Hard Regularization to Prevent Deep Online Clustering Collapse without Data Augmentation

Louis Mahon[1, 3]**, Thomas Lukasiewicz**[2, 3]

[1]School of Informatics, University of Edinburgh, UK
[2]Institute of Logic and Computation, Vienna University of Technology, Austria
[3]Department of Computer Science, University of Oxford, UK
louis.mahon@inf.ed.ac.uk

Abstract

Online deep clustering refers to the joint use of a feature extraction network and a clustering model to assign cluster labels to each new data point or batch as it is processed. While faster and more versatile than offline methods, online clustering can easily reach the collapsed solution where the encoder maps all inputs to the same point and all are put into a single cluster. Successful existing models have employed various techniques to avoid this problem, most of which require data augmentation or which aim to make the average soft assignment across the dataset the same for each cluster. We propose a method that does not require data augmentation, and that, differently from existing methods, regularizes the hard assignments. Using a Bayesian framework, we derive an intuitive optimization objective that can be straightforwardly included in the training of the encoder network. Tested on four image datasets and one human-activity recognition dataset, it consistently avoids collapse more robustly than other methods and leads to more accurate clustering. We also conduct further experiments and analyses justifying our choice to regularize the hard cluster assignments. Code is available at https://github.com/Lou1sM/online_hard_clustering.

1 Introduction

Deep clustering refers to the combination of deep learning and clustering, where the data are first encoded with a deep neural network to a feature space, and then clustering is performed in the feature space. Clustering models can be classified as offline or online. Offline models process the entire dataset, and then assign all cluster labels at once. Online models, in contrast, assign a cluster label to each data point, or each batch, as it is processed. Offline methods can produce more accurate clusterings (e.g., Mahon and Lukasiewicz (2021); Niu, Shan, and Wang (2021) reaching close to supervised performance), as they can leverage information from later data points when assigning cluster labels to earlier data points. However, they are more expensive to train, as they must alternate between encoding the entire dataset, or training the encoder for some number of epochs, and clustering the encoded data. Online models, on the other hand, can jointly encode and cluster, so are less computationally expensive. They are also more versatile, being applicable in real-world settings where new data is constantly

becoming available, as opposed to offline methods, which are limited to fixed pre-defined datasets (Silva et al. 2013).

The disadvantage of online methods is that they are more difficult to train. In particular, they run the risk of producing a degenerate solution where the large majority of data points are concentrated in a small number of clusters. In the extreme case, all points are placed into the same cluster. For example, an intuitive way to formulate a training procedure for online deep clustering is to update both the encoder network and the clustering parameters to make the encoding of each point close to its cluster centroid and far from other cluster centroids. (The clustering parameters are e.g., the centroids in K-means or the first two moments in a Gaussian mixture model.) However, this training objective is trivially minimized by the encoder network mapping all data points to the same point in feature space, where this point is equal to one of the cluster centroids. Techniques for avoiding collapse we refer to as partition support. Several partition support methods have been proposed, but they mostly require data augmentation (DA) (Zbontar et al. 2021; Cai et al. 2021), and those that do not are often ad hoc and lack a rigorous technical foundation (Caron et al. 2020; Deshmukh et al. 2021). Additionally, they take the soft assignments as a measure of partition collapse and propose to regularize the soft assignments to make them more uniform. This paper focuses on deep clustering without DA, which is an advantage because relying on DA limits a method to domains in which sufficient prior knowledge to perform class-preserving augmentations is available. Additionally, we argue that soft assignments are not an accurate measure of collapse, and that we should instead focus on hard assignments.

We propose a DA-free partition support by regularizing hard assignments. Specifically, we consider the problem of how to optimally assign each data point in a batch to the most appropriate cluster. We express this problem probabilistically in a Bayesian framework, where the regularizing element is captured by a prior across clusters. In the case of equally-sized clusters, this can be uniform. We then use this expression to derive a precise optimization objective, which we also show to be equivalent, up to a small error term, to the objective of maximizing the mutual information of the cluster assignments and the data index. This objective itself is too slow to solve exactly, but we devise a greedy approximation algorithm that can be implemented straightforwardly

and results in an intuitive method for fully online clustering, which we term combination assignment. This method outperforms existing online DA-free clustering methods on four popular image clustering datasets. We also analyze the underlying representations and show them to be of high quality. Finally, we analyze the role of hard vs. soft cluster assignments in our partition support method, and in previous methods, and make the case that regularizing hard assignments is a more effective approach. Note that, although existing methods can easily convert soft assignments to hard assignments, this is very different from regularizing the hard assignments, as we propose. While the relation between hard and soft clustering has been studied before (Bora, Gupta, and Kumar 2014; Kearns, Mansour, and Ng 1998), its study in the context of regularizing online deep clustering collapse is new.

Our main contributions are briefly summarized below.

- We articulate a clear Bayesian framework of the problem of hard assignments in online deep clustering models, which avoids the unrealistic assumption of uniformity in each batch made by most existing methods.

- We use this framework to derive an optimization objective and prove that it is approximately equivalent to maximizing the mutual information of the cluster assignments and the data index.

- We show empirically that the resulting method significantly outperforms existing partition support methods in avoiding partition collapse, improving clustering accuracy and leading to more informative representations.

- We conduct further analysis of the performance of different partition support methods, which justifies our choice to focus on hard assignments.

The rest of this paper is organized as follows. Section 2 gives an overview of related work, while Section 3 lays the theoretical foundations of our method, describes our greedy algorithm for optimizing the resulting objective, and proves the equivalence to mutual information maximization. Section 4 reports our empirical results and analysis, and finally Section 5 summarizes our findings.

2 Related Work

A key component of online deep clustering methods is how they avoid the collapsed solution, where (almost) every data point is placed in the same cluster. Methods designed for contrastive learning, and those clustering models that employ it as a part of the training procedure, are more resistant to collapse, because the negative pairs are encouraged to be represented differently. This can mean that they are to be placed in different clusters (Huang and Gong 2021), or just that the encodings should be far apart (Zhang et al. 2021; Cai et al. 2021). Either approach helps resist all points having the same representation. Even without negative pairs, data augmentation is essential for some methods to avoid collapse. Grill et al. (2020) perform representation learning using only positive pairs by training an "online" network to predict the output of a "target" network, which is a slow-moving average of the online network. In (Zbontar et al. 2021), data-augmented pairs are used to reduce redundancy by minimizing off-diagonals in the cross-correlation matrix.

Among online clustering models, several partition support methods have been proposed. The solution in (Kulshreshtha and Guha 2018) is simply to freeze the encoder during clustering, but this requires pretraining it on a separate task offline, so does not work in the fully online setting. A different approach is taken by Gao et al. (2020), who use an online autoencoder-based (AE) clustering model, where the reconstruction loss helps to avoid partition collapse, but this limits the method to simple datasets, because the AE's reconstruction loss requires that different images from the same class have significant pixel overlap. In (Zhan et al. 2020), the loss function is continually reweighted to encourage the assignment to smaller clusters. Additionally, when clusters decrease below a certain threshold, they are deleted, and the largest cluster is split in two using K-means, which also means the method does not work fully online. Data-augmented pairs are used by Cai et al. (2021), with the same method as (Zbontar et al. 2021), treating soft cluster assignments as representations, and using data-augmented pairs. In (Zhong et al. 2020), the sum of squares of the probability (i.e., soft assignment) of each cluster is minimized, marginalized over each training batch. Decomposing the expectation of the square, we see that this also involves minimizing the variance. However, the authors also use contrastive learning and DA to avoid collapse, so they do not fully rely on sum-of-squares minimization.

A similar idea, employed by (Li et al. 2020; Hu et al. 2017; Niu, Shan, and Wang 2021; Niu et al. 2020; Van Gansbeke et al. 2020), is to maximize the entropy of batch-wise marginal soft assignments. For an input distribution X with corresponding soft assigned labels Y, both approximated over a batch, an extra term $-H(Y)$ is added to the loss function. As the entropy of a multinomial is maximized at the uniform distribution, this encourages equally sized clusters. Entropy maximization has the advantage of a solid formal interpretation as part of the maximization of the mutual information between data and cluster assignments: maximizing $H(Y)$ but minimizing $H(Y|X)$ (the latter is minimized explicitly in (Hu et al. 2017) and implicitly via contrastive learning in (Li et al. 2020)), implicitly maximizes

$$I(X;Y) = H(Y) - H(Y|X).$$

However, the marginal entropy term tends to only be partially successful at preventing partition collapse, often the entire dataset is still put into only a small number of clusters. As well as being confirmed in our experiments in Section 4, this empirical weakness of entropy maximization for avoiding partition collapse is reported in (Hu et al. 2017), and better results are found by explicitly constraining the soft cluster assignments using non-linear programming. The work (Li et al. 2020) does not rely entirely on entropy maximization, because they employ contrastive learning, which (as explained above) also helps to avoid partition collapse.

Another approach is to directly impose a constraint on cluster assignments. Based on earlier work (Asano, Rupprecht, and Vedaldi 2019), (Caron et al. 2020; Deshmukh et al. 2021; Kumar et al. 2021) proposed to constrain the soft

cluster assignments to be marginally uniform across each training batch. Though effective in preventing partition collapse, the constraint of exact uniformity across each batch is too strict, as noted in (Kumar et al. 2021). The ground-truth classes will almost certainly violate this constraint.

3 Method

Problem Formulation We want to simultaneously (a) train the encoder and (b) make hard assignments to each batch of points at a time, based on the features extracted by that encoder. If the encoder is $f_{\theta_1} : \mathbb{R}^n \rightarrow \mathbb{R}^m$, our clustering model is $g_{\theta_2} : \mathbb{R}^m \rightarrow \{1, \dots, K\}$, parametrized by θ_2, and our batch size is N, then we seek a function of the form $\Gamma : \mathbb{R}^{N \times m} \rightarrow \{1, \dots, K\}^N$. (The difference between Γ and g_{θ_2} is that the former is used during training to assign an entire batch, while the latter is used at inference time and can assign each point individually.) If we have a method for batchwise assignment, then the training objective can be formulated as minimizing some notion of distance of points from the centroids of their assigned clusters, and the encoder and clustering model can be trained as follows:

$$\underset{\theta_1, \theta_2}{\operatorname{argmin}} \sum_{i=1}^{N} d(f_{\theta_1}(x_i) - \mu_{k_i}), \qquad (1)$$

where x_i is the ith batch elemet, $d(\cdot)$ is a distance function, e.g., Euclidean, $k_i = \Gamma(f_{\theta_1}(x))_i$ is the ith assigned cluster label, and μ_k is the kth cluster centroid.

Finding a suitable assignment function Γ is non-trivial. It should be more likely to assign points to the clusters whose centroids are closer. However, using only this rule, and assigning every vector to its closest centroid, we allow a collapsed solution where (1) is minimized by f_{θ_1} mapping every point to a single centroid: $\forall i, f_{\theta_1}(x_i) = c_k$, for some $k \in \{0, \dots, K-1\}$. The heart of our method is the choice of a suitable Γ, which avoids the collapsed solution. We refer to it as combination assignment, because its prior is the combination of labels under a multinoulli distribution.

Combination Assignment Combination assignment is based on a Bayesian formulation of the online clustering problem. Let $\mathcal{D}$ be the data distribution, $X \sim \mathcal{D}$ be a sampled batch of data, Z be the random variable defined by applying the feature extraction to X (i.e., Z is the output of a deep encoder network), and let Y be the assigned cluster labels. Here, we consider the encoder to be fixed, and we are just interested in the best hard assignment of cluster labels, given the extracted features. That is, we want to determine the values of Y with the maximum probability under the a posteriori distribution $p(Y|Z)$. We assume we have some reasonable estimate of the prior distribution over K clusters: $p : \{0, \dots, K-1\} \rightarrow [0, 1]$ (note, this estimate could change for each batch). Then, the prior probability of a batch containing exactly n_k labels for each cluster $k \in \{1, \dots, K\}$ is

$$\prod_{k=1}^{K} p(k)^{n_k} \frac{N!}{\prod_{k=1}^{K} n_k!}, \qquad (2)$$

where $N = \sum_{k=1}^{K} n_k$ is the batch size. It is often suitable to choose a uniform prior over K cluster labels, $p(i) = 1/K$,

however, we are free to choose any prior we would like. Simply having an estimate of the prior is a weaker assumption than existing methods, which assume equal numbers of each cluster by design. For example, all methods that use k-means as their backbone are implicitly assuming roughly uniform clusters (Satapathy et al. 2015).

We model each cluster as a multivariate normal distribution in feature space, so that the likelihood is given by

$$p(Z = z_1, \dots, z_N | Y = k_1, \dots, k_N) = \qquad (3)$$

$$\prod_{i=1}^{N} \frac{\exp(-\frac{1}{2}(z_i - \mu_{k_i}) \Sigma_{k_i}^{-1} (z_i - \mu_{k_i}))}{\sqrt{(2\pi)^d |\Sigma_{k_i}|}}, \qquad (4)$$

where d is the dimension of the feature space, and μ_k and Σ_k are the centroid and covariance matrix of the kth cluster, respectively. We then maximize the posterior corresponding to the prior in (2) and the likelihood in (4), giving the following optimization problem (more details in the appendix):

$$\underset{Y}{\operatorname{argmin}} \sum_{i=1}^{N} d(z_i, \mu_i, \Sigma_i) - \log p(k_i) + \sum_{k=1}^{K} \log(n_k!), \qquad (5)$$

where $d(z, \mu, \Sigma) = \frac{1}{2}(z - \mu)^T \Sigma^{-1}(z - \mu) + \frac{1}{2} \log (2\pi)^d |\Sigma|$. In matrix notation, the objective is

$$\underset{Q \in \mathcal{B}^{N \times K}}{\operatorname{argmin}} \langle Q, \tilde{Q} \rangle - \mathbb{1}_N^T Q \log P + \log(\mathbb{1}_N^T Q!) \mathbb{1}_K \qquad (6)$$

$$\text{subject to } Q \mathbb{1}_K = \mathbb{1}_N,$$

where $\tilde{Q}_{i,j} = d(z_i, \mu_j, \Sigma_j)$, $\langle \cdot, \cdot \rangle$ denotes the Frobenius inner product, $\mathbb{1}_a$ is an a-dimensional vector of all ones, P is a K-dimensional probability vector specifying the prior, and $\log$ and factorial are applied element-wise to the K-dimensional vector $\mathbb{1}_N^T Q$. Constraining Q to be Boolean enforces hard assignments, constraining $Q\mathbb{1}_K = \mathbb{1}_N$ enforces each latent vector to be assigned to exactly one cluster. The encoder is then trained with gradient descent w.r.t. using only the distance from the chosen cluster. The prior and cluster sizes affect its training indirectly.

Solving the Optimization Problem It is too slow to solve (6) exactly (see appendix for details), but we can motivate an approximation by considering the simpler problem of assigning the last data point in a batch, given that the rest have already been assigned. That is, we maximize the conditional probability of the Nth assignment in a batch, conditioned on the $N-1$ previous assignments. This gives the following (details in the appendix).

$$\underset{k_N=1,\dots,K}{\operatorname{argmin}} \ d(z_N, \mu_{k_N}, \Sigma_{k_N}) - \log p(k_N) + \log(n_{k_N} + 1), \qquad (7)$$

where n_k is the size of cluster k before the nth assignment.

To approximately solve (6), we employ a greedy algorithm that iteratively solves (7) with respect to the most confident assignment possible. That is, at each iteration, select the pair (i, k) (corresponding to (N, k_N) above) for which (7) is smallest, with i ranging over the indices of still-unassigned points, and k ranging over all clusters, and assign the ith point to cluster k.

Intuition To get an intuition on our method, recall that clustering generally should assign points to the closest centroid, but that we should try to resist assigning to a cluster that already has lots of points assigned to it and also take into account the prior. Even if such a cluster has its centroid closest to the Nth point, it may be better to instead assign to a smaller, further away cluster, or one with a higher prior probability. This suggests choosing the cluster assignment so as to minimize a combination of the distance to the centroid, the prior and some increasing function of cluster size, which is precisely what (7) expresses. The first term says to pick a nearby cluster, the second term to pick a cluster with a high prior probability, and the third to penalize clusters that already have many points assigned. It would not be obvious, a priori, what the increasing function of cluster-size should be exactly, but the derivation of (7) shows that $\log(n+1)$ is an appropriate choice. The fact that we take into account the cluster size when assigning points to clusters is an important difference between our method and existing methods, which generally assign to the cluster with the closest centroid (e.g., k-means) or the cluster which assigns the highest probability to the given data point (e.g., GMM clustering). Another important difference is that the centroids are trainable parameters, and can be updated by gradient descent, rather than being set to the empirical mean of the corresponding cluster.

Information-Theoretic Interpretation Under a uniform prior, the greedy algorithm that iteratively solves (7) can be interpreted as iteratively making whatever assignment will maximize the mutual information between the batch index i and the cluster labels. First, we show a close equivalence to maximizing the entropy of cluster labels in each batch.

Let $H^{(k)}$ be the marginal hard entropy of cluster labels after a new assignment to cluster k:

$$H^{(k)} = \frac{x_k+1}{N+1} \log \frac{x_k+1}{N+1} + \sum_{j=1, j\neq k}^{K} \frac{x_j}{N+1} \log \frac{x_j}{N+1},$$

and consider the difference $H^{(k)} - H^{(k')}$ between the entropy after making assignment k vs. after making a different assignment k'. It can be shown (see appendix) that

$$H^{(k)} - H^{(k')} \approx \frac{1}{N+1}\left(\log(x_{k'}+1) - \log(x_k+1)\right).$$

(8)

Thus, if we were to make each assignment so as to maximize $\log(\mathcal{L}_k(x)) + \lambda H^{(k)}$, where $\mathcal{L}(k)$ is the likelihood of the new data point under cluster k, and λ is some hyperparameter, then, subject to the above approximation, for each $k, k' \in \{1, \ldots, K\}$, we would prefer to assign to k iff

$$\log \mathcal{L}(k) + \lambda H^{(k)} > \log \mathcal{L}(k') + \lambda H^{(k')} \iff$$
$$\log \mathcal{L}(k) - \log \mathcal{L}(k') >$$
$$> \frac{\lambda}{N+1}\left(\log(x_{k'}+1) - \log(x_k+1)\right).$$
(9)

Modelling clusters as multivariate normal distributions (as above), and setting $\lambda = N + 1$, (9) becomes equivalent to (7). (See appendix for full proof.)

Algorithm 1: During training, cluster labels are assigned batchwise, with partition support provided by a uniform prior across clusters. During inference, cluster labels are assigned pointwise, without any explicit partition support.

$f_{\theta_1} \leftarrow$ encoder network;
$\theta_2 = \mu_1, \ldots, \mu_K \leftarrow$ centroids for each of the K clusters;
$\sigma \leftarrow$ isotropic variance of all clusters;
function ASSIGNBATCH(Z)
    $counts \leftarrow K$-dimensional array, initially all 0s;
    $isAssigned \leftarrow N$-dimensional Boolean array, initially all False;
    $D \leftarrow N \times K$ matrix, where $D_{ij} = ||Z_i - \mu_j||^2$;
    for r=1,...,N **do**
        $\tilde{D} \leftarrow N \times K$ matrix, where $\tilde{D}_{ij} = D_{ij} + 2\sigma \log(counts[j]+1)$;
        $(a, k) \leftarrow \text{argmin}\{\tilde{D}_{ij} : \neg isAssigned[i]\}$;
        $counts[k] \leftarrow counts[k] + 1$;
        $isAssigned[a] \leftarrow True$;
        $loss \leftarrow loss + D_{a,k}$
    return $loss$
function TRAINONBATCH(X)
    $Z \leftarrow f_{\theta_1}(X)$, encodings for a batch of N data points;
    $loss \leftarrow AssignBatch(Z)$;
    take a train step on $loss$ with respect to θ_1, θ_2
function PREDICTSINGLEDATAPOINT(x)
    $z \leftarrow f_{\theta_1}(x)$ encodings for a batch of N data points;
    $assignment = \text{argmin}_{j=1,\ldots,K} ||z - \mu_j||^2$;
    return assignment

Thus, our method closely approximates a maximization of the entropy of cluster labels. There is some similarity to those methods, discussed in Section 2, that use an additional loss term to encourage greater entropy of soft assignments in each batch, but an important difference here is that we are maximizing the entropy of hard assignments. This means that the entropy of cluster labels given batch index is automatically zero. Therefore, using the decomposition $I(X;Y) = H(X) - H(X|Y)$, the mutual information of the batch index i and the cluster labels equals the entropy of cluster labels, and so our method is a close approximation to maximizing this mutual information.

Training Procedure Our model comprises an encoder network f_{θ_1} and cluster centroids $\theta_2 = \{\mu_1, \ldots, \mu_K\}$. To train on an input batch, we first encode the raw data using f_{θ_1}, then we employ the combination assignment method of Section 3, and use these assignments to minimize (5) with respect to both θ_1 and θ_2. As the second two terms in (5) have no gradient, the updates are made only with respect to the first term, so similarly to (1). At inference time, we do not perform combination assignment. Instead, we simply assign each point to the cluster with the nearest centroid, so the model can assign points individually, and is not restricted to operating batch-wise. The full training and inference methods are described by the functions TrainOnBatch and PredictBatch, respectively, in Algorithm 1.

		CA (ours)	SK	ENT	SS	CKM	no reg
CIFAR 10	Acc	**21.7 (1.50)**	16.7 (0.40)	18.6 (1.52)	11.8 (1.92)	15.2 (0.91)	10.0 -
	NMI	**10.5 (1.07)**	03.8 (0.76)	08.7 (2.94)	01.1 (1.29)	02.8 (0.71)	00.0 -
	ARI	**5.4 (0.59)**	02.7 (0.68)	05.7 (1.97)	00.3 (0.42)	01.4 (0.38)	00.0 -
	KL*	**0.0 (0.01)**	02.5 (0.33)	01.8 (0.21)	00.9 (1.36)	01.1 (0.24)	00.0 -
CIFAR 100	Acc	**6.9 (0.17)**	02.6 (0.24)	02.4 (0.19)	01.2 (0.25)	02.8 (0.34)	01.0 -
	NMI	**14.2 (0.61)**	05.3 (0.46)	06.3 (0.82)	00.6 (1.18)	04.6 (0.89)	00.0 -
	ARI	**1.7 (0.10)**	00.3 (0.04)	00.5 (0.11)	00.0 (0.04)	00.3 (0.12)	00.0 -
	KL*	**0.5 (0.11)**	03.5 (0.80)	01.6 (0.22)	00.2 (0.19)	02.8 (0.49)	00.0 -
FashionMNIST	Acc	**59.3 (4.16)**	25.1 (3.13)	25.5 (6.16)	10.0 (0.04)	22.0 (3.83)	10.0 -
	NMI	**55.3 (2.95)**	17.7 (2.33)	20.9 (11.32)	00.0 (0.04)	15.8 (5.28)	00.0 -
	ARI	**42.5 (4.12)**	09.2 (1.42)	10.6 (6.01)	00.0 -	07.3 (2.84)	00.0 -
	KL*	**0.0 (0.02)**	02.6 (0.35)	01.9 (0.24)	00.0 (0.01)	01.3 (0.31)	00.0 -
STL	Acc	**24.2 (2.56)**	18.5 (1.08)	10.9 (1.97)	10.1 (0.22)	14.1 (0.78)	10.0 -
	NMI	**13.7 (1.32)**	06.7 (1.14)	01.3 (2.86)	00.0 (0.09)	02.5 (0.58)	00.0 -
	ARI	**7.3 (1.11)**	03.3 (0.88)	00.2 (0.36)	00.0 -	01.1 (0.39)	00.0 -
	KL*	**0.1 (0.10)**	00.7 (0.41)	00.1 (0.18)	00.1 (0.18)	01.2 (0.21)	00.0 -
RealDisp	Acc	**57.0 (3.02)**	23.5 (3.09)	18.8 (1.98)	10.7 (1.88)	17.2 (1.33)	10.0 -
	NMI	**55.0 (17.11)**	31.0 (2.93)	29.7 (4.26)	00.5 (0.50)	25.8 (1.75)	00.0 -
	ARI	**48.1 (4.49)**	11.6 (3.02)	06.6 (1.99)	00.2 (0.26)	06.4 (1.91)	00.0 -
	KL*	**0.1 (0.06)**	00.7 (0.13)	01.7 (0.61)	02.0 (0.91)	01.6 (0.13)	02.3 -

Table 1: Effect, on cluster size and clustering performance, of our method compared to two existing partition-support methods. All figures are the mean of 5 runs, with std dev in parentheses. Best results in bold, std dev of zero after rounding, is written '-'.

4 Experimental Evaluation

Datasets and Metrics We report results on CIFAR 10 (C10), CIFAR 100 (C100), FashionMNIST (FMNIST), and STL, with image sizes 32, 32, 28, and 96, respectively, and the human activity recognition (HAR) dataset RealDisp, of 17 subjects performing 33 different activities wearing accelerometers. We use the standard clustering metrics of accuracy (ACC), normalized mutual information (NMI), and adjusted Rand index (ARI), defined as, e.g., in (Sheng and Huber 2020). We also report the KL-divergence from the ground truth of the model's empirical distribution over clusters, denoted KL^*. For a collapsed model, which assigns most points to a few clusters, this will be high.

Clustering Accuracy and Degree of Collapse

Table 1 compares our method with three existing methods: sum of squares minimization, denoted "SS" (Zhong et al. 2020), the Sinkhorn-Knopp algorithm for optimal transport, denoted "SK" (Caron et al. 2020; Kumar et al. 2021), and marginal entropy maximization (Li et al. 2020), denoted "Ent". Each is described in Section 2. To make further explicit the phenomenon of partition collapse, we also include a model without any partition support. Our method significantly outperforms others on all datasets and metrics.

Observe that the unregularized model exhibits total collapse in all experiments, placing all points in the same cluster and consequently achieving a cluster performance no better than random guessing. The performance of SS is not much better. By making a slight change to the author's original method, we could actually significantly improve its results (see appendix), but it was still unreliable and less accurate than our method. The other two existing partition support methods do a reasonable job of avoiding partition collapse. However, entropy maximization occasionally also

reaches the state with all points in the same cluster (this is consistent with previous literature, e.g., (Hu et al. 2017)). The Sinkhorn-Knopp method is more reliable, but by far the most uniform cluster sizes are produced by our method. Note that we do not employ our assignment algorithm at inference time, and instead assign each point to the cluster with the nearest centroid. This shows that our cluster centroids are well-distributed around the data manifold, each capturing a sizeable subset of the data even when the explicit support is removed. Together, these figures show that (a) partition support is necessary to learn anything meaningful, (b) our method of combination assignment is better at avoiding partition collapse than previous methods, and (c) our model has, consequently, better clustering performance.

To better isolate the effect of our proposed partition support method, we do not perform hyperparameter tuning, and use a relatively simple architecture for all datasets. This is the same for all methods being compared. The network consists of two convolutional layers with filter sizes 6 and 16, and ReLU activations, followed by a fully connected layer to a latent space of dimension 128. Training uses Adam, with learning rate 1e-3, $\beta_1 = 0.9$, $\beta_2 = 0.99$, and batch size 256. We follow previous works in setting K, the number of clusters, to the number of ground-truth classes. We set $\Sigma = \sigma I$ with $\sigma = 1e2$. We observe very similar results for a wide range of values for Σ, including a setting where we continually estimate from the empirical distributions.

Although some existing clustering methods produce higher accuracy than those of Table 1, this is not a valid comparison, because (as discussed above) these higher-scoring methods (a) use data-augmentation and (b) even more significantly, are offline clustering methods. It is considerably easier to score highly in the offline setting. As our method operates in the online setting, it is not comparable.

		imb 1	imb 2	imb 3
CIFAR 10	acc	22.2 (1.22)	22.8 (1.46)	24.6 (1.26)
	nmi	10.5 (1.14)	11.3 (0.73)	10.5 (0.50)
	ari	5.9 (0.92)	6.3 (0.55)	6.2 (3.22)
	KL*	0.1 (0.07)	0.1 (0.04)	0.2 (0.11)
CIFAR 100	acc	6.6 (0.31)	6.7 (0.62)	9.0 (0.33)
	nmi	13.3 (0.62)	13.7 (0.99)	15.4 (0.30)
	ari	1.6 (0.19)	1.9 (0.36)	2.7 (0.27)
	KL*	0.8 (0.10)	0.7 (0.10)	0.6 (0.09)
FMNIST	acc	54.6 (6.34)	52.1 (3.39)	53.8 (2.95)
	nmi	52.9 (4.41)	50.6 (1.50)	50.0 (2.35)
	ari	37.7 (5.56)	36.7 (1.84)	38.5 (3.08)
	KL*	0.2 (0.23)	0.1 (0.05)	0.2 (0.17)
STL	acc	23.2 (2.07)	24.6 (1.41)	25.5 (3.13)
	nmi	13.3 (1.69)	14.5 (1.15)	12.6 (2.44)
	ari	6.8 (1.13)	7.8 (0.81)	6.8 (1.90)
	KL*	0.3 (0.20)	0.2 (0.19)	0.1 (0.03)
RealDisp	acc	57.5 (1.45)	46.7 (6.75)	39.2 (6.20)
	nmi	75.1 (1.01)	67.9 (5.65)	62.9 (5.77)
	ari	49.4 (0.90)	38.0 (6.66)	30.0 (6.07)
	KL*	0.1 (0.07)	0.0 (0.01)	0.3 (0.03)

Table 2: Performance on imbalanced data, imb. 1 removes 10% of one class and 20% of another, imb. 2 removes $0\%, 5\%, 10\%, \dots$ of each of the 10 C10 classes, imb.3 is the same but in increments of 10% (scaled proportionally to class size for other datasets). Note, a random guess will score higher with increasing imbalance.

Imbalanced Data

As described in Section 3, our method can use any prior distribution over clusters, not just a uniform distribution (i.e., equal cluster sizes) as most existing methods assume. Here, we conduct an empirical investigation of performance on imbalanced data. We assume that, in this case, we have an idea of the relative frequency of each class/cluster and that this determines our prior. To explore varying levels of class-imbalance, we begin with the balanced datasets from Section 4, and progressively remove parts of some classes, giving three increasing levels of imbalance. As shown in Table 2, our method does not degrade in performance over these settings, showing robustness to varying class distributions.
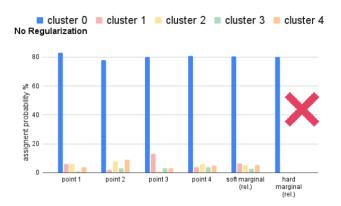
Hard vs. Soft Assignment Regularization

A key element of our method is the regularization of hard assignments, whereas previous methods regularize soft cluster assignments. Ours is a fundamentally different form of regularization, and one which we argue is better able to prevent collapse. For example, assume there are only three clusters and a batch size of four, and consider the following two matrices of assignment probabilities
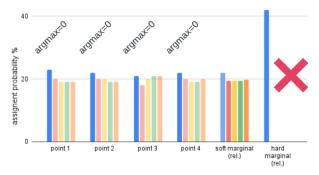
$$D_1 = \begin{bmatrix} .98 & .01 & .01 \\ .98 & .01 & .01 \\ .49 & .50 & .01 \\ .49 & .01 & .50 \end{bmatrix} D_2 = \begin{bmatrix} .34 & .33 & .33 \\ .34 & .33 & .33 \\ .34 & .33 & .33 \\ .34 & .33 & .33 \end{bmatrix}.$$

The hard and soft entropy (i.e., the entropy of marginal hard and soft assignments) are 1.5 and 1.1 for D_1, and 0 and 1.58 for D_2. That means D_2 has higher soft entropy but a much

lower hard entropy. Indeed, despite having nearly maximum soft entropy, D_2 is collapsed with zero hard entropy. A similar scenario is shown for an idealized batch in Figure 1, again revealing that near-uniform soft assignments is a different objective to, and does not guarantee, avoiding collapse.
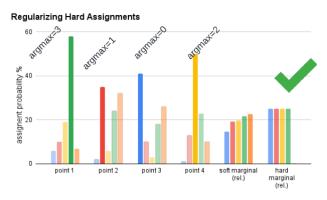

Figure 1: Top: with no regularization, the model collapses. Middle: soft regularization encourages uniform soft assignments but the argmax is constant. Bottom: hard regularization causes the argmax to change, and avoids collapse.

To investigate whether such differences between hard vs. soft regularization manifest in practice, we empirically compare the KL* for the hard and soft assignments. The results are shown in Table 4. The most striking difference between hard and soft entropy is in SS. There, the soft KL* is often close to the maximum value (equal to the logarithm of the number of clusters), but the hard entropy is consistently

		CA (ours)	SK	ENT	SS	CKM
C10	linear	*35.5 (0.73)*	30.0 (0.66)	35.4 (1.53)	26.0 (2.39)	**40.2 (0.30)**
	KNN	30.3 (0.87)	27.1 (1.02)	*33.3 (2.29)*	22.8 (1.74)	**38.3 (0.61)**
C100	linear	**17.1 (0.08)**	8.9 (1.74)	6.1 (0.77)	7.9 (1.03)	*16.9 (0.48)*
	KNN	*12.8 (0.39)*	8.4 (1.88)	5.3 (0.86)	7.7 (1.04)	**13.7 (0.73)**
FM	linear	79.3 (1.83)	75.8 (1.39)	*80.1 (1.36)*	62.8 (2.55)	**81.9 (0.53)**
	KNN	77.0 (1.76)	*80.5 (0.58)*	80.4 (1.09)	78.5 (1.53)	**81.3 (0.80)**
STL	linear	*36.0 (2.03)*	32.4 (0.76)	33.6 (0.80)	33.0 (1.57)	**40.4 (1.49)**
	KNN	**36.8 (1.85)**	27.3 (3.33)	27.5 (0.70)	29.0 (1.49)	*35.6 (1.74)*
RD	linear	**95.9 (0.58)**	79.7 (5.21)	38.5 (6.93)	31.1 (2.44)	*83.4 (1.77)*
	KNN	**98.8 (0.23)**	*97.1 (1.87)*	93.4 (1.03)	40.8 (28.58)	88.7 (2.24)

Table 3: Quality of the learned representations, as assessed by linear and KNN probes. Our method performs comparably to CKM on the vision datasets, and better on RealDISP, fitting with the case made in Mahon and Lukasiewicz (2023) against autoencoders in HAR clustering.

		CA	SK	Ent	SS	CKM
CIFAR 10	hard KL*	**0.04**	0.56	1.05	1.66	1.13
	soft KL*	0.04	**0.00**	0.85	0.51	0.03
CIFAR 100	hard KL*	**0.83**	2.17	3.48	4.47	2.78
	soft KL*	0.13	**0.01**	1.42	0.09	0.02
FMNIST	hard KL*	**0.04**	0.47	0.99	2.30	1.32
	soft KL*	0.04	**0.00**	0.69	0.17	0.02
STL	hard KL*	**0.09**	2.31	2.25	2.25	1.17
	soft KL*	0.08	**0.00**	0.92	0.18	0.02
RealDisp	hard KL*	**0.05**	0.66	1.69	2.05	1.65
	soft KL*	0.00	**0.00**	0.08	0.00	0.02

Table 4: Comparison of KL* for hard and soft assignments. Previous methods regularize soft assignments, but this does not transfer well to hard assignments. Ours (CA) is the only method that closely approximates the true distribution for hard assignments. Lower is better, best results in bold.

close to zero. This shows that this form of regularization produces batch assignment probabilities similar to matrix D_2 above, where the probabilities for each data point are squeezed close to one another, without much change in the order of highest to lowest, in particular the argmax.

A similar discrepancy is found in SK, which produces near-perfect uniformity in the soft assignments, and with maximum entropy (up to rounding) on each dataset. This is because it is a (close approximation to a) hard constraint problem. However, SK's hard assignments still show significant variability, and markedly lower entropy than the soft assignments, suggesting it also produces batch assignment probabilities similar to D_2 above. Our method, on the other hand, explicitly forces the argmax to be more evenly distributed during training and, as Table 4 shows, this transfers to the testing setting as well. (Recall that, during testing, we simply assign each point to the cluster with the nearest centroid.) Our hard entropy is only slightly lower than our soft entropy, and is consistently higher than that of the other three methods. This supports our argument that the mean soft assignments do not contain sufficient information to determine if the clustering model is learning a meaningful partition, and regularizing this quantity is not an optimal way to prevent collapse. The distribution of the argmax is also impor-

tant and is not captured by mean soft assignment.

Our analysis here of hard vs. soft cluster assignments does not contradict Caron et al. (2020). They report better results using soft assignments as training labels, while we show the superiority of *regularizing*, i.e., encouraging equal numbers of, hard assignments. The two are different contexts of hard and soft labels. We also explored training SK (the method used by Caron et al.) using soft assignments as targets, but the results were slightly worse than using hard targets.

Quality of Learned Representations

Although ours is primarily a clustering method, we also examine the quality of the learned representations, using linear-probe protocol (Zhang, Isola, and Efros 2017), which trains a linear model and KNN model ($k = 10$) to predict the class labels from encoded feature vectors. The results are shown in Table 3. It is expected that CKM performs well in this setting, because it uses an autoencoder, which is a widely used method for representation learning. Still, however, our method performs on par with CKM, and outperforms the other methods. Note also CKM's poor performance on RealDisp, suggesting that the effectiveness of a decoder may be specific to simple vision datasets.

5 Conclusion

This paper proposed a data-augmentation-free method to prevent collapse in online deep clustering. We frame probabilistically the problem of deciding which clusters to assign a batch of data points to, given the cluster centroids and features of the data points, and hence derive an intuitve optimization objective for making hard cluster assignments. We then described an algorithm to approximately solve this optimization problem and demonstrated empirically on four datasets that this method outperforms existing methods, both in preventing collapse and in the resulting clustering performance. Finally, we analyzed how the cluster distribution is affected by our partition support method and previous comparable methods. The analysis suggests that regularizing the soft assignments, as is done by existing works, is not sufficient to prevent collapse, and that a better approach is to regularize the hard assignments, as is done by our method.

Acknowledgments

References

Asano, Y. M.; Rupprecht, C.; and Vedaldi, A. 2019. Self-labelling via simultaneous clustering and representation learning. *arXiv preprint ArXiv:1911.05371*.

Bora, D. J.; Gupta, D.; and Kumar, A. 2014. A comparative study between fuzzy clustering algorithm and hard clustering algorithm. *arXiv preprint ArXiv:1404.6059*.

Cai, Y.; Zhang, Z.; Liu, Y.; Ghamisi, P.; Li, K.; Liu, X.; and Cai, Z. 2021. Large-scale hyperspectral image clustering using contrastive learning. *arXiv preprint ArXiv:2111.07945*.

Caron, M.; Misra, I.; Mairal, J.; Goyal, P.; Bojanowski, P.; and Joulin, A. 2020. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint ArXiv:2006.09882*.

Deshmukh, A. A.; Regatti, J. R.; Manavoglu, E.; and Dogan, U. 2021. Representation learning for clustering via building consensus. *arXiv preprint arXiv:2105.01289*.

Gao, B.; Yang, Y.; Gouk, H.; and Hospedales, T. M. 2020. Deep clustering with concrete k-means. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4252–4256. IEEE.

Grill, J.-B.; Strub, F.; Altché, F.; Tallec, C.; Richemond, P.; Buchatskaya, E.; Doersch, C.; Avila Pires, B.; Guo, Z.; Gheshlaghi Azar, M.; et al. 2020. Bootstrap your own latent: A new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33: 21271–21284.

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 International Conference on Machine Learning*, 1558–1567. PMLR.

Huang, J.; and Gong, S. 2021. Deep clustering by semantic contrastive learning. *arXiv preprint ArXiv:2103.02662*.

Kearns, M.; Mansour, Y.; and Ng, A. Y. 1998. An information-theoretic analysis of hard and soft assignment methods for clustering. In *Learning in Graphical Models*, 495–520. Springer.

Kulshreshtha, P.; and Guha, T. 2018. An online algorithm for constrained face clustering in videos. In *Proceedings of the 2018 25th IEEE International Conference on Image Processing (ICIP)*, 2670–2674. IEEE.

Kumar, S.; Haresh, S.; Ahmed, A.; Konin, A.; Zia, M. Z.; and Tran, Q.-H. 2021. Unsupervised activity segmentation by joint representation learning and online clustering. *arXiv preprint ArXiv:2105.13353*.

Li, J.; Zhou, P.; Xiong, C.; and Hoi, S. C. 2020. Prototypical contrastive learning of unsupervised representations. *arXiv preprint ArXiv:2005.04966*.

Mahon, L.; and Lukasiewicz, T. 2021. Selective pseudo-label clustering. In *Proceedings of the German Conference on Artificial Intelligence (Künstliche Intelligenz)*, 158–178. Springer.

Mahon, L.; and Lukasiewicz, T. 2023. Efficient deep clustering of human activities and how to improve evaluation. In *Asian Conference on Machine Learning*, 722–737. PMLR.

Niu, C.; Shan, H.; and Wang, G. 2021. SPICE: Semantic pseudo-labeling for image clustering. *arXiv preprint ArXiv:2103.09382*.

Niu, C.; Zhang, J.; Wang, G.; and Liang, J. 2020. GATCluster: Self-supervised Gaussian-attention network for image clustering. In *Proceedings of the European Conference on Computer Vision*, 735–751. Springer.

Satapathy, S. C.; Govardhan, A.; Raju, K. S.; and Mandal, J. 2015. *Emerging ICT for Bridging the Future-Proceedings of the 49th Annual Convention of the Computer Society of India CSI Volume 2*. Springer.

Sheng, T.; and Huber, M. 2020. Unsupervised embedding learning for human activity recognition using wearable sensor data. In *Proceedings of the The Thirty-Third International Flairs Conference*.

Silva, J. A.; Faria, E. R.; Barros, R. C.; Hruschka, E. R.; Carvalho, A. C. d.; and Gama, J. 2013. Data stream clustering: A survey. *ACM Computing Surveys (CSUR)*, 46(1): 1–31.

Van Gansbeke, W.; Vandenhende, S.; Georgoulis, S.; Proesmans, M.; and Van Gool, L. 2020. Scan: learning to classify images without labels. In *Proceedings of the European Conference on Computer Vision*, 268–285. Springer.

Zbontar, J.; Jing, L.; Misra, I.; LeCun, Y.; and Deny, S. 2021. Barlow twins: self-supervised learning via redundancy reduction. In *Proceedings of the International Conference on Machine Learning*, 12310–12320. PMLR.

Zhan, X.; Xie, J.; Liu, Z.; Ong, Y.-S.; and Loy, C. C. 2020. Online deep clustering for unsupervised representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6688–6697.

Zhang, D.; Nan, F.; Wei, X.; Li, S.; Zhu, H.; McKeown, K.; Nallapati, R.; Arnold, A.; and Xiang, B. 2021. Supporting Clustering with Contrastive Learning. *arXiv preprint ArXiv:2103.12953*.

Zhang, R.; Isola, P.; and Efros, A. A. 2017. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1058–1067.

Zhong, H.; Chen, C.; Jin, Z.; and Hua, X.-S. 2020. Deep robust clustering by contrastive learning. *arXiv preprint ArXiv:2008.03030*.