Unsupervised Manifold Linearizing and Clustering

Tianjiao Ding[1] Shengbang Tong[2] Kwan Ho Ryan Chan[1] Xili Dai[3] Yi Ma[2] Benjamin D. Haeffele[1]

Abstract

We consider the problem of simultaneously clustering and learning a linear representation of data lying close to a union of low-dimensional manifolds, a fundamental task in machine learning and computer vision. When the manifolds are assumed to be linear subspaces, this reduces to the classical problem of subspace clustering, which has been studied extensively over the past two decades. Unfortunately, many real-world datasets such as natural images can not be well approximated by linear subspaces. On the other hand, numerous works have attempted to learn an appropriate transformation of the data, such that data is mapped from a union of general non-linear manifolds to a union of linear subspaces (with points from the same manifold being mapped to the same subspace). However, many existing works have limitations such as assuming knowledge of the membership of samples to clusters, requiring high sampling density, or being shown theoretically to learn trivial representations. In this paper, we propose to optimize the Maximal Coding Rate Reduction metric with respect to both the data representation and a novel doubly stochastic cluster membership, inspired by state-of-the-art subspace clustering results. We give a parameterization of such a representation and membership, allowing efficient mini-batching and one-shot initialization. Experiments on CIFAR-10, -20, -100, and TinyImageNet-200 datasets show that the proposed method is much more accurate and scalable than state-of-the-art deep clustering methods, and further learns a latent linear representation of the data.[4]

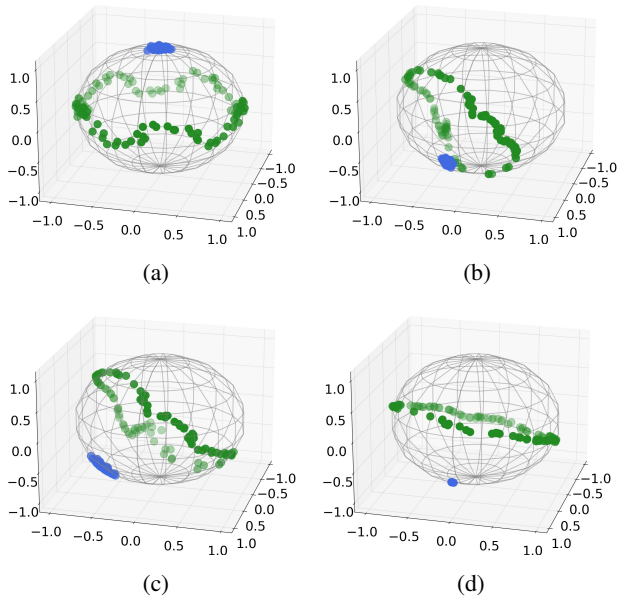


Figure 1: (a) Input data $\boldsymbol{X}$ where 100 points in green lie on a curve and 100 in blue lie close to a point. (b) Stage 0: Features $f_{\boldsymbol{\theta}}(\boldsymbol{X})$ from a neural network $f_{\boldsymbol{\theta}}$ whose parameters $\boldsymbol{\theta}$ are randomly initialized. (c) Stage 1: Features after self-supervised learning. (d) Stage 2: Features further improved by the proposed Manifold Linearizing and Clustering (MLC).

1. Introduction

1.1. Clustering: from Linear to Non-linear Models

Clustering is a fundamental problem in machine learning, allowing one to group data into clusters based on assumptions about the geometry of each cluster. As early as the 1950s, the classic k-means [43, 23, 30, 46] algorithm emerged to cluster data that concentrate around *distinct centroids*, with numerous variants [8, 2, 4] following. This assumption of distinct centroids was later generalized in *subspace clustering* methods, which aim to cluster data lying close to a union of *low-dimensional linear (or affine) subspaces*[5]. This motivated numerous lines of research in the past two decades, leading to various formulations [20, 22, 44, 41, 28, 75, 40, 33] with efficient algorithms [76, 75, 15] and theoretical guarantees on the correctness of the clustering [58, 59, 71, 72, 36, 63, 74, 55, 69]. Subspace clustering has been used in a wide range of applications, such as segmenting image pixels [45, 70, 42], video frames [65, 61, 37], or rigid-body motions [66, §11], along

[1]Mathematical Institute for Data Science, Johns Hopkins University, USA [2]Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, USA [3]The Hong Kong University of Science and Technology (Guangzhou), PRC

[4]Code is available at https://github.com/tianjiaoding/mlc.

[5]Note, a centroid can be seen as a 0-dimensional affine subspace.

with clustering face images [24, 29, 22] or human actions [73, 25, 50].

However, while subspace clustering methods have achieved state-of-the-art performance for certain tasks and datasets, the geometric assumption upon which they rely (namely that the datapoints lie on a union of linear or affine subspaces) is often grossly violated for common high-dimensional datasets. For instance, even in a dataset as simple as MNIST hand-written digits [34], images of a single digit do *not* lie close to a low-dimensional subspace; directly applying subspace clustering methods thus fails. Instead, a more natural idea is to assume that each cluster is defined by a *non-linear low-dimensional manifold* and to learn or design a *non-linear transformation* of the data so that points from one manifold are mapped to one linear subspace. In some cases, one may be able to hand-craft an appropriate transformation of the data, with polynomial or exponential kernel mappings being examples that have been explored in the literature [21], and the authors of [39] show that a subspace clustering method achieves 99% clustering accuracy on MNIST when the data is passed through the scattering-transform [9].

Unfortunately though, hand-crafted design requires one to assume specific and simple families of manifolds which is often unrealistic and challenging to apply on complicated data such as natural images. On the other hand, the authors of [21] propose to cluster data via treating a local neighborhood of the manifold approximately as a linear subspace and applying subspace clustering techniques to local neighborhoods. However, this method requires sufficient sampling density to succeed, which implies a prohibitive number of samples when the manifolds have moderate dimension or curvature. More recently, numerous works propose to learn an appropriate transformation of the data via deep networks and then perform subspace clustering in a latent feature space [53, 31, 1, 80, 32]. Unfortunately, it has been shown that many of these formulations are ill-posed and provably learn trivial representations[6], with much of the claimed benefit coming from ad-hoc post-processing rather than the method itself [27]. This motivates the one of the primary questions we consider here:

Question 1. *Can we efficiently transform data near a union of low-dimensional manifolds, so that the transformed data lie close to a union of low-dimensional linear subspaces to allow for easy clustering?*

1.2. Learning Diverse and Discriminative Features: from Supervised to Unsupervised Learning

Meanwhile, learning a *compact* representation from *multi-modal* data has been a topic of its own interest in machine learning [7]. An ideal property of the learned representation is *between-cluster discrimination*, namely, features from different clusters should be well separated, which is often pursued via a loss such as the classic cross-entropy (CE) objective. However, an important yet often ignored property is that the learned representation maintains *within-cluster diversity*. This allows distances of samples within a cluster to be preserved under the learned transformation, which may facilitate downstream tasks such as generation [16], denoising [68], and semantic interpretation [78, §B.3.1] (see also §A). Unfortunately, the representation learned by CE fails to achieve this property and exhibits *neural collapse*, a phenomenon discovered by [51] with extensive theoretical and empirical analysis [47, 85, 62, 83] (even for non-CE objectives [84]), where latent features from one cluster tend to collapse to a single point. In contrast, [78] recently proposed Maximal Coding Rate Reduction ($\texttt{MCR}^2$) as an objective to pursue both of the mentioned ideal properties. In particular, $\texttt{MCR}^2$ learns a *union-of-orthogonal-subspaces* representation: features from each cluster spread uniformly in a low-dimensional subspace (compact & within-cluster diverse), and the subspaces corresponding to different clusters are orthogonal to each other (between-cluster discriminative). Nevertheless, $\texttt{MCR}^2$ requires ground-truth labels to learn such a representation. This leads to our second question of interest:

Question 2. *For data lying close to a union of manifolds, can we learn a union-of-orthogonal-subspaces representation, without access to the ground-truth labels?*

1.3. Our Contributions

To address the two interrelated questions, we start with the basic idea of blending the philosophies from $\texttt{MCR}^2$ and subspace clustering to explore the best of both worlds. This idea leads us to the following contributions.

- *Formulation (§2)*: We propose Manifold Linearizing and Clustering (MLC) objective (4), which optimizes the $\texttt{MCR}^2$ loss over both the representation and a novel *doubly stochastic* cluster membership . The latter consists of pair-wise similarities between samples, and it is constrained to be doubly stochastic, inspired by state-of-the-art subspace clustering results [39, 18].

- *Algorithm (§3)*: We describe how to parameterize and initialize the representation and membership, as well as to optimize MLC (4). Even though the membership is doubly stochastic, which may appear large in size and hard to constrain, we give an efficient parameterization of it that allows for mini-batching and notably *one-shot initialization*. That is, the membership is initialized with *no additional training* whatsoever leveraging already-initialized representation, which is stable, structured, and efficient.

[6] In this paper, we use 'representation' and 'feature' interchangeably to mean the image of the data under a (learned) transformation.

- *Experiments (§4)*: On CIFAR-10, we demonstrate that MLC learns a union-of-orthogonal-subspaces representation, and achieves more accurate clustering than state-of-the-art subspace clustering methods. Moreover, on CIFAR-10, -20, -100, and TinyImageNet-200, we show that MLC yields *higher clustering accuracy* using *less running time* than state-of-the-art deep clustering methods, even when there are *many or imbalanced clusters*.

1.4. Additional Related Work

Beyond the above, we make connections to a few emerging deep-learning-based works related to this paper.

Self-supervised Representation Learning. An important line of research that learns a representation without using ground-truth labels is that of self-supervised learning. It has seen remarkable recent progress thanks to the so-called *joint-embedding* approach. The basic idea of the latter is to learn a representation such that augmentations of the same input have similar features, while features from different inputs do not collapse to a single point. Extending this idea, self-supervised methods such as SimCLR [12], BYOL [26], and VICReg [5] are able to learn representations on par with those obtained from supervised learning methods; see [57] for an excellent review. Encouraging as it may sound, these methods do not aim to learn a union-of-orthogonal-subspaces representation, nor do they explicitly model clustering in their design. Nevertheless, we shall see that self-supervised methods are key stepping stones for the proposed method, as they will be used to initialize parts of our model.

Clustering and Representation Learning. Numerous works have proposed to jointly perform clustering and representation learning, leveraging the success of self-supervised learning. Roughly speaking, most methods consider the following two steps. The first step is to use self-supervised learning to initialize the representation. Indeed, state-of-the-art methods such as SCAN [64] and SPICE [48] adopt SimCLR [12] and MoCoV2 [14] as their pre-trained features. Starting from the initial representation, the second step is to iteratively refine the representation and clustering, using the idea of pseudo-labeling [10, 64, 52, 48]. Despite the promising clustering performance, the representation learned by these methods is not constrained to be both between-cluster discriminative and within-cluster diverse. In contrast, our approach learns a representation with these two ideal properties (Figure 4) and also achieves state-of-the-art clustering performance (Tables 3, 4 and 6). Finally, the work most closely related to this paper is that of Neural Manifold Clustering and Embedding (NMCE) [38] – we note similarities and differences in terms of formulation, algorithm, and empirical performance at the end of §2.2.

Table 1: Summary of prior works and our contributions.

	Manifold		Self-supervised Initialization	
	Linearizing	Clustering	Representation	Membership
MCR^2 [78]	**yes**	no	n/a	n/a
SCAN [64]	No	**yes**	one shot	one shot
NMCE [38]	**yes**	**yes**	one shot	no
MLC (Ours)	**yes**	**yes**	**one shot**	

2. Formulation

We begin by making clear the problem of interest.

Problem 1 (Unsupervised Manifold Linearizing and Clustering)**.** Suppose $\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_n] \in \mathbb{R}^{D \times n}$ is a dataset of n points lying on an union of k low-dimensional manifolds $\bigcup_{j=1}^{k} \mathcal{M}_j$. Given $\boldsymbol{X}$, we aim to simultaneously

1. *Cluster the samples*: find $\hat{\boldsymbol{y}}$ such that $\boldsymbol{x}_i \in \mathcal{M}_{\hat{y}(i)}$;
2. *Learn a linear representation*: find a transformation $f : \mathbb{R}^D \to \mathbb{R}^d$, such that features $f(\boldsymbol{x}_i)$'s from the same cluster spread uniformly in a low-dimensional linear subspace, and the subspaces arising from different clusters are orthogonal.

In §2.1, we review the principle of Maximal Coding Rate Reduction (MCR^2) which is designed to learn ideal representations in a *supervised* manner, i.e., when the ground-truth membership is given. Then in §2.2, we discuss the challenges of simultaneous clustering and learning a representation (i.e., addressing Problem 1 in its entirety), for which we propose our MLC objective (4). Later in §3, we further give an algorithm to optimize MLC (4).

2.1. Supervised Manifold Linearizing via MCR^2

When the cluster membership is given as supervision, MCR^2 [78] is designed to solve part 2) of Problem 1. To begin with, let $f_{\boldsymbol{\theta}} : \mathbb{R}^D \to \mathbb{S}^{d-1}$ be a transformation parameterized by a neural network; this in turn gives the features $\boldsymbol{Z} := [\boldsymbol{z}_1, \dots, \boldsymbol{z}_n] \in \mathbb{R}^{d \times n}$ of data with $\boldsymbol{z}_i := f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \in \mathbb{S}^{d-1}$. MCR^2 aims to learn an ideal representation by utilizing the coding rate measures. Define the *coding rate*

$$R(\boldsymbol{Z}; \epsilon) := \log\det\left(\boldsymbol{I} + \frac{d}{n\epsilon^2}\boldsymbol{Z}\boldsymbol{Z}^\top\right).$$

Intuitively[7], $R(\boldsymbol{Z}; \epsilon)$ measures some volume of features in $\boldsymbol{Z}$ up to $\epsilon > 0$ precision, so maximizing it would *diversify* features of all the samples. Likewise, one can apply the measure to features of each cluster and take weighted mean over the clusters; namely, define $R_c(\boldsymbol{Z}, \boldsymbol{\Pi}; \epsilon)$ as

$$\sum_{j=1}^{k} \frac{\langle \boldsymbol{\Pi}_j, \boldsymbol{1} \rangle}{n} \log\det\left(\boldsymbol{I} + \frac{d}{\langle \boldsymbol{\Pi}_j, \boldsymbol{1} \rangle \epsilon^2}\boldsymbol{Z}\,\mathrm{Diag}(\boldsymbol{\Pi}_j)\boldsymbol{Z}^\top\right).$$

[7] For a rigorous treatment, see for example [45, §2.1].

Here $\mathbf{\Pi} = [\mathbf{\Pi}_1, \ldots, \mathbf{\Pi}_k] \in \mathbb{R}^{n \times k}$ is a given *point-cluster* membership such that $\Pi_{ij} = 1$ if $\boldsymbol{x}_i \in \mathcal{M}_j$ and $\Pi_{ij} = 0$ otherwise, $\mathbf{1}$ is a vector of all ones so $\langle \mathbf{\Pi}_j, \mathbf{1} \rangle$ is the number of points in cluster j, and for any $\boldsymbol{v} \in \mathbb{R}^n$, $\mathrm{Diag}(\boldsymbol{v})$ denotes a diagonal matrix with the entries of $\boldsymbol{v}$ along the diagonal. Minimizing $R_c(\boldsymbol{Z}, \mathbf{\Pi}; \epsilon)$ thus pushes features in each cluster to stay close. MCR^2 learns an ideal representation by *maximizing the coding rate reduction* (hence the acronym)

$$\max_{\boldsymbol{\theta}} \quad R(\boldsymbol{Z}; \epsilon) - R_c(\boldsymbol{Z}, \mathbf{\Pi}; \epsilon) \quad \text{s.t.} \quad \boldsymbol{Z} = f_{\boldsymbol{\theta}}(\boldsymbol{X}) \tag{1}$$

Notably, it has been shown that given $\mathbf{\Pi}$, the features obtained by maximizing the rate reduction has the property that the features of each cluster spread uniformly within a subspace (within-cluster diverse), and subspaces from different clusters are orthogonal (between-cluster discriminative), under relatively mild assumptions [78, Theorem 2.1].

2.2. Unsupervised Manifold Linearizing and Clustering

While MCR^2 is designed to learn ideal representations (§1) when the membership $\mathbf{\Pi}$ is given, here we are interested in the *unsupervised* setting where one does *not* have access to membership annotations. To address both parts 1) and 2) of Problem 1, a natural idea is to optimize MCR^2 over both the representation $\boldsymbol{Z}$ and membership $\mathbf{\Pi}$ via

$$\max_{\boldsymbol{\theta}, \mathbf{\Pi} \in \Omega_\circ} R(\boldsymbol{Z}; \epsilon) - R_c(\boldsymbol{Z}, \mathbf{\Pi}; \epsilon) \quad \text{s.t.} \quad \boldsymbol{Z} = f_{\boldsymbol{\theta}}(\boldsymbol{X}). \tag{2}$$

Here $\Omega_\circ := \{\mathbf{\Pi} \in \mathbb{R}^{n \times k} : \forall i \in [n],\ \exists \hat{y}(i) \quad \text{s.t. } \Pi_{i\hat{y}(i)} = 1 \text{ and } \Pi_{ij} = 0 \text{ for } j \neq \hat{y}(i)\}$ is the set of all 'hard' assignments, i.e., each row of $\mathbf{\Pi}$ is a one-hot vector. However, this optimization is in general combinatorial: its complexity grows exponentially in n and k, and it does not allow smooth and gradual changes of $\mathbf{\Pi}$. Further, a second challenge is the chicken-and-egg nature of this problem: If one already has an ideal representation $\boldsymbol{Z}$, then existing subspace clustering methods can be applied on $\boldsymbol{Z}$ to estimate the membership. Likewise, if one is given the ground-truth membership $\mathbf{\Pi}$ of clusters, then solving (1) would lead to an ideal representation. However, the $\boldsymbol{Z}$ and $\mathbf{\Pi}$ at the beginning of optimization are typically far from ideal.

In the rest of this section, we propose a so-called *doubly stochastic membership* to deal with the combinatorial challenge. To tackle the chicken-and-egg nature, we parameterize and initialize the representation and membership in an efficient and effective way, as we will discuss in §3.

Doubly Stochastic Subspace Clustering. To address the combinatorial nature of estimating the memberships, we draw inspiration from the closely related problem of *subspace clustering*, where the goal is to cluster n samples assumed to lie close to a union of k low-dimensional subspaces (§1.1). In this case, one typically does not directly learn an $n \times k$ matrix denoting memberships of n points into k subspaces. Instead, one first learns an affinity matrix $\mathbf{\Gamma} \in \mathbb{R}^{n \times n}$ signaling the similarities between pairs of points, and then applies spectral clustering on the learned $\mathbf{\Gamma}$ to obtain a final clustering [20, 22, 44, 41, 28, 75]. In particular, requiring doubly-stochastic constraints on the affinity $\mathbf{\Gamma}$ is shown theoretically to suppress false inter-cluster connections for clustering [18] along with state-of-the-art empirical performance for subspace clustering [39].

Inspired by the above, we propose a constraint set Ω for matrix $\mathbf{\Gamma}$ to be the set of $n \times n$ doubly stochastic matrices,

$$\Omega = \{\mathbf{\Gamma} \in \mathbb{R}^{n \times n} : \mathbf{\Gamma} \geq 0, \quad \mathbf{\Gamma}\mathbf{1} = \mathbf{\Gamma}^\top \mathbf{1} = \mathbf{1}\}. \tag{3}$$

However, this constraint alone is insufficient for strong clustering performance: Consider optimizing MCR^2 with respect to $\mathbf{\Gamma} \in \Omega$ only, and note that the objective is convex with respect to $\mathbf{\Gamma}$. Since we maximize a convex function with respect to convex constraints Ω, an optimal $\mathbf{\Gamma}$ would lie at an extreme point of Ω, which for doubly stochastic matrices is a permutation matrix. This is not ideal for clustering, as it implies that every point is assigned to its own distinct cluster, and there is no incentive to merge points into larger clusters. To resolve this issue, we use a negative entropy regularization $\sum_{ij} \Gamma_{ij} \log(\Gamma_{ij})$ to $\mathbf{\Gamma}$ which biases $\mathbf{\Gamma}$ toward the uniform matrix $\frac{1}{n}\mathbf{1}\mathbf{1}^\top$. By tuning the coefficient of such regularization, we can also tune the sparsity level of $\mathbf{\Gamma}$. This regularization can be conveniently integrated into the network architecture, as we will see in §3. Now we are ready to state our proposed formulation Manifold Linearizing and Clustering (MLC):

$$\begin{aligned} \max_{\boldsymbol{\theta}} \quad & R(\boldsymbol{Z}; \epsilon) - R_c(\boldsymbol{Z}, \mathbf{\Gamma}; \epsilon) \\ \text{s.t.} \quad & \boldsymbol{Z} = f_{\boldsymbol{\theta}}(\boldsymbol{X}),\ \mathbf{\Gamma} = h_{\boldsymbol{\theta}}(\boldsymbol{X}) \in \Omega, \text{ where} \\ R_c(\boldsymbol{Z}, \mathbf{\Gamma}; \epsilon) &= \frac{1}{n} \sum_{j=1}^{n} \log\det\left(\boldsymbol{I} + \frac{d}{\epsilon^2} \boldsymbol{Z}\, \mathrm{Diag}((\mathbf{\Gamma})_j) \boldsymbol{Z}^\top\right). \end{aligned} \tag{4}$$

Note that both $\boldsymbol{Z}$ and $\mathbf{\Gamma}$ are parameterized by neural networks. While a doubly stochastic membership $\mathbf{\Gamma}$ may seem large in size and hard to constrain, we explain in §3 how we parameterize $\mathbf{\Gamma}$ so that the constraints are satisfied by construction and efficient mini-batching is allowed.

Comparison with NMCE. A recent paper (NMCE) [38] studies the same Problem 1 as in this paper, and also proposes to optimize MCR^2 over both the representation and membership. Their method adopts an $n \times k$ matrix $\mathbf{\Pi}$ to model the *point-cluster* membership; in contrast MLC uses a *doubly stochastic point-point* membership $\mathbf{\Gamma}$ inspired from the state-of-the-art subspace clustering (as stated above). Although seemingly not particularly significant, we note that in practice this allows for significantly simpler initialization strategies since we can initialize with a $n \times n$ estimate of an affinity matrix rather than a $n \times k$ estimate of cluster membership. We further elaborate on algorithmic differences at

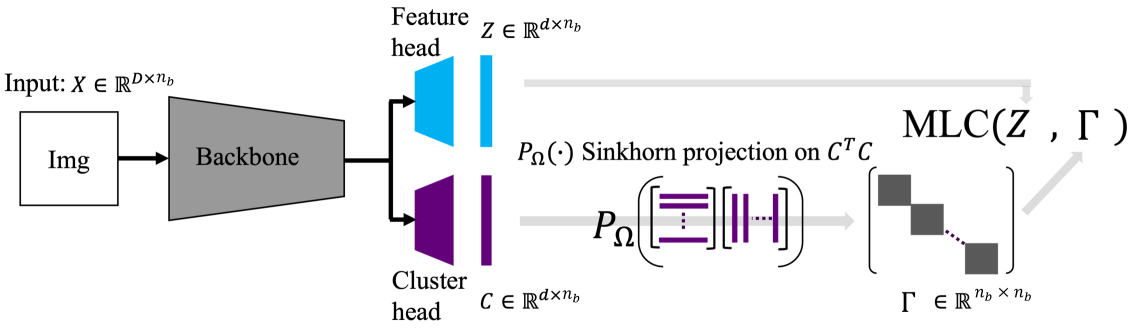


Figure 2: Overall architecture for optimizing the proposed Manifold Linearizing and Clustering (MLC) objective (4). Given a mini-batch of n_b input samples $\boldsymbol{X}$ each lying in $\mathbb{R}^D$, their d-dimensional representation is given by $\boldsymbol{Z}$. Further, their doubly stochastic membership matrix $\boldsymbol{\Gamma}$ is given by taking an inner product kernel of the output of the cluster head $\boldsymbol{C}$ followed by a doubly stochastic projection.

the end of §3.2, and give empirical evidence that MLC is more accurate (Table 3) and stable against randomness (§F).

3. Algorithm

In this section, we describe how to parameterize the representation $\boldsymbol{Z}$ and doubly stochastic membership $\boldsymbol{\Gamma}$ (§3.1), as well as how to initialize them (§3.2) – in an efficient and effective manner. We summarize the meta-algorithm in Algorithm 1, and the overall architecture in Figure 2.

3.1. Efficient Parameterization

Parameterizing $\boldsymbol{Z}$. As is common practice, we take an existing network architecture such as ResNet-18 as the backbone. We append a few affine layers with non-linearities as the *feature head* to further transform the output of the backbone to $\mathbb{R}^d$, followed by a projection layer to respect the unit sphere $\mathbb{S}^{d-1}$ constraint.

Parameterizing $\boldsymbol{\Gamma}$. Different from parameterizing $\boldsymbol{Z}$, this is much less trivial: If one were to directly take $\boldsymbol{\Gamma}$ as decision variables in Ω, it would lead to maintaining $O(n^2)$ variables, which is prohibitive for large datasets (e.g., $n = 10^6$ for ImageNet). To allow efficient computation, we again draw inspiration from *subspace clustering*: There, the membership $\boldsymbol{\Gamma}$ given data $\boldsymbol{X}$ often takes the form of $g(\boldsymbol{X})^\top g(\boldsymbol{X})$ for some transformation g, such as in the inner product kernel [28, 18] where $g(\boldsymbol{X}) = \boldsymbol{X}$ or the least square regression [44] where $g(\boldsymbol{X}) = (\boldsymbol{I} + \lambda \boldsymbol{X}^\top \boldsymbol{X})^{-1/2}\boldsymbol{X}$. It is then tempting to take a neural network $g_{\boldsymbol{\theta}}$ and use $\boldsymbol{C}^\top \boldsymbol{C}$ as the membership where $\boldsymbol{C} = g_{\boldsymbol{\theta}}(\boldsymbol{X})$. Nevertheless, such a matrix is in general *not* doubly stochastic, i.e., $\boldsymbol{C}^\top \boldsymbol{C} \notin \Omega$. To obtain a doubly stochastic membership, we further apply a Sinkhorn projection layer $P_{\Omega,\eta}(\cdot)$ [56, 19], which gives $\boldsymbol{\Gamma} = P_{\Omega,\eta}(\boldsymbol{C}^\top \boldsymbol{C}) \in \Omega$, where η is the coefficient of entropy regularization.[8] As in parameterizing $\boldsymbol{Z}$, we implement $g_{\boldsymbol{\theta}}$ by taking the same backbone and appending layers of the same type to be the *cluster head*. As we shall see soon in §3.2, such a parameterization further allows us to initialize both $\boldsymbol{Z}$ and $\boldsymbol{\Gamma}$ in *one shot* using self-supervised learning.

Complexity. Thanks to the above parameterization, we can do forward and backward passes efficiently via mini-batches. For a mini-batch of n_b samples ($n_b \ll n$ typically), the mini-batched versions of $\boldsymbol{Z}$, $\boldsymbol{C}$ and $\boldsymbol{\Gamma}$ have sizes $d \times n_b$, $d \times n_b$ and $n_b \times n_b$, respectively (Figure 2).[9]

3.2. Efficient Initialization

Since the proposed MLC objective (4) is non-convex, it is important to properly initialize both $\boldsymbol{Z}$ and $\boldsymbol{\Gamma}$ to converge to good (local) minimum.

Initializing $\boldsymbol{Z}$: Self-supervised Representation Learning. Randomly initialized features could be far from being ideal (in the sense defined in §1), and further may not respect the invariance to augmentation, i.e., the augmented samples should have their representation close to each other. Thus, we initialize the features using a self-supervised learning called *total coding rate* (TCR) [38]

$$\begin{aligned} \max_{\boldsymbol{\theta}} \quad & R\Big(\frac{\boldsymbol{Z}+\boldsymbol{Z}'}{2}; \epsilon\Big) + \lambda \sum_{i=1}^{n} |\boldsymbol{z}_i^\top \boldsymbol{z}_i'|, \\ \text{s.t.} \quad & \boldsymbol{z}_i', \boldsymbol{z}_i \in \mathbb{S}^{d-1}, \quad \forall i \in [n], \end{aligned} \tag{5}$$

where for every i, $\boldsymbol{z}_i$ and $\boldsymbol{z}_i'$ are features of different augmentations of the i-th sample. This essentially requires that features from different augmentations of the same sample should be as close as possible, whereas features from different samples should be as uncorrelated as possible. [10]

[8] So in (4), $\boldsymbol{\Gamma} = h_{\boldsymbol{\theta}}(\boldsymbol{X})$ is simply $\boldsymbol{\Gamma} = P_{\Omega,\eta}(g_{\boldsymbol{\theta}}(\boldsymbol{X})^\top g_{\boldsymbol{\theta}}(\boldsymbol{X}))$.

[9] Interested readers are referred to [3] for efficient computation of $\log\det$ via variational forms, and [19] for efficient Sinkhorn iterations via implicit differentiation; these are beyond the scope of this paper.

[10] TCR is introduced here since it also uses coding rate measures. In

Initializing $\mathbf{\Gamma}$. An ideal initialization of $\mathbf{\Gamma}$ would be such that if $\mathbf{\Gamma}_{ij}$ has a high value then points i, j are likely to be from the same true cluster and vice versa. Luckily, after the self-supervised feature initialization mentioned above, $\boldsymbol{Z}$ already have some structures which we can utilize. Thus, we propose to initialize $\mathbf{\Gamma}$ with $P_{\Omega,\eta}(\boldsymbol{Z}^\top \boldsymbol{Z})$; this is easily implemented by copying the parameters from the feature head to the cluster head *once* after the self-supervised initialization of the features, thanks to the parameterization of $\mathbf{\Gamma}$ discussed in §3.1.

With the parameterization and initialization of our doubly stochastic membership $\mathbf{\Gamma}$ set up, we are ready to contrast it with a popular alternative in the sequel.

Doubly Stochastic Membership vs. Point-Cluster Membership. Different from the doubly stochastic *point-point* membership $\mathbf{\Gamma}$ proposed in this paper, prior deep representation learning and clustering works [64, 38, 48] often model a *point-cluster* membership. That is, an $n \times k$ matrix $\mathbf{\Pi}$ where each row represents the probability of a point belonging to k clusters. $\mathbf{\Pi}$ is parameterized by a neural network (or a cluster head if one wishes), initialized randomly or otherwise via an extra training stage after the representation $\boldsymbol{Z}$ is initialized. We highlight a few advantages of using a doubly stochastic point-point membership over a point-cluster one:

• *Stable*: As $\mathbf{\Gamma}$ is initialized deterministically, the performance of MLC is more stable compared to incorporating randomness in initializing the cluster head. We further justify this point empirically in §F.

• *Structured*: Initialization of $\mathbf{\Gamma}$ takes advantage of structures from self-supervised initialized $\boldsymbol{Z}$. As a side benefit, MLC automatically gains from developments of self-supervised representation learning.

• *Efficient*: Once $\boldsymbol{Z}$ is initialized, $\mathbf{\Gamma}$ can be initialized with no additional cost whatsoever, compared to using any extra training stage to initialize the cluster head. In contrast, e.g., SCAN [64] trains the cluster head with 10 different random initializations, which is time-consuming.

Data Augmentation. Beyond initializing $\boldsymbol{Z}$, it is often desirable to incorporate augmentation in optimizing the MLC objective (4). Specifically, from $\{\boldsymbol{X}^{(a)} \in \mathbb{R}^{D\times n}\}_{a=1}^{A}$ the dataset $\boldsymbol{X}$ under A different augmentations, one computes $(\boldsymbol{Z}^{(a)} \in \mathbb{R}^{d\times n}, \mathbf{\Gamma}^{(a)} \in \mathbb{R}^{d\times n})$ for each augmentation a, and use in (4)

$$\boldsymbol{Z} = P_{\mathbb{S}^{d-1}}\left(\frac{1}{A}\sum_{a=1}^{A}\boldsymbol{Z}^{(a)}\right), \quad \mathbf{\Gamma} = \frac{1}{A}\sum_{a=1}^{A}\mathbf{\Gamma}^{(a)} \in \Omega. \tag{6}$$

Note that one can benefit from parallelization by putting $\boldsymbol{X}^{(a)}, \boldsymbol{Z}^{(a)}, \mathbf{\Gamma}^{(a)}$ for each augmentation a on one computing device, since $\mathbf{\Gamma}^{(a)}$ only depends on $\boldsymbol{X}^{(a)}$ but not from other augmentations.

principle, many contrastive learning methods could be used for initializing the features, e.g., as we demonstrate via experiments (Table 3).

Algorithm 1 MLC: Manifold Linearizing and Clustering

Input: $\boldsymbol{X} \in \mathbb{R}^{D\times n}$, $\epsilon, \eta > 0$, $d, k, n_b, T, A \in \mathbb{Z}_{\geq 0}$

initialize $\boldsymbol{Z}$ by self-supervised representation learning ▷ (5)
initialize $\mathbf{\Gamma}$ via parameter copying
for $t = 1, \ldots, T$ **do**
  $\bar{\boldsymbol{X}} \in \mathbb{R}^{D\times n_b} \leftarrow$ sample a batch from $\boldsymbol{X}$
  $\bar{\boldsymbol{X}}^{(1)}, \ldots, \bar{\boldsymbol{X}}^{(A)} \leftarrow$ apply A augmentations to $\bar{\boldsymbol{X}}$
  $\bar{\boldsymbol{Z}}, \bar{\mathbf{\Gamma}} \leftarrow$ forward pass with $\{\bar{\boldsymbol{X}}^{(a)}\}_{a=1}^{A}$ and network parameters $\boldsymbol{\theta}$ ▷ (6)
  $\nabla_{\boldsymbol{\theta}}(4) \leftarrow$ backward pass
  $\boldsymbol{\theta} \leftarrow$update $\boldsymbol{\theta}$ using some optimizer on $\nabla_{\boldsymbol{\theta}}(4)$
end for
run spectral clustering on $\mathbf{\Gamma}$ to estimate labels $\hat{\boldsymbol{y}}$

Output: $\boldsymbol{Z}, \hat{\boldsymbol{y}}$

4. Experiments on Real-World Datasets

We empirically verify that MLC learns a union-of-orthogonal-subspaces representation, and yields more accurate clustering than state-of-the-art subspace clustering methods (§4.1). Further, we show that MLC outperforms state-of-the-art deep clustering methods, even when there are many or imbalanced clusters (§4.2).

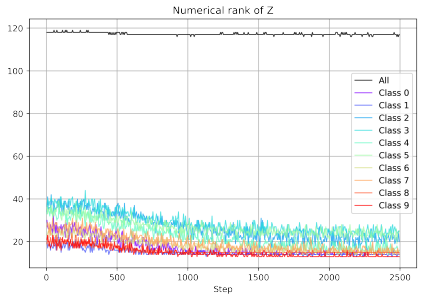
Metrics. To evaluate the clustering quality, we run spectral clustering on learned membership matrix $\mathbf{\Gamma}$, and report the normalized mutual information (NMI, [60]) and clustering accuracy (ACC, [35]), as are commonly used in clustering tasks. To evaluate the learned representation, we define the following metric: for a collection of points $\boldsymbol{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_l] \in \mathbb{R}^{d\times l}$ $(l > d)$ with associated singular values $\{\sigma_i\}_{i=1}^{d}$, define the numerical rank of $\boldsymbol{W}$ as $\arg\min_r \left\{r : \sum_{i=1}^{r}\sigma_i^2 / \sum_{i=1}^{d}\sigma_i^2 > 0.95\right\}$. Now, one can measure the numerical rank of the learned representation $\boldsymbol{Z}$, as well as that of each ground-truth cluster[11] of $\boldsymbol{Z}$. A low numerical rank of $\boldsymbol{W}$ implies that points in $\boldsymbol{W}$ lie close to a low-dimensional subspace. We further report the cosine similarity of learned representation, which is simply $|\boldsymbol{z}_i^\top \boldsymbol{z}_j|$ for points i and j, since $\|\boldsymbol{z}_i\| = 1$ by construction in (4). Finally, to compare the efficiency of methods we report the training time in §4.2, where the experiments are run on 2 Nvidia RTX3090 GPUs.

4.1. Comparison with Subspace Clustering

To demonstrate the ability of MLC to cluster the samples and linearize the manifolds, we conduct experiments on

[11] They are defined by the true membership, so that the numerical rank metric is decoupled from the quality of learned membership $\mathbf{\Gamma}$.

(a) Coding rate of all features R, that of clustered features R_c, and the rate reduction $\Delta R = R - R_c$.



(b) Numerical ranks of all features $\boldsymbol{Z}$ and features from each ground-truth cluster i, $\{\boldsymbol{z}_j : y(j) = i\}$.

Figure 3: Coding rates (as loss terms in MLC (4)) and numerical ranks (§4.1) of the features learned by MLC on CIFAR-10 as epoch varies.

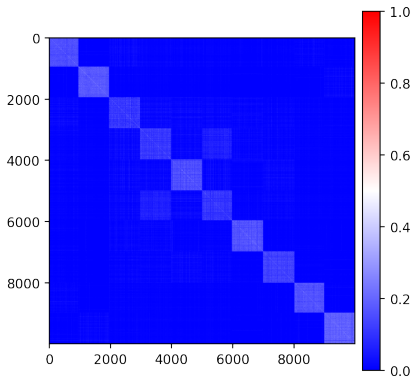


Figure 4: Cosine similarity $|\boldsymbol{Z}_{\text{MLC}}^\top \boldsymbol{Z}_{\text{MLC}}|$ of the features $\boldsymbol{Z}_{\text{MLC}}$ learned by MLC.

CIFAR-10, which consists of RGB images from 10 classes such as planes, birds, and deers. As mentioned in §1 subspace clustering methods rely crucially on the assumption that data lie close to a union of linear subspaces, which many real-world dataset may not satisfy. To show that this is the case, we additionally compare the proposed method with subspace clustering methods. As we shall see, applying subspace clustering directly on self-supervised features of CIFAR-10 will yield low clustering accuracy. In contrast, MLC is able to *achieve high clustering accuracy*, and moreover, *produce a union-of-orthogonal-subspaces representation* on which subspace clustering methods can also achieve high accuracy.

Data. We use the training split of CIFAR-10 containing 50000 RGB images, each of size $3 \times 32 \times 32$. We use the augmentation specified in the Appendix to perform self-supervised representation learning via TCR (5) and get $\boldsymbol{Z}_{\text{TCR}}$. For a fair comparison, the so-learned $\boldsymbol{Z}_{\text{TCR}}$ are used both as initialization for MLC (line 1 of Algorithm 1), and as the input for subspace clustering methods[12]. In MLC, for each image in each batch we randomly sample $A = 2$ augmentations to apply to the image. As an additional comparison, we also run subspace clustering methods on the features $\boldsymbol{Z}_{\text{MLC}}$ learned by MLC.

Methods. We compare with the elastic-net subspace clustering with active-set solver (EnSC, [75]) and sparse subspace clustering with orthogonal matching pursuit solver (SSC-OMP, [76]), using off-the-shelf implementation provided by the authors[13]. We search the parameters of EnSC over $(\gamma, \tau) \in \{1, 5, 10, 50, 100\} \times \{0.9, 0.95, 1\}$ and those of SSC over $(k_{\max}, \epsilon) \in \{3, 5, 10, 20\} \times \{10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}\}$, and report the run with the highest clustering accuracy for each method. We summarize detailed parameters for MLC in the Appendix.

[12] Self-supervised features learned via coding rate measures empirically exhibit some union-of-subspace structure (one subspace per cluster). They have been used for subspace clustering as in, e.g., [78, §3.2] and [81, §4.2].

[13] https://github.com/ChongYou/subspace-clustering.

Table 2: Clustering accuracy and normalized mutual information of subspace clustering (EnSC, SSC-OMP) and proposed manifold linearizing and clustering (MLC). $\boldsymbol{X}$ contains $6 \cdot 10^4$ images from 10 classes of CIFAR-10, $\boldsymbol{Z}_{\text{TCR}}$ are features learned via self-supervised learning TCR, $\boldsymbol{Z}_{\text{MLC}}$ are features learned by MLC.

Method	Input Data	ACC	NMI
EnSC	$\boldsymbol{Z}_{\text{TCR}}$	72.2	67.9
	$\boldsymbol{Z}_{\text{MLC}}$	81.5	**79.2**
SSC-OMP	$\boldsymbol{Z}_{\text{TCR}}$	67.8	64.5
	$\boldsymbol{Z}_{\text{MLC}}$	78.4	76.3
MLC	$\boldsymbol{X}$	**86.3**	78.3

Results. Figure 3 reports the coding rates (as loss terms in (4) and numerical ranks of features learned by MLC as epoch varies. As a first note, the coding rate R of all features (the blue curve in 3a) decreases only slightly as epoch goes, indicating that the overall representation is diverse in the feature space. Indeed, the numerical rank of all features (the dark curve in Figure 3b) stays 118 which is close to the dimension 128 of the feature space. This is in sharp contrast to the deep subspace clustering methods where all the features collapse to a one-dimensional subspace [27]. Moreover, as the coding rate R_c of clustered features (the orange curve in Figure 3a) goes down, the numerical ranks of features from each ground-truth cluster decrease. For instance, the representation from true cluster 3 has a numerical rank of 37 in the first step and 24 in the last step. This implies that most representation gets linearized better and clustered more accurately, even though the MLC objective (4) is unsupervised, i.e., it does not use ground-truth labels.

Last but not least, note that the features within each ground-truth cluster spread well in a low-dimensional subspace, e.g., the numerical ranks for the true clusters at the last step are within $[13, 23]$. This achieves the desired within-cluster diverse property (§1), as opposed to the neural collapse phenomenon that appears with the cross-entropy loss.

Now we compare MLC with subspace clustering methods. Table 2 reports clustering accuracy and normalized mutual information for applying EnSC, SSC-OMP on self-supervised features $\boldsymbol{Z}_{\text{TCR}}$, features $\boldsymbol{Z}_{\text{MLC}}$ learned by MLC, and applying MLC on $\boldsymbol{X}$, where $\boldsymbol{X}$ is $6 \cdot 10^4$ images from 10 classes of CIFAR-10. In addition, we plot the cosine similarity of the features learned by MLC in Figure 4. Remarkably, the highest clustering accuracy is 86.3% achieved by MLC on $\boldsymbol{X}$, which surpasses EnSC (72.2%) and SSC-OMP (67.8%) on $\boldsymbol{Z}_{\text{TCR}}$ by a large margin, even though $\boldsymbol{Z}_{\text{TCR}}$ is used both as initialization for MLC and input for EnSC and SSC-OMP. Interestingly, using instead the features $\boldsymbol{Z}_{\text{MLC}}$ learned by MLC, the clustering performance of EnSC and SSC-OMP increases and even becomes comparable to MLC, e.g., EnSC achieves 79.2% normalized mutual information compared to 78.3% of MLC. This suggests that $\boldsymbol{Z}_{\text{MLC}}$ has a union-of-subspace structure that can be utilized by subspace clustering. Indeed, as seen in Figure 4, features from different clusters tend to have a small similarity, i.e., being orthogonal to each other. This demonstrates the between-cluster discrimination (§1) as desired.

4.2. Comparison with Deep Clustering

We further compare the proposed MLC with state-of-the-art deep clustering methods on large-scale datasets (CIFAR-10, -20, -100, and TinyImageNet-200). Different than MLC, most methods reported (all except NMCE as discussed in §2.2) do *not* aim to learn a union-of-orthogonal-subspaces representation. Be that as it may, MLC achieves comparable or better clustering accuracy than state-of-the-art methods using less running time, even when the dataset presents many or imbalanced clusters.

Datasets. Beyond CIFAR-10 (§4.1), we further use CIFAR-20, CIFAR-100 and TinyImageNet-200 to evaluate the performance of our method. Both CIFAR-100 and CIFAR-20 contain the same 50000 train images and 10000 test images with size $32 \times 32 \times 3$, while the former are split into 100 clusters and the latter 20 super clusters. Finally, TinyImageNet contains 100000 train images and 10000 test images with size $64 \times 64 \times 3$ split into 200 clusters.

Baseline Methods. We include clustering accuracy and normalized mutual information reported by SCAN [64], GCC [82], NNM [17], IMC [49], NMCE [38], SPICE [48] on aforementioned datasets whenever applicable. In addition, to *compare the running time* as well as to have more baseline methods when there are *many clusters*, we conduct experiments with MLC, SCAN [64], and IMC [49] on CIFAR-100. Training details are left to §B.2. For a fair comparison, all methods reported use ResNet-18 as the backbone, commonly adopted by the literature. Note that each method chooses one or more pre-training that best fits its objective, such as SimCLR [12], SwAV [11], TCR [38], MoCoV2 [14]. Hence, for clarity, we indicate the pre-training used after the method, e.g., SCAN-SimCLR means SCAN method initialized with SimCLR.

Table 3: Clustering accuracy and normalized mutual information of different methods on CIFAR-10 and CIFAR-20. For a fair comparison, all methods use ResNet-18 as backbone.

Method vs. Dataset & Metric	CIFAR-10 ACC	CIFAR-10 NMI	CIFAR-20 ACC	CIFAR-20 NMI
SCAN-SimCLR (ECCV '20)	.876	.787	.468	.459
GCC-SimCLR (ICCV '21)	.856	.764	.472	.472
NNM-SimCLR (CVPR '21)	.843	.748	.477	.484
IMC-SwAV (KBS '22)	**.891**	**.811**	**.490**	**.503**
NMCE-TCR (Arxiv '22)	.830	.761	.437	.488
MLC-TCR (Ours)	**.863**	**.783**	**.522**	**.546**
SCAN-MoCoV2 (ECCV '20)	.874	.786	.455	.472
SPICE-MoCoV2 (TIP '22)	.918	.850	.535	.565
MLC-MoCoV2 (Ours)	**.922**	**.855**	**.583**	**.596**

Table 4: Clustering accuracy and normalized mutual information of different methods on CIFAR-100 and TinyImageNet-200. For a fair comparison, all methods use ResNet-18 as backbone.

Method vs. Dataset & Metric	CIFAR-100 ACC	CIFAR-100 NMI	TinyImageNet-200 ACC	TinyImageNet-200 NMI
SCAN-SimCLR (ECCV '20)	34.3	55.7	-	-
GCC-SimCLR (ICCV '21)	-	-	13.8	34.7
IMC-SwAV (KBS '22)	43.9	58.3	28.2	52.6
SPICE-MoCoV2 (TIP '22)	-	-	30.5	44.9
MLC-TCR (Ours)	**49.4**	**68.3**	**33.5**	**67.5**

Table 5: Running time in minutes and clustering accuracy on CIFAR-100. For a fair comparison, all methods use ResNet-18 as backbone.

Method vs. Metric & Stage	Running Time I	Running Time II	Running Time III	Running Time Total	ACC
SCAN-SimCLR (ECCV '20)	308.3	33.3	54.7	396.3	34.3
IMC-SwAV (KBS '22)	529.4	-	-	529.4	43.9
MLC-TCR (Ours)	266.7	17.7	-	**284.4**	**49.4**

Results on CIFAR-10, -20. Table 3 presents clustering accuracy and normalized mutual information of various methods. Overall, MLC is *the most accurate*, among

Table 6: Clustering accuracy on imbalanced datasets: (a) Imb-CIFAR-10, (b) Imb-CIFAR-100. For a fair comparison, all methods use ResNet-18 as backbone.

Method / Dataset	(a)	(b)
SCAN (ECCV '20)	62.9	31.1
IMC-SwAV (KBS '22)	65.7	38.2
MLC (Ours)	**80.0**	**46.1**

methods using either the same pre-training (middle and bottom rows) or any pre-training. To begin with, using TCR as pre-training, MLC achieves 86.3% and 52.2% clustering accuracies on CIFAR-10 and -20, which are 3.3 and 8.5 percentage points higher respectively than those achieved by NMCE[14]. Remarkably, when MLC is initialized with MoCoV2 pre-training, it yields even higher clustering accuracies of 92.2% on CIFAR-10 and 58.3% on CIFAR-20, *surpassing previous state-of-the-art methods* SPICE-MoCoV2 (91.8%, 53.5%) and IMC-SwAV (89.1%, 49.0%). Interestingly, while the clustering performance of MLC-TCR is competitive on CIFAR-20, it is less so on CIFAR-10. After investigation, we find that the learned clusters appear *semantically meaningful*, even though they do not agree with the ground-truth labels of CIFAR-10 used for evaluation; we leave the details to §A.

Results on CIFAR-100, TinyImageNet-200. We report clustering accuracy and normalized mutual information on both datasets in Table 4, and further show running time on CIFAR-100 in Table 5. Notably, MLC outperforms SCAN and IMC-SwAV on CIFAR-100 and TinyImageNet-200 by a large margin, while using *lower running time*: E.g., on CIFAR-100, MLC yields an accuracy of 49.4% in 291 minutes, whereas IMC-SwAV has 43.9% using 529 minutes, and SCAN has 34.3% in 396 minutes.

Results on Imbalanced Clusters. Note that for CIFAR-10 or CIFAR-100 each cluster contains approximately the same number of samples. On the other hand, natural images are typically imbalanced, i.e., the clusters have unequal number of samples. To mimic this setting, we take a naive approach to construct the following imbalanced datasets. For the 10 clusters of CIFAR-10, we remove half of the samples from odd-numbered clusters (i.e., clusters $1, 3, \ldots, 9$) from both the training and test split. We refer to the reduced dataset Imb-CIFAR-10. Likewise we construct Imb-CIFAR-100. We run two state-of-the-art methods IMC-SwAV and SCAN as well as the proposed MLC on Imb-CIFAR-10 and Imb-CIFAR-100.

Table 6 shows clustering accuracy on the imbalanced datasets Imb-CIFAR-10 and Imb-CIFAR-100. As a first observation, the clustering accuracy of all methods is lower on the imbalanced datasets than on the balanced counterparts, as expected. Notably, MLC suffers from the least performance drop, e.g., when moving from CIFAR-10 to Imb-CIFAR-10 the accuracy of MLC drops from 86% to 80%, whereas that of SCAN and IMC-SwAV decreases from above 87% to below 66%.

[14] For a further comparison between MLC and NMCE, see the end of §2.2 for conceptual and algorithmic differences, and §F for an empirical study on the stability against random seeds.

5. Conclusion

This paper studies the problem of simultaneously clustering and learning an union-of-orthogonal-subspaces representation for data, when data lies close to a union of low-dimensional manifolds. To address the problem we propose an objective based on *maximal coding rate reduction* and *doubly stochastic* membership inspired by the state-of-the-art subspace clustering results. We provide an efficient and effective parameterization of the membership variables as well as a meta-algorithm to optimize the representation and membership jointly. We further conduct experiments on datasets with larger number of clusters and imbalanced clusters and show that the proposed method achieves state-of-the-art performance. We believe that our work provides a general and unified framework for unsupervised learning of structured representations for multi-modal data.

Acknowledgements This work was partially supported by ONR grant N00014-22-1-2102, the joint Simons Foundation-NSF DMS grant 2031899, a research grant from TBSI, NSF grant 1704458, the Northrop Grumman Mission Systems Research in Applications for Learning Machines (REALM) initiative, and NSF graduate fellowship DGE2139757.

References

[1] Mahdi Abavisani and Vishal M Patel. Deep multimodal subspace clustering networks. *IEEE Journal of Selected Topics in Signal Processing*, 12(6):1601–1614, Apr. 2018. 2

[2] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, June 2006. 1

[3] Christina Baek, Ziyang Wu, Kwan Ho Ryan Chan, Tianjiao Ding, Yi Ma, and Benjamin D Haeffele. Efficient maximal coding rate reduction by variational forms. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 500–508, 2022. 5, 16

[4] Bahman Bahmani, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. Scalable K-Means++. *Proceedings VLDB Endowment*, 5(7), Mar. 2012. 1

[5] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021. 3

[6] Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-Invariance-Covariance regularization for Self-Supervised learning. In *International Conference on Learning Representations*, 2022. 14

[7] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: a review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, Aug. 2013. 2

[8] Paul Bradley, Olvi Mangasarian, and W Street. Clustering via concave minimization. In *Advances in neural information processing systems*, 1996. 1

[9] Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886, Aug. 2013. 2

[10] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *European conference on computer vision*, pages 132–149, July 2018. 3

[11] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 2020. 8

[12] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 3, 8

[13] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in neural information processing systems*, 33:22243–22255, 2020. 14

[14] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. Mar. 2020. 3, 8, 14

[15] Yanxi Chen, Gen Li, and Yuantao Gu. Active orthogonal matching pursuit for sparse subspace clustering. *IEEE Signal Process. Lett.*, 25(2):164–168, Feb. 2018. 1

[16] Xili Dai, Shengbang Tong, Mingyang Li, Ziyang Wu, Michael Psenka, Kwan Ho Ryan Chan, Pengyuan Zhai, Yaodong Yu, Xiaojun Yuan, Heung Yeung Shum, and Yi Ma. Ctrl: Closed-loop transcription to an ldr via minimaxing rate reduction. *Entropy*, 2022. 2

[17] Zhiyuan Dang, Cheng Deng, Xu Yang, Kun Wei, and Heng Huang. Nearest neighbor matching for deep clustering. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2021. 8

[18] Tianjiao Ding, Derek Lim, Rene Vidal, and Benjamin D Haeffele. Understanding doubly stochastic clustering. In *the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5153–5165. PMLR, 2022. 2, 4, 5

[19] Marvin Eisenberger, Aysim Toker, Laura Leal-Taixé, Florian Bernard, and Daniel Cremers. A unified framework for implicit sinkhorn differentiation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 509–518, 2022. 5, 14, 16

[20] Ehsan Elhamifar and Rene Vidal. Sparse subspace clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2790–2797, June 2009. 1, 4

[21] Ehsan Elhamifar and René Vidal. Sparse manifold clustering and embedding. *Advances in neural information processing systems*, 24, 2011. 2

[22] Ehsan Elhamifar and Rene Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2765–2781, 2013. 1, 2, 4

[23] Edward Forgey. Cluster analysis of multivariate data: Efficiency vs. interpretability of classification. *Biometrics*, 1965. 1

[24] A S Georghiades, P N Belhumeur, and D J Kriegman. From few to many: illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(6):643–660, June 2001. 2

[25] Behnam Gholami and Vladimir Pavlovic. Probabilistic temporal subspace clustering. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, July 2017. 2

[26] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020. 3

[27] Benjamin D Haeffele, Chong You, and René Vidal. A critique of Self-Expressive deep subspace clustering. In *International Conference on Learning Representations*, 2020. 2, 7

[28] Reinhard Heckel and Helmut Bölcskei. Robust subspace clustering via thresholding. *IEEE transactions on information theory*, 61(11):6320–6342, 2015. 1, 4, 5

[29] J Ho, Ming-Husang Yang, Jongwoo Lim, Kuang-Chih Lee, and D Kriegman. Clustering appearances of objects under varying illumination conditions. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 1, pages I–I, June 2003. 2

[30] R C Jancey. Multidimensional group analysis. *Australian Journal of Botany*, 14:127–130, 1966. 1

[31] Pan Ji, Tong Zhang, Hongdong Li, Mathieu Salzmann, and Ian Reid. Deep subspace clustering networks. In *Advances in Neural Information Processing Systems*, volume 2017-Decem, pages 24–33, 2017. 2

[32] Mohsen Kheirandishfard, Fariba Zohrizadeh, and Farhad Kamangar. Deep low-rank subspace clustering. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, June 2020. 2

[33] Connor Lane, Ron Boger, Chong You, Manolis Tsakiris, Benjamin Haeffele, and Rene Vidal. Classifying and comparing approaches to subspace clustering with missing data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019. 1

[34] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998. 2

[35] Minsik Lee, Jieun Lee, Hyeogjin Lee, and Nojun Kwak. Membership representation for detecting block-diagonal

structure in low-rank or sparse subspace clustering. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June:1648–1656, 2015. 6

[36] Chun-Guang Li, Chong You, and René Vidal. On geometric analysis of affine sparse subspace clustering. *IEEE J. Sel. Top. Signal Process.*, 12(6):1520–1533, Dec. 2018. 1

[37] Sheng Li, Kang Li, and Yun Fu. Temporal subspace clustering for human motion segmentation. In *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, Dec. 2015. 1

[38] Zengyi Li, Yubei Chen, Yann LeCun, and Friedrich T Sommer. Neural manifold clustering and embedding. *arXiv [cs.LG]*, Jan. 2022. 3, 4, 5, 6, 8, 14, 15, 16, 17

[39] Derek Lim, René Vidal, and Benjamin D Haeffele. Doubly stochastic subspace clustering. *arXiv [cs.LG]*, Nov. 2020. 2, 4

[40] John Lipor, David Hong, Yan Shuo Tan, and Laura Balzano. Subspace clustering using ensembles of K-Subspaces. Sept. 2017. 1

[41] Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, Yong Yu, and Yi Ma. Robust recovery of subspace structures by low-rank representation. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):171–184, Jan. 2013. 1, 4

[42] Li Liu, Liang Kuang, and Yunfeng Ji. Multimodal MRI brain tumor image segmentation using sparse subspace clustering algorithm. *Comput. Math. Methods Med.*, 2020:8620403, July 2020. 1

[43] Stuart Lloyd. Least squares quantization in PCM. Technical report, Bell Laboratories, 1957. 1

[44] Can-Yi Lu, Hai Min, Zhong-Qiu Zhao, Lin Zhu, De-Shuang Huang, and Shuicheng Yan. Robust and efficient subspace segmentation via least squares regression. In *European conference on computer vision*, pages 347–360. Springer, 2012. 1, 4, 5

[45] Yi Ma, Harm Derksen, Wei Hong, and John Wright. Segmentation of multivariate mixed data via lossy data coding and compression. *IEEE transactions on pattern analysis and machine intelligence*, 29(9):1546–1562, 2007. 1, 3

[46] James B McQueen. Some methods for classification and analysis of multivariate observations. In *Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967. 1

[47] Dustin G Mixon, Hans Parshall, and Jianzong Pi. Neural collapse with unconstrained features. Nov. 2020. 2

[48] Chuang Niu, Hongming Shan, and Ge Wang. SPICE: Semantic Pseudo-Labeling for image clustering. *IEEE Trans. Image Process.*, 31:7264–7278, Nov. 2022. 3, 6, 8

[49] Foivos Ntelemis, Yaochu Jin, and Spencer A Thomas. Information maximization clustering via Multi-View Self-Labelling. *arXiv [cs.CV]*, Mar. 2021. 8, 15

[50] Giancarlo Paoletti, Jacopo Cavazza, Cigdem Beyan, and Alessio Del Bue. Subspace clustering for action recognition with covariance representations and temporal pruning. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 6035–6042, Jan. 2021. 2

[51] Vardan Papyan, X Y Han, and David L Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences of the United States of America*, 117(40):24652–24663, Oct. 2020. 2

[52] Sungwon Park, Sungwon Han, Sundong Kim, Danu Kim, Sungkyu Park, Seunghoon Hong, and Meeyoung Cha. Improving unsupervised image clustering with robust learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12278–12287, 2021. 3

[53] Xi Peng, Jiashi Feng, Shijie Xiao, Jiwen Lu, Zhang Yi, and Shuicheng Yan. Deep sparse subspace clustering. *arXiv [cs.CV]*, Sept. 2017. 2

[54] Herbert Robbins and Sutton Monro. A stochastic approximation method. *Ann. Math. Stat.*, 22(3):400–407, 1951. 15

[55] Daniel P Robinson, Rene Vidal, and Chong You. Basis pursuit and orthogonal matching pursuit for subspace-preserving recovery: Theoretical analysis. *arXiv [cs.LG]*, Dec. 2019. 1

[56] Michael E Sander, Pierre Ablin, Mathieu Blondel, and Gabriel Peyré. Sinkformers: Transformers with doubly stochastic attention. In *International Conference on Artificial Intelligence and Statistics*, Oct. 2021. 5

[57] Ravid Shwartz-Ziv and Yann LeCun. To compress or not to compress- Self-Supervised learning and information theory: A review. Apr. 2023. 3

[58] Mahdi Soltanolkotabi and Emmanuel J Candés. A geometric analysis of subspace clustering with outliers. *Annals of statistics*, 40(4):2195–2238, 2012. 1

[59] Mahdi Soltanolkotabi, Ehsan Elhamifar, and Emmanuel J Candes. Robust subspace clustering. *Annals of statistics*, 42(2):669–699, 2014. 1

[60] Strehl and Ghosh. Cluster ensembles – a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 2002. 6

[61] Stephen Tierney, Junbin Gao, and Yi Guo. Subspace clustering for sequential data. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, June 2014. 1

[62] Tom Tirer and Joan Bruna. Extended unconstrained features model for exploring deep neural collapse. In *International Conference on Machine Learning*, Feb. 2022. 2

[63] Manolis Tsakiris and Rene Vidal. Theoretical analysis of sparse subspace clustering with missing entries. In Jennifer Dy and Andreas Krause, editors, *International Conference on Machine Learning*, pages 4975–4984, 2018. 1

[64] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. SCAN: Learning to classify images without labels. In *European conference on computer vision*. Springer, 2020. 3, 6, 8, 15

[65] René Vidal, Yi Ma, and Shankar Sastry. Generalized principal component analysis (GPCA). *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(12):1945–1959, Dec. 2005. 1

[66] René Vidal, Yi Ma, and S Shankar Sastry. *Generalized principal component analysis*, volume 40. 2016. 1, 14

[67] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007. 16

[68] Bo Wang and Zhuowen Tu. Sparse subspace denoising for image manifolds. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 468–475, June 2013. 2

[69] Peng Wang, Huikang Liu, Anthony Man-Cho So, and Laura Balzano. Convergence and recovery guarantees of the K-Subspaces method for subspace clustering. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 22884–22918. PMLR, 2022. 1

[70] Xiaofang Wang, Yuxing Tang, Simon Masnou, and Liming Chen. A Global/Local affinity graph for image segmentation. *IEEE Trans. Image Process.*, 24(4):1399–1411, Apr. 2015. 1

[71] Yining Wang, Yu-Xiang Wang, and Aarti Singh. A deterministic analysis of noisy sparse subspace clustering for dimensionality-reduced data. In *International Conference on Machine Learning*, 2015. 1

[72] Yu-Xiang Wang and Huan Xu. Noisy sparse subspace clustering. *Journal of Machine Learning Research*, 2016. 1

[73] Fei Wu, Yongli Hu, Junbin Gao, Yanfeng Sun, and Baocai Yin. Ordered subspace clustering with Block-Diagonal priors. *IEEE Trans Cybern*, 46(12):3209–3219, Dec. 2016. 2

[74] Chong You, Chun-Guang Li, Daniel Robinson, and Rene Vidal. Is an affine constraint needed for affine subspace clustering? In *International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2019. 1

[75] Chong You, Chun Guang Li, Daniel P Robinson, and Rene Vidal. Oracle based active set algorithm for scalable elastic net subspace clustering. In *the IEEE conference on computer vision and pattern recognition*, pages 3928–3937, 2016. 1, 4, 7

[76] Chong You, Daniel P Robinson, and René Vidal. Scalable sparse subspace clustering by orthogonal matching pursuit. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2016. 1, 7

[77] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv [cs.CV]*, Aug. 2017. 14

[78] Yaodong Yu, Kwan Ho Ryan Chan, Chong You, Chaobing Song, and Yi Ma. Learning diverse and discriminative representations via the principle of maximal coding rate reduction. In *Neural Information Processing Systems*, June 2020. 2, 3, 4, 7, 15

[79] Yaodong Yu, Kwan Ho Ryan Chan, Chong You, Chaobing Song, and Yi Ma. Learning diverse and discriminative representations via the principle of maximal coding rate reduction. *Advances in Neural Information Processing Systems*, 33:9422–9434, 2020. 14, 16

[80] Junjian Zhang, Chun-Guang Li, Chong You, Xianbiao Qi, Honggang Zhang, Jun Guo, and Zhouchen Lin. Self-Supervised convolutional subspace clustering network. In *IEEE/CVF conference on computer vision and pattern recognition*, 2019. 2

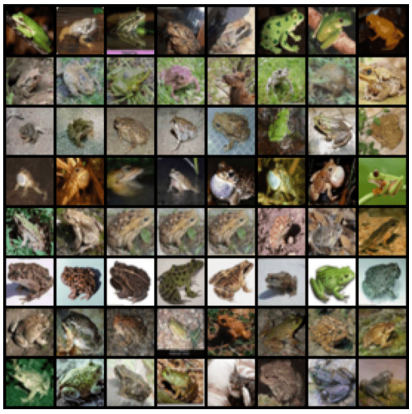
[81] Shangzhi Zhang, Chong You, René Vidal, and Chun-Guang Li. Learning a self-expressive network for subspace clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, Oct. 2021. 7

[82] Huasong Zhong, Jianlong Wu, Chong Chen, Jianqiang Huang, Minghua Deng, Liqiang Nie, Zhouchen Lin, and Xian-Sheng Hua. Graph contrastive clustering. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2021. 8

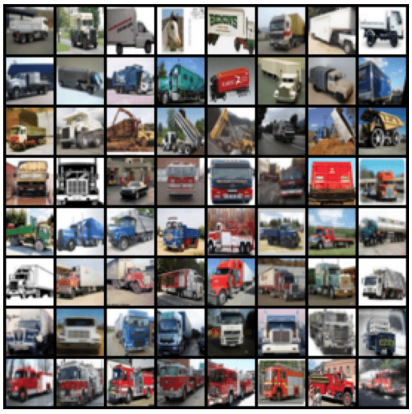
[83] Jinxin Zhou, Xiao Li, Tianyu Ding, Chong You, Qing Qu, and Zhihui Zhu. On the optimization landscape of neural collapse under MSE loss: Global optimality with unconstrained features. In *International Conference in Machine Learning*, Mar. 2022. 2

[84] Jinxin Zhou, Chong You, Xiao Li, Kangning Liu, Sheng Liu, Qing Qu, and Zhihui Zhu. Are all losses created equal: A neural collapse perspective. In Alice H Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. 2

[85] Zhihui Zhu, Tianyu Ding, Jinxin Zhou, Xiao Li, Chong You, Jeremias Sulam, and Qing Qu. A geometric analysis of neural collapse with unconstrained features. *Adv. Neural Inf. Process. Syst.*, 34:29820–29834, 2021. 2



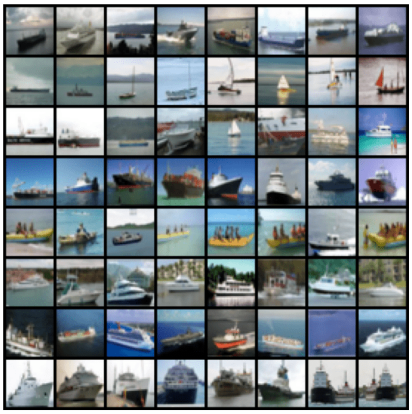
(a) Learned cluster 1



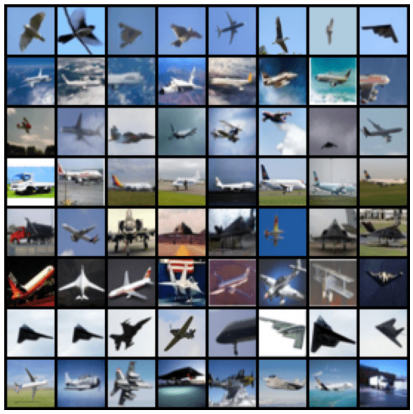
(b) Learned cluster 2

(c) Learned cluster 3

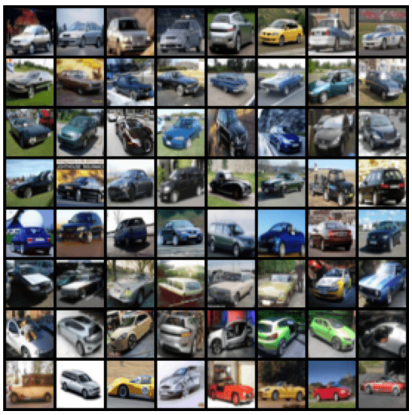
(d) Learned cluster 4



(e) Learned cluster 5



(f) Learned cluster 6



(g) Learned cluster 7

(h) Learned cluster 8

(i) Learned cluster 9

Figure 5: Images along the principal components (see §A) of features from each cluster on CIFAR-10, where features and clusters are learned by `MLC`. For the sake of space, only 9 clusters are shown.

A. Semantic Interpretability and Failure Case Analysis on CIFAR-10

Recall that MLC clusters images and learns a representation for them where images of each cluster lie close to a low-dimensional subspace. A natural question that arises is what are the *different directions within each subspace*? Below we give some interesting visualization. Specifically, after MLC is trained, we take the (learned) features from each (learned) cluster, and apply Principal Component Analysis to them to obtain the first 8 *principal components* (for a review see, e.g., [66, §2.1]). Recall that principal components are mutually orthogonal, indicating uncorrelated directions in the subspace (of each cluster). To visualize these principal components, we show the images from each cluster whose features are the closest to the principal components.

Figure 5 reports the images along the principal components of the 10 clusters using the representation and clusters learned by MLC, where each sub-figure corresponds to a cluster and each row a principal component. Interestingly, the rows of images appear to exhibit some semantic 'concepts': in Figure 5b, row 1 and 8 are respectively white and red trucks, while row 3 are the trucks that ship sand or mud; row 1 of Figure 5d are deers with trees as background. This seems to suggest that the learned representation preserves distance within each cluster, i.e., images that are close/far in semantic meaning will be close/far in the feature space, as desired in §1.

This visualization further allows us to investigate why on CIFAR-10 MLC has a clustering accuracy slightly lower than some state-of-the-art methods (Table 3). We observe that the main clusters samples are in clusters 3 and 8: e.g., rows 1 and 3 of Figure 5h are cats while all other rows in this cluster are dogs. On the other hand, one may argue that Figure 5h is a cluster of pets of lighter colors, and Figure 5c a cluster of pets of darker colors. They could be semantically meaningful clusters despite not aligning with the ground-truth labels. We believe it would be an interesting future work to use MLC to discover new semantics that are not present in the given labels.

B. Details on Experiment Settings

B.1. Synthetic Union-of-Manifold Data

We perform simulations to visualize the properties of the proposed manifold learning and clustering method. As seen in Figure 1a, we generate data $\boldsymbol{X}$ from two manifolds on the sphere $\mathbb{S}^2$, each consisting of 200 samples. The points from the first manifold (green) take the form

$$\boldsymbol{x}_i = \begin{bmatrix} \cos\left(A\sin(\omega\phi_i)\right)\cos\phi_i \\ \cos\left(A\sin(\omega\phi_i)\right)\sin\phi_i \\ \sin\left(A\sin(\omega\phi_i)\right) \end{bmatrix} + \boldsymbol{\epsilon}_i, \tag{7}$$

where $A = 0.2$ and $\omega = 5$ sets the curvature of the manifold, $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\boldsymbol{0}, 0.05\boldsymbol{I}_3)$ is the additive noise, and we take $\phi_i = \frac{2\pi i}{100}$ for $i = 1, \ldots, 100$ to generate 100 points. On the other hand, the points from the second manifold (blue) are simply 100 samples from $\mathcal{N}([0, 0, 1]^\top, 0.05\boldsymbol{I}_3)$. We take the feature dimension $d = 3$ to be equal to he input dimension $D = 3$. We paramterize both the feature head $f_{\boldsymbol{\theta}}$ and the cluster head $g_{\boldsymbol{\theta}}$ to be a simple fully-connected network with 100 hidden neurons, followed by a Rectified Linear Unit as non-linearity and a projection operator onto the sphere $\mathbb{S}^2$. Figures 1b to 1d report the features $\boldsymbol{Z}$ with random initialization (i.e., before line 1 of Algorithm 1), with self-supervised initialization, and at convergence of MLC. Notably, despite $\boldsymbol{Z}$ being noisy and only approximately piece-wise linear, as epoch goes $\boldsymbol{Z}$ gradually transform to two linear subspaces: the green points converge to a 2-dimensional subspace (intersected with $\mathbb{S}^2$) and the blue points converge to a 1-dimension subspace.

B.2. Real-World Datasets

MLC. Since MLC is based on MCR^2 (§2.1), we follow [79, 38] for the choice of batch size and augmentation. For all real-data experiments, we use a batch size of $n_b = 1024$ and VICReg [6] augmentation (Augmentation 1). $A = 2$ augmentations are used, while not using any leads to worse clustering accuracy (§C). Doubling n_b or A improves accuracy by $\leq 1\%$ on CIFAR10-10 with the cost of increased running time. In self-supervised initialization of $\boldsymbol{Z}$ (line 1 of Algorithm 1), we used TCR (see (5) or [38]) or otherwise MoCoV2 [14]. To train TCR, we use the precision (§2.1) parameter $\epsilon^2 = 0.2$, a LARS optimizer [77] (as is also done in [13, 38]) with a learning rate of 0.3 and trained MLC for 1000 epochs. For MoCoV2, we use off-the-shelf pre-trained models[15]. On the other hand, in the training of MLC objective (4), we use $\epsilon^2 = 0.1$ and $\eta = 0.175$ for the entropy regularization in the Sinkhorn projection [19] layer $P_{\Omega,\eta}(\cdot)$. We fix the backbone and for each batch, we perform

[15] https://github.com/vturrisi/solo-learn/tree/d27c7130d19035c0ba0af8f90217e78d8ebe7f48.

one update for parameters in the feature head $\boldsymbol{Z}$ and one update for parameters in the cluster head $\boldsymbol{C}$. For each head we use one SGD optimizer [54] with a learning rate of 10^{-2}, momentum of 0.9, and weight decay of $5 \cdot 10^{-4}$.

Augmentation 1 Augmentations for real datasets

```
import torchvision.transforms as t
t.Compose([
  t.RandomResizedCrop(32,scale=(0.04,
1.0)),
  t.RandomHorizontalFlip(p=0.5),
  t.RandomGrayscale(p=0.2),
  t.RandomApply([t.ColorJitter(0.4,
0.4, 0.4, 0.1)], p=0.8),
  GaussianBlur(p=0.1)
])
```

Table 7: Ablation study on the roles of different parts of Algorithm 1 and on using augmentation.

Ablation Study on CIFAR-10	Clustering Accuracy
Full Algorithm 1	**86.3%**
Replacing self-supervised initialization (line 1) with random initialization	20.0%
Replacing updating MLC loss (4) (lines 3-9) with subspace clustering (EnSC)	73.4%
Not using augmentation in updating (4) (lines 3-9)	80.0%

SCAN and IMC-SwAV. Recall that we conduct experiments on CIFAR-100, Imb-CIFAR-10, and Imb-CIFAR-100 with SCAN [64], IMC-SwAV [49] and MLC, and report clustering and running time in Tables 5 and 6. We use off-the-shelf implementation[16] provided by the authors. For a fair comparison, SCAN, IMC-SwAV and MLC all use ResNet-18 as the backbone. Finally, the hyper-parameters of SCAN and IMC-SwAV are set to be the ones optimally chosen for CIFAR-10 and CIFAR-100 respectively provided by the authors.

C. Role and Ablation Study of Augmentation

Recall that data augmentation was used both in the self-supervised initialization (line 1 of Algorithm 1, see 'Initializing $\boldsymbol{Z}$' in §3) and in updating the MLC objective (lines 3-9 of Algorithm 1, 'Data Augmentation' in §3). Below we give additional clarification on the role of augmentation therein.

C.1. Augmentation for Initializing the Features

Since the proposed MLC objective (4) is highly non-convex, the quality of the (local) solution an optimizer converges to in general depends on the initialization. However, before line 1 of Algorithm 1 is executed, the features $\boldsymbol{Z}$ at initialization could be very far from union-of-orthogonal-subspaces (as pursued by Problem 1), since the neural network has an arbitrary architecture and initialization. To at least promote *some* ideal structures in the features, line 1 of Algorithm 1 is conducted so that the features from an original sample and its augmented copies are close, while features from different samples spread out in the feature space. This is a common idea[17] used in contrastive learning as we review in §1.4. Empirically, initializing the features using augmentation (line 1 of Algorithm 1) is important for the clustering performance: as seen in Table 7, on CIFAR-10, if one uses random initialization to replace this step, then the final clustering accuracy is 20%, in sharp contrast to 86.3%.

C.2. Augmentation for Updating MLC Objective (4)

In optimizing MLC loss (4) (lines 3-9 of Algorithm 1), augmentation empirically improves clustering performance. As one can see in Table 7, on CIFAR-10 using the sample self-supervised initialization of the features, MLC achieves only 80% clustering accuracy without augmentation, in contrast to 86.3% with augmentation. We attribute this difference to the fact that augmentation enriches the diversity of samples the algorithm sees.

[16] https://github.com/wvangansbeke/Unsupervised-Classification, https://github.com/foiv0s/IMC-SwAV-pub

[17] More related are [78, §3.2] and [38, §3.6] which are also based on MCR^2 as in this paper. However, the former does not learn a clustering membership, leading to inferior performance, and we discuss the difference with the latter in §1.4, 2.2 and F.

D. Role and Ablation Study of Components of Algorithm 1

D.1. Initialization of the Features (Line 1)

Please kindly refer to §C.1.

D.2. Updating the MLC Objective (4) (Lines 3-9)

The main novelty of this paper lies in updating the MLC objective (4) that learns both the representation $\boldsymbol{Z}$ and a doubly stochastic membership $\boldsymbol{\Gamma}$. Note that in this step, clustering is pursued by modeling the membership $\boldsymbol{\Gamma}$, as opposed to the self-supervised feature initialization step where no membership is explicitly pursued. This step is indeed important for clustering: as seen in Table 7, on CIFAR-10, the clustering accuracy on the self-supervised initialized features $\boldsymbol{Z}$ is only 73.4%, in contrast to 86.3% obtained after updating the MLC objective (4).

D.3. Spectral Clustering (Line 10)

Since the proposed MLC learns a doubly stochastic membership that signals pair-wise similarity between points, it is standard to run spectral clustering [67] to compute a final set of clusters from the learned membership. This is done only once at the very end of Algorithm 1, and is rather efficient compared to the other parts of Algorithm 1: for instance, using an unaccelerated implementation from SciPy, it takes less than 30 seconds to perform spectral clustering on a $10^4 \times 10^4$ matrix.

E. Complexity Analysis and Efficient Computation

A key observation is that the proposed MLC uses mini-batches (line 4 of Alg. 1). We summarize some important complexities in Table 8.

Table 8: Some key space / time complexities in computing $\log\det$ and Sinkhorn iterations, where d is the dimension of the feature space and n_b is the batch size. Real-data experiments in this paper use $n_b = 1024$ and $d = 128$.

	$\log\det$ terms in (4)	Complexity		Sinkhorn iteration	Complexity
1	Size of matrix within $\log\det$	$d \times d$	4	Size of $\boldsymbol{C}^\top \boldsymbol{C}$	$n_b \times n_b$
2	Evaluate the matrix	$O(n_b d^2)$			
3	Evaluate the $\log\det$	$O(d^3)$			

First, computations 1-3 have the same complexity as in previous works [79, 38]. The only extra overhead comes from the sum in the R_c term in (4) being of size n_b rather than k, where k is # of clusters. However, empirically the extra overhead is negligible using buit-in pararallization from `torch.einsum`. Second, scalability of Sinkhorn iterations on a $n_b \times n_b$ matrix is also not a concern. Since[18], in all real-data experiments, while $n \sim 10^5$ is typically very large, $n_b = 1024$ is much smaller than n. Thus, the time and space complexities remain manageable despite the datasets being large-scale. Indeed, as seen in Table 4, MLC achieves higher accuracy in less time than state-of-the-art alternatives on CIFAR100-100. Finally, we note that efficient computation of $\log\det$ is a subject of research (e.g., [3]), and back propagation of Sinkhorn iterations can be made efficient using implicit differentiation [19]; these are beyond the scope of this paper.

F. Additional Comparison of MLC and NMCE

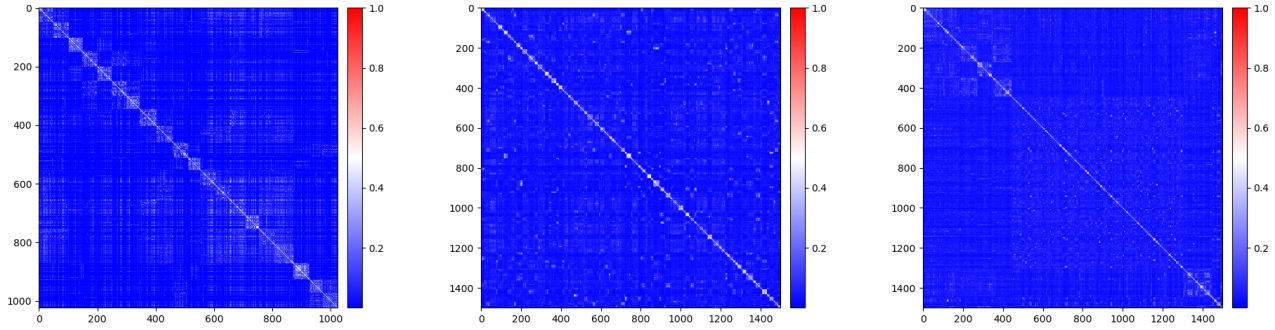
As detailed in §2.2, one of the advantages of the proposed MLC (4) over NMCE [38] is that MLC has a more stable performance with respect to random seeds, since MLC is able to initialize the membership deterministically using structures from the self-supervised initialized features. Below we conduct extra experiments to provide empirical evidence. We first fix a self-supervised initialization of features that is in turn used for both NMCE and MLC. Then, based on this very same initialization of features, we update NMCE and MLC objective respectively with 10 different seeds: recall that NMCE initializes the membership randomly whereas MLC initializes the membership deterministically using the initialized features. To make a valid comparison, for both methods we further use the same optimization strategy and hyper-parameters that are optimally[19] tuned for NMCE (which are not optimal for MLC): precision $\epsilon^2 = 0.2$, # epochs 100, LARS optimizer for $\boldsymbol{Z}$ with an initial learning rate 0.3 decayed to 0 in a cosine annealing manner. Table 9 reports clustering accuracy and normalized mutual

[18] As is also pointed out in §3.

[19] For NMCE, we use the implementation as well as the parameters provided in https://github.com/zengyi-li/NMCE-release.

Table 9: Clustering accuracy and normalized mutual information of MLC and NMCE [38] on CIFAR-10 over 10 random seeds, using the same self-initialized features. Note that MLC is more accurate and stable than NMCE, which is attributed to the fact that the doubly stochastic membership of MLC can be deterministically initialized using self-supervised features (§3).

Method	Metric	Seed										Mean	Std.
		0	1	2	3	4	5	6	7	8	9		
MLC (4)	ACC	84.5	84.8	84.8	84.6	84.4	84.4	84.0	84.3	84.4	84.6	**84.5**	**0.24**
	NMI	76.6	77.1	76.8	76.8	76.5	76.4	76.1	76.4	76.4	76.5	**76.6**	**0.28**
NMCE	ACC	83.7	82.1	81.6	73.7	80.4	77.9	81.7	81.4	72.7	80.9	79.6	3.69
	NMI	74.4	71.2	70.4	65.2	70.0	68.1	72.7	70.8	69.2	69.8	70.2	2.49



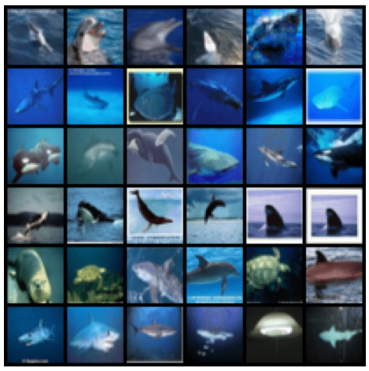
(a) CIFAR-20 (b) CIFAR-100 (c) TinyImageNet-200

Figure 6: Cosine similarity $|\boldsymbol{Z}^\top \boldsymbol{Z}|$ of the features $\boldsymbol{Z}$ learned by MLC on more complicated datasets: CIFAR-20, CIFAR-100, TinyImageNet-200.

information of MLC and NMCE over 10 random seeds. As expected, MLC has a more stable clustering performance by having a standard deviation of clustering accuracy and normalized mutual information less than 0.28, in contrast to more than 2.49 achieved by NMCE. Further, MLC achieves higher mean clustering performance than NMCE, as also observed in Table 3. Last but not least, we note that the numbers in Table 9 are not comparable to those in Table 3, since for MLC the hyper-parameters and optimizers are different, and for NMCE an additional step that fine tunes the backbone is used in Table 3.

G. Addtional Visualization on Learned Representation and Clusters

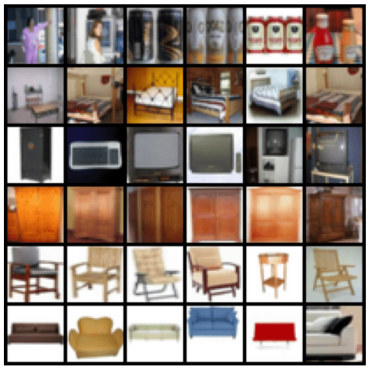
Figure 6 presents the cosine similarity (as defined in the preamble of §4) of the representation learned by MLC on CIFAR-20, CIFAR-100 and TinyImageNet-200 (for the counterpart on CIFAR-10 see §4.1). As seen, the cosine similarity maps form approximately block diagonal structures, showing that the features from different clusters are roughly orthogonal to each other. This is desired by the between-cluster discrimination (§1). Lastly, we provide additional visualization of principal images on CIFAR-20 (see §A for definition) in Figure 8.



(a) Learned cluster 1

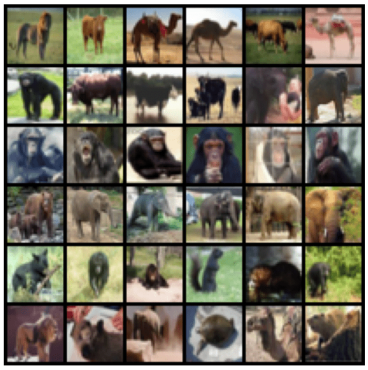
(b) Learned cluster 2

(c) Learned cluster 3



(d) Learned cluster 4

(e) Learned cluster 5



(f) Learned cluster 6

(g) Learned cluster 7

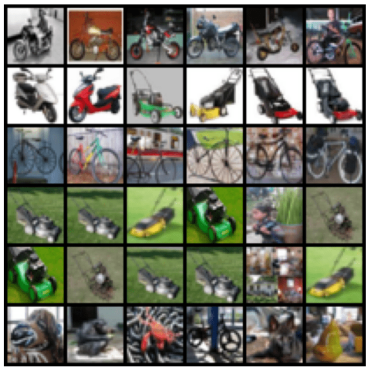
(h) Learned cluster 8

(i) Learned cluster 9

(j) Learned cluster 10

(k) Learned cluster 11

(l) Learned cluster 12



(m) Learned cluster 13

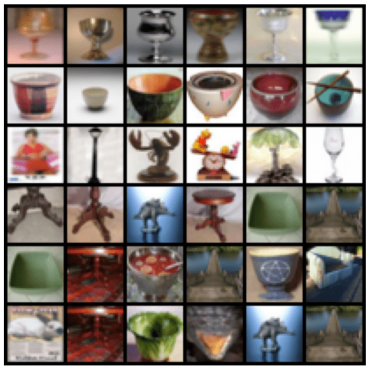
(n) Learned cluster 14

(o) Learned cluster 15

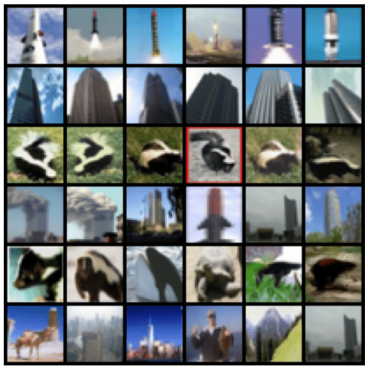
(p) Learned cluster 16

(q) Learned cluster 17

(r) Learned cluster 18



(s) Learned cluster 19



(t) Learned cluster 20

Figure 8: Images along the principal components (defined in §A) of features from each cluster on CIFAR-20, where features and clusters are learned by (4).