Deep Embedded Non-Redundant Clustering

Lukas Miklautz,[*,1] **Dominik Mautz,**[*,2] **Muzaffer Can Altinigneli,**[1,2] **Christian Böhm,**[2,3] **Claudia Plant**[1,4]
[1]Faculty of Computer Science, University of Vienna, Vienna, Austria
[2]Ludwig-Maximilians-Universität München, Munich, Germany
[3]MCML, [4]ds:UniVie
[1]{lukas.miklautz, claudia.plant}@univie.ac.at
[2]{altinigneli, boehm, mautz}@dbs.ifi.lmu.de

Abstract

Complex data types like images can be clustered in multiple valid ways. Non-redundant clustering aims at extracting those meaningful groupings by discouraging redundancy between clusterings. Unfortunately, clustering images in pixel space directly has been shown to work unsatisfactory. This has increased interest in combining the high representational power of deep learning with clustering, termed deep clustering. Algorithms of this type combine the non-linear embedding of an autoencoder with a clustering objective and optimize both simultaneously. None of these algorithms try to find multiple non-redundant clusterings. In this paper, we propose the novel Embedded Non-Redundant Clustering algorithm (ENRC). It is the first algorithm that combines neural-network-based representation learning with non-redundant clustering. ENRC can find multiple highly non-redundant clusterings of different dimensionalities within a data set. This is achieved by (softly) assigning each dimension of the embedded space to the different clusterings. For instance, in image data sets it can group the objects by color, material and shape, without the need for explicit feature engineering. We show the viability of ENRC in extensive experiments and empirically demonstrate the advantage of combining non-linear representation learning with non-redundant clustering.

1 Introduction

Every day massive amounts of complex data like images, texts, videos and audios are generated and most of them have no labels. This makes it nearly impossible to apply supervised methods, because it may be too expensive to label the data or there might not even be a labeling consensus. This calls for unsupervised classification techniques like clustering, which are unsupervised learning algorithms for partitioning data into similar groups. Unfortunately, these kinds of complex domains have been notoriously difficult to handle for classical clustering algorithms (Xie, Girshick, and Farhadi 2016). The deep learning 'revolution' of the last few years targets these data sets explicitly, which suggests that a combination of both approaches is promising. Several techniques have been proposed in this area under the terms 'deep clustering' or 'embedded clustering'. They combine a flat clustering objective—considering only one valid clustering— with unsupervised representation learning through neural networks (Bengio, Courville, and Vincent 2013; Aljalbout et al. 2018). These approaches make it possible to achieve a high-quality clustering even in the above described domains. The advantage of an integrated approach that performs clustering and representation learning simultaneously is that they benefit from each other. The clustering provides feedback to the transformation, and in return the non-linear transformation can alter the embedded space to improve the clustering. All these methods assume that the data can be partitioned into only a single valid clustering. Yet, we argue that most modern complex data sets can be partitioned in multiple valid ways. Techniques from subspace clustering address these issues, but they deliver a multitude of redundant clustering solutions, which might be valid, but are hard to interpret even for domain experts. Deep subspace clustering algorithms like (Ji et al. 2017; Zhang et al. 2018) can only find a single valid clustering, where a cluster can belong to a different subspace. We think it is necessary to constrain the solution space by only consid-

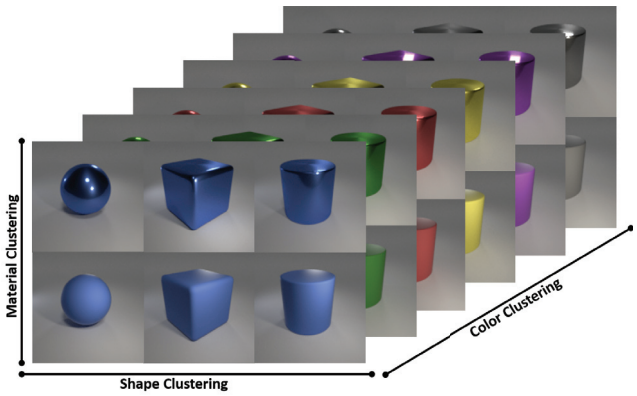


Figure 1: This data set can be meaningfully grouped either by shapes (cube, cylinder, sphere), by colors (red, blue, green, yellow, purple, gray), or by materials (rubber, metal). All of these clusterings are non-redundant.

[*]First authors with equal contribution.

ering clusterings, which are highly non-redundant, meaning that the found clusterings should be as different as possible. Techniques tackling these challenges are not new. Indeed, outside of deep clustering there is a whole range of algorithms for non-redundant clustering (Niu, Dy, and Jordan 2010; Ye et al. 2016; Mautz et al. 2018; Cui, Fern, and Dy 2007).

In Figure 1 we can see examples of rendered objects from a data set, which exhibit different aspects of non-redundancy, such as color, shape or material[1]. A non-redundant clustering objective targets the extraction of these multiple groupings. Applying a deep embedded flat clustering algorithm on the example in Figure 1 would result in only a single clustering that would be even—in a best case scenario—a mixture of shape, color and material. To address the challenge of finding non-redundant clusterings in complex high dimensional data, we propose our novel Embedded Non-Redundant Clustering algorithm (ENRC) that combines the benefits of a non-linear feature transformation with a non-redundant clustering objective. To the best of our knowledge ENRC is the first algorithm that combines these two aspects. The main contributions of the paper can be summarized as follows:

- **Non-redundant clustering layer:** We propose a novel clustering layer, that axis-aligns the different non-redundant structures, captured in the embedded space. Each clustering can have different dimensionalities that are automatically detected.
- **Feature importance:** Existing deep clustering methods have to resort to additional dimensionality reduction techniques like t-sne (Maaten and Hinton 2008) to visualize their results, which might not show a low dimensional representation faithful to the found clustering. In contrast for ENRC each axis-aligned embedded feature has a soft-assignment weight, where a high feature weight indicates that this feature is important for the clustering. This leads to a cluster aware dimensionality reduction for visualization.
- **Joint feature optimization:** Existing non-redundant clustering methods rely on hand engineered features. These features might already predetermine what structures one expects to find, e.g. engineering color features for detecting clusters based on color. We show that the joint optimization of ENRC and an autoencoder is able to to learn the relevant features directly from the data.
- **Preservation of non-redundancy:** To our knowledge we are the first to show empirically that non-redundant structure is preserved in autoencoders and can be recovered by non-redundant clustering algorithms.

2 Embedded Non-Redundant Clustering

2.1 Overview

An autoencoder is an unsupervised (self-supervised) neural network which learns to reconstruct its input. It consists of an encoder $\mathrm{enc}(\cdot)$ network, which embeds the input data in some latent space and a decoder $\mathrm{dec}(\cdot)$ network, which tries to reconstruct the original input from the embedding. To avoid a trivial identity mapping, i.e. simple copying, the latent space dimensionality is often chosen smaller than the input's dimensionality or some other regularization might be used. To explain our ENRC algorithm, we assume some generic pretrained encoder and decoder, which have the non-redundant clustering layer in the middle, as depicted in Figure 2. The novel algorithm finds multiple valid non-redundant clusterings in the embedded space of the autoencoder. To achieve this we introduce two new learnable parameters. A linear transformation matrix V that aligns the structures within each clustering with the axis, acting as an approximation of a rotation. The linearity is sufficient for this task, as the non-linear relationships are already learned by the autoencoder. Additionally, we use feature weights β_s that weigh the importance of each feature for each of the S clusterings acting as a soft separation mechanism of the space. Due to this the standard autoencoder reconstruction loss is adapted to:

$$\mathcal{L}_{\text{rec}} = ||\mathbf{x} - \mathrm{dec}(VV^{\mathsf{T}}\,\mathrm{enc}(\mathbf{x}))||_2^2, \tag{1}$$

where the reconstruction acts as a "non-degeneracy control" (Le et al. 2011) to prevent the matrix V from degeneration.

In the following, we denote with $x[i]$ the i'th entry of vector x and $X[i,j]$ is the value of the i'th row and j'th column of matrix X. Further, we use the weighted squared euclidean distance, defined as

$$||a-b||_\tau^2 := \sum_{i=1}^{D} \tau[i](a[i]-b[i])^2, \tag{2}$$

where $a, b \in \mathbb{R}^D$ are arbitrary vectors—for which we want to measure the distance—and $\tau \in \mathbb{R}^D$ is a weight vector containing weights for each dimension. An overview of other used notations can be found in the Supplementary at https://gitlab.cs.univie.ac.at/lukas/enrcpublic.

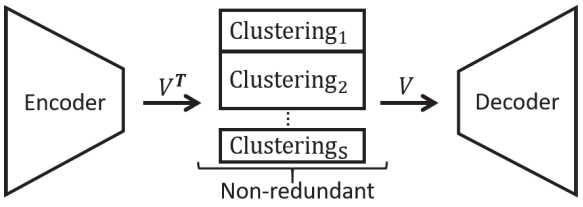


Figure 2: The architecture of ENRC. The linear layer V^{T} aligns the essential structures of the different clusterings along the axes. The clusterings can be of different dimensionality. V maps the clusterings back to the embedded space.

2.2 Feature Weighting

In classical non-redundant subspace clustering each clustering gets its own subspace orthogonal to the other subspaces. We relax this discrete assignment to a continuous weight $\beta_s[d]$ such that each dimension d belongs partially to a clustering s. This relaxation makes the loss function differentiable w.r.t. β. We require that these weights are positive

[1] All figures are best viewed in color.

and for a single dimension the fractions over all clusterings should sum to one:

$$\forall d \in \{1, 2, .., D\} : \sum_{s=1}^{S} \beta_s[d] = 1, \tag{3}$$

where $\forall d \in \{1, 2, .., D\}, \forall s \in S : \beta_s[d] \geq 0$. We can implement these constraints using a soft-max function on a trainable $S \times D$ parameter matrix B, s.t.:

$$\beta_s[d] := \frac{\exp(B[s, d])}{\sum_{i=1}^{S} \exp(B[i, d])}.$$

We can then optimize B without any further constraints. To express the feature importance during clustering, we utilize the βs for the weighted squared euclidean distance.

2.3 Compression Loss

The compression loss 'moves' embedded points $z := \mathrm{enc}(\mathbf{x})$ closer to their centers $\mu_{s,k}$, enhancing the separation between clusters. As we consider multiple non-redundant clusterings, we only want to move points, which are close to their respective centers in each weighted feature space. This is achieved by the following loss function:

$$\mathcal{L}_{\text{comp}} = \frac{1}{S} \sum_{s=1}^{S} \sum_{k=1}^{K_s} \frac{1}{|C_{s,k}|} \sum_{z \in C_{s,k}} ||V^\mathsf{T} z - \mu_{s,k}||^2_{\beta_s}. \tag{4}$$

Note that $\mu_{s,k}$ is the k'th cluster center of clustering s in the already V^T-transformed space. Here $\mu_{s,k}$ is considered a constant (details on its update below) and only V^T, β and z are updated.

2.4 Updating Data Assignment

For given cluster centers, we assign for each clustering s each embedded data point z to the closest center for the β_s weighted euclidean distance in the transformed embedded space:

$$\forall s \in S : \arg \min_{k \in [1; K_s]} ||V^\mathsf{T} z - \mu_{s,k}||^2_{\beta_s} \tag{5}$$

2.5 Updating Cluster Centers

Similar to (Yang et al. 2017) we use an adapted version of mini-batch k-means (Sculley 2010) for updating the cluster centers and assignments. For each incoming data batch, we iterate over the weighted feature spaces and update the clusterings in each space separately. First we update the cluster assignments with the previous cluster centers $\boldsymbol{\mu}^{t-1}_{s,k}$ with Eq. 5. From this new assignments we calculate a per center update for the current mini-batch in the following way:

$$\boldsymbol{\mu}^{t}_{s,k} = \boldsymbol{\mu}^{t-1}_{s,k}(1 - \eta_{s,k}) + \hat{\boldsymbol{\mu}}^{t}_{s,k}\eta_{s,k}. \tag{6}$$

where $\hat{\boldsymbol{\mu}}^{t}_{s,k}$ is the mean of the assigned points from the current mini-batch and $\eta_{s,k}$ is a per cluster learning rate. It is defined as one over the exponential weighted average of past assignments to the center, $\eta_{s,k} = {}^1\!/\!_{A^t_{s,k}}$. The denominator is given by:

$$A^t_{s,k} = (1 - \alpha)A^{t-1}_{s,k} + \alpha \hat{A}^t_{s,k} \tag{7}$$

with $0 < \alpha < 1$ as a discounting parameter and $\hat{A}^t_{s,k}$ as the number of assigned points from the current mini-batch. If a mini-batch does not contain any points for a cluster, the center will not be updated.

2.6 Initialization

At the beginning, we aim to find the initial clustering structures captured in the embedding. Therefore, we only optimize the linear transformation V, the cluster centers $\boldsymbol{\mu}$, and the βs (respectively B), but keep the autoencoder parameters fixed (i.e. $\mathrm{enc}(\cdot)$ and $\mathrm{dec}(\cdot)$). First, we initialize V as a random orthogonal matrix. Next, we soft-assign the first $\lfloor {}^D\!/\!_S \rfloor$ dimensions of the V-rotated space 'strongly' to the first clustering by setting the beta weights of these dimensions and the first clustering to 0.9: $\forall d \in [1 : \lfloor {}^D\!/\!_S \rfloor] : \beta_1[d] := 0.9$. The remaining weight of 0.1 for each of these dimensions is evenly distributed among the other clusterings—reflecting only a 'weak' soft-assignment. Then we assign the next $\lfloor {}^D\!/\!_S \rfloor$ dimensions strongly to the second clustering in the same manner—and so forth. Next, we initialize the cluster centers $\boldsymbol{\mu}$ of each clustering with the initialization procedure of k-means++ (Arthur and Vassilvitskii 2007) using the distance metric in Eq. 2. Finally, we optimize w.r.t. $\mathcal{L}_{\text{comp}}$ and update the cluster centers as described above. To ensure that V does not degenerate, but instead aligns the structural information of each clustering with the axis, we force V to be approximately orthogonal by adding to $\mathcal{L}_{\text{comp}}$ an adapted version of the reconstruction loss:

$$\mathcal{L}_{\text{orth}} = ||\,\mathrm{sg}[z] - VV^\mathsf{T}\,\mathrm{sg}[z]||^2_2, \tag{8}$$

where $\mathrm{sg}[\cdot]$ is the gradient stop iterator such that z is regarded as a constant in this loss term.[2] This differentiable loss avoids degeneration of V by creating an incentive for it to be approximately orthogonal (allowing only rotation and reflection). We only keep this loss term during the initialization phase and drop it in the refinement phase. The latter decision was made based on the results of (Le et al. 2011), which indicate that the reconstruction loss (Eq. (1)) is sufficient for non-degeneracy control. For smaller data sets, one could use alternatively the methods proposed in (Mautz et al. 2018; Cui, Fern, and Dy 2007). This selection of the initial μ's, β's and V is based on the minimal loss and can be performed several times.

2.7 Clustering Phase

As noted in previous work (Yang et al. 2017) the joint optimization of clusterings and embeddings can lead to improved results. The loss function of ENRC combines the reconstruction and compression loss

$$\mathcal{L} = \mathcal{L}_{\text{comp}} + \lambda \mathcal{L}_{\text{rec}} \tag{9}$$

This is motivated by the results of (Guo et al. 2017) that without the space preserving effect of the reconstruction loss, the found clusterings could become meaningless, because $\mathcal{L}_{\text{comp}}$ would be trivially fulfilled if it maps all points into one cluster. The hyperparameter $\lambda > 0$ defines this

[2] Note, that this is equivalent to $||VV^T - I||^2_F$, if z is whitened, see e.g. Lemma 3.1 in (Le et al. 2011).

trade-off, which we set for all experiments to one. During training ENRC updates all parameters of the clustering layer $\beta, V, \boldsymbol{\mu}_{s,i}$ and all weight parameters of encoder and decoder jointly together.

2.8 Cluster Center Reinitialization

Most other recently proposed deep clustering methods do not provide a mechanism for the case when a cluster does not get assigned any points after several mini-batch updates. This can lead to a degradation of performance, as the remaining clusters are merged. This scenario is comparable to the case where a k-means center does not get assigned any data points during the update step. A common strategy—e.g. in the scikit-learn package—is to reinitialize the lost center with a point that is badly represented by its assigned center in terms of euclidean distance. We adapt this strategy by first sampling n_s points from the whole data set and embed them in the rotated feature space. We select the cluster with the highest compression loss and choose the point which is farthest away from this cluster's centroid. After selecting the new centroid, we conduct l k-means update steps in the affected feature space. To avoid unnecessary reinitializations due to unbalanced clusters or small batch sizes we count how often a centroid lost all its points and compare it to a threshold value v, which increases during training by $\lfloor\sqrt{\text{iter}_t}\rfloor$, where iter_t is the current mini-batch iteration count. The latter is a simple heuristic, which accounts for the observation that in the beginning of training, cluster assignments may change more rapidly and become more stable towards the end. After reinitialization, the count for the reinitialized center is set to zero again. The parameters l and n_s should be chosen to fit the computational constraints.

3 Experiments

We evaluated ENRC with four different data sets. As we are the first to address the topic of non-redundant clustering with neural networks, we needed for our benchmark high-dimensional data sets, which are labeled and have enough data points. Note that in real world settings one would use ENRC for exploratory data analysis, without ground truth labels, as we show in Section 3.5. To quantify the ability of finding non-redundant structure in high dimensional data sets, we adapted three commonly used deep learning data sets. Additionally, we use the stickfigures data set, which is often used in the non-redundant clustering literature. Other commonly used data sets from the non-redundant clustering literature contain often less than 500 data points, here stickfigures is already one of the largest, see e.g. (Ye et al. 2016). A summary of the considered data sets is shown in Table 2. We implemented ENRC in Python and trained our networks on a single NVIDIA RTX 2080 Ti. We ran the comparison method on a machine with four Intel(R) Xeon(R) CPU E5-2650 Cores and 32 GB RAM. An implementation of ENRC and all experiments is available at https://gitlab.cs.univie.ac.at/lukas/enrcpublic.

3.1 Data Sets

Concatenated-MNIST We extend the well known MNIST (LeCun et al. 1998) data set by concatenating two digits side by side, resulting in 100 possible combinations, named C-MNIST. With this extension, we show the ability of ENRC to capture positional non-redundancy. This data set can be seen as containing two-digit numbers, from 00 to 99, where each digit (left and right) is independent from the other. To make it easier to use existing convolutional architectures, we added black row wise pixels, so the new format is 56×56 instead of 28×56.

NR-Objects We used a publicly available rendering software[3], which was used in (Johnson et al. 2017), to generate objects with three non-redundant clusterings, called (Non-Redundant) NR-Objects. Each object can be clustered by three shapes, two materials and six colors. Otherwise the default settings, which contained lighting jitter, were kept.

GTSRB The German Traffic Sign Benchmark (GTSRB) data set (Houben et al. 2013) contains real world images of traffic signs, it has been used in object detection literature for self-driving vehicles. We use a subset of 4 different traffic signs 'Speed limit (70km/h)', 'No passing', 'Ahead only', 'Keep right', which can be clustered w.r.t. to the four types of traffic sign and their two colors.

Stickfigures The stickfigures data set contains nine basic objects of dancing figures. It contains three clusterings for the upper and three for the lower body pose, a more detailed explanation can be found in (Ye et al. 2016).

We preprocessed all data sets by a channel-wise z-transformation, to get a zero mean and variance of one. For the GTSRB data set we used additionally histogram equalization, to improve the low contrast in some of the real world images. Examples of each data set can be found in the Supplementary.

3.2 Experimental Setup

For our experiments we use a convolutional autoencoder that utilizes several well-established architectural patterns. By exploiting skip (or identity) connections within convolutional *resnet* blocks as described in (He et al. 2015), we are able to train deeper networks which adapt their depth based on the data set. We applied some additional techniques from (He et al. 2018) to improve our autoencoder performance, e.g. we set the first convolutional layer's kernel size from 7×7 down to 3×3 and compensate the effective receptive field size with additional 3×3 convolutions. Additionally, we start with stride-one convolution instead of stride-two to avoid discarding $\frac{3}{4}$ of the input-image's pixels just within the first layers. We set the architecture parameters, such as the number of feature filters in a convolutional layer, based on the dimensionality and color channels of the data set and set the dropout rate based on the achieved reconstruction error. For all data sets we used the same embedding size of 16, which resulted in good reconstructions. We used this single architecture for all data sets.

For each data set we pretrain ten autoencoders and use them for ENRC and all baseline methods. With this setting we make sure that all methods have the same starting conditions. Similar to (Xie, Girshick, and Farhadi 2016) we pretrain the autoencoder first for 10,000 mini-batch iterations

[3] https://github.com/facebookresearch/clevr-dataset-gen

Table 1: The NMI averages and standard deviations for the ten pretrained autoencoders. Results marked with * had to be run on a subset of 10,000 data points due to memory constraints (>32 GB). Best value in bold.

Data Sets	Clustering	ENRC	Orth1	Orth2	mSC	Nr-Kmeans	ISAAC
NR-Objects	color	**1.00 ±0.00**	0.70 ±0.09	0.73 ±0.06	0.35 ±0.05	0.92 ±0.09	0.15 ±0.06
	material	**1.00 ±0.00**	0.46 ±0.16	0.11 ±0.12	0.03 ±0.07	0.95 ±0.14	0.53 ±0.08
	shape	**1.00 ±0.00**	0.39 ±0.20	0.20 ±0.08	0.03 ±0.03	0.92 ±0.16	0.60 ±0.07
GTSRB	type	**0.74 ±0.01**	0.57 ±0.07	0.73 ±0.15	0.04 ±0.04	0.72 ±0.01	0.60 ±0.07
	color	**0.67 ±0.00**	0.59 ±0.02	0.63 ±0.03	0.04 ±0.06	0.65 ±0.01	0.59 ±0.04
Stickfigures	upper	**1.00 ±0.00**	0.79 ±0.21	0.00 ±0.00	0.33 ±0.20	**1.00 ±0.00**	0.37 ±0.05
	lower	**1.00 ±0.00**	0.77 ±0.24	0.00 ±0.00	0.30 ±0.17	**1.00 ±0.00**	0.39 ±0.08
C-MNIST	left	**0.83 ±0.04**	0.33 ±0.02	0.35 ±0.03	0.07 ±0.02*	0.69 ±0.03	0.29 ±0.13*
	right	**0.82 ±0.01**	0.40 ±0.03	0.41 ±0.04	0.06 ±0.02*	0.70 ±0.03	0.19 ±0.13*

Table 2: Summary of used data sets. The last column shows the number of ground truth clusterings and the corresponding number of clusters.

Name	# Points	# Dimensions	# Clusters
C-MNIST	60,000	3,136	10; 10
NR-Objects	10,000	16,384	6; 3; 2
GTSRB	6,720	1,024	4; 2
Stickfigures	900	400	3; 3

Table 3: The VI averages and standard deviations for the ten pretrained autoencoders. Most methods, were able to find non-redundant clusterings. Results marked with * had to be run on a subset of 10,000 data points due to memory constraints (>32 GB).

Method	NR-Objects	GTSRB	Stickfigures	C-MNIST
ENRC	2.39 ±0.00	1.96 ±0.01	2.20 ±0.00	4.56 ±0.01
Orth1	2.31 ±0.03	1.98 ±0.03	2.06 ±0.17	4.41 ±0.06
Orth2	1.20 ±0.07	1.11 ±0.24	0.90 ±0.56	3.13 ±0.20
mSC	2.32 ±0.05	1.99 ±0.09	2.01 ±0.10	4.48 ±0.05*
NR-Kmeans	2.35 ±0.05	1.94 ±0.00	2.19 ±0.00	4.57 ±0.02
ISAAC	1.25 ±1.04	n.a.	1.14 ±0.25	n.a.*

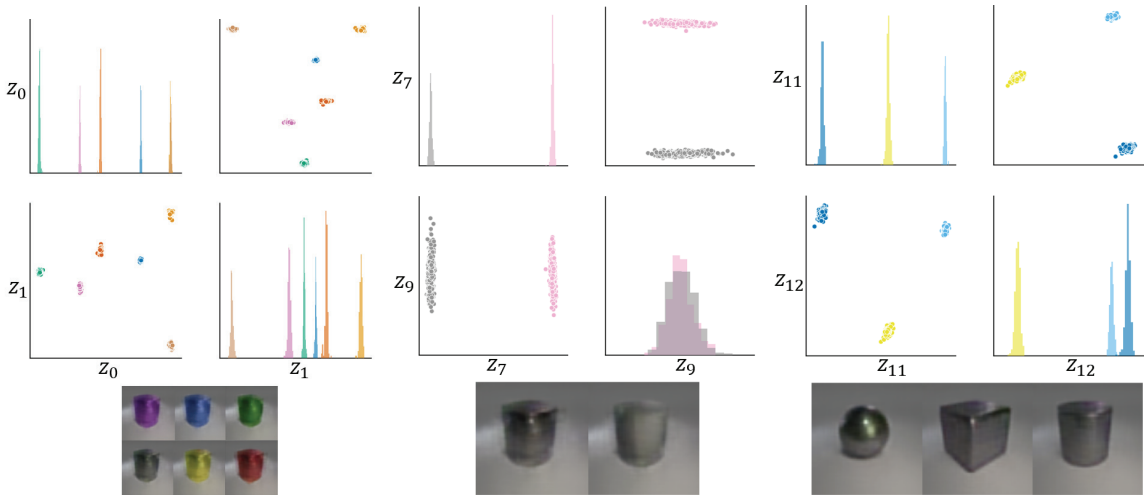
with dropout and then fine tune it for another 5,000 mini-batch iterations with deactivated dropout. For pretraining the autoencoder we use image augmentation (rotation, lighting, zooming), Adam (max-lr $= 0.01$, $\beta_1 = 0.9$, $\beta_2 = 0.99$) (Kingma and Ba 2014) with weight decay of 0.001. We boost the training speed of our network with the *super-convergence* learning rate schedule outlined in (Smith and Topin 2017). The intuition behind this idea is that we start with a low learning rate to warm-up and to slowly find the correct direction on the loss landscape. During the second halve of the training cycle, and close to the end of training process, we decrease the learning rate as we come closer to the optimum. Reaching the maximum learning rate max-lr separates the two training halves and a cosine annealing schedule provides a smooth transition between the two halves of the training cycle. For fine tuning we halve the max-lr to ensure smooth convergence and train for the remaining mini-batch iterations. Further information about the experimental setup and architectural details are presented in the Supplementary and our Python implementation. After pretraining we initialize μ, β and rotation matrix V. For the joint clustering optimization we train for another 20,000 mini-batch iterations, except for the simple stickfigures data set which converged already at 2,000 mini-batch iterations. Similar to (Xie, Girshick, and Farhadi 2016) we decrease the learning rate again and keep decreasing it every 2,000 iterations to ensure a smooth convergence for the joint optimization. We set the initial learning rate for β to max-lr $= 0.01$. This is motivated by the thought that the feature space soft-assignments should be updated faster; before the embedding is linearly transformed by V and clusters are compressed by Eq. 4. The discounting parameter α in Eq. 7, was set to 0.5 giving equal weight to new and old centers. The cluster reinitialization parameters were set to $l = 10$ and $n_s = 1,000$, which ensured a fast re-clustering procedure in case a center got lost. This setting worked well for all considered data sets and shifts most hyperparameter setting decisions of the neural network to the pretraining phase, which is self-supervised (reconstructing the input) for an autoencoder.

3.3 Evaluation Metrics

To evaluate the quality of the found non-redundant clusterings we use the normalized mutual information (NMI) (Vinh, Epps, and Bailey 2010) for each best matching clustering in the found feature spaces, where 1 is a perfect clustering and 0 indicates that no structure was captured. Additionally, we use the average variation of information (VI) (Meilă 2007) for measuring the redundancy of the found feature spaces. The VI calculates the similarity between two different clusterings, where higher values are better. Note that the VI cannot be computed for a single clustering and can only be used to compare different clusterings on the same data set, as the magnitude is data dependent.

3.4 General Results

We compare our ENRC with several state of the art non-redundant clustering algorithms. For each data set we pretrained ten autoencoders and use them for all methods. The



(a) Color (b) Material (c) Shape

Figure 3: The scatter plots show the dimensions z_s with the two highest feature weights β_s for each of the three clusterings. The images below are the reconstructed centers $\mathrm{dec}(V\boldsymbol{\mu}_{s,i})$ of each cluster. Best viewed in color.

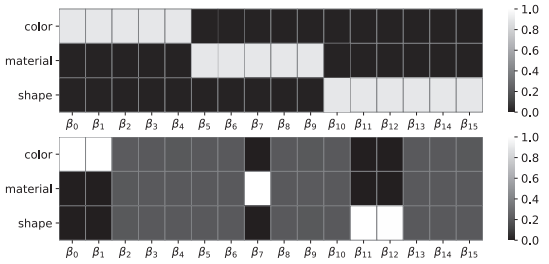


Figure 4: Sorted feature weights β_s for each clustering. **Upper** plot shows initial β_s. **Bottom** plot same feature weights after clustering.

considered comparison methods are Orth1 and 2 (Cui, Fern, and Dy 2007), Nr-Kmeans (Mautz et al. 2018), ISAAC (Ye et al. 2016) and mSC (Niu, Dy, and Jordan 2010). We use these algorithms, because in contrast to subspace clustering algorithms, they have an additional constraint that tries to reduce the redundancy between clusterings. A more detailed discussion can be found in Section 4. Additionally, all of these algorithms are able to find several clusterings in arbitrarily oriented subspaces. This is important, because we assume that the non-redundant clustering structure is preserved in the autoencoder, but the cluster structure does not have to be axis-aligned. For all considered data sets we know the exact number of clusterings and set the methods parameters accordingly. We ran Nr-Kmeans, Orth1 and 2 ten times for each of the ten autoencoders and averaged the best run in terms of their respective loss function. ISAAC and mSC were run once for each autoencoder, due to run time constraints (more than one day). Our experimental results in Table (3) show that, except for ISAAC, each method was able to achieve a quite similar VI. This indicates, that they were indeed able to find several non-redundant clusterings from the learned embeddings of the pretrained autoencoders. Algorithms that found only a single clustering are marked with 'n.a.'. The clustering performance in accordance to the ground truth labels is shown in Table 1. Again we see that all methods are able to recover the non-redundant clusterings from the embedded space of the autoencoder, but only ENRC is able to jointly transform the space, which results in even higher NMI values. For the quite simple stickfigures data set, Nr-Kmeans and ENRC are both able to find the two clusterings perfectly.

3.5 Case Study

In this section we highlight some of ENRC's key strengths by discussing the NR-Objects data set in more detail. In Figure 4 the feature weights β_s before and after training are shown, where higher values indicate stronger memberships. We can see that after training of the 16 dimensional feature space only two are needed for clustering the six colors, another two dimensions for the three shapes and only one for the two materials. The feature weights of the other eleven dimensions are evenly distributed between the three non-redundant clusterings, which indicates that these dimensions are not important for the specific clustering, but encode some shared information. The separation of features, into important and unimportant for the clusterings, allows ENRC to visualize the found embeddings without the need of an additional dimensionality reduction technique like t-sne (Maaten and Hinton 2008). In Figure 3 we can see scatter plots of the two highest β-weighted dimensions for each clustering, indicated by the matching subscripts of the β_s from the bottom plot in Figure 4. On the diagonal axis of the scatter plot are the histograms of each feature. As the material clustering has only one high feature weight β_7 we see that only z_7 contains clustering structure. Since all other feature weights for the material feature space are indifferent between dimensions, we can see here for the second dimension z_9 a unimodal dis-

tribution without clustering structure. For color and shape the feature weights β_s indicate that they need two features for their clustering, which can be seen in the multiple modes of the histograms. Another key benefit of using an autoencoder is that we can use the expressive power of a neural network to find non-redundant clusterings in the feature space, but still keep interpretability by utilizing the decoder. Below each scatter plot in Figure 3 we can see the decoded centers for each clustering as reconstructed images in the original pixel space. Each center of the color feature space is exhibiting only the color feature, but is an average of all other clusterings like shapes and materials, resulting in splodges of color. The same can be seen for material, but here the center for metallic objects is reflective and the one for rubber is not. The two centers appear to be grey because it is an average of all six colors. The latter is true for the three shape clusters as well. While the shape structure is crisp and detailed, the surfaces appear to be greyish and slightly reflective, because the information of the other—non-redundant—clusterings is averaged.

4 Discussion and Related Work

A centroid-based approach like we used in ENRC is quite common in embedded clustering. To the best of our knowledge, none of the proposed embedded clustering methods aim to find multiple non-redundant clusterings within a data set. Algorithms such as DEC (Xie, Girshick, and Farhadi 2016), IDEC (Guo et al. 2017), DMC (Chen, Lv, and Yi 2017) or DEPICT (Ghasedi Dizaji et al. 2017) utilize a pretrained autoencoder to embed the data and a Gaussian or Student-t kernel to softly assign the data points to clusters. DMC directly penalizes the distance of data points embedded to each center weighted by the assignment. The other methods utilize a KL divergence loss between the soft-assignments and an auxiliary probability density function to harden the clusters. All five methods utilize a mini-batch gradient descent optimization scheme as ENRC does. DCN (Yang et al. 2017) is a k-means based method. In contrast to the above described methods, it performs a hard assignment like ENRC, however, it alternates between updating the embedding, the cluster assignments and the cluster centers iterating over the full data set in each step. Recent work in deep subspace clustering, e.g. (Ji et al. 2017; Zhang et al. 2018) find a single clustering, where each cluster can belong to a different subspace. With the work of (Zhang et al. 2018; Fard, Thonet, and Gaussier 2018) we share the idea that hard assignments in different situations can be relaxed with a softmax function. In contrast to these ENRC finds common non-redundant feature spaces for related clusterings. Other recently proposed methods like GMVAE (Dilokthanakul et al. 2016) and VaDE (Jiang et al. 2017) utilize variational approaches or generative adversarial networks, like ClusterGAN (Mukherjee et al. 2019). From all above described method, these last three approaches are the least similar to ENRC. Further algorithms are discussed in two recent survey papers (Aljalbout et al. 2018; Min et al. 2018) that provide a broader overview over proposed embedded clustering techniques. Multiple and non-redundant clustering methods are an active research field and in the classical clustering literature several different methods have been proposed. An overview and different variations can be found in (Müller et al. 2012). Methods like (Chang et al. 2017) assume that the subspaces or views are axis-parallel, however, we assume that the subspaces in the embedded space are arbitrarily-oriented. In the following, we only discuss algorithms that can find non-redundant clusterings in arbitrarily-oriented subspaces. Orth1 and Orth2 (Cui, Fern, and Dy 2007) are two clustering algorithms that extract multiple k-means clustering structures sequentially from spaces orthogonal to the space spanned by the previous cluster centers. The difference between the two versions is that the orthogonal projection can be w.r.t. all clusters or just a single cluster to which a data point is assigned. The main idea of Nr-Kmeans (Mautz et al. 2018) is that the data space can be split into several arbitrarily-oriented orthogonal subspaces and within each subspace the data follows a k-means like clustering. It also allows for an optional noise space without any clustering structure. Like in ENRC the method optimizes all subspaces and the clusterings within simultaneously. Further, it is shown that a parallel clustering extraction can be advantageous compared to a sequential approach used in Orth. ISAAC (Ye et al. 2016) utilizes a two step procedure. First, it uses Independent Subspace Analysis (ISA) to determine the subspaces. Then, it fits a Gaussian mixture model with hard assignments within each subspace. Thereby, all parameters are estimated based on the MDL principle. All four of the above described non-redundant clustering methods have in common that, for each clustering, they find a linear transformation for the respective subspace. In contrast, ENRC utilizes the autoencoder to perform a nonlinear transformation and jointly optimizes the clustering. mSC (Niu, Dy, and Jordan 2010) utilizes the Hilbert-Schmidt independence criterion to find multiple non-redundant views for the relaxed spectral clustering objective. Similar to the autoencoder based embedding, the spectral embedding of mSC can be seen as non-linear, however, its approach is quite different from the one of ENRC.

5 Conclusion

In this paper we proposed the Embedded Non-Redundant Clustering algorithm ENRC, to the best of our knowledge it is the first algorithm that combines an embedded and a non-redundant clustering objective. Its unique characteristics are the joint optimization of multiple non-redundant clusterings together with the non-linear embedding, the intrinsic cluster aware dimensionality reduction and automated feature extraction. Our experiments show that ENRC and its joint training procedure has an advantage over a two step process where the non-linear embedding and non-redundant clustering are separately optimized. This is in accordance with the results shown for flat embedded clustering algorithms. We highlighted some of the benefits of our algorithm ENRC in a case study showing its interpretable results. In future work we want to explore the possibility to incorporate different k-means extensions, such as estimating the number of cluster centers in each clustering. Another interesting direction would be to leverage the concept of non-redundancy for semi-supervised learning.

References

Aljalbout, E.; Golkov, V.; Siddiqui, Y.; and Cremers, D. 2018. Clustering with deep learning: Taxonomy and new methods. *CoRR* abs/1801.07648.

Arthur, D., and Vassilvitskii, S. 2007. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM SODA*, 1027–1035. Society for Industrial and Applied Mathematics.

Bengio, Y.; Courville, A.; and Vincent, P. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35(8):1798–1828.

Chang, Y.; Chen, J.; Cho, M. H.; Castaldi, P. J.; Silverman, E. K.; and Dy, J. G. 2017. Multiple clustering views from multiple uncertain experts. In *Proceedings of the 34th ICML 2017*, 674–683.

Chen, D.; Lv, J.; and Yi, Z. 2017. Unsupervised multi-manifold clustering by learning deep representation. In *Workshops at the 31th AAAI 2017*, 385–391.

Cui, Y.; Fern, X. Z.; and Dy, J. G. 2007. Non-redundant multi-view clustering via orthogonalization. In *Proceedings of the 7th IEEE ICDM, 2007*, 133–142.

Dilokthanakul, N.; Mediano, P. A. M.; Garnelo, M.; Lee, M. C. H.; Salimbeni, H.; Arulkumaran, K.; and Shanahan, M. 2016. Deep unsupervised clustering with gaussian mixture variational autoencoders. *CoRR* abs/1611.02648.

Fard, M. M.; Thonet, T.; and Gaussier, É. 2018. Deep k-means: Jointly clustering with k-means and learning representations. *CoRR* abs/1806.10069.

Ghasedi Dizaji, K.; Herandi, A.; Deng, C.; Cai, W.; and Huang, H. 2017. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *Proceedings of the IEEE ICCV, 2017*, 5736–5745.

Guo, X.; Gao, L.; Liu, X.; and Yin, J. 2017. Improved deep embedded clustering with local structure preservation. In *Proceedings of the 26th IJCAI, 2017*, 1753–1759.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep residual learning for image recognition. *IEEE CVPR, 2015* 770–778.

He, T.; Zhang, Z.; Zhang, H.; Zhang, Z.; Xie, J.; and Li, M. 2018. Bag of tricks to train convolutional neural networks for image classification. In *The IEEE CVPR, 2018*.

Houben, S.; Stallkamp, J.; Salmen, J.; Schlipsing, M.; and Igel, C. 2013. Detection of traffic signs in real-world images: The german traffic sign detection benchmark. In *The 2013 IJCNN*, 1–8. IEEE.

Ji, P.; Zhang, T.; Li, H.; Salzmann, M.; and Reid, I. D. 2017. Deep subspace clustering networks. In *Proceedings of the 30th NeurIPS,2017*, 23–32.

Jiang, Z.; Zheng, Y.; Tan, H.; Tang, B.; and Zhou, H. 2017. Variational deep embedding: an unsupervised and generative approach to clustering. In *Proceedings of the 26th AAAI,19-25, 2017*, 1965–1972. AAAI Press.

Johnson, J.; Hariharan, B.; van der Maaten, L.; Fei-Fei, L.; Lawrence Zitnick, C.; and Girshick, R. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE CVPR, 2017*, 2901–2910.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Le, Q. V.; Karpenko, A.; Ngiam, J.; and Ng, A. Y. 2011. Ica with reconstruction cost for efficient overcomplete feature learning. In *Advances in NIPS, 2011*, 1017–1025.

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

Maaten, L. v. d., and Hinton, G. 2008. Visualizing data using t-sne. *JMLR* 9(Nov):2579–2605.

Mautz, D.; Ye, W.; Plant, C.; and Böhm, C. 2018. Discovering non-redundant k-means clusterings in optimal subspaces. In *Proceedings of the 24th ACM SIGKDD, 2018*, 1973–1982.

Meilă, M. 2007. Comparing clusterings—an information based distance. *J. of multivariate analysis* 98(5):873–895.

Min, E.; Guo, X.; Liu, Q.; Zhang, G.; Cui, J.; and Long, J. 2018. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access* 6:39501–39514.

Mukherjee, S.; Asnani, H.; Lin, E.; and Kannan, S. 2019. Clustergan: Latent space clustering in generative adversarial networks. In *Proceedings of the 33rd AAAI*, 4610–4617.

Müller, E.; Günnemann, S.; Färber, I.; and Seidl, T. 2012. Discovering multiple clustering solutions: Grouping objects in different views of the data. In *Proceedings of the 28th ICDE, Washington, DC, USA, 1-5 April, 2012*, 1207–1210.

Niu, D.; Dy, J. G.; and Jordan, M. I. 2010. Multiple non-redundant spectral clustering views. In *Proceedings of the 27th ICML*, 831–838.

Sculley, D. 2010. Web-scale k-means clustering. In *Proceedings of the 19th WWW*, 1177–1178. ACM.

Smith, L. N., and Topin, N. 2017. Super-convergence: Very fast training of residual networks using large learning rates. *ArXiv* abs/1708.07120.

Vinh, N. X.; Epps, J.; and Bailey, J. 2010. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research* 11(Oct):2837–2854.

Xie, J.; Girshick, R. B.; and Farhadi, A. 2016. Unsupervised deep embedding for clustering analysis. In *Proceedings of the 33rd ICML, 2016*, 478–487.

Yang, B.; Fu, X.; Sidiropoulos, N. D.; and Hong, M. 2017. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *Proceedings of the 34th ICML, 2017*, volume 70, 3861–3870. PMLR.

Ye, W.; Maurus, S.; Hubig, N.; and Plant, C. 2016. Generalized independent subspace clustering. In *Proceedings of the 16th ICDM, 2016, Spain*, 569–578.

Zhang, T.; Ji, P.; Harandi, M.; Hartley, R. I.; and Reid, I. D. 2018. Scalable deep k-subspace clustering. In *ACCV (5)*, volume 11365 of *Lecture Notes in Computer Science*, 466–481. Springer.