LSD-C: Linearly Separable Deep Clusters

Anonymous ICCV submission

Paper ID ****

Abstract

We present LSD-C, a novel method to identify clusters in an unlabeled dataset. Our algorithm first establishes pairwise connections in the feature space between the samples of the minibatch based on a similarity metric. Then it regroups in clusters the connected samples and enforces a linear separation between clusters. This is achieved by using the pairwise connections as targets together with a binary cross-entropy loss on the predictions that the associated pairs of samples belong to the same cluster. This way, the feature representation of the network will evolve such that similar samples in this feature space will belong to the same linearly separated cluster. Our method draws inspiration from recent semi-supervised learning practice and proposes to combine our clustering algorithm with self-supervised pretraining and strong data augmentation. We show that our approach significantly outperforms competitors on popular public image benchmarks including CIFAR 10/100, STL 10 and MNIST, as well as the document classification dataset Reuters 10K. Our code is available at https://www.dropbox.com/s/xnczi9ogw0ar87d/lsd-clusters.zip?dl=0.

1. Introduction

The need for large scale labelled datasets is a major obstacle to the applicability of deep learning to problems where labelled data cannot be easily obtained. Methods such as clustering, which are unsupervised and thus do not require any kind of data annotation, are in principle more easily applicable to new problems. Unfortunately, standard clustering algorithms [7, 10, 38, 40] usually do not operate effectively on raw data and require to design new data embeddings specifically for each new application. Thus, there is a significant interest in automatically learning an optimal embedding while clustering the data, a problem sometimes referred to as simultaneous data clustering and representation learning. Recent works have demonstrated this for challenging data such as images [27, 51] and text [28, 43]. However, most of these methods work with a constrained output space, which

usually coincides with the space of discrete labels or classes being estimated, therefore forcing to work at the level of the semantic of the clusters directly.

In this paper, we relax this limitation by introducing a novel clustering method, *Linearly Separable Deep Clustering* (LSD-C). This method operates in the feature space computed by a deep network and builds on three ideas. First, the method extracts mini-batches of input samples and establishes pairwise pseudo labels (connections) for each pair of sample in the mini-batch. Differently from prior art, this is done in the space of features computed by the penultimate layer of the deep network instead of the final output layer, which maps data to discrete labels. From these pairwise labels, the method learns to regroup the connected samples into clusters by using a clustering loss which forces the clusters to be linearly separable. We empirically show in section 4.2 that this relaxation already significantly improves clustering performance.

Second, we initialize the model by means of a self-supervised representation learning technique. Prior work has shown that these techniques can produce features with excellent linear separability [4, 16, 22] that are particularly useful as initialization for downstream tasks such as semi-supervised and few-shot learning [14, 41, 54].

Third, we make use of very effective data combination techniques such as RICAP [48] and MixUp [55] to produce composite data samples and corresponding pseudo labels, which are then used at the pairwise comparison stage. In section 4 we show that training with such composite samples and pseudo labels greatly improves the performance of our method, and is in fact the key to good performance in some cases.

We comprehensively evaluate our method on popular image benchmarks including CIFAR 10/100, STL 10 and MNIST, as well as the document classification dataset Reuters 10K. Our method consistently outperforms competitors on all datasets, showing promising results. The rest of the paper is organized as follows. We first review the most relevant works in section 2. Next, we develop the details of our proposed method in section 3, followed by the experimental results, ablation studies and analysis in section 4. Our

code is available at https://www.dropbox.com/s/xnczi9ogw0ar87d/lsd-clusters.zip?dl=0.

2. Related work

Deep clustering. Clustering has been a long-standing problem in the machine learning community, including well-known algorithms such as K-means [38], mean-shift [7], DBSCAN [10] or Gaussian Mixture models [40]. Furthermore it can also be combined with other techniques to achieve very diverse tasks like novel category discovery [21, 11] or semantic instance segmentation [9] among others. With the advances of deep learning, more and more learning-based methods have been introduced in the literature [12, 13, 19, 26, 28, 36, 44, 51, 52]). Among them, DEC [51] is one of the most promising method. It is a two stage method that jointly learns the feature embedding and cluster assignment. The model is first pretrained with an autoencoder using reconstruction loss, after which the model is trained by constructing a sharpened version of the soft cluster assignment as pseudo target. This method inspired a few following works such as IDEC [17] and DCED [18]. JULE [53] is a recurrent deep clustering framework that jointly learns the feature representation with an agglomerative clustering procedure, however it requires tuning a number of hyper-parameters, limiting its practical use. More recently, several methods have been proposed based on mutual information [5, 25, 27]. Among them, IIC [27] achieves the current state-of-the-art results on image clustering by maximizing the mutual information between two transformed counterparts of the same image. Closer to our work is the DAC [3] method, which considers clustering as a binary classification problem. By measuring the cosine similarity between predictions, pairwise pseudo labels are generated from the most confident positive or negative pairs. With the generated pairwise pseudo labels, the model can then be trained by a binary cross-entropy loss. DAC can learn the feature embedding as well as the cluster assignment in an end-to-end manner. Our work significantly differs from DAC as it generates pairwise predictions from a less constrained feature space using similarity techniques not limited to cosine distance.

Self-supervised representation learning. Self-supervised representation learning has recently attracted a lot of attention. Many effective self-supervised learning methods have been proposed in the literature [1, 2, 4, 15, 16, 22]. DeepCluster [2] learns feature representation by classification using the pseudo labels generated from K-means on the learned features in each training epoch. RotNet [16] randomly rotates an image, and learns to predict the applied rotations. Very recently, contrastive learning based methods MoCo [22] and SimCLR [4] have achieved the state-of-the-art self-supervised representation performance, surpassing the representation

learnt using ImageNet labels. Self-supervised learning has been also applied in few-shot learning [14], semi-supervised learning [41, 54] and novel category discovery [20], which successfully boosts their performance. In this work we make use of the provably well-conditioned feature space learnt from self-supervised learning method to initialize our network and avoid degenerative cases. Recently, the concurrent work [50] also explored the effectiveness of self-supervised learning for clustering, by mining the nearest neighbors using pretrained self-supervised learning model. The nearest neighbours are used to train the model for clustering by enforcing the consistency between predictions of nearest neighbors. Promising results have been achieved by this method, which is consistent with our finding about the effectiveness of self-supervised learning for clustering. Our paper provides a different view of leveraging self-supervised representation for clustering and also shows promising results that are significantly better than previous approaches. Despite image datasets, we also show that our method works well on the text categorization dataset.

Pairwise pseudo labeling. Pairwise similarity between pairs of sample has been widely used in the literature for dimension reduction or clustering (e.g., t-SNE [37], FINCH [43]). Several methods have shown the effectiveness of using pairwise similarity to provide pseudo labels on-the-fly to train deep convolutional neural networks. In [24], a binary classifier is trained to provide pairwise pseudo labels to train a multi-class classifier. In [20], ranking statistics is used to obtain pairwise pseudo labels on-the-fly for the task of novel category discovery. In [43], the pairwise connection between data points by finding the nearest neighbour is used to cluster images using CNN features. In our method, we compute pairwise labels from a neural network embedding. This way we generate pseudo labels for each pair in each mini-batch and learn cluster assignment without any supervision.

3. Method

Our methods is divided into three stages: (i) self-supervised pre-training, (ii) pairwise connection and clustering, and (iii) data composition. We provide an overview of our pipeline in fig. 1. Our method processes each input data batch x in two steps, by extracting features $\mathbf{f} = \Phi(x) \in \mathbb{R}^{N \times D}$ by means of a neural network Φ, followed by estimating posterior class probabilities $\mathbf{p} = \Psi(\mathbf{f}) \in \mathbb{R}^{N \times K}$ by means of a linear layer Ψ and softmax non-linearity. We use the symbol $\mathbf{p}' = \Phi(\Psi(x'))$ to denote the class predictions for the same mini-batch x' with data augmentation (random transformations) applied to it. We use the letters D, K and N to denote the feature space dimension, the number of clusters and the mini-batch size. We now detail each component of LSD-C.
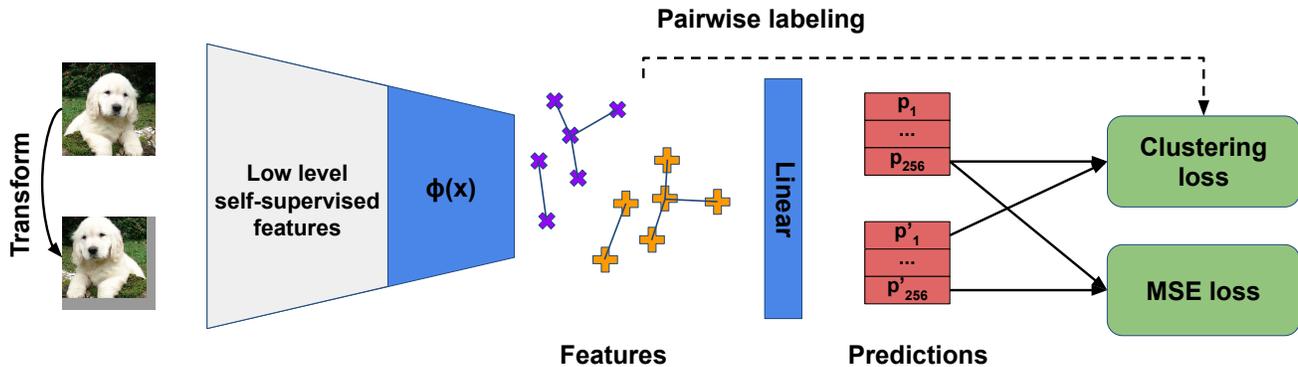
Figure 1: **Overview of LSD-C.** Pairwise labels are extracted at the feature level. They are then used in a clustering loss after the linear classifier. This way, the feature maps will evolve such that connected samples will be grouped in linearly separated clusters. The MSE loss acts a regularizer and enforces the consistency of the cluster predictions when data augmentation is applied.

3.1. Self-supervised pretraining

As noted in the introduction, traditional clustering methods require handcrafted or pretrained features. More recently, methods such as [27] have combined deep learning and clustering to learn features and clusters together; even so, these methods usually *still* require ad hoc pre-processing steps (e.g. pre-processing such as Sobel filtering [2, 27]) and extensive hyperparameter tuning. In our method we address this issue and avoid bad local minima in our clustering results by initializing our representation by means of self-supervised learning. In practice, this amounts to train our model on a pretext task (detailed in section 4) and then retain and freeze the earlier layers of the model when applying our clustering algorithm. As reported in [4, 16], the features obtained from self-supervised pre-training are linearly separable with respect to typical semantic image classes. This property is particularly desirable in our context and also motivates our major design choice: since the feature space of self-supervised pre-trained network is linearly separable, it is therefore easier to directly operate on it to discriminate between different clusters.

3.2. Pairwise labeling

A key idea in our method is the choice of space where pairwise the data connections are established: we extract pairwise labels at the level of the data representation rather than at the level of the class predictions. The latter is a common design choice, used in DAC [3] to establish pairwise connections between data points and in DEC [51] to match the current label posterior distribution to a sharper version of itself.

The collection of pairwise labels between samples in a mini-batch is given by the adjacency matrix A of an undirected graph whose nodes are the samples and whose edges encode their similarities. DAC [3] generates pseudo labels by checking if the output of the network is above or under certain thresholds. The method of [34] proceeds similarly in the semi-supervised setting. In our method, as we work instead at the feature space level, the pairwise labeling step is a separate process from class prediction and we are free to choose any similarity to establish our adjacency matrix A. We denote with $\mathbf{f}_i \in \mathbb{R}^D$ and $\mathbf{f}_j \in \mathbb{R}^D$ the feature vectors for samples i and j in a mini-batch, obtained from the penultimate layer of the neural network Φ. We also use the symbol $A_{ij} \in \{0, 1\}$ to denote the value of the adjacency matrix for the pair of samples (i, j). Next, we describe the different types of pairwise connections considered in this work and summarize them in table 1.

Cosine and L_2 similarity. Let $\tau \in \mathbb{R}^+$ be a threshold hyperparameter and define $C_{ij} = [\cos(\mathbf{f}_j, \mathbf{f}_i) > \tau]$ (cosine) or $C_{ij} = [\|\mathbf{f}_j - \mathbf{f}_i\|^2 < \tau]$ (Euclidean) where $\cos$ denotes the dot product between L_2-normalized vectors. We then define $A_{ij} = \mathbb{1}_{C_{ij}}$ where $\mathbb{1}$ is the indicator function. These definitions connect neighbor samples but do not account well for the local structure of the data. Indeed, it is not obvious that the cosine similarity or Euclidean distance would establish good data connections in feature space.

Symmetric SNE. A possible solution to alleviate the previous issue is to use the symmetric SNE similarity introduced in t-SNE [37]. This similarity is based on the conditional probability $p_{j|i}$ of picking j as neighbor of i under a Gaussian distribution assumption. We make a further assumption compared to [37] of an equal variance for every sample in order to speed up the computation of pairwise similarities

and define:

$$p_{j|i} = \frac{\exp(-\|\mathbf{f}_j - \mathbf{f}_i\|^2/T^2)}{\sum\limits_{k \neq i} \exp(-\|\mathbf{f}_k - \mathbf{f}_i\|^2/T^2)}$$

$$= \frac{\exp(-\|\mathbf{f}_j - \mathbf{f}_i\|^2/T^2)}{Z_i}, \tag{1}$$

$$C_{ij} = \frac{p_{j|i} + p_{i|j}}{2} > \tau$$

$$\Longleftrightarrow \tag{2}$$

$$\frac{\exp(-\|\mathbf{f}_j - \mathbf{f}_i\|^2/T^2)}{H(Z_i, Z_j)} > \tau.$$

As shown in equation (1), we introduce a temperature hyperparameter $T \in \mathcal{R}^+$ and we call Z_i the partition function for sample i. Then the associated adjacency matrix in equation (2) can be written as a function of the L_2 distance between samples and, in the denominator, of the harmonic mean H of the partition functions. As a result, if sample i or j has many close neighbours, it will reduce the symmetric SNE similarity and possibly prevent a connection between samples i and j. Such a phenomenon is shown on the two moons toy dataset in fig. 2.

k-nearest neighbors. We also propose a similarity based on k-nearest neighbours (kNN) [8] where the samples i and j are connected if i is in the k-nearest neighbours of j or if j is in the k-nearest neighbours of i. With this similarity, the hyperparameter is the minimum of neighbours k and not the threshold τ.

3.3. Clustering loss and data composition

Now that we have established pairwise connections between each pair of samples in the mini-batch, we will use the adjacency matrix as target for a binary cross-entropy loss. Denoting with $P(i = j)$ the probability that samples i and j belong to the same cluster, we wish to optimize the clustering loss:

$$L_{\text{clus}} = -\sum_{i,j} A_{ij} \log P(i = j)$$

$$+ (1 - A_{ij}) \log P(i \neq j). \tag{3}$$

The left term of this loss aims at maximizing the number of connected samples (i.e. $A_{ij} = 1$) within a cluster and the right term at minimizing the number of non-connected samples within it (namely, the edges of the complement of the similarity graph $1 - A_{ij} = 1$). Hence the second term prevents the formation of a single, large cluster that would contain all samples.

The next step is to model $P(i = j)$ by using the linear classifier predictions of samples i and j. As seen in equation

(4), for a fixed number of clusters K, the probability of samples i and j belonging to the same cluster can be rewritten as a sum of probabilities over the possible clusters. For simplicity, we assume that samples i and j are independent. This way, the pairwise comparison between samples appear only at the loss level and we can thus use the standard forward and backward passes of deep neural networks where each sample is treated independently. By plugging equation (4) in equation (3) and by replacing $\mathbf{p}_j$ with $\mathbf{p}'_j$ to form pairwise comparisons between the mini-batch and its augmented version, we obtain our final clustering loss $\mathcal{L}_{\text{clus}}$:

$$P(i = j) = \sum_{k=1}^{K} P(i = k, j = k)$$

$$= \sum_{k=1}^{K} P(i = k) P(j = k) \tag{4}$$

$$= \mathbf{p}_i^\top \mathbf{p}_j,$$

$$L_{\text{clus}} = -\sum_{i,j} A_{ij} \log(\mathbf{p}_i^\top \mathbf{p}'_j)$$

$$+ (1 - A_{ij}) \log(1 - \mathbf{p}_i^\top \mathbf{p}'_j). \tag{5}$$

A similar loss is used in [24] but with supervised pairwise labels to transfer a multi-class classifier across tasks. It is also reminiscent of DAC [3], but differs from the latter because the DAC loss does not contain a dot product between probability vectors but between L_2 normalized probability vectors. Hence DAC optimizes a Bhattacharyya distance whereas we optimize a standard binary cross-entropy loss.

In practice L_{clus} can be used in combination with effective data augmentation techniques such as RICAP [48] and MixUp [55]. These methods combine the images from the minibatch and use a weighted combination of the labels of the original images as new target for the cross-entropy loss. We denote with σ permutation of the samples in the minibatch; RICAP and MixUp require 4 and 2 permutations respectively. RICAP creates a new minibatch of composite images by patching together random crops from the 4 permutations of the original minibatch, whereas MixUp produces a new minibatch by taking a linear combination with random weights from 2 permutations. The new target for a composite image is then obtained by taking a linear combination of the labels in the recombined images, weighted by area proportions in RICAP and the mixing weights in MixUp. These techniques were proposed for the standard supervised classification setting, so we adapt them here to clustering. In order to do so, we propose to perform a pairwise labeling between the composite images and the raw original images. Both minibatches of original and composite images are fed to the network. Then, as illustrated in fig. 3, the pairwise label between a composite image and a raw image is the linear

Table 1: **Pairwise labeling with adjacency matrices** $A_{ij} = \mathbb{1}_{C_{ij}}$ **based on different similarities.** τ is the thresholding hyperparameter for L_2, SNE and Cosine. The number of neighbours k is kNN's hyperparameter.

L_2 dist.	SNE	Cosine	kNN
$C_{ij} = \|\mathbf{f}_j - \mathbf{f}_i\|^2 < \tau$	$\frac{\exp(-\|\mathbf{f}_j - \mathbf{f}_i\|^2/T^2)}{H(Z_i, Z_j)} > \tau$	$\frac{\mathbf{f}_j^\top \mathbf{f}_i}{\|\mathbf{f}_j\|\|\mathbf{f}_i\|} > \tau$	$(j \in \text{kNN}(i)) \vee (i \in \text{kNN}(j))$

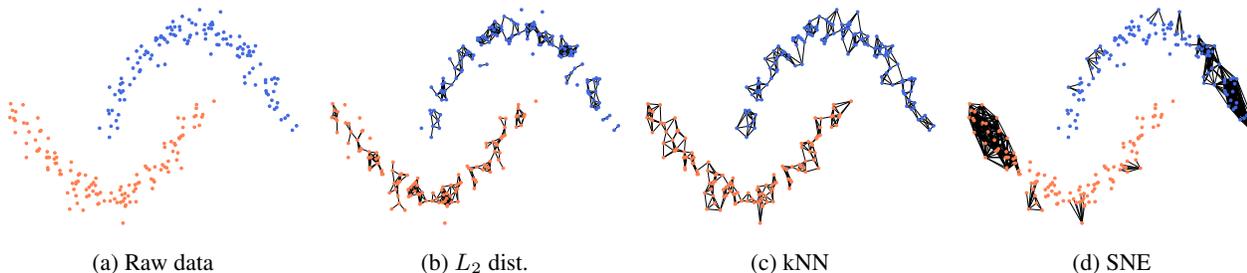(a) Raw data   (b) L_2 dist.   (c) kNN   (d) SNE

Figure 2: **Pairwise connections on the two moons toy data.** From left to right. We apply our algorithm with different connection techniques on a toy dataset shown in (a) where each color represents a class. We use the different connections techniques of table 1 such that there are 650 undirected edges for each similarity. Compared to L_2 distance and SNE, kNN produces neighbourhoods of similar sizes and every sample is connected. SNE captures the local structure of the data: most of the connections are at the external tails of the moons where there are less points.

combination of the pairwise labels between the components of both. To sum up, to obtain the pairwise labels between a minibatch and its composite version we just need to extract the adjacency matrix A of the minibatch and then do a linear combination of the adjacency matrix A with the different column permutations σ:

$$L_{\text{clus}} = -\sum_\sigma \sum_{i,j} w_\sigma A_{i\sigma(j)} \log(\mathbf{p}_i^\top \tilde{\mathbf{p}}_j)$$
$$+ (1 - w_\sigma A_{i\sigma(j)}) \log(1 - \mathbf{p}_i^\top \tilde{\mathbf{p}}_j) \quad (6)$$

Regarding the predicted probability of the 'pure' image i and the composite image j being in the same cluster, we take the dot product between their respective cluster predictions $\mathbf{p}_i$ and $\tilde{\mathbf{p}}_j$.

3.4. Overall loss

The overall loss we optimise is given by

$$L_{\text{tot}} = L_{\text{clus}}(\mathbf{f}, \mathbf{p}, \mathbf{p}') + L_{\text{cons}}(\mathbf{p}, \mathbf{p}'), \quad (7)$$

where

$$L_{\text{cons}} = \frac{\omega(t)}{KN} \sum_{i=1}^N \|\mathbf{p}_i - \mathbf{p}_i'\|^2, \quad (8)$$

and $\omega(t) = \lambda e^{-5(1 - \frac{t}{T})^2}$ is the ramp-up function proposed in [32, 49] with t the current training step, T the ramp-up length and $\lambda \in \mathbb{R}_+$. L_{cons} is a consistency constraint which requires the model to produce the same prediction $\mathbf{p} \approx \mathbf{p}'$ for an image and an its augmented version. We use it in

our method in a similar way as semi-supervised learning techniques [32, 39, 42, 49], i.e. as a regularizer to provide consistent predictions. This differs significantly from clustering methods like IIC [27] and IMSAT [25] where augmentations are used as a main clustering cue by maximizing the mutual information between different versions of an image. Instead, as commonly done in semi-supervised learning, we use the Mean Squared Error (MSE) between predictions as the consistency loss.

4. Experiments

Datasets. We conduct experiments on five popular benchmarks which we use to compare our method against recent approaches whenever results are available. We use four image datasets and one text dataset to illustrate the versatility of our approach to different types of data. We use MNIST [33], CIFAR 10 [30], CIFAR 100-20 [30] and STL 10 [6] as image datasets. All these datasets cover a wide range of image varieties ranging from 28×28 pixels grey scale digits in MNIST to 96×96 higher resolution images from STL 10. CIFAR 100-20 is redesigned from original CIFAR 100 since we consider only the 20 meta classes for evaluation as common practice [27]. Finally we also evaluate our method on a text dataset, Reuters 10K [35]. Reuters 10K contains 10,000 English news labelled with 4 classes. Each news has 2,000 *tf-idf* features. For all datasets we suppose the number of classes to be known.
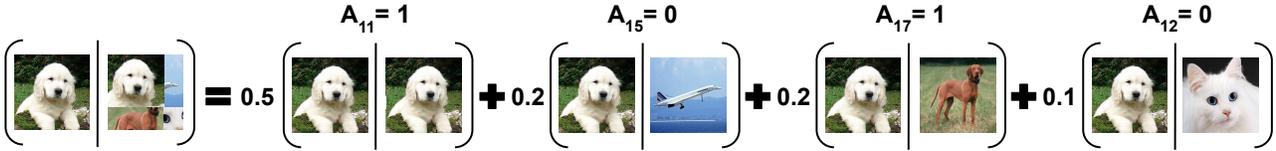
Figure 3: Illustration for eq. (6) of a pairwise target between the "pure" image $i = 1$ and the composite image j with $\sigma(j) \in (1, 5, 7, 2)$. In this case, the resulting pairwise target equals 0.7.

Experimental details. We use ResNet-18 [23] for all the datasets except two. For MNIST we use a model inspired from VGG-4 [45], described in [27] and for Reuters 10K we consider a simple DNN of dimension 2000–500–500–2000–4 described in [51]. We train with batch-size of 256 for all experiments. We use SGD optimizer with momentum [47] and weight decay set to 5×10^{-4} for every dataset except for Reuters 10K where we respectively use Adam [29] and decay of 2×10^{-3}. When comparing with other methods in table 2 and table 3, we run our method using 10 different seeds and report average and standard deviation on each dataset to measure the robustness of our method with respect to initialization. As it is common practice [27], we train and test the methods on the whole dataset (this is acceptable given that the method uses no supervision). Further experimental details about data augmentation and training are available in the appendix.

Evaluation metrics. We take the commonly used *clustering accuracy* (ACC) as evaluation metric. ACC is defined as

$$\max_{g \in \mathrm{Sym}(K)} \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\left\{\overline{y}_i = g\left(y_i\right)\right\}, \qquad (9)$$

where $\overline{y}_i$ and y_i respectively denote the ground-truth class label and the clustering assignment obtained by our method for each sample in the dataset. $\mathrm{Sym}(K)$ is the group of permutations with K elements and following other clustering methods we use the Hungarian algorithm [31] to optimize the choice of permutation.

4.1. Results on standard benchmarks

We compare our method with the K-means [38] baseline and recent clustering methods. In table 2, we report results on image datasets. We use RotNet [16] self-supervised pre-training for each dataset on all the data available (*e.g* including the unlabelled set in STL-10). Our method significantly outperforms the others by a large margin. For example, our method achieves 81.5% on CIFAR 10, while the previous state-of-the-art method IIC [27] gives 61.7%. On CIFAR 10, our method also outperforms the leading semi-supervised learning technique FixMatch [46] which obtains 64.3% in its one label per class setting. Similarly, on

CIFAR 100-20 and STL 10, our method outperforms other clustering approaches respectively by 14.7 and 6.8 points. On MNIST, our method and IIC both achieve a very low error rate around 1%.

These results clearly show the effectiveness of our approach. Unlike IIC that requires to apply Sobel filtering and very large batch size during training, our method does not require such preprocessing and works with a common batch size. We also note that our method is robust to different initialization, with a maximum 3.2% of standard deviation across all datasets.

To analyse further the results on CIFAR 10, we can look at the confusion matrix resulting from our model's predictions.We note that most of the errors are due to the 'cat' and 'dog' classes being confused. If we retain only the confident samples with prediction above 0.9 (around 60% of the samples), the accuracy rises to 94%. We assume that the two classes 'cat' and 'dog' are are more difficult to discriminate due to their visual similarity.

In table 3, we also evaluate our method on the document classification dataset Reuters 10K to show its versatility. We compare with different approaches than in table 2 as clustering methods developed for text are seldom evaluated on image datasets like CIFAR and vice versa. Following existing approaches applied to Reuters 10K, we pretrain the deep neural network by training a denoising autoencoder on the dataset [28]. Our method works notably better than the K-means baseline, and is on par with the best results methods FINCH [43] and VaDE [28]. Most notably one run of our method established state-of-the-art results of 83.5%, 2 points above the current best model.

4.2. Ablation studies

In order to analyze the effects of the different components of our method, we conduct a three parts ablation study on CIFAR 10 and CIFAR 100-20. First, we compare the impact of different possible pairwise labeling methods in the feature space. Second, as one of our key contribution is to choose the space where the pairwise labeling is performed, we test doing so at the level of features and predictions (*i.e.* after the linear classifier but before the softmax layer like DEC [51] or DAC [3]). Third, we analyse the importance of data augmentation in clustering raw images. Results are reported

Table 2: **Comparison with other methods.** Our method almost constantly reaches state-of-the-art performances by a large margin. Note that [27] report best results over all the heads while we report results over ten different initializations. This further shows that our method is overall stable and robust to initialization.

	K-means [38]	JULE [53]	DEC [51]	DAC [3]	IIC [27]	Ours
CIFAR 10	22.9	27.2	30.1	52.2	61.7	**81.7** $\pm$ 0.9
CIFAR 100-20	13.0	13.7	18.5	23.8	25.7	**42.3** $\pm$ 1.0
STL 10	19.2	27.7	35.9	47.0	59.6	**66.4** $\pm$ 3.2
MNIST	57.2	96.4	84.3	97.8	**99.2**	98.6 $\pm$ 0.5

Table 3: **Results on Reuters 10K.** Our method performs on average on par with state of the art. Note that for the best seed we reach state-of-the-art results of 83.5%.

	K-means [38]	IMSAT [25]	DEC [51]	VaDE [28]	FINCH [43]	Ours
Reuters 10K	52.4	71.9	72.2	79.8	**81.5**	79.0 $\pm$ 4.3

Table 4: **Ablation study.** We analyse the effect of different pairwise labeling methods but also the impact of where the labeling is done (feature vs prediction space). We also show the paramount importance of data augmentation for clustering some datasets like CIFAR 10.

	Pairwise labeling				Using the pred. space			Data augmentation		
	L_2	Cosine	kNN	SNE	Cosine	kNN	SNE	RICAP	MixUp	None
CIFAR 10	70.2	81.1	**81.7**	81.5	63.7	64.7	**67.0**	**81.7**	75.3	53.7
CIFAR 100-20	26.1	34.4	**42.3**	40.4	20.4	**32.8**	30.4	**42.3**	37.1	35.4

in table 4 and discussed next.

Pairwise similarity. We compare, in feature space, pairwise labeling methods based on L_2 distance, cosine similarity, kNN and symmetric SNE as described in table 1. For kNN, we set the number of neighbors k to 20 and 10 for CIFAR 10 and CIFAR 100-20 respectively. For the cosine similarity, we use respectively thresholds 0.9 and 0.95. For the L_2 distance, we ran a grid search between 0 and 2 to find an optimal threshold. For SNE, we set the threshold to 0.01 and the temperature to 1 and 0.5, for CIFAR 10 and CIFAR 100-20 respectively. Further details about the hyperparameters are available in the supplementary. We observe that kNN, SNE and cosine similarity perform very well on CIFAR 10 with values around 81%. It is interesting to note that cosine similarity performs noticeably worse than kNN and SNE on CIFAR 100-20 with around 6 points less. We also notice that L_2 distance performs consistently worse than the other labeling methods. We can conclude that kNN and SNE are the best labeling methods empirically with consistent performance on these two datasets.

Feature space embedding. Instead of using these labeling methods before the linear classifier, we apply them after it.

In this case, our overall approach becomes more similar to standard pseudo-labeling methods such as [3, 34, 51], which aim to match the network predictions output with a 'sharper' version of it. We observe that the performance drops considerably for all labeling methods with an average decrease of 16.3 points for CIFAR 10 and 10.6 points for CIFAR 100-20. Hence, this shows empirically that where pseudo labeling is applied plays a major role in clustering effectiveness and that labeling at the feature space level is noticeably better than doing so at the prediction space level.

Data augmentation. We compare RICAP, MixUp, and the case without data composition (denoted as None). As can it can be seen in table 4, data composition is crucial for CIFAR 10 where RICAP and MixUp surpass None by respectively 28 and 22 points. On CIFAR 100-20, the differences are smaller but using data composition still brings a clear improvement with a 5.1 points increase when using RICAP. Interestingly, RICAP clearly outperforms MixUp in both cases.

5. Conclusions

We have proposed a novel deep clustering method, LSD-C. Our method establishes pairwise connections at the fea-

ture space level among different data points in a mini-batch. These on-the-fly pairwise connections are then used as targets by our loss to regroup samples into clusters. In this way, our method can effectively learn feature representation together with the cluster assignment. In addition, we also combine recent self-supervised representation learning with our clustering approach to bootstrap the representation before clustering begins. Finally, we adapt data composition techniques to the pairwise connections setting, resulting in a very large performance boost. Our method substantially outperforms previous approaches in various public benchmarks, including CIFAR 10/100-20, STL 10, MNIST and Reuters 10K.

References

[1] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. In *Proc. ICLR*, 2019. 2

[2] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proc. ECCV*, 2018. 2, 3

[3] Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Deep adaptive image clustering. In *Proc. ICCV*, 2017. 2, 3, 4, 6, 7

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv*, 2020. 1, 2, 3

[5] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Proc. NIPS*, 2016. 2

[6] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011. 5

[7] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *PAMI*, 1979. 1, 2

[8] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 1967. 4

[9] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation with a discriminative loss function. *arXiv*, 2017. 2

[10] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, 1996. 1, 2

[11] Dario Fontanel, Fabio Cermelli, Massimiliano Mancini, Samuel Rota Bulò, Elisa Ricci, and Barbara Caputo. Boosting deep open world recognition by clustering. *arXiv*, 2020. 2

[12] Aude Genevay, Gabriel Dulac-Arnold, and Jean-Philippe Vert. Differentiable deep clustering with cluster size constraints. *arXiv*, 2019. 2

[13] Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *Proc. ICCV*, 2017. 2

[14] Spyros Gidaris, Andrei Bursuc, Nikos Komodakis, Patrick Pérez, and Matthieu Cord. Boosting few-shot visual learning with self-supervision. In *Proc. ICCV*, 2019. 1, 2

[15] Spyros Gidaris, Andrei Bursuc, Nikos Komodakis, Patrick Pérez, and Matthieu Cord. Learning representations by predicting bags of visual words. *arXiv*, 2020. 2

[16] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *Proc. ICLR*, 2018. 1, 2, 3, 6

[17] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. Improved deep embedded clustering with local structure preservation. In *IJCAI*, 2017. 2

[18] Xifeng Guo, Xinwang Liu, En Zhu, and Jianping Yin. Deep clustering with convolutional autoencoders. In *International conference on neural information processing*, 2017. 2

[19] Philip Haeusser, Johannes Plapp, Vladimir Golkov, Elie Aljalbout, and Daniel Cremers. Associative deep clustering: Training a classification network with no labels. In *German Conference on Pattern Recognition*, 2018. 2

[20] Kai Han, Sylvestre-Alvise Rebuffi, Sebastien Ehrhardt, Andrea Vedaldi, and Andrew Zisserman. Automatically discovering and learning new visual categories with ranking statistics. In *Proc. ICLR*, 2020. 2

[21] Kai Han, Andrea Vedaldi, and Andrew Zisserman. Learning to discover novel visual categories via deep transfer clustering. In *Proc. ICCV*, 2019. 2

[22] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv*, 2019. 1, 2

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016. 6

[24] Yen-Chang Hsu, Zhaoyang Lv, Joel Schlosser, Phillip Odom, and Zsolt Kira. Multi-class classification without multi-class labels. In *Proc. ICLR*, 2019. 2, 4

[25] Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. Learning discrete representations via information maximization self-augmented training. In *Proc. ICML*, 2017. 2, 5, 7

[26] Gabriel Huang, Hugo Larochelle, and Simon Lacoste-Julien. Centroid networks for few-shot clustering and unsupervised few-shot classification. *arXiv*, 2019. 2

[27] Xu Ji, João F Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *Proc. ICCV*, 2019. 1, 2, 3, 5, 6, 7

[28] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. *arXiv*, 2016. 1, 2, 6, 7

[29] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2014. 6

[30] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Technical report*, 2009. 5

[31] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 1955. 6

[32] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. In *Proc. ICLR*, 2017. 5

[33] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998. 5

[34] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, 2013. 3, 7

[35] David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 2004. 5

[36] Fengfu Li, Hong Qiao, and Bo Zhang. Discriminatively boosted image clustering with fully convolutional auto-encoders. *Pattern Recognition*, 83:161–173, 2018. 2

[37] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 2008. 2, 3

[38] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967. 1, 2, 6, 7

[39] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 5

[40] Karl Pearson. Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London Series A*, 1894. 1, 2

[41] Sylvestre-Alvise Rebuffi, Sebastien Ehrhardt, Kai Han, Andrea Vedaldi, and Andrew Zisserman. Semi-supervised learning with scarce annotations. In *Proc. CVPR Workshop*, 2020. 1, 2

[42] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in neural information processing systems*, 2016. 5

[43] Saquib Sarfraz, Vivek Sharma, and Rainer Stiefelhagen. Efficient parameter-free clustering using first neighbor relations. In *Proc. CVPR*, 2019. 1, 2, 6, 7

[44] Uri Shaham, Kelly Stanton, Henry Li, Boaz Nadler, Ronen Basri, and Yuval Kluger. Spectralnet: Spectral clustering using deep neural networks. In *Proc. ICLR*, 2018. 2

[45] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv*, 2014. 6

[46] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv*, 2020. 6

[47] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proc. ICML*, 2013. 6

[48] Ryo Takahashi, Takashi Matsubara, and Kuniaki Uehara. Ricap: Random image cropping and patching data augmentation for deep cnns. In *Asian Conference on Machine Learning*, 2018. 1, 4

[49] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Proc. NIPS*, 2017. 5

[50] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Scan: Learning to classify images without labels. In *arXiv preprint arXiv:2005.12320*, 2020. 2

[51] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *Proc. ICML*, 2016. 1, 2, 3, 6, 7

[52] Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *Proc. ICML*. JMLR. org, 2017. 2

[53] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *Proc. CVPR*, 2016. 2, 7

[54] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S4l: Self-supervised semi-supervised learning. In *Proc. ICCV*, 2019. 1, 2

[55] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv*, 2017. 1, 4

LSD-C: Linearly Separable Deep Clusters
–Supplementary Material–

Anonymous ICCV submission

Paper ID ****

In this supplementary material, we provide our implementation details, the confusion matrices on CIFAR 10 using our method with kNN labeling and some additional ablation studies. We also include the code to run our method on CIFAR 10 together with the network pretrained with RotNet [3].

1. Implementation details

Self-supervised pretraining. We train the RotNet [3] (i.e. predicting the rotation applied to the image among four possibilities: $0°$, $90°$, $180°$, and $270°$) on all datasets with the same configuration. Following the authors' released code, we train for 200 epochs using a step-wise learning rate starting at 0.1 which is then divided by 5 at epochs 60, 120, and 160.

Main LSD-C models. After the self-supervised pretraining step, following [4] we freeze the first three macro-blocks of the ResNet-18 [5] as the RotNet training provides robust early filters. We then train the last macro-block and the linear classifier using our clustering method. For all the experiments, we use a batch size of 256. We summarize in table 1 all the hyperparameters for the different datasets and labeling methods.

Table 1: **Hyperparameters.** Optimizer, ramp-up function and parameters of different labeling methods on different datasets.

	Optimizer				Ramp-up		Cosine	SNE		kNN
	Type	Epochs	LR steps	LR init	λ	T	τ	τ	Temp	k
CIFAR 10	SGD	220	[140, 180]	0.1	5	100	0.9	0.01	1.0	20
CIFAR 100-20	SGD	200	170	0.1	25	150	0.95	0.01	0.5	10
STL 10	SGD	200	[140, 180]	0.1	5	50	-	0.01	0.5	-
MNIST	SGD	15	-	0.1	5	50	-	-	-	10
Reuters 10K	Adam	75	-	0.001	25	100	-	-	-	5

Data augmentation techniques. We showed in the main paper that data composition techniques like RICAP [8] and MixUp [9] are highly beneficial to our method. For RICAP, we follow the authors' instructions to sample the width and height of crops for each minibatch permutation by using a Beta(0.3, 0.3) distribution. Regarding MixUp, we note that using a Beta(0.3, 0.3) distribution for the mixing weight works better in our case than the Beta(1.0, 1.0) advised for CIFAR 10 in the MixUp paper. Furthermore, we have to decrease the weight decay to 10^{-4} to make MixUp work.

Miscellaneous. Our method is implemented with PyTorch 1.2.0 [7]. Our experiments were run on NVIDIA Tesla M40 GPUs and can run on a single GPU with 12 GB of RAM. Inference time for CIFAR10/CIFAR100-20/STL10 are 10.647s/10.453s/5.517s on a single GeForce GTX 1080 Ti GPU.

2. Confusion matrices on CIFAR 10

In fig. 1, we show some confusion matrices on CIFAR 10 to analyse how our clustering method performs on the different classes. We notice that there are 8 confident clusters with a very high clustering accuracy of 94.0% for confident samples. The "dog" and "cat" clusters are not well identified possibly due to a huge intra-class variation of the samples.
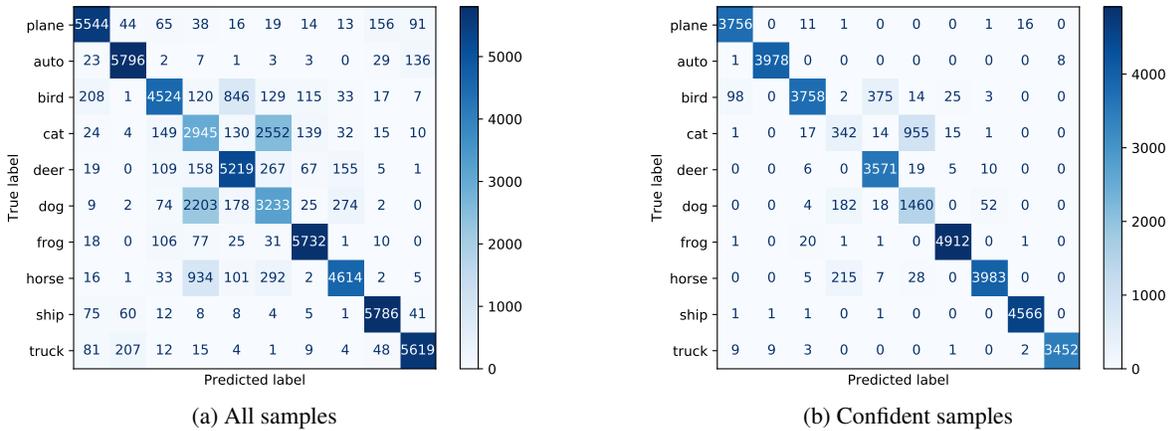
(a) All samples

(b) Confident samples

Figure 1: **Confusion matrices on CIFAR 10 using our method with kNN labeling.** Figure 1a shows that most of the errors are due to the "cat" and "dog" classes. When taking the samples with prediction above 0.9 (60% of the samples) in Figure 1b, there are less than 2000 predictions on classes "cat" and "dog" whereas there are more than 3500 for each of the other classes. Our method manages to ignore the problematic classes when taking the confident samples. Indeed, the accuracy for confident samples is 94.0%.

3. Additional experiments

In this section we report results of additional experiments we carried out. In table 2 we evaluate the impact of more components of our method. For example, we apply K-means [6] on the feature space of the pretrained RotNet model and we note very poor performance on CIFAR 10 and CIFAR 100-20. We can conclude that before training with our clustering loss, the desired clusters are not yet separated in the feature space. After training with our clustering loss, the clusters can be successfully separated. Moreover, if we only use the clustering loss and drop the consistency MSE loss, the performance decreases on both CIFAR 10 and CIFAR 100-20 by 1.5 and 1.3 points respectively, showing that the MSE provides a moderate but clear gain to our method. Finally, if we replace the linear classifier by a 2-layer classifier (i.e. this corresponds to a non-linear separation of clusters in the feature space), it results in a small improvement on CIFAR 10 but a clear decrease of 1.9 points on CIFAR 100-20. Hence using a linear classifier provides more consistent results across datasets.

Table 2: **Impact of the different losses.** From the first column, we observe that the desired clusters are not yet separated in the feature space after the RotNet pretraining. The second column shows that the MSE consistency loss provides a boost of more than 1 point to our method. Finally, we see that using a non-linear classifier harms the performance on CIFAR 100-20. All methods were trained for one seed only.

	K-means + RotNet	Ours (kNN)	Ours (kNN) w/o MSE	Ours (kNN) w/ non-lin.
CIFAR 10	14.3	81.7	80.2	82.0
CIFAR 100-20	9.1	40.5	39.2	38.6

Since we only trained the last macro block and the linear layer of a ResNet-18[5] we test the impact of increasing or decreasing the number of trainable weights in table 3. Unsurprisingly, training only the last linear layer reaches about the accuracy of k-means. However we still get reasonable performances by training from the second macro block.

We also study the impact of the self-supervised pretraining method in table 4. We note that our method has less impact when combined with self-supervised training method that are based on contrastive loss. However we note that the data augmentation we used was not the one proposed in the original paper. Hence there might be opportunities for improvements by tweaking hyper parameters or changing the data augmentation.

Finally since our method uses similarity between elements of a batch, we measure the impact of varying batch size in table 5. We observe different trends in CIFAR-10 and CIFAR100-20. For the former bigger batch sizes are beneficial while this is the opposite for CIFAR100-20. For batch size of 256 the results are stable however. The divergence is probably caused by a higher diversity of data in CIFAR100-20 which results in an unstable training signal.

108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161

162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215

Table 3: **Impact of number of trainable weights.** We test the impact of training the network with our clustering method (SNE) starting from different macro block of ResNet. We report results over two seeds (except #3 which uses 10 seeds). For fairness, we use the same hyper-parameters for each row.

From macro block	#2	#3	#4 (linear layer)
CIFAR 10	53.6±1.4	81.7±0.9	14.7±0.0
CIFAR 100-20	30.6±0.4	42.3±1.0	10.8±0.1

Table 4: **Impact of self supervised method.** We test the impact of using different self-supervised techniques on CIFAR-10 for one seed (except RotNet which used 10 seeds). All self-supervised method are trained with the same data augmentation. For fairness, all clustering methods use the same hyper-parameters for each row. We trained parameters starting from the third macro block of ResNet-18.

	RotNet[3]	SimCLR[1]	MOCO v2[2]
K-means[6]	14.3	59.45	45.48
Ours (SNE)	81.7	43.5	23.7

Table 5: **Impact of batch size.** We test the impact of training with different batch size on our clustering method (SNE). We report results over two seeds (except 256 which used 10 seeds). For fairness, we use the same hyper-parameters for each row.

Batch-size	64	128	256	512
CIFAR 10	72.4±1.3	76.9±1.5	81.7±0.9	80.9±1.2
CIFAR 100-20	46.2±0.2	42.2±0.0	42.3±1.0	31.9±1.0

References

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv*, 2020. 3

[2] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 3

[3] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *Proc. ICLR*, 2018. 1, 3

[4] Kai Han, Sylvestre-Alvise Rebuffi, Sebastien Ehrhardt, Andrea Vedaldi, and Andrew Zisserman. Automatically discovering and learning new visual categories with ranking statistics. In *Proc. ICLR*, 2020. 1

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016. 1, 2

[6] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967. 2, 3

[7] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Proc. NIPS*, 2019. 1

[8] Ryo Takahashi, Takashi Matsubara, and Kuniaki Uehara. Ricap: Random image cropping and patching data augmentation for deep cnns. In *Asian Conference on Machine Learning*, 2018. 1

[9] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv*, 2017. 1