DAC: Deep Autoencoder-based Clustering, a General Deep Learning Framework of Representation Learning

Si Lu[1] and Ruisi Li

Portland State University

Abstract. Clustering performs an essential role in many real world applications, such as market research, pattern recognition, data analysis, and image processing. However, due to the high dimensionality of the input feature values, the data being fed to clustering algorithms usually contains noise and thus could lead to in-accurate clustering results. While traditional dimension reduction and feature selection algorithms could be used to address this problem, the simple heuristic rules used in those algorithms are based on some particular assumptions. When those assumptions does not hold, these algorithms then might not work. In this paper, we propose DAC, Deep Autoencoder-based Clustering, a generalized data-driven framework to learn clustering representations using deep neuron networks. Experiment results show that our approach could effectively boost performance of the K-Means clustering algorithm on a variety types of datasets.

Keywords: clustering, K-Means, representation learning, deep neuron networks, deep autoencoder

1 Introduction

Clustering is the task of grouping samples such that the ones in the same group are more similar to each other than to the ones in other groups. Nowadays, clustering performs as a basic and essential pre-processing step of many real world applications. For example, it could be used to help with fake news identification [6], document analysis [16], marketing and sales, etc. Specifically, clustering algorithms can figure out useful information for the applications via grouping according to a variety of data similarity metrics and data grouping schemes. For example, similar patches could be used for image denoising [1–3] or depth enhancement [9], and clustering could be used to find good similar patches [8].

To let the samples be properly assigned to different groups(called clusters), meaningful feature values of the samples need to be obtained first. However, in real world applications, the data we get is often of high dimensions [5] and usually contains noise, making the clustering difficult to succeed. For example, in the MNIST dataset [7], each input hand-written digit image has 784 pixels. While we know some pixels (e.g. the ones at image corners) might not be as useful as others(e.g. the ones around image centers), it is difficult to manually distinguish them in clustering.

Traditional dimensionality reduction algorithms, namely Principle Component Analysis(PCA) [10], Linear Discriminant Analysis(LDA) [4], and Canonical Correlation

Analysis(CCA) [13], could be used to reduce the number of features. In addition, feature selection algorithms can be used to select from the original feature values a set of useful and noiseless ones. These algorithms aim to extract the core information given the redundant and correlated input high-dimension data features. However, these algorithms often fail mainly due to two reasons. Firstly, most of them require complex mathematical analysis, which is difficult and time consuming as well. Secondly, their is no single approach that could work for all types of datasets. Different datasets could have different dimensions, data sizes and even might be used in totally different applications. Some datasets are linear and some of them are non-linear. As a result, it is difficult to find a way to generally work on all types of datasets.

Recently, due to the emerging of the powerful deep neuron networks, deep learning-based approaches have been introduced to learn better data representations and achieve appealing performance improvements for clustering algorithms. One simple approach is to learn representations using deep auto-encoders. Specifically, the original input high dimensional features are fed into a encoder that generates a low dimensional output. This output is further fed to a decoder that tries to recover the raw input data as much as possible. However, most of the existing approaches [11, 15] are using images as input and thus using convolutional neuron networks in their work.

In this paper, we propose Deep Clustering Autoencoder, a simple but more general framework for representation learning that takes feature vectors as input. Thus, our approach could be applied to more generalized datasets. In addition, according to the group labels, we propose a scheme to adaptively weight all input features. We combine this estimated weight with the loss function computation during training. Experiment results show that our approach could effectively improve the performance of K-Means clustering algorithm on different types of datasets, namely MNIST, Fashion-MNIST [14], as well as Human Activities and Postural Transitions Data Set (HAPT) [12].

The rest of the paper is organized as follows: in section 2 we describe the overview of our deep autoencoder-based clustering. We then describe the deep autoencoder for representation learning in more details in section 3. We finally show experimental results in section 4 and conclude in section 5.

2 Overview of Deep Autoencoder-based Clustering

Figure 1 shows an overview of our deep autoencoder-based clustering framework. There are two main steps: training and clustering testing. In the training step, a deep autoencoder with an encoder and a decoder is trained using the training set. Here a flattened input vector is fed into the multilayer deep encoder which has a low dimensional learned representation. This learned representation is further fed into a decoder that tries to recover an output of the same size as the input. The training process of this autoencoder tries to reconstruct the input as much as possible. In the following clustering step, we apply the autoencoder to the testing set. The output of the encoder (learned representations) is then fed to a classic K-Means algorithm to do clustering. The learned low dimensional representation vector contains key information of the given input, and thus yield better clustering results.

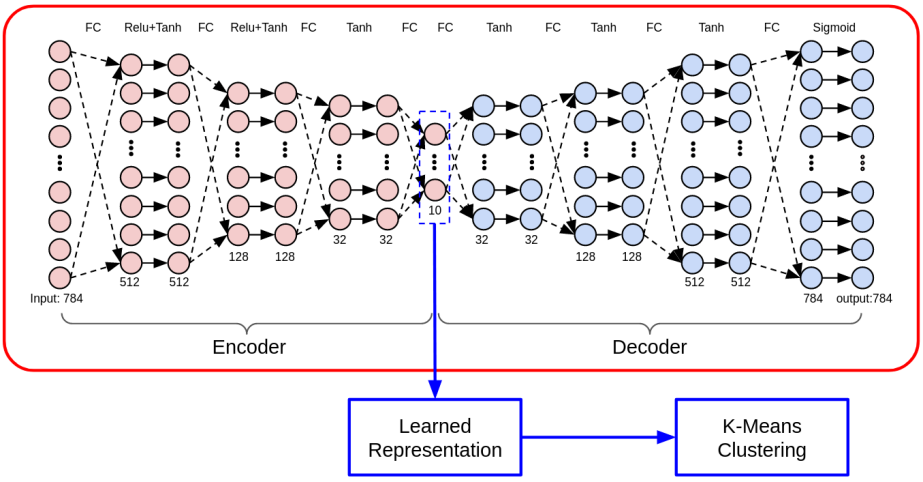

Fig. 1. Overview of our Deep Autoencoder-based Clustering on MNSIT dataset. The autoencoder (consists of an encoder and a decoder) tries to encode and decode the input features such that the decoded output is as close to the input as possible. The input size is 28X28 = 784, the size of the learned low-dimension representation is 10. In the testing stage, the learned encoder output is then fed into the classic K-Means algorithm to do clustering.

3 Deep Autoencoder for Representation Learning

The architecture of our deep autoencoder for representation learning is shown in Figure 1. As could be seen, the model is not as complex as some of the advanced neuron networks. The reason is that we do not want our model to over-fit in two-folds. First, we do not want our model to over-fit on the training dataset over the testing dataset. Second, we do not want our model to over-fit on the reconstruction problem it-self over the clustering problem. Thus, we select a model of reasonable median complexity.

3.1 Encoder

The encoder aims to encode or compress the input data into a smaller size representation, and at the same time preserve as much key information as possible. As shown in Figure 1, the encoder consists of 8 layers, include the input layer and the learned representation output layer. Here the input layer is being normalized such that all its values is in the range of $(0, 1)$. Specifically, from the beginning, each larger layer is fully connected to the next smaller layer followed by a couple of activation layers.

There are mainly two types of activation layers, Relu and Tanh, as shown in Equation 1 and 2. Adding the Relu layers could introduce non-linearity to our model, making it more robust against non-linear input data. The Tanh layer, on the other hand, could transform the data into a normalized range of $(-1, 1)$, to alleviate the gradient vanishing/exploding problem.

$$Relu(x) = max(0, x) \tag{1}$$

$$\begin{aligned} \cosh(x) &= \frac{\mathrm{e}^{x} + \mathrm{e}^{-x}}{2} \\ \sinh(x) &= \frac{\mathrm{e}^{x} - \mathrm{e}^{-x}}{2} \\ \tanh(x) &= \frac{\sinh(x)}{\cosh(x)} = \frac{\mathrm{e}^{x} - \mathrm{e}^{-x}}{\mathrm{e}^{x} + \mathrm{e}^{-x}} \end{aligned} \tag{2}$$

3.2 Decoder

The decoder aims to decode or decompress the encoded output to reconstruct the original input data as much as possible. It contains nine layers, include the input layer, which is the output of the encoder, and the final output layer. Specifically, each smaller layer is fully connected to the next larger layer followed by a Tanh activation layer. In addition, the decoder has a Sigmoid activation layer (shown in Equation 3) at the final stage to enforce the output values lie into the range of $(0, 1)$.

$$Sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

3.3 Objective Function

Clustering-weighted MSE Loss While the goal of the classic autoencoder is to reconstruct the original input as much as possible, it counts each input feature value equally. However, it is possible that each individual input feature contributes differently to the final clustering results. For example, in MNIST dataset, the pixels at the four corners of almost all images are of the same color black (with zero intensity input values), thus have no impact to the final clustering at all. On the other hand, some pixels around the center of the images are likely to perform more important roles. We thus propose a scheme to compute a clustering-weighted MSE loss to let the autoencoder focus more on the reconstruction of more important input feature values, as shown in Equation 4.

Loss →

$$L_{cmse} = \frac{\sum_{i=1}^{n} w_i (y_i - \hat{y}_i)^2}{n} \tag{4}$$

Here w_i is the weight of each feature. It is computed using all ith feature values sampled from a subset of the training dataset with m samples. Denote all ith feature values as $\{x_i k | k = 1, 2, .., m\}$ and the corresponding ground truth group/cluster labels of the m samples $\{l_k | k = 1, 2, .., m\}$. The corresponding feature weight will be

large if both of the two following conditions are met. First, all sampled values in the same groups/clusters have small differences. Second, all sampled values in different groups/clusters have large differences. Thus, the weight is computed as:

$$w_i = \frac{\sum\limits_{l_p=l_q} e^{-(x_{ip}-x_{iq})^2}}{\sum\limits_{l_p=l_q} 1} \cdot \frac{\sum\limits_{l_p\neq l_q} (1 - e^{-(x_{ip}-x_{iq})^2})}{\sum\limits_{l_p\neq l_q} 1} \tag{5}$$

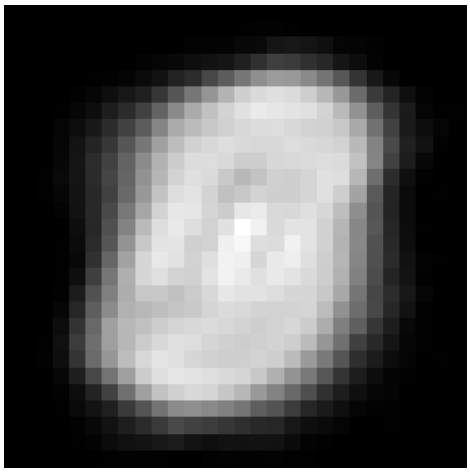

Fig. 2. A map of the clustering weight computed for MNIST dataset using 1000 samples from the training set. It could be seen that pixels at boundaries and corners are less important than the ones around image centers.

Figure 2 shows a map of the clustering weight computed for MNIST dataset using 1000 samples from the training set. Pixels at boundaries and corners are less important than the ones around image centers, thus have smaller weights(white means larger weights).

Final Objective Function The final objective function then combines the Clustering-weighted MSE Loss and a standard L2 norm regularization, as shown in Equation 6. Here the L2 norm regularization L_r is computed using all parameters from the autoencoder. β is a balancing factor with a default value of 0.00001.

$$L = L_{cmse} + \beta \dot{L}_r \tag{6}$$

4 Experimental Results

4.1 Dataset

We evaluate our approach on the classic MNIST hand-written digits dataset. This dataset has $50,000$ images as the training set and $10,000$ images as the testing set. There are 10 groups in total. We show some samples of MNIST dataset in Figure 3.

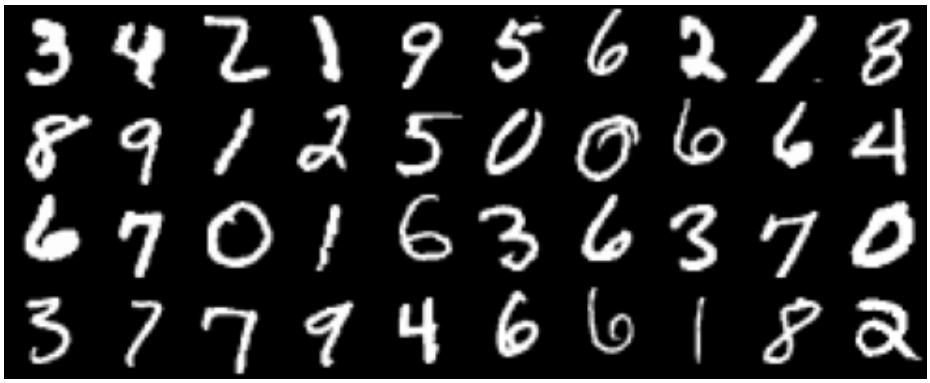

Fig. 3. Samples of the MNIST dataset.

4.2 Measurement Metrics

To evaluate our framework, we apply our trained encoder to the testing dataset. We then compare the generated representations from our trained encoder to the raw input features by applying them to the K-Means algorithm. To measure the performance of clustering algorithms, we use the Adjusted Rand Index (ARI). Specifically, this metrics computes a similarity between two clustering results by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and ground truth clustering results. The proposed approach is denoted as DAC.

4.3 Experiment setup

We implement our framework in Python and PyTorch and test it on a desktop with RTX 2080-Ti. We train the autoencoder for 200 epochs using Adam Optimization Algorithm. The initial learning rate is set to 0.003 and will decrease with the number of epochs during training.

4.4 Results on MNIST

Table 1 shows the quantitative performance of the proposed approach in terms of ARI. Comparing to the raw K-Means algorithm, our approach (DAC) boosts the K-Means algorithm's performace from 0.3477 to 0.6624, which is a 90.50% boost. We also show some of the reconstructed results by our trained autoencoder in Figure 4. It shows that our trained autoencoder can properly reconstruct the raw input hand-written digits.

Table 1. Clustering results on MNIST testing dataset.

	K-Mesn	DAC
ARI	0.3477	0.6624

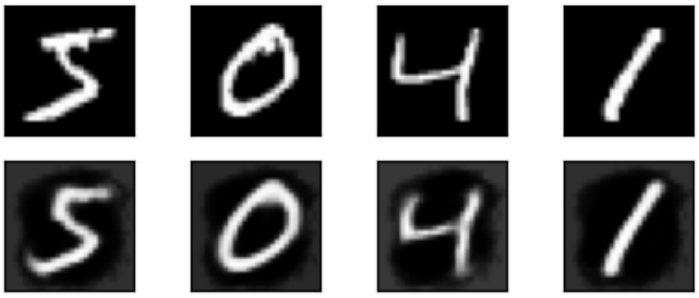

Fig. 4. Sample results of our trained autoencoder on MNIST dataset. Top: raw input images. Bottom: reconstructed images

4.5 Results on other datasets

To test the robustness of our approach against different data types, we apply our method to two other datasets: Fashion-MNIST [14], and Human Activities and Postural Transitions Data Set (HAPT) [12].

Fashion-MNIST is a similar dataset to MNIST, with the same image format and image size. It has $60,000$ images as training set and $10,000$ images as testing set. The only difference is the content: it contains images of 10 types of clothes. The ten categories are shown in Table 2. We show some samples of this dataset in Figure 5.

Human Activities and Postural Transitions Data Set is a dataset that has been captured by smart phone's sensors [12]. The authors captured 3-axial linear acceleration

Table 2. Fashion-MNIST category labels.

T-shirt/top	Trouser	Pullover	Dress	Coat
Sandal	Shirt	Sneaker	Bag	Ankle boot

Fig. 5. Samples of the Fashion-MNIST dataset.

and 3-axial angular velocity at a constant rate of 50Hz using the embedded accelerometer and gyroscope of the device, which is a smartphone (Samsung Galaxy S II). There are 30 volunteers whose ages are in the range of 19-48 years old. In their data capturing experiment, the volunteers was doing one of twelve activities. There are six basic activities: three static postures (standing, sitting, lying) and three dynamic activities (walking, walking downstairs and walking upstairs). Another six postural transitions that occurred between the static postures have also been added to the dataset. These are: stand-to-sit, sit-to-stand, sit-to-lie, lie-to-sit, stand-to-lie, and lie-to-stand. All twelve types of activities are shown in Table 3.

Table 3. HAPT category labels.

walking	walking upstairs	walking downstairs	sitting
standing	laying	stand to sit	sit to stand
standing	laying	stand to sit	sit to stand
sit to lie	lie to sit	stand to lie	lie to stand

The sensor signals (accelerometer and gyroscope) were then denoised by some noise filters. The authors then sampled in fixed-width sliding windows of 2.56 sec and 50% overlap (128 readings/window), leading to a sample size of 561 features. Each sample is captured when the volunteer is doing one type of activities. During the capture process, 70% of the volunteers were randomly selected to generate the training set

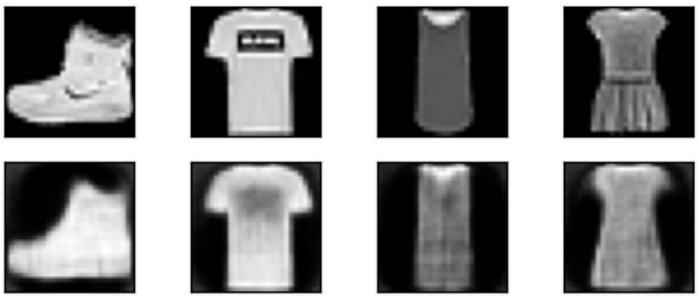

Fig. 6. Sample results of our trained on Fashion-MNIST dataset. Top: raw input images. Bottom: reconstructed images

and 30% were selected to generate the testing set. In total, this dataset has 7767 samples for training and 3162 samples for testing.

We apply our method to Fashion-MNIST dataset and report the results in Table 4. Here as the Fashion-MNIST is a more complex dataset, we modified the autoencoder and show the modified autoencoder architecture in Figure 7. It can be seen that comparing to using raw input features in K-Means clustering, our method boosts ARI from 0.3039 to 0.4702, yields to a improvement of 54.7%.

We then apply our method to the HAPT dataset and report the results in Table 3. Here as this dataset's inputs are of lower dimension than MNIST, we modified the autoencoder accordingly and show the modified autoencoder architecture in Figure 8. It can be seen that even with this temporal sequence dataset, our method could effectively improve the K-Means algorithm's performance by 30%. These results also show that our method could be generally applied to other data types. We also show some of the reconstructed results by our trained autoencoder in Figure 6. It shows that our trained autoencoder can properly reconstruct the raw input fashion images.

Table 4. Clustering results on Fashion-MNIST testing dataset.

	K-Mesn	DAC
ARI	0.3039	0.4702

Table 5. Clustering results on HAPT testing dataset.

	K-Mesn	DAC
ARI	0.4290	0.5594

5 Conclusion

In this paper, we propose DAC, Deep Autoencoder-based Clustering, a generalized data-driven framework to learn low dimensional clustering representations using trained deep neuron networks. Specifically, we train a multi-layer deep autoencoder to encode and decode the raw input samples. The encoded output of the encoder is then fed to a classic K-Means algorithm to do clustering. We design a scheme to compute a clustering-based weight in the training objective function to train the autoencoder and let it focus more on the reconstruction of more important features. Experimental results show that our approach could effectively boost the performance of a classic clustering algorithm: K-Means by 30% to 90% on MNIST dataset. In addtion, our method could be also applied to other types of clustering datasets, such as Fashion-MNIST and Human Activities and Postural Transitions Data Set (HAPT). Experimental results show that our framework could still be able to improve K-Means algorithm's performance by as much as 55%

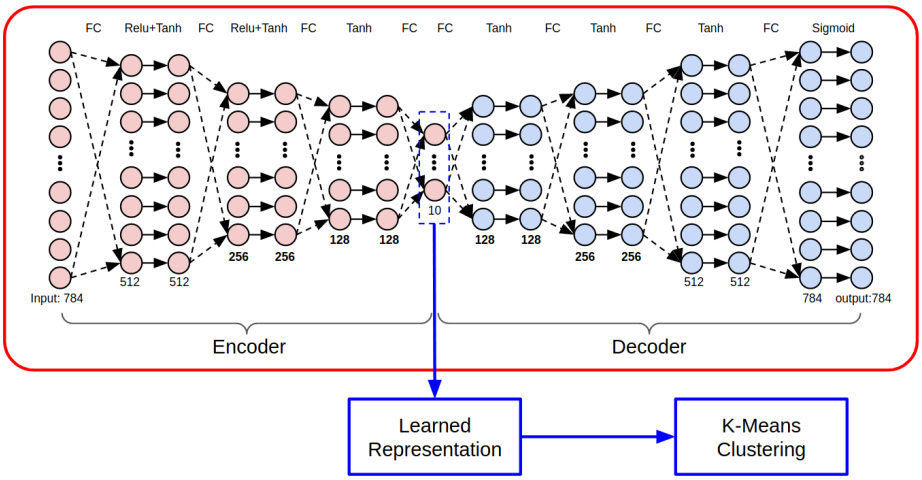

Fig. 7. Overview of our Deep Autoencoder-based Clustering on Fashion-MNSIT dataset.

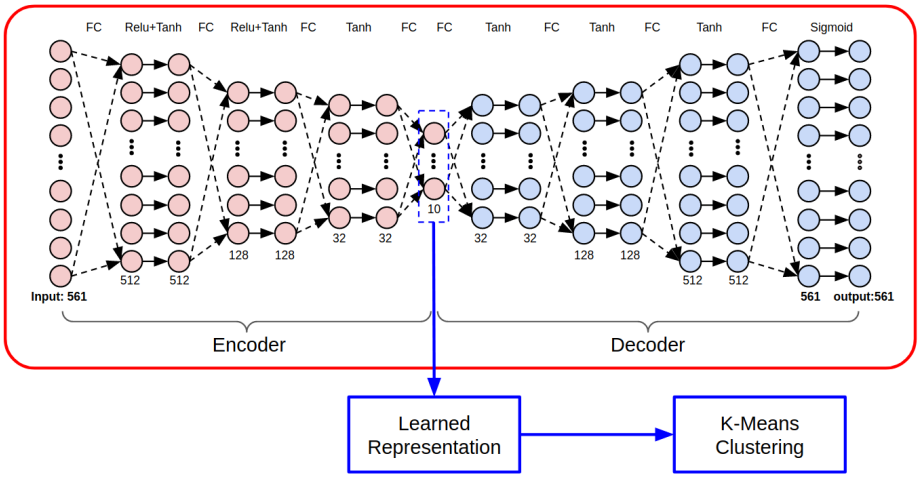

Fig. 8. Overview of our Deep Autoencoder-based Clustering on HAPT dataset.

References

1. Buades, A., Coll, B., Morel, J.: A non-local algorithm for image denoising. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 2, pp. 60–65 (2005)
2. Chen, F., Zhang, L., Yu, H.: External patch prior guided internal clustering for image denoising. In: IEEE International Conference on Computer Vision (ICCV), pp. 603–611 (2015)
3. Dabov, K., Foi, A., Katkovnik, V., Egiazarian, K.: Image denoising by sparse 3-d transform-domain collaborative filtering. IEEE Transactions on Image Processing **16**(8), 2080–2095 (2007)
4. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern classification. john wiley & sons. Inc., New York **2** (2001)
5. Han, J., Pei, J., Kamber, M.: Data mining: concepts and techniques. Elsevier (2011)
6. Hosseinimotlagh, S., Papalexakis, E.E.: Unsupervised content-based identification of fake news articles with tensor decomposition ensembles. In: Proceedings of the Workshop on Misinformation and Misbehavior Mining on the Web (MIS2) (2018)
7. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
8. Lu, S.: Good similar patches for image denoising. In: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 1886–1895. IEEE (2019)
9. Lu, S., Ren, X., Liu, F.: Depth enhancement via low-rank matrix completion. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3390–3397 (2014)
10. Pearson, K.: Liii. on lines and planes of closest fit to systems of points in space. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science **2**(11), 559–572 (1901)
11. Pu, Y., Gan, Z., Henao, R., Yuan, X., Li, C., Stevens, A., Carin, L.: Variational autoencoder for deep learning of images, labels and captions. Advances in neural information processing systems **29**, 2352–2360 (2016)
12. Reyes-Ortiz, J.L., Oneto, L., Samà, A., Parra, X., Anguita, D.: Transition-aware human activity recognition using smartphones. Neurocomputing **171**, 754–767 (2016)
13. Sun, Q.S., Zeng, S.G., Liu, Y., Heng, P.A., Xia, D.S.: A new method of feature fusion and its application in image recognition. Pattern Recognition **38**(12), 2437–2448 (2005)
14. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017)
15. Yang, X., Deng, C., Zheng, F., Yan, J., Liu, W.: Deep spectral clustering using dual autoencoder network. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
16. Zhao, Y., Karypis, G.: Evaluation of hierarchical clustering algorithms for document datasets. In: Proceedings of the eleventh international conference on Information and knowledge management, pp. 515–524 (2002)