Residential Load Profile Clustering via Deep Convolutional Autoencoder

Seunghyoung Ryu*, Hyungeun Choi*, Hyoseop Lee†, Hongseok Kim* and Vincent W.S. Wong‡
*Department of Electronic Engineering, Sogang University, Seoul, Korea
†Encored Technologies, Inc., Seoul, Korea
‡Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, Canada
Email : {shryu, hyungeun, hongseok}@sogang.ac.kr*, hslee@encoredtech.com†, vincentw@ece.ubc.ca‡

***Abstract*—In energy data analytics, load profile clustering is essential for various smart grid applications such as demand response, load forecasting, and tariff design. Most of the conventional clustering techniques are based on a representative time domain load profile within a certain period, and the daily and seasonal variations are not well captured. In this paper, we propose a deep learning based customer load profile clustering framework that jointly captures daily and seasonal variations. By leveraging convolutional autoencoder (CAE), the *yearly* load profile in the time domain is converted into a representative vector in the smaller dimensional encoded space. The clusters are then determined based on the vectors encoded by the CAE. We apply the proposed framework to 1,405 households' yearly load profiles and verify that the trained CAE can encode those load profiles into approximately 100 times smaller dimensional space. The encoded load profiles can be decoded by the CAE with a negligible loss between 1–3%. The clustered load images can visualize both daily and seasonal variations, and clustering in the encoded space speeds up the clustering process by almost three orders of magnitude.**

I. Introduction

Recent developments in the smart grid have a significant impact on the power systems. Smart meters and advanced metering infrastructure (AMI) are rapidly being deployed these days. According to the United States Energy Information Administration, 70.8 million smart meters were installed in 2016, and 88% of them are in the residential area [1]. The European Commission requires the European Union member countries to have at least 80% of consumers equipped with smart meters by 2020 [2]. With a large number of smart meters, utilities can now monitor low/medium voltage loads including residential and commercial buildings with finer granularity, e.g., 15-minutes interval. Analyzing a large volume of data generated by smart meters is crucial to better understand the consumption pattern of electricity customers, and support applications such as demand side management. Specifically, clustering plays an important role in such data analytics.

In the literature, clustering is mostly performed using the *daily* load profiles. For example, the work of [3] compared K-means clustering, hierarchical clustering, and follow-the-leader clustering with representative daily load profiles. To capture the variability of daily load profiles, the concept of entropy, which quantifies the cluster variation within one year, is applied in [4]. In [5], four seasonal load profiles per household are derived and clustered by K-means clustering with home owner survey data. In [6], a two-stage load profile clustering combined with fast wavelet transform and g-means clustering is introduced. Multi-resolutional clustering of load profiles in spectral domain is introduced in [7].

Most of the existing works focus on clustering with one or a few representative load profiles, e.g., a collection of average load profiles within a predetermined time period. However, it is not clear whether customers can be characterized with a few time-averaging representative load profiles. Kwac *et al.* in [4] reported that even though two customers may have the same average load profiles, the raw daily load profiles can vary significantly. To cluster load profiles while considering both year-round daily and seasonal variations, one approach is to cluster customers with year-round raw load data directly. However, it may not be efficient due to the large portion of redundant load profiles induced by periodic human behavior. Furthermore, when it is applied to clustering a large group of customers, the computation complexity can be significant. To reduce the complexity, in [8], only 5% of randomly selected hourly electricity consumption from yearly load data is used for clustering customers. Another approach is to use several selected representative daily load profiles. However, it is challenging to derive the representative load profiles and determine how many representatives are sufficient to characterize a customer.

To address the above challenges, in this paper, we propose a novel framework of clustering customers based on the encoded load profile using deep learning. Deep learning has been applied to smart grid applications, e.g., load forecasting [9], system monitoring [10], and clustering [11]. The proposed clustering framework mainly consists of the encoding stage and a clustering stage. In the encoding stage, yearly load data is encoded through convolutional autoencoder (CAE), while preserving the characteristics of electricity consumption pattern. Then, in the clustering stage, the encoded data is being clustered by using K-means clustering. We summarize our main contributions as follows:

- We propose a novel framework of deep learning-based load profile clustering that fully exploits the *year-round* hourly load data to capture both daily and seasonal patterns. The proposed CAE-based load profile clustering is verified by real data from 1,405 nationwide residential customers.

- By using the proposed CAE, a year-round load profile in 8,640 dimensional space is converted into a vector in 100 dimensional encoded space. This vector can be decoded to recover the original yearly load profile with only 1–3% of reconstruction error.
- With the significant dimension reduction by CAE, the required memory space and computational time in clustering is also reduced in proportion, thus making the proposed technique suitable for big data analysis in smart grid applications. Clustering in the encoded space requires only 1/500 computational time compared to clustering in the original space.

The rest of this paper is organized as follows. We present an overview of the proposed framework in Section II. Section III describes the structural design of CAE. In Section IV, we present the CAE-based clustering results with real data and verify the effectiveness of the proposed framework. Finally, we conclude the paper in Section V.

II. The Proposed CAE-based Load Profile Clustering Framework

The overall framework of the proposed CAE-based load profile clustering is illustrated in Fig. 1. The framework can be divided into three main steps: data preprocessing, CAE encoding, and clustering. In the data preprocessing step, we collect target customers' load data from an established metering database. Then, the collected load data undergoes several substages of data cleansing, normalization, augmentation, and transformation to provide proper input for training CAE. In the encoding step, CAE is trained with the preprocessed data. Once the structure of CAE is determined, load data is encoded by CAE. In the clustering stage, the encoded data is clustered by K-means clustering. In order to validate the goodness of the clustering results, validation indices are calculated to select a proper value of K. Finally, we analyze the load data associated with the clustering results, and the results can be utilized for smart grid applications. In the following subsection, we describe the proposed framework in detail.

A. Data Preprocessing for Convolutional Autoencoder

Due to the periodic behavior of human, time-series electricity load data has daily and weekly periodicity. Our objective is to learn and extract periodic and seasonal characteristics of load data by fully utilizing load data of one year through CAE. We consider the 2-dimensional load image which is used in convolutional neural network (CNN).

1) Data cleansing: The real data we used in this paper is collected as part of a demand response pilot program for small-scale customers in South Korea. The dataset includes household customers' hourly load data from smart meters. Note that the typical housing type in South Korea is apartment complex and there is relatively small difference in per house maximum load. Moreover, no electric vehicles are included in this pilot program. Among the households dispersed all around the country, we extract 1,405 households that have full records from March 1, 2016 to February 23, 2017 (i.e., 360 days).

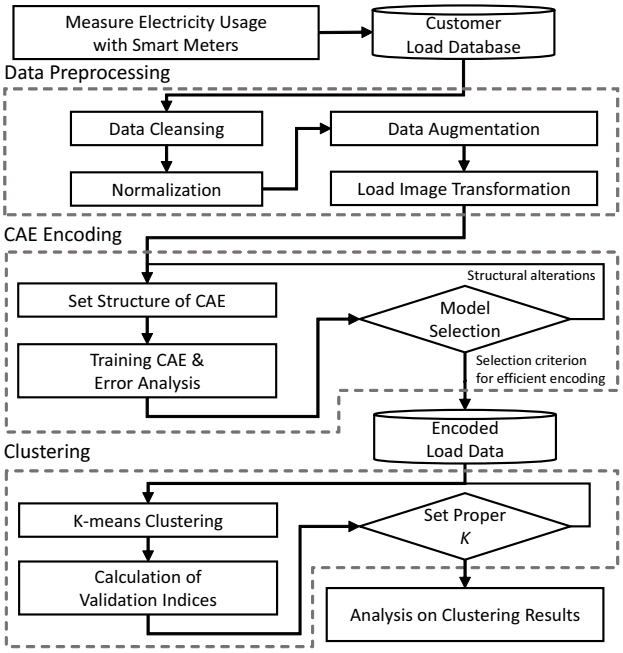


Fig. 1. The framework of CAE-based load profile clustering.

The abnormal data (missing values and unreasonably excessive consumption) is replaced by the average consumption of highly correlated time intervals. We select the number of days to be 360 because it is the largest number of days in a year while having small prime numbers $(2, 3, 5)$ that can be used as *strides* in the convolutional layers of CAE. In addition, by starting from March 1, load data covers all four seasons and each season occupies a region of load image in order. Consequently, a yearly load profile $\mathbf{x}^n$ for household n is $\{x_t^n\}$ with $|\mathbf{x}^n| = 8{,}640$, where t represents an hour index from March 1, 2016 to February 23, 2017.

2) Normalization: After the data cleansing stage, the data is normalized for better training as typically done in deep learning [9]. There are several ways in normalizing the data. In this paper, minmax normalization is adopted per household to focus on the *patterns* of load variations of individual households. The normalized yearly load profile $\mathbf{l}^n = \{l_t^n\}$ is

$$l_t^n = \frac{x_t^n - \min(\mathbf{x}^n)}{\max(\mathbf{x}^n) - \min(\mathbf{x}^n)}, \tag{1}$$

which is between zero and one. By normalization, *patterns* of load variations can better be captured.

3) Data augmentation and load image transformation: For better training of CAE, we enlarge the size of the data by generating virtual households load data set of $\hat{\mathbf{l}}^n = \{\hat{l}_t^n\}$, where $\hat{l}_t^n$ follows the Gaussian distribution $\mathcal{N}\left(l_t^n, (0.05 \times l_t^n)^2\right)$. Then, the load data $\mathbf{l}^n$ and $\hat{\mathbf{l}}^n$ are transformed into 2-dimensional load images. The time index t of l_n^t is converted to (d, h), which is a pair of date index $d = \lfloor t/24 \rfloor$ and an hour index $h = \text{mod}(t, 24)$. From 1-dimensional load profile $\mathbf{l}^n$, a 2-dimensional load image for household n is given by

$$L^n = (l_{d,h}^n) \in \mathbb{R}^{360 \times 24}. \tag{2}$$

B. Convolutional Autoencoder

An *autoencoder* is an artificial neural network trained to encode a set of data into a lower dimension. When the input data, denoted by $\mathbf{x}$, is fed into the autoencoder, it is nonlinearly transformed into an encoded vector, denoted by $\mathbf{z}$, while passing through multiple fully-connected layers in encoder. Next, from the encoded vector $\mathbf{z}$, the output data $\mathbf{x}'$ is reconstructed through the decoder. In training, the autoencoder updates the weights in order to minimize the reconstruction error $||\mathbf{x} - \mathbf{x}'||^2$ by using the back propagation algorithm.

CAE is an autoencoder in which nonlinear transformation is obtained by CNN. A *feature map* is produced by taking convolution operation between the input (or the previous feature) matrix and the learnable filter. The convolution operator * for matrices $\mathbf{A}$ and $\mathbf{B}$ is defined as [12]

$$\mathbf{A}(i,j)*\mathbf{B}(i,j) = \sum_{\tau_1=-\infty}^{\infty} \sum_{\tau_2=-\infty}^{\infty} \mathbf{A}(i-\tau_1, j-\tau_2)\mathbf{B}(\tau_1,\tau_2),$$

where i and j are the row and column indices, respectively. Thus, a feature map is also a matrix. The pixel of a feature map is a measure of activation for a given filter.

1) Convolutional layer operation: The operation in the convolutional layer is obtained by replacing $\mathbf{A}$ and $\mathbf{B}$ with the corresponding filter and feature map, respectively. Let the k-th feature map of the l-th layer be $\mathbf{F}_l^k \in \mathbb{R}^{w_l \times h_l}$, and the corresponding filter be $\mathbf{W}_{k'}^k \in \mathbb{R}^{w_l' \times h_l'}$ that convolutes with the k'-th feature map of the previous layer $\mathbf{F}_{l-1}^{k'}$, where w_l and h_l (w_l' and h_l') represent the width and the height of the feature map (of the filter) in layer l, respectively. Then, $\mathbf{F}_l^k$ is obtained by

$$\mathbf{F}_l^k(i,j) = \sigma\Big(\sum_{k'=1}^{d_{l-1}} \mathbf{W}_{k'}^k(i,j) * \mathbf{F}_{l-1}^{k'}(i,j) + b_k\Big), \qquad (3)$$

where d_{l-1} is the depth, i.e., the number feature maps, in the $l-1$ layer. Note that the activation function σ is included in (3), and leaky ReLU is used as a nonlinear activation function. By having multiple filters, CNN learns various combinations of features in each convolutional layer.

2) Reducing and expanding the size of feature map: Since the vector lies in reduced dimension, we need to consider how to reduce and expand the size of the feature map. For reducing operation in the encoder, we exploit the convolution with strides, where the size of the feature map shrinks by the values of strides. For expanding operation in the decoder, by considering the mirrored structure of CAE, we apply the transposed convolutional layers which restores filter values after enlarging the feature map according to the given strides. In addition, we match up the size of filter with strides to avoid the checker board artifacts induced by overlapping.

III. Convolutional Autoencoder Design

In this section, we compare various CAEs with different model complexity and select the final CAE structure in terms of the root means square error (RMSE) between the original and reconstructed data. After studying various model structures, we select two convolutional layers and two fully connected layers in the encoder and use a symmetrical structure in the decoder with tied weights as illustrated in Fig. 2. Note that the hatched box in Fig. 2 is the layer that outputs the encoded vector $\mathbf{z}$. In addition, row and column strides in convolutional layer are set to $(3, 2)$, which are the divisors of height and width of load image. We set the number of neurons in the middle layer of CAE as 100 so that one year data is compressed to roughly four day-long data, i.e., just 1% of the original dimension. The dimension of the encoded data, however, can vary according to user's choice (e.g, 10 days, 40 days), and there is a trade-off between the dimension, reconstruction error, and the size of CAE.

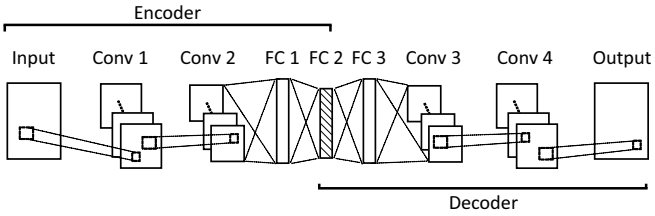


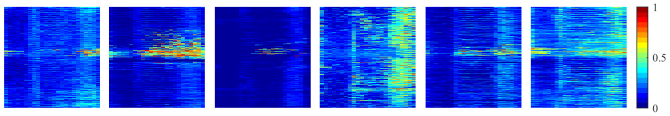
Fig. 2. The proposed structure of convolutional autoencoder. Conv 1, 2, FC 1, 2, 3 and Conv 3, 4 represent convolutional layers, fully connected layers and transposed convolutional layers, respectively.

After setting the basic structure as above, we compare RMSE of CAEs with varying numbers of weights. The configurations of the encoder part in CAE, model complexity (the number of weights and biases in the encoder), and RMSE of 1,405 households are shown in Table I. By modifying the numbers of filters and neurons in the second convolutional layer and the third fully connected layer, we have CAEs with different model complexities. Due to space limitation, only three of them are shown in Table I. The results show that RMSE decreases in the number of parameters, i.e., 0.072, 0.028, 0.016. The number of parameters increases roughly 10 times compared with the upper row, and the RMSE is inverse proportional to the model complexity. Fig. 3 shows the original load images of six randomly selected households and their reconstructions along with configurations in Table I. The horizontal and vertical axes of load image represent 24 hours and 360 days, respectively. As can be seen, the load image can be effectively reconstructed from the 100 dimensional encoded data. The reconstructed images in the second row are roughly similar to the original images whereas the images in Figs. 3 (c) and (d) are restored more accurately. Although the RMSE in Fig. 3 (d) is lower than the one in Fig. 3 (c), there is little visual difference.

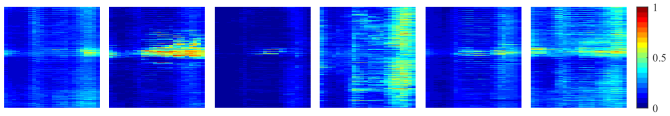
Based on the result of Table I and Fig. 3, the final structure used in the clustering process is selected to be B by considering the tradeoff between RMSE and model complexity. The number of filters are 128 and 64 in the two convolutional layers, and the number of neurons are 500 and 100 in two fully connected layers in the encoder. In training CAE, we use *Tensorflow* [13], and the weights are updated with Adam optimizer [14]. The final CAE is trained with 150 epochs,

TABLE I
COMPARING RMSES ACCORDING TO DIFFERENT CAE STRUCTURES AFTER TRAINING.

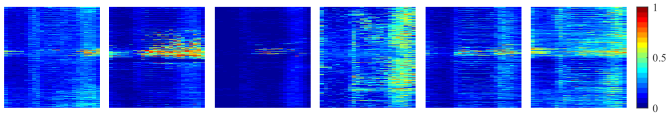
Subject	Layers					Model Complexity	Average Cost (RMSE)
	Input	Convolutional		Fully Connected			
		$1st$	$2nd$	$3rd$	$4th$		
	Input Size	Output Size (The Number of Filters @ Filter Size)					
Varying Model Complexity	360×24	$120 \times 12 \times 128$ (128 @ 3×2)	$40 \times 6 \times 16$ (16 @ 3×2)	125	100	505,925 (A)	0.072
			$40 \times 6 \times 64$ (64 @ 3×2)	500	100	7,780,712 (B)	0.028
			$40 \times 6 \times 256$ (256 @ 3×2)	1,000	100	61,738,860 (C)	0.016



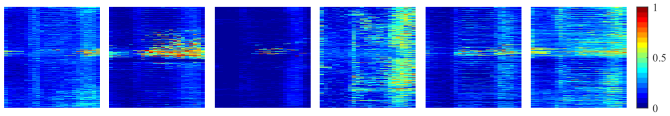
(a) The original load image.



(b) Reconstructed load image (16×125, RMSE = 0.072).



(c) Reconstructed load image (64×500, RMSE = 0.028). This is our choice.



(d) Reconstructed load image (256×1000, RMSE = 0.016).

Fig. 3. Comparison between the original load images and their reconstructions with different model parameters and RMSEs. The numbers in parenthesis are the number of filters and neurons in the second and the third layers.

which is also a knee point of RMSE vs. epoch plot. The RMSE of the final model is 0.02793. Since we use the normalized data for CAE, this RMSE can be considered as 2.79% loss of information in the reconstruction.

IV. CAE-BASED CLUSTERING AND EXPERIMENTS

A. CAE-based Clustering

After the encoding stage, we cluster the encoded data $\{\mathbf{z}_n \in \mathbb{R}^{100}, n = 1, \ldots, N\}$ by K-means clustering, which assigns N observations into K clusters to minimize the sum of squared Euclidean distances to centroid of assigned cluster. Given $\mathbf{z}_n, n = 1, \ldots, N$, and K, the K-means clustering is an NP-hard optimization problem, and iterative algorithms (e.g., Lloyd [15], and Hartigan-Wong algorithms [16]) are commonly employed.

Two important parts of clustering are how to measure the goodness of clustering results and how to determine the proper number of clusters K. Load profile clustering is an unsupervised learning problem, and there are no explicit classes of electricity customers. In principle, clustering should maximize the inter-cluster distances and minimize the intra-cluster distances. To quantify the goodness of clustering, we exploit four popular clustering validation indices (CVIs) [17]: Dunn index, Calinski-Harabasz index, Davies-Bouldin index, and SD index. Note that different validation indices come from different definitions of inter and intra-cluster distances. In the selection of K, we calculate each validation index with varying K and choose the proper K according to its selection rule. One may use other clustering techniques such as partitioning around medoids, hierarchical clustering, or self-organizing map within our framework.

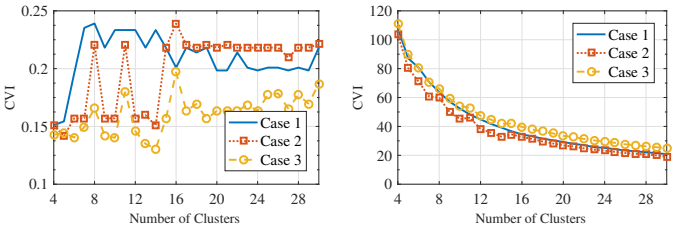
TABLE II
CALCULATION OF CVI IN DIFFERENT CASES.

Case	Clustering	Validation
1	original space	original space
2	encoded feature space	original space
3	encoded feature space	encoded feature space

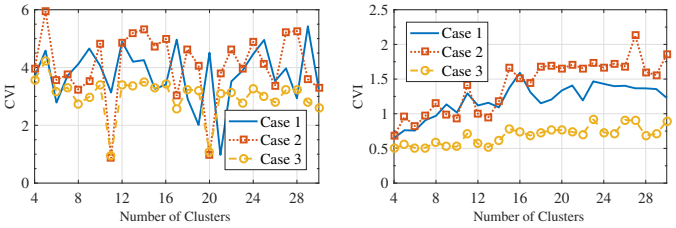
B. Verification of CAE-based Clustering

After the clustering stage, validation needs to be performed to verify the effectiveness of the clustering in the encoded space by investigating CVIs. We compare the CVIs of the proposed clustering in the *encoded* space $\mathbb{R}^{100}$ with the one in the *original* space $\mathbb{R}^{8,640}$. As clustering can be performed in both spaces, the calculation of CVIs and the selection of K can also be determined in both spaces. Hence, we compare CVIs for three different cases. As shown in Table II, both clustering and validation are performed in $\mathbb{R}^{8,640}$ for Case 1. In Case 2, customers are clustered in $\mathbb{R}^{100}$, but CVIs are calculated in $\mathbb{R}^{8,640}$. In Case 3, both clustering and validations are performed in $\mathbb{R}^{100}$.

If the encoding and clustering are well performed with the proposed framework, we can expect (a) the *raw* values of CVIs of Cases 1 and 2 to be similar, and (b) the *relative* variational patterns in K are similar for Cases 2 and 3, considering they are measured in different spaces. The former and the latter support the validity of clustering in the encoded space and the compatibility of hyperparameter selection in the encoded space, respectively.



(a) Dunn (max). (b) Calinski-Harabasz (max).



(c) Davies-Bouldin (min). (d) SD (min).

Fig. 4. Four CVIs and the selection of the number of clusters K. The parenthesis (max or min) indicates the selection rule for good clustering.

TABLE III
VERIFICATION OF CAE-BASED CLUSTERING

Value	Dunn	Calinski-Harabasz	Davies-Bouldin	SD
$R_{1,2}$	0.25	0.04	0.31	0.33
$R_{2,3}$	0.76	0.94	0.92	0.81

Fig. 4 shows the results of those three cases with different CVIs. As shown in Figs. 4 (a)-(d), the CVI curves of Cases 1 and 2 fluctuate closely irrespective of the indices, and the variational patterns of Cases 2 and 3 in K are similar to each other, i.e., when Case 2 has a peak shape at certain K, CVI of Case 3 also has a peak shape, and vice versa.

To quantify the aforementioned characteristics, we use the following metrics, and the results are shown in Table III,

$$R_{1,2} = \frac{\frac{1}{27}\sum_{k=4}^{30} |v_k^1 - v_k^2|}{\max(\mathbf{v}^1) - \min(\mathbf{v}^1)}, \tag{4}$$

$$R_{2,3} = corr(\Delta\mathbf{v}^2, \Delta\mathbf{v}^3), \tag{5}$$

where $\mathbf{v}^j = \{v_k^j\}$, $\Delta\mathbf{v}^j = \{v_{k+1}^j - v_k^j\}$, and v_k^j corresponds to the CVI of Case j with the number of clusters k.

To support the validity and compatibility of the proposed clustering framework, low values of $R_{1,2}$ (less than 1) is desired, whereas $R_{2,3}$ which is the Pearson's correlation coefficient between $\Delta\mathbf{v}^2$ and $\Delta\mathbf{v}^3$ should be close to 1 , and both can be easily seen in Table III. According to these observations, CAE-based clustering in the encoded space can be a good substitute for the clustering in the original high dimensional space.

C. Computational Advantages of CAE-based Clustering

Clustering in the encoded space has substantial advantage in computational time. If the dimension of data d and the number of clusters K are fixed, the optimal solution of K-means clustering can be obtained in $O(N^{dK+1})$ [18]. For iterative algorithm, the running time of Lloyd algorithm is $O(NKdi)$, where i is the number of iterations [19]. Recall that in our case $N = 1{,}405$ and $d = 8{,}640$ in the original space. On the other hand, the dimension of the encoded data reduces to $d' = 100$, and thus the computational time with the same N and K decreases dramatically. In our experiment with Intel(R) Core(TM) i7-3770 CPU @ 3.40 GHz, the computational time of K-means clustering with $K = 11$ in the original space takes 27.6 seconds for Lloyd algorithm and 15.8 seconds for Hartigan-Wong algorithm on average of five attempts. On the other hand, clustering in the encoded space takes millisecond order; only 0.06 and 0.03 second, respectively, for each algorithm. Therefore, the empirical clustering speed in the encoded space is 500 times faster than in the original space while preserving the clustering performance In addition to the advantage in computational time, clustering in the proposed framework occupies smaller memory than traditional clustering due to the reduced dimension. Thus, it is possible to cluster a large number of households with limited memory size, which makes our work suitable for big data applications.

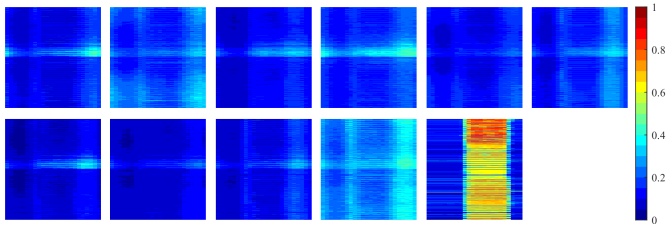
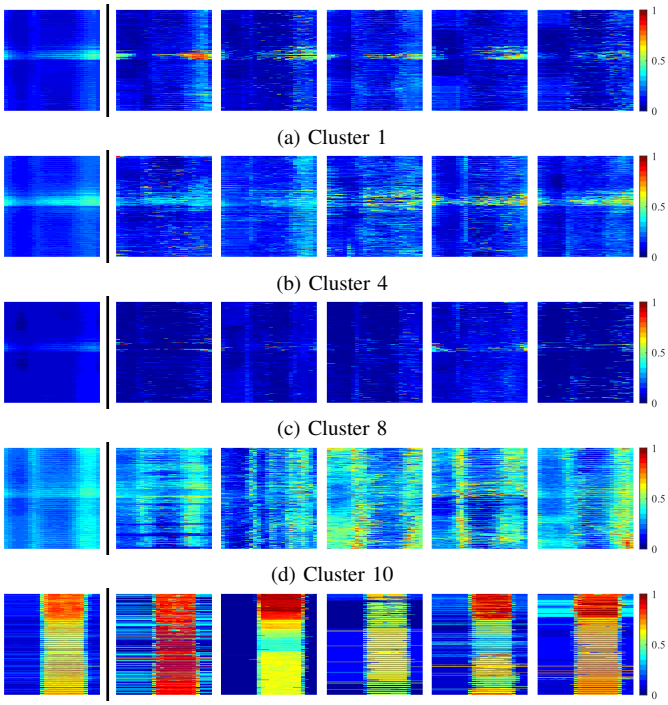


Fig. 5. Load images of cluster centers (the average of cluster members) from Cluster 1 (top-left) to Cluster 11 (bottom-right).

D. Observations from CAE-based Clustering

For the detailed analysis of load profiles, we examine the clustering results with properly selected K based on CVIs of Case 3. According to the selection rules, K is set to 11 because the minimum Davies-Bouldin index value and high Dunn index value occurs at $K = 11$. Fig. 5 visualizes the load images of each cluster center. Overall, there are two bright vertical lines and one horizontal line. For example, vertical lines in Cluster 10 are noticeable; the thin line on the left and the broad line on the right imply the morning peak before going to work and the evening peak after work, respectively, which is consistent with the well-known facts of two peaks in a day. The horizontal line clearly appears in Cluster 4 while in other clusters such as Clusters 1, 3 and 7, only a narrow segment during evening is observed. For an exceptional case of Cluster 11, there is only one thick vertical line, which implies that households in Cluster 11 use their electricity constantly from late morning to evening.

Fig. 6 illustrates five distinctive clusters and their members. As can be seen, the samples in the same cluster have similar patterns. For example, members in Clusters 1 and 4 seem to use more electricity in summer. The difference between two clusters is daily pattern in summer; in the case of Cluster 1, the electricity consumption is concentrated during night whereas



(a) Cluster 1

(b) Cluster 4

(c) Cluster 8

(d) Cluster 10

(e) Cluster 11

Fig. 6. Illustration of the centers and samples of distinctive clusters. The first column illustrates cluster centers and the remaining columns are random samples in each cluster.

electricity consumption in daytime is also high in Cluster 4. In Cluster 8, the overall color tone of samples is the same with dark blue, indicating small variations of electricity usage throughout the day and year. Seasonal variations of Clusters 10 and 11 are indistinguishable, but the characteristic of daily pattern is distinctive.

V. Conclusion

This paper proposed a novel framework of load profile clustering based on deep convolutional autoencoder. By utilizing CAE, the yearly load data in a high dimensional space ($\mathbb{R}^{8,640}$) can be effectively encoded to a vector in $\mathbb{R}^{100}$, i.e., 1–3% of information loss and 1.15% of original dimension. Next, encoded load profiles are clustered by K-means clustering. The effectiveness of proposed clustering is verified by four CVIs, which confirm that clustering and its validation can *both* be performed in the encoded space. The advantages of CAE based clustering are two-fold. First, the computational time of clustering is reduced by three orders of magnitude, e.g., 500 times, when we applied Lloyd algorithm and Hartigan-Wong algorithm. Second, the reduced dimension also contributes to saving memory space during clustering by approximately 100 times. Thus the proposed framework is suitable smart meters data analytics in smart grid applications.

The proposed clustering framework can be further extended in several directions. We are currently working on comparing the proposed CAE to other dimension reduction methods, e.g., principle component analysis and singular value decomposition. In addition, additional research is required on the applications of the proposed CAE based clustering. For example, CAE can be used as a prototype or pre-trained network of CNN applications such as forecasting models. Tariff offers can also be designed based on the proposed framework.

Acknowledgment

This work was supported by the Korea Institute of Energy Technology Evaluation and Planning (KETEP) and the Ministry of Trade, industry & Energy (MOTIE) of Korea (20161210200410) and by the Weather See-at Technology Development Program of Korea Meteorological Institute (KIMPA 2015-4070).

References

[1] "How many smart meters are installed in the United States, and who has them?" [Online]. Available: https://www.eia.gov/tools/faqs/faq.php?id=108&t=3

[2] S. Zhou and M. A. Brown, "Smart meter deployment in Europe: A comparative case study on the impacts of national policy schemes," *Journal of Cleaner Production*, vol. 144, pp. 22–32, Feb. 2017.

[3] G. Chicco, "Overview and performance assessment of the clustering methods for electrical load pattern grouping," *Energy*, vol. 42, no. 1, pp. 68–80, Jun. 2012.

[4] J. Kwac, J. Flora, and R. Rajagopal, "Household energy consumption segmentation using hourly data," *IEEE Trans. on Smart Grid*, vol. 5, no. 1, pp. 420–430, Jan. 2014.

[5] J. D. Rhodes, W. J. Cole, C. R. Upshaw, T. F. Edgar, and M. E. Webber, "Clustering analysis of residential electricity demand profiles," *Applied Energy*, vol. 135, pp. 461–471, Dec. 2014.

[6] K. Mets, F. Depuydt, and C. Develder, "Two-stage load pattern clustering using fast wavelet transformation," *IEEE Trans. on Smart Grid*, vol. 7, no. 5, pp. 2250–2259, Sep. 2016.

[7] R. Li, F. Li, and N. D. Smith, "Multi-resolution load profile clustering for smart metering data," *IEEE Trans. on Power Systems*, vol. 31, no. 6, pp. 4473–4482, Nov. 2016.

[8] T. Räsänen, D. Voukantsis, H. Niska, K. Karatzas, and M. Kolehmainen, "Data-based method for creating electricity use load profiles using large amount of customer-specific hourly measured electricity use data," *Applied Energy*, vol. 87, no. 11, pp. 3538–3545, Nov. 2010.

[9] S. Ryu, J. Noh, and H. Kim, "Deep neural network based demand side short term load forecasting," *Energies*, vol. 10, no. 1, pp. 1–20, Jan. 2017.

[10] L. Wang, Z. Zhang, J. Xu, and R. Liu, "Wind turbine blade breakage monitoring with deep autoencoders," *IEEE Trans. on Smart Grid*, vol. 9, no. 4, pp. 2824–2833, Jul. 2016.

[11] E. D. Varga, S. F. Beretka, C. Noce, and G. Sapienza, "Robust real-time load profile encoding and classification framework for efficient power systems operation," *IEEE Trans. on Power Systems*, vol. 30, no. 4, pp. 1897–1904, Jul. 2015.

[12] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[13] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning." in *Proc. of USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, Nov. 2016.

[14] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. of Int'l Conf. on Learning Representations (ICLR)*, May. 2015.

[15] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. on Information Theory*, vol. 28, no. 2, pp. 129–137, Mar. 1982.

[16] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A K-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[17] C. C. Aggarwal and C. K. Reddy, *Data Clustering: Algorithms and Applications*. CRC Press, 2013.

[18] M. Inaba, N. Katoh, and H. Imai, "Applications of weighted Voronoi diagrams and randomization to variance-based K-clustering," in *Proc. of ACM Symposium on Computational Geometry*, Jun. 1994.

[19] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.