

# Learning Neural Models for End-to-End Clustering

Benjamin Bruno Meier<sup>1,2</sup>, Ismail Elezi<sup>1,3</sup>, Mohammadreza Amirian<sup>1,4</sup>,  
Oliver Dürr<sup>1,5</sup>, and Thilo Stadelmann<sup>1</sup>

<sup>1</sup> ZHAW Datalab & School of Engineering, Winterthur, Switzerland

<sup>2</sup> ARGUS Data Insights, Zurich, Switzerland

<sup>3</sup> Ca' Foscari University of Venice, Venice, Italy

<sup>4</sup> Institute of Neural Information Processing, Ulm University, Germany

<sup>5</sup> Institute for Optical Systems, HTWG Konstanz, Germany

**Abstract.** We propose a novel end-to-end neural network architecture that, once trained, directly outputs a probabilistic clustering of a batch of input examples in one pass. It estimates a distribution over the number of clusters and, for each number of clusters up to a maximum, distributions over the respective data partitioning. The neural network is trained in a supervised fashion to group data by any perceptual similarity criterion based on pairwise labels (same/different group). It does not expect to have seen any of the groups that appear during model application already during training. We demonstrate promising performance on high-dimensional data like images (COIL-100) and speech (TIMIT). We call this “learning to cluster” and show its conceptual difference from deep metric learning, semi-supervised clustering and other related approaches while having the advantage of performing learnable clustering fully end-to-end.

**Keywords:** speaker clustering · image clustering · deep neural network

## 1 Introduction

Consider the illustrative task of grouping images of cats and dogs by *perceived* similarity: depending on the intention of the user behind the task, the similarity could be defined by animal type (foreground object), environmental nativeness (background landscape, cp. Fig. 1) etc. This is a characteristic of perceptual, high-dimensional data like images [14] or sound [22]: a user typically has some similarity criterion in mind when thinking about naturally arising groups (e.g., pictures by holiday destination, or persons appearing; songs by mood, or use of solo instrument). As defining such a similarity for every case is difficult, it is desirable to learn it. At the same time, the learned model will in many cases not be a classifier—the task will not be solved by classification—since at least the number of specifically arising groups at application time are not known in advance (e.g., speakers in TV recordings; persons in front of a surveillance camera; object types in the picture gallery of a large web shop).



Fig. 1: Images of cats (top) and dogs (bottom) in urban (left) and natural (right) environments.

Grouping objects with machine learning is usually approached with clustering algorithms [15]. Typical representatives like K-means [23], EM [8], hierarchical clustering [27], or DBSCAN [9] each have a specific inductive bias towards certain similarity structures present in the data (e.g., K-means: Euclidean distance from a central point; DBSCAN: common point density). Hence, to be applicable to above-mentioned tasks, they need to rely on high-level features that already encode the aspired similarity measure. This is usually solved by learning salient embeddings [26] with a deep metric learning approach [13], followed by an off-line clustering phase using one of the above-mentioned algorithm.

However, it is desirable to combine these distinct phases (learning salient features, and subsequent clustering) into an end-to-end approach that can be trained globally [18]: it has the advantage of each phase being perfectly adjusted to the other by optimizing a global criterion, and removes the need of manually fitting parts of the pipeline. Numerous examples have demonstrated the success of neural networks for end-to-end approaches on such diverse tasks as speech recognition [2], robot control [19], scene text recognition [32], or music transcription [33].

In this paper, we present a conceptually novel approach that we call “*learning to cluster*” in the above-mentioned sense of grouping high-dimensional data by some perceptually motivated similarity criterion. For this purpose, we define a novel neural network architecture with the following properties: (a) during training, it receives pairs of similar or dissimilar examples to learn the intended similarity function, implicitly or explicitly; (b) during application, it is able to group objects of groups never encountered before; (c) it is trained end-to-end in a supervised way to produce a tailor-made clustering model and (d) is applied like a clustering algorithm to find both the number of clusters as well as the cluster membership of test-time objects in a fully probabilistic way.

Our approach builds upon ideas of *deep metric embedding*, namely to learn an embedding of the data into a representational space that allows for specific perceptual similarity evaluation via simple distance computation on feature vectors. However, it goes beyond this by adding the actual clustering step—grouping by similarity—directly to the same model, making it trainable end-to-end. Our approach is also different from *semi-supervised clustering* [4], which uses labels for some of the data points in the inference phase to guide the creation of groups. In contrast, our method uses absolutely no labels during inference, and moreover doesn’t expect to have seen any of the groups it encounters during

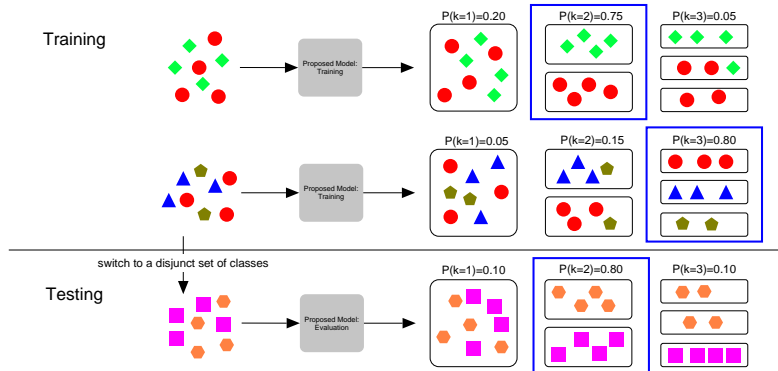


Fig. 2: Training vs. testing: cluster types encountered during application/inference are never seen in training. On the right hand side of the arrows we show exemplary outputs: for each possible cluster count (1–3 in this example) there is a partition and a corresponding probability (the best results are highlighted blue).

inference already during training (think e.g. of a model trained to cluster portraits by face appearance: our model learns to cluster previously unseen faces by being trained on a different face data set, cp. Fig. 2). Its training stage is to some degree conceptually similar to the original process to create K-means, DBSCAN etc.: it creates a specific clustering model, applicable to data with certain similarity structure, and once created/trained, the model performs “unsupervised learning” in the sense of finding groups. Finally, our approach differs from traditional *cluster analysis* [15] in how the clustering algorithm is applied: instead of looking for patterns in the data in an unbiased and exploratory way, as is typically the case in unsupervised learning, our approach is geared towards the use case where users know perceptually very well what they are looking for, and can make this explicit using examples. We then learn appropriate features and the similarity function simultaneously, taking full advantage of end-to-end learning.

Our main contribution in this paper is the creation of a neural network architecture that learns to *group* data, i.e., that outputs the same “label” for “similar” objects regardless of (a) it has ever seen this group before; (b) regardless of the actual value of the label (it is hence not a “class”); and (c) regardless of the number of groups it will encounter during a single application run, up to a predefined maximum. To the best of our knowledge, this approach is novel in its generality (i.e., applicable to previously unseen groups/clusters; applicable to high-dimensional data with only perceptually defined similarity). Additionally, we experimentally show its ability to implicitly learn a metric for the data at hand in a novel way. In Sec. 2, we compare our approach to related work, before presenting the model and training procedure in detail in Sec. 3. We evaluate our approach on different datasets in Sec 4, showing promising performance and a high degree of generality for data types ranging from 2D points to audio snippets and images, and discuss these results with conclusions for future work in Sec. 5.

## 2 Related Work

Learning to cluster based on neural networks has been approached mostly as a supervised learning problem to extract embeddings for a subsequent off-line clustering phase. The core of all deep metric embedding models is the choice of the loss function. Motivated by the fact that the softmax-cross entropy loss function has been designed as a classification loss and is not suitable for the clustering problem per se, *Chopra et al.* [7] developed a “Siamese” architecture, where the loss function is optimized in a way to generate similar features for objects belonging to the same class, and dissimilar features for objects belonging to different classes. A closely related loss function called “triplet loss” has been used by *Schroff et al.* [30] to get state-of-the-art accuracy in face detection. The main difference from the Siamese architecture is that while in the former the network is fed roughly half of the time with pairs from the same class, and half of the time with pairs from a different classes, in the latter case the network is always fed triplets containing an anchor object, an additional object of the same class, and an object of a different class. The network is then optimized to jointly learn their feature representation in a way that objects of the same class have similar features, while objects of different classes have different features. A problem of both approaches is that they are typically difficult to train compared to networks which utilize the cross entropy loss function.

*Song et al.* [35] developed an algorithm for taking full advantage of all the information available in training batches. They later refined the work [34] by proposing a new metric learning scheme based on structured prediction, which is designed to optimize a clustering quality metric (normalized mutual information [25]). Even better results were achieved by *Wong et al.* [36], where the authors proposed a novel angular loss. It takes angle relationships into account in order to learn a better similarity metric, and achieves state-of-the-art results on the challenging real-world datasets *Stanford Cars* [16] and *Caltech Birds* [5]. On the other hand, *Lukic et al.* [21] showed that for certain problems, a carefully chosen deep neural network can simply be trained with softmax-cross entropy loss and still achieve state-of-the-art performance in challenging problems like speaker clustering. Alternatively, *Wu et al.* [24] showed that state-of-the-art results can be achieved simply by using a traditional margin loss function and being careful on how sampling is performed during the creation of mini-batches.

On the other hand, attempts have been made recently that are more similar to ours in spirit, using deep neural networks only and performing clustering end-to-end. They are trained in a fully unsupervised fashion, hence solve a different task than the one we motivated above (that is inspired by speaker- or image clustering based on some human notion of similarity). Perhaps the first attempt to group objects together in an unsupervised deep learning based manner was performed in the seminal paper of *Le et al.* [17] where the authors were able to detect high-level concepts like cats or humans. *Xie et al.* [38] used an autoencoder architecture to do clustering, but experimental results were mostly done on simplistic datasets like *MNIST* with only a few different groups/clusters. Soon, CNN-based approaches followed, e.g. by *Yang et al.* [40], where clustering

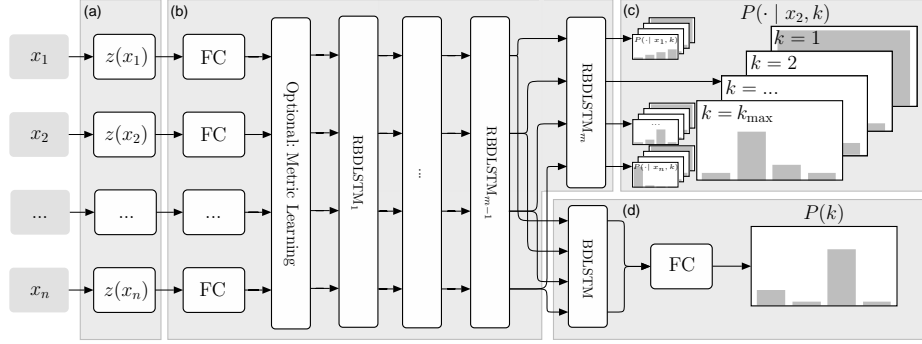


Fig. 3: Our complete model, consisting of (a) the embedding network, (b) clustering network (including an optional metric learning part, see Sec. 3.3), (c) cluster-assignment network and (d) cluster-count estimating network.

and feature representation are optimized together. *Greff et al.* [11] performed perceptual grouping (of pixels within an image into the objects constituting the complete image, hence a different task than ours) fully unsupervised using a neural expectation maximization algorithm. Our work differs to above-mentioned works in several respects: it has no assumption on the type of data we want to cluster, and solves the different problem of grouping objects in a perceptual way instead of doing a data driven cluster analysis. An up-to-date survey of other deep learning-based clustering approaches is given by *Aljalbout et al.*[1].

### 3 A model for end-to-end clustering of arbitrary examples

Our method learns to cluster end-to-end purely ab initio, without the need to explicitly specify a notion of similarity, only providing the information whether two examples belong together. It uses as input  $n \geq 2$  examples  $x_i$ , where  $n$  may be different during training and evaluation and constitutes the number of objects that can be clustered at a time, i.e. the maximal number of objects that can be grouped together. The network’s output is two-fold: a distribution  $P(k)$  over the cluster count  $k$ ; and a distribution  $P(\cdot | x_i, k)$  over the cluster index for each input example  $x_i$  and for each  $k$ .

#### 3.1 Network architecture

The network architecture (see Fig. 3) allows the flexible use of different input types, e.g. images, audio or 2D points. An input  $x_i$  is first processed by an embedding network (a) that produces a lower-dimensional representation  $z_i = z(x_i)$ . The dimension of  $z_i$  may vary depending on the used data type. For example, 2D points do not require any embedding network. A fully connected layer (FC) at the

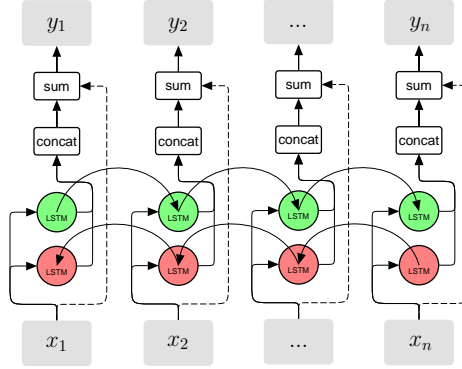


Fig. 4: RBDLSTM-layer: A BDLSTM with residual connections (dashed lines).

beginning of the clustering network (b) with 288 units and LeakyReLU activation ( $\alpha = 0.3$ ) is then used to bring all embeddings to the same size. This approach allows to use the identical subnetworks (b)–(d) and only change the subnetwork (a) for any data type. The goal of the subnetwork (b) is to compare each input  $z(x_i)$  with all other  $z(x_{j \neq i})$ , in order to learn an abstract grouping which is then concretized into an estimation of the number of clusters (subnetwork (d)) and a cluster assignment (subnetwork (c)).

To be able to process an arbitrary number of examples  $n$  as input, we use a recurrent neural network. Specifically, we use  $m = 14$  stacked residual bi-directional LSTM-layers (RBDLSTM), which are similar to the cells described in [37] and visualized in Fig. 4. The residual connections allow a much more effective gradient flow during the training [12] and avoid vanishing gradients. Additionally, the network can learn to use or bypass certain layers using the residual connections, thus reducing the architectural decision on the number of recurrent layers to the simpler one of finding a reasonable upper bound.

The first of overall two outputs is modeled by the cluster assignment network (c). It contains a softmax-layer to produce  $P(\ell \mid x_i, k)$ , which assigns a cluster index  $\ell$  to each input  $x_i$ , given  $k$  clusters (i.e., we get a distribution over possible cluster assignments for each input and every possible number of clusters). The second output, produced by the cluster-count estimating network (d), is built from one BDLSTM-layer with 128 units. Due to the bi-directionality of the network, we concatenate its first and the last output vector into a fully connected layer (256 units) using again LeakyReLUs ( $\alpha = 0.3$ ). The subsequent softmax-activation finally models the distribution  $P(k)$  for  $1 \leq k \leq k_{\max}$ .

### 3.2 Training and loss

In order to define a suitable loss-function, we first approximate (assuming independence) the probability that  $x_i$  and  $x_j$  are assigned to the same cluster for a

given  $k$  as

$$P_{ij}(k) = \sum_{\ell=1}^k P(\ell | x_i, k) P(\ell | x_j, k).$$

By marginalizing over  $k$ , we obtain  $P_{ij}$ , the probability that  $x_i$  and  $x_j$  belong to the same cluster:

$$P_{ij} = \sum_{k=1}^{k_{\max}} P(k) \sum_{\ell=1}^k P(\ell | x_i, k) P(\ell | x_j, k).$$

Let  $y_{ij} = 1$  if  $x_i$  and  $x_j$  are from the same cluster (e.g., have the same group label) and 0 otherwise. The loss component for *cluster assignments*,  $L_{ca}$ , is then given by the weighted binary cross entropy as

$$L_{ca} = \frac{-2}{n(n-1)} \sum_{i < j} (\varphi_1 y_{ij} \log(P_{ij}) + \varphi_2 (1 - y_{ij}) \log(1 - P_{ij}))$$

with weights  $\varphi_1$  and  $\varphi_2$ . The idea behind the weighting is to account for the imbalance in the data due to there being more dissimilar than similar pairs  $(x_i, x_j)$  as the number of clusters in the mini batch exceeds 2. Hence, the weighting is computed using  $\varphi_1 = c\sqrt{1-\varphi}$  and  $\varphi_2 = c\sqrt{\varphi}$ , with  $\varphi$  being the expected value of  $y_{ij}$  (i.e., the a priori probability of any two samples in a mini batch coming from the same cluster), and  $c$  a normalization factor so that  $\varphi_1 + \varphi_2 = 2$ . The value  $\varphi$  is computed over all possible cluster counts for a fixed input example count  $n$ , as during the training, the cluster count is randomly chosen for each mini batch according to a uniform distribution. The weighting of the cross entropy given by  $\varphi$  is then used to make sure that the network does not converge to a sub-optimal and trivial minimum. Intuitively, we thus account for permutations in the sequence of examples by checking rather for pairwise correctness (probability of same/different cluster) than specific indices.

The second loss term,  $L_{cc}$ , penalizes a wrong *number of clusters* and is given by the categorical cross entropy of  $P(k)$  for the true number of clusters  $k$  in the current mini batch:

$$L_{cc} = -\log(P(k)).$$

The complete loss is given by  $L_{\text{tot}} = L_{cc} + \lambda L_{ca}$ . During training, we prepare each mini batch with  $N$  sets of  $n$  input examples, each set with  $k = 1 \dots k_{\max}$  clusters chosen uniformly. Note that this training procedure requires only the knowledge of  $y_{ij}$  and is thus also possible for weakly labeled data. To show that the network really learns an intra-class distance and not just classifies objects of a fixed set of classes, the network is applied on totally different clusters at evaluation time than used in the training.

### 3.3 Implicit vs. explicit distance learning

To elucidate the importance and validity of the implicit learning of distances in our subnetwork (b), we also provide a modified version of our network architecture for comparison, in which the calculation of the distances is done explicitly. Therefore, we add an extra component to the network before the RBDLSTM layers, as can be seen in Figure 3: the optional metric learning block receives the fixed-size embeddings from the fully connected layer after the embedding network (a) as input and outputs the pairwise distances of the data points. The recurrent layers in block (b) then subsequently cluster the data points based on this pairwise distance information [6,3] provided by the metric learning block.

We construct a novel metric learning block inspired by the work of *Xing et al.* [39]. In contrast to their work, we optimize it end-to-end with backpropagation. This has been proposed in [31] for classification alone; we do it here for a clustering task, for the whole covariance matrix, and jointly with the rest of our network. We construct the non-symmetric, non-negative dissimilarity measure  $d_A^2$  between two data points  $x_i$  and  $x_j$  as

$$d_A^2(x_i, x_j) = (x_i - x_j)A(x_i - x_j)$$

and let the neural network training optimize  $A$  through  $L_{\text{tot}}$  without intermediate losses. The matrix  $A$  as used in  $d_A^2$  can be thought of as a trainable distance metric. In every training step, it is projected into the space of positive-semi definite matrices.

## 4 Experimental results

To assess the quality of our model, we perform clustering on three different datasets: For a proof of concept, we test on a set of generated *2D points* with a high variety of shapes, coming from different distributions. For speaker clustering, we use the *TIMIT* [10] corpus, a dataset of studio-quality speech recordings frequently used for pure speaker clustering in related work. For image clustering, we test on the *COIL-100* [28] dataset, a collection of different isolated objects in various orientations. To compare to related work, we measure the performance with the standard evaluation scores misclassification rate (MR) [20] and normalized mutual information (NMI) [25]. The used code and more details for the experiments are available online<sup>1</sup>.

We set  $k_{\text{max}} = 5$  and  $\lambda = 5$ . For the 2D point data, we use  $n = 72$  network inputs and a batch-size of  $N = 200$  (We used the batch size of  $N = 50$  for metric learning with 2D points). For TIMIT, the network input consists of  $n = 20$  audio snippets with a length of 1.28 seconds, encoded as mel-spectrograms with  $128 \times 128$  pixels (identical to [22]). For COIL-100, we use  $n = 20$  inputs with a dimension of  $128 \times 128 \times 3$ . For TIMIT and COIL-100, a simple CNN with 3 conv/max-pooling layers is used as subnetwork (a). For TIMIT, we use 430 of

<sup>1</sup> <https://github.com/kutoga/learning2cluster>



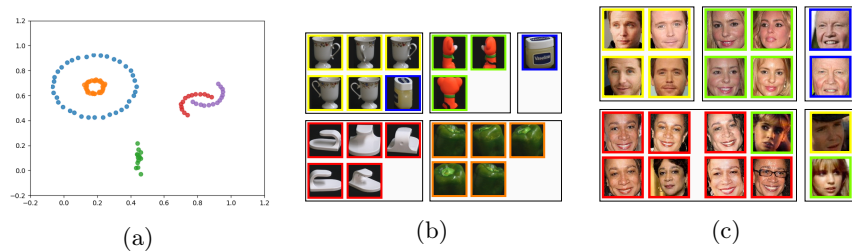


Fig. 5: Clustering results for (a) 2D point data, (b) COIL-100 objects, and (c) faces from FaceScrub (for illustrative purposes). The color of points / colored borders of images depict true cluster membership.

Table 1:  $\text{NMI} \in [0, 1]$  and  $\text{MR} \in [0, 1]$  averaged over 300 evaluations of a trained network. We abbreviate our “learning to cluster” method as “L2C”.

	<b>2D Points (self generated)</b>		<b>TIMIT</b>		<b>COIL-100</b>	
	<b>MR</b>	<b>NMI</b>	<b>MR</b>	<b>NMI</b>	<b>MR</b>	<b>NMI</b>
L2C (=our method)	0.004	0.993	0.060	0.928	0.116	0.867
L2C + Euclidean	0.177	0.730	0.093	0.883	0.123	0.884
L2C + Mahalanobis	0.185	0.725	0.104	0.882	0.093	0.890
L2C + Metric Learning	0.165	0.740	0.101	0.880	0.100	0.880
Random cluster assignment	0.485	0.232	0.435	0.346	0.435	0.346
Baselines (related work)	k-Means: MR = 0.178, NMI = 0.796 DBSCAN: MR = 0.265, NMI = 0.676		[22]: MR = 0		[41]: NMI = 0.985	

the 630 available speakers for training (and 100 of the remaining ones each for validation and evaluation). For COIL-100, we train on 80 of the 100 classes (10 for validation, 10 for evaluation). For all runs, we optimize using Adadelta [42] with a learning rate of 5.0. Example clusterings are shown in Fig. 5.

The results on 2D data as presented in Fig. 5a demonstrate that our method is able to learn specific and diverse characteristics of intuitive groupings. This is superior to any single traditional method, which only detects a certain class of cluster structure (e.g., defined by distance from a central point). Although [22] reach moderately better scores for the speaker clustering task and [41] reach a superior NMI for COIL-100, our method finds reasonable clusterings, is more flexible through end-to-end training and is not tuned to a specific kind of data. Tab. 1 gives the numerical results for said datasets in the row called “L2C” without using the explicit metric learning block. Extensive preliminary experiments on other public datasets like e.g. FaceScrub [29] confirm these results: learning to cluster reaches promising performance while not yet being on par with tailor-made state-of-the-art approaches.

We compare the performance of our implicit distance metric learning method to versions enhanced by different explicit schemes for pairwise similarity computation prior to clustering. Specifically, three implementations of the optional metric learning block in subnetwork (b) are evaluated: using a fixed diagonal matrix  $A$  (resembling the Euclidean distance), training a diagonal  $A$  (resembling Mahalanobis distance), and learning the entire coefficients of the distance matrix

A. Since we argue above that our approach combines *implicit* deep metric embedding with clustering in an end-to-end architecture, one would not expect that adding *explicit* metric computation changes the results by a large extent. This assumption is largely confirmed by the results in the “L2C+...” rows in Tab. 1: for COIL-100, Euclidean gives slightly worse, and the other two slightly better results than L2C alone; for TIMIT, all results are worse but still reasonable. We attribute the considerable performance drop on 2D points using all three explicit schemes to the fact that in this case much more instances are to be compared with each other (as each instance is smaller than e.g. an image,  $n$  is larger). This might have needed further adaptations like e.g. larger batch sizes (reduced here to  $N = 50$  for computational reasons) and longer training times.

## 5 Discussion and conclusions

We have presented a novel approach to learn neural models that directly output a probabilistic clustering on previously unseen groups of data; this includes a solution to the problem of outputting similar but unspecific “labels” for similar objects of unseen “classes”. A trained model is able to cluster different data types with promising results. This is a complete end-to-end approach to clustering that learns both the relevant features and the “algorithm” by which to produce the clustering itself. It outputs cluster membership of all inputs as well as the number of clusters in the data. The learning phase only requires pairwise labels between examples, and no explicit similarity measure needs to be provided. This is especially useful for high-dimensional, perceptual data like images and audio, where similarity is usually semantically defined by humans. Our experiments confirm that our algorithm is able to implicitly learn a metric and directly use it for the included clustering.

We observe that the clustering accuracy depends on the availability of a large number of different classes during training. We attribute this to the fact that the network needs to learn intra-class distances, a task inherently more difficult than just to distinguish between objects of a fixed amount of classes like in classification problems. We understand the presented work as a first investigation into a new paradigm of learning to cluster by perceptual similarity specified through examples. It is inspired by our work on speaker clustering with deep neural networks, where we increasingly observe the need to go beyond surrogate tasks for learning, training end-to-end specifically for clustering to close a performance leak. While this works satisfactory for initial results, points for improvement revolve around scaling the approach to practical applicability:

The number  $n$  of input examples to assess simultaneously is very relevant in practice: if an input data set has thousands of examples, incoherent single clusterings of subsets of  $n$  points would be required to be merged to produce a clustering of the whole dataset. As the (RBD)LSTM layers responsible for assessing points simultaneously in principle have a long, but still local (short-term) horizon, they are not apt to grasp similarities of thousands of objects. Several ideas exist to change the architecture, including to replace recurrent layers with

temporal convolutions, or using our approach to seed some sort of differentiable K-means or EM layer on top of it. Preliminary results on this exist. Increasing  $n$  is a prerequisite to also increase the maximal number of clusters  $k$ , as  $k \ll n$ . For practical applicability,  $k$  needs to be increased by an order of magnitude; we plan to do this in the future.

## References

1. Aljalbout, E., Golkov, V., Siddiqui, Y., Cremers, D.: Clustering with deep learning: Taxonomy and new methods. arXiv preprint arXiv:1801.07648 (2018)
2. Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., et al.: Deep speech 2: End-to-end speech recognition in english and mandarin. In: ICML (2016)
3. Arias-Castro, E.: Clustering based on pairwise distances when the data is of mixed dimensions. IEEE Transactions on Information Theory (2011)
4. Basu, S., Banerjee, A., Mooney, R.: Semi-supervised clustering by seeding. In: ICML (2002)
5. Branson, S., Horn, G.V., Wah, C., Perona, P., Belongie, S.J.: The ignorant led by the blind: A hybrid human-machine vision system for fine-grained categorization. IJCV (2014)
6. Chin, C.F., Shih, A.C.C., Fan, K.C., et al.: A novel spectral clustering method based on pairwise distance matrix. J. Inf. Sci. Eng. (2010)
7. Chopra, S., Hadsell, R., LeCun, Y.: Learning a similarity metric discriminatively, with application to face verification. In: CVPR (2005)
8. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. J. of the royal stat. society, series B (1977)
9. Ester, M., Kriegel, H.P., Sander, J., Xu, X., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Kdd (1996)
10. Garofolo, J.S., Lamel, L.F., Fisher, W.M., Fiscus, J.G., Pallett, D.S., Dahlgren, N.L.: Darpa timit acoustic phonetic continuous speech corpus cdrom (1993)
11. Greff, K., van Steenkiste, S., Schmidhuber, J.: Neural expectation maximization. In: NIPS (2017)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
13. Hoffer, E., Ailon, N.: Deep metric learning using triplet network. In: International Workshop on Similarity-Based Pattern Recognition (2015)
14. Kampffmeyer, M., Løkse, S., Bianchi, F.M., Livi, L., Salberg, A.B., Robert, J.: Deep divergence-based clustering. In: Machine Learning for Signal Processing (MLSP), 2017 IEEE 27th International Workshop on (2017)
15. Kaufman, L., Rousseeuw, P.J.: Finding groups in data: an introduction to cluster analysis (2005)
16. Krause, J., Stark, M., Deng, J., Li, F.F.: 3d object representations for fine-grained categorization. In: Computer Vision Workshops of ICCV (ICCVW) (2013)
17. Le, Q.V., Ranzato, M., Monga, R., Devin, M., Corrado, G., Chen, K., Dean, J., Ng, A.Y.: Building high-level features using large scale unsupervised learning. In: ICML (2012)
18. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE (1998)

19. Levine, S., Finn, C., Darrell, T., Abbeel, P.: End-to-end training of deep visuomotor policies. *JMLR* (2016)
20. Liu, D., Kubala, F.: Online speaker clustering. In: *Proceedings of ICASSP* (2003)
21. Lukic, Y., Vogt, C., Dürr, O., Stadelmann, T.: Speaker identification and clustering using convolutional neural networks. In: *Machine Learning for Signal Processing (MLSP)*, 2016 IEEE 26th International Workshop on (2016)
22. Lukic, Y., Vogt, C., Dürr, O., Stadelmann, T.: Learning embeddings for speaker clustering based on voice equality. In: *Machine Learning for Signal Processing (MLSP)*, 2017 IEEE 27th International Workshop on (2017)
23. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: *Proc. 5th Berkeley symposium on math. stat. and prob.* (1967)
24. Manmatha, R., Wu, C., Smola, A.J., Krähenbühl, P.: Sampling matters in deep embedding learning. In: *ICCV* (2017)
25. McDaid, A.F., Greene, D., Hurley, N.: Normalized mutual information to evaluate overlapping community finding algorithms. *arXiv preprint arXiv:1110.2515* (2011)
26. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013)
27. Murtagh, F.: A survey of recent advances in hierarchical clustering algorithms. *The Computer Journal* (1983)
28. Nayar, S., Nene, S., Murase, H.: Columbia object image library (coil 100). Department of Comp. Science, Columbia University, Tech. Rep. CUCS-006-96 (1996)
29. Ng, H.W., Winkler, S.: A data-driven approach to cleaning large face datasets. In: *ICIP* (2014)
30. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: *CVPR* (2015)
31. Schwenker, F., Kestler, H.A., Palm, G.: Three learning phases for radial-basis-function networks. *Neural networks* **14**(4-5), 439–458 (2001)
32. Shi, B., Bai, X., Yao, C.: An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *PAMI* (2017)
33. Sigtia, S., Benetos, E., Dixon, S.: An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM TASLP* (2016)
34. Song, H.O., Jegelka, S., Rathod, V., Murphy, K.: Deep metric learning via facility location. In: *CVPR* (2017)
35. Song, H.O., Xiang, Y., Jegelka, S., Savarese, S.: Deep metric learning via lifted structured feature embedding. In: *CVPR* (2016)
36. Wang, J., Zhou, F., Wen, S., Liu, X., Lin, Y.: Deep metric learning with angular loss. In: *ICCV* (2017)
37. Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al.: Google’s neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* (2016)
38. Xie, J., Girshick, R., Farhadi, A.: Unsupervised deep embedding for clustering analysis. In: *ICML* (2016)
39. Xing, E.P., Jordan, M.I., Russell, S.J., Ng, A.Y.: Distance metric learning with application to clustering with side-information. In: *NIPS* (2003)
40. Yang, J., Parikh, D., Batra, D.: Joint unsupervised learning of deep representations and image clusters. In: *CVPR* (2016)
41. Yang, J., Parikh, D., Batra, D.: Joint unsupervised learning of deep representations and image clusters. In: *CVPR* (2016)
42. Zeiler, M.D.: Adadelata: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* (2012)