Reducing Objective Function Mismatch in Deep Clustering with the Unsupervised Companion Objective

Daniel J. Trosten* [1], Robert Jenssen[1], and Michael C. Kampffmeyer[1]

[1]UiT Machine Learning Group, https://machine-learning.uit.no

Abstract

Preservation of local similarity structure is a key challenge in deep clustering. Many recent deep clustering methods therefore use autoencoders to help guide the model's neural network towards an embedding which is more reflective of the input space geometry. However, recent work has shown that autoencoder-based deep clustering models can suffer from objective function mismatch (OFM). In order to improve the preservation of local similarity structure, while simultaneously having a low OFM, we develop a new auxiliary objective function for deep clustering. Our Unsupervised Companion Objective (UCO) encourages a consistent clustering structure at intermediate layers in the network – helping the network learn an embedding which is more reflective of the similarity structure in the input space. Since a clustering-based auxiliary objective has the same goal as the main clustering objective, it is less prone to introduce objective function mismatch between itself and the main objective. Our experiments show that attaching the UCO to a deep clustering model improves the performance of the model, and exhibits a lower OFM, compared to an analogous autoencoder-based model.

1 Introduction

Deep clustering is a subfield of deep learning [7] which considers the design of unsupervised loss functions, in order to train deep learning models for clustering. The loss functions developed in this field have made it possible to train deep architectures to discover the underlying group structure in large datasets, containing data types with complex geometrical structure, such as images [22, 2, 5] and time series [18]. The ever growing amount of unlabeled data has caused unsupervised learning to be identified as a main next goal in machine learning research [7].

Many of the recent deep clustering models include deep neural networks that have been pre-trained as autoencoders [20, 1, 21, 10]. In these models, the unsupervised clustering loss is attached to the code space of the autoencoder, and the model is fine tuned using either the clustering loss alone, or both the clustering loss, and the reconstruction loss from the autoencoder.

Deep Embedded Clustering (DEC) [20] is a cornerstone method in deep clustering. In DEC, the network is pre-trained as a denoising autoencoder. After pre-training, DEC fine-tunes a set of cluster centroids in order to compute the final cluster assignments. However, Guo *et al.* [1] argue that discarding DEC's decoder in the fine-tuning stage hinders the preservation of the local similarity structure between samples. They therefore propose Improved DEC (IDEC), which aims to alleviate this issue, by retaining the decoder and reconstruction loss during fine-tuning.

Despite the popularity of the autoencoder approach, we hypothesize that the representation produced by the autoencoder does not necessarily emphasize properties which are desirable for clustering. These models can therefore suffer from *objective function mismatch* (OFM) [12]. OFM occurs when the optimization of an auxiliary objective (*e.g.* reconstruction) has a negative impact on the optimization of the main objective (*e.g.* clustering). In fact, Mrabah *et al.* [13, 14] show that the aforementioned IDEC suffers from OFM during training – supporting our hypothesis on OFM occurring in

*Corresponding author: daniel.j.trosten@uit.no

This work was partially funded by the Research Council of Norway grant no. 302022 and no. 309439.

https://doi.org/10.7557/18.5709

autoencoder-based deep clustering models.

Deep Divergence-based Clustering (DDC) [5] is another deep clustering approach, which in contrast to DEC and IDEC, does not rely on an auxiliary autoencoder to train its neural network. In fact, DDC can be trained end-to-end from randomly initialized parameters using only its clustering loss – outperforming DEC on several deep clustering tasks [5, 18]. Despite the promising performance of DDC however, we do not know whether it is prone to suffer from the same issues regarding similarity preservation, as was observed in DEC.

In this paper, we present an *Unsupervised Companion Objective* (UCO), whose task is to preserve the similarity structure between samples in deep clustering models. Inspired by Deeply-supervised Nets [9], the UCO consists of a set of auxiliary clustering objectives attached to intermediate layers in the neural network, which encourage a common cluster structure at the output of these layers. Since the UCO is based on clustering, we expect a low OFM between the UCO and the main clustering objective. Our experiments show that the UCO both exhibits reduced OFM when compared to a reconstruction objective, and improves the overall clustering performance of a deep clustering model.

2 Method

Throughout this paper, we will assume that the deep clustering model follows this general design:

$$\boldsymbol{z} = f_{\boldsymbol{\theta}}(\boldsymbol{x}), \quad \boldsymbol{\alpha} = g_{\boldsymbol{\phi}}(\boldsymbol{z}) \tag{1}$$

where $f_{\boldsymbol{\theta}}$ denotes the neural network producing a learned representation $\boldsymbol{z}$, from the input $\boldsymbol{x}$, and $g_{\boldsymbol{\phi}}$ denotes the clustering module producing the cluster membership vector $\boldsymbol{\alpha}$. $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ represent the parameters of the neural network, and the parameters of the clustering module, respectively. See Figure 1 for an overview of the assumed clustering model.

For generality, we define *blocks* to be generic computational units in a deep neural network. Layers are perhaps the most familiar examples of blocks, but a block can also represent, for instance, a collection of adjacent layers, or individual components within a specific layer. Since $f_{\boldsymbol{\theta}}$ represents a neural network, it can be decomposed block-wise as:

$$f_{\boldsymbol{\theta}} = f^{L}_{\boldsymbol{\theta}_L} \circ f^{L-1}_{\boldsymbol{\theta}_{L-1}} \circ \cdots \circ f^{1}_{\boldsymbol{\theta}_1} \tag{2}$$

where $f^l_{\boldsymbol{\theta}_l}$ is the mapping performed by block l, and L is the number of blocks in f.

If we let $\boldsymbol{y}^l$ be the output of block l, we have:

$$\boldsymbol{y}^l = f^l_{\boldsymbol{\theta}_l}(\boldsymbol{y}^{l-1}) \tag{3}$$

with $\boldsymbol{y}^0 = \boldsymbol{x}$ and $\boldsymbol{y}^L = \boldsymbol{z}$.

Finally, we assume that $f_{\boldsymbol{\theta}}$ and $g_{\boldsymbol{\phi}}$ are optimized jointly with a clustering loss, $\mathcal{L}_{\text{cluster}}$, which is computed by a clustering module. The focus of this work is on the proposed UCO, and the neural network f, to which it is attached. We therefore use the clustering module and loss from DDC [5], and treat this as fixed.

2.1 Unsupervised companion objective

The unsupervised companion objective, $\mathcal{L}_{\text{UCO}}$, consists of the terms $\mathcal{L}^1_{\text{UCO}}, \ldots, \mathcal{L}^L_{\text{UCO}}$, which are added to the final clustering loss, in order to help preserve the similarity structure between samples. All of these terms are designed to encourage the same cluster structure at their respective blocks, as the one found by the clustering module. Following [5], we do this by maximizing the Cauchy-Schwarz divergence [4] between clusters in the space of intermediate representations. For block l, we have

$$\mathcal{L}^l_{\text{UCO}} = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \ell_{ij} \tag{4}$$

with

$$\ell_{ij} = \frac{\binom{k}{2}^{-1} \sum_{a=1}^{n} \sum_{b=1}^{n} \alpha_{ai} k^l_{ab} \alpha_{bj}}{\sqrt{\sum_{a=1}^{n} \sum_{b=1}^{n} \alpha_{ai} k^l_{ab} \alpha_{bi} \sum_{a=1}^{n} \sum_{b=1}^{n} \alpha_{aj} k^l_{ab} \alpha_{bj}}}. \tag{5}$$

Here, k denotes the number of clusters, α_{ai} is the soft assignment of sample a to cluster i. k^l_{ab} denotes a Gaussian kernel evaluated at $(\boldsymbol{y}^l_a, \boldsymbol{y}^l_b)$:

$$k^l_{ab} = \exp\left(-\frac{||\operatorname{flat}(\boldsymbol{y}^l_a) - \operatorname{flat}(\boldsymbol{y}^l_b)||^2}{2\sigma^2}\right) \tag{6}$$

where σ is a hyperparameter. Since the intermediate representations can be arrays with more than one dimension (*e.g.* 3 for convolutional layers), we apply the $\operatorname{flat}(\cdot)$ function to reshape the arrays into vectors before evaluating the Gaussian kernel.

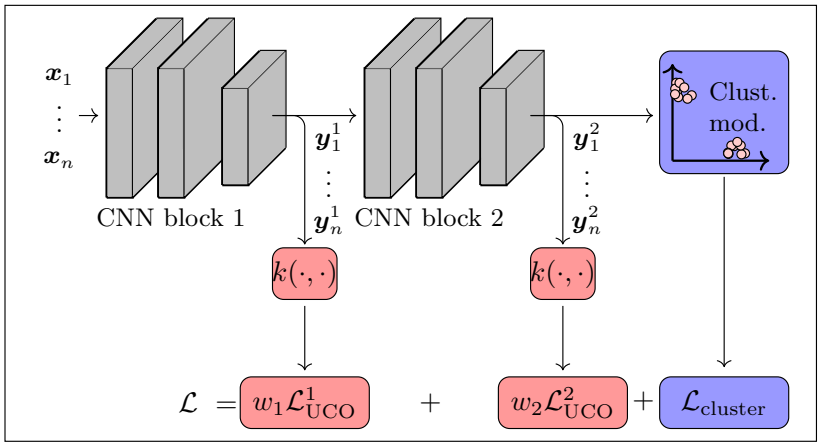

Figure 1: Overview of a deep clustering model augmented with the UCO. Gray, blue, and red elements indicate components of the neural network, clustering module (Clust. mod.), and our proposed UCO, respectively.

The UCO can now be obtained as a weighted sum of the terms for each block:

$$\mathcal{L}_{\text{UCO}} = \sum_{l=1}^{L} w_l \mathcal{L}^l_{\text{UCO}} \tag{7}$$

where w_l is the weight for the term attached to block l. In order to avoid that the number of hyperparameters (weights) scales linearly with the depth of the network, we adopt the alternative weighting strategy:

$$w_l = \lambda \cdot \omega(l) \tag{8}$$

where λ is a base weight, and $\omega : \{1, \ldots, L\} \to [0, 1]$ is a function computing the relative weight for block l.

3 Experiments

3.1 Setup

Models. In order to evaluate the performance of the proposed unsupervised companion objective, we train the DDC model with and without the UCO, using the same network architecture and hyperparameter setup for both models. We refer to these configurations as DDC-UCO and DDC, respectively. Drawing inspiration from IDEC [1], we also train an autoencoder-based DDC model, which includes a decoder network whose task is to reconstruct the input data. We refer to this model as DDC-AE.

Block	Layer type	Filter size	Filters	Batch-norm [3]
1	`Conv`	5×5	32	✗
	`Conv`	5×5	32	✓
	`MaxPool`	2×2	–	✗
2	`Conv`	3×3	32	✗
	`Conv`	3×3	32	✓
	`MaxPool`	2×2	–	✗

Table 1: Layers in the CNN used to train DDC, DDC-AE, and DDC-UCO. All `Conv` layers use ReLU activation functions. Layers using batch-norm apply the normalization before the activation function.

Implementation. As in [5], we use a small convolutional neural network for our experiments. Our CNN consists of two sequential blocks, both having two convolutional layers, followed by max pooling operations. An overview of the layers in the network is shown in Table 1. In DDC-AE, we create the decoder network by mirroring the CNN, and replace convolutions with transpose convolutions, and max-pooling operations with nearest-neighbor upsampling.

The models are implemented in the PyTorch framework [16]. All models are trained on stochastic

Name	Image size	Color	n	k
MNIST [8]	28×28	Gray	60000	10
USPS	16×16	Gray	9298	10
F-MNIST [19]	28×28	Gray	60000	10
COIL-100 [15]	128×128	RGB	7200	100

Table 2: Overview of the datasets used for evaluation. n and k denote the total number of images, and the number of categories, respectively.

mini-batches of size 120, using the Adam optimizer [6], with a learning rate of 0.0001. Following [5], we train each model 20 times, and report the accuracy (ACC) and normalized mutual information (NMI) of the model resulting in the lowest value of the total loss function. For each block, the σ hyperparameter is set to 15% of the median pairwise distance between the intermediate outputs for samples in the minibatch [5].

Since hyperparameter tuning can be difficult when labeled data is unavailable, we set $\lambda = 0.1$ and $\omega(l) = 10^{l-L}$ for all datasets. This choice is further investigated below.

Datasets. We test the models on the MNIST, USPS, Fashion-MNIST (F-MNIST), and COIL-100 datasets. These datasets represent clustering tasks which are often encountered in computer vision, and are thus widely used in the deep clustering literature [20, 1, 17, 5, 22]. An overview of the datasets can be found in Table 2.

3.2 Results

Quantitative results. Table 3 shows the clustering results for DDC, DDC-AE, and DDC-UCO on the baseline datasets. These results indicate that adding the UCO to DDC tends to improve the clustering performance of the model. Adding a decoder and a reconstruction loss however, leads to a drop in performance on all datasets. Finally, we observe that DDC-UCO outperforms DEC and IDEC on MNIST and USPS.

Objective function mismatch. In order to demonstrate that the UCO leads to reduced OFM when compared to an autoencoder-based method, we use the following metric to measure the OFM

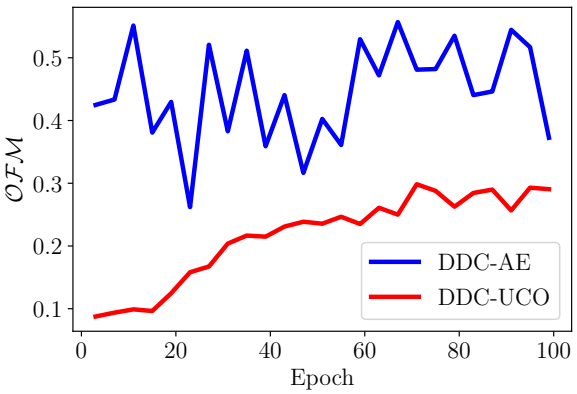

Figure 2: OFM in DDC-AE and DDC-UCO during training.

during training:

$$\mathcal{OFM} = \frac{1 - \Delta_{FD}}{2} \tag{9}$$

where Δ_{FD} is the *feature drift* [13]:

$$\Delta_{FD} = \cos\left(\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{cluster}}, \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{pretext}}\right). \tag{10}$$

Δ_{FD} measures the "agreement" between the gradient of the clustering objective, $\mathcal{L}_{\text{cluster}}$, and a *pretext task*, $\mathcal{L}_{\text{pretext}}$, with respect to the weights in the network, $\boldsymbol{\theta}$. For DDC-AE, the pretext task is reconstruction, and for DDC-UCO, it is the set of auxiliary clustering objectives introduced by the UCO. $\mathcal{OFM}$ is a re-parametrization of Δ_{FD}, which is scaled to be in the range $[0, 1]$, and negated, such that higher values indicate a larger degree of mismatch between the objectives.

Figure 2 shows the observed OFM when training DDC-AE and DDC-UCO. The models were trained on 2000 randomly selected images from the first four digits of the MNIST dataset. The plots indicate that the OFM is significantly lower in DDC-UCO, compared to DDC-AE, at all epochs. This both confirms our hypothesis about the presence of OFM in autoencoder-based deep clustering models, and shows that replacing the reconstruction loss with the UCO reduces the OFM.

Interestingly, we observe that the OFM in DDC-UCO tends to increase during training. We hypothesize that this is because, in the early stages of training, the clustering loss and UCO work together to strengthen the cluster structure at intermediate

	MNIST		USPS		F-MNIST		COIL-100	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
DEC	86.6	83.7	74.1	75.3	–	–	–	–
IDEC	88.1	86.7	76.1	78.5	–	–	–	–
DDC	87.3	84.7	**77.0**	78.5	**67.1**	**60.9**	62.3	82.9
DDC-AE	84.7	83.7	74.2	77.3	62.1	58.9	55.9	78.8
DDC-UCO	**88.6**	**87.0**	76.9	**79.9**	61.2	57.3	**62.6**	**83.7**

Table 3: Clustering results (ACC [%] and NMI [%]) for MNIST, USPS, F-MNIST, and COIL-100. The results for DEC and IDEC were obtained from [1]. Best results are highlighted in bold.

outputs throughout the network, and determine the appropriate cluster memberships. Later in the training, the clustering loss encourages the clusters to be as compact and separable as possible, which can be achieved by mapping one or more clusters to a single point. The UCO counteracts this behavior by enforcing the cluster structure to be more similar to the cluster structure found at earlier blocks in the network, and thus is more reflective of the cluster structure found in the input data. This, in turn, leads to a mismatch between the clustering objective and the UCO in later stages of training. This hypothesis will be further investigated in future work.

UCO weighting scheme. The base weight λ and the relative weighting function ω are important hyperparameters in the UCO. To investigate the impact of these hyperparameters, we vary λ between 0.01, 0.1 and 1. Then, for each of these values, we train DDC-UCO with the following three ω-functions: (i) Constant: $\omega(l) = 1$; (ii) Linear: $\omega(l) = l/L$; and (iii) Exp: $\omega(l) = 10^{l-L}$.

The results for the different configurations are shown in Table 4. These results show that the choice of λ and ω does influence the performance of the model. Setting $\lambda = 1$ for instance, leads to slightly worse overall performance, compared to $\lambda = 0.01$ and $\lambda = 0.1$. This shows that equally weighing $\mathcal{L}_{\text{cluster}}$ and $\mathcal{L}_{\text{UCO}}$, results in too much emphasis being put on $\mathcal{L}_{\text{UCO}}$, which in turn leads to a slightly worse final clustering. Despite these variations in performance, none of the configurations cause the model to fail completely. This is an important property, due to the difficulties of validating hyperparameters in a fully unsupervised setting. Lastly, we note that there is not one best configuration for all datasets. We could therefore

MNIST

ω →	Constant		Linear		Exp.	
λ ↓	ACC	NMI	ACC	NMI	ACC	NMI
0.01	**95.7**	**91.0**	81.2	78.6	87.3	85.9
0.10	80.1	80.8	78.9	79.6	<u>88.6</u>	<u>87.0</u>
1.00	70.9	71.1	87.1	84.3	77.0	74.6

USPS

ω →	Constant		Linear		Exp.	
λ ↓	ACC	NMI	ACC	NMI	ACC	NMI
0.01	74.6	77.6	74.3	76.0	**84.8**	**78.5**
0.10	73.8	77.7	69.4	72.8	<u>76.9</u>	<u>79.9</u>
1.00	76.7	80.3	75.9	79.1	75.6	79.4

Fashion-MNIST

ω →	Constant		Linear		Exp.	
λ ↓	ACC	NMI	ACC	NMI	ACC	NMI
0.01	54.7	51.2	54.9	52.4	39.6	43.6
0.10	**66.1**	**60.0**	64.6	59.5	<u>61.2</u>	<u>57.3</u>
1.00	63.1	57.6	57.7	54.3	61.2	56.1

COIL-100

ω →	Constant		Linear		Exp.	
λ ↓	ACC	NMI	ACC	NMI	ACC	NMI
0.01	61.5	82.7	61.3	84.1	62.4	82.6
0.10	58.3	80.6	58.3	81.5	<u>**62.6**</u>	<u>**83.7**</u>
1.00	54.8	76.8	56.2	75.7	54.8	77.4

Table 4: Clustering results for DDC-UCO with different base weights (λ), and different weighting functions (ω). Results from the best configurations are highlighted in bold. Results from our selected configuration are underlined.

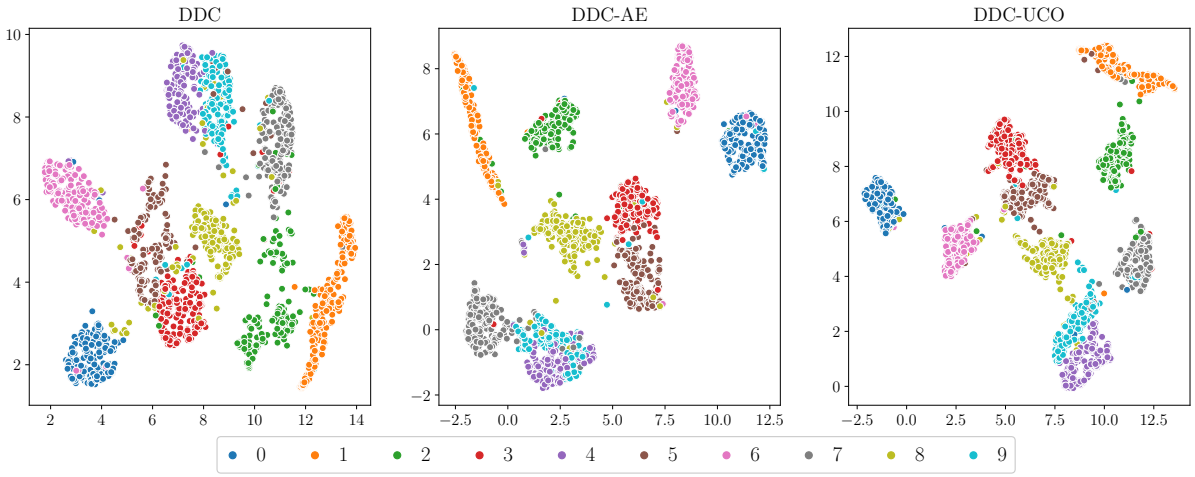

Figure 3: Intermediate outputs from the network's first block in DDC, DDC-AE, and DDC-UCO, trained on the MNIST dataset. The representations were projected to two dimensions using UMAP [11].

improve the results of DDC-UCO in Table 3 by tuning the hyperparameters individually for each dataset. However, we refrain from dataset-specific hyperparameter tuning as it would not be a viable option in a general unsupervised setting.

Separability of intermediate representations. Figure 3 shows the intermediate outputs from the first block in the neural network, for DDC, DDC-AE, and DDC-UCO, trained on MNIST. The plots show that both DDC-AE and DDC-UCO produce more separable clusters, compared to DDC. This illustrates that DDC does indeed benefit from the improved similarity preservation introduced by the respective auxiliary objectives. When comparing the representations from DDC-AE and DDC-UCO, we observe that although both models are able to separate most of the digits from each other, the clusters corresponding to 4, 7, and 9, seem to be somewhat more separable in the representation produced by DDC-UCO.

4 Conclusion

We've presented the unsupervised companion objective (UCO). It is a new auxiliary objective for deep clustering, which consists of several clustering losses attached to different blocks in the model's neural network. By encouraging a consistent cluster structure throughout the network, we make the network embedding more preservative of the local similarity structure in the input space. Furthermore, the clustering-based nature of the UCO makes the resulting model less prone to suffer from objective function mismatch, compared to deep clustering models based on autoencoders.

Our experiments show that attaching the UCO to DDC results in an improvement in the overall clustering performance. We also demonstrate that the UCO reduces the OFM in DDC, when compared to an analogous autoencoder-based approach.

References

[1] X. Guo, L. Gao, X. Liu, and J. Yin. Improved Deep Embedded Clustering with Local Structure Preservation. In *International Joint Conference on Artificial Intelligence*, 2017.

[2] W. Hu, T. Miyato, S. Tokui, E. Matsumoto, and M. Sugiyama. Learning Discrete Representations via Information Maximizing Self-Augmented Training. In *International Conference on Machine Learning*, 2017.

[3] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning*, 2015.

[4] R. Jenssen, J. C. Principe, D. Erdogmus, and T. Eltoft. The Cauchy–Schwarz Divergence and Parzen Windowing: Connections to Graph Theory and Mercer Kernels. *Journal of the Franklin Institute*, 343(6), 2006.

[5] M. Kampffmeyer, S. Løkse, F. M. Bianchi, L. Livi, A.-B. Salberg, and R. Jenssen. Deep Divergence-Based Approach to Clustering. *Neural Networks*, 113, 2019.

[6] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015.

[7] Y. LeCun, Y. Bengio, and G. Hinton. Deep Learning. *Nature*, 521(7553), 2015.

[8] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11), 1998.

[9] C.-Y. Lee, S. Xie, P. W. Gallagher, Z. Zhang, and Z. Tu. Deeply-Supervised Nets. *Journal of Machine Learning Research*, 38, 2015.

[10] F. Li, H. Qiao, and B. Zhang. Discriminatively Boosted Image Clustering with Fully Convolutional Auto-Encoders. *Pattern Recognition*, 83, 2018.

[11] L. McInnes, J. Healy, and J. Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv:1802.03426 [cs, stat]*, 2018.

[12] L. Metz, N. Maheswaranathan, B. Cheung, and J. Sohl-Dickstein. Meta-Learning Update Rules for Unsupervised Representation Learning. In *International Conference on Learning Representations*, 2019.

[13] N. Mrabah, M. Bouguessa, and R. Ksantini. Adversarial Deep Embedded Clustering: On a better trade-off between Feature Randomness and Feature Drift. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[14] N. Mrabah, N. M. Khan, R. Ksantini, and Z. Lachiri. Deep Clustering with a Dynamic Autoencoder: From Reconstruction towards Centroids Construction. *arXiv:1901.07752 [cs, stat]*, 2020.

[15] S. A. Nene, S. K. Nayar, and H. Murase. Columbia Object Image Library (COIL-100). Techincal Report CUCS-006-96, 1996.

[16] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Neural Information Processing Systems*, 2019.

[17] U. Shaham, K. Stanton, H. Li, R. Basri, B. Nadler, and Y. Kluger. SpectralNet: Spectral Clustering Using Deep Neural Networks. In *International Conference on Learning Representations*, 2018.

[18] D. J. Trosten, A. S. Strauman, M. Kampffmeyer, and R. Jenssen. Recurrent Deep Divergence-Based Clustering for Simultaneous Feature Learning and Clustering of Variable Length Time Series. In *International Conference on Acoustics, Speech and Signal Processing*, 2019.

[19] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv:1708.07747 [cs, stat]*, 2017.

[20] J. Xie, R. Girshick, and A. Farhadi. Unsupervised Deep Embedding for Clustering Analysis. In *International Conference on Machine Learning*, 2016.

[21] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong. Towards K-Means-Friendly Spaces: Simultaneous Deep Learning and Clustering. In *International Conference on Machine Learning*, 2017.

[22] J. Yang, D. Parikh, and D. Batra. Joint Unsupervised Learning of Deep Representations and Image Clusters. In *International Conference on Computer Vision and Pattern Recognition*, 2016.