# Supplementary material of "Towards K-means-friendly Spaces: Simultaneous Deep Learning and Clustering"

**Bo Yang** [1] **Xiao Fu** [1] **Nicholas D. Sidiropoulos** [1] **Mingyi Hong** [2]

## 1. Additional Synthetic-Data Experiments

### 1.1. Additional Generative Models

In this section, we provide two more examples to illustrate the ability of DCN in recovering K-means-friendly spaces under different generative models. We first consider the transformation as follows:

$$\boldsymbol{x}_i = \left(\sigma(\boldsymbol{W}\boldsymbol{h}_i)\right)^2, \tag{1}$$

where $\sigma(\cdot)$ is the sigmoid function as before and $\boldsymbol{W} \in \mathbb{R}^{100\times 2}$ is similarly generated as in the paper. We perform elementwise squaring on the result features to further complicate the generating process. The corresponding results can be seen in Fig. 1 of this supplementary document. One can see that a similar pattern as we have observed in the main text is also presented here: The proposed DCN recovers a 2-D K-means-friendly space very well and the other methods all fail.

In Fig. 2, we test the algorithms under the generative model

$$\boldsymbol{x}_i = \tanh\left(\sigma(\boldsymbol{W}\boldsymbol{h}_i)\right), \tag{2}$$

where $\boldsymbol{W} \in \mathbb{R}^{100\times 2}$. Same as before, the proposed DCN gives very clear clusters in the recovered 2-D space.

The results in this section and the synthetic-data experiment presented in main text are encouraging: Under a variety of complicated nonlinear generative models, DCN can output clustering-friendly latent representations.

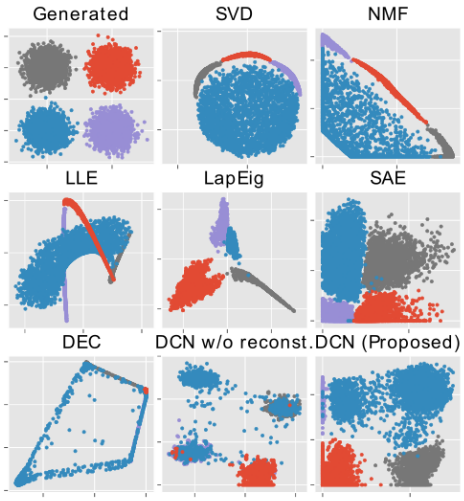


*Figure 1.* The generated latent representations $\{\boldsymbol{h}_i\}$ in the 2-D space and the recovered 2-D representations from $\boldsymbol{x}_i \in \mathbb{R}^{100}$, where $\boldsymbol{x}_i = \left(\sigma(\boldsymbol{W}\boldsymbol{h}_i)\right)^2$.

## 2. Additional Real-Data Experiments

### 2.1. Pendigits

Beside the real datasets in the paper, we also conduct experiment on the Pendigits dataset. The Pendigits dataset consists of 10,992 data samples. Each sample records 8 coordinates on a tablet, on which a subject is instructed to write the digits from 0 to 9. So each sample corresponds to a vector of length 16, and represents one of the digits. Note that this dataset is quite different from MNIST – each digit in MNIST is represented by an image (pixel values) while digits in Pendigits are represented by 8 coordinates of the stylus when a person was writing a certain digit. Since each digit is represented by a very small-size vector of length 16, we use a small network who has three forward layers which are with 16, 16, and 10 neurons. Table 1 shows the results: The proposed methods give the best clustering performance compared to the competing methods, and the methods using DNNs outperform the 'shallow' ones that do not use

[1]Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis MN 55455, USA. [2]Department of Industrial and Manufacturing Systems Engineering, Iowa State University, Ames, IA 50011, USA. Correspondence to: Bo Yang <yang4173@umn.edu>, Xiao Fu <xfu@umn.edu>, Nicholas D. Sidiropoulos <nikos@ece.um.edu>, Mingyi Hong <mingyi@iastate.edu>.

*Proceedings of the $34^{th}$ International Conference on Machine Learning*, Sydney, Australia, PMLR 70, 2017. 

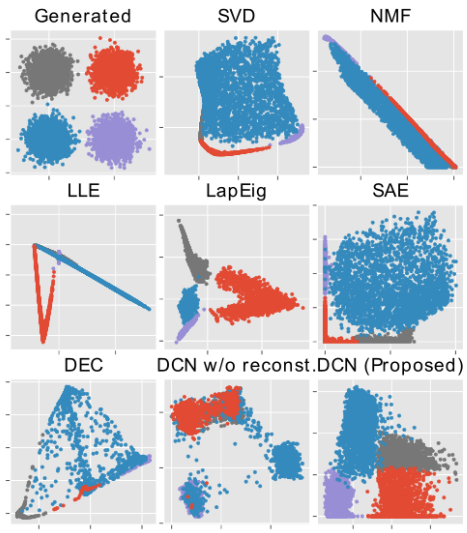


*Figure 2.* The generated latent representations $\{\boldsymbol{h}_i\}$ in the 2-D space of the recovered 2-D representations from $\boldsymbol{x}_i \in \mathbb{R}^{100}$, where $\boldsymbol{x}_i = \tanh(\sigma(\boldsymbol{W}\boldsymbol{h}_i))$.

*Table 1.* Evaluation on the Pendigits dataset

| Methods | DCN | SAE+KM | SC | KM |
|---|---|---|---|---|
| NMI | **0.69** | 0.65 | 0.67 | 0.67 |
| ARI | **0.56** | 0.53 | 0.55 | 0.55 |
| ACC | **0.72** | 0.70 | 0.71 | 0.69 |

neural networks for DR.

### 2.2. DCN as Feature Learner

We motivate and develop DCN as a clustering method that directly works on unlabeled data. In practice, DCN can also be utilized as a feature-learning method when training samples are available – i.e., one can feed labeled training data to DCN, tune the parameters of the network to learn well clustered latent representations of the training samples, and then use the trained DCN (to be specific, the forward network) to reduce dimension of unseen testing data.

Here, we provide some additional results to showcase the feature-learning ability of DCN. We perform a 5-fold cross-validation experiment on the raw MNIST dataset, where each fold is a 80/20 training/testing random split. The performance of SAE+KM on the training sets is presented as a baseline.

The obtained NMI, ARI, and ACC (mean and standard deviation) are listed in Table. 2. One can see that the training and testing stages of DCN output similar results, which is rather encouraging. This experiment suggests that DCN is very promising as a representation learner.

*Table 2.* The mean (stand deviation) of the evaluation results of the 5-fold cross-validation on MNIST.

| | NMI | ARI | ACC |
|---|---|---|---|
| DCN-Training | 0.80 (0.001) | 0.74 (0.002) | 0.83 (0.002) |
| DCN-Testing | 0.81 (0.003) | 0.75 (0.005) | 0.83 (0.004) |
| SAE+KM | 0.73 (0.001) | 0.67 (0.002) | 0.80 (0.002) |

## 3. Detailed Settings of Real-Data Experiments

### 3.1. Algorithm Parameters

There is a set of parameters in the proposed algorithm which need to be pre-defined. Specifically, the learning rate $\alpha$, the number of epochs $T$ (recall that one epoch responds to a pass of all the data samples through the network), and the balancing regularization parameter $\lambda$. These parameters vary from case to case since they are related to a number of factors, e.g., dimension of the data samples, total number of samples, scale (or energy) of the samples, etc. In practice, a reasonable way to tune these parameters is through observing the performance of the algorithm under various parameters on a small validation subset whose labels are known.

Note that the proposed algorithm has two stages, i.e., pre-training and the main algorithm and they usually use two different sets of parameters since the algorithmic structure of the two stages are quite different (to be more precise, the pre-training state does not work with the whole network but only deals with a pair of encoding-decoding layers greedily). Therefore, we distinguish the parameters of the two stages as listed in Table 3, to better describe the settings.

We implement SGD for solving the subproblem w.r.t. $\mathcal{X}$ using the Nesterov-type acceleration (Nesterov, 2013), the mini-batch version, and the momentum method. Batch normalization (Ioffe & Szegedy, 2015) that is recently proven to be very effective for training supervised deep networks is also employed. Through out the experiments, the momentum parameter is set to be $0.9$, the mini-batch size is selected to be $\approx 0.01 \times N$, and the other parameters are adjusted accordingly in each experiments – which will be described in detail in the next section.

### 3.2. Network Parameters

The considered network has two parts, namely, the forward encoding network that reduces the dimensionality of the data and the decoding network that reconstructs the data. We let two networks to have a mirrored structure of each other. There are also two parameters of a forward network, i.e., the width of each layer (number of neurons) and the depth of the network (number of layers). There is no strict rule for setting up these two parameters, but the rule of thumb is to adjust them according the amounts of data sam-

*Table 3.* List of parameters used in DCN.

| Notations | Meaning |
|---|---|
| $\lambda$ | regularization parameter |
| $\alpha_p$ | base pre-training stepsize |
| $\alpha_l$ | base learning stepsize |
| $T_p$ | pre-traing epochs |
| $T_l$ | learning epochs |

ples of the datasets and the dimension of each sample. Using a deeper and wider network may be able to better capture the underlying nonlinear transformation of the data, as the network has more degrees of freedom. However, finding a large number of parameters accurately requires a large amount of data since the procedure can be essentially considered as solving a large system of nonlinear equations – and finding more unknowns needs more equalities in the system, or, data samples in this case. Therefore, there is a clear trade-off between network depth/width and the overall performance.

### 3.3. Detailed Parameter Settings

The detailed parameter settings for experiments on RCV1-v2 are shown in Tables 4. Parameter settings for 20Newsgroup, raw MNIST, pre-processed MNIST, and Pendigits are shown in Tables 5, 6, 7, and 8, respectively.

## 4. More Discussions

We have the following several more points as further discussion:

1. We have observed that runing SAE for epochs may even worsen the clustering performance in the two-stage approach. In Fig. 3, we show how the clustering performance indexes change with the epochs when we run SAE without K-means regularization.One can see that the performance in fact becomes worse compared to merely using pre-training (i.e., initialization). This means that using the SAE does not necessarily help clustering – and this supports our motivation for adding a K-means-friendly structure-enhancing regularization.

2. To alleviate the effect brought by the intrinsic randomness of the algorithms, e.g., random initialization of pre-training, the reported results are all obtained via running the experiments several times and taking average (specifically, we run the experiments with smaller size, i.e., 20Newsgroup, raw and processed MNIST, and Pendigits for ten times and the results of the much larger dataset RCV-v2 are average of five runs; the results for DEC in Table 1 is from a single run.). Therefore, the presented results reflect the performance of the algorithms in an average sense.

3. We treat this work as a proof-of-concept: Joint DNN learning and clustering is a highly viable task according to our design and experiments. In the future, many practical issues will be investigated – e.g., designing theory-backed ways of setting up network and algorithm parameters. Another very intriguing direction is of course to design convergence-guaranteed algorithms for optimizing the proposed criterion and its variants. We leave these interesting considerations for future work.

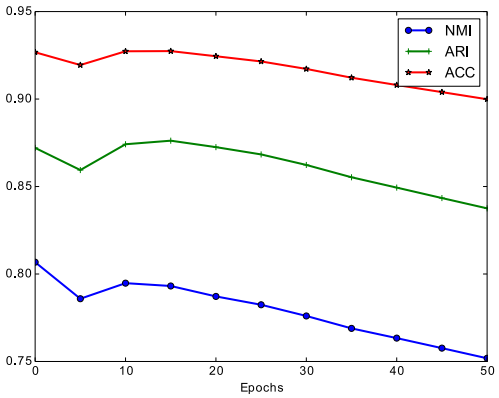


*Figure 3.* Clustering performance degrades when training with only reconstruction error term. This is in sharp contrast with Figure 5(a) in the paper, where clustering performance improves when training the proposed DCN model.

*Table 4.* Parameter settings for RCV1-v2

| parameters | description |
|---|---|
| $\boldsymbol{f}(\boldsymbol{x}_i;\mathcal{W})$: $\mathbb{R}^M \to \mathbb{R}^R$ | $M = 2,000$ and $R = 50$ |
| Sample size $N$ | 178,603 or 267,466 |
| forward net. depth | 5 layers |
| layer width | 2000/1000/1000/1000/50 |
| $\lambda$ | 0.1 |
| $\alpha_p$ | 0.01 |
| $\alpha_l$ | 0.05 |
| $T_p$ | 50 |
| $T_l$ | 50 |

*Table 7.* Parameter settings for Pre-Processed MNIST

| parameters | description |
|---|---|
| $\boldsymbol{f}(\boldsymbol{x}_i;\mathcal{W})$: $\mathbb{R}^M \rightarrow \mathbb{R}^R$ | $M = 10$ and $R = 5$ |
| Sample size $N$ | 70,000 |
| forward net. depth | 3 layers |
| layer width | 50/ 20/ 5 |
| $\lambda$ | 0.1 |
| $\alpha_p$ | 0.01 |
| $\alpha_l$ | 0.01 |
| $T_p$ | 10 |
| $T_l$ | 50 |

*Table 8.* Parameter settings for Pendigits

| parameters | description |
|---|---|
| $\boldsymbol{f}(\boldsymbol{x}_i;\mathcal{W})$: $\mathbb{R}^M \rightarrow \mathbb{R}^R$ | $M = 16$ and $R = 10$ |
| Sample size $N$ | 10,992 |
| forward net. depth | 3 layers |
| layer width | 50/ 16/ 10 |
| $\lambda$ | 0.5 |
| $\alpha_p$ | 0.01 |
| $\alpha_l$ | 0.01 |
| $T_p$ | 50 |
| $T_l$ | 50 |

*Table 5.* Parameter settings for 20Newsgroup

| parameters | description |
|---|---|
| $\boldsymbol{f}(\boldsymbol{x}_i;\mathcal{W})$: $\mathbb{R}^M \rightarrow \mathbb{R}^R$ | $M = 2,000$ and $R = 20$ |
| Sample size $N$ | 18,846 |
| forward net. depth | 3 layers |
| layer width | 250/100/20 |
| $\lambda$ | 10 |
| $\alpha_p$ | 0.01 |
| $\alpha_l$ | 0.001 |
| $T_p$ | 10 |
| $T_l$ | 50 |

*Table 6.* Parameter settings for raw MNIST

| parameters | description |
|---|---|
| $\boldsymbol{f}(\boldsymbol{x}_i;\mathcal{W})$: $\mathbb{R}^M \rightarrow \mathbb{R}^R$ | $M = 784$ and $R = 50$ |
| Sample size $N$ | 70,000 |
| forward net. depth | 4 layers |
| layer width | 500/ 500/ 2000/10 |
| $\lambda$ | 0.05 |
| $\alpha_p$ | 0.01 |
| $\alpha_l$ | 0.05 |
| $T_p$ | 50 |
| $T_l$ | 50 |

## References

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 448–456, 2015.

Nesterov, Y. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.