Evaluation Criteria for Deep Clustering Algorithms

Jayanth Regatti [*1] **Aniket Deshmukh** [*2] **Eren Manavoglu** [2] **Urun Dogan** [2]

Abstract

In this paper, we discuss a new method for deep clustering and unsupervised representation learning for images and focus our attention on the evaluation of the clustering. In particular, we focus on the challenges in real world clustering tasks, such as in the case of distribution shift and note that existing evaluation criteria may not be sufficient. Based on this, we propose various criteria for evaluating representation learning for clustering images under more realistic situations such as cross model performance (distribution shift), robustness to the choice of hyperparameters,etc. We demonstrate the utility of these criteria by evaluating clustering on five popular computer vision datasets using a new method ConCURL (a representation learning and clustering algorithm). The project website is here and the code for ConCURL is available here.

1. Introduction

Clustering is a ubiquitous task and it has been actively used in many different scientific and practical pursuits (Frey & Dueck, 2007; Masulli & Schenone, 1999; Jain et al., 1999; Xu & Wunsch, 2005). Traditional clustering algorithms do not learn the representations and are hence limited to data for which we have a good representation available. Self-supervised learning addresses the issue of learning representations without labeled data. Self-supervised learning is a sub-field of unsupervised learning in which the main goal is to learn general purpose representations by exploiting user-defined tasks (pretext tasks) (Wu et al., 2018; Zhuang et al., 2019; He et al., 2020; Zhuang et al., 2019; Chen et al., 2020; Grill et al., 2020; Caron et al., 2020). The representations learnt using these algorithms can be used with an out of the box clustering algorithm (e.g:k-means) for clustering. Alternatively, deep clustering algorithms involves simultaneously learning cluster assignments and features using deep neural networks (Caron et al., 2018; Xie et al., 2016; Caron et al., 2018; Shah & Koltun, 2018; Ji et al., 2019; Niu et al., 2020b; Wu et al., 2019; Huang et al., 2020b; Tao et al.).

Evaluating clustering algorithms is a notoriously hard problem. As the reference text (Jain & Dubes, 1988) puts it: *The validation of clustering structures is the most difficult and frustrating part of cluster analysis. Without a strong effort in this direction, cluster analysis will remain a black art accessible only to those true believers who have experience and great courage.*

In this work, we focus on representation learning for the unsupervised learning task of clustering images. In the representation learning for clustering literature, e.g. (Li et al., 2021; Huang et al., 2020a; Tao et al.), for evaluating performance of different algorithms the following methodology has been used: a set of models with some hyper-parameters are trained and these models are sorted by the observed clustering performance. Finally the best model's results are reported. We refer to this as the **max-performance** methodology in the rest of the paper.

Although max-performance procedure gives some insights about the performance of the method under consideration, it does not provide the full picture. In practice it is desirable that the learned models can be utilized for different dataset other than the training dataset. Max-performance method may not be suitable for this purpose. To address this, we have designed two additional experiments that focus on performance of cross-model features (under distribution shift). Finally, we present a detailed ablation study to assess the impact of hyperparameters and data augmentations in order to have a more complete picture. We believe that, expanding the evaluation criteria for clustering in the representation learning paradigm will further the state of the art in unsupervised learning algorithms.

In order to demonstrate the utility of the proposed evaluation criteria, we assess the quality of clustering using a new method called Consensus clustering using unsupervised representation learning (ConCURL) (see Appendix A) for the details. We focus on evaluation in the paper by using five challenging image data sets for clustering and report the results with max-performance strategy, cross-model (out of distribution) performance, variation of performance over choice of hyperparameters and data augmentations.

[*]Equal contribution [1]The Ohio State University [2]Microsoft. Correspondence to: Jayanth Regatti <regatti.1@osu.edu>, Aniket Deshmukh <aniketde@umich.edu>.

ICML 2021 Workshop: Self-Supervised Learning for Reasoning and Perception.

Table 1. Clustering with max-performance

Method	STL-10			ImageNet-10			ImageNet-Dogs			CIFAR-10			CIFAR100-20		
	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
GATCluster (Niu et al., 2020a)	0.583	0.446	0.363	0.762	0.609	0.572	0.333	0.322	0.200	0.610	0.475	0.402	0.281	0.215	0.116
PICA (Huang et al., 2020a)	0.713	0.611	0.531	0.870	0.802	0.761	0.352	0.352	0.201	0.696	0.591	0.512	0.337	0.310	0.171
CC (Li et al., 2021)	**0.850**	**0.746**	**0.726**	0.893	0.859	0.822	0.429	0.445	0.274	0.790	0.705	0.637	0.429	0.431	0.266
ID(Tao et al.)	0.726	0.64	0.526	0.937	0.867	0.865	0.476	0.47	0.335	0.776	0.682	0.616	0.409	0.392	0.243
IDFD(Tao et al.)	0.756	0.643	0.575	0.954	0.898	0.901	0.591	0.546	0.413	0.815	0.711	0.663	0.425	0.426	0.264
ConCURL	0.749	0.636	0.566	**0.958**	**0.907**	**0.909**	**0.695**	**0.63**	**0.531**	**0.846**	**0.762**	**0.715**	**0.479**	**0.468**	**0.3034**

1.1. Method

In this section, we provide a brief summary of the algorithm used for representation learning and clustering of images. The algorithm ConCURL is discussed in more detail in Appendix A.

ConCURL performs representation learning and clustering of images by optimizing a combination of two losses $L_{\mathcal{Z}}$ and L_b. L_b is referred to as a backbone loss, where an unsupervised representation learning loss is used to enforce closeness in representation space for different augmentations of the same data point. For the backbone loss, we used the instance discrimination task from (Wu et al., 2018). $L_{\mathcal{Z}}$ is referred to as clustering loss, where a soft clustering objective is used to enforce consistency in the cluster assignments of randomly transformed versions of the representations. For the soft clustering objective, we used the framework from (Caron et al., 2020). We used gaussian random projections and diagonal transformations for randomly transforming the representations.

2. Empirical Evaluation

As stated earlier, evaluating clustering algorithms is a difficult task. To start with, in real world situations, it is even not practical to assume that the number of clusters is known. However, most recent works (including ours) keep this assumption since it is already difficult to learn clustering and representation learning simultaneously. Even with this, performing deep clustering involves a variety of hyperparameters, tricks (such as image augmentations, etc) that are not required when using standard clustering algorithms on user defined features. Therefore the standard evaluation criteria of measuring clustering accuracy, NMI, ARI for the best set of hyper parameters is not a very useful method to evaluate deep clustering algorithms that has several moving parts. There is a strong need to use more relevant criteria that measure the clustering quality of the representations with out of distribution samples, the robustness of the algorithms to the tricks, choice of hyperparameters while evaluating clustering algorithms. Towards this end, we propose to use following criteria in addition to the **max-performance** criteria to evaluate the algorithms. 1) Distribution of accuracies for the set of hyperparameters, 2) Out of distribution data and 3) Impact of Data Augmentations.

It is important to note that none of the criteria alone (including max-performance) is self sufficient in evaluating the deep clustering algorithms completely. In this section, we first use the **max-performance** criterion to evaluate ConCURL with other similar state of the art deep clustering algorithms on five popular computer vision datasets. Then, we show results for evaluation of ConCURL on the proposed evaluation criteria.

Table 2. Dataset Summary

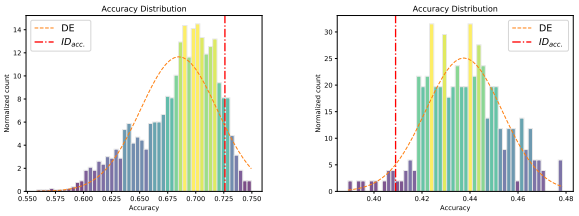
Dataset	Classes	Split	Samples	Resolution
ImageNet-10	10	train	13000	160 × 160
ImageNet-Dogs	15	train	19500	160 × 160
STL-10	10	train+test	13000	96× 96
CIFAR-10	10	train	50000	32× 32
CIFAR100-20	20	train	50000	32× 32

2.1. Image Clustering max-performance

We evaluated ConCURL on some popular image data sets namely ImageNet-10 , ImageNet-Dogs , STL-10 , CIFAR-10 , CIFAR100-20 . For CIFAR100-20 , we used the 20 meta classes as the class labels while evaluating clustering. For STL-10 similar to earlier approaches PICA (Huang et al., 2020b) and GATCluster (Niu et al., 2020b), we used both train and test splits for training and evaluation. Note that PICA also uses unlabelled data split of 100k points in STL-10 which we don't use. ImageNet-10 and ImageNet-Dogs are subsets of ImageNet and we only used train split for these two datasets (Deng et al., 2009). We used the same classes as (Chang et al., 2017) for evaluating on ImageNet-10 and ImageNet-Dogs datasets. The dataset summary is given in Table 2. We evaluate for the cluster accuracy, NMI, ARI of the computed cluster assignments (see Appendix for details).

In our comparison, we considered some state-of-the-art methods that are developed for image clustering problems and are targeting end-to-end training scenarios from random initialization. We should note that we do not consider baselines that use prior information e.g.nearest neighbors derived by using pre-trained models. The implementation details of ConCURL are provided in Appendix and the results are presented in Table 1.

We observe that ConCURL outperforms the baseline algo-



(a) (b)

Figure 1. The dotted red lines show the accuracy of baseline,i.e. ID(Tao et al.) , on the corresponding data set and DE is the density estimate of the empirical distribution. (a)Empirical accuracy distribution for STL-10 (b)Empirical accuracy distribution for CIFAR100-20 .

rithms considered on all the three metrics in all the data sets except STL-10 . ConCURL improves the state-of-the art clustering accuracy by app. 17.5% on ImageNet-Dogs , by 12.7% on CIFAR100-20 and by 3.8% on CIFAR-10 . Although ConCURL improves over ID (Tao et al.), please note that ID is backbone used in this paper, and is slightly worse than IDFD as shown in (Tao et al.).

2.2. Distribution of accuracies for set hyperparameters

Table 3. Hyperparamers and the range values used for the experiments

	min	max	step size
τ = temperature parameter	0.3	1	0.1
l = learning rate	0.015	0.09	0.015
η = natural log of number of transformations	0	7	1
d = dimension of projection space	80	168	8

Max-performance procedure provides some insights to the performance of the algorithms at hand albeit it does not uncover the whole picture as it does not consider the robustness of the performance differences. In Table 8, we provide the hyperparameters that yield us the max-performance results, however, it does not indicate how difficult it is to achieve the max performance results. In other words, just finding a hyper-parameter set which provides better performance than the baseline is the core idea behind max-performance procedure. We ask the following question: given a hyper-parameter grid, how likely is it to get a better accuracy than the baseline? In Figure 1, we report the empirical accuracy distribution for STL-10 and CIFAR100-20 for all hyper-parameters given in Table 10. The red dotted lines show the corresponding baseline accuracy for each dataset. For STL-10 only app. 12.5% of the hyper-parameter sets provide better results than the baseline. On the other hand, for CIFAR100-20 , app. 90% of the hyper-parameter sets provide better results than the baseline. In other words, it doesn't require significant amount of computational power to find a better model than state-of-the art models for CIFAR100-20 however the situation is opposite for STL-10 . The results given in Figure 1 suggests that for comparing models multiple metrics need to be considered in addition to the max-performance procedure.

2.3. Cross Model accuracy (out of distribution)

Here we calculate the clustering perfomance when the model is trained on one dataset but evaluated on a different dataset which may have different number of clusters. In particular, we use the trained model (on one dataset) to extract the features for the other dataset to use towards clustering.

2.3.1. ImageNet-10 vs. ImageNet-Dogs and CIFAR-10 vs. CIFAR100-20

First row in table 4 gives performance of model trained on ImageNet-10 and evaluated on both ImageNet-10 and ImageNet-Dogs . Similarly second row gives performance of model trained on ImageNet-Dogs . We found that ImageNet-10 performance was decreased to 37% when the model trained on ImageNet-Dogs was used instead of ImageNet-10 . Similarly ImageNet-Dogs performance was decreased to 25% when the model trained on ImageNet-10 was used instead of ImageNet-Dogs . Table 4 also gives the same perfomance metric for CIFAR-10 and CIFAR100-20 . It is clear that these performance drops are significant and the generalization performance of learned embedding needs to be assessed by taking into account out of distribution data sets. However, since there are only 2 sets of different data sets it is hard to reach a definitive conclusion. Hence, in the following section we propose a new evaluation methodology which will shed more light on out of distribution performance of learned embeddings.

Table 4. ImageNet-10 vs ImageNet-Dogs and CIFAR-10 vs CIFAR100-20 cross model performance

Model Trained on	ImageNet-10			ImageNet-Dogs		
	ACC	NMI	ARI	ACC	NMI	ARI
ImageNet-10	0.958	0.908	0.910	0.177	0.127	0.068
ImageNet-Dogs	0.356	0.298	0.184	0.695	0.630	0.532

Model Trained on	CIFAR-10			CIFAR100-20		
	ACC	NMI	ARI	ACC	NMI	ARI
CIFAR-10	0.846	0.762	0.715	0.178	0.158	0.061
CIFAR100-20	0.464	0.359	0.250	0.480	0.468	0.304

2.3.2. ImageNet-10-random and ImageNet-15-random accuracies

Here we compare the baseline model trained with ID (Wu et al., 2018) with ConCURL. We randomly sample 10 and 15 classes from 1000 class ImageNet and evaluate the clustering accuracy on train split of the data using model trained from original ImageNet-10 and ImageNet-Dogs . We repeat the process 100 times for both cases: 10 classes and 15 classes datasets; and call them as random-10 and random-15

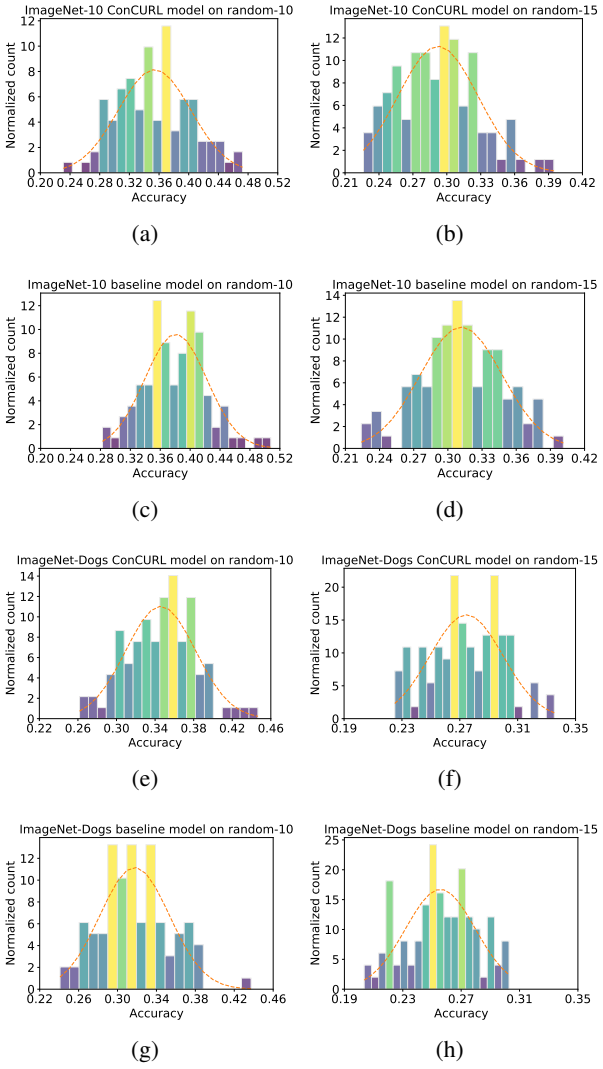


Figure 2. Histogram of clustering accuracies for models trained on ImageNet-10 : (a) ConCURLmodel evaluated on random-10, (b) ConCURLmodel evaluated on random-15, (c) Baseline (ID) model evaluated on random-10, (d) Baseline (ID) model evaluated on random-15. Models trained on ImageNet-Dogs : (e) ConCURLmodel evaluated on random-10, (f) ConCURLmodel evaluated on random-15, (g) Baseline (ID) model evaluated on random-10, (h) Baseline (ID) model evaluated on random-15

datasets respectively. Note that we don't retrain the model on the randomly sampled dataset but we only evaluate on it. We show the histogram of the accuracies we observed in these 100 random datasets. In Figure 2, we compare the accuracy on both random-10 and random-15 for ConCURL model and baseline ID model trained on ImageNet-10 . Along with the histograms we show the Gaussian distribution (in red dotted line) with the first and second moments equal to the average and standard deviations of of all accuracies respectively. Similarly in Figure 2, we show accuracies based on models trained on ImageNet-Dogs . For models trained on ImageNet-10 , baseline ID model performs slightly better compared to the proposed ConCURL model. The trend is reversed for the evaluation based on models trained on ImageNet-Dogs where ConCURL performs better compared to the baseline model. Even though ConCURL method performs best in max-performance it performs slightly worse on random-10. This result strengthens our argument of the need of going beyond traditional reporting of max-performance based on Acc, NMI and ARI.

2.4. Effect of Data Augmentations (DA)

Table 5. Data Augmentation Details

DA Type	**Abbreviation**	**CIFAR-10**	**CIFAR100-20**
All Data augmentations	All DA	0.8459	0.4798
No random horizontal flip	No RHF	0.7689	0.4551
No random gray scale	No RGS	0.7750	0.4357
No random apply blur	No RAB	0.7907	0.4295
No color jitter	No CJ	0.6463	0.2536
No random resized crop	No RRC	0.2988	0.1322

Augmenting the training data is a standard technique for training deep learning methods (Shorten & Khoshgoftaar, 2019). The backbones used in this study rely on the different views that are generated by applying different augmentations to the input image. Recently (Tian et al., 2020) investigated the impact of data augmentations for contrastive learning methods and shed some light to the topic. In our setting, we would like to quantify the impact of data augmentations on ConCURL. In the table 5, we show the maximum accuracy achieved when all data augmentations are used and when we skip one data augmentation technique at a time. When random resized crop data augmentation is dropped, we have the maximum drop in the accuracy in both datasets followed by color jitter. Other data augmentation techniques are important to get the best possible accuracy but do not have as much effect as color jitter and random resized crop.

3. Conclusion

In this work, we questioned the adequacy of the current evaluation metrics for the deep clustering literature. We argue that the existing max-performance criterion is not strong enough to evaluate clustering algorithms for real world tasks. Towards this end, we propose additional criteria to evaluate deep clustering algorithms that focus on the robustness of the algorithm to various factors such as choice of hyper parameters, out of distribution data, and other algorithmic tricks/techniques. We provide evaluation results for ConCURL and other state of the art clustering algorithms on max-performance criteria where ConCURL outperforms other algorithms in most datasets. However, it's average performance on out-of-distribution criterion highlights the need to use the proposed evaluation methods for deep clustering algorithms.

References

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

Caron, M., Bojanowski, P., Joulin, A., and Douze, M. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 132–149, 2018.

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

Chang, J., Wang, L., Meng, G., Xiang, S., and Pan, C. Deep adaptive image clustering. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

Cuturi, M. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*, pp. 2292–2300, 2013.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Fern, X. Z. and Brodley, C. E. Random projection for high dimensional data clustering: A cluster ensemble approach. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pp. 186–193, 2003.

Fred, A. L. and Jain, A. K. Combining multiple clusterings using evidence accumulation. *IEEE transactions on pattern analysis and machine intelligence*, 27(6):835–850, 2005.

Frey, B. J. and Dueck, D. Clustering by passing messages between data points. *science*, 315(5814):972–976, 2007.

Ghosh, J. and Acharya, A. Cluster ensembles. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(4):305–315, 2011.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. A., Guo, Z. D., Azar, M. G., et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2019.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

Hsu, K., Levine, S., and Finn, C. Unsupervised learning via meta-learning. *arXiv preprint arXiv:1810.02334*, 2018.

Huang, J., Gong, S., and Zhu, X. Deep semantic clustering by partition confidence maximisation. In *CVPR*, June 2020a.

Huang, J., Gong, S., and Zhu, X. Deep semantic clustering by partition confidence maximisation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8849–8858, 2020b.

Jain, A. K. and Dubes, R. C. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

Jain, A. K., Murty, M. N., and Flynn, P. J. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

Ji, X., Henriques, J. F., and Vedaldi, A. Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9865–9874, 2019.

Kuhn, H. W. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

Kuhn, H. W. Variants of the hungarian method for assignment problems. *Naval Research Logistics Quarterly*, 3 (4):253–258, 1956.

Li, Y., Hu, P., Liu, Z., Peng, D., Zhou, J. T., and Peng, X. Contrastive clustering. In *AAAI*, 2021.

Masulli, F. and Schenone, A. A fuzzy clustering based segmentation system as support to diagnosis in medical imaging. *Artificial intelligence in medicine*, 16(2):129–147, 1999.

Niu, C., Zhang, J., Wang, G., and Liang, J. Gatcluster: Self-supervised gaussian-attention network for image clustering. In *ECCV*, pp. 735–751, 2020a.

Niu, C., Zhang, J., Wang, G., and Liang, J. Gatcluster: Self-supervised gaussian-attention network for image clustering. *arXiv preprint arXiv:2002.11863*, 2020b.

Schops, T., Schonberger, J. L., Galliani, S., Sattler, T., Schindler, K., Pollefeys, M., and Geiger, A. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3260–3269, 2017.

Shah, S. A. and Koltun, V. Deep continuous clustering. *arXiv preprint arXiv:1803.01449*, 2018.

Shorten, C. and Khoshgoftaar, T. M. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019.

Strehl, A. and Ghosh, J. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002.

Tao, Y., Takagi, K., Nakata, K., and Center, C. R. Clustering-friendly representation learn-ing via instance discrimination and feature decorrelation.

Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., and Isola, P. What makes for good views for contrastive learning. *arXiv preprint arXiv:2005.10243*, 2020.

Wu, J., Long, K., Wang, F., Qian, C., Li, C., Lin, Z., and Zha, H. Deep comprehensive correlation mining for image clustering. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 8150–8159, 2019.

Wu, Z., Xiong, Y., Yu, S. X., and Lin, D. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3733–3742, 2018.

Xie, J., Girshick, R., and Farhadi, A. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pp. 478–487, 2016.

Xu, R. and Wunsch, D. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678, 2005.

Zhuang, C., Zhai, A. L., and Yamins, D. Local aggregation for unsupervised learning of visual embeddings. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6002–6012, 2019.

A. Consensus Clustering

One of the distinguishing factors between supervised learning and unsupervised learning is the existence of ground-truth labels which construct a global constraint over examples. In most of the self-supervised learning methods, ground-truth is replaced with some consistency constraint (Chen et al., 2020). Without a doubt, the performance of any self-supervised method is a function of the power of the used consistency constraint. We define two types of consistency constraints - exemplar consistency and population consistency.

Definition 1. ***Exemplar consistency****: Representation learning algorithms that learn closer representation (in some distance metric) for different augmentations of the same data point are said to follow exemplar consistency.*

Examples of usage of exemplar consistency include contrastive learning methods like MoCo (He et al., 2019), SimCLR (Chen et al., 2020), etc. In these methods, positive pairs of images are defined as any two image augmentations of the same image and negative pairs are any two different images.

Definition 2. ***Population consistency****: Representation learning algorithms which ensure learned representations satisfy the consistency constraint that two similar data points or any augmentation of the same data points should belong to the same cluster (or population) are said to follow population consistency.*

Deep Cluster (Caron et al., 2018) is a prominent self-supervised method which utilizes a population consistency, i.e. Definition 2, by enforcing a clustering on the data set. Please note that each cluster assignments contain data points that are similar to each other. Similarly, SwAV (Caron et al., 2020) is an example of population consistency method.

Definition 3. ***Consensus consistency****: Representation learning algorithms which are able to learn representations that induce similar partitions for variations in the representation space (subsets of features, random projection, etc), different clustering algorithms (k-means, GMM, etc) or different initialization of clustering algorithms are said to follow consensus consistency.*

Earlier works on *consensus consistency* do not consider representation learning and use the knowledge reuse framework (see (Strehl & Ghosh, 2002),(Ghosh & Acharya, 2011)) where the cluster partitions are available (features are irrelevant) or the features of the data are fixed. In contrast, the notion of consensus consistency here deals with learning representations that achieve a consensus on the cluster assignments of multiple clustering algorithms.

Unfortunately, the definition of consensus consistency is ill-posed, there can be arbitrarily many different partitions which can satisfy the given condition[1]. We show that when exemplar consistency used as an inductive bias the resulting objective function shows impressive performance on challenging data sets. Combining exemplar and population constraints with consensus consistency seamlessly and effectively for clustering is the basis of our proposed method.

A.1. Loss for Consensus and Population Consistency

We assume that there is an underlying latent space $\mathcal{Z}^*$ (possibly not unique) such that all clusterings (based on variation

[1] a) Degenerate solution where all clusters assignments are the same, b) Random assignment can satisfy this condition given that all clustering are producing the same but random assignments

of latent space, algorithm or initialization) that take input data from this latent space produce similar partitions of the data. Furthermore, every clustering algorithm that also takes the true number of clusters as input, produces the partition that is closest to the hypothetical ground truth. Moreover, we are assuming that there exists a function $h : X \rightarrow \mathcal{Z}^*$, where X represents the input space and $\mathcal{Z}^*$ represents the underlying latent space. We call this assumption the **principle of consensus**. The open question is how one constructs an efficient loss which reflects the **principle of consensus**. We define one such way below.

Given an input batch of images $\mathcal{X}_b \subset \mathcal{X}$, the goal is to partition these images into K clusters. We get p views (by different image augmentations) of these images and define a loss such that cluster assignment of any of the p views match the targets estimated from any other view. Without loss of generality we define a loss for $p = 2$ views. The two views $\mathcal{X}_b^1, \mathcal{X}_b^2$ are generated using two randomly chosen image augmentations. We learn a representation space $\mathcal{Z}_0$ at the end of every training iteration and get M variations of $\mathcal{Z}_0$ as $\{\mathcal{Z}_1, \mathcal{Z}_2, ..., \mathcal{Z}_M\}$ (e.g. random projections, etc). The goal is to build an efficient loss by the **principle of consensus** among $\mathcal{Z}_0$ and it's M variations $\{\mathcal{Z}_1, \mathcal{Z}_2, ..., \mathcal{Z}_M\}$ such that we learn the latent space $\mathcal{Z}^*$ at the end of training (i.e. the learned features lie in the latent space described above). For a given batch of images $\mathcal{X}_b$ and representation space $\mathcal{Z}_m, \forall m \in [0, 1, ..., M]$, we denote the cluster assignment probability for image i and cluster j for view 1 as $\mathbf{p}_{i,j}^1(\mathcal{Z}_m)$ and for view 2 as $\mathbf{p}_{i,j}^2(\mathcal{Z}_m)$. Here we define the loss that incorporates "population consistency" and "consensus consistency". We assume that oracle has given us target cluster assignment probabilities for the representation $\mathcal{Z}_0$ (like in DeepCluster(Caron et al., 2018)) and denote it as $\mathbf{q}_{i,j}^1$ for view 1 and $\mathbf{q}_{i,j}^2$ for view 2.

We define loss for any representation space $\mathcal{Z}$ and batch of images $\mathcal{X}_b$ as

$$
\begin{aligned}
L_{\mathcal{Z}}^1 &= -\frac{1}{2B}\sum_{i=1}^{B}\sum_{j=1}^{K}\mathbf{q}_{ij}^2 \log \mathbf{p}_{ij}^1(\mathcal{Z}) \\
L_{\mathcal{Z}}^2 &= -\frac{1}{2B}\sum_{i=1}^{B}\sum_{j=1}^{K}\mathbf{q}_{ij}^1 \log \mathbf{p}_{ij}^2(\mathcal{Z}), \\
L_{\mathcal{Z}} &= \sum_{i=0}^{M}\Big(L_{\mathcal{Z}_i}^1 + L_{\mathcal{Z}_i}^2\Big).
\end{aligned} \tag{1}
$$

Note that, here consensus over the clusterings is defined via common targets $\mathbf{q}$. The exact details of how to get variations of $\mathcal{Z}_0$, calculate cluster assignment probabilities $\mathbf{p}$ and targets $\mathbf{q}$ are described in the next section.

A.2. End-to-End SGD Trainable Consensus Loss

In this section we propose an end-to-end trainable algorithm and define a way to compute $\mathbf{p}$ and $\mathbf{q}$. When cluster assignment probabilities $\mathbf{p}$ can take any value in the set $[0, 1]$ then we refer to it as soft clustering and when $\mathbf{p}$ is restricted to the set $\{0, 1\}$ then we refer to it as hard clustering. The proposed method can be used with hard clustering and soft clustering.

Without loss of generality, in this paper, we focus on soft clustering which makes it easier to define a loss using the probabilities and update the parameters using the gradients to enable end-to-end learning. We follow the soft clustering framework presented in SwAV (Caron et al., 2020), which is a centroid based technique and aims to maintain consistency between the clusterings of the augmented views $\mathcal{X}_b^1$ and $\mathcal{X}_b^2$. We store a set of randomly initialized prototypes $C_0 = \{\mathbf{c}_0^1, \cdots, \mathbf{c}_0^K\} \in \mathbb{R}^{d\times K}$, where K is the number of clusters, and d is the dimension of the prototypes. These prototypes are used to represent clusters and define a "consensus consistency" loss. We compute M variations of C_0 as $C_1, ..., C_M$ exactly like we compute M variation of $\mathcal{Z}_0$.

A.2.1. Cluster assignment probability $\mathbf{p}$

We use a two layer MLP g to project the features $\mathbf{f}^1 = f_\theta(\mathcal{X}_b^1)$ and $\mathbf{f}^2 = f_\theta(\mathcal{X}_b^2)$ to a lower dimensional space $\mathcal{Z}_0$ (of size d). The output of this MLP (referred to as cluster embeddings) is denoted using $Z_0^1 = \{\mathbf{z}_0^{1,1}, \ldots, \mathbf{z}_0^{1,B}\}$ and $Z_0^2 = \{\mathbf{z}_0^{2,1}, \ldots, \mathbf{z}_0^{2,B}\}$ for view 1 and view 2 respectively. Note that $h : \mathcal{X} \rightarrow \mathcal{Z}$ defined in A.1 is equivalent to composite function of $f : \mathcal{X} \rightarrow \Phi$ and $g : \Phi \rightarrow \mathcal{Z}$. For a latent space $\mathcal{Z}$, we compute the probability of assigning a cluster j to image i using the normalized vectors $\bar{\mathbf{z}}^{1,i} = \frac{\mathbf{z}^{1,i}}{||\mathbf{z}^{1,i}||}$, $\bar{\mathbf{z}}^{2,i} = \frac{\mathbf{z}^{2,i}}{||\mathbf{z}^{2,i}||}$ and $\bar{\mathbf{c}}_j = \frac{\mathbf{c}^{\mathbf{j}}}{||\mathbf{c}^{\mathbf{j}}||}$ as

$$
\begin{aligned}
\mathbf{p}_{i,j}^1(\mathcal{Z}, C) &= \frac{\exp(\frac{1}{\tau}\langle\bar{\mathbf{z}}_i^1, \bar{\mathbf{c}}_j\rangle)}{\sum_{j'}\exp(\frac{1}{\tau}\langle\bar{\mathbf{z}}_i^1, \bar{\mathbf{c}}_{j'}\rangle)}, \\
\mathbf{p}_{i,j}^2(\mathcal{Z}, C) &= \frac{\exp(\frac{1}{\tau}\langle\bar{\mathbf{z}}_i^2, \bar{\mathbf{c}}_j\rangle)}{\sum_{j'}\exp(\frac{1}{\tau}\langle\bar{\mathbf{z}}_i^2, \bar{\mathbf{c}}_{j'}\rangle)}.
\end{aligned} \tag{2}
$$

We concisely write $\mathbf{p}_i^1(\mathcal{Z}) = \{\mathbf{p}_{i,j}^1(\mathcal{Z}, C)\}_{j=1}^K$ and $\mathbf{p}_i^2 = \{\mathbf{p}_{i,j}^2(\mathcal{Z}, C)\}_{j=1}^K$. Here, τ is a temperature parameter and we set the value to 0.1 similar to Caron *et al.,* (Caron et al., 2020). Note that, we use $\mathbf{p}_i$ to denote the predicted cluster assignment probabilities for image i (when not referring to a particular view), and a shorthand $\mathbf{p}$ is used when i is clear from context.

A.2.2. TARGETS $\mathbf{q}$

The predicted assignments $\mathbf{p}$ are compared against high-confidence estimates $\mathbf{q}$ (referred to as codes henceforth). We follow (Asano et al., 2019; Caron et al., 2020) to compute $\mathbf{q}$ by enforcing an equipartition constraint using the Sinkhorn-Knopp algorithm (Cuturi, 2013).

$$\begin{aligned} Q^1 &= \arg\max_{Q\in\mathcal{Q}} \text{Tr}(Q^T C_0^T Z_0^1) + \epsilon H(Q) \\ Q^2 &= \arg\max_{Q\in\mathcal{Q}} \text{Tr}(Q^T C_0^T Z_0^2) + \epsilon H(Q), \end{aligned} \tag{3}$$

where $Q = \{\mathbf{q}_1, \ldots, \mathbf{q}_B\} \in \mathbb{R}_+^{K\times B}$, $\mathcal{Q}$ is the transportation polytope defined by

$$\mathcal{Q} = \{\mathbf{Q} \in \mathbb{R}_+^{K\times B} | \mathbf{Q}\mathbf{1}_B = \frac{1}{K}\mathbf{1}_K, \mathbf{Q}^T\mathbf{1}_K = \frac{1}{B}\mathbf{1}_B\}$$

$\mathbf{1}_K$ is a vector of ones of dimension K and $H(Q) = -\sum_{i,j} Q_{i,j} \log Q_{i,j}$. The above optimization is computed using a fast version of the Sinkhorn-Knopp algorithm (Cuturi, 2013) as described in Caron *et al.,* (Caron et al., 2020).

After computing the codes Q^1 and Q^2, in order to maintain consistency between the clusterings of the augmented views, the loss is computed using the probabilities $\mathbf{p}_{ij}$ and the assigned codes $\mathbf{q}_{ij}$ by comparing the probabilities of view 1 with the assigned codes of view 2 and vice versa like in equation 1.

A.2.3. DEFINING VARIATIONS OF Z_0 AND C_0

To compute $\{Z_1, ..., Z_M\}$, we project d dimensional space Z_0 to a D dimension space using random projection matrix. We follow the same procedure to compute $\{C_1, ..., C_M\}$ from C_0. At the beginning of the algorithm, we randomly initialize M such transformations and fix them throughout training. Suppose using a particular random transformation (a randomly generated matrix A), we get $\tilde{\mathbf{z}} = A\mathbf{z}$, $\tilde{\mathbf{c}} = A\mathbf{c}$. We then compute the softmax probabilities using the normalized vectors $\tilde{\mathbf{z}}/||\tilde{\mathbf{z}}||$ and $\tilde{\mathbf{c}}/||\tilde{\mathbf{c}}||$. Repeating this with the M transformations results in M predicted cluster assignment probabilities for each view. When the network is untrained, the embeddings $\mathbf{z}$ are random and applying the random transformations followed by computing the predicted cluster assignments leads to a diverse set of soft cluster assignments. The parameter weights are trained by using the stochastic gradients of the loss for updates.

A.2.4. TOTAL LOSS

To capture exemplar consistency better and based on previous evidence of successful clustering with Instance Discrimination (ID) approach (Tao et al.), we use Instance Discrimination (ID) (Wu et al., 2018) as one of the losses like in (Tao et al.). The exemplar objective of ID is to classify each image as it's own class using loss L_b. We exactly follow a framework from (Wu et al., 2018) to implement the ID loss L_b.

The final loss that we sought to minimize is the combination of the losses $L_{\mathcal{Z}}$ (equation 1) and L_b

$$L_{\text{total}} = \alpha L_{\mathcal{Z}} + \beta L_b. \tag{4}$$

where α, β are non-negative constants.

A.2.5. COMPUTING THE CLUSTER METRICS

In this section, we describe the approach to compute the cluster assignments and the metrics evaluate their quality. Note that we assume the number of true clusters (K) in the data is known. We use the embeddings generated by the backbone (here it is the output of the ID block $f_\theta(x)$) and perform K-means clustering.

We evaluate the quality of the clusterings using metrics such as cluster accuracy, NMI, ARI. To compute clustering accuracy, we are required to solve an assignment problem (computed using a Hungarian match(Kuhn, 1955; 1956)) between the true class labels and the cluster assignment.

A.3. Generating Multiple Clusterings

Table 6. Different ways to generate ensembles

Data Representation	Clustering algorithms
Different data preprocessing techniques	Multiple clustering algorithms (k-means, GMM, etc)
Subsets of features	Same algorithm with different parameters or initializations
Different transformations of the features	Combination of multiple clustering and different parameters or initializations

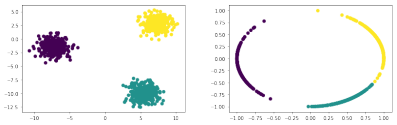
Fred and Jain, (Fred & Jain, 2005) discuss different ways to generate an ensemble of clusterings which are tabulated in Table 6. In our proposed algorithm, we focus on choosing of data representation to generate cluster ensembles.

By fixing a stable clustering algorithm, we can generate arbitrarily large ensembles by applying different transformations on the embeddings. Random projections were successfully used in Consensus Clustering previously (Fern & Brodley, 2003). By generating ensembles using random projections, we have control over the amount of diversity we can induce into the framework, by varying the dimension of the random projection. In addition to Random Projections, we

also used diagonal transformations (Hsu et al., 2018) where different components of the representation vector are scaled differently. Hsu *et al.,* (Hsu et al., 2018) illustrate that such scaling enables a diverse set of clusterings which is helpful for their meta learning task.

B. Understanding the Consensus Objective

We investigate a potential hypothesis of "training driven by noisy cluster assignments" that can shed light on the success of ConCURL [2]. The hypothesis stems from the following intuition: *Using different clustering algorithms, the generated cluster assignments are noisy versions of the hypothetical ground truth; as training progresses, the noise in the cluster assignments reduces and eventually all different clustering algorithms considered generate similar cluster assignments*.



(a) (b)

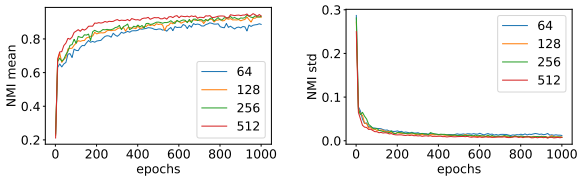
Figure 3. a: Three cluster dataset = z, b: Normalized data

We verify this hypothesis empirically with the help of the following experiments on STL-10 dataset: (i) observe noisy clusterings generated by using Random Projections and (ii) verify that the noise in the cluster assignments reduces as training progresses.

For the purpose of demonstrating noisy cluster assignments, we use synthetic data as follows. We generate three clusters in $\mathbb{R}^2$ as shown in Fig 3(a) and compute the centroids of each cluster. Here, the centroids act as the prototypes. We then generate a Gaussian Random Projection matrix A of dimension $\mathbb{R}^{2\times 2}$. We first normalize the embeddings (2 dim features) and the centroids (see Fig 3(b)). Using the matrix A, we transform both the embeddings and prototypes to the new space and normalize the resultant vectors (see Fig **??**).

We follow the soft clustering framework discussed earlier and compute the soft cluster assignments for the original and the transformed data. We observe that, these probability of cluster assignments in the new space are a noisy version of probability of cluster assignments in the original space (see Table 7).

To verify that the noise in the cluster assignment probabilities reduces as training progresses, we perform the following experiment. We measure the similarity in cluster assignments at every epoch to observe the effect of consensus as training progresses. For each random projection used, we use cluster assignment probabilities $\tilde{\mathbf{p}}$ and compute cluster assignments by taking an $\arg\max$ on $\tilde{\mathbf{p}}$ of each image. We get M such cluster assignments due to the M random projections. We then compute a pairwise NMI (Normalized Mutual Information) (similar analysis to Fern and Brodley, (Fern & Brodley, 2003)) between every two cluster assignments, and compute the average and standard deviation of the pairwise NMI across the $\frac{M(M-1)}{2}$ pairs. An NMI score of 1.0 signifies that the two clusterings perfectly correlate with each other, and a score of 0.0 implies that the two clusterings are uncorrelated. We observe from Figure 2 that the pairwise NMI increases as training progresses and becomes closer to 1. At the beginning of training, the cluster assignments are very diverse (small NMI score with a larger standard deviation); and as training progresses, the diversity is reduced (large NMI score with a smaller standard deviation). This observation leads us to conclude that for the clustering algorithms (defined using different random projections) we have learned an embedding space where the different cluster assignments concur. In other words, "consensus consistency" is achieved. Also, it is evident from our empirical results in Section 2 that we achieve better clustering accuracy overall.



(a) (b)

Figure 4. Pairwise NMI as a way to measure the diversity in the ensemble. The results are for STL-10 dataset, and shows pairwise NMI for different random projection dimensions (the original dimension of $\mathbf{z}$ is 256).

If noisy cluster assignments are to be the reason behind the improved performance, one might wonder if it is sufficient to simply add noise to the original cluster assignments rather than computing multiple cluster assignments. However this may not be fruitful because if noise is added externally, one will have to define a scheduler to reduce the noise as training progresses. But in the case of ConCURL, end-to-end learning algorithm figures out the rate of consensus or agreement between $\mathbf{p}$ and $\tilde{\mathbf{p}}$ itself. In the next section we give empirical evidence of the effectiveness of our method.

C. Implementation Details

In this section, we discuss the implementation of the proposed algorithm. We use Pytorch version 1.7.1 for the im-

[2] A theoretically grounded explanation of ConCURL is considered as a future work due to the non-convexity of deep learning methods and non-convexity of proposed loss.

Table 7. Predicted cluster assignment probability and target probability from Sinkhorn algorithm for four data points

Cluster assignment probability $\mathbf{p}$

	Cluster 1	Cluster 2	Cluster 3
1	0.6473	0.2587	0.0940
2	0.7180	0.1812	0.1008
3	0.0832	0.3160	0.6008
4	0.1543	0.5917	0.2541

Cluster assignment probability $\tilde{\mathbf{p}}$

	Cluster 1	Cluster 2	Cluster 3
1	0.6764	0.2271	0.0965
2	0.7305	0.1680	0.1015
3	0.0802	0.3371	0.5827
4	0.1357	0.5784	0.2860

targets $\mathbf{q}$ from sinkhorn

	cluster 1	cluster 2	cluster 3
1	1.0	8.9e-09	1.7e-17
2	1.0	9.1e-13	8.7e-18
3	6.8e-18	2.2e-6	1.0
4	2.5 e-12	1.0	5.4e-8

plementation. The ID block of the algorithm uses the code from the implementation of ID (Wu et al., 2018) available at https://github.com/zhirongw/lemniscate.pytorch. The repository uses Pytorch version 0.3, and appropriate changes were made to use it with the latest version of Pytorch. We used ResNet-18 and ResNet-50 blocks during our experiments. In the ID block, the ResNet architecture is modified as follows. The final fully connected layer consists of 128 dimensions instead of the usual 1000 dimensions. The output of the final fully connected layer proceeds to compute the NCE loss, and the feature representations (layer before the fully connected layer) are fed to the clustering part.

For the clustering part, the MLP projection head g consists of a hidden layer of size 2048, followed by batch normalization and ReLU layers, and an output layer of size 256. The prototypes are thus chosen to be of dimension 256. Note that we fixed the number of prototypes to be equal to the number of ground truth classes of the dataset. It was shown however that over-clustering leads to better representations (Caron et al., 2020; Ji et al., 2019; Asano et al., 2019) and we can extend our model to include an over-clustering block with a larger set of prototypes (Ji et al., 2019) and alternate the training procedure between the blocks.

We train the algorithm for 2000 epochs for all datasets. We use SGD optimizer with a learning rate decay of 0.1 at prespecified epochs (600, 950, 1300, 1650, 2000) to perform the updates for all datasets. We performed a coarse learning rate search and found 0.03 to be best performing. We used a batch size of 128 for all the datasets. To evaluate the cluster accuracy, we computed cluster assignments using MiniBatchKMeans[3] with a batch size of 6000 and 20 random initializations.

C.1. Image Augmentations

The different views $\mathcal{X}_b^1, \mathcal{X}_b^2$ are not the same as views in multi-view datasets (Schops et al., 2017). The views referred to in this paper correspond to different augmented views that are generated by image augmentation techniques, such as 'RandomHorizontalFlip', 'RandomCrop'. We explain the generation of multiple augmented views that are shown to be very effective in unsupervised learning (Chen et al., 2020). Indeed it is possible to use more than two augmented views, but we limit to two for the sake of simplicity. (Caron et al., 2020) propose an augmentation technique (Multi-Crop) to use more than two views. In this work, we use the augmentations used in (Chen et al., 2020; Grill et al., 2020). We first crop a random patch of the image with scale ranging from 0.08 to 1.0, and resize the cropped patch to 224×224 (96×96 in the case of smaller resolution datasets such as STL10). The resulting image was then flipped horizontally with a probability of 0.5. We then apply color transformations, starting by applying gray-scale with a probability of 0.2 followed by randomly changing the brightness, contrast, saturation and hue with a probability of 0.8. Then we applied a Gaussian Blur with kernel size (23×23) and a sigma chosen uniformly randomly between 0.1 and 2.0. The probability of applying the Gaussian Blur was 1.0 for view 1 and 0.5 for view 2. During evaluation, we resized the image such that the smaller edge of the image is of size 256 (not required for STL, CIFAR10, CIFAR100-20), and a center crop is performed with the resolution mentioned in the main paper. We finally normalize the image channels with the mean and standard deviation computed on ImageNet. Additionally, during training, we also experimented with applying a Sobel filter after all the image augmentations are performed before the forward pass. Applying a Sobel filter reduces the number of channels of the input images to 2. We also experimented with augmenting the RGB images with the output of the Sobel transform, resulting in 5 channel input images. In both these cases, the input channels in the first convolution layer were modified accordingly. All image augmentations were computed using Pytorch's torchvision module available in version 1.7.1.

C.2. Random Transformations

To compute the random transformations on the embeddings $\mathbf{z}$, we followed two techniques. We used Gaussian random projections with output dimension d, and transformed the embeddings $\mathbf{z}$ to the new space with dimension d. In Gaussian random projections, the projection matrix is generated by picking rows from a Gaussian distribution such that they are orthogonal. We also used diagonal transformation (Hsu et al., 2018) where we multiply $\mathbf{z}$ with a randomly generated diagonal matrix of the same dimension as $\mathbf{z}$. We initialized

[3] https://scikit-learn.org/stable/modules/clustering.html#mini-batch-kmeans

M random transformations at the beginning and are kept fixed throughout the training.

Table 8. Hyperparameters for max-performance

	τ	l	$\exp(\eta)$	d
STL-10	0.8	0.03	4	80
ImageNet-10	1.0	0.03	1	104
ImageNet-Dogs	0.5	0.06	64	136
CIFAR-10	0.85	0.015	64	152
CIFAR100-20	0.35	0.06	16	32

D. Evaluation Metrics

We evaluate our algorithm by computing traditional clustering metrics (Cluster Accuracy, Normalized Mutual Information, and Adjusted Rand Index) which we discuss below in detail.

D.1. Cluster Accuracy

The clustering accuracy is computed by first computing a cluster partition of the input data. Once the partitions are computed and cluster indices assigned to each input data point, the linear assignment map is computed using Kuhn-Munkres (Hungarian) algorithm that reassigns the cluster indices to the true labels of the data. Clustering accuracy is then given by

$$ACC = \frac{\sum_{i=1}^{N} \mathbb{I}\{y_{true}(x_i) = c(x_i)\}}{N},$$

where $y_{true}(x_i)$ is a true label of x_i and $c(x_i)$ is the cluster assignment produced by an algorithm (after Hungarian mapping).

D.2. Normalized Mutual Information

For two clusterings U, V, with each containing $|U|, |V|$ clusters respectively, and let $|U_i|$ be the number of samples in cluster U_i of clustering U (similarly for V) , Mutual Information (MI) is given by

$$MI(U,V) = \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} \frac{|U_i \cap V_i|}{N} \log \frac{N|U_i \cap V_j|}{|U_i||V_j|}$$

where N is the number of data points under consideration. Normalized Mutual Information is defined as

$$NMI(U,V) = \frac{MI(U,V)}{\sqrt{MI(U,U)MI(V,V)}}$$

D.3. Adjusted Rand Index

[4] Suppose R is the groundtruth clustering and S is a partition, the RI of S is given as follows. Let a be the number of pairs of elements that are in the same set in R as well as in S; b be the number of pairs of elements that are in diferent sets in R, and different sets in S. Then

$$RI = \frac{a+b}{\binom{n}{2}}$$

$$ARI = \frac{RI - \mathbb{E}[RI]}{\max(RI) - \mathbb{E}[RI]}$$

E. Ablation Studies

In this section, we provide abalation studies for ConCURL. Although the proposed method is trained end-to-end, each different component of the method may have different impact. We have conducted five controlled experiments to quantify impact of used effect of data augmentations (Section 2.4), image resolution (Section E.1), the number of transformations (Section E.2), the dimension of transformation (Section E.2) and architecture choice (Section E.3).

E.1. Effect of image resolution

Table 9. Effects of different resolutions for STL-10 , ImageNet-10 and ImageNet-Dogs .

	32	64	96	160	224
STL-10	0.531	0.724	0.749	NA	NA
ImageNet-10	NA	NA	0.887	0.958	0.946
ImageNet-Dogs	NA	NA	0.629	0.695	0.679

Image resolution is often considered a free parameter and its effect on clustering performance not evaluated rigorously (Niu et al., 2020a; Tao et al.). We try to quantify the effect of different resolutions to the extent possible, given that some datasets are available only in specific resolutions. For STL-10 we have used 32×32, 64×64 and 96×96 as resolutions. For ImageNet-10 and ImageNet-Dog-15 we have used 96×96, 160×160 and 224×224 as resolutions. The results are given in Table 9.

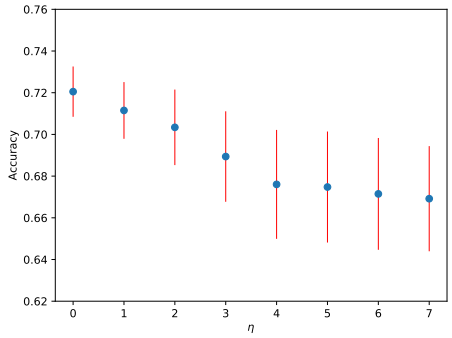
The best performance for ImageNet-10 and ImageNet-Dogs was at resolution 160 and for STL-10 it was at resolution 96. It is not clear why ImageNet-10 and ImageNet-Dogs did not perform the best at high resolution and further investigation is needed which we keep it as an open problem.

E.2. Components of Consensus Loss

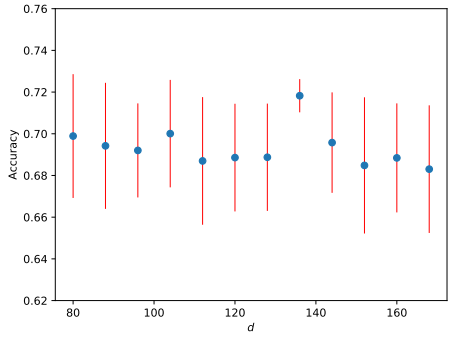
The proposed consensus loss has two parameters. First one is the number of transformations used and the second one is the dimension of projection space. To understand the proposed loss, we have conducted a detailed experimental study on STL-10 and CIFAR100-20 [5]. The used hyper-

[4] https://scikit-learn.org/stable/modules/clustering.html#adjusted-rand-score

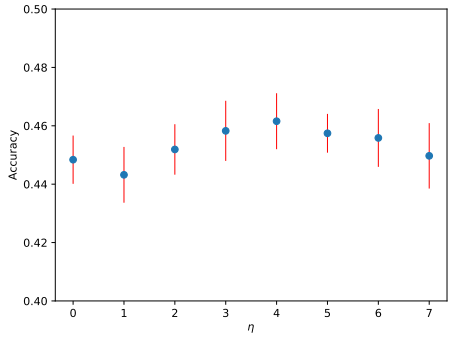
[5] We have conducted similar but a smaller study on remaining data set and we have observed similar trends.



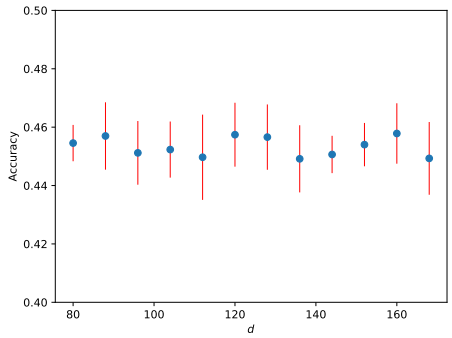
(a) STL-10 : log of number of transformations



(b) STL-10 dimension of projection space



(c) CIFAR100-20 log of number of transformations



(d) CIFAR100-20 dimension of projection

Figure 5. Components of consensus loss: ablation of STL-10 and CIFAR100-20

Table 10. Hyperparamers and the range values used for the experiments. τ : temperature parameter, l : learning rate, η : natural log of number of transformations, d: dimension of projection space

	min	**max**	**step size**
τ	0.3	1	0.1
l	0.015	0.09	0.015
η	0	7	1
d	80	168	8

parameters are given in Table 10.

Due to the sheer number of conducted experiments, we will supply a summary statistics on a random set. We report empirical mean and standard deviation of marginal distribution of the quantity under investigation. Let $P_{\tau,\eta,d,l}$ be the joint distribution over hyperparameters τ (temperature parameter), l (learning rate), η (natural log of number of transformations) and d (dimension of projection space). We consider n_h as number of distinct values used in experiment for each hyperparameter $h \in \{\tau, \eta, d, l\}$ based on Table 10. We denote accuracy from each experiment based on the hyperparameters used as $a_{\tau,\eta,d,l}$ Let $P_{h_i|h_j}$ be the conditional marginal distribution of hyperparameter h_i given h_j and conditional empirical mean of $P_{h_i|h_j}$ be $m(P_{h_i|h_j})$. In this case conditional empirical mean $m(P_{h_i|h_j})$ when $h_i = d$ and $h_j = \tau$ can be calculated using $m(P_{d|\tau}) = \frac{1}{n_\eta \times n_l} \sum_\eta \sum_l a_{\tau,\eta,d,l}$. Conditional empirical mean and standard deviation of other hyperparameters is calculated in the same way. In the figure 5, we show conditional empirical mean with blue dot and red line around the dot represents a standard deviation. For both, STL-10 and CIFAR100-20 , we see the trend in number of projections. In STL-10 , smaller the number of random projections, better it is and in CIFAR100-20 increasing random projections is helpful for the clustering accuracy upto some point. Note that when number of random projections is equal to zero, our setting is equivalent to baseline ID and we always perform better than ID. This means optimal number of random projections is greater or equal to one. There is no such a clear trend in number of dimensions of random projections.

E.3. Effect of architecture choice

In this work we have used ResNet-18 and ResNet-50 as network architectures. For both ResNet-18 and ResNet-50 we sweep over the same set of hyper parameter choices i.e., temperature, number of projections and projection dimension and report the results. Figure 6 shows the distribution of Δ_{acc} which is defined as accuracy difference between ResNet-50 and ResNet-18. Figure 6 indicates that ResNet-50 slightly outperforms ResNet-18, i.e. the mean difference is app. 0.5%.

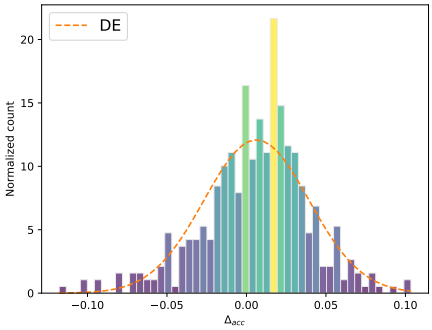


Figure 6. Empirical distribution of performance difference between ResNet-50 and ResNet-18. DE is the density estimate of the empirical distribution.