A Deep Convolutional Auto-Encoder with Embedded Clustering

A. Alqahtani, M. Ali, X. Xie, M.W.Jones, Deng, J.
Department of Computer Science, Swansea University, Swansea, UK

***Abstract*—In this paper, we propose a clustering approach embedded in deep convolutional auto-encoder. In contrast to conventional clustering approaches, our method simultaneously learns feature representation and cluster assignment through deep convolutional auto-encoder. Deep convolutional auto-encoders have been found effective in image processing as it fully utilizes the properties of convolutional neural networks. Our method consists of clustering and reconstruction objective functions. All data points are assigned to their new corresponding cluster centers during the optimization, after that, clustering centers are iteratively updated to obtain a stable performance of clustering. The experimental results on the MNIST dataset show that the proposed method substantially outperforms deep clustering models in term of clustering quality.**

***Index Terms*—deep learning, convolutional auto-encoder, embedded clustering**

I. Introduction

Clustering is one of the most important topics in the machine learning algorithms. It utilizes to define a grouped structure of unlabelled data. The task of clustering seeks for homogeneous features and group them together based on their similarities. Traditional clustering algorithms can be restricted to demonstrate satisfying performance due to high dimensionality and weak variance of the nature of data. Therefore, dealing with high-level representation provides beneficial components, benefiting the achievement of such a clustering task. As features do not provide any information about an appropriate group for their objects, they help to describe each object in the dataset, allowing a clustering algorithm to learn and extract useful information for its structure.

Deep auto-encoder (DAE) and deep convolutional auto-encoder (DCAE) are, unsupervised models, designed for representation learning. They map inputs into new space representation, allowing to obtain useful features via their internal layer. These features can assist to improve the quality of clustering task. Based on the encode-decode manner, data is transformed, through the encoding part, into new space representation as a set of features. After that, in the decoding part, the obtained features are retrieved into an appropriate approximation of the underlying data representation by minimizing the reconstruction error of the original input. The difference between such deep models is that instead of fully-connected layers in the architecture of DAE, DCAE consists of convolutional layers in the encoding part and deconvolutional layers in the decoding part. DCAE can be beneficial for image processing because it fully utilizes the properties of convolutional neural networks (CNN), which is shown to outperform all other techniques on image data [1].

In this paper, we present a clustering approach embedded into a deep convolutional auto-encoder framework. In contrast to conventional clustering approaches, our method makes use of deep learning approaches, which can help clustering assignment to extract similar patterns in new space representation and find ideal representative centers for distributed data. Our clustering model aims to learn feature representation and cluster assignment simultaneously utilizing the strength of deep convolutional auto-encoder to learn high-level features. This method consists of two objective functions: one is embedded into a convolutional auto-encoder model to minimize the distance between data representations and their identical cluster centers, while the other one is minimizing the reconstruction error. During optimization, all data points are assigned to their new corresponding centroids and then clustering centers are updated iteratively allowing the model to achieve a stable achievement of clustering. The defined clustering objective, as well as the reconstruction objective, are simultaneously optimized to update parameters. We train our clustering model and evaluate the clustering quality utilizing the MNIST data-set. We compare our method with three baseline methods, showing that our model yield substantially better in both reconstruction and clustering quality.

The rest of the paper is organized as follow. In section 2, we present related works, and describe our methodology and approach in section 3. In section 4 and 5, we present our experimental results. Finally, conclusion remarks are given in section 5.

II. Related Works

Deep auto-encoder and deep convolutional auto-encoder have been shown to be efficient approaches to extract unsupervised deep features and reduce dimensionality. These deep models have been exploited for the purpose of clustering. Existing works can be classified into four categories: (1) extracting features through auto-encoder and and then clustering the learned features, (2) extracting features through convolutional auto-encoder and then clustering the learned features, (3) developing embedded clustering approach in an auto-encoder framework, (4) developing embedded clustering approach in a convolutional

auto-encoder framework. The related works that apply those approaches are summarised in Table 1.

TABLE I
DEEP CLUSTERING METHODS

Clustering Approaches	After	Within
AE	[2], [3]	[4], [5]
CAE	[6]	[7]

Extracting features through deep networks (e.g. AE or CAE) allows to learn low-dimensional representation, follows by a clustering algorithm to perform clustering. Even though this method takes advantages of a deep neural network to map the similarity features into a low-dimensional space, which used to obtain clustering-oriented representation, it considers as two-steps procedures, and it separates the learning process to extract features from the clustering task. The representation learning and clustering are not jointly optimized. Huang et al. [3] and Tian et al [2] used AE to learn low-dimensional representation, seeking to obtain effective features used for clustering, thereafter, k-means is applied to cluster the obtained features while Lia at el. [6] utilizes the CAE to learn representation, thereafter, the decoder part is neglected and a soft k-means model is added on top of the encoder to make a unified clustering model.

Developing embedded clustering approach in a deep network allows extracting features and clustering assignments simultaneously. The clustering method can be done via simultaneously considering data reconstruction. This approach learns feature representation and cluster assignment simultaneously using deep networks. A deep network can be used for clustering through utilizing a joint objective function that is embedded into the network, simultaneously minimizing the reconstruction error and clustering objective. This allows clustering data points into their corresponding centroids in a new space. Song et al. [4] and Xie et al. [5] develop embedded clustering in an auto-encoder framework while Guo et al. [7] recently propose clustering with convolutional auto-encoder work.

Most of these methods fundamentally relies on pre-training the parameters, using different settings, while we train our model in an end-to-end way in fixed settings without any pre-training or fine-tuning procedures, enabling faster training process. Our method simultaneously learns effective feature representation and cluster assignment through deep convolutional auto-encoder. We apply deep convolutional auto-encoder because it can be better appropriate in image-processing tasks as it takes advantage of the convolutional neural networks properties [8]. In contrast to Guo work, our proposed method differs in several key respects. First, for clustering approach, instead of clustering with KL divergence, we apply an objective function that restricts the distance between learned feature representations, in a latent space, and their identical centroids, producing a stable representation, which is appropriate for clustering process. Accordingly, the centroids are iteratively updated. Second, our work particularly differs from their approach in the term of architecture, cost functions, and optimization. Finally, our results show that our model yield substantially better for both reconstruction and clustering quality.

III. Approach

In this paper, we use fully unsupervised manner throughout the paper. The proposed clustering approach is embedded in a deep convolutional auto-encoder framework. Our model architecture is shown in Fig. 2. Through this procedure, the model maps a 2D input image, via a series of convolutional layers, into latent representation and then uses deconvolutional layers to reconstruct the data representation to its original shape. An effective representation, via the internal code, can be exploited to support our clustering task. The objective function minimizes the distance between data samples and their identical centroids and the reconstruction error. The following subsections clarify our methodology. We initially explain how a deep convolutional auto-encoder works and then how it is been utilized for our clustering approach.

A. Deep Convolutional Auto-encoder (DCAE)

In contrast to deep auto-encoder (DAE) model, DCAE [9] uses convolutional and deconvolutional layers instead of fully connected layers. DCAE can be better appropriate in image-processing tasks because it takes advantage of the convolutional neural networks (CNN) properties [8]. Local connections and parameter sharing distinguish CNN to have a property in translation latent features [10]. In the encoding part, convolutional layers are used, as feature extractors, to learn features through mapping the data into an internal layer. A latent representation of the n^{th} feature map of the existing layer is given by the following form:

$$h^n = \sigma(x * W^n + b^n) \tag{1}$$

where W is the filters and b is the corresponding bias of the n^{th} feature map, σ is the activation function (e.g. sigmoid, ReLU), and $*$ denotes the 2D convolution operation.

In contrast, the deconvolutional layers invert this process and reconstruct the latent representation back to its original shape, so this process maps the obtained features into pixels [11] by using the following form:

$$y = \sigma(\sum_{n \in H} h^n * \tilde{W}^n + c) \tag{2}$$

where H denotes the group of latent feature maps, $\tilde{W}$ is the flip operation over both dimensions of the weights, c is the corresponding a bias, σ is the activation function, and $*$ denotes the 2D convolution operation.

DCAE allows extracting latent representation through its internal layer by minimizing reconstruction error. We use the cross-entropy (logistic) loss via Eq.(3) because experiments have shown that the euclidean (L2) loss function is not robust to convolutional neural networks designed with deconvolutional layers,

and networks trained with perceptual loss tend to produce much better results [12]–[14]. In a like manner of standard networks, the backpropagation method computes the gradient of the error with respect to all parameters.

$$E_1 = -\frac{1}{N}\sum_{n=1}^{N}(y_n \log \widehat{y}_n + (1-y_n)\log(1-\widehat{y}_n)) \tag{3}$$

Where $\widehat{y}$ is the pixel value of the reconstructed image, and y is the pixel value of the target image. For further detail of convolutional auto-encoder readers can refer to [9].

B. Clustering Embedded on Deep Convolutional Auto-encoder

Using DCAE model described in Section 3.1, we now utilize the strength of such model as training procedure for feature transformations. The goal of our clustering model is to learn feature representation and cluster assignment simultaneously. Using DCAE, as features extractor, supports the achievement of such a clustering process. This idea allows clustering method to deal with learned features instead of raw data. We follow [4] to develop deep clustering model, instead of classic auto-encoder, we apply DCAE for clustering task. Although DCAE provides an effective representation in a new latent space, it does not internally impose compact representation constraints using clustering. Therefore, we add a clustering objective function to the convolutional auto-encoder model, which minimizes the distance between data samples and assigned centroids in latent space follows [4]:

$$E_2 = \lambda \cdot \frac{1}{2N}\sum_{n=1}^{N} \parallel h^t(x_n) - c_n^* \parallel^2 \tag{4}$$

where N denotes the number of samples, λ is clustering weight-parameter that control the contribution percentage of clustering cost function in the overall cost function Eq.(5), $h^t(*)$ is the internal representation obtained by the encoder mapping at the t^{th} iteration, (x_n) is the n^{th} sample in the dataset, and c_n^* is the assigned cluster center to the n^{th} sample. The overall cost function is a combination of two parts: the first part is essentially cross-entropy loss minimizing reconstruction error, while the second part is clustering objective function minimizing the distance between data representations in a latent space and their corresponding cluster centers.

$$\min_{W,b} E_1 + E_2 \tag{5}$$

C. Optimization

At each epoch, our model optimizes two components using stochastic gradient descent and backpropagation: (1) conventional auto-encoder parameters as well as mapping function h, and (2) cluster centers c. At each epoch, the model optimizes the mapping function h, while keeps the cluster centers fixed at c. Thereafter, each obtained new internal representation is assigned to the closest centroid, following [4], this is defined as:

$$c_n^* = arg\min_{c_m^{t-1}} \parallel h^t(x_n) - c_m^{t-1} \parallel^2 \tag{6}$$

where c_m^{t-1} denotes the cluster centers computed at the previous epoch. After each internal representation is assigned to the closest cluster center, the cluster center is updated using the sample assignment computed in the previous epoch via the following equation as [4]:

$$c_m^t = \frac{\sum_{x_n \in c_m^{t-1}} h^t(x_n)}{\sum c_m^{t-1}} \tag{7}$$

where c_m^{t-1} is all samples that belong to the m^{th} cluster at the previous epoch, and $\sum c_m^{t-1}$ is the number of samples that belong to the m^{th} cluster.

D. Architecture

We adopt the base architecture proposed in [15]. Our contributions to the architecture are the following. Firstly, instead of two-loss layers (Cross-entropy loss and Euclidean loss) to minimize reconstruction error, we only use cross-entropy loss as previous studies have shown that euclidean loss function is not robust to convolutional neural networks designed with deconvolutional layers [12]–[14]. Also with only the cross-entropy loss, our experiments have shown that only cross-entropy reconstruction loss can provide good training convergence, Fig. 1 shows loss convergence during training of our model on MNIST. Secondly, we exploit the learned features via the internal layer and feed it to clustering loss which minimizing the distance between data points and their assigned cluster centers, embedding clustering techniques in a deep convolutional auto-encoder model. Thirdly, instead of optimizing conventional auto-encoder to reach optimal reconstruction, we sequentially optimize the mapping function h and cluster centers to obtain efficient clustering results. The final architecture of our model for deep clustering embedded in deep convolutional auto-encoder is presented in Fig. 2.

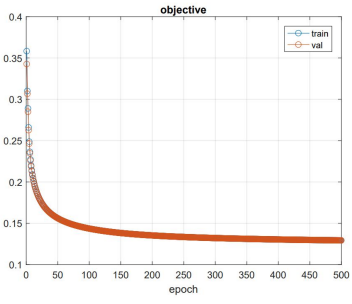

Fig. 1. Loss convergence during training of our model on MNIST

The network architecture consists of two convolutional layers with filter sizes of 9×9 with 8 kernels in the first convolutional layer and 4 kernels in the second convolutional layer. This followed by two fully-connected layers, which have 250 neurons and 10 neurons respectively, in the encoding part. In the decoding part, a single fully-connected layer of 250

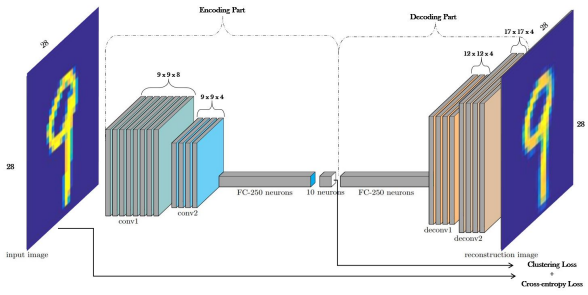

Fig. 2. The architecture of our proposed model. The objective function is a combination of two parts: clustering loss (To minimize the distance between data representations in a latent space and their corresponding cluster centers) and cross-entropy loss (To minimize reconstruction error). In our model, the defined clustering objective, as well as the reconstruction objective, are simultaneously optimized to update parameters in an end-to-end manner.

neurons followed by two deconvolutional layers. The first deconvolutional layer consists of 4 kernels with the size of 12×12, and the second deconvolutional layer consists of 4 kernels with the size of 17×17. As an activation function, a standard sigmoid activation function is utilized.

IV. Experiment

A. Training Details

We train our model end-to-end in an unsupervised manner. We do not perform any pre-training and fine-tuning procedures, which are frequently utilized in the deep approaches particularly deep models that be based on encode-decode manner. MatConvNet [16] is used to train our model. 50,000 unlabeled examples are used for training the model. For parameters optimization, we use stochastic gradient descent with one hundred images in a mini-batch to update all parameters. The gradient is calculated via the cost function Eq.(5). This objective function ensures that the obtained new data representations in the internal layer are close to their identical cluster centers, and meanwhile minimizing the reconstruction error. Throughout our experiment, we use a fixed learning rate of 0.006, a momentum of 0.9, and weight decay of 0.0005. The clustering weight-parameter that controls the contribution percentage of clustering λ is set to 0.2.

For parameters initialization, we initialize all weights in the convolutional auto-encoder network using Xavier initialization [17] and set all biases to 0. With an initial estimate of cluster centroids, we initialize the first cluster centers based on the random initialization.

B. DataSet

- **MNIST** [18] is handwritten digits images of 28×28 pixel size. No modification has been applied to input images, but we just normalize a scale image to be in the range of [0,1]. MNIST labels have only been used as ground truth to evaluate clustering results in the last stage.

V. Evaluation

A. Clustering Evaluation

To justify our model to distinguish between the clustering results generated by our convolutional auto-encoder clustering model and the ground truth labels, two evaluation approaches are used to compute the cluster quality: accuracy (ACC) and normalized mutual information (NMI).

*1) **Normalized Mutual Information (NMI)**:* The NMI is used to measure clustering quality. It is defined between two random variables as [19]:

$$NMI(X;Y) = \frac{I(X;Y)}{\sqrt{H(X)H(Y)}} \tag{8}$$

where X denotes ground truth labels, and Y is the obtained cluster, $I(X;Y)$ is mutual information between X and Y, and $H(X)$ and $H(Y)$ denotes utilized entropies, which normalize the value of mutual information into [0,1] range.

*2) **Accuracy (ACC)**:* Clustering accuracy is another widely used measurement to evaluate clustering results. It is computed between obtained clusters and ground truth labels by using the following form follows [20], [21]:

$$Accuracy = \frac{\sum_{i=1}^{n} \delta(y_i, map(c_i))}{n} \tag{9}$$

where N is the number of samples, y_i denotes ground truth labels, and c_i is obtained clusters, $\delta(y,c)$ is a function that equals one if $y = c$ and zero otherwise, and $map(c_i)$ is permutation function that maps obtained cluster labels into its corresponding ground truth labels.

TABLE II
Comparison of clustering quality in term of NMI and Accuracy

	MNIST	
	NMI	***ACC***
DEC [5]	-	84.30
AEC [4]	66.90	76.00
DCEC [7]	-	85.29
Proposed	**84.97**	**92.14**

*3) **Baseline Methods**:* To demonstrate the effectiveness of our clustering model, we compare our method with three baseline methods. Those clustering methods are embedded in deep networks. Therefore, models can learn features and cluster assignments simultaneously. DEC by Xie et al. [5] and AEC by Song et al. [4] learn feature representation utilizing deep auto-encoder while DCEC by Guo et al. [7] learn feature representation utilizing convolutional auto-encoder.

*4) **Results**:* Our results for both ACC and NMI on MNIST test-set with all compared methods results are presented in table 2. From the results, we can see that the models, which utilizing convolutional auto-encoder to learn feature representation (i.e. our clustering model and DCEC by Guo et al. [7]), perform much better compared with the models that utilize conventional auto-encoder because CAE takes advantage of CNN as it's better appropriate in image-processing

tasks. Indeed, our method makes use of deep learning approaches, which can assist clustering assignment to extract similar patterns in new space representation and find ideal representative centers for distributed data. It particularly takes advantage of deep convolutional auto-encoder to preserve the local structure of the data and have a property to learn latent features. Comparing with all three baselines, our clustering results demonstrate that the proposed model substantially outperforms all embedded clustering approach in teams of NMI and ACC. I

Fig. 3. Visualization of input and reconstruction images with respect to digits 0, 3, 5 and 9

B. Visualization

We consider not only clustering quality, but the reconstruction quality of the DCAE is also our focus because it is highly desirable to obtain efficient clustering results as well as typical reconstruction quality. As convolutional auto-encoder is trained to transform the data into latent representation and then reconstruct the original input or obtain an appropriate approximation of the underlying data representation by minimizing the reconstruction error. We visualize original inputs and reconstruction images provided by our model, allowing to visually differentiate and evaluate how idealistic our model reconstruct original images. As a result, our model reconstructs the original images into ideally equivalent shape. Fig. 4 shows the quality of reconstruction images for random examples of MNIST test-set. From Fig. 4, we can see that our model tends to reach optimal reconstruction and the original data points are retrieved as flawlessly as possible. The reconstructed images look qualitatively similar to original ones.

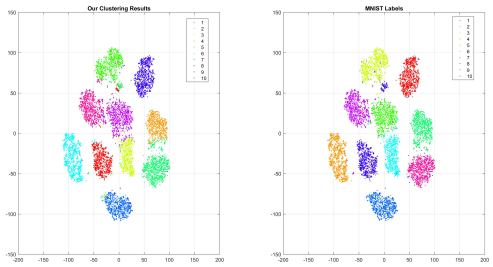

Fig. 4. Visualization of Latent Representation

In addition, we apply the t-SNE visualization method by [22] to evaluate our clustering results. Fig. 5 shows latent representation via the internal code for our clustering results and MNIST labels. The visualization shows the compactness of 10 clusters. We use 10 different colors to indicate data labels and the clustering results of our model. From visualization, we can notice that each color, which indicates whether data labels or our clustering results, determines certain compact group (cluster), shown typical clustering result.

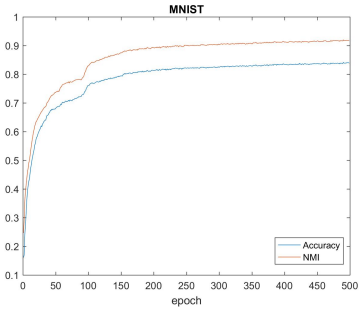

Fig. 5. Changing of accuracy and NMI during training on MNIST

Lastly, Fig.3 shows the change of accuracy and NMI during training, which demonstrates the influence of each epoch in our clustering method. This allows noticing that the performance is enhanced at each epoch, shown that the effectiveness of deep clustering model.

VI. Conclusion

In this paper, we introduce a deep clustering method based on deep convolutional auto-encoder. Our clustering model is embedded in a deep convolutional auto-encoder allowing to learn a powerful non-linear mapping to obtain compact representation. Our model contains two objective functions minimizing (1) the distance between latent representations in the new space and their corresponding cluster centers to obtain compact groups (2) the reconstruction error. The experimental results have demonstrated the effectiveness of our proposed method to cluster data into their appropriate group. Our potential future work is to experiment more difficult datasets and improve the accuracy of such deep clustering model.

References

[1] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[2] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, "Learning deep representations for graph clustering." in *AAAI*, 2014, pp. 1293–1299.

[3] P. Huang, Y. Huang, W. Wang, and L. Wang, "Deep embedding network for clustering," in *Pattern Recognition (ICPR), 2014 22nd International Conference on*. IEEE, 2014, pp. 1532–1537.

[4] C. Song, F. Liu, Y. Huang, L. Wang, and T. Tan, "Auto-encoder based data clustering," in *Iberoamerican Congress on Pattern Recognition*. Springer, 2013, pp. 117–124.

[5] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *International Conference on Machine Learning*, 2016, pp. 478–487.

[6] F. Li, H. Qiao, B. Zhang, and X. Xi, "Discriminatively boosted image clustering with fully convolutional auto-encoders," *arXiv preprint arXiv:1703.07980*, 2017.

[7] X. Guo, X. Liu, E. Zhu, and J. Yin, "Deep clustering with convolutional autoencoders," in *International Conference on Neural Information Processing*. Springer, 2017, pp. 373–382.

[8] Y. Wang, Z. Xie, K. Xu, Y. Dou, and Y. Lei, "An efficient and effective convolutional auto-encoder extreme learning machine network for 3d feature learning," *Neurocomputing*, vol. 174, pp. 988–998, 2016.

[9] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," *Artificial Neural Networks and Machine Learning–ICANN 2011*, pp. 52–59, 2011.

[10] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[11] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2018–2025.

[12] V. A. Knyaz, O. Vygolov, V. V. Kniaz, Y. Vizilter, V. Gorbatsevich, T. Luhmann, N. Conen, W. Forstner, K. Khoshelham, S. Mahendran *et al.*, "Deep learning of convolutional auto-encoder for image matching and 3d object reconstruction in the infrared range," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2155–2164.

[13] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2536–2544.

[14] M. T. McCann, K. H. Jin, and M. Unser, "A review of convolutional neural networks for inverse problems in imaging," *arXiv preprint arXiv:1710.04011*, 2017.

[15] V. Turchenko and A. Luczak, "Creation of a deep convolutional auto-encoder in caffe," in *2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, vol. 2, Sept 2017, pp. 651–659.

[16] A. Vedaldi and K. Lenc, "Matconvnet: Convolutional neural networks for matlab," in *Proceedings of the 23rd ACM International Conference on Multimedia*, ser. MM '15. New York, NY, USA: ACM, 2015, pp. 689–692. [Online]. Available: http://doi.acm.org/10.1145/2733373.2807412

[17] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[18] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[19] A. Strehl and J. Ghosh, "Cluster ensembles—a knowledge reuse framework for combining multiple partitions," *Journal of machine learning research*, vol. 3, no. Dec, pp. 583–617, 2002.

[20] M. Wu and B. Schölkopf, "A local learning approach for clustering," in *Advances in neural information processing systems*, 2007, pp. 1529–1536.

[21] W. Y. Chen, Y. Song, H. Bai, C. J. Lin, and E. Y. Chang, "Parallel spectral clustering in distributed systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 3, pp. 568–586, March 2011.

[22] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.