Dip-based Deep Embedded Clustering with k-Estimation

Collin Leiber*
LMU Munich & MCML
Munich, Germany
leiber@dbs.ifi.lmu.de

Lena G. M. Bauer*
ds:UniVie
University of Vienna, Vienna, Austria
lena.bauer@univie.ac.at

Benjamin Schelling
Faculty of Computer Science
University of Vienna, Vienna, Austria
benjamin.schelling@univie.ac.at

Christian Böhm
LMU Munich & MCML
Munich, Germany
boehm@dbs.ifi.lmu.de

Claudia Plant
Faculty of Computer Science &
ds:UniVie
University of Vienna, Vienna, Austria
claudia.plant@univie.ac.at

ABSTRACT

The combination of clustering with Deep Learning has gained much attention in recent years. Unsupervised neural networks like autoencoders can autonomously learn the essential structures in a data set. This idea can be combined with clustering objectives to learn relevant features automatically. Unfortunately, they are often based on a k-means framework, from which they inherit various assumptions, like spherical-shaped clusters. Another assumption, also found in approaches outside the k-means-family, is knowing the number of clusters a-priori. In this paper, we present the novel clustering algorithm DipDECK, which can estimate the number of clusters simultaneously to improving a Deep Learning-based clustering objective. Additionally, we can cluster complex data sets without assuming only spherically shaped clusters. Our algorithm works by heavily overestimating the number of clusters in the embedded space of an autoencoder and, based on Hartigan's Dip-test - a statistical test for unimodality - analyses the resulting micro-clusters to determine which to merge. We show in extensive experiments the various benefits of our method: (1) we achieve competitive results while learning the clustering-friendly representation and number of clusters simultaneously; (2) our method is robust regarding parameters, stable in performance, and allows for more flexibility in the cluster shape; (3) we outperform relevant competitors in the estimation of the number of clusters.

CCS CONCEPTS

• **Information systems** → **Clustering**; • **Computing methodologies** → **Cluster analysis**; *Dimensionality reduction and manifold learning*; *Neural networks.*

KEYWORDS

Deep Clustering; Dip-test; Estimating the number of clusters

*First authors with equal contribution.

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467316

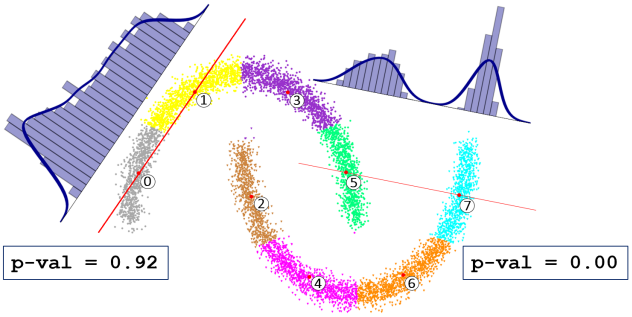


Figure 1: Main idea of our Dip-based clustering. The depicted two moons data set contains two true clusters. We overestimate the number of clusters by executing k-means with $k_{\text{init}} = 8$. If we draw a straight line between two k-means centroids (red) and project the data points assigned to either one of the clusters onto this line, the Dip-test can be applied to the projected points. The blue histograms show the distribution of the projected points of clusters 0 and 1 on the left side and clusters 5 and 7 on the right side. High Dip-p-values indicate unimodality. Thus, according to the Dip-test, clusters 0 and 1 should be merged, while 5 and 7 should not.

ACM Reference Format:
Collin Leiber, Lena G. M. Bauer, Benjamin Schelling, Christian Böhm, and Claudia Plant. 2021. Dip-based Deep Embedded Clustering with k-Estimation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467316

1 INTRODUCTION

Finding patterns in large amounts of unlabelled data is one of the major data mining research branches. The goal is to partition the data into groups of similar data points. However, in practice, it is often not known how many clusters there are.

There is a (wide) range of potential methods for traditional clustering algorithms to address this problem. Many of them are based on the k-means framework like X-means [23] or Dip-means [16]. Some approaches like PG-Means [8] are EM-based, which allows for more flexibility in the cluster shape. However, in these frameworks, they automatically inherit the Gaussian cluster assumption. While

this might be accurate for some data sets, it is too restrictive for others, resulting in an arbitrary clustering. There are, of course, other clustering approaches that are able to determine the number of clusters in a data set automatically and are not limited to a Gaussian cluster shape. One of the best known is the density-based method DBSCAN [7]. It can, just like some variants of Spectral Clustering-based approaches [31], estimate the number of clusters and is very flexible regarding the cluster shape. These approaches, however, trade a comparably easy to understand parameter - the number of clusters - for more complicated parameters (e.g., the neighbourhood range or the number of neighbours). The detected number of clusters is, to a large extent, controlled by these parameters. Thus, these methods substitute one parameter with others.

The major drawback for all of the mentioned methods is that their performance becomes unsatisfactory for modern data sets consisting of large and high dimensional data such as images, videos, and text. While run time and memory issues could be resolved with high-performance implementations, other problems such as the curse of dimensionality remain since many methods are based on the Euclidean distance. The trend for these types of data sets has become to cluster them with Deep Learning (DL) approaches. For this task, most methods use an autoencoder to learn a cluster-friendly lower-dimensional representation of the data such that the clustering can be executed in sufficient run time and to overcome the curse of dimensionality. Thus, methods to estimate the correct number of clusters should ideally be integrated into these types of approaches such that they are compatible and can exploit the benefits of DL.

Until now, to the best of our knowledge, there is no DL-based method to cluster a data set while simultaneously estimating the correct number of clusters. All existing strategies are built on traditional clustering approaches that do not scale well to high-dimensional and/or large data sets.

To address the above problems, we propose **DipDECK** - **Dip**-based **D**eep **E**mbedded **C**lustering with **k**-estimation. It simultaneously improves the estimation of the number of clusters k, the cluster assignments, and the data embedding. We overestimate the number of clusters in the embedded space of an autoencoder by k_{init} and use Hartigan's Dip-test [12], a statistical test for modality in one-dimensional samples, to identify clusters that share structural similarities. It returns a Dip-value, which translates to a p-value that is equal to the probability of a sample's unimodality. We define a clustering loss that forces the autoencoder to push clusters sharing a high Dip-p-value together, resulting in compact cluster shapes. These can then be combined into a common cluster. We induce the assumption that $k \leq k_{\text{init}}$. However, this is a significant relaxation compared to when k needs to be given as a fixed value. Fig. 1 serves as an illustration of the idea behind our Dip-based clustering. Assume the depicted data points represent a two-dimensional autoencoder embedding of a higher-dimensional data set. We can see how the Dip-test indicates which sub-clusters are connected. Additionally, it shows that we can even use this strategy to identify non-convex cluster shapes.

Our contributions can be summarised as follows:

- We introduce a novel deep clustering method that operates oblivious of the correct number of clusters k in the data. Despite being at the disadvantage of not knowing k, we achieve competitive clustering results regarding NMI on a wide range of benchmark and additional data sets.
- Although partly centroid-based by implicitly taking a k-means like loss term into account, our method is more flexible regarding the cluster shape.
- We exploit the Dip-test, a statistical test for unimodality, which quantifies the structure within a data set. The Dip-test is introduced for the first time to DL.
- Since we outperform relevant competitors in estimating the number of clusters on various data sets, our method can also be used merely to estimate this value. This estimation can then be used in combination with other clustering methods.

2 RELATED WORK

We introduce DipDECK, a method for k-estimation in a DL context. Thus, we have two relevant research branches: DL approaches for clustering and estimation methods for k. Since our method is based on the Dip-test, we also describe it in more detail in this section.

Deep Clustering. The first branch consists of DL methods created for clustering. For this, we focus on the essential and foundational methods which have been established as some of the most important Deep Clustering (DC) methods. These include DEC [28], IDEC [10] and DCN [29], which are all centroid-based approaches without excessive tuning of the autoencoder or exploitation of domain knowledge such as augmentation for images. These are closely relatable to us; the same autoencoder can be used for comparisons, and they also have a degree of kinship to k-means. VaDE [15] embeds the probabilistic clustering problem into a variational autoencoder framework. Although this strategy is different to the other mentioned methods, there is still a relation because it models the data generative procedure by a Gaussian Mixture Model and is therefore also Gaussian-based. Other DC approaches like ClusterGAN [22], JULE [30], DEPICT, [9] or DTI [21] are already rather distant or partly use extensive techniques (e.g. CNNs or augmentation). In our work, we do not focus on topics such as autoencoder architecture optimisation or exploitation of domain knowledge in, e.g., the form of augmentation. These directions of research are orthogonal to our approach of optimising the clustering objective and simultaneous k-estimation.

Estimating k. The second, more important, branch is k- estimation methods. Most of them are based on k-means and proceed in the following way: They start with a low initial number of clusters k_{init} (usually a single cluster) and then apply a criterion to determine whether a cluster should be split. Thus, their most significant distinction is the applied criterion, for which they use, e.g., the Bayesian Information Criterion (X-means [23]), the Dip-test (Dip-means [16], respectively its continuation pDip-means [3]), Anderson-Darling hypothesis test (G-means [11]) or the Kolmogorov-Smirnov test (PG-means [8]). They either stop determined by their criterion, or the value of the maximal accepted k_{max} is reached. Some approaches (e.g. [1]) start with k_{max} and merge clusters until the termination condition is reached, i.e. the applied criterion is satisfied, or k equals k_{min} (usually a single cluster). Our approach also follows the route of merging clusters to find a suitable value of k. The advantage of starting with a large k is that each micro-cluster can preserve some

Table 1: Description of the used symbols.

Symbol	Interpretation
$d \in \mathbb{N}$	Dimensionality of the original feature space
$m \in \mathbb{N}$	Dimensionality of the embedded space
$k \in \mathbb{N}$	Number of clusters
$X \subseteq \mathbb{R}^d$	Set of all objects
$C_i \subseteq X$	Objects assigned to cluster i
$\mathcal{B} \subseteq X$	A mini-batch
$\mu_i \in C_i$	Cluster centre of cluster i
$\mu_i^{\text{km}} \in \mathbb{R}^m$	$i-$th k-means cluster centre
$x \in X$	A single object of the data set
$d_{i,j} \in [0, 0.25]$	Dip-value of the combined clusters i and j
$p_{i,j} \in [0, 1]$	Dip-p-value of $d_{i,j}$ (0 if $d_{i,j} \approx 0.25$; 1 if $d_{i,j} \approx 0$)
$T \in [0, 1]$	Dip-p-value threshold

underlying structure in the embedded space. With our approach, the shape of the Dip-connected micro-clusters is - with the help of the autoencoder - transformed, and they start to merge depending on the Dip-test until we have only clearly separated clusters transformed into roughly spherical shapes.

Another research branch covers postprocessing methods that can be used to enhance an existing clustering solution. An example is RIC [2] which utilises MDL to purify and improve the given clustering structures. This improvement includes a refinement of the number of clusters. These methods are, however, orthogonal to our goal of optimising the initial clustering objective.

The only DC-related approach that includes a strategy for cluster number estimation is SCDE [6]. However, they use an additional softmax autoencoder network for the cluster estimation and do not integrate the estimation as an automated process during DC. Instead, they use the estimated amount of clusters to execute Spectral Clustering on the embedded samples. Furthermore, the embedded space dimension of this additional softmax network is systematically chosen as twice the number of ground truth clusters - precisely the number we are trying to estimate. Another DC approach that does not need k as an input parameter is DeepECT [20]. In contrast to SCDE and DipDECK, DeepECT does not estimate a concrete k but returns a hierarchical clustering structure which can be further analysed to choose an appropriate number of clusters afterwards.

Dip-test. In order to identify coherent patterns, our approach makes use of the Dip-test [12]. The Dip-test is a statistical test developed in the 1980s by Hartigan and Hartigan to measure modality. It returns a Dip-value $Dip \in [0, 0.25]$, which, when close to zero, indicates unimodality. The corresponding Dip-p-value, which indicates how likely it is that a sample set is unimodal, is then close to 1. Larger Dip-values indicate that the data set contains at least two modes, which will yield a Dip-p-value almost equal to zero. The advantages are the run time and the fact that it is parameter-free. It has, until now, never been used in DL. There are some approaches in traditional clustering that make use of it [16], and it has recently gained interest in the data mining community [3, 19, 24, 25].

3 DIP-BASED DEEP EMBEDDED CLUSTERING

In this section, we describe our method Dip-based Deep Embedded Clustering with k-estimation (DipDECK). It utilises an autoencoder [17] to simultaneously estimate the number of clusters and define

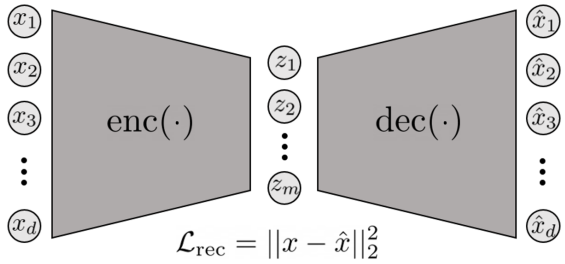


Figure 2: Autoencoder network architecture. The clustering is performed on the data points $z = \mathbf{enc}(x)$.

cluster assignments in the embedded space. All symbols used in this work are described in Table 1.

An autoencoder is a two-part unsupervised neural network that consists of an encoder and a decoder network. The encoder embeds the input data into a latent, usually lower-dimensional space. The decoder, on the other hand, tries to reconstruct the embedded data into its original state. The autoencoder can learn the properties of the embedded space by minimising the reconstruction loss $\mathcal{L}_{rec}$. Usually, the mean squared error is used for this. For a mini-batch $\mathcal{B}$, this loss reads as follows:

$$\mathcal{L}_{rec} = \frac{1}{|\mathcal{B}|} \sum_{x \in \mathcal{B}} ||x - \text{dec}(\text{enc}(x))||_2^2, \tag{1}$$

where enc$(\cdot)$ describes the data after applying the encoder, dec$(\cdot)$ is the result of the decoder, and $||\cdot||_2^2$ denotes the squared Euclidean distance. Fig. 2 illustrates the autoencoder architecture.

In general, our method requires two main parameters: First, an initial number of clusters k_{init}, which should be significantly larger than the expected value, and a threshold T for the Dip-p-value, which determines whether two clusters should be merged.

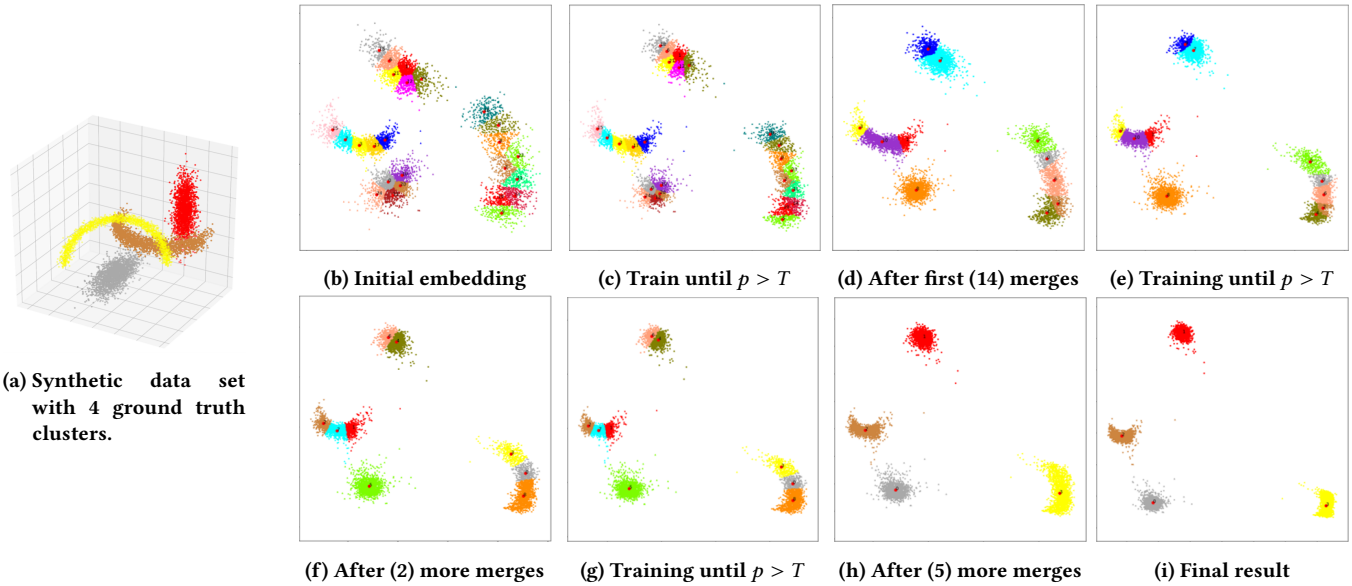
In our experiments, we use a simple feed-forward network architecture. However, other domain-specific architectures may be used as well. Afterwards, we execute k-means in the embedded space with the overestimated number of clusters to get the initial cluster centres and cluster assignments. Since we want to optimise the positions of the cluster centres simultaneously to the data embedding, we use the objects within the data set that are closest to the k-means centres (μ^{km}) as actual cluster centres.

$$\mu_i = \arg\min_{x \in C_i} \left(\left\| \text{enc}(x) - \mu_i^{\text{km}} \right\|_2^2 \right) \tag{2}$$

We apply the Dip-test to obtain the Dip-values for each pairwise combination of clusters i and j in the embedded space. Since the input of the Dip-test must be one-dimensional, we use the dot product $(\cdot)$ to project each point assigned to either one of the two clusters onto the connection line of the corresponding centres μ_i and μ_j. Normalisation is not necessary because the Dip-test is scale-invariant.

$$C_{i,j}^{1d} = \{\text{enc}(x) \cdot (\text{enc}(\mu_i) - \text{enc}(\mu_j)) \mid x \in C_i \cup C_j\}$$

This one-dimensional ($1d$) data set can then be used to calculate the Dip-value and consequently the Dip-p-value. Fig. 1 illustrates this idea.



(a) Synthetic data set with 4 ground truth clusters.

(b) Initial embedding (c) Train until $p > T$ (d) After first (14) merges (e) Training until $p > T$

(f) After (2) more merges (g) Training until $p > T$ (h) After (5) more merges (i) Final result

Figure 3: (a) A synthetic 3d data set to visualise our method. (b) k-means is executed on the 2d-embedding with $k_{\text{init}} \gg k_{\text{true}}$. (c) The autoencoder is trained until a Dip-p-value is found larger than the threshold T leading to (d) merging of clusters. (e) Training resumes, and the autoencoder repositions the clusters according to their Dip-p-value until T is reached again and (f) merging is instantiated. This process repeats itself in (g) and (h) until training finds no more clusters which need to be merged. The final result i) shows four almost perfectly separated clusters (NMI=0.99) in the embedded space.

The Dip-test might sometimes identify two groups as unimodal even though there is a large spatial distance between them if they differ greatly in size. To account for this, we calculate a second Dip-value that includes only the closest points of the larger cluster to the centre of the smaller cluster and the complete smaller cluster. Ultimately, the larger of the two Dip-values (and therefore the smaller Dip-p-value) is used. The intuition is that the transition between the clusters must also be unimodal. This idea is described in detail in Appendix C.

Furthermore, we define that each cluster with itself receives a Dip-value of 0 and therefore a Dip-p-value of 1. Since $p_{i,j} = p_{j,i}$, we get the following symmetric Dip-p-value matrix:

$$P = \begin{pmatrix} 1 & p_{1,2} & \cdots & p_{1,k} \\ p_{2,1} & 1 & \cdots & p_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ p_{k,1} & p_{k,2} & \cdots & 1 \end{pmatrix}$$

We normalise P by dividing each entry of the i-th row by the sum of the respective i-th row. The resulting matrix is termed $\hat{P}$.

Now we can start to optimise the autoencoder in a mini-batch fashion. We first encode all contained objects and all the cluster centres for each batch $\mathcal{B}$ in the data set. Then, we update the cluster assignments in a k-means fashion by assigning each point to the closest centre.

We define our novel clustering objective $\mathcal{L}_{clu}$ as

$$\mathcal{L}_{clu} = \frac{(1 + \text{std}(D_C))}{\text{mean}(D_C)} \frac{1}{|\mathcal{B}|} \sum_{x \in \mathcal{B}} \sum_{i=1}^{k} \hat{P}_{c_x,i} ||\text{enc}(x) - \text{enc}(\mu_i)||_2^2, \quad (3)$$

where c_x is the label of the cluster x is assigned to, std$(\cdot)$ and mean$(\cdot)$ are the standard deviation and the mean of a set, respectively. D_C is the set of Euclidean distances between all cluster centres,

$$D_C = \left\{ \sqrt{||\text{enc}(\mu_i) - \text{enc}(\mu_j)||_2^2} \mid i \in [1, k-1] \text{ and } j \in [i+1, k] \right\}.$$

The row-wise normalisation of P ensures that the innermost sum of Eq. 3 is an affine sum. This means that the weights $\hat{P}_{c_x,i}$ sum up to 1 over all clusters i for a fixed x. The intuition is to force the network to strongest push a data point enc(x) to the centre it is assigned to - because 1 is the maximal possible Dip-p-value- but also proportionately pushing it in the direction of those centres μ_i whose points - according to the Dip-test- share a unimodal distribution with the points of the cluster of enc(x). Since the respective Dip-p-value $\hat{P}_{c_x,i}$ will then be large, the network will try to minimise the loss by reducing the respective distances to these centres.

The division by mean(D_C) is used to hinder the autoencoder of just reducing the embedding scale in order to minimise $\mathcal{L}_{clu}$. However, mean(D_C) could still be kept sufficiently large even if the network reduces the scale if it simultaneously pushes individual clusters far away. To prevent this, we include the term std(D_C).

Finally, we calculate $\mathcal{L}_{rec}$ using Eq. 1 and define the full loss function of DipDECK as

$$\mathcal{L} = \mathcal{L}_{rec} + \mathcal{L}_{clus}.$$

After each epoch, we update all labels by assigning each embedded point to its closest embedded centre to acknowledge the new autoencoder structure. Furthermore, we update the cluster centres similar to Eq. 2, but this time, we choose the points closest to the embedded cluster means instead of the k-means centres.

Algorithm 1: Pseudocode of DipDECK

Input: data set X, starting number of clusters k_{init}, Dip-p-value threshold T, number of epochs n
Output: *labels*, k

$k = k_{\text{init}}$
AE = pretrained autoencoder
(*kmCentres*, *labels*) = K-Means(AE.encode(X), k)
centres = find closest points to *kmCentres* (Eq. 2)
DipMatrix = calculate pairwise Dip-p-values of the clusters in the embedded space
$i = 0$
while $i < n$ **do**
  for $\mathcal{B}$ ***in*** X **do**
    if $i \neq 0$ **then**
      update *labels* of $\mathcal{B}$
    calculate $\mathcal{L} = \mathcal{L}_{rec}$ (Eq. 1) + $\mathcal{L}_{clu}$ (Eq. 3) for $\mathcal{B}$
    optimise AE using $\mathcal{L}$
  update all *labels*, *centres* (Eq. 4) and the *DipMatrix*
  i++
  // Start merging process
  while *max(DipMatrix)* $\geq T$ **do**
    k--
    merge clusters with highest Dip-p-value → add the new centre (Eq. 5) to *centres* and overwrite *labels*
    update the *DipMatrix*
    $i = 0$
return *labels*, k

$$\mu_i = \underset{x \in C_i}{\arg\min} \left(\left\| \text{enc}(x) - \frac{1}{|C_i|} \sum_{y \in C_i} \text{enc}(y) \right\|_2^2 \right) \tag{4}$$

Afterwards, the Dip-p-value matrices P and $\hat{P}$ are updated, and the cluster merging process starts.

3.1 Merging Process

To merge two clusters, we first check whether the maximum Dip-p-value in P is larger than a specified Dip-p-value threshold T. If so, the number of clusters will be reduced by one and the two clusters i and j that produce the corresponding Dip-p-value will be merged. Therefore, we assign the same label to all points assigned to these clusters, and a new centre will be created by using the closest point to the weighted mean of the old centres.

$$\mu_{new} = \underset{x \in C_i \cup C_j}{\arg\min} \left(\left\| \text{enc}(x) - \frac{|C_i|\text{enc}(\mu_i) + |C_j|\text{enc}(\mu_j)}{|C_i| + |C_j|} \right\|_2^2 \right) \tag{5}$$

With this new centre, we can then update the Dip-p-value matrices P and $\hat{P}$. In the end, we again check if the maximum Dip-p-value in P is larger than our threshold. If this is the case, we repeat the merging process. Otherwise, we reset the epoch counter and start optimising the autoencoder. In the first following epoch, we will not update the labels of the batches to allow the autoencoder to adjust to the new cluster structures and compress clusters that occupy a large space due to the prior merging.

Algorithm 1 shows the complete procedure of DipDECK.

Fig. 3 shows the procedure applied to a three-dimensional data set with four true clusters - two arbitrarily oriented moons and two arbitrarily oriented elongated Gaussian clusters. While this data set could probably be clustered successfully without the help of an autoencoder (e.g., with Spectral Clustering), it serves as a good illustration example to follow our clustering process in a visualisable two-dimensional embedded space.

4 EXPERIMENTAL EVALUATION

We evaluate our approach regarding several aspects. In Sec. 4.1, we compare DipDECK to various k-estimation methods while also considering the quality of the resulting clustering. Additionally, we compare the performance of related DC approaches. We then explore the interpretability of our found clusters in more detail (Sec. 4.2). In several robustness tests, we investigate the influence of our parameters on the results (Sec. 4.3).

Evaluation Metrics. For the quantitative evaluation, we consider the estimated number of clusters k and the normalised mutual information (NMI) [26]. NMI ranges in $[0, 1]$, where 1 indicates a perfect and 0 an arbitrary result. It is commonly used in the evaluation of unsupervised tasks.

Data sets. We conduct experiments on the image data sets USPS [14], MNIST [18], Fashion-MNIST (F-MNIST) [27], Kuzushiji-MNIST (K-MNIST) [4], Optdigits [5] and GTSRB [13], as well as the numerical data sets Pendigits [5] and Letterrecognition [5]. A detailed description of the data sets can be found in the Appendix A. All image data sets are reshaped into a one-dimensional vector and preprocessed by a channel-wise z-transformation. The numerical data sets are preprocessed using a feature-wise z-transformation.

Experimental Setup. We set the autoencoder dimensions to d-500-500-2000-m-2000-500-500-d. This is equal to the settings described in [28]. For the Dip-test, we have to project the data from dimension m to 1. The larger m, the more information gets lost due to this projection. Therefore, we choose $m = 5$. Furthermore, we use the ADAM optimiser and a constant learning rate of 0.001 for the pre-training as well as 0.0001 for the actual clustering process in all experiments. We set the initial k_{init} in all cases to 35, as this value is an overestimation for all correct k in the considered data sets. The batch size is set to 256, the Dip-p-value threshold to 0.9, the number of epochs for the pre-training to 100 and for the clustering process to 50. Our implementation of DipDECK is based on PyTorch (https://pytorch.org/) and can be downloaded at https://dmm.dbs.ifi.lmu.de/downloads.

We use the same parameters as for DipDECK if possible for the DC algorithms DEC, IDEC, DCN and VaDE. One exception is m, which we set to 10 since this is the dimensionality used in their respective papers. Additionally, we set the number of epochs for the clustering process to 150. All algorithm-specific parameters, including those for the non-DC algorithms, are set using the recommended values specified in the respective papers. For the algorithms X-means, G-means, PG-means, Dip-means, pDip-means in combination with an autoencoder (AE+), we use the same network architecture as for DipDECK, including $m = 5$. As these methods

Table 2: The resulting NMI values of various methods on different data sets. Additionally, for algorithms that can determine the number of clusters, this value is given. Best estimation of clusters and best NMI are highlighted in bold. Values marked with † could either not be executed due to memory constraints or aborted after 24h. All results are mean ± std of 10 executions.

Method	USPS		MNIST		F-MNIST		K-MNIST	
	k	NMI	k	NMI	k	NMI	k	NMI
Ground truth	10	-	10	-	10	-	10	-
DipDECK (ours)	**9.4 ± 0.47**	**0.846 ± 0.02**	**11.2 ± 0.50**	**0.889 ± 0.01**	**12.2 ± 0.75**	**0.679 ± 0.01**	15.8 ± 0.94	**0.658 ± 0.01**
X-means	35.0 ± 0.00	0.607 ± 0.01	35.0 ± 0.00	0.551 ± 0.00	35.0 ± 0.00	0.512 ± 0.00	35.0 ± 0.00	0.505 ± 0.00
G-means	35.0 ± 0.00	0.608 ± 0.00	35.0 ± 0.00	0.550 ± 0.00	35.0 ± 0.00	0.511 ± 0.00	35.0 ± 0.00	0.503 ± 0.00
PG-means	2.4 ± 0.63	0.136 ± 0.07	2.1 ± 0.79	0.175 ± 0.09	4.1 ± 1.93	0.312 ± 0.11	2.4 ± 0.76	0.135 ± 0.05
Dip-means	4.0 ± 0.00	0.438 ± 0.00	†	†	†	†	†	†
pDip-means	35.0 ± 0.00	0.617 ± 0.01	35.0 ± 0.00	0.554 ± 0.00	35.0 ± 0.00	0.511 ± 0.00	35.0 ± 0.00	0.502 ± 0.00
AE+X-means	2.0 ± 0.00	0.293 ± 0.01	18.1 ± 18.1	0.620 ± 0.10	24.4 ± 3.57	0.570 ± 0.01	2.1 ± 0.29	0.072 ± 0.05
AE+G-means	35.0 ± 0.00	0.669 ± 0.01	35.0 ± 0.00	0.686 ± 0.01	35.0 ± 0.00	0.550 ± 0.01	35.0 ± 0.00	0.569 ± 0.01
AE+PG-means	3.9 ± 1.24	0.379 ± 0.12	3.6 ± 1.22	0.453 ± 0.10	2.8 ± 0.71	0.387 ± 0.05	2.6 ± 0.76	0.103 ± 0.07
AE+Dip-means	6.8 ± 0.57	0.617 ± 0.02	†	†	†	†	†	†
AE+pDip-means	4.9 ± 1.31	0.519 ± 0.07	8.0 ± 1.60	0.705 ± 0.04	5.8 ± 0.57	0.522 ± 0.01	**12.4 ± 3.05**	0.474 ± 0.09
DEC	-	0.805 ± 0.02	-	0.847 ± 0.01	-	0.607 ± 0.03	-	0.551 ± 0.01
IDEC	-	0.811 ± 0.02	-	0.867 ± 0.01	-	0.641 ± 0.02	-	0.553 ± 0.02
DCN	-	0.748 ± 0.02	-	0.844 ± 0.02	-	0.617 ± 0.02	-	0.522 ± 0.04
VaDE	-	0.749 ± 0.03	-	0.806 ± 0.03	-	0.642 ± 0.02	-	0.558 ± 0.01

	Optdigits		Pendigits		Letterrecognition		GTSRB	
	k	NMI	k	NMI	k	NMI	k	NMI
Ground truth	10	-	10	-	26	-	5	-
DipDECK (ours)	**10.4 ± 0.76**	0.858 ± 0.02	14.0 ± 0.85	**0.817 ± 0.00**	**20.6 ± 1.80**	**0.491 ± 0.03**	**5.2 ± 0.87**	**0.612 ± 0.06**
X-means	35.0 ± 0.00	0.709 ± 0.01	35.0 ± 0.00	0.703 ± 0.01	35.0 ± 0.00	0.406 ± 0.01	35.0 ± 0.00	0.426 ± 0.01
G-means	35.0 ± 0.00	0.715 ± 0.01	35.0 ± 0.00	0.702 ± 0.01	35.0 ± 0.00	0.404 ± 0.01	35.0 ± 0.00	0.420 ± 0.01
PG-means	1.1 ± 0.29	0.024 ± 0.07	3.0 ± 1.41	0.335 ± 0.18	1.7 ± 0.61	0.068 ± 0.06	†	†
Dip-means	1.0 ± 0.00	0.000 ± 0.00	**10.4 ± 0.47**	0.691 ± 0.02	1.0 ± 0.00	0.000 ± 0.00	1.0 ± 0.00	0.000 ± 0.00
pDip-means	35.0 ± 0.00	0.709 ± 0.01	35.0 ± 0.00	0.705 ± 0.01	35.0 ± 0.00	0.418 ± 0.01	35.0 ± 0.00	0.427 ± 0.01
AE+X-means	12.8 ± 1.40	0.804 ± 0.01	35.0 ± 0.00	0.719 ± 0.01	2.0 ± 0.00	0.104 ± 0.02	2.2 ± 0.38	0.296 ± 0.01
AE+G-means	35.0 ± 0.00	0.730 ± 0.01	35.0 ± 0.00	0.711 ± 0.01	35.0 ± 0.00	0.476 ± 0.01	35.0 ± 0.00	0.455 ± 0.01
AE+PG-means	2.6 ± 1.61	0.290 ± 0.13	4.9 ± 1.93	0.534 ± 0.09	1.8 ± 0.93	0.048 ± 0.07	1.4 ± 0.47	0.048 ± 0.07
AE+Dip-means	1.0 ± 0.00	0.000 ± 0.00	9.4 ± 0.63	0.689 ± 0.01	1.0 ± 0.00	0.000 ± 0.00	1.0 ± 0.00	0.000 ± 0.00
AE+pDip-means	1.0 ± 0.00	0.000 ± 0.00	**10.4 ± 1.36**	0.695 ± 0.03	1.0 ± 0.00	0.000 ± 0.00	1.0 ± 0.00	0.000 ± 0.00
DEC	-	**0.885 ± 0.02**	-	0.763 ± 0.01	-	0.404 ± 0.03	-	0.555 ± 0.02
IDEC	-	0.864 ± 0.02	-	0.753 ± 0.01	-	0.443 ± 0.03	-	0.577 ± 0.06
DCN	-	0.857 ± 0.02	-	0.723 ± 0.02	-	0.450 ± 0.02	-	0.528 ± 0.04
VaDE	-	0.743 ± 0.04	-	0.718 ± 0.02	-	0.108 ± 0.02	-	0.200 ± 0.02

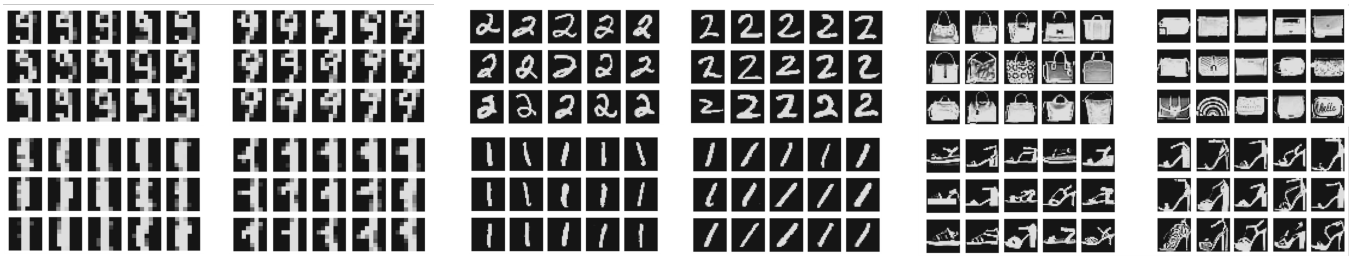
expect a maximal value of k, we set it to 35 to allow a fair comparison with our method, which starts with $k_{\text{init}} = 35$. We can not compare to SCDE because no source code is available.

4.1 Quantitative Experiments

The evaluation results can be seen in Table 2 (ARI results in Appendix B). Classical approaches like X-means often have trouble estimating a reasonable value for k. In most cases, they either stop close to their starting value of 1, i.e. they cannot find a decent cluster structure at all, or continue splitting clusters until they reach their termination condition, i.e. have found more than 35 clusters. In fact, their estimation of k deviates by more than 5 from the ground truth in all but 10 out of 73 experiments (excluding the k=1-estimations on GTSRB). Another drawback is that some experiments were stopped because they exceeded 24 hours, and for many approaches, a single run still took more than an hour. The autoencoder embedding does not lead to a notable improvement for our k-estimation competitors. Since the embedding here needs to be learned oblivious of the respective method, this shows that it is crucial to simultaneously integrate cluster number estimation into the actual DC process. DipDECK gives reliably good estimations for k and, at the same time, impressive clustering results, which rival, if not also surpass, other DC approaches.

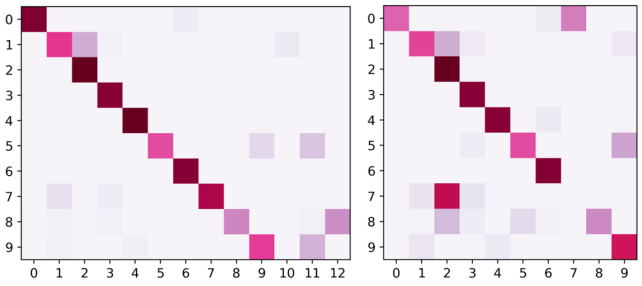
We wish to discuss two cases in more detail to highlight the characteristics of our approach.

4.1.1 Number of clusters. There are two cases where DipDECK's estimations do not seem to be the best ones at first glance. The first is Pendigits. Here, we estimate an average of $k = 14.0$, which does not seem impressive compared to the results of, e.g. Dip-means which estimates $k = 10.4$. However, one has to consider the quality of the found clusters as well. While our k-estimation is worse, our



(a) Subclasses in Optdigits (b) Subclasses in MNIST (c) Subclasses in F-MNIST

Figure 4: Meaningful substructures as found by DipDECK. Ground truth labels are equal for all top and all bottom images respectively in each subfigure (a), (b), (c). (a) Optdigits: [top] The digit '9' can be written with a round or a straight lower part and [bottom] '1' either as a straight line or with an additional skewed serif on top. (b) MNIST: [top] The digit '2' is either written with a loop or as a mirrored 'S'. [bottom] Without augmentation strategies, straight digits and rotated digits are split. (c) [top] For Fashion-MNIST, the class 'bags' can be split into bags with and without straps and the class 'sandal' [bottom] into flat sandals and such with high heels.



(a) DipDECK result (b) Dip-means result

Figure 5: Confusion matrices of DipDECK and Dip-means applied to Pendigits. The rows stand for the ground truth, while columns show the predicted clusters. Darkness of a square represents the number of data points in the cluster.

NMI is by far better with 0.817 vs 0.691. This shows that while we find more clusters, the ones we find better represent the ground truth. An obvious hypothesis follows: our approach partitions the ground truth clusters into multiple parts, while Dip-means splits the data set contrary to the ground truth. We verify this with the confusion matrices. A typical result of DipDECK with an NMI of 0.807 and 13 found clusters can be seen in Fig. 5a. The last two rows of the confusion matrix show that two ground-truth clusters are split into two. Furthermore, column 10 shows us a cluster consisting of less than 100 points, which can be easily dismissed. There are, of course, various data points assigned to the wrong clusters, but a comparison to the confusion matrix of Dip-means (Fig. 5b) shows us how much more helpful the result of DipDECK is.

We assume that DipDECK's overestimation of k on Pendigits is due to the structure found in a cluster, i.e. it splits clusters into meaningful sub-clusters. For example, in MNIST, digits are written in distinct styles. Splitting a cluster along these lines is reasonable, although not corresponding to the ground truth. In Pendigits, we can observe that clusters are split. However, we cannot verify that this split is based on a particular style since the data points are not images that can be visualised in a natural way. It is, however, possible for data sets as the mentioned MNIST. In Sec. 4.2, we take a closer look and see that DipDECK does indeed split along the lines of such meaningful sub-clusters.

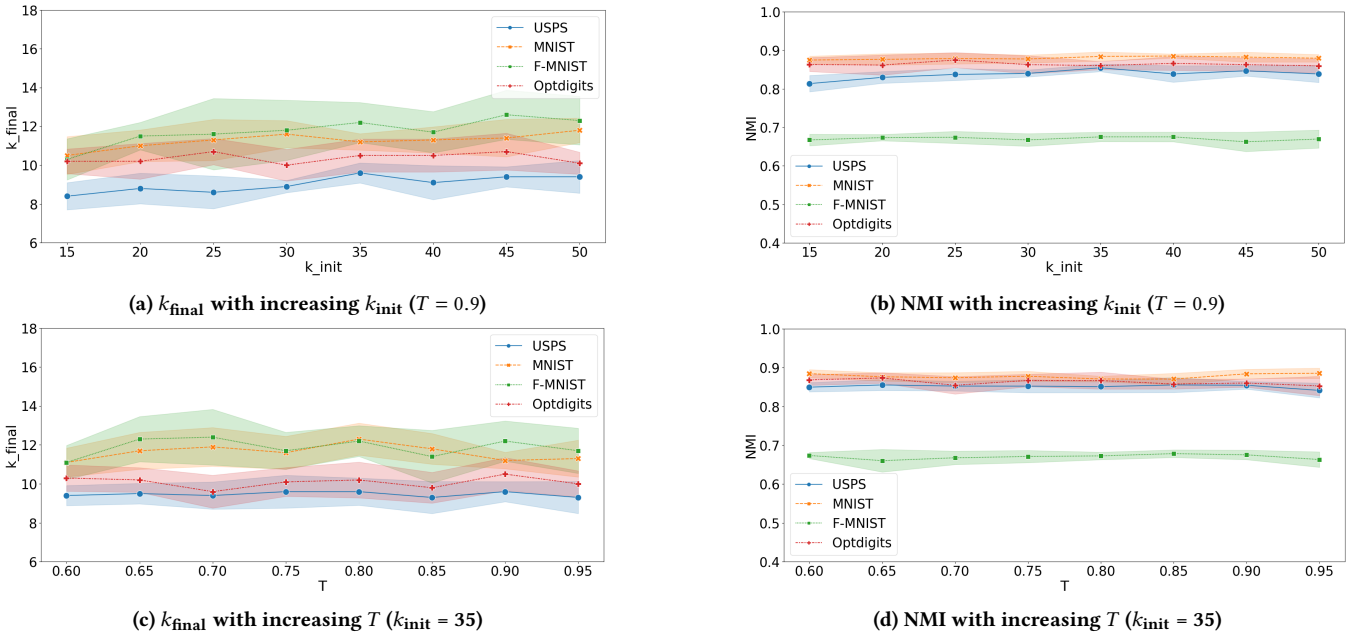
4.1.2 Variance. The second result, where DipDECK does not seem too impressive at first glance, is K-MNIST. The estimated k is relatively large compared to AE+pDip-means, but here again, the NMI (0.658 vs 0.474) shows us that the result of DipDECK consists of far purer clusters. Besides the NMI, there is a second factor to consider here. The variance of the k-estimation of AE+pDip-means is surprisingly large (> 3), with k-values in the range from 5 to 16. Contrary to that, DipDECK's standard deviation on K-MNIST is below 1, as it is on all data sets (except Letterrecognition with 1.80). DipDECK is very stable in this regard, and its results are reliable.

To substantiate the claim that our found clusters are very close to the ground truth, we tested against various DC approaches, in particular those comparable in their architecture. The results can be seen in Table 2. These methods are given the correct number of clusters as a parameter, which is usually unknown in unsupervised settings. DipDECK only needs a rough estimate of the cluster number but still performs better than all of these methods on all but one data set. This supports our claim that more initial micro-clusters are beneficial for the training of the autoencoder. DipDECK is the only tested DC procedure that utilises this idea.

All of this shows us how important it is to evaluate k together with NMI. k alone only gives the number of clusters, while NMI estimates how 'correct' these clusters are. In Table 2, one can see that DipDECK has the best estimate of k in most cases and the best NMI results in all but one case. More importantly, DipDECK shows the best results in the combined evaluation of both - the k-estimation and the clustering task.

4.2 Qualitative Experiments

On some data sets (e.g. Pendigits), there is a slight tendency for DipDECK to overestimate the correct number of clusters (see Table 2). As stated in Sec. 4.1.1, this might be due to meaningful sub-clusters. In the case of data sets like Optdigits, MNIST or Fashion-MNIST, the resulting clusters might reveal substructures within the



(a) k_{final} with increasing k_{init} ($T = 0.9$)

(b) NMI with increasing k_{init} ($T = 0.9$)

(c) k_{final} with increasing T ($k_{\text{init}} = 35$)

(d) NMI with increasing T ($k_{\text{init}} = 35$)

Figure 6: The robustness tests show the flexibility of the main input parameters of DipDECK. All tests are repeated 10 times, and all data sets contain 10 ground truth clusters. Points indicate the mean value, while the coloured area displays the standard deviation.

data set that appear reasonable to human perception. In Fig. 4, the clustering results of DipDECK on these data sets are shown. As can be observed, we indeed find certain meaningful sub-clusters.

Such a run of DipDECK, where the k for Optdigits was estimated as 12, is shown in Fig. 4a. Our algorithm extracted substructures that show two styles for the digits '1' and '9', respectively.

Fig. 4b shows a similar case for MNIST. The top panel shows that two styles for the digit '2' could be detected. The lower panel shows a well-known effect in the MNIST data set. The digit '1' is written as a straight line in both sub-clusters but with a different rotation. The same could sometimes be observed for the digit '5'. However, this can be easily overcome with augmentation strategies, where the network structure learns explicitly to be invariant regarding certain functions such as rotation or shifts.

F-MNIST is the most complex of the three considered data sets. Clustering results with DipDECK- as with other methods - often are mixed clusters containing objects of two to three ground truth classes. Apart from that, also meaningful substructures separating a ground truth cluster can be found. Fig. 4c shows how DipDECK separates 'bags' and 'sandals' into subclasses.

In Appendix D we show a similar analysis for K-MNIST.

4.3 Robustness Experiments

An essential aspect of every method is its stability regarding its (hyper-)parameters. For our method, this entails the DipDECK-specific parameters of the starting value k_{init} and its Dip-p-value threshold T. The correct number of clusters for the data sets in our experiments in Sec. 4.1 range from $k_{\text{true}} = 5$ to $k_{\text{true}} = 26$. For all experiments, we used a starting value of $k_{\text{init}} = 35$, and DipDECK could successfully estimate the correct k, thus making a solid first point for DipDECK's stability regarding k_{init}.

Table 3: DipDECK on different subsets of USPS. The first i digits were extracted from USPS (e.g. 0, 1, 2 and 3 for 4-USPS). The average of 10 runs is given (mean ± std).

Data set	k	NMI	Data set	k	NMI
1-USPS	1.0 ± 0.00	1.000 ± 0.00	6-USPS	5.9 ± 0.67	0.859 ± 0.02
2-USPS	2.0 ± 0.00	0.978 ± 0.00	7-USPS	6.7 ± 0.63	0.858 ± 0.02
3-USPS	3.1 ± 0.29	0.895 ± 0.04	8-USPS	7.4 ± 0.47	0.866 ± 0.02
4-USPS	4.2 ± 0.570	0.862 ± 0.06	9-USPS	8.7 ± 0.44	0.862 ± 0.01
5-USPS	5.0 ± 0.00	0.887 ± 0.02	10-USPS	9.4 ± 0.47	0.851 ± 0.02

4.3.1 ***k_{init}.*** To further investigate the stability regarding k_{init}, we conducted the following experiment. We executed DipDECK on all subsets of the USPS data set consisting of the first i digits, i.e., i-USPS consists of the digits $0, 1, \ldots, i-1$. If k_{init} significantly influences DipDECK, it will have problems estimating the correct number of clusters for data sets with similar characteristics but different correct k. The results can be seen in Table 3. DipDECK finds exactly as many clusters as there are in the data set. For 5-USPS as an example, it finds the correct k in 10 out of 10 runs with an average NMI of 0.89. The only difference in these data sets is the number of clusters. The characteristics of the data sets are the same, as well as the starting value of $k_{\text{init}} = 35$. Independently of k_{init}, DipDECK managed to estimate the correct k, whether it was 1 or 10.

Further stability results can be taken from Fig. 6a. We chose four of the data sets from our main experiments in Table 2 and executed DipDECK with a starting k_{init} ranging from 15 to 50. Despite this

wide range of values, DipDECK shows almost no difference regarding the final k it converges to, with only a very slight increase in k with increased k_{init}. Optdigits, as an example, ranges from an average found k_{final} from 10.0 to 10.7 and USPS from 8.4 to 9.6. In earlier sections, we have already seen how important it is to keep the NMI score in mind when evaluating k-estimates. These can be seen in Fig. 6b and are consistently good. DipDECK finds reliable estimates for k while keeping NMI stable, showing us its capability of finding the correct k independently of the chosen k_{init}.

4.3.2 ***Dip-p-value threshold T.*** The second parameter set by us is the Dip-p-value threshold chosen as 90%. This value is a commonly used threshold regarding significance, but we, nevertheless, also tested DipDECK for different thresholds. The results can be seen in Fig. 6c. Following the experiments, the merging process and the final number of clusters is relatively unaffected by this value. When evaluating various runs of DipDECK in more detail, it becomes evident that the Dip-test is rarely close to a medium value. If two separate clusters are tested, the Dip-p-value is often no higher than 10%, but if a cluster is cut in two and its parts are tested together, the Dip-p-value often surges above 99%. This, again, shows the impressive stability of DipDECK regarding its parameters.

5 CONCLUSION

In this paper, we introduced DipDECK, a novel DC algorithm with an implicit estimation of the number of clusters. To the best of our knowledge, we are the first DC approach with a simultaneous k-estimation. Existing methods in this field heavily rely on the given number of clusters in a data set. In this regard, our algorithm offers considerable relaxation, as one can heavily overestimate this number. Based on the Dip-test connecting structures are identified, and the corresponding clusters are merged.

Furthermore, DipDECK is more flexible regarding the shape of the clusters, while most of the previously proposed DC methods are k-means based and inherit the Gaussian cluster assumption. This is also true for many classical k-estimation methods such as X-means or Dip-means. Additionally, they are limited in their scalability and are, therefore, no longer adequate for modern large and high-dimensional data sets.

In extensive experiments, we demonstrate that our cluster performance and our k-estimation results most often outperform relevant k-estimation and DC baselines on various data sets. We further show that our results are highly stable across multiple runs and robust regarding our two parameters.

DipDECK offers several possibilities to extend this work. The bottom-up like cluster number estimation provides a reasonable basis for a hierarchical clustering algorithm, especially since we could observe that our substructures in case of overestimation are often meaningful for human perception and pure in terms of NMI.

ACKNOWLEDGMENTS

We owe our gratitude to Dominik Mautz and Lukas Miklautz for their support during the creation of this work.

This work has been partially funded by the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A. The authors of this work take full responsibility for its content.

REFERENCES

[1] Horst Bischof, Aleš Leonardis, and Alexander Selb. 1999. MDL principle for robust vector quantisation. *Pattern Analysis & Applications* 2, 1 (1999), 59–72.
[2] Christian Böhm, Christos Faloutsos, Jia-Yu Pan, and Claudia Plant. 2006. Robust information-theoretic clustering. In *SIGKDD*. 65–75.
[3] Theofilos Chamalis and Aristidis Likas. 2018. The Projected Dip-Means Clustering Algorithm. In *Proceedings of the 10th Hellenic Conference on Artificial Intelligence*.
[4] Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. 2018. Deep learning for classical Japanese literature. *arXiv preprint arXiv:1812.01718* (2018).
[5] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml
[6] L. Duan, C. Aggarwal, S. Ma, and S. Sathe. 2019. Improving Spectral Clustering with Deep Embedding and Cluster Estimation. In *ICDM*. 170–179.
[7] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise.. In *SIGKDD*, Vol. 96. 226–231.
[8] Yu Feng and Greg Hamerly. 2007. PG-means: learning the number of clusters in data. In *Advances in neural information processing systems*. 393–400.
[9] Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. 2017. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *Proceedings of the IEEE international conference on computer vision*. 5736–5745.
[10] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. 2017. Improved Deep Embedded Clustering with Local Structure Preservation. In *IJCAI*.
[11] Greg Hamerly and Charles Elkan. 2004. Learning the k in k-means. In *Advances in neural information processing systems*. 281–288.
[12] J. A. Hartigan and P. M. Hartigan. 1985. The Dip Test of Unimodality. *Ann. Statist.* 13, 1 (03 1985), 70–84. https://doi.org/10.1214/aos/1176346577
[13] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. 2013. Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *IJCNN*. IEEE, 1–8.
[14] Jonathan J. Hull. 1994. A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence* 16, 5 (1994), 550–554.
[15] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. 2017. Variational Deep Embedding: An Unsupervised and Generative Approach to Clustering. In *IJCAI*. 1965–1972. https://doi.org/10.24963/ijcai.2017/273
[16] Argyris Kalogeratos and Aristidis Likas. 2012. Dip-means: an incremental clustering method for estimating the number of clusters. In *Advances in Neural Information Processing Systems*.
[17] Yann Lecun. 1987. *PhD thesis: Modeles connexionnistes de l'apprentissage (connectionist learning models)*. Universite P. et M. Curie (Paris 6).
[18] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.
[19] Samuel Maurus and Claudia Plant. 2016. Skinny-dip: clustering in a sea of noise. In *SIGKDD*. 1055–1064.
[20] Dominik Mautz, Claudia Plant, and Christian Böhm. 2019. Deep embedded cluster tree. In *ICDM*. IEEE, 1258–1263.
[21] Tom Monnier, Thibault Groueix, and Mathieu Aubry. 2020. Deep Transformation-Invariant Clustering. In *NeurIPS*.
[22] Sudipto Mukherjee, Himanshu Asnani, Eugene Lin, and Sreeram Kannan. 2019. ClusterGAN: Latent Space Clustering in Generative Adversarial Networks. In *AAAI*. 4610–4617.
[23] Dan Pelleg and Andrew W. Moore. 2000. X-Means: Extending K-Means with Efficient Estimation of the Number of Clusters. In *ICML (ICML '00)*.
[24] B. Schelling, L. Bauer, S. Behzadi Soheil, and C. Plant. 2020. Utilizing Structure-rich Features to improve Clustering. In *ECML-PKDD 2020*.
[25] B. Schelling and C. Plant. 2018. DipTransformation: Enhancing the Structure of a Dataset and Thereby Improving Clustering. In *ICDM*.
[26] Nguyen Xuan Vinh, Julien Epps, and James Bailey. 2010. Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance. *Journal of Machine Learning Research* (2010).
[27] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. arXiv:cs.LG/1708.07747 [cs.LG]
[28] Junyuan Xie, Ross B. Girshick, and Ali Farhadi. 2016. Unsupervised Deep Embedding for Clustering Analysis. In *ICML (JMLR Workshop and Conference Proceedings)*.
[29] Bo Yang, Xiao Fu, Nicholas D. Sidiropoulos, and Mingyi Hong. 2017. Towards K-means-friendly Spaces: Simultaneous Deep Learning and Clustering. In *ICML*.
[30] Jianwei Yang, Devi Parikh, and Dhruv Batra. 2016. Joint Unsupervised Learning of Deep Representations and Image Clusters. In *CVPR*.
[31] Lihi Zelnik-Manor and Pietro Perona. 2005. Self-Tuning Spectral Clustering. In *Advances in Neural Information Processing Systems*.

APPENDIX

In the appendix, we present details about the data sets used in the experiments (Appendix A), add ARI results for our quantitative experiments (Appendix B) and give a detailed explanation of how we deal with imbalanced micro-cluster sizes (Appendix C). We also take a closer look at the sub-structures found in K-MNIST (Appendix D).

A DATA SETS

A summary of our used data sets is given in Table 4.

Table 4: Information regarding the used data sets.

Data set	# Samples	# Features	# Classes
USPS	9298	256	10
MNIST	70000	784	10
F-MNIST	70000	784	10
K-MNIST	70000	784	10
Optdigits	5620	64	10
Pendigits	10992	16	10
Letterrecognition	20000	16	26
GTSRB	7860	3072	5

USPS [14]: Greyscale image data set consisting of 9298 handwritten digits (0 to 9) with a size of 16×16 pixels.

MNIST [18]: Greyscale image data set consisting of 70000 handwritten digits (0 to 9) with a size of 28×28 pixels.

Fashion-MNIST (F-MNIST) [27]: Greyscale image data set consisting of 70000 articles from the *Zalando* online store. Each sample belongs to one of 10 product groups and has a size of 28×28 pixels.

Kuzushiji-MNIST (K-MNIST) [4]: Greyscale image data set consisting of 70000 Kanji characters (10 different characters, each representing one column of hiragana) with a size of 28×28 pixels.

Optdigits [5]: This 8×8 image data set consists of 5620 samples, each representing a digit (0 to 9). Each pixel depicts the number of marked pixels within a 4×4 block of the original 32×32 bitmaps.

Pendigits [5]: This data set consists of 10992 vectors of length 16, representing 8 coordinates. The coordinates were taken from the task of writing digits (0 to 9) on a tablet.

Letterrecognition [5]: This data set consists of 20000 data samples whereby each sample represents one of the 26 capital letters in the English alphabet. All samples are composed of 16 numerical stimuli describing the respective letter.

GTSRB [13]: Image data set containing 32×32 images of German traffic signs. Each image contains 3 colour channels. We used a subset of 5 different signs ('Speed limit (50)', 'No passing', 'Ahead only', 'Right of way', 'Attention'), resulting in 7860 images.

B ARI RESULTS

In addition to the quantitative evaluation via NMI, we calculated the corresponding ARI values. ARI also ranges from 0 to 1, where 1 marks a perfect result. The results can be seen in Table 5.

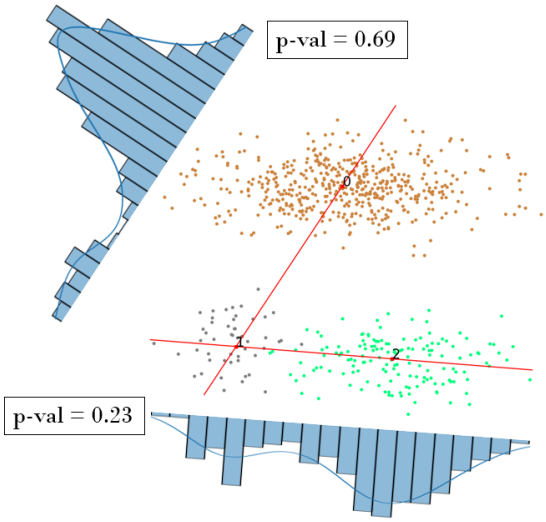


Figure 7: Synthetic data set consisting of three Gaussian clusters containing 500 (brown), 50 (grey) and 150 (green) objects. The left histogram shows the distribution of clusters 0 (brown) and 1 (grey) projected to the connection line (red) between their respective centres. The lower histogram shows the analogous distribution of clusters 1 (grey) and 2 (green). The Dip-p-values of those distributions are 0.69 **for clusters 0 and 1 and** 0.23 **for clusters 1 and 2.**

C IMBALANCED CLUSTERS

We mentioned in Sec. 3 that there are problems applying the Dip-test if two clusters are very different in size. To analyse this, we created a two-dimensional data set with three Gaussian clusters of different sizes as an example. The setting is displayed in Fig. 7. Cluster 0 (brown) contains 500, cluster 1 (grey) 50 and cluster 2 (green) 150 points. It can be seen that cluster 1 lies relatively close to cluster 2. It is conceivable that these two clusters belong together. Cluster 0 and cluster 1, on the other hand, should not be merged, as a clear separation can be observed. However, the Dip-p-values indicate the opposite. With a value of 0.69, cluster 1 is said to have a higher affiliation with cluster 0 than with cluster 2 (Dip-p-value of 0.23). This means that in the case of DipDECK, contrary to the natural perception, the autoencoder is more likely to move cluster 1 closer to cluster 0 than to cluster 2. This is because cluster 1 is so small relative to cluster 0 that it is perceived merely as outliers and not as a second mode. In the histogram of cluster 1 and cluster 2, on the other hand, two distinct modes can be seen, which prevents a higher value. When the autoencoder starts moving the clusters towards each other, these effects are amplified.

To counteract this behaviour, in addition to the Dip-value of the fully combined clusters, we calculate the Dip-value of the transition from one cluster to another if the sizes differ significantly. For this, we define a threshold S. If $|C_j| > S|C_i|$, we calculate a second Dip-value by only taking the $S|C_i|$ points closest to μ_i from C_j in combination with all points from C_i. This gives us two different Dip-values, from which we choose the maximum.

Table 5: The resulting ARI values of the experiments as described in the Sec. 4

Method	USPS	MNIST	F-MNIST	K-MNIST	Optdigits	Pendigits	Letterrec.	GTSRB
DipDECK (ours)	**0.834 ± 0.05**	**0.860 ± 0.02**	**0.494 ± 0.02**	**0.519 ± 0.02**	0.833 ± 0.02	**0.740 ± 0.01**	**0.221 ± 0.03**	0.518 ± 0.09
X-means	0.315 ± 0.01	0.271 ± 0.01	0.230 ± 0.01	0.263 ± 0.01	0.415 ± 0.01	0.430 ± 0.02	0.155 ± 0.01	0.150 ± 0.01
G-means	0.313 ± 0.00	0.270 ± 0.01	0.232 ± 0.01	0.259 ± 0.01	0.432 ± 0.02	0.423 ± 0.02	0.154 ± 0.01	0.148 ± 0.01
PG-means	0.041 ± 0.05	0.008 ± 0.04	0.139 ± 0.06	0.067 ± 0.02	0.007 ± 0.02	0.189 ± 0.11	0.011 ± 0.01	†
Dip-means	0.292 ± 0.00	†	†	†	0.000 ± 0.00	0.537 ± 0.02	0.000 ± 0.00	0.000 ± 0.00
pDip-means	0.333 ± 0.02	0.261 ± 0.00	0.227 ± 0.00	0.248 ± 0.01	0.391 ± 0.01	0.409 ± 0.01	0.161 ± 0.01	0.148 ± 0.00
AE+X-means	0.143 ± 0.00	0.409 ± 0.09	0.362 ± 0.03	0.030 ± 0.03	0.752 ± 0.01	0.517 ± 0.02	0.023 ± 0.01	0.239 ± 0.01
AE+G-means	0.436 ± 0.02	0.430 ± 0.02	0.300 ± 0.02	0.327 ± 0.01	0.460 ± 0.02	502 ± 0.01	0.217 ± 0.01	0.178 ± 0.01
AE+PG-means	0.157 ± 0.09	0.142 ± 0.27	0.206 ± 0.05	0.055 ± 0.04	0.09 ± 0.06	0.347 ± 0.11	0.003 ± 0.08	0.022 ± 0.04
AE+Dip-means	0.537 ± 0.04	†	†	†	0.000 ± 0.00	0.564 ± 0.01	0.000 ± 0.00	0.000 ± 0.00
AE+pDip-means	0.395 ± 0.10	595 ± 0.07	0.304 ± 0.10	0.348 ± 0.08	0.000 ± 0.00	0.581 ± 0.04	0.000 ± 0.00	0.000 ± 0.00
DEC	0.716 ± 0.02	0.755 ± 0.04	0.441 ± 0.04	0.411 ± 0.01	**0.850 ± 0.05**	0.621 ± 0.03	0.152 ± 0.02	0.489 ± 0.02
IDEC	0.733 ± 0.03	0.807 ± 0.02	0.475 ± 0.03	0.410 ± 0.03	0.811 ± 0.04	0.634 ± 0.02	0.197 ± 0.03	**0.520 ± 0.07**
DCN	0.642 ± 0.03	0.790 ± 0.03	0.449 ± 0.03	0.360 ± 0.05	0.829 ± 0.04	0.587 ± 0.04	0.209 ± 0.02	0.437 ± 0.05
VaDE	0.605 ± 0.08	0.716 ± 0.05	0.480 ± 0.02	0.401 ± 0.02	0.621 ± 0.06	0.586 ± 0.05	0.023 ± 0.01	0.102 ± 0.02

Table 6: Impact of different choices of S on the Dip-p-values from the example displayed in Fig. 7.

S	1	2	3	4	5	∞
$p_{0,1}$	0.0012	0.0082	0.0240	0.0544	0.1039	0.6892
$p_{1,2}$	0.0196	0.0879	0.2307	0.2307	0.2307	0.2307

In total, there are three possible cases to determine the Dip-value:

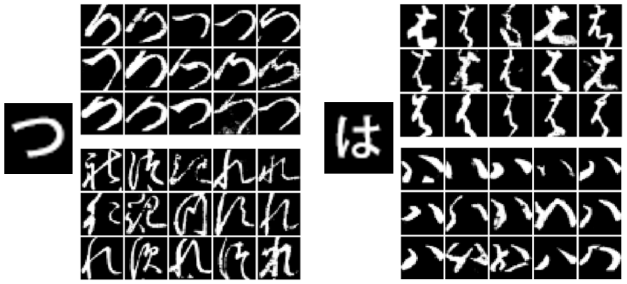
$$d_{i,j} = \begin{cases} \max\{\mathrm{Dip}(C^{1d}_{i,j}), \mathrm{Dip}(C^{1d}_{i,j_S})\}, & \text{if } |C_j| > S|C_i| \\ \max\{\mathrm{Dip}(C^{1d}_{i,j}), \mathrm{Dip}(C^{1d}_{i_S,j})\}, & \text{if } |C_i| > S|C_j| \\ \mathrm{Dip}(C^{1d}_{i,j}), & \text{otherwise} \end{cases}$$

Here $C^{1d}_{i_S,j}$ only contains the $S|C_j|$ nearest points of C_i to μ_j in the embedded space, and C^{1d}_{i,j_S} is defined analogously.

To see the impact of the choice of S, Table 6 can be inspected, which shows the effects using the example data set described above. 2 seems to be a good compromise to solve the described problem while not preventing interesting cluster connections, like the one between clusters 1 and 2. Note, however, that all of the S values (except ∞) lead to Dip-p-values that correspond to our perception of the relative connections between the clusters 0, 1 and 2.

D K-MNIST QUALITATIVE EXPERIMENTS

K-MNIST seems to be one of the data sets where DipDECK does not perform too well, as the average estimated number of clusters is with 15.8 significantly larger than the desired amount of 10. However, as with MNIST, F-MNIST, and Optdigits, we can again show that DipDECK finds meaningful subdivisions within the ground truth clusters. For example, consider the letter 'tsu' (displayed in Fig. 8a). We can see that the first identified cluster in the top well represents the modern counterpart. The lower cluster on the other hand looks very different. In this case it seems reasonable to perform a separation. The same applies to the character 'ha' (displayed in Fig. 8b). Here, the objects from the upper cluster show certain similarities to the second sign of the modern counterpart. The objects of the lower cluster, on the contrary, often consist of two single dashes. The examples show once again that splitting ground truth clusters can lead to additional insights into a data set in some cases.



(a) The character 'tsu' **(b) The character 'ha'**

Figure 8: Found substructures of DipDECK within the characters 'tsu' and 'ha' in the K-MNIST data set. The single characters on the left show the modern hiragana counterparts. On the right we see two different styles of writing found by DipDECK.