# Learning Neural Models for End-to-End Clustering

Benjamin Bruno Meier[1,2(✉)], Ismail Elezi[1,3], Mohammadreza Amirian[1,4], Oliver Dürr[1,5], and Thilo Stadelmann[1]

[1] ZHAW Datalab & School of Engineering, Winterthur, Switzerland
benjamin.meier70@gmail.com
[2] ARGUS DATA INSIGHTS Schweiz AG, Zurich, Switzerland
[3] Ca' Foscari University of Venice, Venice, Italy
[4] Institute of Neural Information Processing, Ulm University, Ulm, Germany
[5] Institute for Optical Systems, HTWG Konstanz, 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 $k$, and for each $1 \leq k \leq k_{\max}$, a distribution over the individual cluster assignment for each data point. The network is trained in advance in a supervised fashion on separate data to learn grouping by any perceptual similarity criterion based on pairwise labels (same/different group). It can then be applied to different data containing different groups. 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 to deep metric learning, semi-supervise clustering and other related approaches while having the advantage of performing learnable clustering fully end-to-end.

**Keywords:** Perceptual grouping · Learning to cluster
Speech & image clustering

## 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 characteristic of clustering perceptual, high-dimensional data like images [15] or sound [24]: 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 the number and specific type of groups present at application time are not known


L. Pancioni et al. (Eds.): ANNPR 2018, LNAI 11081, pp. 126–138, 2018.
https://doi.org/10.1007/978-3-319-99978-4_10

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).

Grouping objects with machine learning is usually approached with clustering algorithms [16]. Typical ones like K-means [25], EM [14], hierarchical clustering [29] with chosen distance measure, or DBSCAN [8] 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 high-level features that already encode the aspired similarity measure. This may be solved by learning salient embeddings [28] with a deep metric learning approach [12], 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 [19]: 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 [21], scene text recognition [34], or music transcription [35].

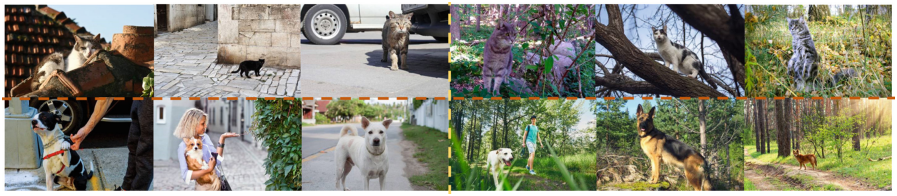

**Fig. 1.** Images of cats (top) and dogs (bottom) in urban (left) and natural (right) environments.

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 from *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 inference already during training (cp. Fig. 2). Its training stage may be compared to creating K-means, DBSCAN etc. in the first place: 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* [16] 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 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.

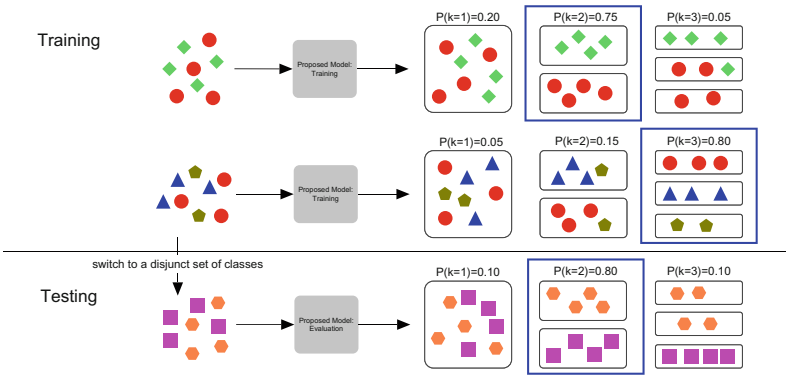


**Fig. 2.** Training vs. testing: cluster types encountered during application/inference are never seen in training. Exemplary outputs (right-hand side) contain a partition for each $k$ (1–3 here) and a corresponding probability (best highlighted blue). (Color figure online)

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. This is novel in its concept and generality (i.e., learn to cluster previously unseen groups end-to-end for arbitrary, high-dimensional input without any optimization on test data). Due to this novelty in approach, we focus here on the general idea and experimental demonstration of the principal workings, and leave comprehensive hyperparameter studies and optimizations for future work.

In Sect. 2, we compare our approach to related work, before presenting the model and training procedure in detail in Sect. 3. We evaluate our approach on different datasets in Sect. 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 Sect. 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.* [32] to get state-of-the-art accuracy in face detection. The main difference from the Siamese architecture is that in the latter case, the network sees same and different class objects with every example. It is then optimized to jointly learn their feature representation. A problem of both approaches is that they are typically difficult to train compared to a standard cross entropy loss.

*Song et al.* [37] developed an algorithm for taking full advantage of all the information available in training batches. They later refined the work [36] by proposing a new metric learning scheme based on structured prediction, which is designed to optimize a clustering quality metric (normalized mutual information [27]). Even better results were achieved by *Wong et al.* [38], where the authors proposed a novel angular loss, and achieved state-of-the-art results on the challenging real-world datasets *Stanford Cars* [17] and *Caltech Birds* [5]. On the other hand, *Lukic et al.* [23] 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.* [26] 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 [1]. They are trained in a fully unsupervised fashion, hence solve a different task then the one we motivated above (that is inspired by speaker- or image clustering based on some human notion of similarity). Perhaps first to group objects together in an unsupervised deep learning based manner where *Le et al.* [18], detecting high-level concepts like cats or humans. *Xie et al.* [40] used an autoencoder architecture to do clustering, but experimental evaluated it only simplistic datasets like *MNIST*. CNN-based approaches followed, e.g. by *Yang*

*et al.* [42], where clustering and feature representation are optimized together. *Greff et al.* [10] 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 from above-mentioned works in several respects: it has no assumption on the type of data, and solves the different task of grouping whole input objects.

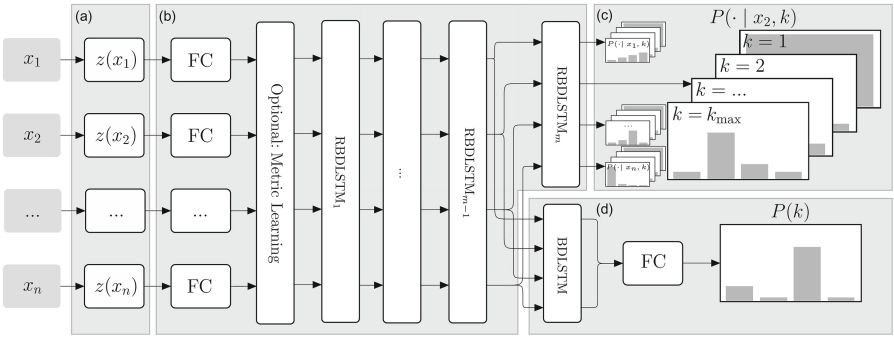


**Fig. 3.** Our complete model, consisting of (a) the embedding network, (b) clustering network (including an optional metric learning part, see Sect. 3.3), (c) cluster-assignment network and (d) cluster-count estimating network.

## 3 A Model for End-to-End Clustering of Arbitrary Data

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 application and constitutes the number of objects that can be clustered at a time, i.e. the maximum number of objects in a partition. The network's output is two-fold: a probability distribution $P(k)$ over the cluster count $1 \leq k \leq k_{\max}$; and probability distributions $P(\cdot \mid x_i, k)$ over all possible cluster indexes 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 data type. For example, 2D points do not require any embedding network. A fully connected layer (FC) with LeakyReLU activation at the beginning of the clustering network (b) 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 subnet (a) for any data type. The goal of the subnet (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 (subnet (d)) and a cluster assignment (subnet (c)).

To be able to process a non-fixed number of examples $n$ as input, we use a recurrent neural network. Specifically, we use stacked residual bi-directional LSTM-layers (RBDLSTM), which are similar to the cells described in [39] and visualized in Fig. 4. The residual connections allow a much more effective gradient flow during training [11] 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.

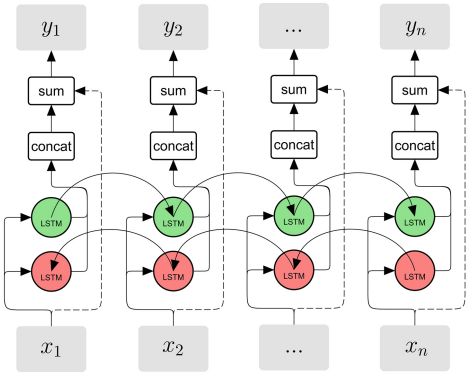


**Fig. 4.** RBDLSTM-layer: A BDLSTM with residual connections (dashed lines). The variables $x_i$ and $y_i$ are named independently from the notation in Fig. 3.

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 another BDLSTM-layer. Due to the bi-directionality of the network, we concatenate its first and the last output vector into a fully connected layer of twice as many units using again LeakyReLUs. The subsequent softmax-activation finally models the distribution $P(k)$ for $1 \leq k \leq k_{\max}$. The next subsection shows how this neural network learns to approximate these two complicated probability distributions [20] purely from pairwise constraints on data that is completely separate from any dataset to be clustered. No labels for clustering are needed.

### 3.2 Training and Loss

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

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

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

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

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*, $L_{\text{ca}}$, is then given by the weighted binary cross entropy as

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

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 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, $L_{\text{cc}}$, 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:

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

The complete loss is given by $L_{\text{tot}} = L_{\text{cc}} + \lambda L_{\text{ca}}$. During training, we prepare each mini batch with $N$ sets of $n$ input examples, each set with $k = 1 \ldots k_{\max}$ clusters chosen uniformly. Note that this training procedure requires only the knowledge of $y_{ij}$ and is thus also possible for weakly labeled data. All input examples are randomly shuffled for training and testing to avoid that the network learns a bias w.r.t. the input order. To demonstrate that the network really learns an intra-class distance and not just classifies objects of a fixed set of classes, it is applied on totally different clusters at evaluation time than seen during 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 Fig. 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 [3,6] provided by the metric learning block.

We construct a novel metric learning block inspired by the work of *Xing et al.* [41]. In contrast to their work, we optimize it end-to-end with backpropagation. This has been proposed in [33] 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

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

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 semidefinite 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* [9] 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* [30] 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) [22] and normalized mutual information (NMI) [27]. Architecturally, we choose $m = 14$ BDLSTM layers and 288 units in the FC layer of subnetwork (b), 128 units for the BDLSTM in subnetwork (d), and $\alpha = 0.3$ for all LeakyReLUs in the experiments below. All hyperparameters where chosen based on preliminary experiments to achieve reasonable performance, but not tested nor tweaked extensively. The code and further material and experiments are available online[1].

We set $k_{\max} = 5$ and $\lambda = 5$ for all experiments. For the 2D point data, we use $n = 72$ 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

[1] See https://github.com/kutoga/learning2cluster.

$n = 20$ audio snippets with a length of 1.28 s, encoded as mel-spectrograms with $128 \times 128$ pixels (identical to [24]). 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 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 [43] with a learning rate of 5.0. Example clusterings are shown in Fig. 5. For all configurations, the used hardware set the limit on parameter values: we used the maximum possible batch size and values for $n$ and $k_{\max}$ that allow reasonable training times. However, values of $n \geq 1000$ where tested and lead to a large decrease in model accuracy. This is a major issue for future work.

(a) (b) (c)

**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. (Color figure online)

**Table 1.** NMI $\in [0, 1]$ and MR $\in [0, 1]$ averaged over 300 evaluations of a trained network. We abbreviate our "learning to cluster" method as "L2C".

| | 2D points (self generated) | | TIMIT | | COIL-100 | |
|---|---|---|---|---|---|---|
| | MR | NMI | MR | NMI | MR | NMI |
| 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<br>DBSCAN: MR = 0.265, NMI = 0.676 | | [24]: MR = 0 | | [42]: NMI = 0.985 | |

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 [24] reach moderately better scores for the speaker clustering task and [42] 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. Hence, we assume, backed by the additional experiments to be found online, that our model works well also for other data types and datasets, given a suitable embedding network. Table 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 [31] 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 extend. This assumption is largely confirmed by the results in the "L2C+..." rows in Table 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 probabilities for cluster membership of all inputs as well as the number of clusters in test data. The learning phase only requires pairwise labels between examples from a separate training set, 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.

This is similar in spirit to the very recent work of *Hsu et al.* [13], but does not need and optimization on the test (clustering) set and finds $k$ autonomously. It is a novel approach to *learn to cluster*, introducing a novel architecture and loss design.

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 an early investigation into the 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, which foremost means to get rid of the dependency on $n$ for the partition size.

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 based on our model. 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 maximum 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. This might open up novel applications of our model in the area of transfer learning and domain adaptation.

**Acknowledgements.** We thank the anonymous reviewers for helpful feedback.

## 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., et al.: Deep speech 2: end-to-end speech recognition in English and Mandarin. In: ICML, pp. 173–182 (2016)
3. Arias-Castro, E.: Clustering based on pairwise distances when the data is of mixed dimensions. IEEE Trans. Inf. Theory **57**, 1692–1706 (2011)
4. Basu, S., Banerjee, A., Mooney, R.: Semi-supervised clustering by seeding. In: ICML, pp. 19–26 (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 **108**, 3–29 (2014)
6. Chin, C.F., Shih, A.C.C., Fan, K.C.: A novel spectral clustering method based on pairwise distance matrix. J. Inf. Sci. Eng. **26**, 649–658 (2010)

7. Chopra, S., Hadsell, R., LeCun, Y.: Learning a similarity metric discriminatively, with application to face verification. In: CVPR, vol. 1, pp. 539–546 (2005)
8. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD, pp. 226–231 (1996)
9. 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)
10. Greff, K., van Steenkiste, S., Schmidhuber, J.: Neural expectation maximization. In: NIPS, pp. 6694–6704 (2017)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR, pp. 770–778 (2016)
12. Hoffer, E., Ailon, N.: Deep metric learning using triplet network. In: Feragen, A., Pelillo, M., Loog, M. (eds.) SIMBAD 2015. LNCS, vol. 9370, pp. 84–92. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24261-3_7
13. Hsu, Y., Lv, Z., Kira, Z.: Learning to cluster in order to transfer across domains and tasks. In: ICLR (2018, accepted)
14. Jin, X., Han, J.: Expectation maximization clustering. In: Sammut, C., Webb, G.I. (eds.) Encyclopedia of Machine Learning, pp. 382–383. Springer, Boston (2011). https://doi.org/10.1007/978-0-387-30164-8
15. Kampffmeyer, M., Løkse, S., Bianchi, F.M., Livi, L., Salberg, A.B., Robert, J.: Deep divergence-based clustering. In: IEEE International Workshop on Machine Learning for Signal Processing (MLSP) (2017)
16. Kaufman, L., Rousseeuw, P.J.: Finding Groups in Data: An Introduction to Cluster Analysis. Wiley, Hoboken (1990)
17. Krause, J., Stark, M., Deng, J., Li, F.F.: 3D object representations for fine-grained categorization. In: Workshop on 3D Representation and Recognition at ICCV (2013)
18. Le, Q.V., et al.: Building high-level features using large scale unsupervised learning. In: ICML, pp. 8595–8598 (2012)
19. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**, 2278–2324 (1998)
20. Lee, H., Ge, R., Ma, T., Risteski, A., Arora, S.: On the ability of neural nets to express distributions. In: COLT, pp. 1271–1296 (2017)
21. Levine, S., Finn, C., Darrell, T., Abbeel, P.: End-to-end training of deep visuomotor policies. JMLR **17**(1), 1334–1373 (2016)
22. Liu, D., Kubala, F.: Online speaker clustering. In: ICASSP, vol. 1, pp. I-333-6 (2003)
23. Lukic, Y., Vogt, C., Dürr, O., Stadelmann, T.: Speaker identification and clustering using convolutional neural networks. In: IEEE International Workshop on Machine Learning for Signal Processing (MLSP) (2016)
24. Lukic, Y., Vogt, C., Dürr, O., Stadelmann, T.: Learning embeddings for speaker clustering based on voice equality. In: 2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP) (2017)
25. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: 5th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297 (1967)
26. Manmatha, R., Wu, C., Smola, A.J., Krähenbühl, P.: Sampling matters in deep embedding learning. In: ICCV, pp. 2840–2848 (2017)
27. McDaid, A.F., Greene, D., Hurley, N.: Normalized mutual information to evaluate overlapping community finding algorithms. arXiv preprint arXiv:1110.2515 (2011)
28. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)

29. Murtagh, F.: A survey of recent advances in hierarchical clustering algorithms. Comput. J. **26**, 354–359 (1983)
30. Nayar, S., Nene, S., Murase, H.: Columbia object image library (COIL 100). Department of Computer Science, Columbia University, Technical report, CUCS-006-96 (1996)
31. Ng, H.W., Winkler, S.: A data-driven approach to cleaning large face datasets. In: ICIP, pp. 343–347 (2014)
32. Schroff, F., Kalenichenko, D., Philbin, J.: FaceNet: A unified embedding for face recognition and clustering. In: CVPR, pp. 815–823 (2015)
33. Schwenker, F., Kestler, H.A., Palm, G.: Three learning phases for radial-basis-function networks. Neural Netw. **14**(4–5), 439–458 (2001)
34. 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 **39**, 2298–2304 (2017)
35. Sigtia, S., Benetos, E., Dixon, S.: An end-to-end neural network for polyphonic piano music transcription. IEEE/ACM TASLP **24**(5), 927–939 (2016)
36. Song, H.O., Jegelka, S., Rathod, V., Murphy, K.: Deep metric learning via facility location. In: CVPR, pp. 5382–5390 (2017)
37. Song, H.O., Xiang, Y., Jegelka, S., Savarese, S.: Deep metric learning via lifted structured feature embedding. In: CVPR, pp. 4004–4012 (2016)
38. Wang, J., Zhou, F., Wen, S., Liu, X., Lin, Y.: Deep metric learning with angular loss. In: ICCV, pp. 2593–2601 (2017)
39. Wu, Y., et al.: Google's neural machine translation system: bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144 (2016)
40. Xie, J., Girshick, R., Farhadi, A.: Unsupervised deep embedding for clustering analysis. In: ICML, pp. 478–487 (2016)
41. Xing, E.P., Jordan, M.I., Russell, S.J., Ng, A.Y.: Distance metric learning with application to clustering with side-information. In: NIPS, pp. 521–528 (2003)
42. Yang, J., Parikh, D., Batra, D.: Joint unsupervised learning of deep representations and image clusters. In: CVPR, pp. 5147–5156 (2016)
43. Zeiler, M.D.: ADADELTA: an adaptive learning rate method. arXiv preprint arXiv:1212.5701 (2012)