer  $\lambda^{(k+1)}$ .* This layer corresponds to the final step in Eq. (8). Given  $\mathbf{Z}^{(k+1)}$  and  $\mathbf{E}^{(k+1)}$  as the inputs, the output of this layer is

$$\lambda^{(k+1)} = \lambda^{(k)} + \beta^{(k)} \circ (\mathcal{A}_{\vartheta_1^k}(\mathbf{Z}^{(k+1)}) + \mathbf{E}^{(k+1)} - \mathbf{X}), \quad (11)$$

where the weight matrix  $\beta^{(k)} \in \mathbb{R}^{d \times n}$  is the same as that in the affinity updating layer.

### 4.2 Differentiable SSC objective

We now construct the optimization target for training our differentiable SSC module. Note that the optimization objective for the SSC problem is shown in Eq. (2). We choose a generalized objective instead of directly using Eq. (2) as a training objective. In Eq. (2), each column norm for  $\mathbf{E}$  shares a common weight  $\lambda$ . Here, we assign different weights to different columns to ease the unbalanced data problem. Assuming that the outputs of our differentiable SSC are  $\mathbf{Z}^{(K)}$  and  $\mathbf{E}^{(K)}$ , and we define the training loss for our differentiable SSC as follows:

$$L_{\text{D-SSC}} = \sum_{i=1}^n w_i \|\mathbf{E}^{(K)}\|_{:,i} \|_2 + \lambda \|\mathbf{Z}^{(K)}\|_1, \quad (12)$$

where  $n$  is the batch size for training and the adaptive weight  $w_i$  can be calculated by

$$w_i = 1 - \frac{(1 + \mathcal{T}(\|\mathbf{E}_n\|_{:,i}\|_2))^{-1}}{\sum_{i=1}^n (1 + \mathcal{T}(\|\mathbf{E}_n\|_{:,i}\|_2))^{-1}}, \quad (13)$$

where  $\mathcal{T}(\cdot)$  is a truncation function that clips the large value to a pre-defined maximum value. Here,  $w_i$  is calculated by a variant of the student's  $t$ -distribution [43] with a different power order. Inspired by adaptive boosting, the data sample that is difficult to reconstruct can obtain a larger weight  $w_i$  than the others, and the truncation function can ensure that the loss is not too sensitive to the outliers contained in the data and prevent the outliers from obtaining too large weights. This re-weighting strategy can not only make the module focus on the hard examples but also alleviate the problem of unbalanced data distribution.

### 4.3 Graph matrix construction

We assume that the output of the differentiable SSC is  $\mathbf{Z}^{(K)}$ . Similar to the classic subspace clustering method, we construct the graph matrix  $\mathbf{W}$  based on the coefficient matrix  $\mathbf{Z}^{(K)}$ .  $W_{ij}$  represents the similarity between the coefficients  $[\mathbf{Z}^{(K)}]_{:,i}$  and  $[\mathbf{Z}^{(K)}]_{:,j}$ . The  $(i, j)$ -th entry of the matrix  $\mathbf{W}$  is defined by

$$W_{ij} = \begin{cases} \exp(-\frac{\|\mathbf{z}_i - \mathbf{z}_j\|_2^2}{\sigma_i}), & \mathbf{z}_i \in KNN(\mathbf{z}_i; N), \\ 0, & \text{otherwise,} \end{cases} \quad (14)$$

where  $\mathbf{z}_i = [\mathbf{Z}^{(K)}]_{:,i}$  corresponds to the  $i$ -th column of  $\mathbf{Z}^{(K)}$  and  $KNN(\mathbf{z}_i; N)$  represents the  $N$  nearest neighbors of  $\mathbf{z}_i$ . We choose  $N$  from the range [3, 6] in the experiments. We set the scalar  $\sigma_i > 0$  by the median of all the positive  $\|\mathbf{z}_i - \mathbf{z}_j\|_2^2$ . We symmetrize  $\mathbf{W}$  by setting  $\mathbf{W} = (\mathbf{W} + \mathbf{W}^T)/2$ .

Note that we can easily make this module differentiable by setting  $\mathbf{W}$  as the Gaussian gram matrix of  $\mathbf{Z}^{(K)}$ , i.e., by Eq. (3), but we still choose the nearest neighbors here since it produces sparse neighbors, which can prevent our method from being affected by dense non-essential neighbors produced by its differentiable counterpart. More importantly, the non-differentiable  $KNN$  operator is equivalent to setting some values to zero in  $\mathbf{W}$ . Clearing some channels to zero, such as dropout, is quite common in deep learning training, which essentially introduces explicit and implicit regularization effects. Instead of causing divergent learning, the regularization benefits generalization [44].

## 5 Differentiable spectral clustering

In this section, we provide a method to differentiate spectral clustering. The output of our differentiable SSC is the input of the differentiable spectral clustering module. Given the cluster number  $\tilde{k}$ , traditional spectral clustering contains two steps: (1) find the  $\tilde{k}$  eigenvectors  $\mathbf{Y}$  corresponding to the 2nd to the  $(\tilde{k} + 1)$ -th smallest eigenvalues

of the graph Laplacian matrix  $\mathbf{L}$ ; (2) take each row of the  $n \times \tilde{k}$  eigenvectors matrix  $\mathbf{Y}$  as a new feature of the input data and then perform  $k$ -means clustering on all the rows.

We first reformulate the eigenvector decomposition and provide a differentiable method to perform the decomposition approximately. Then, we differentiate the  $k$ -means clustering using a self-supervised strategy [33]. Finally, we introduce the training objective for the whole differentiable spectral clustering module.

### 5.1 Differentiable eigenvector approximation

Recall the definition of the graph Laplacian matrix  $\mathbf{L} = \mathbf{D} - \mathbf{W}$ , where  $\mathbf{D}$  is a diagonal matrix with the  $i$ -th diagonal entry being  $\sum_j W_{ij}$ . The  $\tilde{k}$  eigenvectors corresponding to the graph Laplacian matrix  $\mathbf{L}$  are the solution of the following optimization problem:

$$\min_{\mathbf{Y} \in \mathbb{R}^{n \times \tilde{k}}} \sum_{i,j} W_{ij} \|\mathbf{Y}_{:,i} - \mathbf{Y}_{:,j}\|_2^2, \quad \text{s.t. } \mathbf{Y}^T \mathbf{Y} = \mathbf{I}, \quad (15)$$

where  $\tilde{k}$  is the cluster number. However, we do not directly perform the eigenvalue decomposition on the graph Laplacian matrix  $\mathbf{L}$  here. The exact eigenvalue decomposition is non-differentiable, and how to differentiate it is still an open problem in the deep learning community. Indyk et al. [25] approximated the low-rank decomposition by a modified differentiable power method. However, this method might fail when the gaps among eigenvalues are small. Another problem with exact eigenvalue decomposition is that we cannot perform incremental learning for batch training, which is important for large-scale data. Therefore, an eigenvalue-gap independent and differentiable method is needed to solve the problem in Eq. (15). This method is provided below.

*Optimization with orthogonality constraints.* The usual method for solving the general orthogonality-constrained optimization problem is to perform the manifold gradient descent on the Stiefel manifold, which evolves along the manifold geodesic. Specifically, manifold gradient descent (GD) updates the variable in the manifold tangent space along the objective function gradient projected into the tangent plane. Then, the procedure is repeated in the tangent space of the updated variable [45]. As usual, manifold GD requires SVDs to generate feasible points on the geodesic, which is non-differentiable. Fortunately Refs. [46, 47] developed a technique to approximately solve an orthogonality-constrained optimization problem that only consists of matrix multiplication and addition and does not rely on SVDs.

Note that the variable is  $\mathbf{Y} \in \mathbb{R}^{n \times \tilde{k}}$ . Denote by  $\mathbf{G} \in \mathbb{R}^{n \times \tilde{k}}$  the gradient of the objective function in Eq. (15) at  $\mathbf{Y}$ . Then, the projection of  $\mathbf{G}$  in the tangent plane of the Stiefel manifold at  $\mathbf{Y}$  is  $\mathbf{P}\mathbf{Y}$ , where  $\mathbf{P} = \mathbf{G}\mathbf{Y}^T - \mathbf{Y}\mathbf{G}^T$  and  $\mathbf{P} \in \mathbb{R}^{n \times n}$  [46]. We choose the canonical metric on the tangent space as

the equipped Riemannian metric. Instead of parameterizing the geodesic of the Stiefel manifold along direction  $\mathbf{P}$  using the exponential function, inspired by Refs. [46, 47], we generate feasible points by the following Cayley transform:

$$\mathbf{Y}(t) = \mathbf{C}(t)\mathbf{Y}, \quad (16)$$

where

$$\mathbf{C}(t) = \left( \mathbf{I} + \frac{t}{2}\mathbf{P} \right) \left( \mathbf{I} - \frac{t}{2}\mathbf{P} \right)^{-1}, \quad (17)$$

where  $\mathbf{I}$  is the identity matrix and  $t$  is a parameter used for updating the current  $\mathbf{Y}$ . One can easily verify that  $\mathbf{Y}(t)$  has the following properties:

- 1)  $d\mathbf{Y}(0)/dt = -\mathbf{P}\mathbf{Y}$ ;
- 2)  $\mathbf{Y}(t)$  is smooth in  $t$ ;
- 3)  $\mathbf{Y}(0) = \mathbf{Y}$ ;
- 4)  $(\mathbf{Y}(t))^T \mathbf{Y}(t) = \mathbf{I}$ ,  $\forall t \in \mathbb{R}$ , given  $\mathbf{Y}^T \mathbf{Y} = \mathbf{I}$ .

It is obvious that  $\mathbf{Y}(t)$  can result in a smaller objective function value than  $\mathbf{Y}$  on the Stiefel manifold when  $t$  is in a proper range.

$\mathbf{Y}(t)$  is a reparameterization of the geodesic w.r.t.  $t$  on Stiefel manifold. When computing  $\mathbf{Y}(t)$ , no SVD is needed. A matrix inversion and several matrix multiplications are needed instead, which sheds light on solving the problem in Eq. (15) in a differentiable way. Note that matrix inversion may also be difficult to differentiate. Fortunately, when  $t$  is in a proper range, we can approximate the matrix inversion by a polynomial of  $\mathbf{P}$ .

*Eigenvector approximation.* To solve the problem in Eq. (15), we compute the gradient  $\mathbf{G}$  of the objective function

$$\sum_{ij} W_{ij} \|\mathbf{Y}_{:,i} - \mathbf{Y}_{:,j}\|_2^2, \quad (18)$$

w.r.t.  $\mathbf{Y}$  and search for a geodesic on the Stiefel manifold in the gradient direction to update the current  $\mathbf{Y}$ .

Note that the matrix inversion  $(\mathbf{I} - t\mathbf{P}/2)^{-1}$  is time-consuming and hard to differentiate during back propagation. Observing that  $\mathbf{C}(t)$  contains Neumann series, we consider approximating  $\mathbf{C}(t)$  by a polynomial in  $\mathbf{P}$ . Hence, given the current  $\mathbf{Y}$ , we consider searching in the following curve:

$$\mathbf{Y}(t) = \left( \mathbf{I} + \sum_{i=1}^r 2^{-\frac{(i-1)}{2}} (-t\mathbf{P})^i \right) \mathbf{Y}, \quad (19)$$

where  $r$  is the degree of the polynomial and  $\mathbf{P} = \mathbf{G}\mathbf{Y}^T - \mathbf{Y}\mathbf{G}^T$ .  $\mathbf{P}\mathbf{Y}$  corresponds to the projection of  $\mathbf{G}$  in the tangent plane at  $\mathbf{Y}$ . In general, the approximation in Eq. (19) is the optimal  $r$ -th order polynomial for maintaining orthogonality.

**Proposition 1** Assume that  $\mathbf{Y}$  is from the Stiefel manifold, i.e.,  $\mathbf{Y}^T \mathbf{Y} = \mathbf{I}$  and  $\|\mathbf{P}\|$  is bounded, then we have

$$\|(\mathbf{Y}(t))^T \mathbf{Y}(t) - \mathbf{I}\| = \mathcal{O}(t^{2r} 2^{-r(r-1)}), \quad (20)$$

where  $\|\cdot\|$  is the matrix spectral norm. Furthermore, given the degree  $r$ ,  $\mathbf{Y}(t)$  in Eq. (19) is the optimal polynomial for maintaining the orthogonality.

Note that the boundness of  $\mathbf{P}$  inherits from  $\mathbf{G}$ , which is the gradient of the objective w.r.t.  $\mathbf{Y}$ . By Proposition 1, we have  $\mathbf{Y}(t) \approx \mathbf{C}(t)\mathbf{Y}$  when  $r$  is large. Moreover, we avoid the matrix inversion here and make the curve absolutely differentiable in  $\mathbf{Y}$ .

Obviously, we have  $d\mathbf{Y}(0)/dt = -\mathbf{P}\mathbf{Y}$  for the polynomial in Eq. (19). However, when  $t > 0$ , the direction of the Cayley transform approximation  $\mathbf{Y}(t)$  in Eq. (19) may deviate from the ideal direction  $-\mathbf{P}\mathbf{Y}$ . Fortunately, we can still ensure the descent of the objective when  $t$  satisfies some conditions.

**Proposition 2** Assume that  $\|\mathcal{P}_{\mathbf{Y}}^{\perp} \mathbf{P}\|_F := c_p < \infty$ , where  $\mathcal{P}_{\mathbf{Y}}^{\perp} = \mathbf{I} - \mathbf{Y}\mathbf{Y}^T$  is the projector of the complementary space of  $\text{Span}\{\mathbf{Y}\}$ . Given  $t \in \mathbb{R}^+$ , and let  $\rho_t := tc_p$ , if  $\rho_t$  satisfies

$$g(\rho_t) \cdot \left( 1 - \text{erf} \left( \frac{2 \ln 2 - 2 \ln \rho_t}{2\sqrt{\ln 2}} \right) \right) + \frac{\rho_t}{\ln 2} < 1, \quad (21)$$

where

$$g(\rho_t) := \frac{2\sqrt{\pi} \exp\left(\frac{(\ln \rho_t)^2}{\ln 2}\right) \ln \rho_t}{\rho_t (\ln 2)^{\frac{3}{2}}}, \quad (22)$$

then we have

$$\langle \mathcal{P}_{\mathcal{T}_{\mathbf{Y}}}(\dot{\mathbf{Y}}(t)), \mathcal{P}_{\mathcal{T}_{\mathbf{Y}}}(-\mathbf{G}) \rangle_c > 0, \quad (23)$$

where  $\mathcal{P}_{\mathcal{T}_{\mathbf{Y}}}(\cdot)$  is the projector of the tangent space of  $\mathbf{Y}$ ,  $\langle \mathbf{A}, \mathbf{B} \rangle_c := \text{Tr}(\mathbf{A}^T \mathcal{P}_{\mathbf{Y}}^{\perp} \mathbf{B})$  is the canonical inner product at the tangent space of  $\mathbf{Y}$ , and  $\mathbf{G}$  is the gradient of the objective.

Equation (23) indicates that  $\mathbf{Y}(t)$  is a descent curve as long as  $t$  is in a wide range. Numerically, the condition in Eq. (21) can be satisfied when  $\rho_t < 0.8$ . We may find the optimal  $t^*$  such that

$$t^* = \underset{0 \leq t \leq \varepsilon}{\text{argmin}} f(t) := \left( \sum_{ij} W_{ij} \|\mathbf{Y}(t)_{:,i} - \mathbf{Y}(t)_{:,j}\|_2^2 \right), \quad (24)$$

where  $\varepsilon$  is a pre-defined parameter to ensure the magnitude of  $t^*$ . Recall that we only concern that the optimal  $t^*$  satisfies the condition in Eq. (21). We expand  $f(t)$  at 0 via Taylor expansion:

$$f(t) = f(0) + f'(0) \cdot t + f''(0) \cdot t^2 + \mathcal{O}(t^2), \quad (25)$$

where  $f'(0)$  and  $f''(0)$  are the first and second order derivatives of  $f(t)$  evaluated at 0, respectively. These two derivatives can be computed efficiently (see Appendixes A, B and C). We can obtain an approximated optimal solution  $t^*$  via

$$t^* = \min\{\varepsilon, \tilde{t}\}, \quad \tilde{t} = -\frac{f'(0)}{f''(0)}. \quad (26)$$

Then we can update  $\mathbf{Y}$  by  $\mathbf{Y}(t^*)$ .

*Differentiable eigenvector mapping layer  $\mathbf{Y}^{(l)}$ .* Based on the above iterative steps for the eigenvector approximation, i.e.,  $\mathbf{Y} \rightarrow \mathbf{Y}(t^*)$ , we introduce the following eigenvector mapping layer, which is generalized from one step for approximation. We set  $r = 2$  for our layers.

Given the last layer output  $\mathbf{Y}^{(l)}$ , we have the following updating rules:

$$\begin{cases} \mathbf{P}^{(l+1)} = 2(\mathbf{L}\mathbf{Y}\mathbf{Y}^T - \mathbf{Y}\mathbf{Y}^T\mathbf{L}) + \mathbf{Y}\mathbf{W}_2^{(l)}\mathbf{Y}^T, \\ \tilde{t}^{(l+1)} = \frac{\text{Tr}(\mathbf{Y}^T\mathbf{L}\mathbf{P}\mathbf{Y})}{\text{Tr}(\mathbf{Y}^T\mathbf{L}\mathbf{P}\mathbf{Y} - \mathbf{Y}^T\mathbf{L}\mathbf{P}^2\mathbf{Y})}, \\ t^* = \min\{\varepsilon, \tilde{t}^{(l+1)}\}, \\ \mathbf{Y}^{(l+1)} = (\mathbf{I} - t^*\mathbf{P} + \frac{(t^*\mathbf{P})^2}{2}\mathbf{P})\mathbf{Y}, \end{cases} \quad (27)$$

where  $\mathbf{W}_2^{(l)} \in \mathbb{R}^{\tilde{k} \times \tilde{k}}$  is the learnable matrix at each layer,  $\text{Tr}(\cdot)$  is the trace operator,  $\varepsilon$  is a parameter used to ensure Eq. (21) for  $t^*$ , and  $\mathbf{L}$  is the graph Laplacian matrix of  $\mathbf{W}$ . We randomly choose  $\mathbf{Y}^{(0)}$  from the Stiefel manifold and fix it during training. We omit the superscript in the updating rules for convenience and clear writing. Note that when  $\mathbf{W}_2^{(l)} = \mathbf{0}$ , Eq. (27) is almost the same as one iteration of the manifold gradient descent, the introduced learnable parameters can help to find better updating direction when the gradient is inaccurate.

We bypass the matrix inversion and the exact SVD, and aim to solve the problem in Eq. (15) approximately. Consequently, we can solve the problem in Eq. (15) in a differentiable and learning-based manner by stacking the proposed eigenvector mapping layers in Eq. (27). Our eigenvector mapping layer can be easily modified to perform differentiable SVD, which is still difficult and is the main problem when connecting classic low rank-based structure methods and prevalent DNNs.

## 5.2 Differentiable $k$ -means clustering

This section provides a method to differentiate  $k$ -means clustering by a self-supervised strategy. The input of this module is  $\mathbf{Y}^{(l)}$ , which is the output of the eigenvector mapping layers.

In general, batch training makes the  $k$ -means clustering challenging to differentiate. The sampled data from different batches do not have the same eigenvector space. Thus, they cannot share cluster centers or parameters. We adopt a parametric function  $\mathcal{G}_\eta: \mathbb{R}^{\tilde{k}} \rightarrow \mathbb{R}^{\tilde{k}}$  to further transform the input  $\mathbf{Y}^{(l)}$ , where  $\eta$  is the learnable parameter of the

parametric function. We want the function  $\mathcal{G}_\eta$  to embed the eigenvectors of different batches into a common feature space, where we can share the clusters and distance metric. In the experiments, we choose  $\mathcal{G}_\eta$  as a three-layer fully connected neural network.

On the other hand,  $k$ -means clustering assigns data to a cluster center, which is a discrete process. To overcome the non-differentiability, we utilize a self-supervised structure [33] to transform the clustering step into a classification step. Specifically, we alternate between clustering the input of this module to produce pseudo-labels using  $k$ -means and updating the parameters of this differentiable module by predicting these pseudo-labels. This self-supervised structure is illustrated in Fig. 1. In summary, we convert the non-differentiable assignment process into a differentiable classifier via the self-supervised structure.

## 5.3 Training objective

We provide the training objective for the two modules of the proposed differentiable spectral clustering. For the eigenvector mapping layers, we let

$$L_e = \text{Tr}((\mathbf{Y}^{(\tilde{l})})^T \mathbf{L} (\mathbf{Y}^{(\tilde{l})})), \quad (28)$$

where  $\mathbf{Y}^{(\tilde{l})}$  is the output of the final eigenvector mapping layer and  $\mathbf{L}$  is the graph Laplacian matrix. Note that  $L_e$  is the same as the objective of the problem in Eq. (15). Due to the property 4) of  $\mathbf{Y}(t)$ ,  $\mathbf{Y}^{(\tilde{l})}$  approximately satisfies the constraint in Eq. (15).

We define  $L_K$  as the loss of the  $k$ -means module. Denote by  $\tilde{y}_i \in \mathbb{N}$  the pseudo-label of the data  $\mathbf{x}_i \in \mathbb{R}^d$  and let  $\mathbf{f}_i \in \mathbb{R}^{\tilde{k}}$  be the final output of the classifier in terms of  $\mathbf{x}_i$  in the  $k$ -means module, where  $\tilde{k}$  is the number of cluster. We utilize the softmax loss for the multi-classification problem for  $L_K$

$$L_K = - \sum_{i=1}^n \ln \left( \frac{\exp(\mathbf{f}_{i, \tilde{y}_i} - \max\{\mathbf{f}_i\})}{\sum_{j=1}^{\tilde{k}} \exp(\mathbf{f}_{i,j} - \max\{\mathbf{f}_i\})} \right), \quad (29)$$

where  $\mathbf{f}_{i,j}$  is the  $j$ -th entry of the vector  $\mathbf{f}_i$  and  $\max\{\mathbf{f}_i\}$  is the maximal entry of this vector.  $L_K$  is smooth and easy to differentiate. Notably, pseudo-label  $\tilde{y}_i$  is obtained by performing  $k$ -means clustering on  $\mathcal{G}_\eta(\mathbf{Y}^{(\tilde{l})})$  row-wisely, and  $\mathbf{f}_i$  is the output of a two-layer classifier with  $\mathcal{G}_\eta(\mathbf{Y}^{(\tilde{l})})$  being the input.

## 6 Training algorithm and complexity

### 6.1 Training algorithm

We recast all three steps of subspace clustering as differential modules and propose the SSCNet. One of the main differences between our proposed unified differentiable modules and traditional deep learning lies in the training algorithm. In most cases, other deep learning methods design a final loss function and use it to update all



the network parameters in one iteration. In contrast, following the commonly used ideas of pre-training and fine-tuning, we assign each module an objective to retain the interpretability of the model. The training algorithm for the proposed SSCNet is outlined in Algorithm 1. We denote the learnable parameters of the differentiable modules of SSCNet as  $\Theta_1$ ,  $\Theta_2$  and  $\Theta_3$ , respectively. We update the parameters of differentiable SSC more frequently because it is the first differentiable module with the main parameters. We solve all the optimization problems in this algorithm via SGD.

Note that the non-differentiable *KNN* operator in graph construction does not harm the learning. Based on the proposed sparse graph matrix, SSCNet concerns only the local structure when learning the feature  $\mathbf{Y}^{(i)}$ . The local distance is more significant than the global distance for the

**Algorithm 1** Training algorithm for the proposed SSCNet

**Input:** Data matrix  $\mathbf{X}$ , parameter  $\gamma_{\max}$ ,  $\lambda$ ,  $c_1$ ,  $c_2$ ,  $\epsilon$ ,  $\varepsilon$ , and an orthogonal matrix  $\mathbf{Y}^{(0)}$  randomly chosen from the Stiefel manifold.

- 1: **while** not converged **do**
- 2: Sample a batch data from  $\mathbf{X}$  and obtain the pseudo-labels of the batches by a forward propagation;
- 3: Train the differentiable modules:
  - if**  $\gamma \bmod 3 = 0$ ; solve:  $\min_{\Theta_1} L_{D\text{-SSC}}$ ;
  - if**  $\gamma \bmod 3 = 1$ ; solve:  $\min_{\Theta_1, \Theta_2} L_{D\text{-SSC}} + c_1 L_e$ ;
  - if**  $\gamma \bmod 3 = 2$ ; solve:  $\min_{\Theta_1, \Theta_2, \Theta_3} L_{D\text{-SSC}} + c_1 L_e + c_2 L_K$ ;
- 4: Check the convergence conditions:  $|L_{D\text{-SSC}}^{t+1} - L_{D\text{-SSC}}^t| + |L_e^{t+1} - L_e^t| + |L_K^{t+1} - L_K^t| \leq \epsilon$  or  $\gamma > \gamma_{\max}$ ;
- 5: **If** not converged, set iteration number  $\gamma = \gamma + 1$ .
- 6: **end while**

**Output:**  $\mathbf{Z}^K$ ,  $\mathbf{E}^K$  and pseudo-labels.

next module—*k*-means clustering. Usually, large spacing among clusters often leads to better results.

**6.2 Complexity analysis**

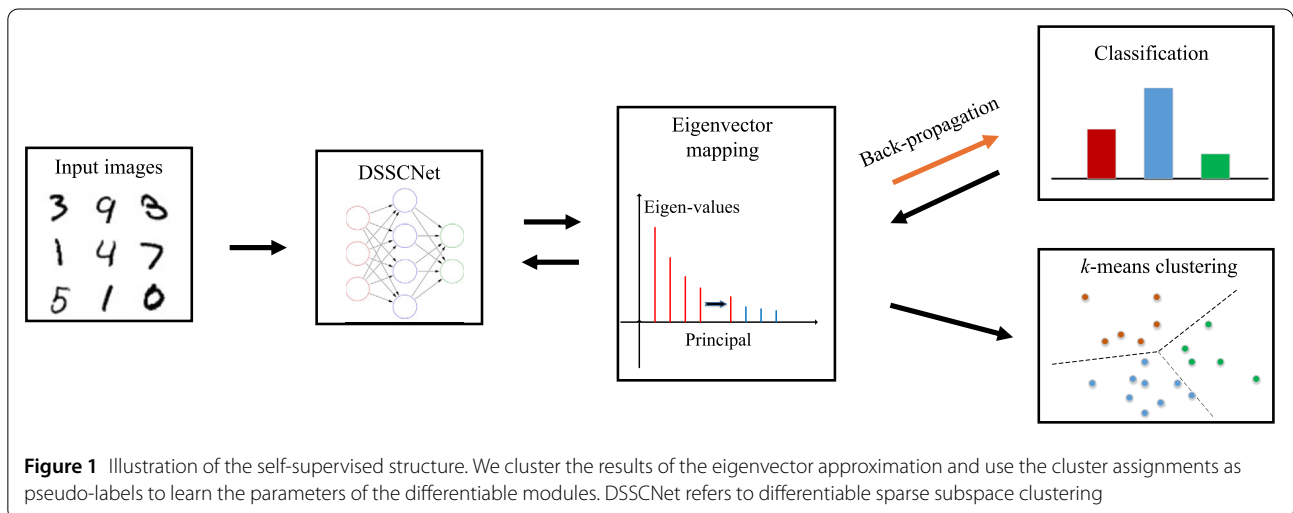
With the new data coming in, our SSCNet can provide their labels by forward propagation, while some traditional methods, such as LRR and SSC, need to perform the optimization again.

The computational complexity equals the consumption of a forward step. The data size is  $n$ , the clustering number is  $\tilde{k}$ , and  $\mathbf{Z} \in \mathbb{R}^{m \times n}$ . In our paper,  $\mathcal{A}_{\theta_1}(\cdot)$  and  $\mathcal{A}_{\theta_2}^T(\cdot)$  are shallow convolutional networks with  $3 \times 3$  kernels. Considering element-wise operations, the computational complexity of one block of our differentiable SSC module is  $O(ndm)$ . For the eigenvector mapping module, it stacks the layers in Eq. (27), and each step of it consumes  $\tilde{k}n^2$  FLOPs. Hence, the computational complexity of this module is  $O(\tilde{k}n^2)$ . As shown in Fig. 1, the last *k*-means clustering module during the testing phase is a shallow classifier with  $\mathcal{G}_\eta(\mathbf{Y}^{(i)})$  being the input. Assuming that the shallow neural networks are in the width  $O(h)$ , the last module consumes  $O(h^2 + h\tilde{k})$  FLOPs. In summary, the total complexity is  $O(ndm + \tilde{k}n^2 + h^2 + h\tilde{k})$ .

The computational complexity of one block of D-SSC is the same as that of one iteration of L-ADMM, which is  $O(ndm)$ . The results of D-LADMM [22] indicate that the block number of D-LADMM is much smaller than the total iteration number of L-ADMM. Thus, the total complexity of one forward step of SSCNet is much lower than that of SSC.

**7 Experiments**

In this section, we verify the effectiveness of our SSCNet for clustering. Detailed comparisons with other methods and analyses are provided.



**Figure 1** Illustration of the self-supervised structure. We cluster the results of the eigenvector approximation and use the cluster assignments as pseudo-labels to learn the parameters of the differentiable modules. DSSCNet refers to differentiable sparse subspace clustering

## 7.1 Experiment settings

### 7.1.1 Datasets

To evaluate the performance of our proposed methods, we conduct experiments on three commonly used datasets, namely the MNIST [48], the USPS, and the CIFAR-10 [49] datasets.

*MNIST* [48]. The MNIST dataset contains a total 70,000 handwritten digits of 10 classes. Each image is in the size of  $28 \times 28$ . The digits are centered and size-normalized. In experiments, we adopt all images for clustering.

*USPS*. The USPS dataset consists of 9298 handwritten digits of 10 classes. Each image is  $16 \times 16$  in size, and the pixel values are in the range of  $[0, 2]$ .

*CIFAR-10* [49]. The CIFAR-10 dataset has 10 classes of objects. It contains a total of 60,000 color images of size  $32 \times 32$ . We also adopt the entire dataset.

### 7.1.2 Method comparison

We compare our SSCNet with many state-of-the-art methods, including  $k$ -means, spectral embedded clustering (SEC) [50], autoencoder based  $k$ -means (AE +  $k$ -means) [51], deep embedded clustering (DEC) [6], improved DEC (IDEC) [10], joint unsupervised learning (JULE) [7], cascade subspace clustering (CSC) [9], deep subspace clustering (DSC) [12], and SpectralNet [32].

### 7.1.3 Evaluation metrics

We adopt two commonly used metrics, including the clustering accuracy (ACC) and the normalized mutual information (NMI) [52], to measure the performance. The  $\text{NMI}(C, C')$  can be information-theoretically interpreted. It is defined by

$$\begin{aligned} \text{NMI}(C, C') & \\ & := \frac{\sum_{i=1}^K \sum_{j=1}^S |C_i \cap C'_j| \log \frac{N|C_i \cap C'_j|}{|C_i||C'_j|}}{\sqrt{(\sum_{i=1}^K |C_i| \log \frac{C_i}{N})(\sum_{j=1}^S |C'_j| \log \frac{C'_j}{N})}}, \end{aligned} \quad (30)$$

where  $C$  and  $C'$  represent the predicted partition and the ground truth partition, respectively.

For the ACC, we first need to map clusters to the corresponding ground truth labels by the best permutation mapping function  $\text{map}(\cdot)$  obtained by the Hungarian algorithm [53]. The accuracy is defined as follows:

$$\text{ACC} = \frac{\sum_{i=1}^N \delta(c_i, \text{map}(r_i))}{N}, \quad (31)$$

where  $c_i$  is the ground truth label,  $r_i$  is the predicted label, and

$$\delta(a, b) = \begin{cases} 1, & \text{if } a = b, \\ 0, & \text{otherwise.} \end{cases} \quad (32)$$

For these two metrics, the higher value represents better performance.

### 7.1.4 Network architecture

We use a convolutional neural network with three layers for each block's nonlinear mapping function  $\mathcal{A}_{\theta_1}$ . The kernel size is set to  $3 \times 3$ , and the numbers of feature maps in each layer are 32, 64, and 128. The corresponding generalized adjoint mapping function  $\mathcal{A}_{\theta_2}$  is designed symmetrically. In addition, we set  $\lambda = c_1 = c_2 = 0.15$ , and  $\varepsilon = 1.0 \times 10^{-2}$ . We use SGD with a learning rate  $lr = 3.0 \times 10^{-4}$  to train the network, and we set the batch size to 256 for all datasets. Our implementation is based on Python and TensorFlow [54].

## 7.2 Performance comparison

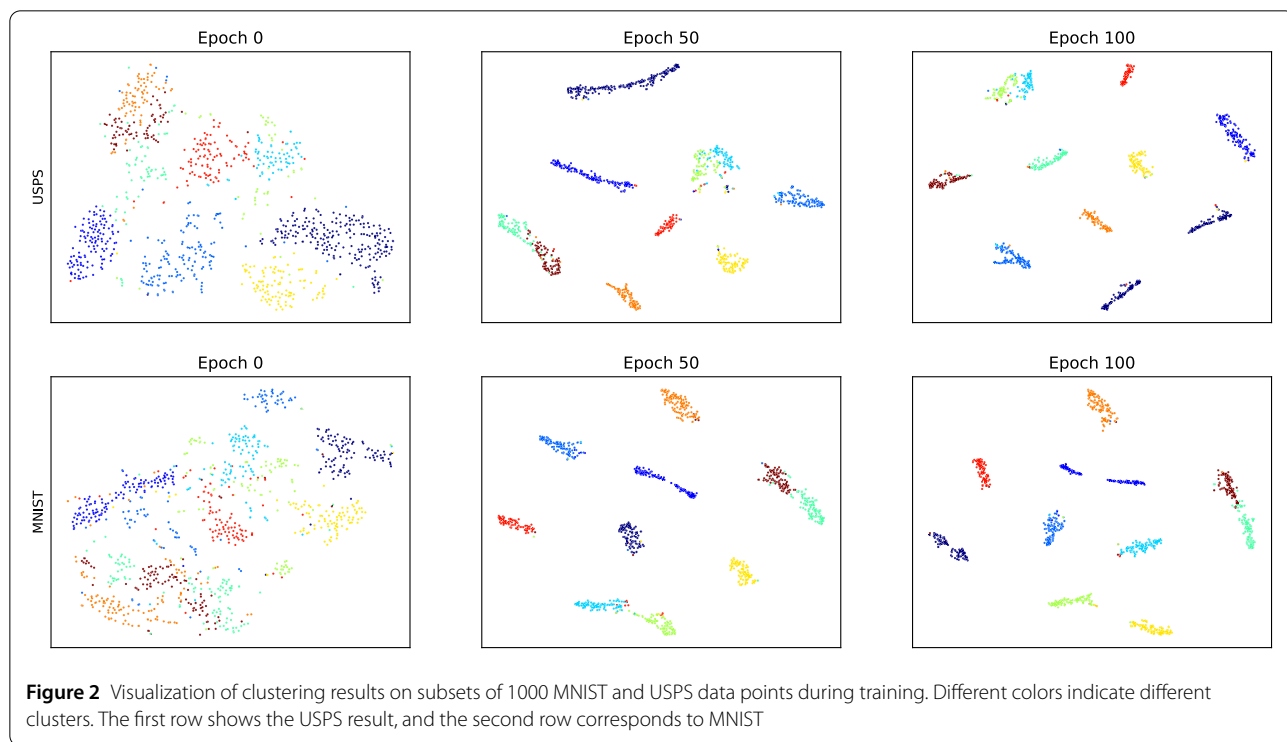
In Table 1, we present the results of these related approaches on these three datasets. Some of the results are taken directly from their papers. Our method performs best on all these datasets under the ACC and NMI metrics. For example, on the USPS dataset, the NMI of our method is 0.9482, which is 1.6% higher than the second best result 0.9321 achieved by SpectralNet [32], while the third best result is 0.9130 of JULE [7]. Compared with traditional clustering methods, including  $k$ -means and SEC, these deep learning-based methods show much better results due to the powerful representation ability of deep neural networks. In addition, we can see that DEC achieves much better results than AE +  $k$ -means on all the datasets, which reflects the importance of joint learning. Our method can jointly learn the optimal parameters for dictionary construction, affinity matrix computing, and clustering, so it achieves outstanding performance.

All the clustering methods exhibit weaker performance on the CIFAR-10 dataset than on the other datasets. It is essential to emphasize that most unsupervised clustering methods employed in this study operate without the benefit of labeled data during training. The complex semantic information in CIFAR-10 images introduces additional complexity, which may challenge traditional clustering techniques. These methods tend to rely heavily on low-level features, such as color, which can lead to misclassifications, such as grouping images of grassy landscapes and animals in grassy environments.

The complexity of CIFAR-10 poses a significant challenge for unsupervised clustering and traditional methods. Our approach, which autonomously learns high-level semantic features, surpasses traditional clustering techniques when dealing with such complex datasets. However, while our method mitigates these challenges, it cannot completely overcome the inherent limitations of clustering methods and may not entirely eliminate feature-to-semantic mismatches.

**Table 1** Experimental results on the MNIST, USPS, and CIFAR-10 datasets. The best results are in bold. NMI and ACC represent normalized mutual information and accuracy, respectively

Methods	MNIST		USPS		CIFAR-10	
	NMI	ACC	NMI	ACC	NMI	ACC
<i>k</i> -means	0.4997	0.5723	0.6310	0.6682	0.0871	0.2289
SSC	0.6937	0.6765	0.6532	0.6174	0.0610	0.1982
SEC [50]	0.7275	0.8037	0.7266	0.7420	–	–
AE + <i>k</i> -means [51]	0.7473	0.8182	0.6620	0.6931	0.2468	0.2937
DEC [6]	0.8372	0.8655	0.7529	0.7408	0.2568	0.3010
IDEC [10]	0.8672	0.8806	0.7846	0.7650	0.2675	0.3097
JULE [7]	0.9130	–	0.9130	–	0.1923	0.2715
DSC [12]	0.8816	0.8639	0.8254	0.8063	–	–
CSC [9]	0.7550	0.8716	–	–	0.070	0.2188
SpectralNet [32]	0.9239	0.9709	0.9321	0.9800	0.2311	0.3004
Ours	<b>0.9349</b>	<b>0.9752</b>	<b>0.9482</b>	<b>0.9890</b>	<b>0.3041</b>	<b>0.3515</b>



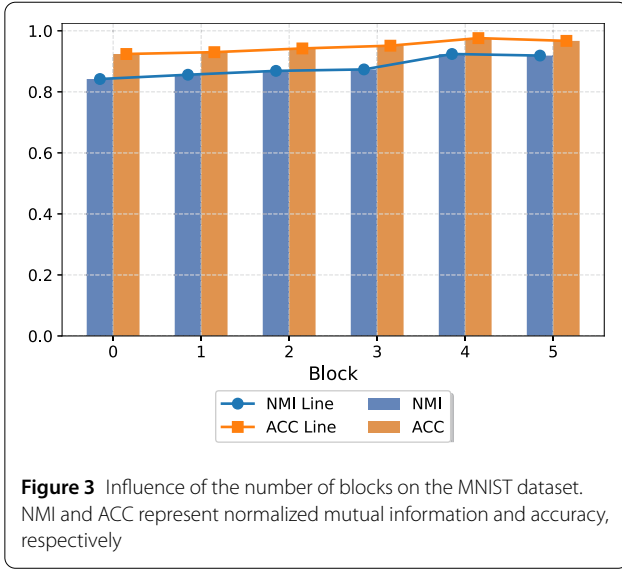
### 7.2.1 Visualization

In Fig. 2, we visualize the clustering results with 1000 data points from the MNIST and the USPS datasets during training. We can observe that our proposed SSCNet converges quickly even though each module has its objective. The points of both datasets are well separated. SSCNet increases the separability of the results. Note the points with red and blue colors in epoch 50 of the USPS dataset; they are slightly mixed but separated in epoch 100.

### 7.3 Influence of blocks

In Fig. 3, we show the influence of the block number *b* on the results of the MNIST dataset. One block is generated from one iteration of the L-ADMM. The perfor-

mance is improved with increased blocks initially. This phenomenon is consistent with the optimization process of traditional SSC, whose objective value decreases with increasing iteration number. The network tends to be stable when *b* is greater than 4. Therefore, we set the number of blocks for all the datasets to 4. Another observation is that when *b* = 0, the differentiable SSC module of our network is a variant of the autoencoder since we let  $\mathbf{Z}^{(1)} = \mathcal{A}_{\vartheta_2}^T(\mathcal{A}_{\vartheta_1}(\mathbf{X}))$ . However, the performance is much better than AE + *k*-means. This phenomenon verifies that our other differentiable modules benefit the clustering, i.e., eigenvector approximation and self-supervised *k*-means. We can also conclude that the unified DP model can ob-



tain better results than separated traditional deep learning models even without our D-LADMM structure. Specifically, the NMI of our method is 0.8421 when  $b = 0$ , which is 0.0948 higher than that of AE +  $k$ -means. Our D-LADMM framework further improves the NMI to 0.9349 when  $b = 4$ . These results demonstrate the superiority of the joint learning strategy and the proposed DP structure.

#### 7.4 Discussion for exploring applications with larger datasets

A significant consideration is the practical applicability of our method to larger datasets. While our experiments use moderate-sized datasets, it is essential to consider their potential for dealing with larger and real-world datasets. As demonstrated in the algorithm complexity section, our method surpasses existing techniques in terms of efficiency and supports incremental learning for continued training on new data. However, one computational bottleneck that remains consistent with the original SSC algorithm is the step resembling eigenvalue decomposition, with a complexity of  $O(kn^2)$ , where  $k$  signifies the number of clusters and  $n$  represents the dataset size. This computational aspect could be a limitation when applied to larger datasets. Our current study primarily focuses on differentiable programming. Hence, we maintain alignment with the original SSC algorithm w.r.t. the algorithm steps.

In addressing this scalability issue for larger datasets, a promising avenue for future research would involve the replacement of graph-based spectral clustering, which exhibits second-order complexity, with an efficient first-order clustering method. This strategic change could overcome the scalability limitations observed in our current implementation and open up new possibilities for applying our method to large-scale datasets in practice.

## 8 Conclusions

In this paper, we propose a novel SSCNet to address the existing limitations of SSC. We first recast the optimization step of the L-ADMM as a multi-block deep neural network. We then apply this approach to the SSC problem, learning dictionary, and affinity matrix. Second, a spectral embedding network is used to approximate the eigenvalue decomposition. A general and novel differentiable eigenvector mapping layer is proposed that can be applied to other problems. Finally, we adopt a self-supervised structure to overcome the non-differentiability of  $k$ -means. Experiments validate the effectiveness of our SSCNet.

## Appendix A: Derivatives of $f(t)$

Note that

$$f(t) = \text{Tr}(\mathbf{Y}(t)^T \mathbf{L} \mathbf{Y}(t)).$$

Hence, we have

$$\frac{d}{d\mathbf{Y}(t)} f(t) = \mathbf{Y}(t)^T (\mathbf{L} + \mathbf{L}^T) = 2\mathbf{Y}(t)^T \mathbf{L}.$$

On the other hand, we have

$$\frac{d}{dt} \mathbf{Y}(t) = \frac{d}{dt} \left( \mathbf{I} - t\mathbf{P} + \frac{t^2 \mathbf{P}^2}{2} \right) \mathbf{Y} = -(\mathbf{I} - t\mathbf{P})\mathbf{P}\mathbf{Y}.$$

### A.1 First-order derivative

By the chain rule, we can easily obtain the first-order derivative  $f'(t)$ :

$$f'(t) = \text{Tr} \left( \frac{df(t)}{d\mathbf{Y}(t)} \frac{d}{dt} \mathbf{Y}(t) \right) = -2 \text{Tr}(\mathbf{Y}(t)^T \mathbf{L} (\mathbf{I} - t\mathbf{P})\mathbf{P}\mathbf{Y}).$$

Recall that

$$\mathbf{P} = \mathbf{G}\mathbf{Y}^T - \mathbf{Y}\mathbf{G}^T, \quad \mathbf{G} = 2\mathbf{L}\mathbf{Y}.$$

Then, we can obtain

$$f'(0) = -2 \text{Tr}(\mathbf{Y}^T \mathbf{L} \mathbf{P} \mathbf{Y}).$$

### A.2 Second-order derivative

We have

$$\begin{aligned} f''(t) &= \frac{df'(t)}{dt} \\ &= -2 \text{Tr} \left( \left( \frac{d}{dt} \mathbf{Y}(t) \right)^T \mathbf{L} (\mathbf{I} - t\mathbf{P})\mathbf{P}\mathbf{Y} \right. \\ &\quad \left. - \mathbf{Y}(t)^T \mathbf{L} \mathbf{P}^2 \mathbf{Y} \right) \end{aligned}$$



$$= 2 \text{Tr}((\mathbf{I} - t\mathbf{P})\mathbf{P}\mathbf{Y})^T \mathbf{L}(\mathbf{I} - t\mathbf{P})\mathbf{P}\mathbf{Y} + \mathbf{Y}(t)^T \mathbf{L}\mathbf{P}^2\mathbf{Y}.$$

Hence, we can further obtain

$$f''(0) = 2 \text{Tr}(-\mathbf{Y}^T \mathbf{P}\mathbf{L}\mathbf{P}\mathbf{Y} + \mathbf{Y}^T \mathbf{L}\mathbf{P}^2\mathbf{Y}).$$

We finish the calculation.

### Appendix B: Proof of Proposition 1

Due to the skew-symmetry of  $\mathbf{P}$ , we can obtain

$$\mathbf{Y}(t)^T = \left( \mathbf{I} + \sum_{i=1}^r 2^{-\frac{(i-1)i}{2}} (-1)^{i+1} (t\mathbf{P})^i \right) \mathbf{Y},$$

then, we can verify that

$$(\mathbf{Y}(t))^T \mathbf{Y}(t) = \mathbf{Y}^T (\mathbf{I} + 2^{-r(r-1)} t^{2r} \mathbf{P}^{2r}) \mathbf{Y}.$$

We can see that all the terms with a degree less than  $2^r$  have zero coefficients. The optimality is obvious. Moreover, when  $\mathbf{P}$  is bounded, we can obtain

$$\|\mathbf{Y}^T (2^{-r(r-1)} t^{2r} \mathbf{P}^{2r}) \mathbf{Y}\| = \mathcal{O}(2^{-r(r-1)} t^{2r}).$$

We now finish the proof.

### Appendix C: Proof of Proposition 2

From Eq. (19), we have

$$\frac{d}{dt} \mathbf{Y}(t) = \left( -\mathbf{P} + \sum_{i=2}^r 2^{-\frac{(i-1)i}{2}} i(-\mathbf{P})^i t^{i-1} \right) \mathbf{Y}.$$

We denote by

$$\mathbf{R} = \left( \sum_{i=2}^r 2^{-\frac{(i-1)i}{2}} i(-\mathbf{P})^i t^{i-1} \right).$$

Due to the skew symmetry of  $\mathbf{P}$ , we have  $\mathbf{R} = -\mathbf{R}^T$ . Then

$$(-\mathbf{P} + \mathbf{R})^T = \mathbf{P} - \mathbf{R}.$$

We now project  $\frac{d}{dt} \mathbf{Y}(t)$  to the tangent space of  $\mathbf{Y}$

$$\begin{aligned} \mathcal{P}_{\mathcal{T}_{\mathbf{Y}}}(\dot{\mathbf{Y}}(t)) &= \dot{\mathbf{Y}}(t) - \mathbf{Y}(\dot{\mathbf{Y}}(t))^T \mathbf{Y} \\ &= (-\mathbf{P}\mathbf{Y} + \mathbf{R}\mathbf{Y}) - \mathbf{Y}(\mathbf{Y}^T \mathbf{P} - \mathbf{Y}^T \mathbf{R}) \mathbf{Y} \\ &= (\mathbf{I} + \mathbf{Y}\mathbf{Y}^T)(-\mathbf{P}\mathbf{Y}) + \mathcal{P}_{\mathcal{T}_{\mathbf{Y}}}(\mathbf{R}\mathbf{Y}). \end{aligned}$$

Note that we already have

$$\mathcal{P}_{\mathcal{T}_{\mathbf{Y}}}(-\mathbf{G}) = -\mathbf{P}\mathbf{Y}.$$

Then we obtain

$$\begin{aligned} &\langle \mathcal{P}_{\mathcal{T}_{\mathbf{Y}}}(\dot{\mathbf{Y}}(t)), \mathcal{P}_{\mathcal{T}_{\mathbf{Y}}}(-\mathbf{G}) \rangle_c \\ &= \text{Tr}(\mathcal{P}_{\mathcal{T}_{\mathbf{Y}}}(-\mathbf{G}) \mathcal{P}_{\mathcal{T}_{\mathbf{Y}}}^\perp \mathcal{P}_{\mathcal{T}_{\mathbf{Y}}}(\dot{\mathbf{Y}}(t))) \\ &= \text{Tr}(-\mathbf{Y}^T \mathbf{P}^T \mathcal{P}_{\mathcal{T}_{\mathbf{Y}}}^\perp (-\mathbf{P}\mathbf{Y} + \mathbf{R}\mathbf{Y})) \\ &= \|\mathbf{P}\mathbf{Y}\|_c - \langle \mathbf{P}\mathbf{Y}, \mathbf{R}\mathbf{Y} \rangle_c. \end{aligned}$$

Hence, it is obvious that when

$$\|\mathcal{P}_{\mathcal{T}_{\mathbf{Y}}}^\perp(\mathbf{R})\|_F < \|\mathcal{P}_{\mathcal{T}_{\mathbf{Y}}}^\perp(\mathbf{P})\|_F := c_p, \tag{C.1}$$

we have

$$\langle \mathcal{P}_{\mathcal{T}_{\mathbf{Y}}}(\dot{\mathbf{Y}}(t)), \mathcal{P}_{\mathcal{T}_{\mathbf{Y}}}(-\mathbf{G}) \rangle_c > 0.$$

Then, we obtain

$$\begin{aligned} &\|\mathcal{P}_{\mathcal{T}_{\mathbf{Y}}}^\perp(\mathbf{R})\|_F \\ &= \left\| \sum_{i=2}^r 2^{-\frac{(i-1)i}{2}} i(-\mathcal{P}_{\mathcal{T}_{\mathbf{Y}}}^\perp(\mathbf{P}))^i t^{i-1} \right\| \\ &\leq \sum_{i=2}^r 2^{-\frac{(i-1)i}{2}} i c_p^i t^{i-1} \leq c_p \sum_{i=2}^r 2^{-\frac{i^2}{4}} i (c_p t)^{i-1} \\ &\leq c_p \int_2^\infty 2^{-\frac{x^2}{4}} x (c_p t)^{x-1} dx \\ &= c_p \cdot g(t) \cdot \left( 1 - \text{erf}\left(\frac{2 \ln 2 - 2 \ln t}{2\sqrt{\ln 2}}\right) \right) + \frac{t}{\ln 2}, \end{aligned} \tag{C.2}$$

where the last step is given in the following section,  $\rho_t = c_p t$  and

$$g(t) := \frac{2\sqrt{\pi} \exp\left(\frac{\ln^2 t}{\ln 2}\right) \ln t}{t \ln^{\frac{3}{2}} 2}.$$

A necessary condition for the inequality in Eq. (C.1) is

$$g(t) \cdot \left( 1 - \text{erf}\left(\frac{2 \ln 2 - 2 \ln t}{2\sqrt{\ln 2}}\right) \right) + \frac{t}{\ln 2} < 1,$$

which implies the condition in Proposition 2. We finish the proof.

### C.3 Integral calculations for Eq. (C.2)

We have

$$\int \frac{x t^{x-1}}{2^{\frac{x^2}{4}}} dx = \frac{1}{t} \int \frac{x t^x}{2^{\frac{x^2}{4}}} dx.$$

Note that

$$\frac{dt^x}{dx} = \ln(t) t^x, \quad \int \frac{x}{2^{\frac{x^2}{4}}} dx = -\frac{2^{1-\frac{x^2}{4}}}{\ln 2}.$$

Integrating by parts, we obtain

$$\int \frac{xt^x}{2^{\frac{x^2}{4}}} dx = -\frac{t^x \cdot 2^{1-\frac{x^2}{4}}}{\ln 2} + \frac{2 \ln t}{\ln 2} \int \frac{t^x}{2^{\frac{x^2}{4}}} dx.$$

Now, we aim to solve the following

$$\begin{aligned} \int \frac{t^x}{2^{\frac{x^2}{4}}} dx &= \int \exp\left(\ln tx - \frac{\ln 2x^2}{4}\right) dx \\ &= \int \exp\left(\frac{\ln^2 t}{\ln 2} - \left(\frac{\sqrt{\ln 2}x}{2} - \frac{\ln t}{\sqrt{\ln 2}}\right)^2\right) dx. \end{aligned}$$

We substitute

$$u = \frac{\ln 2x - 2 \ln t}{2\sqrt{\ln 2}},$$

then we have

$$\begin{aligned} \int \frac{t^x}{2^{\frac{x^2}{4}}} dx &= \int \frac{2 \exp\left(\frac{\ln^2 t}{\ln 2} - u^2\right)}{\sqrt{\ln 2}} du \\ &= \frac{\sqrt{\pi} \exp\left(\frac{\ln^2 t}{\ln 2}\right)}{\sqrt{\ln 2}} \int \frac{2 \exp(-u^2)}{\sqrt{\pi}} du \\ &= \frac{\sqrt{\pi} \exp\left(\frac{\ln^2 t}{\ln 2}\right) \operatorname{erf}(u)}{\sqrt{\ln 2}} \\ &= \frac{\sqrt{\pi} \exp\left(\frac{\ln^2 t}{\ln 2}\right) \operatorname{erf}\left(\frac{\ln 2x - 2 \ln t}{2\sqrt{\ln 2}}\right)}{\sqrt{\ln 2}}. \end{aligned}$$

Then, we can conclude that

$$\begin{aligned} \int \frac{xt^{x-1}}{2^{\frac{x^2}{4}}} dx &= -\frac{t^x \cdot 2^{1-\frac{x^2}{4}}}{t \ln 2} \\ &\quad + \frac{2 \ln t}{t \ln 2} \frac{\sqrt{\pi} \exp\left(\frac{\ln^2 t}{\ln 2}\right) \operatorname{erf}\left(\frac{\ln 2x - 2 \ln t}{2\sqrt{\ln 2}}\right)}{\sqrt{\ln 2}} \\ &= \frac{2\sqrt{\pi} \exp\left(\frac{\ln^2 t}{\ln 2}\right) \ln t \operatorname{erf}\left(\frac{\ln 2x - 2 \ln t}{2\sqrt{\ln 2}}\right)}{t \ln^{\frac{3}{2}}(2)} \\ &\quad - \frac{t^{x-1} \cdot 2^{1-\frac{x^2}{4}}}{\ln 2}. \end{aligned}$$

Due to

$$\frac{t^{x-1} \cdot 2^{1-\frac{x^2}{4}}}{\ln 2} \rightarrow 0, \quad \text{as } |x| \rightarrow \infty,$$

we have

$$\begin{aligned} \int_{-\infty}^{\infty} \frac{xt^{x-1}}{2^{\frac{x^2}{4}}} dx &= g(t) \operatorname{erf}\left(\frac{\ln 2x - 2 \ln t}{2\sqrt{\ln 2}}\right) \Big|_2^{\infty} \\ &= g(t) \left(1 - \operatorname{erf}\left(\frac{2 \ln 2 - 2 \ln t}{2\sqrt{\ln 2}}\right)\right) + \frac{t}{\ln 2}, \end{aligned}$$

where

$$g(t) := \frac{2\sqrt{\pi} \exp\left(\frac{\ln^2 t}{\ln 2}\right) \ln t}{t \ln^{\frac{3}{2}}(2)}.$$

We have now finished the calculation.

#### Funding

This paper was supported by the National Natural Science Foundation of China (No. 62276004), the major key project of Pengcheng Laboratory, China (No. PCL2021A12) and Qualcomm.

#### Abbreviations

ADMM, alternating direction method of multipliers; D-LADMM, differentiable linearized ADMM; DP, differential programming; GD, gradient descent; KNN,  $k$ -nearest neighbor; L-ADMM, linearized ADMM; LSTM, long short-term memory; LRR, low-rank representation; NMI, normalized mutual information; SGD, stochastic gradient descent; SVD, singular value decomposition; SSC, sparse subspace clustering.

#### Data availability

The datasets analyzed during the current study are available in MNIST [48], the USPS (<https://cs.nyu.edu/~roweis/data.html>), and the CIFAR-10 [49].

#### Declarations

##### Competing interests

Guangcan Liu is an Associate Editor for Visual Intelligence and was not involved in the editorial review of, or the decision to publish, this article. The authors declare that there are no other competing interests.

##### Author contributions

JW collected the data and performed the experiments. XX performed the data analysis and drafted the manuscript. GL and ZL contributed substantially to the analysis with constructive discussions and revised the manuscript. All authors read and approved the final manuscript.

##### Author details

<sup>1</sup>National Key Lab of General AI, School of Intelligence Science and Technology, Peking University, Beijing, China. <sup>2</sup>School of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), Shenzhen, China. <sup>3</sup>School of Automation, Southeast University, Nanjing, China. <sup>4</sup>Institute for Artificial Intelligence, Peking University, Beijing, China. <sup>5</sup>Peng Cheng Laboratory, Shenzhen, China.

Received: 30 June 2023 Revised: 11 March 2024 Accepted: 11 March 2024  
Published online: 08 April 2024

#### References

- Ehsan, E., & Vidal, R. (2009). Sparse subspace clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2790–2797). Piscataway: IEEE.
- Liu, G., Lin, Z., & Yu, Y. (2010). Robust subspace segmentation by low-rank representation. In J. Fürnkranz & T. Joachims (Eds.), *Proceedings of the 27th international conference on machine learning* (pp. 663–670). Strouds: International Machine Learning Society.
- Liu, G., Lin, Z., Yan, S., Sun, J., Yu, Y., & Ma, Y. (2013). Robust recovery of subspace structures by low-rank representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1), 171–184.

4. Xie, X., Guo, X., Liu, G., & Wang, J. (2018). Implicit block diagonal low-rank representation. *IEEE Transactions on Image Processing*, 27(1), 477–489.
5. Ng, A. Y., Jordan, M. I., & Weiss, Y. (2001). On spectral clustering: analysis and an algorithm. In T. G. Dietterich, S. Becker, & Z. Ghahramani (Eds.), *Proceedings of the 14th international conference on neural information processing systems* (pp. 849–856). Red Hook: Curran Associates.
6. Xie, J., Girshick, R. B., & Farhadi, A. (2016). Unsupervised deep embedding for clustering analysis. In M.-F. Balcan & K. Q. Weinberger (Eds.), *Proceedings of the 33rd international conference on machine learning* (pp. 478–487). Stroudsburg: International Machine Learning Society.
7. Yang, J., Parikh, D., & Batra, D. (2016). Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 5147–5156). Piscataway: IEEE.
8. Peng, X., Xiao, S., Feng, J., Yau, W.-Y., & Yi, Z. (2016). Deep subspace clustering with sparsity prior. In S. Kambhampati (Ed.), *Proceedings of the 25th international joint conference on artificial intelligence* (pp. 1925–1931). Cham: Springer.
9. Peng, X., Feng, J., Lu, J., Yau, W.-Y., & Yi, Z. (2017). Cascade subspace clustering. In S. Singh & S. Markovitch (Eds.), *Proceedings of the 31st AAAI conference on artificial intelligence* (pp. 2478–2484). Palo Alto: AAAI Press.
10. Guo, X., Gao, L., Liu, X., & Yin, J. (2017). Improved deep embedded clustering with local structure preservation. In C. Sierra (Ed.), *Proceedings of the 26th international joint conference on artificial intelligence* (pp. 1753–1759). Cham: Springer.
11. Chang, J., Wang, L., Meng, G., Xiang, S., & Pan, C. (2017). Deep adaptive image clustering. In *Proceedings of the IEEE international conference on computer vision* (pp. 5880–5888). Piscataway: IEEE.
12. Ji, P., Zhang, T., Li, H., Salzmann, M., & Reid, I. D. (2017). Deep subspace clustering networks. In I. Guyon, U. von Luxburg, & S. Bengio, et al. (Eds.), *Proceedings of the 30th international conference on neural information processing systems* (pp. 24–33). Red Hook: Curran Associates.
13. Yang, B., Fu, X., Sidiropoulos, N. D., & Hong, M. (2017). Towards  $k$ -means-friendly spaces: simultaneous deep learning and clustering. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th international conference on machine learning* (pp. 3861–3870). Stroudsburg: International Machine Learning Society.
14. Wu, J., Long, K., Wang, F., Qian, C., Li, C., Lin, Z., et al. (2019). Deep comprehensive correlation mining for image clustering. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 8149–8158). Piscataway: IEEE.
15. Sprechmann, P., Bronstein, A. M., & Sapiro, G. (2015). Learning efficient sparse and low rank models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9), 1821–1833.
16. Zhou, J. T., Di, K., Du, J., Peng, X., Yang, H., Pan, S. J., et al. (2018). SC2Net: sparse LSTMs for sparse coding. In S. A. McIlraith & K. Q. Weinberger (Eds.), *Proceedings of the 30th AAAI conference on artificial intelligence* (pp. 4588–4595). Palo Alto: AAAI Press.
17. Peng, X., Zhou, J. T., & Zhu, H. (2018). K-meansNet: when  $k$ -means meets differentiable programming. arXiv preprint. [arXiv:1808.07292](https://arxiv.org/abs/1808.07292).
18. Tijmen, T., & Geoffrey, H. (2012). Lecture 6.5-rmsprop: divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2), 26–31.
19. Kingma, D. P., & Adam, J. B. (2015). A method for stochastic optimization. In Y. Bengio & Y. LeCun (Eds.), *Proceedings of the 3rd international conference on learning representations*, San Diego, USA.
20. Xie, X., Zhou, P., Li, H., Lin, Z., & Adan, S. Y. (2022). Adaptive Nesterov momentum algorithm for faster optimizing deep models. arXiv preprint. [arXiv:2208.06677](https://arxiv.org/abs/2208.06677).
21. Zhou, P., Xie, X., & Win, Y. S. (2023). Win: weight-decay-integrated Nesterov acceleration for adaptive gradient algorithms. In *Proceedings of the 11th international conference on learning representations* (pp. 1–28). Retrieved March 1, 2024, from <https://openreview.net/pdf?id=CPdc775QfQ5>.
22. Xie, X., Wu, J., Liu, G., Zhong, Z., & Lin, Z. (2019). Differentiable linearized ADMM. In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th international conference on machine learning* (pp. 6902–6911). Stroudsburg: International Machine Learning Society.
23. Liu, R., Cheng, S., He, Y., Fan, X., Lin, Z., & Luo, Z. (2019). On the convergence of learning-based iterative methods for nonconvex inverse problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(12), 3027–3039.
24. Heaton, H., Chen, X., Wang, Z., & Yin, W. (2023). Safeguarded learned convex optimization. In B. Williams, Y. Chen, & J. Neville (Eds.), *Proceedings of the 37th AAAI conference on artificial intelligence* (pp. 7848–7855). Palo Alto: AAAI Press.
25. Indyk, P., Vakilian, A., & Yuan, Y. (2019). Learning-based low-rank approximations. In H. M. Wallach, H. Larochelle, A. Beygelzimer, et al. (Eds.), *Proceedings of the 33rd international conference on neural information processing systems* (pp. 7400–7410). Red Hook: Curran Associates.
26. Li, C.-G., & Vidal, R. (2015). Structured sparse subspace clustering: a unified optimization framework. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 277–286). Piscataway: IEEE.
27. Feng, J., Lin, Z., Xu, H., & Yan, S. (2014). Robust subspace segmentation with block-diagonal prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3818–3825). Piscataway: IEEE.
28. Lu, C., Feng, J., Lin, Z., Mei, T., & Yan, S. (2019). Subspace clustering by block diagonal representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2), 487–501.
29. Wu, J., Xie, X., Nie, L., Lin, Z., & Zha, H. (2020). Unified graph and low-rank tensor learning for multi-view clustering. In *Proceedings of the 34th AAAI conference on artificial intelligence* (pp. 6388–6395). Palo Alto: AAAI Press.
30. Xie, X., Wu, J., Liu, G., & Wang, J. (2019). Matrix recovery with implicitly low-rank data. *Neurocomputing*, 334, 219–226.
31. Bojanowski, P., & Joulin, A. (2017). Unsupervised learning by predicting noise. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th international conference on machine learning* (pp. 517–526). Stroudsburg: International Machine Learning Society.
32. Shaham, U., Stanton, K. P., Li, H., Basri, R., Nadler, B., & Spectralnet, Y. K. (2018). SpectralNet: spectral clustering using deep neural networks. In *Proceedings of the 6th international conference on learning representations* (pp. 1–20). Retrieved March 1, 2024, from [https://openreview.net/pdf?id=HJ\\_aoCyRZ](https://openreview.net/pdf?id=HJ_aoCyRZ).
33. Caron, M., Bojanowski, P., Joulin, A., & Douze, M. (2018). Deep clustering for unsupervised learning of visual features. In V. Ferrari, M. Hebert, C. Sminchisescu, et al. (Eds.), *Proceedings of the 15th European conference on computer vision* (pp. 139–156). Cham: Springer.
34. Gregor, K., & LeCun, Y. (2010). Learning fast approximations of sparse coding. In J. Fürnkranz & T. Joachims (Eds.), *Proceedings of the 27th international conference on machine learning* (pp. 399–406). Stroudsburg: International Machine Learning Society.
35. Yang, Y., Sun, J., Li, H., & Xu, Z. (2016). Deep ADMM-Net for compressive sensing MRI. In D. D. Lee, M. Sugiyama, & U. von Luxburg, et al. (Eds.), *Proceedings of the 29th international conference on neural information processing systems* (pp. 10–18). Red Hook: Curran Associates.
36. Chen, X., Liu, J., Wang, Z., & Yin, W. (2018). Theoretical linear convergence of unfolded ISTA and its practical weights and thresholds. In S. Bengio, H. M. Wallach, & H. Larochelle, et al. (Eds.), *Proceedings of the 32nd international conference on neural information processing systems* (pp. 9079–9089). Red Hook: Curran Associates.
37. Liu, J., Chen, X., Wang, Z., & Yin, W. (2019). ALISTA: analytic weights are as good as learned weights in LISTA. In *Proceedings of the 7th international conference on learning representations* (pp. 1–33). Retrieved March 1, 2024, from <https://openreview.net/pdf?id=B1lnzn0ctQ>.
38. Lin, Z., Liu, R., & Su, Z. (2011). Linearized alternating direction method with adaptive penalty for low-rank representation. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, et al. (Eds.), *Proceedings of the 25th international conference on neural information processing systems* (pp. 612–620). Red Hook: Curran Associates.
39. Lin, Z., Chen, M., & Ma, Y. (2010). The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices. arXiv preprint. [arXiv:1009.5055](https://arxiv.org/abs/1009.5055).
40. Beck, A., & Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202.
41. Liu, Y., Yuan, X., Zeng, S., & Zhang, J. (2018). Partial error bound conditions and the linear convergence rate of the alternating direction method of multipliers. *SIAM Journal on Numerical Analysis*, 56(4), 2095–2123.
42. Wang, Z., Liu, D., Chang, S., Ling, Q., Yang, Y., & Huang, T. S. (2016). D3: deep dual-domain based fast restoration of JPEG-compressed images. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2764–2772). Piscataway: IEEE.
43. Van der Maaten, L., & Geoffrey, H. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86), 2579–2605.
44. Wei, C., Kakade, S. M., & Ma, T. (2020). The implicit and explicit regularization effects of dropout. In *Proceedings of the 37th international*

conference on machine learning (pp. 10181–10192). Stroudsburg: International Machine Learning Society.

45. Edelman, A., Arias, T. A., & Smith, S. T. (1998). The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2), 303–353.
46. Wen, Z., & Yin, W. (2013). A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142(1–2), 397–434.
47. Ren, X., & Lin, Z. (2013). Linearized alternating direction method with adaptive penalty and warm starts for fast solving transform invariant low-rank textures. *International Journal of Computer Vision*, 104(1), 1–14.
48. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.
49. Krizhevsky, A., & Hinton, G. (2009). *Learning multiple layers of features from tiny images*. Technical report, University of Toronto.
50. Nie, F., Zeng, Z., Tsang, I. W., Xu, D., & Zhang, C. (2011). Spectral embedded clustering: a framework for in-sample and out-of-sample spectral clustering. *IEEE Transactions on Neural Networks*, 22(11), 1796–1808.
51. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P.-A. (2010). Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11, 3371–3408.
52. Xu, W., Liu, X., & Gong, Y. (2003). Document clustering based on non-negative matrix factorization. In C. L. A. Clarke, G. V. Cormack, J. Callan, et al. (Eds.), *Proceedings of the 26th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 267–273). New York: ACM.
53. Cai, D., He, X., & Han, J. (2005). Document clustering using locality preserving indexing. *IEEE Transactions on Knowledge and Data Engineering*, 17(12), 1624–1637.
54. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., et al. (2016). Tensorflow: a system for large-scale machine learning. In K. Keeton & T. Roscoe (Eds.), *Proceedings of the 12th USENIX symposium on operating systems design and implementation* (pp. 265–283). Berkeley: USENIX Association.

## Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Submit your manuscript to a SpringerOpen<sup>®</sup> journal and benefit from:

- Convenient online submission
- Rigorous peer review
- Open access: articles freely available online
- High visibility within the field
- Retaining the copyright to your article

---

Submit your next manuscript at ► [springeropen.com](https://www.springeropen.com)

---