Neural Networks 113 (2019) 91–101

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Deep divergence-based approach to clustering

Michael Kampffmeyer [a,*], Sigurd Løkse [a], Filippo M. Bianchi [a], Lorenzo Livi [b,c], Arnt-Børre Salberg [d], Robert Jenssen [a,d]

[a] Machine Learning Group, UiT the Arctic University of Norway, Norway [1]
[b] Department of Computer Science, University of Exeter, UK
[c] Departments of Computer Science and Mathematics, University of Manitoba, Canada
[d] Norwegian Computing Center, Oslo, Norway

ARTICLE INFO

Article history:
Received 8 June 2018
Received in revised form 14 January 2019
Accepted 29 January 2019
Available online 8 February 2019

Keywords:
Deep learning
Clustering
Unsupervised learning
Information-theoretic learning
Divergence

ABSTRACT

A promising direction in deep learning research consists in learning representations and simultaneously discovering cluster structure in unlabeled data by optimizing a discriminative loss function. As opposed to supervised deep learning, this line of research is in its infancy, and how to design and optimize suitable loss functions to train deep neural networks for clustering is still an open question. Our contribution to this emerging field is a new deep clustering network that leverages the discriminative power of information-theoretic divergence measures, which have been shown to be effective in traditional clustering. We propose a novel loss function that incorporates geometric regularization constraints, thus avoiding degenerate structures of the resulting clustering partition. Experiments on synthetic benchmarks and real datasets show that the proposed network achieves competitive performance with respect to other state-of-the-art methods, scales well to large datasets, and does not require pre-training steps.

1. Introduction

Deep neural networks (Goodfellow, Bengio, & Courville, 2016; Krizhevsky, Sutskever, & Hinton, 2012) excel at hierarchical representation learning (Bengio, Courville, & Vincent, 2013), and yield state-of-the-art performance in image classification (Krizhevsky et al., 2012), object detection (Ren, He, Girshick, & Sun, 2015), segmentation (Kampffmeyer, Salberg, & Jenssen, 2016; Long, Shelhamer, & Darrell, 2015), time series prediction (Bianchi, Maiorino, Kampffmeyer, Rizzi, & Jenssen, 2017) and speech recognition (Graves, Mohamed, & Hinton, 2013), to name a few. However, deep networks are usually trained in a supervised manner, hence requiring a large amount of labeled data. This is a challenge in many application domains.

Clustering (Jain, 2010; Von Luxburg, 2007), one of the fundamental areas in machine learning, aims at categorizing unlabeled data into groups (clusters). A promising direction in deep learning research is to learn representations and simultaneously discover cluster structure in unlabeled data by optimizing a discriminative loss function. Deep Embedded Clustering (DEC) (Xie, Girshick, & Farhadi, 2016) exemplifies this line of work and represents, to the best of our knowledge, the state-of-the-art. DEC is based on an optimization strategy in which a neural network is pre-trained by means of an autoencoder and then fine-tuned by jointly optimizing cluster centroids in output space and the underlying feature representation. Another example is (Yang, Fu, Sidiropoulos and Hong, 2016), where the authors propose a joint optimization for dimensionality reduction using a neural network and k-means clustering. Alternative approaches to unsupervised deep learning based on adversarial networks have recently been proposed (Goodfellow et al., 2014). These approaches are different in spirit but can also be used for clustering (Makhzani, Shlens, Jaitly, Goodfellow, & Frey, 2015; Springenberg, 2015).

In this work, we propose what we called the Deep Divergence-based Clustering (DDC) algorithm. Our method takes inspiration from the vast literature on traditional clustering techniques that optimize discriminative loss functions based on information-theoretic measures (Dhillon, Mallela, & Kumar, 2003; Jenssen, Erdogmus, Hild, Principe, & Eltoft, 2007; Tishby & Slonim, 2001; Vikjord & Jenssen, 2014). The main motivation for this choice is that the divergence, as a measure of dissimilarity between clusters represented by their probability density functions, builds on two fundamental objectives (Fig. 1): the separation between clusters and the compactness within clusters. These are desirable properties to increase identifiability of nonparametric mixtures (Aragam, Dan, Ravikumar, & Xing, 2018). Our new divergence-based loss function for deep clustering supports end-to-end learning and explicitly exploits knowledge about the geometry of the output space during the optimization. DDC achieves state-of-the-art performance without requiring hand-crafted feature design, reducing also the importance of a pre-training phase.

* Corresponding author.
E-mail address: michael.c.kampffmeyer@uit.no (M. Kampffmeyer).

[1] http://machine-learning.uit.no/.

https://doi.org/10.1016/j.neunet.2019.01.015

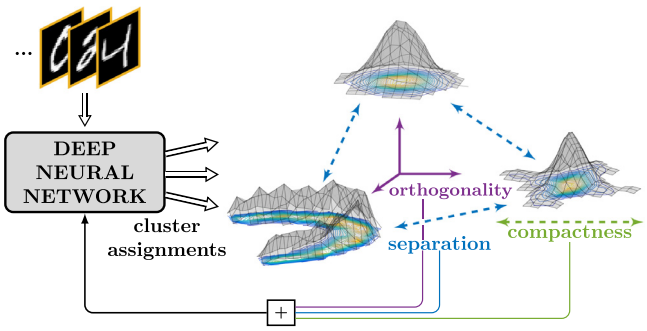


Fig. 1. Our approach takes advantage of the power of deep learning to extract features and perform clustering in an end-to-end manner. The proposed loss function is rooted in two fundamental objectives of clustering: separation and compactness of clusters.

A preliminary version of this paper appeared in Kampffmeyer, Løkse, Bianchi, Livi, Salberg and Jenssen (2017). The preliminary version was targeted towards image clustering combining a convolutional neural network architecture with our proposed clustering loss function. Here, we extend our work by (i) modifying the proposed architecture such that it can also handle textual data; (ii) conducting experiments and comparisons on additional datasets (including textual data — Reuters dataset); (iii) providing a thorough analysis of the proposed cost function and its components via ablation experiments; (iv) illustrating and discussing the functioning of the method in controlled settings; (v) interpreting predictions of the network by means of guided backpropagation (Springenberg, Dosovitskiy, Brox, & Riedmiller, 2014); and finally (vi) providing a more thorough literature background discussion, placing our work into a broader context.

This paper is structured as follows. Section 2 provides an overview of related works. Section 3 presents the proposed methodology for performing clustering with deep networks. In Section 4, we show the experimental results on several datasets and analyze the proposed cost function in detail. Finally, in Section 5 we draw conclusions and point to future directions.

2. Related work

Common approaches to unsupervised deep learning include methods based on deep belief networks, autoencoders, and generative adversarial networks (Bengio et al., 2013; Goodfellow et al., 2014). These methods have been mainly used for unsupervised pre-training (Erhan, Bengio, Courville, Manzagol, Vincent, & Bengio, 2010). Deep belief networks were the first of these models to be proposed and consist of stacked restricted Boltzmann machines that are trained in a greedy fashion (Hinton, Osindero, & Teh, 2006). Once trained, deep belief networks can be used to initialize neural networks.

Although several types of autoencoders have been proposed, all share a common underlying architecture consisting of an encoding and a decoding layer. The encoder is responsible for producing a hidden representation; the decoder re-generates inputs from the hidden representation. Both can efficiently be learned using backpropagation, by minimizing the reconstruction loss between original input and decoder output. Variations include, among others, denoising autoencoders (Vincent, Larochelle, Bengio, & Manzagol, 2008), which regularize the original autoencoder model by adding noise to inputs and then changing the objective to both include reconstruction and denoising, contractive autoencoders (Rifai, Vincent, Muller, Glorot, & Bengio, 2011), and more recently autoencoders that are regularized by preserving similarities in input space (Kampffmeyer, Løkse, Bianchi, Jenssen and Livi, 2017). Variational autoencoders (Kingma & Welling, 2013) have been used recently for several unsupervised tasks, such as image generation (Gregor, Danihelka, Graves, Rezende, & Wierstra, 2015) and segmentation (Sohn, Lee, & Yan, 2015). This approach assumes that data are generated from directed graphical models and uses a variational approach to learn latent representations.

Adversarial generative models (Goodfellow et al., 2014) are more recent approaches to unsupervised deep learning. Here, two networks are trained: one is responsible for discriminating between real and generated images; the other is responsible for generating realistic-enough images to confuse the first network.

Clustering is a classic information processing problem, particularly important in machine learning (Bianchi, Livi, & Rizzi, 2016; Jain, 2010; Myhre, Mikalsen, Løkse, & Jenssen, 2018; Nie, Tian, & Li, 2018; Rodriguez & Laio, 2014). Countless approaches exist for clustering, with mean shift (Comaniciu & Meer, 2002), k-means and expectation–maximization algorithms (Aggarwal & Reddy, 2013), being some of the most well-known ones. In the last decade, *spectral clustering* played a prominent role in the field, see for instance (Jenssen, 2010; Ng, Jordan, Weiss, et al., 2002; Nie, Zeng, Tsang, Xu, & Zhang, 2011; Von Luxburg, 2007; Yang, Xu, Nie, Yan, & Zhuang, 2010). Spectral clustering exploits the spectrum of similarity matrices to partition input data. Although these methods have demonstrated good performance in complex problems, they suffer from lack of scalability with respect to the number of input data points; cubic computational complexity for eigensolvers and quadratic complexity in terms of memory occupation. Attempts to solve these problems have been made by designing approximations or employing different optimization techniques (Dhillon, Guan, & Kulis, 2004; Han & Filippone, 2016; Yan, Huang, & Jordan, 2009).

Only a few methods have been proposed to exploit deep learning architectures for clustering, thereby taking advantage of hierarchical feature representations learned by such networks. CatGAN (Springenberg, 2015) and AAE (Makhzani et al., 2015) are based on the idea of adversarial networks. CatGAN is a method for learning a discriminative model, trained by optimizing a loss function implementing two different objectives. The first accounts for mutual information and predicted categorical distribution of classes in the data. The second objective maximizes the robustness of the discriminative network against an adversarial generative model. AAE instead assumes that data are generated from two latent variables, one associated with a categorical distribution and the other with a Gaussian distribution, and uses two adversarial networks to impose these distributions on the data representation. In a recent contribution (Bojanowski & Joulin, 2017), the authors propose an unsupervised training algorithm for CNNs and test its performance on image classification problems. The idea is to deal with the so-called "feature collapse problem" by mapping the learned features on random targets uniformly distributed on a d-dimensional unit sphere. A combination of recurrent and convolutional networks has also been used to perform image clustering by interpreting agglomerative clustering as a recurrent process (Yang, Parikh and Batra, 2016). Another recent approach to clustering based on the idea of hierarchical feature representation learning is provided by Zhang (2018), who proposes a multilayer bootstrap network where each layer performs multiple mutually independent k-centroids clusterings. Each layer gets trained individually in a bottom-up fashion and the input of consecutive layers is an indicator vector of which centroids are closest to a given input. Unlike the previously discussed methods, the multilayer bootstrap network does not offer end-to-end training.

To the best of our knowledge, DEC (Xie et al., 2016) is the method that is most closely related to our approach, as it is also founded on traditional clustering approaches. DEC simultaneously learns a feature representation as well as a cluster assignment in

a two-step procedure. In the first step, soft assignments are computed between the data and cluster centroids based on a Student's t-distribution. Then, the parameters are optimized by matching soft assignments to a target distribution, which expresses confidence in assignments. The matching is performed by minimizing the Kullback–Leibler divergence. However, the effectiveness of DEC depends on a pre-training step implemented with autoencoders and does require explicit feature design to handle complex image data, e.g., Histogram of Oriented Gradients (HOG) features (Dalal & Triggs, 2005).

3. Deep clustering

We first describe, in Section 3.1 the proposed clustering loss function and present a description of the overall algorithm in Section 3.2. Successively, in Section 3.3, we discuss the deep network architectures that we propose to use for clustering problems, namely one that is based on convolutional layers for image clustering and one based on fully connected layers for vectorial data. Finally, we discuss scalability in Section 3.4.

Inspired by recent successes of introducing companion losses (Lee, Xie, Gallagher, Zhang, & Tu, 2015) to supervised deep learning models, we propose a loss function for clustering that includes terms computed over several network layers. The details of the loss function are outlined in what follows.

3.1. The loss function for clustering

The design of a loss function that allows the network to learn via gradient descent the intrinsic cluster structure in the input data is a fundamental part of this work. As illustrated in Fig. 2 and explained below, in addition to exploiting the geometry of the output space induced by the softmax activation, we adopt a kernel-based approach to estimate the divergence between clusters.

3.1.1. Loss term based on information-theoretic divergence measures

An information-theoretic divergence measure computes the dissimilarity between probability density functions (PDFs). For a clustering application, one would model each cluster by its PDF and optimize cluster assignments such that the divergence between their PDFs is maximized. Several different formulations of divergence measures exist in the literature (Basseville, 2013), many of which are suitable for clustering. In this work, we focus on one particular divergence measure that has been proven useful for clustering in the past, namely the Cauchy-Schwarz (CS) divergence (Jenssen, Principe, Erdogmus, & Eltoft, 2006; Vikjord & Jenssen, 2014), also referred to as the Information Cut in a graph clustering perspective (Jenssen et al., 2007). The CS divergence can be used in multi-cluster problems (i.e., problems with more than two clusters) by averaging the pairwise divergence over all pairs of cluster PDFs.

Considering $k \geq 2$ distinct PDFs, the CS divergence is defined as

$$D_{\mathrm{cs}}(p_1, \ldots, p_k) = -\log\left(\frac{1}{k}\sum_{i=1}^{k-1}\sum_{j>i}\frac{\int p_i(\mathbf{x})p_j(\mathbf{x})d\mathbf{x}}{\sqrt{\int p_i^2(\mathbf{x})d\mathbf{x}\int p_j^2(\mathbf{x})d\mathbf{x}}}\right). \tag{1}$$

For a pair of PDFs, p_i and p_j, we have $0 \leq D_{\mathrm{cs}}(p_1, p_2) < \infty$, where we obtain the minimum value iff $p_i = p_j$. Thus, in order to maximize cluster separation and compactness, we want the divergence to be as large as possible. Since the logarithm is a monotonic function, maximizing (1) is in practice equivalent to minimizing the argument of the logarithm. We observe that the minimum is obtained when the numerator is small and the denominator is large. Intuitively, this fact implies that the similarity between samples in different clusters is small (numerator) and similarity of samples within the same cluster is large (denominator).

In this paper, we make use of the divergence in (1) to measure the distance between clusters. Since the underlying true densities p_i, p_j are unknown, we follow a data-driven approach and approximate the PDFs using a Parzen window estimator, configured with a Gaussian kernel having bandwidth σ. We define the matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$ that encodes the similarities between n input data. The matrix element $k_{i,j}$ stores the value $\exp(-d(\mathbf{x}_i, \mathbf{x}_j)^2/(2\sigma^2))$, where $d(\mathbf{x}_i, \mathbf{x}_j)$ is the Euclidean distance between data point $\mathbf{x}_i$ and $\mathbf{x}_j$. Using the Parzen window estimator, the CS divergence can be expressed as (Jenssen et al., 2006)

$$D_{\mathrm{cs}} = -\log\left(\frac{1}{k}\sum_{i=1}^{k-1}\sum_{j>i}\frac{\sum_{q\in C_i}\sum_{l\in C_j}k_{q,l}}{\sqrt{\sum_{q,q'\in C_i}k_{q,q'}\sum_{l,l'\in C_j}k_{l,l'}}}\right). \tag{2}$$

Note that this estimate in (2) of the CS divergence can also be interpreted as measuring the cosine of the angle between cluster means in a Reproducing Kernel Hilbert Space (Jenssen et al., 2006) and is closely related to maximum mean discrepancy (Gretton, Borgwardt, Rasch, Schölkopf, & Smola, 2012). Assume that we have a $n \times k$ cluster assignment matrix $\mathbf{A} = [\alpha_{q,i}]$, with elements $\alpha_{q,i} \in \{0, 1\}$ that represent the crisp cluster assignment of data point q to cluster C_i. Thus, each data point q is represented by a one-hot vector. Then,

$$\sum_{q\in C_i}\sum_{l\in C_j}k_{q,l} = \sum_{q,l=1}^{n}\alpha_{q,i}\alpha_{l,j}k_{q,l} = \boldsymbol{\alpha}_i^T\mathbf{K}\boldsymbol{\alpha}_j,$$

where $\boldsymbol{\alpha}_i$ is the ith column of $\mathbf{A}$. The CS-divergence becomes $D_{\mathrm{cs}} = -\log(d_\alpha)$, where

$$d_\alpha = \frac{1}{k}\sum_{i=1}^{k-1}\sum_{j>i}\frac{\boldsymbol{\alpha}_i^T\mathbf{K}\boldsymbol{\alpha}_j}{\sqrt{\boldsymbol{\alpha}_i^T\mathbf{K}\boldsymbol{\alpha}_i\boldsymbol{\alpha}_j^T\mathbf{K}\boldsymbol{\alpha}_j}}. \tag{3}$$

The formulation of the CS-divergence in (2) generalizes to soft cluster assignments, $\alpha_{q,i}$, preserving the differentiability of the loss function. In our architecture, the soft cluster assignments correspond to the softmax outputs and thereby the probability of a data point q to belong to cluster C_i.

The similarity values in $\mathbf{K}$ depend on the data representation. In particular, as data are processed by the neural network, several non-linear transformations map inputs onto different feature spaces, representing different levels of abstraction. The kernel bandwidth σ is computed based on the statistics of the learned representations. More details can be found in Section 4.6. To take advantage of the different representations and use the idea of companion losses for restricting the intermediate representations of the network, we use the hidden representation computed by the last fully connected layer before the output layer in addition to the soft cluster assignments produced by the softmax output layer. We do this by computing a (kernel) similarity matrix $\mathbf{K}_{\mathrm{hid}}$, which, by considering the corresponding $d_{\mathrm{hid},\alpha}$ in (3), yields a similarity score.

3.1.2. Loss term based on the geometry of the output space

The output space has a fixed number of dimensions (corresponding to the number of output neurons/clusters) and a precise geometry induced by the softmax activations used in the output layer (whose elements sum to 1), which we exploit in our algorithm:

1. The output space is a simplex in $\mathbb{R}^k$;
2. A data points degree of membership to a given cluster is maximized if the cluster assignment lies in a corner of the simplex (i.e., $\alpha_{q,i} = 1$ if data point q is fully assigned to cluster C_i);

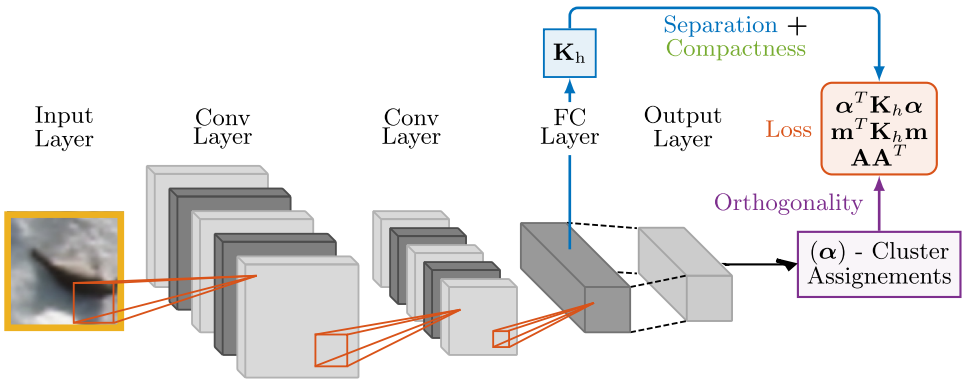


Fig. 2. Schematic depiction of the proposed architecture for image datasets and details of the loss function. The network consists of two convolution layers (each one followed by a pooling layer, not depicted in the figure) and a fully connected (FC) layer. Each layer is followed by a non-linear ReLU transformation. Finally, a fully connected output layer implements a logistic function (softmax). The unsupervised loss function operates on the kernel matrix $\mathbf{K}_h$ encoding data similarities evaluated on the hidden representation, and the values of the cluster assignment returned by the softmax function. The orthogonality constraint is derived from cluster assignments, while separation and compactness constraints come from the Cauchy-Schwarz divergence, computed on the similarity matrix (weighted by cluster assignments). The convolutional layers are replaced when non-image data are considered.

3. Following from the previous point, cluster assignment vectors of data points assigned to different clusters, in the optimal case, should be orthogonal to each other.

This intuition about the geometry enables us to introduce a term that avoids degenerate solutions by addressing the aforementioned problem of collapsing features/clusters and encourages diversity in cluster assignment. For a given cluster assignment matrix, $\mathbf{A}$, the strictly upper triangular elements of $\mathbf{A}\mathbf{A}^T$, denoted by $\text{triu}(\mathbf{A}\mathbf{A}^T)$, consists of inner products between cluster assignment vectors. Unless explicitly stated, $\text{triu}(\mathbf{A}\mathbf{A}^T)$ will denote the sum of these elements. Note that we do not include the elements on the diagonal. Further, $\mathbf{A}\mathbf{A}^T$ will consist entirely of non-negative elements because $\mathbf{A}$ is non-negative; cluster assignment vectors are orthogonal if and only if these inner products are zero. Thus, our criterion consists of enforcing low values in the upper triangular elements. This also has the effect of a regularization term if the number of clusters is smaller than the number of input data points. Indeed, not all data points in the restricted space can be orthogonal to each other, forcing data points to repel each other, thereby acting against the terms that try to improve similarity. This term also encourages a balanced distribution of data points in the different classes, which makes our loss ideal for problems with balanced classes. Alternative regularization methods that are not based on the balanced class assumption will be investigated in future work.

The fact that cluster assignment vectors are orthogonal, however, does not imply that such vectors are embedded in a corner of the simplex. As an example, assume that $\alpha_{q,i} = 1$ and $\alpha_{l,i} = 0$. Due to the restrictions of the simplex geometry, it follows that $\alpha_{q,k} = 0, \;\; k \neq i$ and therefore $\boldsymbol{\alpha}_q^T\boldsymbol{\alpha}_l = 0$ independently of the values of $\alpha_{l,k}, \;\; k \neq i$. Thus, l is not restricted to a simplex corner. Therefore, in order to enforce closeness to a corner of the simplex, we define an additional term for the loss function that reads

$$m_{q,i} = \exp(-\|\boldsymbol{\alpha}_q - \mathbf{e}_i\|^2),$$

where $\mathbf{e}_i \in \mathbb{R}^k$ is a vector denoting the ith corner of the simplex; representing cluster C_i. This exponential evaluates to one only when $\boldsymbol{\alpha}_q$ is located in a corner of the simplex. We make use of this fact by defining a third similarity term $d_{\text{hid,m}}$, where $\mathbf{m}_i = [m_{q,i}] \in \mathbb{R}^n$ takes the place of $\boldsymbol{\alpha}_i$ in (3).

3.1.3. The final clustering loss function

The weights in the neural network architecture described in Section 3.3 can then be trained by minimizing the sum of the three different terms discussed in the previous section:

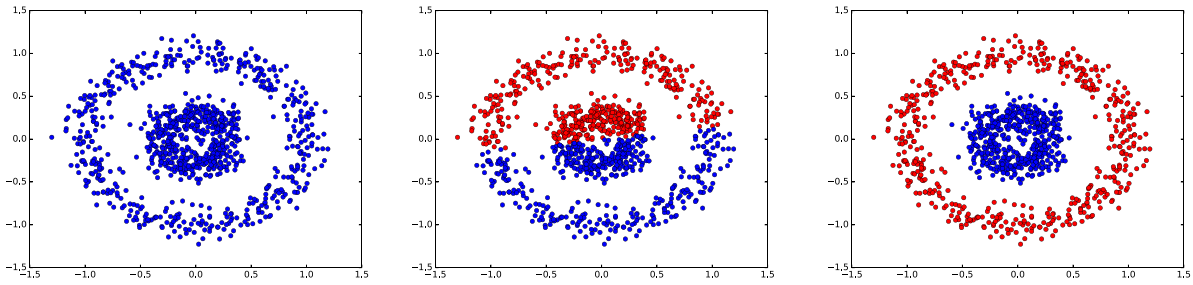
$$\begin{aligned} L &= d_{\text{hid},\,\alpha} + \text{triu}(\mathbf{A}\mathbf{A}^T) + d_{\text{hid,m}} \\ &= \frac{1}{k}\sum_{i=1}^{k-1}\sum_{j>i}\frac{\boldsymbol{\alpha}_i^T\mathbf{K}_{hid}\boldsymbol{\alpha}_j}{\sqrt{\boldsymbol{\alpha}_i^T\mathbf{K}_{hid}\boldsymbol{\alpha}_i\boldsymbol{\alpha}_j^T\mathbf{K}_{hid}\boldsymbol{\alpha}_j}} + \text{triu}(\mathbf{A}\mathbf{A}^T) \\ &\quad + \frac{1}{k}\sum_{i=1}^{k-1}\sum_{j>i}\frac{\mathbf{m}_i^T\mathbf{K}_{hid}\mathbf{m}_j}{\sqrt{\mathbf{m}_i^T\mathbf{K}_{hid}\mathbf{m}_i\mathbf{m}_j^T\mathbf{K}_{hid}\mathbf{m}_j}} \end{aligned} \tag{4}$$

As a proof of concept, we illustrate the functioning of our clustering method on a classical synthetic dataset where one class is represented by a small circle and the other class is a ring. Note, that we use a fully-connected architecture (described in Section 3.3) for these experiments as we are considering non-image data. The dataset is shown in Fig. 3a. Figs. 3b and 3c illustrate the clustering outcome using k-means and the proposed method, respectively. It can be observed that the proposed method captures the highly nonlinear structure in the data and is able to discover clusters of non-spherical shapes, a highly desirable quality of clustering algorithms (Rodriguez & Laio, 2014).

Further, in Fig. 4 we visualize the output space for a three-cluster experiment. We chose three classes of the MNIST dataset (for dataset details see Section 4) and visualize the output space configuration during three different stages of the optimization process. As expected, the proposed clustering loss function (4) attempts to separate the data points by grouping similar points in ideal cluster centers located at the corners of the simplex. Note, we are using the convolutional architecture described in Section 3.3 as we are considering images in this experiment.

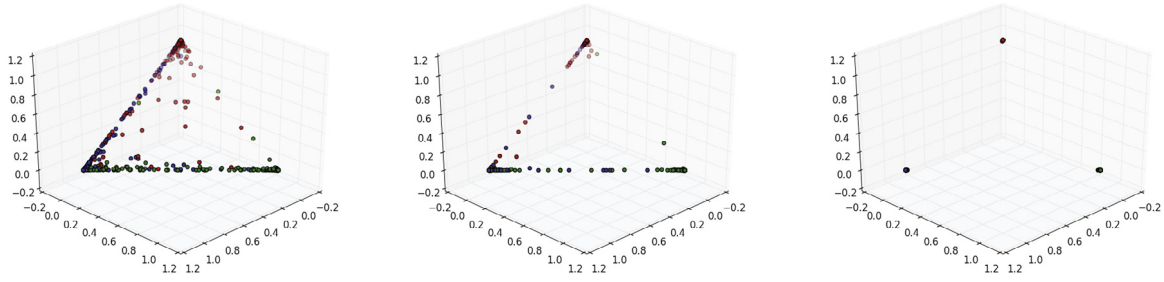
3.2. Description of complete algorithm

In this section, we summarize the proposed algorithm. For a given data batch, the assignment vectors and the hidden representation are obtained via a single forward pass. Based on the hidden representation, the kernel bandwidth, σ, is estimated and the kernel matrix is computed. From the assignment vectors, the distance to each of the ideal cluster centers (simplex corners) is computed, obtaining $\boldsymbol{m}_q$ for each data point q in a given minibatch. Using Eq. (4), we then can compute the loss based on the kernel matrix, the assignment vector $\boldsymbol{\alpha}_q$, and $\boldsymbol{m}_q$. Finally, all weights in the network are updated using Adam (Kingma & Ba, 2014). The full algorithm is outlined in Algorithm 1.



(a) Sample dataset. (b) Outcome of k-means. (c) Outcome of DDC.

Fig. 3. Illustration of DDC clustering outcome on a synthetic dataset, showing the capability of learning non-linear structures.



(a) Epoch 0. (b) Epoch 5. (c) Epoch 15.

Fig. 4. Illustration of DDC output space for three class MNIST training. Colors indicate class information of data points.

Algorithm 1 Deep Divergence-Based Clustering

Input: $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$
Output: Cluster assignments $\mathcal{A} = \{\boldsymbol{\alpha}_i\}_{i=1}^n$
1: Randomly initialize network parameters $\boldsymbol{\Theta}$
2: **while** not converged **do**
3: Sample data batch $\mathcal{X}^{(b)}$ from $\mathcal{X}$
4: Obtain assignment vectors $\mathcal{A}^{(b)}$ and hidden representations $\mathcal{H}^{(b)}$
5: Compute $m_{q,i} = \exp(-\|\boldsymbol{\alpha}_q - \mathbf{e}_i\|^2),\ \forall\ \boldsymbol{\alpha}_q \in \mathcal{A}^{(b)}$
6: Estimate kernel bandwidth σ and compute $\mathbf{K}_{\text{hid}}^{(b)}$ from $\mathcal{H}^{(b)}$
7: Compute loss with (4) and update $\boldsymbol{\Theta}$ with gradient descent
8: **end while**

3.3. Architecture overview

The network architecture is a design choice, and as such there are many options. In this paper, we choose an approach based on convolutional neural networks to process different image datasets, using a LeNet-inspired architecture (LeCun, Bottou, Bengio, & Haffner, 1998). We selected LeNet since it is a well-known benchmark network that supports end-to-end learning and has been widely used for image classification. The architecture is depicted in Fig. 2. It consists of two convolutional layers: the first one with 32 and the second one with 64 5 $\times$ 5 filters, each of them followed by a 2 $\times$ 2 max pooling layer and a ReLU activation. The last convolutional layer is followed by a fully connected layer with 100 nodes, another ReLU nonlinearity and, finally, the softmax output layer, whose dimensionality corresponds to the number of desired clusters. Batch-normalization (Ioffe & Szegedy, 2015) is applied in the fully connected layer. This design choice was made to increase the models' robustness and is explained in Section 4.6.

Our approach can also be applied to cluster data that are not images simply by replacing the convolutional and pooling layers with fully connected layers. In particular, for the experiments conducted on non-image data, we use an architecture with four fully connected layers of size 200 $-$ 200 $-$ 500 $-$ C. The 500 unit fully connected layer includes batch-normalization and the C unit layer corresponds to the softmax output layer with dimensionality equal to the number of clusters.

Recently, theoretical advances in the theory of neural networks (Giryes, Sapiro, & Bronstein, 2016) highlighted how the metric structure of input data is preserved by deep neural networks with random i.i.d Gaussian weights. One restriction is the fact that this is only true in the case where the intrinsic dimensionality of the data is proportional to the network width. However, Giryes et al. (2016) proved that the intrinsic dimensionality of the data does not increase as the data propagate through the network, which suggests that the width of the network (the number of hidden units per layer) that we consider for our experiments should suffice. This theoretical property supports the design choice behind the proposed loss function, which estimates the divergence over the hidden representation, rather than in input space.

3.4. Main memory footprint

Using gradient-based optimization in neural networks allows us to process large datasets, overcoming well-known limitations of spectral methods mentioned in the introduction with regards to memory requirements. The memory cost of our approach is kept low by the use of mini-batch training and scales linearly with the number of input data points, n, compared to the quadratic or super quadratic complexities encountered in spectral methods. The proposed method scales quadratically with the mini-batch size m as the kernel matrix is computed over the hidden representation for a given mini-batch; however, this is generally not an issue as $m \ll n$.

Fig. 5. Examples from the SEALS-3 dataset. The top row displays examples from the harp seal class, the middle row from the hooded seal, and the bottom row from the background class.

4. Experiments

We evaluate DDC on the MNIST handwritten image dataset as it represents a well-known benchmark dataset in the literature. In addition, we evaluate our algorithm on two more challenging real-world datasets: one dataset for detection of seal pups in aerial images here referred to as the SEALS-dataset and the Reuters dataset for news story clustering. In the results, we compare our method to four alternative clustering approaches.

4.1. MNIST dataset

The MNIST dataset contains 70 000 handwritten images of the digits 0 to 9 (LeCun et al., 1998) and consists of images that were originally in the National Institute of Standards and Technology (NIST) dataset. The images are grayscale with the digits centered in the 28×28 images.

4.2. SEALS dataset

The SEALS-3 dataset consists of several thousand aerial RGB images acquired during surveys in the West Ice east of Greenland in 2007 and 2012 and east of New Foundland, Canada, in 2012. The images are acquired from approximately 300m altitude, and the pixel spacing is about 3 cm (depending on the exact flight altitude). A typical image size is $11\,500 \times 7500$ pixels. From these images 64×64 image crops of harp seal pups, hooded seal pups and background (non-seals) were extracted and down-sampled to 28×28 to fit our chosen architecture. As the smallest class consists of 1000 images, we select a reduced set of 1000 images from each class to create a balanced dataset. Fig. 5 shows example images for the three classes in the SEALS-3 dataset.

As the background class contains a large variety of images, such as white snow and black water images, unsupervised algorithms are likely to partition these instances into different clusters. Therefore, we additionally created and tested another dataset (SEALS-2), where the background class was not included.

4.3. Reuters dataset

The Reuters dataset consists of 800 000 news stories that have been manually categorized into a category tree (Lewis, Yang, Rose, & Li, 2004). In this work, similarly to Xie et al. (2016), we chose the four root categories as labels, removed stories that are labeled with multiple root categories and represent each news story as a feature vector consisting of the Term Frequency-Inverse Document Frequency (TF-IDF) of the 2000 most frequently occurring word stems. As done for the SEALS dataset, we select 54 000 datapoints from each class in order to balance the dataset.

4.4. Performance measures

To evaluate the partition quality obtained after training, we consider two different supervised measures. The first measure is the Normalized mutual information (NMI), defined as

$$NMI = \frac{I(l, c)}{\frac{1}{2}[H(l) + H(c)]}, \tag{5}$$

where $I(\cdot, \cdot)$ and $H(\cdot)$ denote mutual information and entropy functionals, respectively. The second evaluation metric is the unsupervised clustering accuracy

$$ACC = \max_{\mathcal{M}} \frac{\sum_{i=1}^{n} \delta(l_i = \mathcal{M}(c_i))}{n}, \tag{6}$$

where l_i refers to the ground truth label, c_i to the assigned cluster of data point i, and $\delta(\cdot)$ is the Dirac delta. $\mathcal{M}$ is the mapping function that corresponds to the optimal one-to-one assignment of clusters to label classes implemented by means of the Hungarian algorithm (Kuhn, 1955), which solves the linear assignment problem of assigning a cluster to its label class in polynomial time.

4.5. Baseline methods

As methods for comparison, we use k-means (with the so-called k-means++ initialization Arthur & Vassilvitskii, 2007) as a well-known baseline and a hierarchical information theoretic clustering approach (Vikjord & Jenssen, 2014) based on implicit cluster density estimation using (1) a k-NN approach (ITC-kNN) and (2) a parzen window approach (ITC-parzen). Further we compare our approach to a representative subset of state-of-the-art methods in clustering, namely *Deep Embedded Clustering* (DEC) (Xie et al., 2016),[2] *Spectral Embedded Clustering* (SEC) (Nie et al., 2011), and *Local Discriminant Models and Global Integration* (LDMGI) (Yang et al., 2010). SEC and LDMGI are spectral clustering algorithms based on the foundations discussed in Ng et al. (2002). In SEC, the authors jointly optimize the normalized cut loss function and a linear transformation from input to the embedding space for spectral clustering, such that the transformed data is close to the embedded data. The similarity is modeled using a Gaussian kernel. LDMGI optimizes a similar objective function, but the Laplacian matrix is learned by exploiting manifold structure and discriminant information, in contrast to most spectral clustering methods where the Laplacian is calculated by using a Gaussian kernel. Both of these methods use *spectral rotation* (Yu & Shi, 2003) to obtain the final cluster assignments instead of k-means, which is common for many spectral clustering methods. To the authors best knowledge, these two methods represent the current state-of-the-art in spectral clustering, outperforming conventional spectral clustering methods in a wide variety of clustering tasks (Nie et al., 2011; Yang et al., 2010).

Following the experiment setup of Xie et al. (2016), the parameters in the baseline models are set according to the suggestions in their respective papers, varying their hyperparameters over 9 possible choices. For each one, we run the baseline models 20 times. The best result is shown in the experiments. Due to the lack of hyperparameters in k-means (except the number of clusters k, which is fixed in our experiments), the accuracy for the best run from 20 different random initializations is reported.

4.6. Implementation

The proposed network model is trained end-to-end by using Adam (Kingma & Ba, 2014) and implemented using the Theano framework (Theano Development Team, 2016). For each image

[2] Caffe version of DEC publicly available: https://github.com/piiswrong/dec.

datasets we used the same convolutional architecture and for each vectorial datasets we used the same fully connected architecture. Training is performed on mini-batches of size 100. By avoiding a fine-tuning for each problem at hand, we show the robustness of our architecture. Training is performed with a learning rate of 0.001 for the convolutional neural network and 10^{-5} for the fully connected network. The network is trained for 70 000 iterations and the ordering of the mini-batches was reshuffled at each epoch. Weights of the network are initialized following He, Zhang, Ren, and Sun (2015). Following the rule-of-thumb in Jenssen (2010), σ of the Gaussian kernel was chosen to be 15% of the median pairwise Euclidean distances between the feature representation produced by the first fully-connected layer, which produced satisfying results for all datasets. The median is adaptive and recomputed as part of the cost function evaluation. In our experiments, we observed that this rule-of-thumb benefited considerably from activation rescaling through batch-normalization.

As DDC is prone to get stuck in local minima, a common problem for unsupervised deep architectures, we run DDC for 20 runs and report the accuracy of the run with the lowest value of our unsupervised loss function. We also report the results of a voting scheme of the top three runs according to our unsupervised loss function. Following Strehl and Ghosh (2002) and Vikjord and Jenssen (2014), clustering results of the best performing run are used as a starting point and the clustering results of the other two runs are aligned to it via the mapping function provided by the Hungarian algorithm in an unsupervised manner. Once the results are aligned, we combine them via a simple voting procedure and compute the final unsupervised clustering accuracy using (6). In the following, this network ensemble is referred to as DDC-VOTE. Note that the voting procedure is completely unsupervised and is commonly used in ensemble approaches.

4.7. Results

We compare DDC and DDC-VOTE to the baseline algorithms on the MNIST and SEALS datasets and observe that they outperform the baseline methods on all datasets. The results can be found in Table 1. Due to very high computational complexity, the ITC algorithm could not be evaluated on MNIST and large datasets in general. This also highlights an important advantage of our formulation with regards to previous clustering approaches based on the CS divergence. Unlike cluster algorithms that estimate the optimal number of clusters from data, our method and the baseline approaches require the user to specify the number of clusters beforehand. By following a common practice, we have chosen the number of clusters equal to the number of classes in the corresponding datasets. It can be observed that DDC-VOTE generally outperforms DDC, except in the case of the SEALS-3 dataset, where all tested clustering algorithms perform poorly due to the high variation characterizing the background class. Methods based on adversarial networks, namely AAE and CatGAN, have shown to perform well on the MNIST clustering task (the only real dataset analyzed in these papers), by clustering the dataset into a large number of groups ($\geq$ 16), and mapping these into the 10 original classes in a post-processing step. A similar approach could potentially be employed by DDC to boost performance and will be explored in future work.

In what follows, we qualitatively analyze obtained clustering performance. Fig. 6 displays clustering results for the SEALS-3 dataset, where each row corresponds to the top ten scoring images for each of the three clusters. It is possible to note that the clustering result for the second and third cluster corresponds to a mix of background and seal images, with the third one containing white background images and harp seals and the second cluster containing black background images and hooded seals. As clustering is an unsupervised task and does not necessarily agree with the available supervised information, this result is not unexpected due to the fact that the background class includes large variations.

Table 1
Clustering accuracy for DDC, DDC-VOTE, and the baseline models. Best results are highlighted in bold.

Datasets	Method	NMI	ACC[%]
MNIST	K-means	0.50	53.33
	ITC (parzen)	–	–
	ITC (kNN)	–	–
	SEC	0.77	68.82
	LDMGI	0.81	83.03
	DEC	0.81	84.31
	DDC	0.83	86.58
	DDC-VOTE	**0.87**	**88.49**
SEALS-3	K-means	0.13	51.33
	ITC (parzen)	0.003	35.30
	ITC (kNN)	0.10	53.95
	SEC	0.15	49.00
	LDMGI	0.13	50.43
	DEC	**0.17**	50.33
	DDC	0.14	**55.97**
	DDC-VOTE	0.13	53.30
SEALS-2	K-means	0.015	56.85
	ITC (parzen)	0.003	51.55
	ITC (kNN)	0.020	57.20
	SEC	0.021	58.15
	LDMGI	0.018	57.85
	DEC	0.005	54.04
	DDC	0.15	72.05
	DDC-VOTE	**0.18**	**74.65**
Reuters	K-means	0.38	60.5
	ITC (parzen)	–	–
	ITC (kNN)	–	–
	SEC	–	–
	LDMGI	–	–
	DEC	0.38	63.79
	DDC	0.50	73.06
	DDC-VOTE	**0.51**	**77.62**

To further illustrate the clusters found in the dataset, we increased the number of clusters for the SEALS-3 dataset to 10. Fig. 7a shows an overview over the learned clusters. It can be observed that DDC generally finds reasonable clusters, for example by grouping water (dark patches) in one cluster and white background images in another. Also, it is possible to note that DDC generally groups the two different seal classes into separate clusters and assigns images containing both water and snow to a specific class.

Fig. 7b illustrates clustering results on MNIST. Interestingly, each cluster represents a distinct number. However, it can be observed that the 7's and 9's are mixed, which is expected due to their shape similarity. Furthermore, the MNIST dataset contains 1's that are straight and 1's that are rotated. Our results indicate that some of the far leaning 1's are clustered together with the 2 class, which has a similar diagonal line.

The results for the SEALS-3 dataset coincide with our intuition that the background class is likely to be divided into different classes. For the SEALS-2 dataset, we observe that DDC outperforms the competitor algorithms by a large margin (Table 1). As the SEALS dataset is less structured and more challenging datasets, we observe that methods that operate directly in pixel space (i.e., raw input space) perform poorly, stressing the importance of extracting higher-level features for clustering. Fig. 8 shows clustering results for DDC, where it can be clearly observed that hooded seals and harp seals are separated into two distinct clusters.

The proposed clustering cost function is not dependent on the convolutional architecture used in the previous two experiments and can also be used for training fully connected neural networks — which are used when working with vectorial data. For this purpose, we consider the Reuters dataset and substitute convolutional layers with fully connected ones as described in Section 3.3. Results

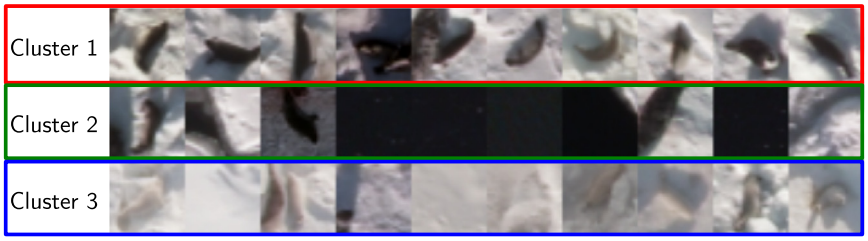


Fig. 6. Clustering result for the three classes in the SEALS-3 dataset. The first cluster appears to correspond to hooded seals, whereas the other two clusters correspond to a mix of background and seal images.

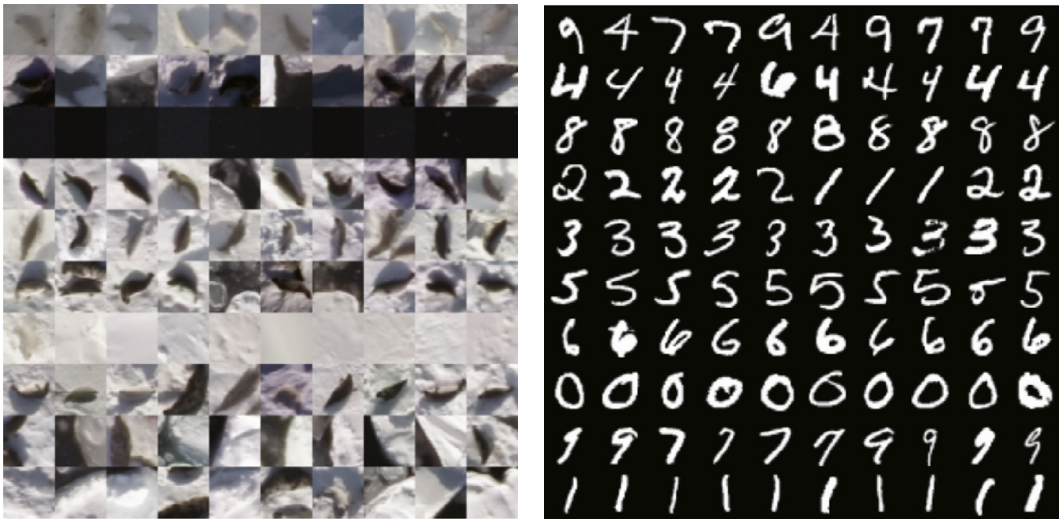


Fig. 7. (a) Results for the SEALS-3 dataset when clustering into ten clusters. (b) Clustering result for the ten-class MNIST dataset.

Fig. 8. Clustering result for the two classes in the SEALS-2 dataset. DDC groups the two seal types in distinct clusters. The dark seals in the top row corresponding to hooded seals and the light seals in the bottom row corresponding to harp seals. The red box indicates a mismatch.

are shown in Table 1, where it is possible to observe that DDC outperforms the competitors. Note that, due to the size of the Reuters dataset, running LDMGI and SEC was impossible as a consequence of their memory requirements discussed in Section 3.4. Note that, from the results presented in Xie et al. (2016), we can see that DEC still performs well when handling the imbalanced Reuters dataset, where the balanced assumption of DDC does not hold.

4.8. Loss function

In the following, we analyze the proposed loss function (4), providing empirical evidence of the importance of the different terms; moreover, we evaluate whether the different terms in the loss are related to the performance of the model. Fig. 9a shows the different terms in the loss function during the training phase as we monitor the accuracy of the best run on MNIST. It is clearly possible to observe that all terms (and also the overall loss) agree reasonably well with the overall clustering accuracy.

Considering that the network architecture is identical to networks used in supervised approaches and the availability of labels in our datasets, we can also monitor the terms of the loss function during supervised training. Fig. 9b shows how each term in the loss function and the overall loss decrease as we perform supervised training on MNIST using a cross-entropy loss function. Again, it is possible to notice that the individual terms have a similar decreasing trend. Note that large variations in the second loss term correspond to the aforementioned regularization effect (see Section 3.1.2).

In order to further analyze our method, we perform an ablation experiment (Nguyen, Yosinski, & Clune, 2015) to investigate the effect of the different terms on the clustering result. To this end, we recompute clustering accuracy on MNIST and Reuters for all different combinations of cost function terms. We repeat the experiments five times (20 runs each), compute the overall accuracy for the run with the lowest cost function each time and report the mean, standard deviation and the maximum accuracy values over these five results. The maximum accuracy value is

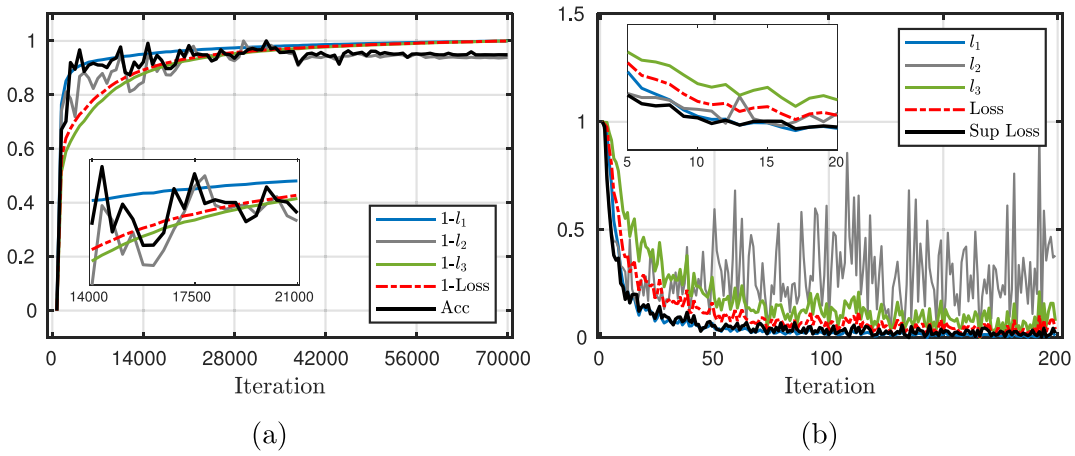


Fig. 9. (a) Comparison of the supervised accuracy with the unsupervised loss function (Loss) and its three terms ($l_1 = d_{\text{hid},\alpha}$, $l_2 = \text{triu}(\mathbf{A}\mathbf{A}^T)$, and $l_3 = d_{\text{hid,m}}$) during training. (b) Values of the loss function and its three terms when training the network using a supervised loss function (Sup Loss). Note, the losses have been rescaled to range [0,1].

Table 2
Results of the ablation experiment for the MNIST and the Reuters datasets, illustrating the effect of the three different terms ($l_1 = d_{\text{hid},\alpha}$, $l_2 = \text{triu}(\mathbf{A}\mathbf{A}^T)$, and $l_3 = d_{\text{hid,m}}$) composing the loss function (4).

Cost	MNIST		Reuters	
	Mean ± std	Max	Mean ± std	Max
l1	26.1 ± 2.9	31.5	39.3 ± 4.8	48.7
l2	48.9 ± 1.5	51.8	41.1 ± 2.9	46.0
l3	74.0 ± 2.9	77.0	70.2 ± 6.4	78.8
l1+l2	71.9 ± 8.0	83.5	66.0 ± 8.2	75.9
l1+l3	84.7 ± 5.1	88.6	64.8 ± 5.2	72.7
l2+l3	74.8 ± 5.5	84.3	70.3 ± 4.4	75.4
l1+l2+l3	80.8 ± 5.8	87.5	69.8 ± 7.3	82.6

reported solely in order to illustrate the maximum potential of the proposed method. In practice, the strategy from Section 4.6 would be used, wherein the best models according to the unsupervised cost function from each run are combined, denoted as DDC-Vote. Note, this differs from the results reported in Table 1 for DDC, where we report the accuracy only over 20 runs. Results for the MNIST and Reuters datasets are reported in Table 2. Our results illustrate that, by using all three terms together, we generally obtain better performance. However, the contribution of each term to the final performance is not consistent over all datasets. For instance, we observe that the $l2$ regularization term does not improve the overall result on the MNIST dataset, but does have a positive effect on the Reuters dataset.

The three terms in the loss function (4) were equally weighted in all experiments. However, better performance might be achieved by weighing such terms according to the data properties. For instance, decreasing the importance on $l_2 = \text{triu}(\mathbf{A}\mathbf{A}^T)$ might allow our method to better handle imbalanced datasets. The analysis of more advanced weighting schemes is left for future work.

We further conducted experiments where we replaced the CS divergence (2) with a symmetrized Kullback–Leibler (KL) divergence, the divergence used by DEC. However, in our experiments we noticed that the performance of the proposed method drops considerably. On MNIST DDC with KL divergence only obtains an accuracy of 53.66%. Further, using a cosine similarity together with the KL divergence for DDC obtains 35.48% accuracy. We hypothesize that this is mainly due to the fact that the KL divergence encourages separation of clusters, but does not necessarily enforce their compactness. A more thorough analysis of alternative divergence measures is left for future work.

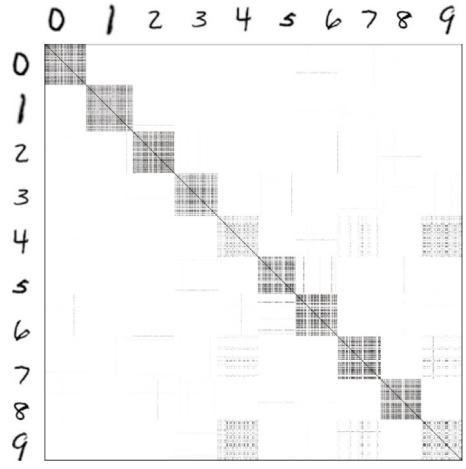


Fig. 10. Kernel matrix computed over the learned hidden representation. White colors correspond to low values in the kernel matrix, whereas dark values indicate large values.

4.9. Learned representation

Fig. 10 illustrates the final kernel matrix $\mathbf{K}$ computed over the hidden layer for the best MNIST run. Here, unlike in the case of training, where data points are fed to the model in a random order, the data points have been sorted according to their class labels. A clear block structure is evident from the figure. White and black values indicate low and high similarity, respectively. However, especially the 4 and 9 class show high in-between class similarity, which is not surprising due to their closeness in shape.

4.10. Interpretability of neural network predictions

A recent trend in deep learning is the development of methods to interpret predictions of neural networks trained with supervised information (Montavon, Lapuschkin, Binder, Samek, & Müller, 2017; Springenberg et al., 2014). However, interpretability is not only a problem in supervised settings. It could be argued that it is even more essential for unsupervised training, where learning is task-driven. One such approach is the guided backpropagation (Springenberg et al., 2014), which allows visualizing pixels in the image that are most influential for a given output decision. We use this technique to visualize what the model learns to recognize MNIST images. Fig. 11 illustrates the results for 10 random

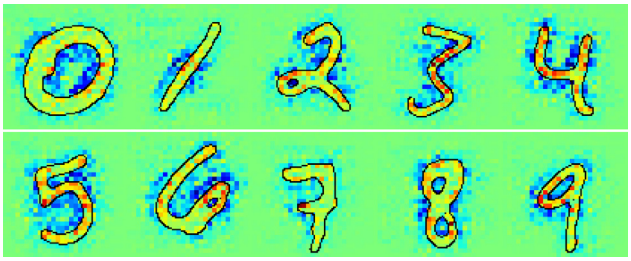


Fig. 11. Interpretability results for a random subset of MNIST examples using the guided backpropagation technique.

numbers of the MNIST dataset. Red pixels indicate pixels positively correlated with the output, in our case the input to the softmax layer (the unnormalized class score) and lowering the value of the red pixels will lead to a reduced class score. We observe that our method focuses on features that are unique for a specific number. This includes, for instance, the loop for the 6 and the top part of the 4. Rendering of the interpretability results overlaying the MNIST number was inspired by Lapuschkin, Binder, Montavon, Müller, and Samek (2016).

5. Conclusion and future work

In this paper, we proposed a novel approach to clustering, dubbed DDC, which (i) takes advantage of the power of deep learning architectures, (ii) is trainable end-to-end in a fully unsupervised way, (iii) does not require pre-training or complex feature design such as HOG (Dalal & Triggs, 2005) and SIFT (Lowe, 2004), and finally (iv) achieves results that outperform or are comparable with state-of-the-art methods on two real-world image datasets and a news story text dataset. We have also evaluated the performance of an ensemble DDC approach, which generally outperformed a single DDC model in the considered benchmarks. Overall, experimental results presented here are promising and stress the importance of unsupervised learning in modern deep learning methods.

In future works, we intend to study the robustness of our method. Further, we will explore alternative loss function formulations, including approaches that are not based on divergence measures and information-theoretic learning. Finally, it would be interesting to explore the use of the proposed method for related clustering settings, such as for instance multi-view clustering (Nie et al., 2018) and constraint clustering (Li et al., 2018).

Acknowledgments

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the GPU used for this research. This work was partially funded by the Research Council of Norway FRIPRO grant no. 239844 on developing the *Next Generation Learning Machines* and IKTPLUSS grant no. 270738 *Deep Learning for Health*. Thanks to the Research Council of Norway (UAVSEAL project, no. 234339) for funding, and the Institute of Marine Research, Norway, and Northwest Atlantic Fisheries Centre, Canada, for providing images and ground truth information about the location of the seal pups in the images.

References

Aggarwal, C. C., & Reddy, C. K. (2013). *Data clustering: Algorithms and applications*. Boca Raton, Florida, US: CRC Press.

Aragam, B., Dan, C., Ravikumar, P., & Xing, E. P. (2018). Identifiability of nonparametric mixture models and Bayes optimal clustering, arXiv preprint arXiv:1802.04397.

Arthur, D., & Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics.

Basseville, M. (2013). Divergence measures for statistical data processing–an annotated bibliography. *Signal Processing*, *93*(4), 621–633. http://dx.doi.org/10.1016/j.sigpro.2012.09.003.

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1798–1828. http://dx.doi.org/10.1109/TPAMI.2013.50.

Bianchi, F. M., Livi, L., & Rizzi, A. (2016). Two density-based k-means initialization algorithms for non-metric data clustering. *Pattern Analysis and Applications*, *19*(3), 745–763.

Bianchi, F. M., Maiorino, E., Kampffmeyer, M. C., Rizzi, A., & Jenssen, R. (2017). An overview and comparative analysis of recurrent neural networks for short term load forecasting, arXiv preprint arXiv:1705.04378.

Bojanowski, P., & Joulin, A. (2017). Unsupervised learning by predicting noise, arXiv preprint arXiv:1704.05310.

Comaniciu, D., & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(5), 603–619.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Computer vision and pattern recognition, 2005. CVPR 2005. IEEE computer society conference on, Vol. 1* (pp. 886–893). IEEE.

Dhillon, I. S., Guan, Y., & Kulis, B. (2004). Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 551–556). ACM.

Dhillon, I. S., Mallela, S., & Kumar, R. (2003). A divisive information-theoretic feature clustering algorithm for text classification. *Journal of Machine Learning Research (JMLR)*, *3*(Mar), 1265–1287.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P. -A., Vincent, P., & Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research (JMLR)*, *11*(Feb), 625–660.

Giryes, R., Sapiro, G., & Bronstein, A. M. (2016). Deep neural networks with random gaussian weights: A universal classification strategy? *IEEE Transactions on Signal Processing*, *64*, 3444–3457.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press, http://www.deeplearningbook.org.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., et al. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672–2680).

Graves, A., Mohamed, A. -r., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing* (pp. 6645–6649). IEEE.

Gregor, K., Danihelka, I., Graves, A., Rezende, D. J., & Wierstra, D. (2015). Draw: A recurrent neural network for image generation, arXiv preprint arXiv:1502.04623.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., & Smola, A. (2012). A kernel two-sample test. *Journal of Machine Learning Research (JMLR)*, *13*(Mar), 723–773.

Han, Y., & Filippone, M. (2016). Mini-batch spectral clustering, arXiv preprint arXiv:1607.02024.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision* (pp. 1026–1034).

Hinton, G. E., Osindero, S., & Teh, Y. -W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*(7), 1527–1554.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift, arXiv preprint arXiv:1502.03167.

Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, *31*(8), 651–666.

Jenssen, R. (2010). Kernel entropy component analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(5), 847–860. http://dx.doi.org/10.1109/TPAMI.2009.100.

Jenssen, R., Erdogmus, D., Hild, K. E., Principe, J. C., & Eltoft, T. (2007). Information cut for clustering using a gradient descent approach. *Pattern Recognition*, *40*(3), 796–806.

Jenssen, R., Principe, J. C., Erdogmus, D., & Eltoft, T. (2006). The Cauchy–Schwarz divergence and Parzen windowing: Connections to graph theory and Mercer kernels. *Journal of the Franklin Institute*, *343*(6), 614–629.

Kampffmeyer, M., Løkse, S., Bianchi, F. M., Jenssen, R., & Livi, L. (2017). Deep kernelized autoencoders. In *Scandinavian conference on image analysis* (pp. 419–430). Springer.

Kampffmeyer, M., Løkse, S., Bianchi, F. M., Livi, L., Salberg, A. -B., & Jenssen, R. (2017). Deep divergence-based clustering. In *Proceedings of the IEEE workshop on machine learning for signal processing*. Tokyo, Japan.

Kampffmeyer, M., Salberg, A. -B., & Jenssen, R. (2016). Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (pp. 1–9).

Kingma, D., & Ba, J. (2014). Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.

Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes, arXiv preprint arXiv:1312.6114.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, *2*(1–2), 83–97.

Lapuschkin, S., Binder, A., Montavon, G., Müller, K. -R., & Samek, W. (2016). The lrp toolbox for artificial neural networks. *Journal of Machine Learning Research (JMLR)*, *17*(114), 1–5, URL http://jmlr.org/papers/v17/15-618.html.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Lee, C. -Y., Xie, S., Gallagher, P., Zhang, Z., & Tu, Z. (2015). Deeply-supervised nets. In *AISTATS, Vol. 2* (p. 6).

Lewis, D. D., Yang, Y., Rose, T. G., & Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research (JMLR)*, *5*(Apr), 361–397.

Li, Z., Nie, F., Chang, X., Nie, L., Zhang, H., & Yang, Y. (2018). Rank-constrained spectral clustering with flexible embedding. *IEEE Transactions on Neural Networks and Learning Systems*.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3431–3440).

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*(2), 91–110.

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., & Frey, B. (2015). Adversarial autoencoders, arXiv preprint arXiv:1511.05644.

Montavon, G., Lapuschkin, S., Binder, A., Samek, W., & Müller, K. -R. (2017). Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, *65*, 211–222.

Myhre, J. N., Mikalsen, K. Ø., Løkse, S., & Jenssen, R. (2018). Robust clustering using a knn mode seeking ensemble. *Pattern Recognition*, *76*, 491–505.

Ng, A. Y., Jordan, M. I., Weiss, Y., et al. (2002). On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, *2*, 849–856.

Nguyen, A., Yosinski, J., & Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 427–436).

Nie, F., Tian, L., & Li, X. (2018). Multiview clustering via adaptively weighted procrustes. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 2022–2030). ACM.

Nie, F., Zeng, Z., Tsang, I. W., Xu, D., & Zhang, C. (2011). Spectral embedded clustering: A framework for in-sample and out-of-sample spectral clustering. *IEEE Transactions on Neural Networks*, *22*(11), 1796–1808.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems (NIPS)*.

Rifai, S., Vincent, P., Muller, X., Glorot, X., & Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th international conference on machine learning (ICML-11)* (pp. 833–840).

Rodriguez, A., & Laio, A. (2014). Clustering by fast search and find of density peaks. *Science*, *344*(6191), 1492–1496.

Sohn, K., Lee, H., & Yan, X. (2015). Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems* (pp. 3483–3491).

Springenberg, J. T. (2015). Unsupervised and semi-supervised learning with categorical generative adversarial networks, arXiv preprint arXiv:1511.0639.

Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2014). Striving for simplicity: The all convolutional net, arXiv preprint arXiv:1412.6806.

Strehl, A., & Ghosh, J. (2002). Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research (JMLR)*, *3*(Dec), 583–617.

Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions, arXiv e-prints abs/1605.02688. URL http://arxiv.org/abs/1605.02688.

Tishby, N., & Slonim, N. (2001). Data clustering by Markovian relaxation and the Information Bottleneck Method. In *Advances in neural information processing systems, Vol. 13* (pp. 640–646).

Vikjord, V. V., & Jenssen, R. (2014). Information theoretic clustering using a k-nearest neighbors approach. *Pattern Recognition*, *47*(9), 3070–3081. http://dx.doi.org/10.1016/j.patcog.2014.03.018.

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. -A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning* (pp. 1096–1103). ACM.

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing*, *17*(4), 395–416.

Xie, J., Girshick, R., & Farhadi, A. (2016). Unsupervised deep embedding for clustering analysis. In *Proceedings of the 33rd international conference on international conference on machine learning, Vol. 48* (pp. 478–487). JMLR.org.

Yan, D., Huang, L., & Jordan, M. I. (2009). Fast approximate spectral clustering. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 907–916). ACM.

Yang, B., Fu, X., Sidiropoulos, N. D., & Hong, M. (2016). Towards k-means-friendly spaces: Simultaneous deep learning and clustering, arXiv preprint arXiv:1610.04794.

Yang, J., Parikh, D., & Batra, D. (2016). Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 5147–5156).

Yang, Y., Xu, D., Nie, F., Yan, S., & Zhuang, Y. (2010). Image clustering using local discriminant models and global integration. *IEEE Transactions on Image Processing*, *19*(10), 2761–2773.

Yu, S. X., & Shi, J. (2003). Multiclass spectral clustering. In *Computer vision, 2003. Proceedings. Ninth IEEE international conference on* (pp. 313–319). IEEE.

Zhang, X. -L. (2018). Multilayer bootstrap networks. *Neural Networks*, *103*, 29–43.