


Exploring the Relationship Between the Geometry of a Fixed Embedding of Image Data and Its Underlying Cluster Structure

Yan-Bin Chen, Khong-Loon Tiong & Chen-Hsiang Yeang


To cite this article: Yan-Bin Chen, Khong-Loon Tiong & Chen-Hsiang Yeang (05 Feb 2025): Exploring the Relationship Between the Geometry of a Fixed Embedding of Image Data and Its Underlying Cluster Structure, Journal of Computational and Graphical Statistics, DOI: [10.1080/10618600.2024.2444321](https://doi.org/10.1080/10618600.2024.2444321)

To link to this article: <https://doi.org/10.1080/10618600.2024.2444321>

 View supplementary material 

 Published online: 05 Feb 2025.

 Submit your article to this journal 

 Article views: 137

 View related articles 

 View Crossmark data 



Exploring the Relationship Between the Geometry of a Fixed Embedding of Image Data and Its Underlying Cluster Structure

Yan-Bin Chen^{a,b}, Khong-Loon Tiong^a, and Chen-Hsiang Yeang^a

^aInstitute of Statistical Science, Academia Sinica, Taipei City, Taiwan; ^bMaster Program in Statistics, National Taiwan University, Taipei City, Taiwan

ABSTRACT

Standard self-supervised clustering algorithms transform input data via embedding models that are trained to fit the data and then cluster the embedded vectors. Despite its flexibility, a data driven embedding model may not be applicable when the raw data are unavailable due to privacy or security concerns, and it cannot be adapted to the increasingly common framework of transfer learning. We previously proposed a Merge & Expand (ME) framework for clustering images using a fixed embedding model. In this study, we have substantially modified our ME framework and conducted a series of experimental analysis to explore the relationship between the geometry of a fixed embedding space and the underlying cluster structure. We demonstrate that the clustering outcomes are robust against varying hyperparameter values. We assessed the heterogeneity of predicted labels in each region, revealing that it is a strong indicator of the quality of clustering outcomes. We further exploited the heterogeneity information to modify the ME framework, improving clustering accuracy by introducing a second embedding. Moreover, we provide intuitive explanations for sources of confusion in merged seed regions. Comparisons with numerous other clustering methods on five datasets indicate that our ME framework performs competitively despite employing a fixed embedding, a simple CNN architecture, and a common loss function. Thus, our ME framework enables users to better understand the relationship between the geometry of the embedding space and the underlying cluster structure. Supplementary materials for this article are available online.

ARTICLE HISTORY

Received August 2023
Accepted December 2024

KEYWORDS

Clustering; Convolutional neural network; Fixed embedding; Self-supervised learning

1. Introduction

Clustering unlabeled high-dimensional data is an age-old problem but remains a challenge as the vast majority of existing data lack labels. Recently, the self-supervised learning approaches have been applied in many image clustering algorithms that adopt deep neural network (DNN) models such as convolutional neural networks (CNN) (LeCun et al. 1998) and transformers (Garg et al. 2022). Such approaches acquire supervisory information by integrating the original and augmented data, often requiring that the data share the same semantic content. Generally, a self-supervised clustering algorithm comprises two major components: embedding and clustering/classification. Raw data are first embedded or transformed into feature vectors typically through a DNN. The requirement for clustering together original and augmented images is often encoded in the embedding operations. The embedded data is then passed through a clustering or classification algorithm. Pseudo labels are assigned to the embedded images when a classification algorithm is employed. Self-supervised image clustering algorithms fall into two general categories in terms of the links between the two components: (a) sequential methods (or representation learning) separately optimize the embedding and clustering components (Bengio, Courville, and Vincent 2013; Xie, Girshick, and Farhadi 2016; Som et al. 2018; van den Oord, Li, and Vinyals 2018); and

(b) joint methods (or end-to-end learning) unite embedding learning and clustering in training the same DNNs (Xie, Girshick, and Farhadi 2016; Yang, Parikh, and Batra 2016; Hu et al. 2017; Chang et al. 2017; Glasmachers 2017; Caron et al. 2019; Ji, Henriques, and Vedaldi 2019; Vedaldi, Asano, and Rupprecht 2020). A combination of both sequential and joint methods has also been proposed (Van Gansbeke et al. 2020; Park et al. 2021).

In most current self-supervised image clustering algorithms, the embedding models are trained to fit the input data. Hence, we term such embedding models “data driven”. An alternative approach is to generate “fixed embeddings”, where the model parameters are pre-specified rather than learned from the input data. Instances of fixed embeddings include dimension reduction algorithms such as PCA (Hotelling 1933), t-SNE (Maaten and Hinton 2008) and UMAP (McInnes, Healy, and Melville 2018), as well as pre-trained DNN models such as VGG (Tammina 2019), ResNet (He et al. 2016), BERT (Kenton and Toutanova 2019), Vision Transformer (Cordonnier, Loukas, and Jaggi 2020; Dosovitskiy et al. 2020), GPT-1 (Vaswani et al. 2017; Radford et al. 2018), GPT-2 (Radford et al. 2019), and GPT-3 (Brown et al. 2020). Both embedding approaches have advantages and disadvantages. Data driven embeddings are flexible in meeting the specific requirements and hypotheses of self-supervised learning and better fit the input data, yet they are

also severely restricted by the input data. Fixed embeddings are less tuned to fit the input data, but they are also less dependent on it and they can incorporate powerful pre-trained models. For instance, there are three scenarios where we assert that fixed embeddings are either beneficial or essential. First, the raw data may not be directly accessible in some applications due to privacy or security concerns or the data size is too large for convenient transfer. Consequently, providers may compress or project the raw data and then distribute the embedded data of much smaller dimensions or have sensitive information removed. Second, in transfer learning, a large DNN model pre-trained on extensive existing data can be applied to the input data to generate fixed embeddings (Xian, Schiele, and Akata 2017; Han et al. 2021). Third, parallelization is difficult to achieve for data driven embedding given that the embedding is trained on the entire input data. Nevertheless, the data are readily parallelizable with a fixed embedding as the embedding parameters can be independent of the input data.

Previously, we proposed a “Merge & Expand” (ME) algorithm that uses a fixed embedding and a CNN classifier to cluster unlabeled image data (Chen, Tiong, and Yeang 2022). In that short conference proceeding paper, the respective concepts, algorithm and experimental analysis were all preliminary. Herein, we have substantially developed the concepts, modified and improved the algorithm, and conducted a more thorough experimental analysis. The major advancement of the present study is our scrutiny of the relationship between the geometry of a fixed embedding space and the underlying cluster structure of the data, allowing us to improve clustering by exploiting this link. The current paper differs from that of Chen, Tiong, and Yeang (2022) in five specific aspects. First, we have simplified the procedures for generating and merging seed regions by considerably reducing the number of free parameters. Second, we demonstrate that the ME clustering outcomes are robust against varying hyperparameter values. Some hyperparameters yield invariant clustering outcomes against a wide range of values. We fixed the values of these hyperparameters and further reduced the number of free parameters. Third, we have assessed three more image datasets in our experimental analysis and validate the conclusions from all five datasets. Fourth, we have leveraged the heterogeneity characteristics of predicted class labels to devise a novel method for improving cluster accuracy by integrating two embedding models. Fifth, we provide intuitive explanations for the source of confusion arising from two datasets by inspecting representative images of the erroneously merged seed regions.

2. Methods

The bulk of our ME algorithm is dedicated to identifying a training set that represents the entire data and annotations of its labels for CNN predictions. We partitioned the data into small and tight clusters termed “regions”, and solicited some regions as the training data and assigned them pseudo labels. Information about the underlying class structure was revealed by both the geometry of the entire embedded space and patterns in subsets of features. We extracted these two types of information via a sequential process. In brief, a number of representative “seed regions” was selected according to the geometry in the

embedded space. The pseudo labels of these seed regions were determined by comparing the CNN prediction outcomes with various combinations of seed regions as training data. The initial training data of the seed regions and their pseudo labels were then iteratively expanded according to CNN predictions. Overall, our ME framework comprises five phases as illustrated in Figure 1. In Phase 0 (preparation), the raw data are transformed by a fixed embedding into feature vectors. In Phase 1, the embedded data are clustered into many regions. In Phase 2, some of these regions are selected as the seeds for the training data in subsequent phases. In Phase 3, four types of pairwise scores between seed regions are evaluated according to the CNN prediction outcomes from numerous combinations of seed regions as the training data. In Phase 4, the seed regions are merged and assigned to distinct pseudo labels by integrating the pairwise scores. Finally, in Phase 5, the merged seed regions are treated as the initial training data, and the training data is expanded by iteratively applying the CNN classifier to the entire dataset. We give a more intuitive explication of the algorithms in each phase in this section and place the pseudo-code description of these algorithms in supplementary text S1.

We selected three fixed embeddings for the current study. t-SNE projects each raw image into a three-dimensional vector, whereas ResNet-18 (He et al. 2016) and VGG-16 (Tammina 2019) are large CNN models pre-trained on 14 million images of 1000 categories from the ImageNet database (Deng et al. 2009). These models transform a raw image into a 1000-component feature vector, indicating the activation strength of these categories.

2.1. Phase 1: Clustering and Filtering the Embedded Data into Regions

We applied spectral clustering (using the R package “kkn” with parameter $nn=30$, default otherwise) to group the embedded data into 200 regions. Ideally, data points of the same regions should possess the same hidden class labels. However, this property is not guaranteed. Without knowing the true class labels, two procedures (geometry-based and informative marker-based, see below) were used to prune the “periphery members” of each region likely possessing true class labels differing from those of the majority of “region members” and to discard regions likely possessing data points of heterogeneous true class labels. If the dimensions of the embedded space do not have obvious interpretations, such as from PCA, t-SNE, and UMAP, then we filtered the data points and regions according to their geometry in the embedded space. Otherwise, such as from ResNet-18 and VGG-16, we filtered the data points and regions based on the output values of selected informative markers.

Geometry-based filtering procedure: This procedure examines the subcluster structure of a region in the embedded space and prunes the periphery members that are relatively differentiated from the majority of region members. In Figure 2, we present a schematic of the procedure. The data points of a region comprise a large “central subcluster” and two smaller subclusters (Figure 2(a)). The two smaller subclusters are treated as periphery members and are pruned when building the training data for

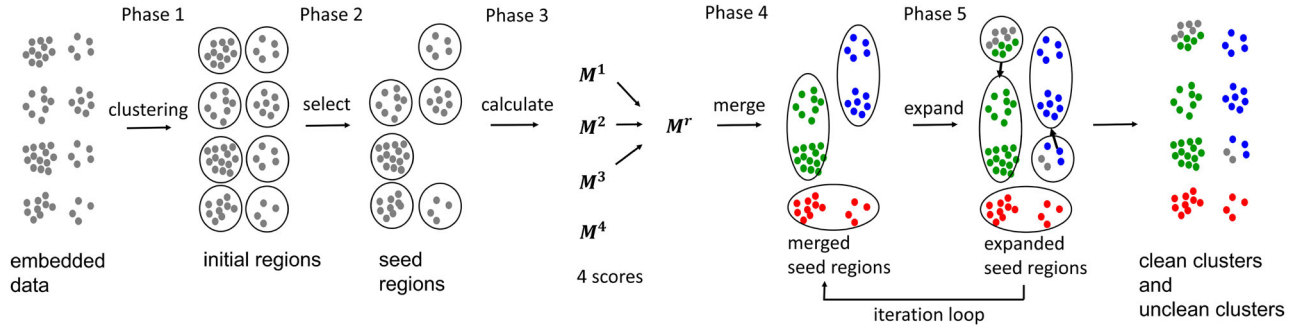


Figure 1. A schematic of the ME algorithm. Gray dots denote the images that have not yet been assigned to pseudo labels or have not been confidently classified. Colored dots denote the images that have been assigned to pseudo labels.

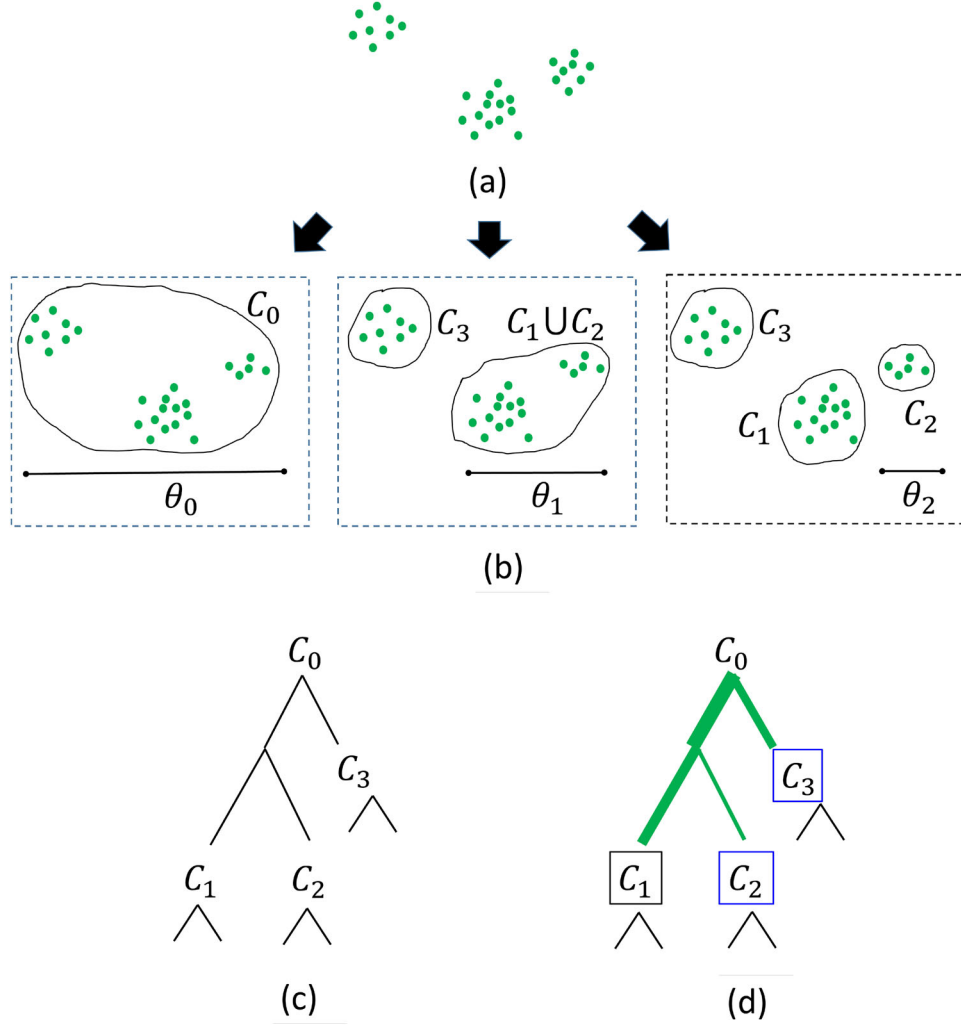


Figure 2. (a) The center and periphery members are assigned to the same region. (b) An undirected graph is built that connects data points whose distances are within the threshold, and define the connected components $C_0 - C_3$ of the graphs as subclusters. (c) Subsumption relationships of subclusters represented as a tree. (d) The two smaller subclusters C_2 and C_3 (blue boxes) are pruned as periphery members and the central subcluster C_1 (black box) is retained. Branch width (green bars) reflects the size of split data points.

CNN classifiers. To detect these periphery members, we apply a hierarchical clustering-like process to construct a family of subclusters whose relationships are represented by a tree. We vary the distance threshold with a decreasing order ($\theta_0 - \theta_2$ in Figure 2(b)), build an undirected graph connecting data points whose distances are within the threshold, and find the connected components of the graph. These subclusters constitute subsumption relations and are represented by a tree (Figure 2(c)).

All the data points form one subcluster C_0 at the largest distance threshold θ_0 . C_3 splits from $C_1 \cup C_2$ at threshold θ_1 , and C_2 splits from C_1 at threshold θ_2 . We iteratively traverse along the largest branches of the tree, stop when encountering a major split where no single child dominates other children in terms of their sizes, and assign the parent subcluster of the major split as the central subcluster. In Figure 2(d), C_1 is the central subclusters and C_2 and C_3 are periphery members.

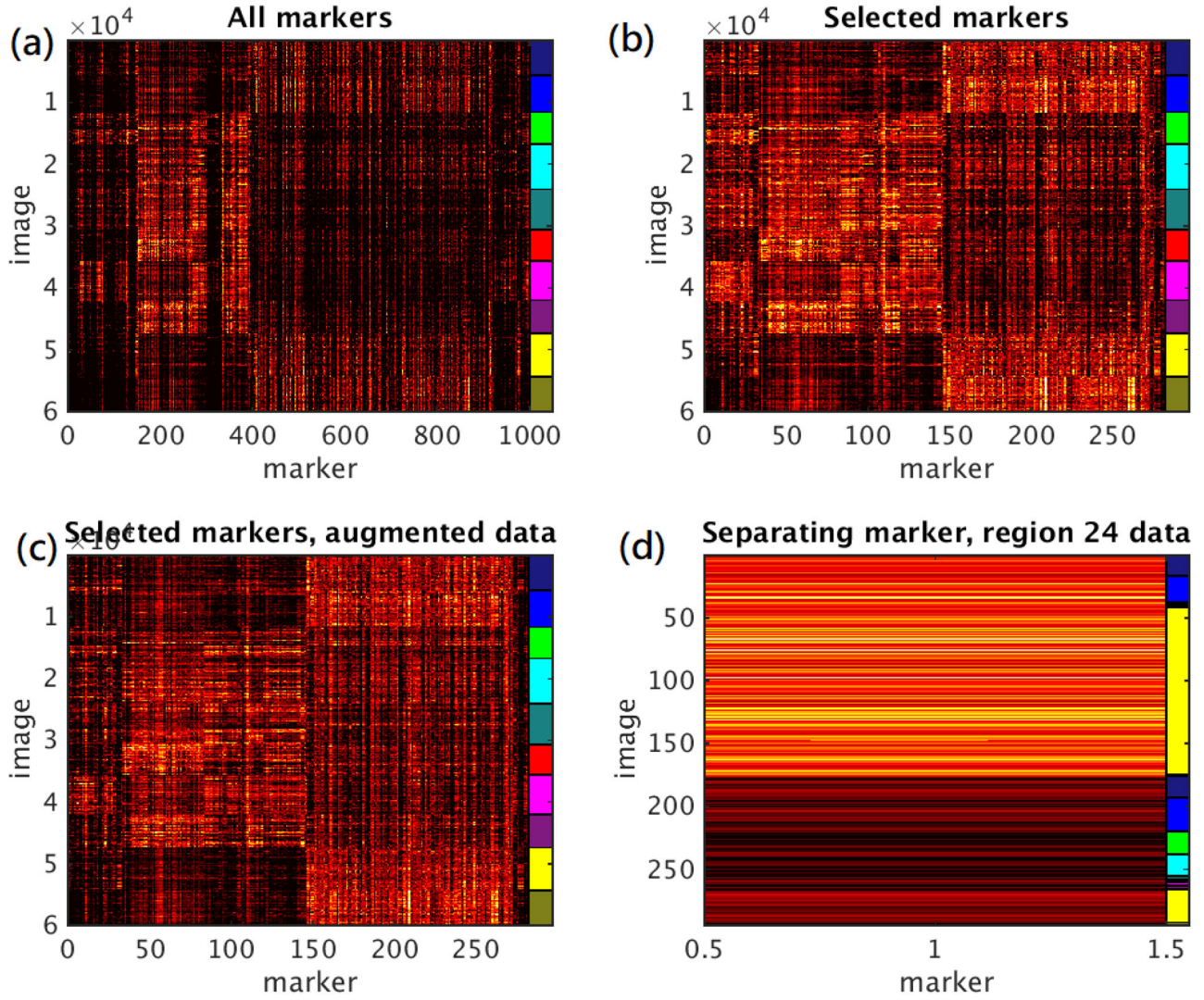


Figure 3. (a) The VGG-16 outputs of 1000 features for 60,000 CIFAR-10 images. The 200 regions have been sorted according to their true dominant class labels, and the images have been sorted by those regions. The true dominant class labels of the sorted regions are displayed in the colored vertical panel at right. (b) The same data restricted to 282 selected markers. (c) The VGG-16 outputs of the augmented CIFAR-10 data for the selected markers. (d) The VGG-16 outputs of one separating marker in region 24. The majority group has high output values for the marker and is dominated by class label 8 (ship). The minority group has low output values for the marker and displays heterogeneous true class labels. The minority group is deemed to comprise periphery members and is removed from the training data.

Informative marker-based filtering procedure: In this procedure, a few markers of the embedded data are assumed to carry relevant information about the underlying class labels. The “informative markers” in each region are determined and then periphery members likely possessing nondominant class labels in terms of those markers are identified and removed, to create the training set for Phase 2. We illustrate this procedure in Figure 3. In Figure 3(a), we present the full VGG-16 activation outputs of 1000 features, which display considerable variations between images of different regions. Candidate markers manifesting selective activation in each targeted region are selected (282 selected markers in Figure 3(b)). In order to be informative markers, these candidate markers should not be sensitive to data augmentation, hence, should display similar levels of activation in their corresponding data augmented by rotations and translations (Figure 3(c)). Candidate markers which exhibit differential output values in the augmented data are removed. In the similar fashion, periphery members are assumed to be sensitive to data augmentation, and are the minority in a region by

definition. For a given marker, the member images are grouped according to the marker’s output values. Markers that exhibit large gaps between the high-value and low-value portions of member images in the original and augmented data are selected, allowing the construction of a partition informed by the selected marker (Figure 3(d)). The member images which consistently belong to the minority group across all selected markers form the periphery data points. However, if there are too many or no separating markers, then the assumption about the informative markers is likely invalid and so the periphery members are not reported.

2.2. Phase 2: Selecting Seed Regions

Seed regions serve as the initial training data for subsequent CNN predictions. To make complete but compact training data and pseudo labels they should have two desirable properties: (a) they represent the entire data as every region is similar to at least one seed region, (b) they are mutually dissimilar.

In Phase 2, ME selects two widely-separated regions attracting similar numbers of members, and then iteratively adds regions distinct from all other existing seed regions. The distinctiveness between a candidate region and existing seed regions is quantified according to two criteria: (a) the candidate region is distant from all existing seed regions in the embedded space; (b) the classifier trained on the existing seed regions differs substantially from the modified classifiers upon replacing any seed region with the candidate region.

More precisely, we introduce the following notations and equations. Denote \mathbf{X} as the collection of data points in the embedded space, R_1, R_2, \dots, R_K the K ($K = 200$) regions generated by spectral clustering where each $R_i \subset \mathbf{X}$ is a subset of data points, and $R_i \cap R_j = \emptyset$ for each (i, j) pair. Define $d(R_i, R_j)$ as the average Euclidean distance between members of regions R_i and R_j :

$$d(R_i, R_j) \equiv \frac{1}{|R_i||R_j|} \sum_{\{\mathbf{x} \in R_i, \mathbf{x}' \in R_j\}} d(\mathbf{x}, \mathbf{x}') \quad (1)$$

We also build classifiers by incorporating the data points of selected regions as the training data. Suppose r_1, \dots, r_l are l selected regions. Denote $f(r_1, \dots, r_l) : \mathbf{X} \rightarrow \{0, \dots, l-1\}$ as an l -class classifier built from members in r_1, \dots, r_l as the training data. Here, we employ a simple and fast k -nearest neighbor (k -NN) classifier with $k = 10$ because $f(r_1, \dots, r_l)$ is incurred many times in seed region selection.

The first two seed regions are selected based on the criteria that: (a) their average distance is large among all region pairs; and (b) they attract similar numbers of data points. Criterion 2 is included in order to avoid selecting outlier regions that are distant from most other regions in the dataset. Specifically, we evaluate $d(R_i, R_j)$ for all pairs of regions, sort them in a decreasing order, and choose the first region pair for which the ratio of the numbers of images assigned to the two regions is below a threshold θ_j . The selected region pair $S = \{r_1, r_2\}$ constitutes the initial seed regions. From these initial seed regions we iteratively select the next seed region by a max-min criterion, and add \hat{r} to S :

$$\hat{r} = \arg \max_{r \notin S \cup F} \min_{r' \in S} d(r, r'), \quad (2)$$

where the candidate regions exclude the existing seed regions S and the filtered regions F . A filtered region within the set F may contain data points having mixed class labels or those that do not share class labels with their neighboring regions. The max-min criterion iteratively selects the region which is the most dissimilar to all existing seed regions in terms of their average Euclidean distances. We impose max-min selection on a fixed number of iterations to obtain a list S of seed regions.

Because seed regions are chosen to facilitate accurate predictions of underlying class labels, we also filter seed regions in terms of classification outcomes. We treat each seed region as a unique class and train a reference classifier from the unfiltered members of the existing seed regions (obtained in Phase 1). When a new seed region is added, we check whether it can replace any existing seed region in a modified classifier and yield similar prediction outcomes. A seed region remains on the list only if it cannot be replaced by any preceding seed regions in terms of the classification outcomes.

Here, we give a more concrete description of the criteria for filtering seed regions. Denote $f_0 \equiv f(r_1, \dots, r_K)$ as the reference k -NN classifier trained from data points in the K existing seed regions. A modified classifier $f_{r-i,r} \equiv f(r_1, \dots, r_{i-1}, \boxed{r}, r_{i+1}, \dots, r_K)$ replaces members of region r_i with members of region r in the training data. Region r is distinct from K existing seed regions if $f_{r-i,r}$ yields different classification outcomes from f_0 for each $i = 1, \dots, K$. We consider the differences of the prediction outcomes in the entire dataset and restricted to each class. For each class i , define $n_i^{\text{diff}}(r)$ (3) as the number of data points with different predicted labels between f_0 and $f_{r-i,r}$, $n_i^{\text{conf}}(r)$ (4) as the number of data points with predicted label i by either f_0 or $f_{r-i,r}$ but not both, $n_i^{\text{cons}}(r)$ (5) as the number of data points with predicted label i by both f_0 and $f_{r-i,r}$, and $r_i^{\text{conf}}(r)$ (6) as the size difference between the incompatible and compatible prediction sets restricted to class i and normalized by the size of the compatible prediction set.

$$n_i^{\text{diff}}(r) \equiv \left| \{ \mathbf{x} : f_0(\mathbf{x}) \neq f_{r-i,r}(\mathbf{x}) \} \right|. \quad (3)$$

$$n_i^{\text{conf}}(r) \equiv \left| \{ \mathbf{x} : f_0(\mathbf{x}) = i, f_{r-i,r}(\mathbf{x}) \neq i \} \right| + \left| \{ \mathbf{x} : f_0(\mathbf{x}) \neq i, f_{r-i,r}(\mathbf{x}) = i \} \right|. \quad (4)$$

$$n_i^{\text{cons}}(r) \equiv \left| \{ \mathbf{x} : f_0(\mathbf{x}) = f_{r-i,r}(\mathbf{x}) = i \} \right| \quad (5)$$

$$r_i^{\text{conf}}(r) \equiv \frac{n_i^{\text{conf}} - 2n_i^{\text{cons}}}{n_i^{\text{cons}}}. \quad (6)$$

The aforementioned indicators enable us to quantify the similarity between a newly incorporated seed region r and all other previously incorporated seed regions r_1, \dots, r_K in terms of their classification outcomes. We select several existing seed regions $S_c(r)$ whose confusion ratios $r_i^{\text{conf}}(r)$ surpass a threshold. Members of $S_c(r)$ are the most similar to r among the existing seed regions. We then pick the member in $S_c(r)$ that gives the smallest $n_i^{\text{diff}}(r)$ and denote this quantity $n_{\min}^{\text{diff}}(r)$. Among the list S of seed regions selected by the max-min criterion, we keep the ones that have sufficient high $n_{\min}^{\text{diff}}(r)$ scores. The procedure of selecting seed regions is reported in Algorithm 1 of supplementary Text S1. The hyperparameters of the Phase 2 algorithm and their default values are explained in supplementary Text S1.

2.3. Phase 3: Calculating Four Pairwise Scores

Multiple seed regions may share the same underlying class labels because some of their differences in the entire embedded space may reflect non-semantic attributes of images such as the lighting conditions of the background or the geometric transformations of objects. Therefore, to provide more precise pseudo label assignments to the training data, it is necessary to merge seed regions that likely share the same underlying class labels. Information about the true class labels typically resides in local patterns of subsets of features rather than the global patterns of all features. Accordingly, deep neural networks such as CNNs better capture these local patterns. Therefore, we reason that semantic similarities between seed regions can be inferred from CNN classification outcomes. We propose four types of pairwise scores for seed regions, based on CNN classification outcomes,

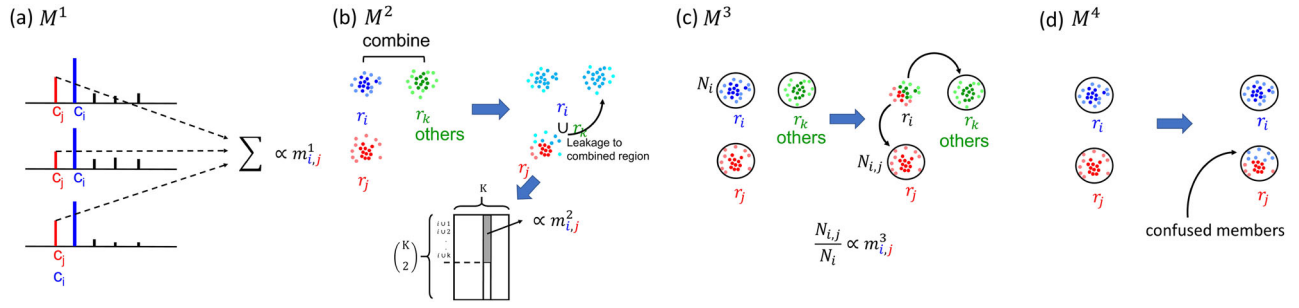


Figure 4. Illustration of the procedures of computing four types of pairwise scores (a) M^1 , (b) M^2 , (c) M^3 , (d) M^4 .

to provide information for merging seed regions that likely share the same underlying class labels, as shown in Figure 4.

Co-contribution score M^1 : The most direct way to build a CNN classifier from K seed regions is to treat each seed region as one class. CNN generates a normalized activation output vector $\mathbf{f}(\mathbf{x}) \equiv (f_1(\mathbf{x}), \dots, f_K(\mathbf{x}))$ representing the probabilities of all class predictions. Intuitively, two seed regions are similar if their prediction probabilities are concurrently high among many images. We translate this intuition into a $K \times K$ co-contribution score matrix M^1 with the following procedure. According to the training data of K seed regions, we run CNN classifiers n_r rounds ($n_r = 5$ in this work), and then the CNN classifier of each round maps the embedded vector of an image \mathbf{x} to a K -component vector $\mathbf{f}^l(\mathbf{x})$, where l is from 1 to n_r . The n_r CNN classifiers share the same training data but may not give identical outputs due to the stochastic gradient descent (SGD) procedure of parameter estimation. The co-contribution score m_{ij}^1 is proportional to the sum of the output values of class j over the data points with consensus label i among the n_r predictions:

$$M^1 = [m_{ij}^1] \equiv \frac{1}{|T_i| \cdot n_r} \sum_{\mathbf{x} \in T_i} \sum_{l=1}^{n_r} f_j^l(\mathbf{x}), \quad (7)$$

where i and j are from 1 to K ; T_i denotes the set of data points with consensus label i ; and $f_j^l(\mathbf{x})$ is the j th component of $\mathbf{f}^l(\mathbf{x})$. The score is normalized to make the entries range in $[0, 1]$.

Leakage score M^2 : The next simplest way to rearrange pseudo labels of seed regions is to combine each pair of seed regions into one class in the training data and build $\binom{K}{2}$ predictors with $(K-1)$ classes accordingly. Similarity of seed regions is indirectly revealed by the prediction outcomes of the merged classes. Some data points assigned to a third class in the K -class predictor will “leak” to the merged class in a $(K-1)$ -class predictor if the third class shares the same true label with either one of the merged classes. Consequently, we construct a leakage score matrix M^2 to formalize this observation. First, we build a K -class predictor $f_0(\mathbf{x})$ by assigning each seed region a unique class. $f_0(\mathbf{x})$ is trained by the same procedure as the CNN classifiers in calculating M^1 , but reports only the most probable predicted class label rather than the probability vector of class predictions for each image, and it is generated once instead of n_r times. For each pair (i, k) of seed regions, we consider a $(K-1)$ -class predictor $f_{i \cup k}(\mathbf{x})$ merging seed regions i and k in the same class. By comparing $f_0(\mathbf{x})$ with each $f_{i \cup k}(\mathbf{x})$, we construct a $\binom{K}{2} \times K$ leakage matrix

$L_{i \cup k, j}$ to count the number of data points that are assigned to class j according to $f_0(\mathbf{x})$ and to the combined class $i \cup k$ according to $f_{i \cup k}(\mathbf{x})$:

$$L_{i \cup k, j} \equiv |\{\mathbf{x} : f_0(\mathbf{x}) = j, f_{i \cup k}(\mathbf{x}) = (i \cup k)\}|. \quad (8)$$

If seed regions i and j share the same true class labels, then a considerable number of data points will leak from class j in $f_0(\mathbf{x})$ to the combined class $i \cup k$ in $f_{i \cup k}(\mathbf{x})$ for multiple third classes k . Consequently, on the j th column of L , the high leakage counts should be enriched in the combined classes containing seed region i . We quantify this enrichment by comparing two subsets of L matrix entries on column j . $L(R_i, j)$ denotes the leakage counts from class j to all merged classes containing seed region i , and $L(R_{1:K}, j)$ denotes the leakage counts of the entire column j . We quantify enrichment of high leakage counts in $L(R_i, j)$ by calculating $W(L(R_i, j), L(R_{1:K}, j))$, the Wilcoxon rank sum test (Wilcoxon 1945) of $L(R_i, j)$ against the background $L(R_{1:K}, j)$, and its p -value $P_W(L(R_i, j), L(R_{1:K}, j))$. The leakage score m_{ij}^2 becomes:

$$M^2 = [m_{ij}^2] \equiv P_W(L(R_i, j), L(R_{1:K}, j)), \quad (9)$$

Replacement score M^3 : Apart from merging seed region pairs in the same class, we also remove each seed region from the training data and build $(K-1)$ -class predictors accordingly. The prediction outcomes of these reduced classifiers readily manifest similarity between seed regions. After removing a seed region from the training data, the images originally assigned to the removed seed region are then reassigned to its most similar seed regions. Consequently, the replacement score m_{ij}^3 is the fraction of data points that are reassigned from pseudo class i in the original K -class predictor $f_0(\mathbf{x})$ to pseudo class j in the $(K-1)$ -class predictor $f_{-i}(\mathbf{x})$ with seed region i removed from the training data:

$$M^3 = [m_{ij}^3] \equiv \frac{|\{\mathbf{x} : f_0(\mathbf{x}) = i, f_{-i}(\mathbf{x}) = j\}|}{|\{\mathbf{x} : f_0(\mathbf{x}) = i\}|}. \quad (10)$$

Confusion score M^4 : An even simpler way to quantify seed region similarity without rearranging pseudo labels is to inspect the confusion matrix in the training data of $f_0(\mathbf{x})$. Seed regions i and j likely share the same true class labels if many seed region i members are assigned to seed region j by $f_0(\mathbf{x})$. Formally, we define the confusion score matrix M^4 as

$$M^4 = [m_{ij}^4] \equiv \frac{|\{\mathbf{x} : \mathbf{x} \in r_i, f_0(\mathbf{x}) = j\}|}{|\{\mathbf{x} : \mathbf{x} \in r_i\}|}. \quad (11)$$

2.4. Phase 4: Merging Seed Regions from Pairwise Scores

We propose an algorithm to merge seed regions by using the four pairwise scores in a hierarchical manner. It consists of three steps. First, we combine $\mathbf{M}^1 - \mathbf{M}^4$ by calculating their sums and rank sums, and apply graph-theoretical algorithms to the combined matrices to subdivide seed regions into a few coarse-grained components of smaller sizes. Second, for each component we enumerate all partitions respecting conditions derived from the pairwise scores, and select the best partition(s) according to several criteria derived from those pairwise scores. Third, we refine the selected partitions by identifying potential singletons, seed regions which likely possess class labels distinct from all others. Below, we elaborate on each step of its algorithm. The hyperparameters of the Phase 4 algorithm and their default values are explained in supplementary Text S1.

2.4.1. Subdividing Seed Regions into Coarse-Grained Components

The first step is to subdivide the seed regions into a few coarse-grained components so that the inter-component members are unlikely to share the same underlying class labels. To do so, we convert the nonnegative valued $\mathbf{M}^1 - \mathbf{M}^4$ matrices into a binary symmetric matrix G_s specifying potentially mergeable relationship. $G_s(i, j) = 1$ if seed regions i and j potentially share the same underlying class labels. The connected components of G_s constitute the outputs of this step.

We construct G_s using three criteria derived from the pairwise scores. First, two seed regions are adjacent if members of one seed region tend to be confused for the other seed region in CNN predictions according to \mathbf{M}^4 . Second, we rank-transform $\mathbf{M}^1 - \mathbf{M}^3$ of each row separately, add their ranks to form the reduced rank scores \mathbf{M}^r , and make $G_s(i, j) = G_s(j, i) = 1$ if i and j are on the top n_τ ($n_\tau = 3$ in our setting) partners of each other in \mathbf{M}^r . Third, we complement the second criterion by taking the values of $\mathbf{M}^1 - \mathbf{M}^3$ scores into account beyond their ranks. We construct another reduced sum score matrix \mathbf{W} , where $w(i, j) = m^1(i, j) + \frac{-\log(m^2(i, j) + \frac{1}{n_{perms}}))}{\log(n_{perms})} + m^3(i, j)$, and n_{perms} denotes the number of random permutations for the Wilcoxon rank sum test (100,000 in our setting). We then apply a variation of the spectral clustering algorithm (Ng, Jordan, and Weiss 2002) to \mathbf{W} and add links between members within the same clusters to G_s . The connected components of the modified G_s graph are the outputs of step 1. The procedure for generating the coarse-grained components is described in Algorithm 2 in supplementary text S1.

2.4.2. Identifying Optimal Partitions of Coarse-Grained Components

The coarse-grained components separate the seed regions that unlikely share the same underlying class labels, yet they do not possess sufficient resolution to delineate the seed regions of distinct class labels. In the second step, we enumerate all unique and valid partitions of each coarse-grained component by introducing additional constraints. Then, we propose several scores and apply them to narrow down the valid partitions. The outputs of this step are the refined partitions of all coarse-grained components.

G_s specifies mergeable relationships between pairs of seed regions. To complement G_s we construct another matrix G_r of “repulsive relationships” of seed region pairs. $G_r(i, j) = 1$ if $m^2(i, j)$, the p -value of the leakage score, exceeds a threshold value ($\theta_p = 0.2$ in our setting). For each coarse-grained component, we enumerate all unique partitions and narrow down the valid partitions by introducing two additional constraints pertaining to G_r . First, no two repulsive seed regions could be assigned to the same cluster. Second, if two seed regions are assigned to the same cluster, then they are connected by at least one shortest path in G_s , such that all seed regions along the path are assigned to the same cluster and comprise no repulsive relations.

The resulting valid partitions are all compatible with the mergeable and repulsive relations G_s and G_r but may be too numerous to report in the final output. We reduce the number of valid partitions to just a few by introducing several metrics for the quality of valid partitions. Intuitively, a good partition should possess two properties: (a) intra-cluster seed region pairs are generally more similar than inter-cluster pairs in terms of reduced rank and sum score matrices \mathbf{M}^r and \mathbf{W} ; and (b) it prefers a few large clusters to many small clusters.

We quantify property 1 for each valid partition by checking whether the top-ranking entries in \mathbf{M}^r and \mathbf{W} matrices are enriched with intra-cluster links. For \mathbf{M}^r , we sort the off-diagonal entries in increasing order and assign each entry a binary indicator for intra-cluster links. Then we construct a step function $y(x)$ of the rank of the sorted entries x , and $y(x)$ counts the number of intra-cluster links among the top x entries. Strong enrichment of intra-cluster pairs among the top entries would rapidly increase the value of $y(x)$. Thus, the enrichment score ζ_1 is the difference between areas under $y(x)$ and a linear function $y_0(x)$ connecting the $(0, 0)$ and $(x_{\max}, y(x_{\max}))$. The enrichment score ζ_2 of \mathbf{W} is constructed analogously by sorting the entries in decreasing order. The joint enrichment score is $\zeta = \zeta_1 + \zeta_2$. This enrichment score is derived from Gene Set Enrichment Analysis (GSEA) commonly used in bioinformatics (Subramanian et al. 2005).

Given the enrichment scores and the selection criteria we choose a valid partition based on the following procedure. First, we find all valid partitions with the minimum number of isolated seed regions. Among the selected partitions, we select the ones with the maximum enrichment score ζ . If multiple partitions remain, then we choose the ones that maximizes the size of the smallest cluster in the partition. The procedure for step 2 is described in Algorithm 3 in supplementary Text S1.

2.4.3. Identifying Singletons and Refining the Partitions Accordingly

The partitions generated by steps 1–2 (Algorithms 2 and 3 in supplementary Text S1) disfavor singletons because the parsimony criterion prefers the partitions with smaller numbers of clusters. In step 3, we propose a procedure to identify the potential singletons and to refine the partitions accordingly.

If the marker information of the fixed embedding is not used to merge seed regions, then we repeat Algorithm 3 in supplementary Text S1 for each cluster induced from the reported partitions. In each cluster, we enumerate all valid subpartitions

with the same procedure for step 2. For each subpartition, we identify the seed region pairs within the same subclusters and assign their sum of the W entries as the subpartition score. We then retain only the valid subpartitions with the maximal score. A seed region is labeled as a singleton if it is isolated in all the valid subpartitions.

If the marker information of the fixed embedding is used to merge seed regions, then we find singletons by checking whether each seed region possesses a substantial number of markers distinguishing it from all other seed regions. More concretely, for each ordered pair of seed regions (i, j) and each feature k of the embedded data, we extract the members of regions r_i and r_j and their feature k values $X(k, r_i)$ and $X(k, r_j)$. We employ a previously proposed statistic $p_{diff}(i, j, k)$ (Tiong et al. 2022) in Algorithm 4 in supplementary Text S1 to quantify the difference between the distributions of $X(k, r_i)$ and $X(k, r_j)$. A high $p_{diff}(i, j, k)$ denotes that feature k has significantly higher values in seed region i than in seed region j . Feature k is a singleton marker of seed region i if: (1) $p_{diff}(i, j, k)$ exceeds a threshold value for each $j \neq i$; and (2) the median of $X(k, r_i)$ is within a top percentile of all feature k values. Finally, a seed region is labeled as a singleton if its number of singleton markers exceeds a threshold.

2.5. Phase 5: Expanding Training Data and Clustering Images

After Phase 4, each seed region is assigned to a pseudo label. Hence, the clustering challenge is transformed into a classification problem that can be solved using CNN algorithms. CNNs require a relatively large training set, but the initial training data induced by seed regions is usually small. Therefore, we iteratively expand the training data by including other regions that give homogeneous prediction labels. For each iteration, we train 5 CNN classifiers with the same training data and report the consensus labels of the five prediction outcomes for all images (−1 if no consensus labels exist). To justify the choice of running the CNN classifiers on the same training data five times, we varied the number of CNN rounds from 1 to 15 and reported the fractions of images with consensus predictions in a subset of the “PlantVillage” dataset in supplementary Table S2. The fraction of images with consensus labels declines with the number of CNN rounds, yet the rate of decline is quite mild. We found that 84% of the images have consensus labels in 15 CNN rounds. We chose the 5 CNN rounds as it was in the first relatively stable regime. The difference in consensus rates between consecutive CNN rounds, as shown in column 3 of supplementary Table S2, drops below 0.006 when the number of CNN rounds exceeds 5, whereas the difference varies between 0.008 and 0.06 for fewer rounds.

For each region outside the training data, we count the fraction of members possessing the dominant (the most frequent) prediction labels. If the fraction of these members exceeds a threshold value, then this region can be more confidently included in the training data. Consequently, we select the regions with homogeneous prediction labels, and add their members with the dominant prediction labels to the training data for the CNN predictions in the next iteration. Iterations

continue until no additional training data are included. The cluster assignments represent the final classification outcomes. Images without consensus predicted class labels are not assigned to any cluster.

The expansion procedure subdivides regions into two groups according to the criterion whether a region is included in the training data or not. Regions that are included in the training data possess homogeneous prediction labels across their members. We mark their members with the dominant prediction labels “clean” images. All other images (including the members of regions with heterogeneous prediction labels and members of regions with homogeneous prediction labels but that are assigned to nondominant labels) are marked as “unclean” images. We elaborate further on the importance of prediction homogeneity in the Results section. The iterative expansion algorithm is described in Algorithm 5 in supplementary Text S1.

Note that we have slightly modified the Phase 2 algorithm and substantially modified the Phase 4 algorithms compared to the previous version (Chen, Tiong, and Yeang 2022). In the previous ME framework, we added a procedure in Phase 2 to filter out periphery members of regions using their silhouette values, which has been excluded from the procedure described herein. Although the formulas of the three pairwise scores ($M_1 - M_3$) remain intact (Phase 3), the current version of the Phase 4 algorithm incorporates more information about pairwise scores, yet is simpler in terms of its procedures and the numbers of hyperparameters. In the previous version, the coarse-grained components were connected components of a graph G_s in which edges denoted seed region pairs with top-ranking pairwise scores with respect to each other. In the current version, apart from the rankings we also considered the magnitudes of the pairwise scores when constructing G_s . To avoid imposing threshold values on the pairwise scores, we employed spectral clustering to the weight matrix W constructed by summing over the three pairwise scores, and then connected seed regions belonging to the same spectral clusters. In the previous version, we deployed a sequence of filters to select one remaining partition after enumerating all valid partitions of seed regions. In the current version, we quantify the strength of a valid partition based on the enrichment of intra-component pairs among the top-ranking pairs of the pairwise scores, and identify the partitions with the highest enrichment scores. In the previous version, we uncovered the seed region pairs with contradictory information about proximity (i.e., they were nearest neighbors in the embedded data but were assigned to different components according to the CNN prediction outcomes), and selected one member as a singleton. In the modified version, we apply the partition-finding algorithm (Algorithm 3 in supplementary Text S1) again to each component of a valid partition and identify the singletons from the subpartitions.

3. Results

We have validated our improved ME framework on five datasets, and conducted several analyses to justify its utility in clustering images and to specify clustering quality for different subsets of images. We found that clustering outcomes were robust for various hyperparameters at Phases 1, 2, and 4. Next, we assessed

Table 1. Details of the five assessed datasets and their characteristics.

Dataset	Size	Raw pixels	Embedded vector	Classes
MNIST	55,000	28×28	1×3	10
CIFAR-10	60,000	32×32	1×1000	10
NCT	100,000	224×224	1×1000	9
PlantVillage	43,217	256×256	1×1000	9
RVL	50,000	1000×750	1×1000	10

NOTE: The embedded vector refers to the fixed embedded vector size. For experimental convenience, we reduced the PlantVillage dataset from 54,303 to 43,217 images and the number of classes from 38 to 9. We also reduced the RVL dataset from 440,000 to 50,000 images and the number of classes from 16 to 10.

Table 2. Time complexity and actual running time of each phase of the ME framework for the PlantVillage dataset. Numbers within parentheses denote the running times for the original ME framework (Chen, Tiong, and Yeang 2022). n : number of images; r : total number of regions; K : number of seed regions; l : maximum size of coarse-grained components in Phase 4; m : size of each (raw or embedded) image; T : maximum number of epochs in training CNNs.

Phase	Time complexity	Running time
1	$O(n^3)$	7 hr
2	$O(mn^2)$	1.5 hr (2.47 hr)
3	$O(K^2 nmT)$	4 hr
4	$O(l^l)$	30 sec (1 min)
5	$O(nmT)$	30 min

heterogeneity of predicted class labels in all regions, labeling them as “clean” and “unclean” regions accordingly. The drastically different clustering accuracy between these clean and unclean regions highlights a major benefit of our algorithm. We modified the ME framework by employing a second embedding to expand the clean regions and improve clustering accuracy. We further examined the embedded data of the merged seed regions in two datasets, pointing out the sources of confusion in their clustering outcomes. Finally, we compared the clustering accuracy of our ME framework with a number of baseline and state-of-the-art methods, demonstrating that ME performs better or at least competitively against all these methods. In Table 1, we summarize the size and dimensionality of the five datasets encompassing various application domains. MNIST consists of images of ten handwritten digits (LeCun, Cortes, and Burges 2021). CIFAR-10 consists of color images of ten object classes, including several types of animals and artifacts (CIFAR-10 2021). NCT comprises the microscopic images of nine cell types from colorectal cancer tissues (Kather, Halama, and Marx 2018). PlantVillage consists of images of crop plant leaves with infectious diseases (Hughes and Salathé 2015), from which we selected 43,217 images from nine crops species. The RVL dataset harbors grayscale images of 16 types of documents (Harley, Ufkes, and Derpanis 2015), from which we selected 50,000 images of ten document classes.

Our computing environment comprised a Dell Precision Tower 7910 featuring an x64-based “Intel® Xeon® CPU E5-2667 v4 3.20 GHz” processor with 256 GB RAM, and an NVIDIA GPU “Quadro P4000” graphics card. In Table 2, we present crude estimates of time complexity for each phase of the ME framework and actual running times on the PlantVillage dataset. Time complexity is dependent on the bottleneck procedures at each phase: spectral clustering for Phase 1, evaluation of average distances between region pairs for Phase 2, training the CNN models for combinations of seed region pseudo labels for Phase 3, enumerating all valid partitions of coarse-grained

components for Phase 4, and training and applying CNNs to a relatively small number of iterations for Phase 5. The procedures in Phases 1 and 3 are the most time-consuming given that spectral clustering and training CNNs $O(K^2)$ times are computationally expensive. Our original ME algorithms differ from those in the revised ME framework only for Phases 2 and 4. The previous Phase 2 algorithm requires about twice the running time of the current Phase 2 algorithm since calculating silhouette values also entails a time complexity $O(mn^2)$.

3.1. Robustness Analysis

First, we show that the ME clustering outcomes were robust for variations of hyperparameter values in the algorithms. The sole hyperparameter in Phase 1 is the number of regions generated by spectral clustering. We split the data into 150, 200, and 250 regions and present the clustering outcomes from ME prediction according to two criteria (Table 3(a)). The “consensus” criterion (ACC_{5con}) assigns an image to the consensus predicted label if it exists and an unspecified mark otherwise. The “majority” criterion (ACC_{maj}) reports the majority of the predicted labels of every image.

To evaluate clustering accuracy, we generated a confusion matrix \mathcal{C} of CNN predictions where $\mathcal{C}(i, j)$ represents the number of images with true class label i and predicted class label j . We then aligned the true and predicted labels by permuting columns of \mathcal{C} to maximize the sum of diagonal entries. The accuracy rate is represented by the ratio of the sum of diagonal entries and the sum of all entries in the permuted \mathcal{C} .

In Table 3(a), we report the clustering accuracy rates of the MNIST and CIFAR-10 datasets for varying numbers of regions (K250, K200, K150) and two prediction criteria. The accuracy rates remain stable (difference of approximately 3% or less) across the numbers of seed regions for each dataset and prediction criterion. In those three region sets (K250, K200, K150), the accuracy rates for consensus predictions are moderately higher than the majority predictions since the former is restricted to the images with reliable predicted labels. However, the price for a higher accuracy rate is a smaller coverage of valid predictions. In the subsequent analyses, we set the number of regions to 200.

Most of the other hyperparameters occur in Phases 2 and 4. Due to the computational costs of running “global robustness tests” (e.g., varying parameter values and assessing clustering accuracy rates as shown in Table 3(a)), we applied “local robustness tests” for each hyperparameter separately. For each hyperparameter of interest, we varied its value while keeping default values for all other hyperparameters. We reported the outputs of the corresponding phase (seed region indices for Phase 2

Table 3. (a) Clustering accuracy rates of MNIST and CIFAR-10 data for three sets of regions (K150, K200, K250) according to two prediction criteria (ACC_{5con} , ACC_{maj}).

Dataset	Number of regions	Accuracy	
		ACC_{5con}	ACC_{maj}
MNIST	K250	0.977 (48844)	0.937
	K200	0.989 (50089)	0.964
	K150	0.941 (49157)	0.904
CIFAR-10	K250	0.599 (47883)	0.554
	K200	0.628 (51262)	0.586
	K150	0.604 (46705)	0.556

(a) Clustering accuracy rates for three sets of regions (K150, K200, K250)

Phase	Parameter	Mean/Std	Phase	Parameter	Mean/Std
2	maxseeds (N_t)	0.749/0.172	4	prednumthre	0.991/0.011
2	diffratiothre (θ_d)	0.640/0.233	4	pvalthre (θ_p)	0.982/0.019
2	rthre (θ_r)	0.629/0.318	4	ntoprakthre (n_τ)	0.984/0.017
2	nclassesthre	1/0	4	smallclustersizethre	1/0
2	ratiothre (θ_r)	1/0	4	jumpfoldthre	1/0
2	foldthre	1/0	4	gapfoldthre (r_γ)	1/0
4	sizethre (θ_s)	1/0	4	smallratiothre	1/0
4	confusionratiothre (θ_c)	1/0	4	reducedrankscorethre	0.976/0.018

(b) Local robustness test results

NOTE: The numbers of images reported by the five-consensus predictions (ACC_{5con}) are also reported in parentheses. For example, an accuracy 0.977 for the 48,844 five-consensus images implies that approximately 47,720 images are correctly clustered. (b) Local robustness test results on the 16 hyperparameters of Phases 2 and 4 algorithms, represented by the mean and standard deviation of the similarity matrices.

Table 4. Accuracy results for clean and unclean images achieved by the ME framework.

Dataset	ACC_{5con}	Clean	Unclean	ACC_{maj}
MNIST	0.989	0.973 (50721)	0.039 (4279)	0.964
CIFAR-10	0.628	0.756 (26236)	0.367 (33764)	0.586
NCT	0.789	0.871 (66083)	0.399 (32484)	0.749
PlantVillage	0.730	0.829 (27438)	0.214 (15779)	0.646
RVL	0.580	0.624 (36191)	0.193 (13809)	0.543

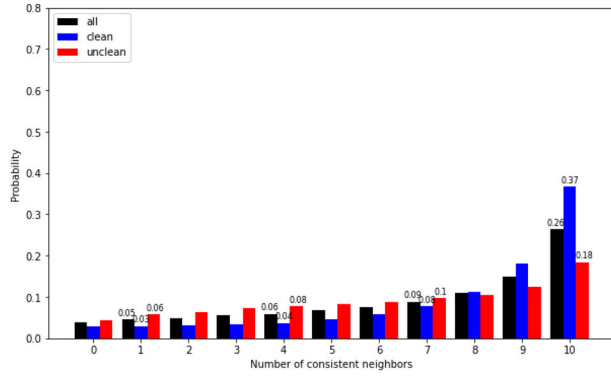
NOTE: CIFAR-10 refers to the CIFAR-10 embedded data generated by VGG-16. NCT, PlantVillage and RVL refer to the embedded data generated by ResNet-18. The numbers of clean and unclean images reported according to the ACC_{5con} criterion are also shown in the parentheses. For example, there are 50,721 clean images in the MNIST dataset, and an accuracy of 0.973 indicates that 49,352 of them are correctly clustered.

and seed region partitions for Phase 4), and compared the similarities of these outputs (Jaccard similarity scores for Phase 2 outputs and Rand Indices for Phase 4 outputs). Furthermore, for each hyperparameter, we generated a similarity matrix by comparing the outputs of pairs of hyperparameter values. In Table 3(b), we present the summary statistics of the similarity matrices for six hyperparameters in Phase 2 and 10 hyperparameters in Phase 4 of the PlantVillage dataset. The complete similarity matrices of these hyperparameters are reported in supplementary Table S3. Among the six Phase 2 hyperparameters, maxseeds, diffratiothre and rthre have relatively higher fluctuations in their similarity matrices (mean/standard deviation of 0.749/0.172, 0.640/0.233, and 0.639/0.318, respectively), but the similarities centered on their default values are relatively stable. The remaining three hyperparameters display perfect similarities (mean/standard deviation of 1/0) among all values considered. Among the ten Phase 4 hyperparameters, six exhibited hyperparameters have perfect similarities among all values considered and the remaining four present non-perfect but still strong similarities among all values considered. For instance, mean and standard deviation of the similarity matrix for pvalthre are 0.982 and 0.019, respectively. Consequently, ME outputs are robust to variations in hyperparameter values. We have fixed the values of parameters presenting perfect local robustness in the source code of the ME program, and allowed users to vary the remaining parameter values. In the current

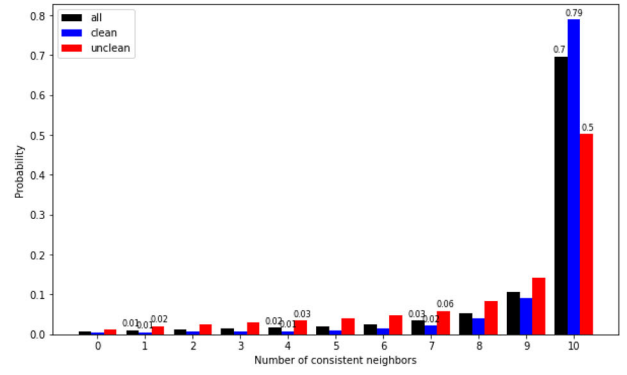
setting, for each hyperparameter, we identified the range of values which gave rise to similar clustering outcomes. The value with the largest range of robust values was selected. Similar approaches for determining hyperparameter values have been adopted in several major image clustering algorithms using deep neural networks, such as IIC (Ji, Henriques, and Vedaldi 2019), SCAN (Van Gansbeke et al. 2020), and TSUC (Han et al. 2020)

3.2. Heterogeneity of Predicted Labels Across Regions

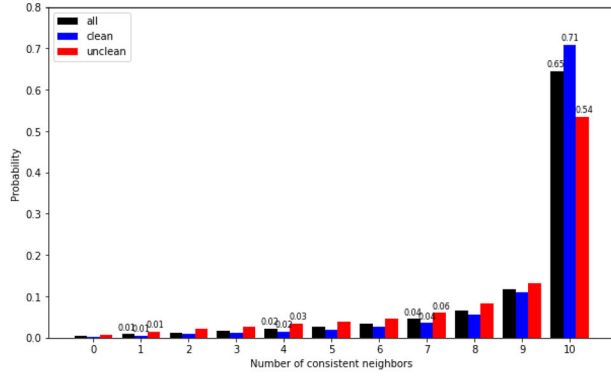
A unique feature of our ME framework is how it demarcates embedded images in terms of heterogeneity of predicted class labels. A region is marked clean if the fraction of the members assigned to the dominant predicted label exceeds a threshold value, and marked unclean otherwise. Members of clean regions with the dominant predicted labels are marked clean and otherwise unclean. Intriguingly, the clustering accuracy rates between clean and unclean regions are drastically different in all five datasets we assessed, as shown in Table 4. For the MNIST data, only a small number of images are unclean and have near zero accuracy rates, whereas the vast majority of images are clean and have near perfect accuracy rates. For the remaining datasets, accuracy rates for clean images are approximately 2–4 fold those of unclean images, but there are substantial numbers of unclean images, ranging from 30% to 60% of the datasets. Therefore, the majority of clustering errors occur in unclean regions.



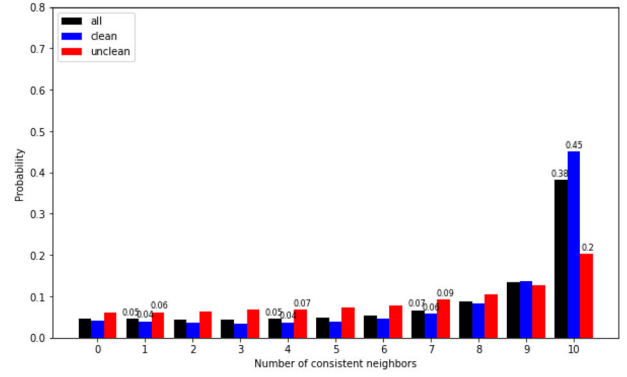
(a) CIFAR10 embedded data by VGG-16



(b) NCT embedded data by ResNet-18



(c) PlantVillage embedded data by ResNet-18



(d) RVL embedded data by ResNet-18

Figure 5. Distributions of the number of consistent labels among ten nearest neighbors.

We suspected that the unclean regions with heterogeneous predicted labels were error-prone due to the incongruence between the geometry of the embedded data and their true cluster structure. To substantiate this speculation, we demonstrated that members in clean regions were more likely to share the same true class labels with their nearest neighbors than unclean regions. For each image, we counted the fraction of its ten nearest neighbors in the embedded space having the same true class label, and then calculated the distributions of the consistent neighbor counts (from 0 to 10) in the entire dataset, clean image subset, and unclean image subset. In Figure 5, we show that a considerably higher fraction of clean images tend to have higher numbers of consistent neighbors. In the CIFAR-10 dataset with VGG-16 embedding, 37% of clean images share the same true class labels with all 10 of their nearest neighbors, but only 18% of unclean images do so. The difference in image counts with consistent neighbors between clean and unclean images is slightly lower but still substantial for the NCT dataset with ResNet-18 embedding (79% vs. 50%). The distinction in distributions between clean and unclean regions is also apparent for the PlantVillage and RVL datasets.

3.3. Improving Clustering Accuracy via a Second Embedding

The close links between heterogeneity and the accuracy of predicted labels indicate a potential means of improving clustering

accuracy. Since the accuracy rates of clean images far exceed those of unclean images, we could potentially improve the accuracy of our ME framework by converting some unclean regions into clean ones. As previously mentioned, the geometry of the embedded data in unclean regions is misaligned with the underlying cluster structure. Thus, applying another embedding to the same image data may reshape their geometry to better align with the underlying cluster structure. Consequently, we targeted the unclean regions of the first embedding and attempted to clean some of them in a second embedding. More concretely, first we applied the ME framework to the first embedded data to identify clean and unclean regions and images. Then we retrained the CNN with the second embedded data of the clean images and reapplied Phase 5 to predict the pseudo class labels of the unclean images, allowing us to identify unclean regions, which became clean in the second embedding. We term these regions “unclean-clean” since they are unclean and then clean in the first and second embeddings, respectively. Analogously, “unclean-unclean” regions are unclean in both embeddings. Finally, we added the unclean-clean images to the training data of the second embedding and classified all images accordingly.

In Figure 6, we display summary statistics in the process of combining two embeddings in two datasets. For CIFAR-10, we first applied VGG-16 and then ResNet-18 embeddings to the data, generating 103 clean regions and 1 unclean-clean region. For RVL, we first applied ResNet-18 and then VGG-16 embeddings to the data, resulting in 136 clean regions and 2 unclean-clean regions. Clustering accuracy rates were

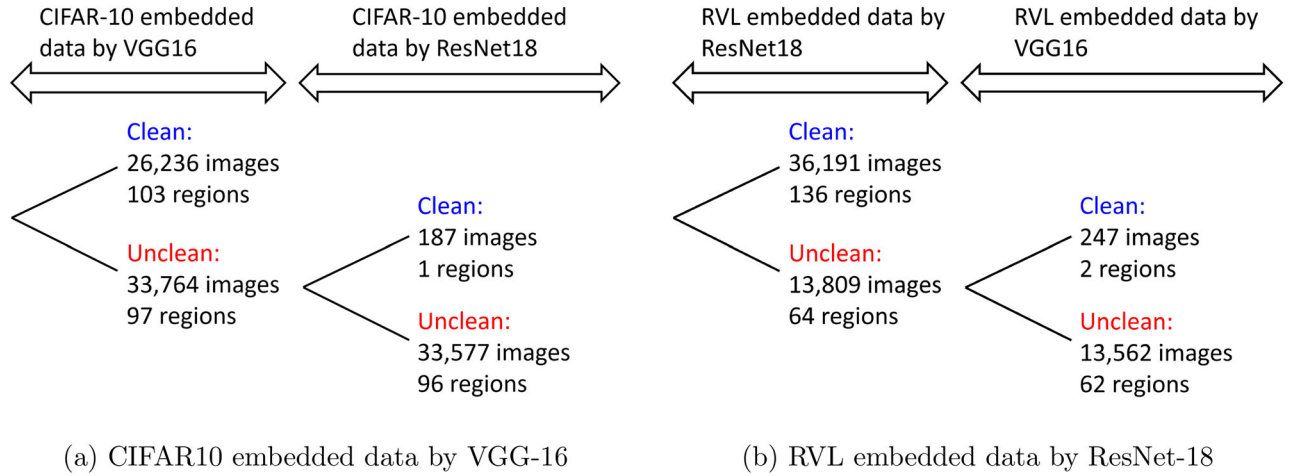


Figure 6. Integration of two embedded datasets enables the ME framework to distinguish more clean images.

Table 5. Overall accuracy rates (based on the majority criterion) for various clustering algorithms and ME frameworks.

Method	MNIST	CIFAR-10	NCT	PlantVillage	RVL
K-means	0.572/0.733	0.229/0.527	0.367/0.572	0.218/0.466	0.192/0.308
SPECTRAL CLT	0.696/0.966	0.247/0.598	0.417/0.628	0.248/0.541	0.228/0.339
Triplets	0.525	0.205	N/A	N/A	N/A
AE	0.812	0.314	N/A	N/A	N/A
GAN2015	0.828	0.315	N/A	N/A	N/A
JULE2016	0.964	0.272	N/A	N/A	N/A
DEC2016	0.843	0.301	N/A	N/A	N/A
DAC2017	0.978	0.522	N/A	N/A	N/A
DEEP CLT2018	0.656	0.374	N/A	N/A	N/A
ADC2018	0.992	0.325	N/A	N/A	N/A
IIC	0.984	0.576	N/A	N/A	N/A
SCAN	N/A	0.883	0.908	0.922	0.804
TSUC	N/A	0.810	0.738	0.684	0.526
ME	0.964	0.586	0.749	0.646	0.543
ME second embedded data	N/A	0.610	0.761	0.645	0.621

NOTE: For k-means and spectral clustering, accuracy rates on raw data or embedded data are presented as “raw data/embedded data.” For the MNIST dataset, its embedded data was encoded by t-SNE. For the remaining datasets, the embedding data was encoded using either VGG-16 (CIFAR-10) or ResNet-18 (others). Accuracy rates were calculated by running the corresponding methods of the top two and bottom four rows on the five datasets and copied from (Ji, Henriques, and Vedaldi 2019), (Van Gansbeke et al. 2020), and (Han et al. 2020) for other methods on the MNIST and CIFAR-10 datasets. The values of SCAN and TSUC on the CIFAR-10 dataset were also copied from prior studies. Missing entries denote either the respectively programs were unavailable or unnecessary to run advanced clustering algorithms on the MNIST dataset.

marginally improved by introducing the second embedding (the last two rows in Table 5). Only a few unclean regions were cleaned by the second embedding, since the two embedding models are trained on the same ImageNet data.

3.4. Sources of Confusion in the CIFAR-10 and NCT Datasets

Clustering errors arise primarily in Phase 4 when seed regions are merged to determine pseudo labels, with seed regions having different true class labels potentially being merged or seed regions with identical true class labels being assigned to distinct pseudo labels. The effects of such confusion are amplified over subsequent steps, as additional training data are incorporated according to the initial assignments of merged seed regions. It is difficult to deduce general rules about how such confusion occurs. We inspected the t-SNE projections of the merged seed regions in CIFAR-10 and NCT data and provide intuitive explanations for how confusion arose from these two specific datasets.

First, we visualized the merged seed regions of CIFAR-10 dataset in Figure 7. The three-dimensional t-SNE projections of

26 seed region members are colored according to their dominant true class labels. Confusion is apparent for two merged pseudo label classes. In the upper-left corner, one seed region of images of horses (gray) coalesces with two seed regions of deer (purple). In the lower-central corner, two seed regions of airplanes (blue) coalesce with two seed regions of ships (yellow-green). All other merged seed regions possess identical dominant true class labels. These two types of confusion are intuitive, as horses and deer display similar physical builds and sometimes the same body color, and airplanes and ships have similar streamlined shapes and background blue hues of sky and sea.

Next, we visualized 16 merged seed regions of NCT dataset in Figure 8. Representative thumbnails of all seed regions are illustrated since tissue structures may not be easily interpreted from their annotations. The only area of confusion is on the middle-left portion of the plot where the seed region of cancer-associated stroma (STR, gray) is merged with one seed region of smooth muscle (MUS, brown), and this latter is separated from another MUS seed region. The stromal images indeed resemble the images of smooth muscle images in the upper seed region in terms of their textures. In contrast, the smooth muscle images of

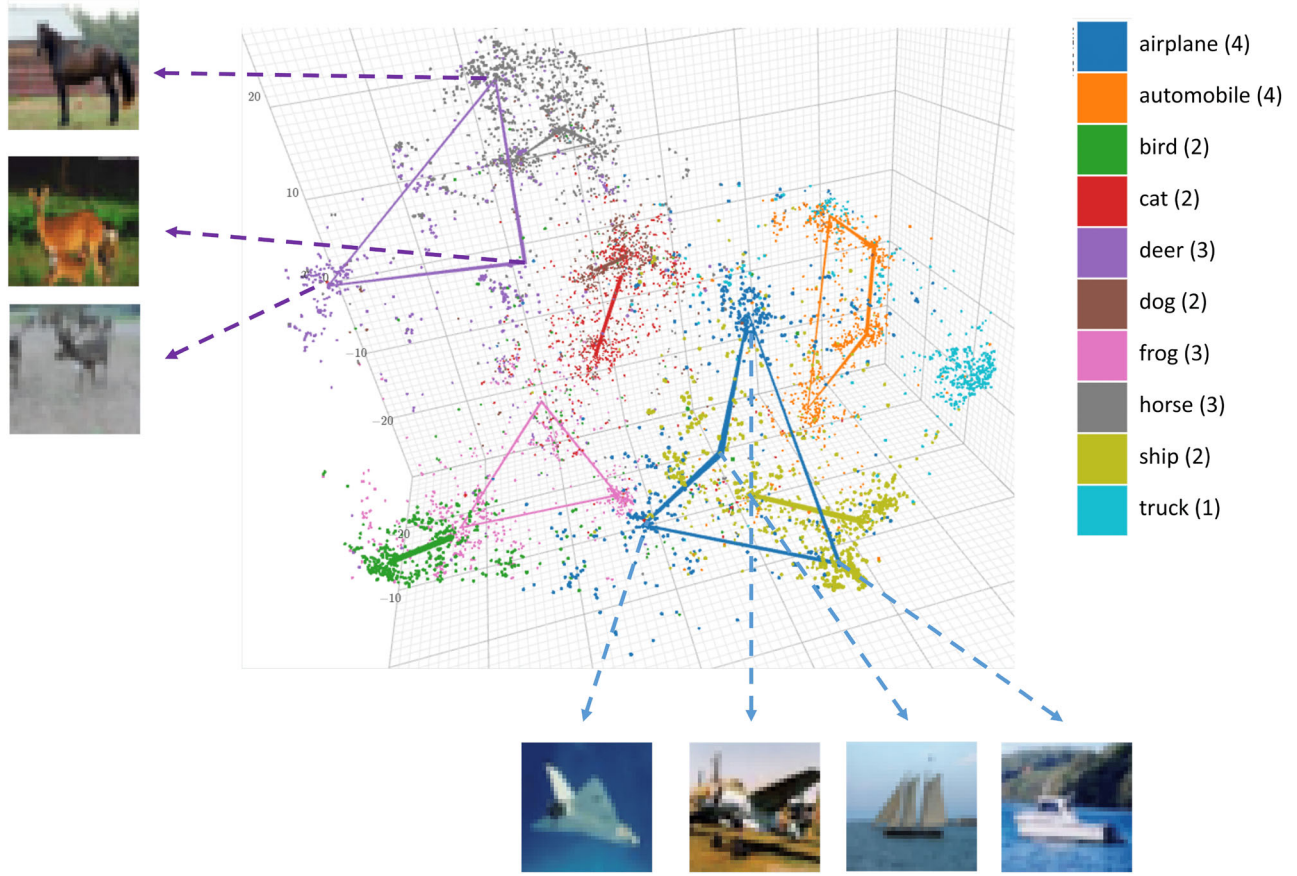


Figure 7. t-SNE projections of members from 26 seed regions in the CIFAR-10 embedded data. Colors denote the dominant true class labels of the seed regions they belong to. Numbers alongside the colored bar indicate the numbers of dominant seed regions. Centroids of seed regions are connected by solid lines if they are merged together in Phase 4. The erroneously merged seed regions are annotated by the representative thumbnail images.

the lower seed region are distinct from those of the upper seed region as they are quite distinctly colored and fibrous. All of the remaining merged seed regions share the same tissue class labels and texture characteristics.

3.5. Accuracy of Clustering Results

Finally, we compared the accuracy rates of our ME framework (using the majority criterion) against 13 other clustering methods across five datasets in Table 5. We found that SCAN outperforms other methods across four of the datasets (MNIST was excluded for SCAN and TSUC since it was an easy task for most of the clustering methods). The next best algorithms include TSUC, the original ME framework and the revised ME with the second embedding (presented herein). The two ME algorithms proved superior to TSUC for the NCT and RVL datasets, yet inferior to TSUC for the CIFAR-10 and PlantVillage datasets. The two baseline methods (k-means and spectral clustering) generally performed worse than SCAN, TSUC and both ME approaches, though spectral clustering with the selected fixed embeddings yielded comparable performance to the ME approach on the MNIST and CIFAR-10 datasets. The accuracy rates for the other methods on MNIST and CIFAR-10 are mostly comparable or inferior to the top-tier algorithms and the baseline algorithms with fixed embeddings. Although the ME approaches do not outperform all benchmark methods, they do

yield comparable or even superior performance relative to some state-of-the-art self-supervised learning algorithms (TSUC and IIC). This outcome is remarkable given the rigid constraints (fixed embedding, a simple CNN architecture, a common loss function) of the ME frameworks.

In addition to accuracy rates, we also assessed the quality of clustering by the Rand index (Rand 1971), which determines the fraction of data point pairs accurately assigned to identical or different clusters. In Table 6, we report the Rand indices for the clustering results by the two conventional clustering methods and the two ME approaches. The Rand index values were higher than the accuracy rates, yet their respective orders on different methods and datasets were generally congruent. The Rand indices of other methods were not available as their clustering labels were not provided. For instance, the SCAN and TSUC packages only report overall accuracy rates rather than the cluster labels of individual images.

4. Discussion

In this study, we presented a more complete and extended version of the Merge & Expand framework to cluster image data with a fixed embedding, and have conducted a more thorough experimental analysis to justify its utility. First, we demonstrate that the clustering outcomes are robust against varying values of hyperparameters deployed in the ME framework. Then, after

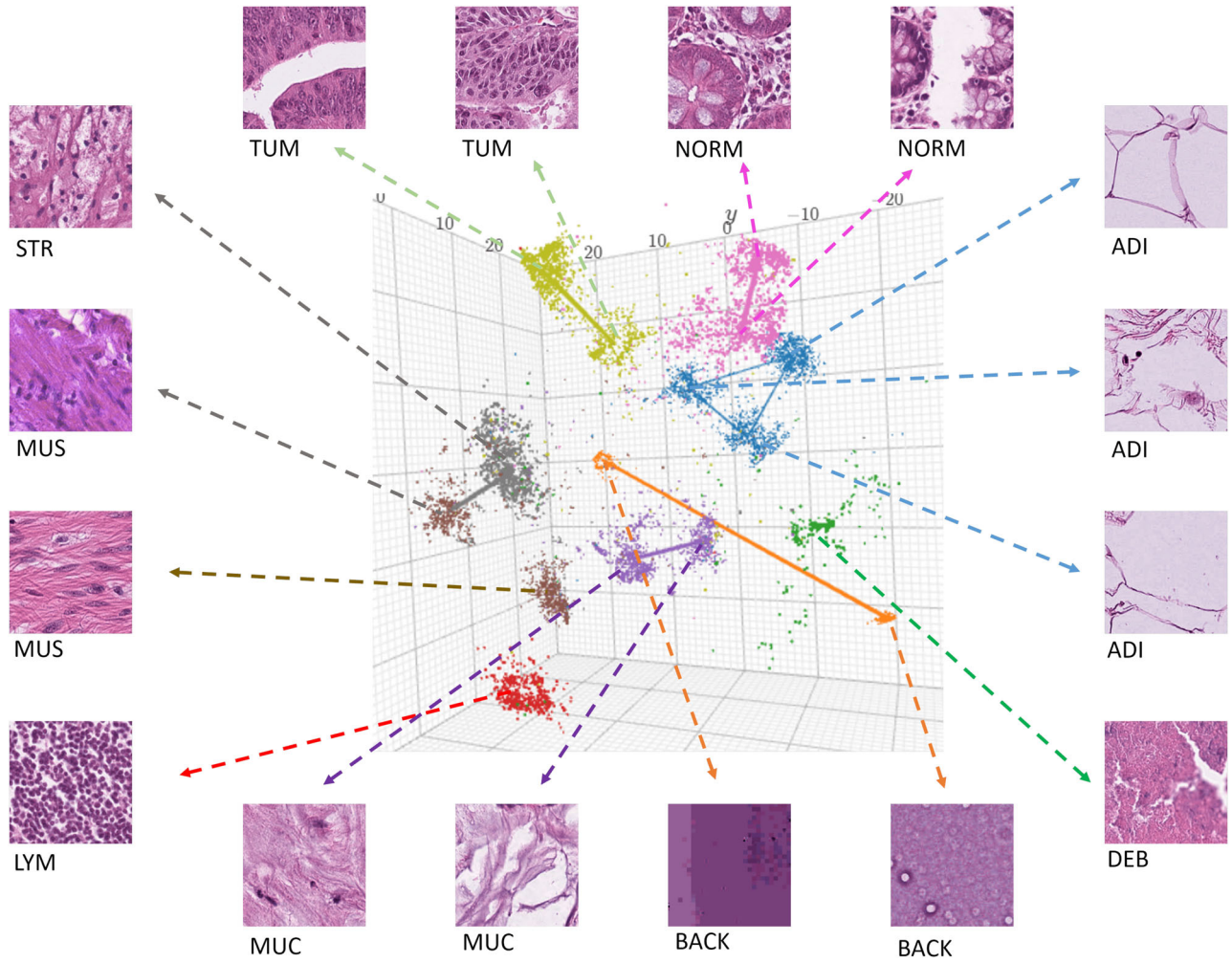


Figure 8. t-SNE projections of members from 16 seed regions in NCT embedded data. Colors and representative thumbnail images are similarly presented as in Figure 7. The nine class labels are: adipose (ADI), background (BACK), debris (DEB), lymphocytes (LYM), mucus (MUC), smooth muscle (MUS), normal colon mucosa (NORM), cancer-associated stroma (STR), and colorectal adenocarcinoma epithelium (TUM).

Table 6. Rand index values of two conventional clustering algorithms and ME frameworks.

Method	MNIST	CIFAR-10	NCT	PlantVillage	RVL
K-means	0.872/0.914	0.823/0.877	0.801/0.876	0.769/0.733	0.817/0.811
SPECTRAL CLT	0.905/0.987	0.821/0.882	0.827/0.845	0.752/0.863	0.818/0.826
ME	0.984	0.872	0.907	0.882	0.879
ME second embedded data	N/A	0.878	0.912	0.881	0.893

NOTE: The Rand Index is a similarity score ranging from 0 to 1, inclusive, where 1 represents a perfect match.

assessing heterogeneity of predicted labels in each region, we show that it is a strong indicator of alignment between the geometry of the embedded space and the underlying cluster structure, denoting regions as being “clean” or “unclean” accordingly. Consequently, we modified ME framework by introducing a second embedding to “clean” the unclean regions and improve clustering accuracy. A major source of clustering error is the confusion in merging seed regions. We examined the projections of the merged seed region members and their representative images in the CIFAR-10 and NCT datasets and provide intuitive explanations for the discrepancies. Finally, we have evaluated the clustering accuracy rates of several algorithms on five datasets and found our enhanced ME framework performed competitively despite using a fixed embedding, a simple CNN architecture, and a common loss function.

A major source of errors in the ME algorithm is the erroneous assignment of pseudo labels in the training data. Since the true class labels of images are not used in clustering, we alleviate this problem through several indirect approaches. First, we attempt to prune the peripheral members of a region that may possess distinct class labels from the majority of members in that region. We elaborate on this procedure in Phase 1 and Figure 2. Peripheral members of a region are images that differ from the majority either in the entire embedding space or in selected markers. Second, in Phases 2–4, we establish the initial training data, where each merged seed region represents a unique class. Third, in Phase 5, we iteratively incorporate regions with homogeneous predicted class labels in the training data. Thus, in a general sense, all five phases consist of procedures designed to handle noise and outliers.

A major issue for self-supervised learning algorithms is misalignment between the geometry of the embedded data and their underlying cluster structure. We tackled this issue by two approaches. First, we found that heterogeneity of predicted labels in a neighborhood of the embedded space was informative about the clustering quality, and used this outcome to demarcate the regions with good or poor alignments between the embedded space and cluster structure. Second, we modified our ME framework by introducing a second embedding to recognize regions that had heterogeneous predicted class labels in the first embedding and homogeneous predicted class labels in the second embedding. These so-called “unclean-clean” regions are likely to be accurately clustered in the second embedding. In our experimental analysis, these unclean-clean regions did display improved clustering accuracy but were scarce in the data. Hence, the overall improvement in clustering accuracy was limited. We suspect that this limited improvement is attributed to the strong dependency of the two embedding models in our experiments. Both embedding models—VGG-16 and ResNet-18—were trained on the same ImageNet dataset. Hence, the geometry of the embedded data from these models is very similar, so the second embedding only marginally improves the clustering accuracy. An optimal second embedding model should capture orthogonal features from the first model, thereby clustering images that are distinct in the first embedding. In principle, we could substantially improve clustering accuracy with fixed embeddings if multiple distinct DNN models pre-trained on distinct large datasets were applied.

Heterogeneity is a natural factor that is difficult to avoid and affects clustering outcomes. The unique feature of our ME framework is that it demarcates embedded images into clean and unclean regions based on heterogeneity. This is the key idea behind the development of the ME framework. Furthermore, we introduced a second embedding to select the subtle clean images from the unclean images identified in the first embedding. We summarize the logical flow of how we leverage the heterogeneity of predicted labels to improve clustering accuracy. First, we acknowledge that clean regions, characterized by homogeneous predicted labels, have much higher clustering accuracy than unclean regions, which are characterized by heterogeneous predicted labels, as shown in Section 3.2. Second, we aim to improve clustering accuracy by converting some unclean regions into clean regions through the introduction of a second embedding, as detailed in Section 3.3. Specifically, we select images from the clean regions of the first embedding and then use the second embedding, training on the clean regions, to further classify the unclean regions into clean ones. Third, we incorporate the regions that become clean in the second embedding into the training data to enhance the clustering of images. Fourth, this procedure moderately improves clustering accuracy, as shown in Figure 6. We discussed that the second embedding in our experiment yields limited improvement in clustering accuracy since the two embedding models are trained on the same ImageNet data.

As mentioned in our Introduction, a fixed embedding is beneficial or essential when the raw data are inaccessible for privacy or security concerns, but did not explicitly show how privacy could be preserved in the deep learning setting through the fixed embedding experimental analysis. Here, we discuss

possible integration of the privacy preservation techniques with the ME framework. Privacy is an important issue, especially when multiple participants collaborate in certain deep learning tasks but do not want to reveal their own data to others. Since the participants wish to collaborate but may not necessarily trust each other, there is a tradeoff between the model performance and the extent of information sharing. In a review paper, Antwi-Boasiako et al. (2021) assessed various distributed deep learning techniques and classified them into two major categories. For direct methods, participants share their data to a central server, and the server builds a global model from the joint data from all participants. To avoid revealing privacy information, participants apply certain privacy preservation operations to the raw data before sending them to the server. Common techniques include differential privacy and homomorphic encryption. The former method injects random noise into the data such that the output of the algorithm cannot reveal the private information of the input. The latter method encrypts the data to erase private information but also enables invariant under common mathematical operations such as addition and multiplication. For indirect methods, participants do not share their data to the server. Instead, each participant trains a local model with her/his own data and sends the weights or gradients of the model to the server. The server aggregates the local weights/gradients to build a global model and then sends the global weights/gradients to all participants to update their local models. Direct methods can be viewed as a special case of fixed embedding. The differential privacy or homomorphic encryption operations are predefined functions to protect privacy/security rather than being tuned to fit a machine learning task. Therefore, our ME framework is applicable to the direct methods. In our experimental analysis, all of the datasets underwent transformations, rendering it difficult to convert the embeddings back to the raw images. In an extreme case, the MNIST images had been projected by t-SNE onto 3D vectors, but the clustering results on the projected data were still almost perfect. Like other fixed embeddings, the geometry of the encrypted data is not necessarily aligned with the cluster structure of the raw data. Our ME framework marks the clean/unclean regions, thereby indicating the quality of the alignment. Nevertheless, designing an encryption operation that simultaneously preserves the cluster structure and erases privacy information remains a considerable challenge.

The four pairwise scores have their pros and cons. The main advantage of M^1 (co-contribution) is the computational efficiency. Without exhausting combinations of seed regions for classifications one only need to incur a K-class CNN classifier (K denotes the number of seed regions) five times and report the relative contributions from last layer neurons. The main disadvantage is that it is tied to neural network models and does not apply to other types of classifiers (k-NN, SVM, logistic regression, etc). The main advantage of M^2 (leakage) is the statistical power in quantifying the similarity of each pair of seed regions. Similarity of a seed region pair (i,j) is determined by the leakage from j to the merged classes of i and each of the remaining seed regions k, and the same operation by swapping i and j. Furthermore, the M^2 score is represented by the Wilcoxin rank sum p-value, hence, M^2 can better filter out dissimilar seed region pairs with insignificant p-values. The main disadvantage is the computational cost to perform CNN predictions by

merging each pair of seed regions. The main advantage of M^3 (replacement) is the complement of the disadvantages of M^1 and M^2 . M^3 in principle can be applied to any multi-class classifiers (in contrast to the shortcoming of M^1), and the computational cost is substantially lower than M^2 (the number of replacements is linear to the number of seed regions compared to the number of pairwise merges which is quadratic). M^4 (confusion) serves primarily as a sanity check to remove the seed regions which have high training errors. The computational bottleneck is on M^2 as it incurs $\binom{K}{2}$ CNN predictors.

Apart from incorporating multiple fixed embeddings, our ME framework can be extended in multiple directions. First, the pairwise scores of seed regions are currently generated by applying CNN to all pairwise combinations (M^2) and singleton removals (M^3) of seed regions. To overcome the bottleneck in Phase 3, it would be possible to select a smaller subset of these partitions that yields similar pairwise scores. This approach bears some resemblance to multi-class classification with binary classifiers using error-correction code design techniques (Nguyen et al. 2021). Second, the ME framework could be applied to diverse data modalities beyond images such as text, speech, and graphs, and incorporate more powerful pre-trained models such as GPT or more flexible DNN architectures such as transformers. Third, it would be a natural next step to extend the ME framework to tackle the problems of semi-supervised learning where both labeled and unlabeled data are provided, and of active learning where one could choose to label small amount of data.

Supplementary Materials

Supplementary Text S1: Pseudo code description of the algorithms in each phase of ME. Explanation of the hyperparameters and their default values at each phase of the ME framework.

Supplementary Table S2: Consensus rates of CNN predictions on PlantVillage dataset. We built a CNN model using the training data obtained from Phase 5 of ME framework and applied the same CNN model multiple times to predict the class labels. Column 1 shows the number of CNN rounds and column 2 reports the fraction of images that have identical predicted labels over multiple CNN rounds.

Supplementary Table S3: Local robustness test outcomes of hyperparameters in Phases 2 and 4. The first sheet reports the hyperparameter names, their phases, and their values in the local robustness tests. Each subsequent sheet reports a similarity matrix by comparing the Phase 2/4 outputs under a reference hyperparameter value with those under another hyperparameter value. Rows and columns denote the reference and varying hyperparameter values. Phase 2 outputs are seed region indices, so similarities are represented by the Jaccard similarities between two sets of seed region indices. Phase 4 outputs are clustering of seed regions, so similarities are represented by the Rand index values between two clustering outcomes of the same seed regions.

Acknowledgments

We thank Dmytro Luzhbin and Vahid Golderzahi for reviewing the article and providing comments.

Disclosure Statement

The authors report there are no competing interests to declare.

Funding

The work is partially supported by Institute of Statistical Science, Academia Sinica, and National Science and Technology Council, Taiwan (grant numbers: NSTC 108-2118-M-001-001-MY2 and NSTC 113-2118-M-002-011).

Data Availability Statement

The codes, ReadMe, small data for test-running the codes, and information necessary for obtaining the codes for the ME framework proposed in this article are available at <https://github.com/chyeang/ME>. We also include an explanation of the hyperparameters of the ME framework and their default values in supplementary text S1.

References

- Antwi-Boasiako, E., Zhou, S., Liao, Y., Liu, Q., Wang, Y., and Owusu-Agyemang, K. (2021), "Privacy Preservation in Distributed Deep Learning: A Survey on Distributed Deep Learning, Privacy Preservation Techniques Used and Interesting Research Directions," *Journal of Information Security and Applications*, 61, 102949. [15]
- Bengio, Y., Courville, A., and Vincent, P. (2013), "Representation Learning: A Review and New Perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35, 1798–1828. [1]
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., et al. (2020), "Language Models are Few-Shot Learners," in *Advances in Neural Information Processing Systems* (Vol. 33), eds. H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, pp. 1877–1901, Curran Associates, Inc. [1]
- Caron, M., Bojanowski, P., Mairal, J., and Joulin, A. (2019), "Unsupervised Pre-Training of Image Features on Non-Curated Data," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2959–2968. [1]
- Chang, J., Wang, L., Meng, G., Xiang, S., and Pan, C. (2017), "Deep Adaptive Image Clustering," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 5879–5887. [1]
- Chen, Y.-B., Tiong, K.-L., and Yeang, C.-H. (2022), "Clustering Image Data with a Fixed Embedding," in *2022 21st IEEE International Conference on Machine Learning and Applications (IEEE ICMLA)*, pp. 891–896. [2,8,9]
- CIFAR-10. (2021), "The cifar-10 Dataset," available at <https://www.cs.toronto.edu/~kriz/cifar.html>. Accessed: 2021. [9]
- Cordonnier, J.-B., Loukas, A., and Jaggi, M. (2020), "On the Relationship between Self-Attention and Convolutional Layers," in *International Conference on Learning Representations 2020 (ICLR 2020)*, pp. 1–22. Open Review. [1]
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009), "Imagenet: A Large-Scale Hierarchical Image Database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255. [2]
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2020), "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale." arXiv preprint arXiv:2010.11929. [1]
- Garg, S., Tsipras, D., Liang, P. S., and Valiant, G. (2022), "What Can Transformers Learn in-Context? A Case Study of Simple Function Classes," in *Advances in Neural Information Processing Systems* (Vol. 35), eds. S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, pp. 30583–30598, Curran Associates, Inc. [1]
- Glasmachers, T. (2017), "Limits of End-to-End Learning," in *Proceedings of the Ninth Asian Conference on Machine Learning*, Volume 77 of *Proceedings of Machine Learning Research*, eds. M.-L. Zhang and Y.-K. Noh, pp. 17–32, Seoul, Republic of Korea: Yonsei University, PMLR. [1]
- Han, S., Park, S., Park, S., Kim, S., and Cha, M. (2020), "Mitigating Embedding and Class Assignment Mismatch in Unsupervised Image Classification," in *Computer Vision – ECCV 2020*, eds. A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, pp. 768–784, Cham: Springer International Publishing. [10,12]

- Han, X., Zhang, Z., Ding, N., Gu, Y., Liu, X., Huo, Y., Qiu, J., Yao, Y., Zhang, A., Zhang, L., Han, W., Huang, M., Jin, Q., Lan, Y., Liu, Y., Liu, Z., Lu, Z., Qiu, X., Song, R., Tang, J., Wen, J.-R., Yuan, J., Zhao, W. X., and Zhu, J. (2021), "Pre-Trained Models: Past, Present and Future," *AI Open*, 2, 225–250. [2]
- Harley, A. W., Ufkes, A., and Derpanis, K. G. (2015), "Evaluation of Deep Convolutional Nets for Document Image Classification and Retrieval," in *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pp. 991–995. [9]
- He, K., Zhang, X., Ren, S., and Sun, J. (2016), "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. [1,2]
- Hotelling, H. (1933), "Analysis of a Complex of Statistical Variables into Principal Components," *Journal of Educational Psychology*, 24, 417. [1]
- Hu, W., Miyato, T., Tokui, S., Matsumoto, E., and Sugiyama, M. (2017), "Learning Discrete Representations via Information Maximizing Self-Augmented Training," in *Proceedings of the 34th International Conference on Machine Learning*, Volume 70 of Proceedings of Machine Learning Research, eds. D. Precup and Y. W. Teh, pp. 1558–1567, PMLR. [1]
- Hughes, D. P., and Salathé, M. (2015), "An Open Access Repository of Images on Plant Health to Enable the Development of Mobile Disease Diagnostics." arXiv preprint arXiv:1511.08060. [9]
- Ji, X., Henriques, J. F., and Vedaldi, A. (2019), "Invariant Information Clustering for Unsupervised Image Classification and Segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9865–9874. [1,10,12]
- Kather, J. N., Halama, N., and Marx, A. (2018), "100,000 Histological Images of Human Colorectal Cancer and Healthy Tissue," accessed: 2023. [9]
- Kenton, J. D. M.-W. C., and Toutanova, L. K. (2019), "Bert: Pre-Training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of NAACL-HLT*, pp. 4171–4186. [1]
- LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998), "Gradient-based Learning Applied to Document Recognition," *IEEE*, 86, 2278–2324. [1]
- LeCun, Y., Cortes, C., and Burges, C. (2021), "Mnist Handwritten Digit Database," Available at <http://yann.lecun.com/exdb/mnist>. [9]
- Maaten, L., and Hinton, G. (2008), "Visualizing Data Using t-SNE," *Journal of Machine Learning Research*, 9, 2579–2605. [1]
- McInnes, L., Healy, J., and Melville, J. (2018), "Umap: Uniform Manifold Approximation and Projection for Dimension Reduction." arXiv preprint arXiv:1802.03426. [1]
- Ng, A. Y., Jordan, M. I., and Weiss, Y. (2002), "On Spectral Clustering: Analysis and an Algorithm," in *Advances in Neural Information Processing Systems*, pp. 849–856. [7]
- Nguyen, H. D., Khan, M., Kaegi, N., Ho, S., Moore, J., Borys, L., and Lavalva, L. (2021), "Ensemble Learning Using Error Correcting Output Codes: New Classification Error Bounds," in *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, Los Alamitos, CA, USA, pp. 719–723. IEEE Computer Society. [16]
- Park, S., Han, S., Kim, S., Kim, D., Park, S., Hong, S., and Cha, M. (2021), "Improving Unsupervised Image Clustering with Robust Learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12278–12287. [1]
- Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018), "Improving Language Understanding by Generative Pre-Training." [1]
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019), "Language Models are Unsupervised Multitask Learners." OpenAI blog 1.8: 9. [1]
- Rand, W. (1971), "Objective Criteria for the Evaluation of Clustering Methods," *Journal of the American Statistical Association*, 66, 846–850. [13]
- Som, A., Thopalli, K., Ramamurthy, K. N., Venkataraman, V., Shukla, A., and Turaga, P. (2018), "Perturbation Robust Representations of Topological Persistence Diagrams," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 617–635. [1]
- Subramanian, A., Tamayo, P., Mootha, V. K., Mukherjee, S., Ebert, B. L., Gillette, M. A., Paulovich, A., Pomeroy, S. L., Golub, T. R., Lander, E. S., and Mesirov, J. P. (2005), "Gene Set Enrichment Analysis: A Knowledge-based Approach for Interpreting Genome-Wide Expression Profiles," *Proceedings of the National Academy of Sciences of the United States of America*, 102, 15545–15550. [7]
- Tammina, S. (2019), "Transfer Learning Using VGG-16 with Deep Convolutional Neural Network for Classifying Images," *IJSRP*, 9, 143–150. [1,2]
- Tiong, K.-L., Sintupisut, N., Lin, M.-C., Cheng, C.-H., Woolston, A., Lin, C.-H., Ho, M., Lin, Y.-W., Padakanti, S., and Yeang, C.-H. (2022), "An Integrated Analysis of the Cancer Genome Atlas Data Discovers a Hierarchical Association Structure Across Thirty Three Cancer Types," *PLoS Digital Health*, 1, 1–53. [8]
- van den Oord, A., Li, Y., and Vinyals, O. (2018), "Representation Learning with Contrastive Predictive Coding." arXiv preprint arXiv:1807.03748. [1]
- Van Gansbeke, W., Vandenhende, S., Georgoulis, S., Proesmans, M., and Van Gool, L. (2020), "SCAN: Learning to Classify Images Without Labels," in *Computer Vision – ECCV 2020*, eds. A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, pp. 268–285, Cham: Springer International Publishing. [1,10,12]
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017), "Attention Is All You Need," in *Advances in Neural Information Processing Systems* (Vol. 30), eds. I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Curran Associates, Inc. [1]
- Vedaldi, A., Asano, Y., and Rupprecht, C. (2020), "Self-Labeling via Simultaneous Clustering and Representation Learning," in *International Conference on Learning Representations 2020 (ICLR 2020)*, pp. 1–22, Open Review. [1]
- Wilcoxon, F. (1945), "Individual Comparisons by Ranking Methods," *Biometrics Bulletin*, 1, 80–83. [6]
- Xian, Y., Schiele, B., and Akata, Z. (2017), "Zero-Shot Learning – The Good, the Bad and the Ugly," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA, USA, pp. 3077–3086, IEEE Computer Society. [2]
- Xie, J., Girshick, R., and Farhadi, A. (2016), "Unsupervised Deep Embedding for Clustering Analysis," in *Proceedings of The 33rd International Conference on Machine Learning*, Volume 48 of Proceedings of Machine Learning Research, New York, NY, eds. M. F. Balcan and K. Q. Weinberger, pp. 478–487, PMLR. [1]
- Yang, J., Parikh, D., and Batra, D. (2016), "Joint Unsupervised Learning of Deep Representations and Image Clusters," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5147–5156. [1]