IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON EVOLVING TECHNOLOGIES IN ENERGY STORAGE SYSTEMS FOR ENERGY SYSTEMS APPLICATIONS

Received September 29, 2020, accepted October 25, 2020, date of publication November 3, 2020,
date of current version November 12, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3035421

Cooperating Edge Cloud-Based Hybrid Online Learning for Accelerated Energy Data Stream Processing in Load Forecasting

CHANGHA LEE, SEONG-HWAN KIM, (Associate Member, IEEE), AND CHAN-HYUN YOUN, (Senior Member, IEEE)
School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon 34141, South Korea

Corresponding author: Chan-Hyun Youn (chyoun@kaist.ac.kr)

This work was supported in part by the Korea Electric Power Corporation under Grant R18XA05, and in part by the Electronics and Communications Research Institute (ETRI) & National Research Council of Science and Technology (NST) grant funded by the Korea Government [Ministry of Science and ICT (MSIT)] (Research on Foundation Technology of IDX Platform) under Grant 19ZS1200.

ABSTRACT The data analysis platform used in smart grid is important to provide more accurate data validation and advanced power services. Recently, the researches based on deep neural network have been increasing in data analytic platforms to address various problems using artificial intelligence. The main problem to analyze multiple meter data based on deep learning is that the data distribution is varying according to both different client and time flow. Some studies, such as continual learning, are effective in dynamically fluctuating data distributions, but require additional complex computational procedures that make it difficult to construct an online learning system for processing data streams. In this paper, we proposed a hybrid deep learning scheduling algorithm to improve accuracy and accelerate learning performance in a multiple smart meter source environment, of which biased data feature varies dynamically. We use a simple analysis method, cosine similarity, to reduce computation complexity. By analyzing the frequency distribution of cosine similarity, a model recognizes that biased data feature of power consumption patterns. The skewed data distribution is reduced by using the zero skewness property of of an uniform distribution. The diversity of memory buffer was increased by update strategy which maximizes variance of pattern. When scheduling an online and offline gradient in different computational complexity, the proposed model reduces processing time by selectively calculating gradient considering the degree of data feature transition. To verify the performance of the proposed algorithm, we conducted three experiments with AMI stream data on the proposed method and the existing method of online learning. The experimental results demonstrate that our method can achieve reasonable performance in terms of trade-off between accuracy and processing time.

INDEX TERMS Deep learning scheduling, online learning, load forecasting, energy data stream processing.

I. INTRODUCTION

With the rapid growth of global energy demand and the emergence of new technologies, Advanced Metering Infrastructure (AMI) consisting of automatic meters, bi-directional communication and a data repository, have been developed for energy-efficiency and bi-directional resource management in worldwide. In actual case, Korea Electric Power Corporation (KEPCO) has been supplying AMI since 2013 and installed 7.4 million AMI in 2018 [1]. KEPCO is now planning to install approximately 22 million AMI units in 2020. Various information containing time series power data collected from smart meters is transmitted to the data center of power corporation at every 15 minute or one hour. To validate, estimate and edit the enormous data stream, the power utility company builds the Meter Data Management System (MDMS) which refers to software of data center that performs data storage and management to import, validate, edit enormous quantities of data for billing or data analysis.

To implement data analysis based on deep learning or learning-based model in data stream such as a AMI meter

The associate editor coordinating the review of this manuscript and approving it for publication was Eklas Hossain.

data, it is required to keep training on continuously incoming data stream according to [2], [3]. The time-variant data stream is different from the general big data or structured data. In current studies on both machine learning and deep learning, the learning-based model which is trained by historical data shows performance degradation on future time-series data [4] due to non-stationary feature, particularly named as a concept drift. The concept drift [5] means that the statistical properties of the target variable, which the model is trying to predict, change over time in unforeseen ways. This causes problems because the predictions become inaccurate as time passes. Reminding the necessity of incorporating new information from a data stream, Incremental Learning or Online Learning [6], [7] is a common method to train a learning-based model expanding the knowledge of existing model by incoming input data containing untrained feature. When incrementally learning model with a small batch size, the incremental learning scheme proposed how to incrementally train without loss of model performance by using incremental learning algorithm-based dynamic learning strategy and mathematical formula.

When multiple meter data of each customer is submitted to meter data management system, most of researches about deep learning system have studied multi-agent model for multi-client data according to [8]. This system is accurate, however it requires more resources than single agent. Therefore, considering resources and hardware limitations, it is obvious that the single model is more adequate than the multi-agent for the online learning system. However, there are the problems when operating single model based online learning on the environment of multi-AMI and big data stream. The main reason of low performance of incremental learning is biased data distribution problem according to each customer. This is also named as non-iid data in [9]. In real case, when multi-AMI is continuously incoming, the AMI meter data distribution representing consumption pattern is dynamically changed due to totally different consumption pattern among multiple AMI users. Considering multiple AMI users, incoming stream data can have skewed and dramatically different distribution. For a set of streams separated by multiple AMI users of different colors, the consumption pattern frequency distribution, measured using cosine similarity, has a different biased distribution when comparing initial pattern.

To apply online learning for multi user data stream in physical system, efficient resource and learning scheduling are basically required. Therefore, this paper focuses on developing hybrid deep learning scheduling solution for feasible online learning system.

The main contributions of this paper are as follows:

- We proposed a hybrid deep learning scheduling algorithm to enhance and accelerate learning performance in a multi-AMI ID environment where the distribution of biased data varies dynamically.
- We first analyze the basis vector to establish frequency distribution based on cosine similarity for recognizing concept drift. We focus on a general feature of power consumption data for AMI data feature analysis.
- We then consider the accuracy performance of deep learning model using the variety of historical data to maintain generality of centralized model on cluster side.
- According to scheduling algorithm, we design the learning system for a given edge-cloud computing. We present the system model in multi-client AMI data stream environment.
- To demonstrate that our approach can achieve objective performance, we implement valid experiment. Then, we discuss results in terms of performance trade-offs among related algorithms.

The remaining section of this paper is as follows. We describe useful conventional methods corresponding challenges in Section II. Then, the proposed system model is described in edge-cloud environment in Section III. After designing system model, we focus on the proposed hybrid deep learning scheduling scheme with detailed derivations in Section IV. Performance evaluation in Section V is dedicated to show the performance of proposed system and discuss the trade-offs. Finally, Section VI summarizes the paper with conclusions and future works.

II. PROBLEM DESCRIPTION FOR EFFICIENT LOAD FORECASTING

In this section, we describe the existing online learning algorithm of previous research which optimizes the learning-based model in continuously incoming data stream. We describe both a simple analysis method based on cosine similarity and method for representing discrete data distribution. Then, we also describe the multi-AMI meter data characteristics, in detail, considering the dynamically changing data distribution as time flows. Then, we address the limitation of existing online learning algorithm, when applied to the real-world data stream.

A. ONLINE DEEP LEARNING FOR STREAM DATA ANALYSIS

The data analysis based on deep learning is focusing on generalize the model to represent data distribution including observed data and unobserved data having similar characteristics of the observed examples. However, there is still the problem in non-stationary environment because the unobserved data has no relation to observed data. In this environment, data stream analysis aims to extract principle feature or original characteristics from a continuous or discrete stream of information to infer the classes for the accurate classifier, or the predictive value or missing value, which is continuous stream, for the predictor or classifier.

Incremental learning is an ordinary learning scheme [7], [10] to keep train models using incoming data stream to extend the generalized information of a current learning-based model, as shown in Figure 1. When continuously learning adaptive model in a sequential data, the problem how to constantly train a learning-based model without

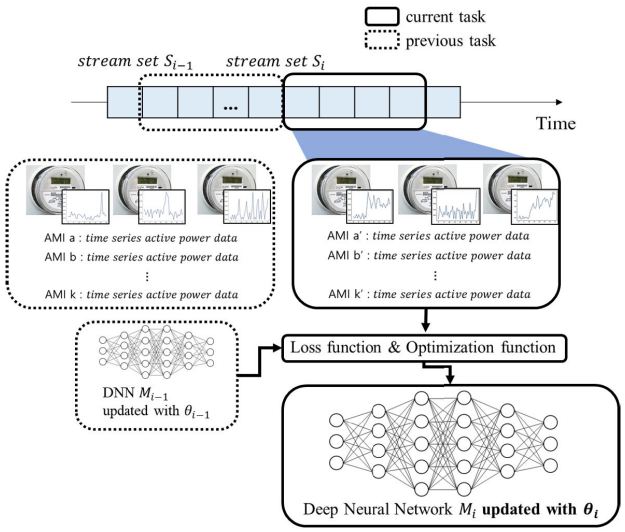


FIGURE 1. An illustration of existing online learning system [7], [10] when applying it on multiple AMI meter data stream.

loss of model performance on previous information is solved with incrementally learning algorithm-based dynamic learning technology and mathematical formula. The main purpose of incremental learning is that the new information is trained to existing model without forgetting existing knowledge over time. However, the existing model still show low performance in rapidly varying data distribution in real-world environment.

To handle dynamically changing data characteristics, continual learning methods store small amounts of data at the learning stage in limited memory that do not affect the system, preserving the previous information only by incorporating the stored memory data into the learning as if it were a human being in subsequent training. On limited memory space, the experience replay algorithms [11], [12] considering the training strategy for the conditions under which the number of classes of images is increasing, and the gradient regularization algorithms [9], [13] was proposed to have a positive effect on the model in which the previous data is currently updated. In other ways, it continues to learn the data generated by the productive model learned from the previous information without separate memory space, and it also learns the productive model simultaneously to preserve the information in the current work [14]. However, continual learning scheme requires additional complex computation with reserved memory buffer.

B. DATA ANALYSIS BASED ON COSINE SIMILARITY FOR ELECTRICITY

To measure the similarity among power consumption patterns, the cosine similarity score is commonly used with simple calculation between two vector indicating power consumption pattern. The cosine similarity, which is used as a score to measure the similarity of the pattern [15], is also used as a similarity indicator for the AMI meter data [16], [17]. Therefore, the cosine similarity score between two power consumption vector is defined as their dot product divided by the product of their L2 norm, i.e. $\frac{X \cdot Y}{||X|| \cdot ||Y||}$ as referred in [18].

C. FREQUENCY DISTRIBUTION TO EXPRESS DISCRETE DATA STREAM DISTRIBUTION

According to reference [19], counting how many similar data fall into each interval, the similarity histogram $D_{S_i} = \{D_1, D_2, \ldots, D_j, \ldots, D_n\}$ could express data distribution with total count $|D_j|$ and each histogram bins D_j is defined as follows:

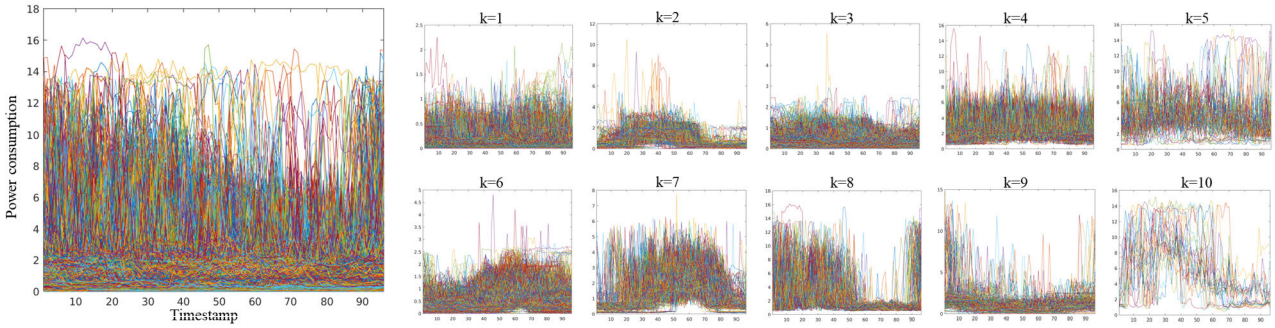
$$D_j = \{x^p_{i,k} | x^p_{i,k} \in C_i\} \tag{1}$$

where stream set S_i with time sequence i, similar data set C_i based on K-means clustering. The referred journal paper [19] emphasized that histogram definition, despite of simplicity, covers all binned density estimators. Moreover, according to [20], a histogram or a data frequency distribution can also provide precise representations of the underlying true data distribution. The reference denotes that the similar data is searched by K-means clustering, however, K-means is iterative method and NP-hard problem which causes the large processing time.

D. ONLINE LEARNING IN MULTI-CLIENT AMI DATA

We conducted a preliminary analysis to confirm the effect of the distribution of the transmitted data actually changed dynamically in the context of the incremental multiple AMI meters. To investigate how diverse the pattern of daily power consumption in low-voltage was, the data sets were classified using k-means as shown in Figure 2. The x-axis indicates power consumption and the y-axis represents timestamp in both (a) and (b). The subfigure (a) shows the entire description of dataset, and (b) presents the k-means clustering results when k is set to 10. Focusing on the clustering results, the data is not precisely separated however this result is a rebuttal of the varying distribution of data. We then applied incremental learning techniques to these real-world datasets, with data streams transmitted to the data center sequentially and with the condition of only-once observation consistent with [9]. Figure 3 shows test results with average root mean square error (ARMSE) of existing online learning after training each stream set. We note that despite applying incremental learning techniques, the error is not reduced.

In order to resolve these problems of previous work, we propose a hybrid online learning scheduling using cosine similarity to utilize edge and cloud computing. Utilizing cosine similarity and frequency distribution based on it, the proposed approach estimates the underlying characteristics of incoming data stream and handles offline experience buffer. Based on it, leveraging recognized data distribution, the proposed method allocates gradient computation task to edge and cloud computing, whereas the others only concern centralized computing resources. Below we describe our proposed approach in details.



(a) The description of dataset.

(b) K-means clustering results when k is 10.

FIGURE 2. The description of Kepco dataset for daily power consumption, and k-means result when k is 10. A line in the each figure shows a daily low-voltage power consumption pattern. The results indicate that the data is hard to be clustered.

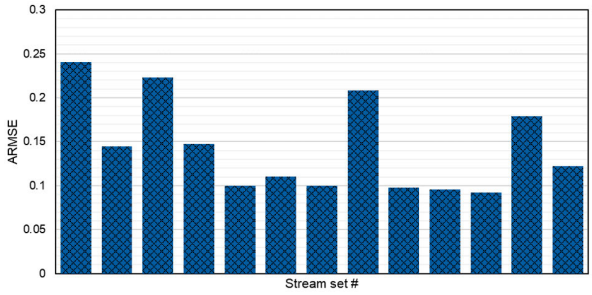


FIGURE 3. The fluctuating accuracy problem on dynamically varying data distribution in existing online learning [7], [10].

III. SYSTEM MODEL IN EDGE-CLOUD ENVIRONMENT

Assuming a power system that transmits large-scale AMI time series power consumption data over the network, the data stream is preprocessed from edge servers to enable deep running learning processing. The pre-processed training data stream has customer data randomly mixed, but the time order is sequentially arranged. The proposed edge-cloud system transmits and receives the required data streams over the network and in-memory, but does not store data in the file system as described in Figure 4. The proposed system in which edge and cloud cooperate trains the neural network for continually pre-processed data stream by optimizing model parameter to reduce loss.

A. PREPROCESSOR IN EDGE

First, the raw data stream is received in edge server which implements preprocessing data as described in Figure 4. The time-series power data is denoted as a real power $w_m(t)$ where t is a given time and m is a smart meter identifier. To train a neural network, the data of each smart meter is collected into queue in fixed small size p which is enough to learn spatial and temporal data characteristic. Therefore, the small data chunk of each smart meter is expressed by $x_{i,m}$ following:

$$x_{i,m}(t) = \{w_m(t), w_m(t+1), \ldots w_m(t+p)\} \tag{2}$$

where i is a time sequence. This small data chunk is a trainable atomic input unit and size p is directly related to deep learning model. Additionally, The predictive target is given by $y_{i,m}$. In proposed system, $y_{i,m}$ is same as $w_m(t+p+1)$ because our target application is short-term power consumption prediction. The training dataset T_i which proceeds feed-forward in neural network at time sequence i consists of N pairs of data $(x_{i,m}, y_{i,m})$. The N value is small enough to train model in real-time. The preprocessor forwards training dataset to both edge and cloud server as described in Figure 4.

B. INTER-NODE TRANSCEIVER

In proposed system, edge and cloud server need to communicate with each other during training procedure. The inter-node transceiver follows binary protocols for gradient data serialization and de-serialization to deliver data through interconnected network. We set a transceiver model function $Comm(g, Src)$ where g is a gradient, Src is a gradient transmitter location $Edge$ or $Cloud$. When call this function, the gradient values are synchronized according to a gradient g calculated from Src. We describe the data flow for gradient information by blue line in Figure 4. Therefore, the edge and cloud computing can implement parallel processing before communication procedure.

C. OFFLINE EXPERIENCE BUFFER

In cloud node, a historical data is stored in a limited sized memory to enhance the accuracy performance of deep neural network in multi-AMI data stream environment. This memory named as the offline experience buffer B_i at time sequence i has limited size $n \times p$ where n is a number of preserved vectors, and p is a same value of Equation (2).

D. COSINE SIMILARITY WITH CRITERION VECTOR AND FREQUENCY DISTRIBUTION

We particularly define the cosine similarity score between AMI meter data with criterion vector $\vec{v}$ to resolve similarity well as follows:

$$score_{\vec{v}}(x^p) = cos(\theta_{\vec{v},x^p}) = \frac{\vec{v} \cdot x^p}{||\vec{v}|| \cdot ||x^p||} \tag{3}$$

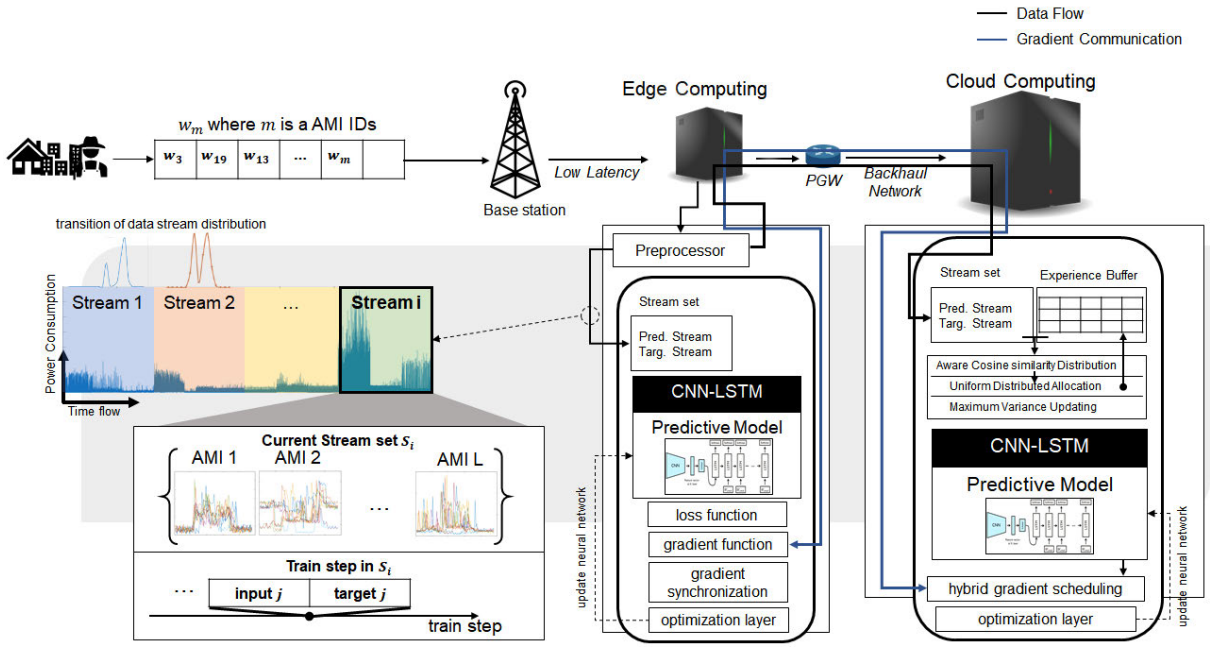


FIGURE 4. An illustration of the edge-cloud environment for the proposed system. The black line shows data flow and the blue line means bidirectional gradient communication path. The preprocessed data stream is described in the bottom-left of the illustration.

In terms of physical meaning, we used $\vec{1}$ vector as a criterion vector in a p-dimensional space to express this evenly because all of electricity consumption is positive. We will discuss the other criterion vector later with the results of the experiment in Section V.

In addition, the cosine similarity frequency distribution is defined for measuring distribution plots based on cosine similarity. Considering the width ϵ of cosine similarity and faster processing histogram than reference [19] using K-means, we define the cosine similarity frequency distribution function $D(i, \epsilon) = \{D_1, D_2, \ldots, D_n\}$ where time sequence i, cosine similarity score set C_i $c_{min} = \min C_i$, $c_{max} = \max(C_i)$, and range width $\epsilon = c_{max} - c_{min}$. With function $D(i, \epsilon)$, the cosine similarity histogram buffer is defined as illustrated in Equation (4) and Figure 5:

$$D_j = \{x^p_{i,k} | c_{min} + (j-1) * \frac{\epsilon}{n} \leq c_k \leq c_{min} + j * \frac{\epsilon}{n}\} \quad (4)$$

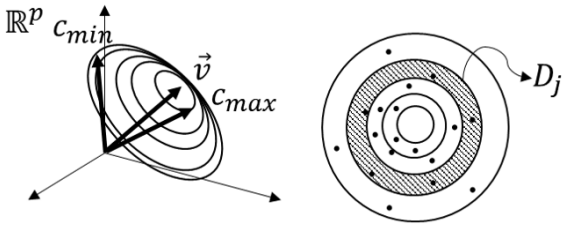


FIGURE 5. The mathematical description of criterion vector $\vec{v}$ with c_{max} and c_{min} in Equation (3) and cosine similarity histogram buffer D_j with inputs $x^p_{i,k}$ expressed in dots in Equation (4).

IV. HYBRID DEEP LEARNING SCHEDULING SCHEME

The main objective of the proposed system that the scheduler uses incoming stream set and experience buffer to perform projected gradient calculations that require complex computations through cloud computing to recognize the data distribution in phase 1. Based on this, there is phase 2 to update the offline experience buffer with a set selection with a high variance to create an uniform distribution set to address the skewed distribution problem. Finally, it consists of phase 3, which reduces processing time through a scheduling method based on a different distribution chart.

To train deep learning model with high accuracy on both previous and current data feature, a proposed scheduler updates current deep learning model using online gradient with low latency on edge computing when skewed distribution problem is not occurred. In other words, a large-scale time-series power consumption data is transmitted to edge server first and update deep learning model using current gradient to reflect current feature as fast as possible. Furthermore, when skewed distribution transition is happened, proposed scheduler processes three phases to calculate projected gradient requiring high and complex computation using cloud computing, such as MDMS.

A. PHASE 1 : RECOGNIZING COSINE SIMILARITY FREQUENCY DISTRIBUTION

First, Phase-1 recognizes the cosine similarity score with criterion vector to express the degree of similarity among the power consumption because cosine similarity has a linear time complexity. Following cosine similarity score with criterion vector in Euqation (3), the criterion vector is $\vec{1}$

representing same distance from dimensional axis and well describes the power consumption pattern of time-series AMI meter data. After calculating cosine similarity score, a scheduler builds the cosine similarity frequency distribution D using uniform sampling to signify the number of similar consumption pattern according to cosine similarity score. To recognize the cosine similarity distribution in both stream set S_i and offline buffer B_{i-1} the scheduler concatenates both stream data and build cosine similarity set C_i with criterion vector $\vec{1}$ as follows:

$$C_i = \{c_k | c_k = score_{\vec{1}}(x^p_{i,k})\} \tag{5}$$

where $x^p_{i,k} \in S_i \cup B_{i-1}$. As can be seen in Figure 6, considering continuous domain of cosine similarity score, the scheduler build histogram buffer D_i using Equation (4), $D(i, \epsilon)$ where $c_{max} = \max(C_i)$, $c_{min} = \min(C_i)$, and $\epsilon = c_{max} - c_{min}$ as follows:

$$D_j = \{x^p_{i,k} | c_{min} + (j-1) * \frac{\epsilon}{n} \leq c_k \leq c_{min} + j * \frac{\epsilon}{n}\} \tag{6}$$

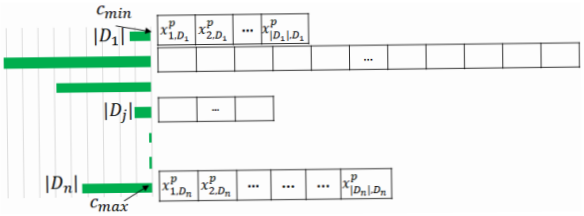


FIGURE 6. An illustration of histogram buffer D shown as a green bar according to Equation (6). Each green bar length indicates the total number of power consumption elements.

Figure 6 describes a bucket including similar power consumption pattern x^p_{i,D_i} and indicates the how many similar power consumption pattern. According to Equation (6), the first subset includes a power consumption pattern with cosine similarity c_{min} and the last subset includes a power consumption pattern with cosine similarity c_{max}. According to the histogram buffer D_i, the total element number of histogram buffer, denoted as $|D_i|$, is a range of biased consumption pattern in a frequency distribution based on cosine similarity score.

B. PHASE 2 : UPDATING OFFLINE EXPERIENCE BUFFER

Second, Phase-2 decides to update the offline experience-buffer using maximizing variance of a subset of the AMI meter data for preserving various consumption patterns. First of all, to reduce the skewness of cosine similarity frequency distribution, a heuristic method which does not increase a computational load is proposed with uniform distribution feature of which skewness is zero. Therefore, using the skewness of uniform distribution equals 0, a power consumption pattern x^p_j within subset D_j is selected to make uniform distributed set $\mathcal{U}$ where x^p_j is randomly selected in uniform probability distribution with D_j for diversity as follows in Equation (7):

$$\mathcal{U} = \{x^p_1, x^p_2, \ldots, x^p_j, \ldots, x^p_n\} \tag{7}$$

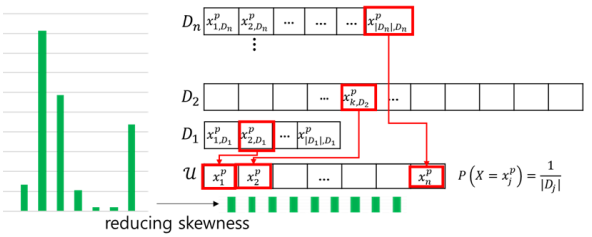


FIGURE 7. A description of the uniform distributed set $\mathcal{U}$ based on Equation (7). The red box means a randomly selected power consumption pattern x^p_{k,D_j} from a histogram buffer D_j.

where $P(X = x^p_j)$ is $\frac{1}{|D_j|}$. The random sampling from a histogram buffer D_j uses the index k of power consumption pattern x^p_{k,D_j} where $1 \leq k \leq |D_j|$. Due to using discrete frequency domain, a histogram buffer could not include any power consumption pattern. Therefore, if there is no power consumption pattern in some histogram buffer, then the proposed scheduler neglect empty histogram buffer because it means that there is no need to consider the power consumption pattern corresponding cosine similarity score scope. In Figure 7, the red box shows selected power consumption pattern with random probability. Therefore, the green bar denoting total count of power consumption pattern with similar cosine similarity score shows transformation from skewed distribution to uniform distribution without skewness. Because of the uniformly distributed cosine similarity score of power consumption pattern in reconstructed set, the uniform distributed set is named to reconstructed set $\mathcal{U}$.

Now, the scheduler determine whether to update using update policy maximizing variance for various representative pattern of preserved buffer. Referred to bias-variance analysis in previous paper [21], when a variance of the ensemble is large, the data feature diversity of the ensemble is more guaranteed than low variance. Therefore, we decide to update when variance of $\mathcal{U}$ is larger than B_{i-1} with buffer transition indicator τ like bellow:

$$B_i = \underset{x \in \{B_{i-1}, \mathcal{U}\}}{\arg\max} \, (VAR(C_x)) \tag{8}$$

where $VAR(\cdot)$ means a function calculating variance from a set. In Equation (8), the cosine similarity set C_x indicates that the cosine similarity score of x including both previous experience buffer B_{i-1} and $\mathcal{U}$ as an argument. Therefore, new updated buffer is assigned by buffer or set having larger variance than other.

Furthermore, when the updated experience buffer B_i is different from previous experience buffer B_{i-1}, it means that the power consumption pattern is more divergent than previous. Hence the scheduler designates the indicator τ of variance transition as follows in Equation (9):

$$\tau = \begin{cases} 0 & \text{if } B_i = B_{i-1} \\ 1 & \text{otherwise} \end{cases} \tag{9}$$

The hybrid deep learning scheduler is a maximum variance selective for various data representation.

C. PHASE 3 : REFLECTING OFFLINE INFORMATION WITH HYBRID SCHEDULING

To update a deep learning model, a gradient should be calculated as a vector reducing the predictive error on current target data. However, as mentioned at Section II, the projected gradient requiring additional computation is necessary to diminish the negative effect of skewed data distribution transition. Therefore, Phase-3 calculates both current and projected gradient using online and offline gradient. A deep learning learner projects the offline gradient to current gradient to reflect previous information of preserved data distribution. Using phase-2 transition indicator τ, the calculated gradient is scheduled to eliminate the additional computation for projected gradient.

To reflect current information into updated deep learning model M_i, the proposed system calculates online gradient g_{on} using edge computing as follows:

$$g_{on} = \frac{\partial l(M_{i-1}(S_i), t_i)}{\partial \theta_{i-1}} \tag{10}$$

where M_{i-1} is a previous model, $\frac{\partial l(\cdot)}{\partial \theta_{i-1}}$ is a partial differentiation for a gradient vector denoting a direction decreasing error of the predictive value, current stream set S_i, and current target t_i. In Equation (10), the gradient g_{on}, named as a online gradient, implies the current information which has different skewed data distribution from previous.

To reflect previous information into updated deep learning model M_i, the proposed system calculates offline gradient g_{off} including previous information in cloud computing as follows:

$$g_{off,k} = \frac{\partial l(M_{i-1}, B_k)}{\partial \theta_{i-1}} \tag{11}$$

where $k \in \mathbb{N}$ is less than i. B_i is a newly updated experience buffer containing previous information. In Equation (11), the gradient g_{off}, named as a offline gradient, includes the information of previous skewed data distribution.

Furthermore, to decrease negative effect between previous and current due to different skewed data distribution, the proposed scheduler computes the projected gradient g_{proj} as referred in [9]. The projected gradient g_{proj} can reflect the previous information including different skewed data distributions into deep learning model. Moreover, the projected gradient is only computed when a skewed data distribution is changed with a transition indicator τ, so, the proposed scheduler can reduce the entire processing time.

To reduce the processing time and skewed data distribution hazard, the computed gradient in the proposed scheduler is different from the conventional learning system. The adadelta optimizer computes gradient g_{i-1} as described in Equation (12) and accumulates gradient as shown in Equation (13) with scheduled gradients as follows:

$$g_{i-1} = \tau g_{proj} + (1-\tau) g_{on} \tag{12}$$

$$E[g^2]_{i-1} = \gamma * E[g^2]_{i-2} + (1-\gamma) * (\tau g_{proj} + (1-\tau) g_{on})^2 \tag{13}$$

where the hyperparameter γ is a decay constant and $E[g^2]_i$ is a moving average at time i. In Equation (12), the gradient g_{i-1} is scheduled with a transition indicator τ, then it affects a moving average of adadelta optimizer reducing gradient computation time and increasing the accuracy performance of predictor for skewed data distribution.

Finally, to update the current deep learning model for increasing accuracy performance in skewed data distribution than incremental deep learning scheme, we proposed scheduled model parameter updating algorithm as follows:

$$\theta_i = \theta_{i-1} - \frac{\sqrt{E[\Delta\theta^2]_{i-2} + \epsilon}}{\sqrt{E[g^2]_{i-1} + \epsilon}} \odot (\tau g_{proj} + (1-\tau) g_{on}) \tag{14}$$

where a constant ϵ is added to moving average.

In other words, when the transition indicator τ is 1 which means that there is a transition of skewed data distribution, the proposed scheduler computes projected gradient with online gradient g_{on} and offline gradient g_{off}. On the other hand, when the transition indicator τ is 0 meaning no change of data distribution, the hybrid scheduler ignore the projected gradient and only use online gradient without additional computation for projected gradient to reduce the processing time.

The pseudo code of proposed scheduler is shown in Algorithm 1. The whole procedure, from line 1 to line 36, is repeated at every time sequence i. The pre-processed stream set S_i is submitted to both edge and cloud computing. The lines 8 to 15 represent phase 1 and lines 16 to 25 describe phase 2. The lines from 2 to 6 in edge computing and lines from 26 to 35 in cloud computing depict final phase 3 of proposed scheduler. The proposed scheduler profiles the current and previous skewed data distribution based on cosine similarity score $score_1^-(\cdot)$ at phase 1. Then, the proposed scheduler selects one of consumption data from a distributed histogram buffer D_j at phase 2. In addition to, the newly built buffer $\mathcal{U}_i$ contains each consumption data u_j and the variance of cosine similarity in previous experience buffer B_{i-1} and newly built buffer $\mathcal{U}_i$ is compared and chosen to increase the diversity of power consumption pattern in lines 19 to 20. After that, with transition indicator τ the gradient for model parameter update is calculated for reducing accuracy deterioration in transition of skewed data distribution and decreasing processing time due to additional computation for projected gradient.

V. PERFORMANCE EVALUATION

A. EXPERIMENT ENVIRONMENT

1) CLUSTER SETTING

The edge-cloud cluster is built using the heterogeneous computing nodes. The specification of computing nodes are described in Table 1. We distinguish two nodes as follows: Computing Node 1 is used for representing Edge Computing which is regarded as having small computing power, and Computing Node 2 is also used for Cloud Computing having more speedy computational. All computing nodes are

Algorithm 1 Hybrid Deep Learning Scheduling Algorithm

procedure Hybrid(S_i, B_{i-1})
  if Edge **then**
    Calculate g_{on} in Equation (10)
    Send g_{on} by using $Comm(g_{on}, Edge)$
    Receive g_{i-1} from $Comm(g_{i-1}, Cloud)$
  end if
  if Cloud **then**
    $A \leftarrow S_i \cup B_{i-1}$
    $C_i \leftarrow score_1^-(A)$
    $\epsilon = \max(C_i) - \min(C_i)$
    for $j \leftarrow 1$ to n **do**
      $r_{i,lower} = min_i + \frac{(j-1)\cdot\epsilon}{n}$
      $r_{i,upper} = min_i + \frac{j\cdot\epsilon}{n}$
      $D_j = \{x_{i,k}^p | r_{i,lower} \leq c_k \leq r_{i,upper}\}$
    end for
    for $j \leftarrow 1$ to n **do**
      $u_j \leftarrow random(D_j)$
    end for
    $\mathcal{U}_i \leftarrow u_1, u_2, \ldots, u_n$
    $B_i = \arg\max_{x \in \{B_{i-1}, \mathcal{U}\}}(C_x)$
    if $B_i == B_{i-1}$ **then**
      $\tau \leftarrow 0$
    else
      $\tau \leftarrow 1$
    end if
    Receive g_{on} from $Comm(g_{on}, Edge)$
    for $k \leftarrow 1$ to $i-1$ **do**
      $g_{off,k} \leftarrow \frac{\partial l(M_{i-1}, B_k)}{\partial \theta_{i-1}}$
    end for
    Calculate g_{proj} following [9]
    $g_{i-1} \leftarrow \tau g_{proj} + (1-\tau) g_{on}$
    Send g_{i-1} by using $Comm(g_{i-1}, Cloud)$
  end if
  $E[g^2]_{i-1} \leftarrow \gamma * E[g^2]_{i-2} + (1-\gamma) * (\tau g_{proj} + (1-\tau) g_{on})$
  $\theta_i \leftarrow \theta_{i-1} - \frac{\sqrt{E[\Delta\theta^2]_{i-2}+\epsilon}}{\sqrt{E[g^2]_{i-1}+\epsilon}} \odot (\tau g_{proj} + (1-\tau) g_{on})$
  Update M_{i-1} to M_i by using θ_i
end procedure

TABLE 1. Detailed specification of each computing nodes.

	Computing Node 1	Computing Node 2
CPU	Intel i7-4790 @ 3.6 GHz	Intel i7-8700K @ 3.7 GHz
Core	4	6
Memory Bandwidth	25.6 GB/s	41.6 GB/s

installed with Ubuntu 16.04 with deep learning framework, Tensorflow 1.14 [22] and Keras. The input data streams in AMI dataset are transmitted by using Kafka messaging API.

2) AMI METER DATASET

Our experiment data is designed to validate the purpose of the proposed technique. Thus, we use a time-series data stream generated from smart meters that measure low voltage power in a field, such as household, general use, education, industrial and agricultural use in a certain area in Korea. Low voltage AMI meter data is collected by Korea Electric Power Corporation in 2017 at every 15 minutes. According to the paper [23], the low voltage is more challenging issue in predicting or regressing data than high voltage because the individuals using low voltage have more expanded complexity than high voltage customer. That is why we choose low voltage dataset to predict next short-term power consumption. The data stream is separated into each task training deep learning model because the proposed scheme is based on the environment where AMI data is continuously generated from smart meter and fed into the data analysis system. We use preprocessor to transform the meter data into time series data with length $p = 36$. Therefore, the physical time period for one pattern is 540 minutes when collection interval is 15 minutes. AMI data stream has total task $T = 12$ stream set containing different skewed data distribution because we assume that. On the power consumption data stream, each task has a disjoint subset with 4 classes in training. In other words, the total number of trained AMI ID is 48. The test data is untrained 12 AMI ID and there is one AMI ID for each task. It means that the entire test AMI ID is 12. Each stream set contains around 11500 AMI meter data, and the total number of data is around 140000. Figure 8 shows a correlation coefficient map for power consumption patterns according to data sequences. The dashed line shows the border of stream sets in data sequence according to set sequences, and it has same values of horizontal dashed lines. The blue line means the value caused by all-zero patterns, and it means that some

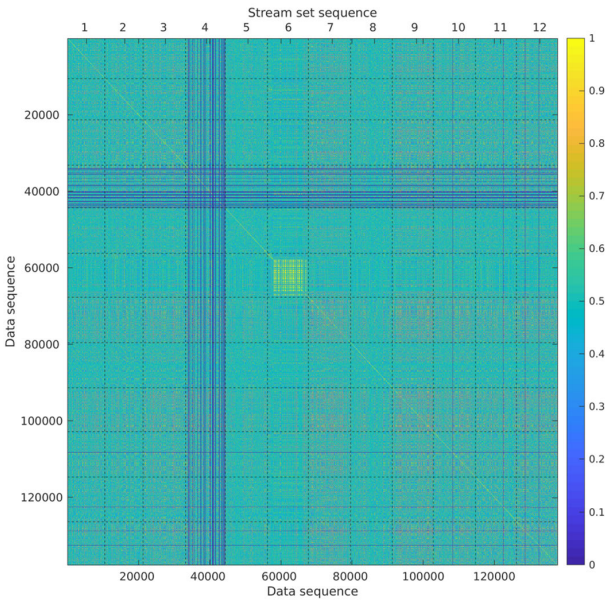


FIGURE 8. A correlation coefficient map of training power consumption patterns according to data sequence. The vertical dashed lines indicates the border of stream sets according to top x-axis and it has same values of horizontal dashed lines.

customer may not use any power during 540 minutes in physical. All blue lines should be presented symmetrically but it did not because of scale-down during printing out. However, we note that we can still check data pattern distribution. When focusing on one line except for blue lines regardless of horizontal or vertical, the correlation coefficient values are diverse from -1 to 1 and most of consumption patterns has 0 correlation coefficient. This means that the data patterns are rapidly varying among data meters and time sequences.

3) CNN-LSTM MODEL AND EXPERIENCE BUFFER

A hybrid model, Convolutional Neural Network Long Short-term Memory (CNN-LSTM), has advantages both spatially and temporally hidden feature training by having CNN layers for spatial feature extraction with LSTM supporting the prediction of sequential data, such as time-series data. We set the input dimension as 36 with spatially reformation as 6 by 6 and the output dimension as 1 for short-term prediction with fully-connected (FC) layer. In Conv1D layer, the filter size is 64 and kernel size is 3 with using relu activation layer. LSTM layer has 50 hidden units to extract temporal feature. The model has 24267 parameters and all of parameter is trainable. Additionally, the total size of offline experience buffer is limited to 100 examples for each stream set because of limited memory resource assumption in Section II.

B. PERFORMANCE METRICS

1) AVERAGE RMSE

The objective of online deep learning is stability with high precision and service level agreement for real-time processing ability, when forecasting future power consumption demand with greater precision in order to decrease energy cost and supply a stable power provision. To measure stability under the changing consumption pattern frequency distribution, a prediction error $E_{i,j}$ at test task i is evaluated at the end of each training task j defined to examine the effect of changing data distribution. It means that the predictive performance after training j on test dataset i. In other words, when the high $E_{i,j}$, it also means that training task j causes a positive effect on test dataset i. We use root-mean-square-error (RMSE) as a error metric because it is commonly used to measure predictive error E in regardless of positive and negative value. Moreover, to evaluate the accuracy performance on all test dataset having different data distribution, the average-RMSE (ARMSE) $ARMSE_j$ is calculated at the end of each learning task j as described by below:

$$ARMSE_j = \frac{1}{T_{total}} \sum_{i=1}^{T_{total}} E_{i,j} \quad 1 \leq j \leq T_{total} \tag{15}$$

The ARMSE shows how well the deep learning model learns the current tasks without forgetting the previous ones. We decide to use ARMSE metric because the objective of the proposed algorithm is not only high predictive error in current task, but also high accuracy in previous task.

2) PROCESSING TIME

The other objective of our proposed scheduler is the reduction of processing time for AMI meter data stream of multiple user in different skewed data distribution. To reduce the performance degradation by multiple AMI IDs, according to related works [9], [13], the representative pattern of skewed data distribution is needed to be preserved in memory buffer and training procedure with gradient regularization for deep learning model is required. That requirement causes the memory processing time and training time. Therefore, first we measure the training time T_{train} to evaluate the proposed scheduler comparing to conventional continual learning method. Second, memory processing time T_{mem} is calculated to validate the proposed algorithm for low latency system in online learning procedure comparing the other possible memory buffer processing scheme.

3) COMPARED METHODS

To evaluate the our proposed algorithm effectively, we compare our proposed method to four alternatives:

- **RingBuffer** referred to in [9] uses gradient regularization for incremental class learning and preserves only recent data in experience buffer using ringbuffer algorithm to manage limited memory space. This method does not require additional memory processing time because it does not consider what data to preserve in experience buffer.
- **ClassWise-Kmeans (Cw-Kmeans)** We use K-means for each clients to store representative patterns in experience buffer. We uses gradient regularization method for different skewed data distribution.
- **CosSim** with gradient regularization uses cosine similarity distribution to store representative data in offline experience buffer. However, it is not scheduled as shown in phase 3 of proposed scheduler. This method is mainly used to represent the effective reduction of our proposed scheme at experiment 2.
- **Incremental Learning** is commonly used to continuously incorporate data stream into learning-based model. This method is compared to show the increasing accuracy in our proposed method for multi-AMI data stream.

C. RESULTS AND DISCUSSION

In this section, we show evaluations with the metrics on the dataset introduced in previous section and discuss about the results. The average of 50 experimental results was obtained because the results may vary by randomly initialized CNN-LSTM model and its parameter values.

1) EXPERIMENT 1: VALIDATION OF ACCURACY FOR MULTI-AMI DATA STREAM

In this experiment, we validate the accuracy performance in proposed scheduler. Therefore, we computes an averaged results, ARMSE denoting how well the deep learning model

learns current data feature without losing different skewed data distribution, at each task round is presented in Figure 9.

In Figure 9, the time sequence of stream set starts from 0 to 12 which means that predictive error on untrained initialized CNN-LSTM model at zero sequence and final all trained deep learning model at twelve sequence. The ARMSE shows that accuracy performance on all test dataset. The dark gray with triangle mark shows ringbuffer method which is conventional method of gradient regularization for continual learning. It shows best performance at time sequence 2, however, the ARMSE at other time sequence is not good when comparing the other method. On the other hand, the Cw-Kmeans method has two best accuracy point in time sequence 3, 5, 12. The CosSim method shows shows best performance at 4, 6, 8, 11 and proposed method has best ARMSE at 9.

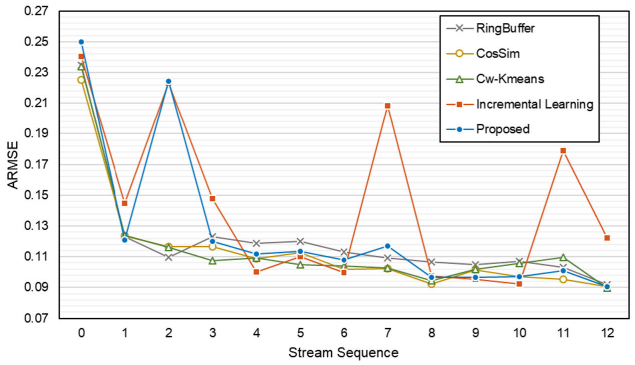


FIGURE 9. The performance comparison of ARMSE with respect to each time sequence round. The x-axis shows time sequence of stream set and the y-axis describes ARMSE. The blue line shows accuracy performance of proposed scheme on test data set and the other shows the accuracy performance of compared methods.

For test set 6, 7, 8 and 10, 11, 12, we checked the prediction results comparing incremental learning and proposed method as shown in Figure 10. The proposed method shows better accuracy than the incremental learning. The proposed technique for each set of streams, except for the second time sequence, showed a relatively similar accuracy to the performance of other techniques that refer to previous information in each stream without scheduling it. In the case of the second time sequence, we can see that there are other patterns that occur when the variance of the cosine similarity core of the stored pattern is not increased, and this needs to be improved later. Moreover, the most important time sequence is final round because it means that the deep learning model trains all data stream at final round and high accuracy at final test task states that the method learns well all data stream without losing all skewed data distribution. For more detail evaluation value, final ARMSE at final round 12 is shown in Table 2.

According to Table 2, comparing Incremental Learning, the all method based on gradient regularization show better accuracy performance. It means that the changing distribution problem in multiple AMI customers is decreasing with projected gradient method. In addition to, the final ARMSE evaluation results, which show predictive performance for both

TABLE 2. Final ARMSE evaluations. The best value and proposed method is denoted as bold. Therefore, in final ARMSE at time sequence 12, Class-Wise Kmeans shows best accuracy performance. The proposed algorithm shows as similar as CosSim method without hybrid scheduling.

Method	final ARMSE
Ring buffer	0.0918
Class-Wise Kmeans	**0.0897**
CosSim	0.0904
Proposed	0.0905
Incremental Deep Learning	0.1223

current and previous distribution plots for various distribution plots, showed the third lowest error in the proposed algorithm compared to the existing method. For the CosSim method of which the strategy of experience buffer management is the same, but increasing processing time by regularizing gradient for every time, it shows the final ARMSE, which is almost the same as the proposed method of hybrid scheduling.

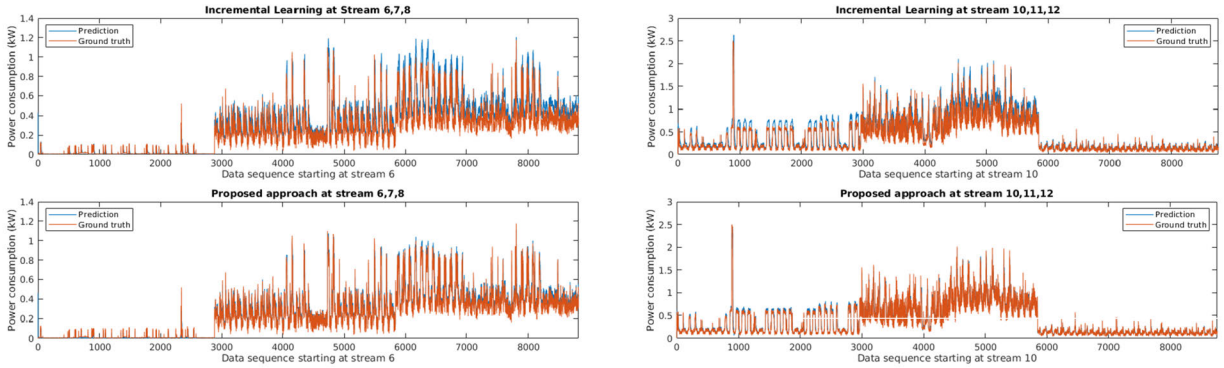
2) EXPERIMENT 2: VALIDATION OF MEMORY PROCESSING AND TRAINING TIME

To validate the proposed hybrid deep learning scheduling scheme, we evaluated two processing time: memory processing time and training time. To reduce the performance degradation by multiple AMI IDs, according to related works [9], [13], the representative pattern of skewed data distribution is needed to be preserved in memory buffer. That requirement causes the memory processing time as described with logarithmic scale in Figure 11 (a).

Comparing other memory processing method, RingBuffer method is denoted as a baseline without processing memory because it just preserves recent data pattern in a memory buffer. The blue bar shows the proposed scheme and the other methods are described as a dark dark gray (Class-Wise Kmeans), a light gray bar (RingBuffer), and a yellow bar (CosSim). For each update method that selects and stores multiple distributions in the buffer, the memory processing time cannot be faster than RingBuffer that stores only the most recent data, but the proposed scheme has less memory processing time than the one that uses K-means clustering for higher accuracy.

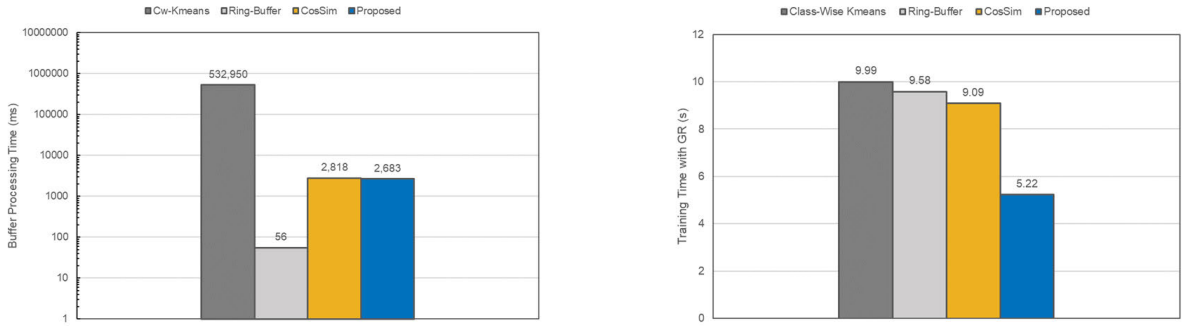
Additionally, we estimates the training time to certify the accelerated training process by hybrid deep learning scheduling. When implementing experiment, the elapsed time between start and end of training is measured. Furthermore, we calculates averaged training time for more explicit evaluations as described in Figure 11 (b). For each methods shows training time as like followings: about 9.99 seconds in a dark gray bar (Class-Wise Kemans), about 9.58 seconds in a light gray bar (Ring-Buffer), about 9.09 seconds in a yellow bar (CosSim), and a 5.22 seconds in a blue bar (Proposed Scheme). The training time of proposed scheme is 43% to 47% faster than traditional methods implementing gradient regularization for every stream.

We summarizes the processing time with final ARMSE results in Table 3. As shown in Table 3, we note that there



(a) Prediction results on test set 6,7,8 (b) Prediction results on test set 10,11,12

FIGURE 10. A comparable test results of predictions in Incremental Learning and Proposed method. (a) is the results on test stream set 6, 7, and 8. (b) is the prediction results on test stream set 10, 11, and 12.



(a) Memory processing time (b) Training time

FIGURE 11. The performance evaluations. (a) shows the performance comparison of memory processing time and (b) presents the performance comparison of training time, according to four methods: Class-Wise Kmeans, Ring-Buffer (as a baseline without memory processing), CosSim (as a baseline without scheduling), and Proposed scheme.

TABLE 3. Final ARMSE, memory processing time, and training time for each stream data. The best method in three metric is denoted as bold. Therefore, in final ARMSE at time sequence 12, Class-Wise Kmeans shows best accuracy performance. The ringbuffer method has the lowest memory processing time because it just preserves recent data. In training time with gradient regularization (GR), the proposed algorithm shows the lowest training latency.

Method	final ARMSE	Buffer Time (ms)	Training Time with GR (s)
Ring buffer	0.0918	**56**	9.58
Class-Wise Kmeans	**0.0897**	532,960	9.99
CosSim	0.0904	2,818	9.09
Proposed	0.0905	2,683	**5.22**
Incremental Deep Learning	0.1223	-	-

is a trade-off relation between final ARMSE and memory processing time. The Class-Wise Kmeans method has lowest predictive error but can not satisfy real-time service level agreement for online deep learning platform. Adding buffer time and training time, the processing time of proposed method equals 7.903 seconds. Therefore, the processing performance is improved as much as 21.95% when compared with Ring-Buffer method.

3) EXPERIMENT 3: VALIDATION OF CRITERION VECTOR $\vec{1}$

Finally, when measuring cosine similarity, we conducted an experiment based on three criterion vector: the proposed $\vec{1}$ vector, randomly selected vectors from a given set of streams, and the first power consumption pattern vectors, as described in Figure 12 and Table 4.

TABLE 4. A performance comparison of variance based on criterion vector. When variance is lower than other, it means that the given criterion vector has more stability on various skewed data distribution than other.

Criterion Vector	Proposed	Random	Initial
Variance of ARMSE	**0.1165**	0.1182	0.1174

In Figure 12, the dashed blue line shows proposed scheme and grayish dashed lines represent other method's ARMSE fluctuation. If the proposed method is more stable in expressing data distribution than other methods, the degree of variance in ARMSE would be lower than

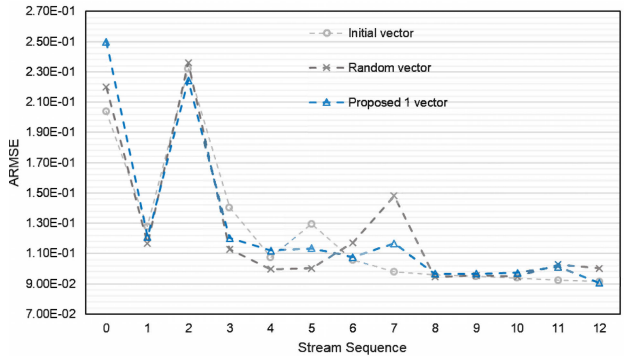


FIGURE 12. A performance comparison of ARMSE corresponding to the time sequence of stream set based on criterion vectors: the proposed $\vec{1}$ vector in Equation (3), randomly selected vectors from a given set of streams, and the first power consumption pattern vectors.

in other methods. Therefore, we measured the variance of ARMSE, as described in Table 4, and it shows our proposed criterion vector $\vec{1}$ shows better stability than others.

VI. CONCLUSION

In this paper, we proposed a cooperating edge cloud-based hybrid online learning scheme to improve and accelerate learning performance in a multi-AMI ID environment where the distribution of biased data is dramatically different. By analyzing cosine similarity frequency distribution, a model that can recognize skewed distribution of pattern was proposed to reduce skewness and increase the diversity of information in experience buffer. Controlling online and offline gradient computation, a selective model is robust to different distribution in continuously incoming data stream and also reduces the processing time.

In order to verify the performance of the proposed algorithm, we conducted a comparative experiment on the proposed method and the existing common learning methods. When comparing performance against average prediction errors for the varying distribution of test data, the accuracy performance was higher than the incremental learning method and similar to the traditional conceptual learning technique. Compared to the training time performance, processing performance was 43% to 47% faster than conventional learning techniques. Although the proposed approach requires additional computation procedure to regulate online and offline gradients, the processing performance is increased as much as 21.95% when compared with traditional Ring-Buffer method.

Our approach shows several directions for future works. To reduce error caused by valiance failures, the proposed algorithm should be supplemented by supplementing the steps for recognizing distribution and sustainable scheduling methods. Other algorithms may need to be considered because the cosine similarity used to recognize the distribution of data may be ineffective if the data are not power consumption data. Despite of future enhancement points, it is hoped that this study will serve as a platform from which studies of greater depth and specificity may be undertaken in smart grid applying deep learning and edge-cloud computing.

REFERENCES

[1] Y. Kim, N.-G. Myoung, and S.-Y. Lee, "Study on AMI system of KEPCO," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Nov. 2010, pp. 459–460.

[2] B. Neupane, K. S. Perera, Z. Aung, and W. L. Woon, "Artificial neural network-based electricity price forecasting for smart grid deployment," in *Proc. Int. Conf. Comput. Syst. Ind. Informat.*, Dec. 2012, pp. 1–6.

[3] X. Qiu, P. N. Suganthan, and G. A. J. Amaratunga, "Ensemble incremental learning random vector functional link network for short-term electric load forecasting," *Knowl.-Based Syst.*, vol. 145, pp. 182–196, Apr. 2018.

[4] Y. Lan, Y. C. Soh, and G.-B. Huang, "Ensemble of online sequential extreme learning machine," *Neurocomputing*, vol. 72, nos. 13–15, pp. 3391–3395, Aug. 2009.

[5] A. Tsymbal, "The problem of concept drift: Definitions and related work," *Comput. Sci. Dept., Trinity College Dublin*, vol. 106, no. 2, p. 58, 2004.

[6] D. M. J. Tax and P. Laskov, "Online SVM learning: From classification to data description and back," in *Proc. IEEE XIII Workshop Neural Netw. Signal Process.*, Sep. 2003, pp. 499–508.

[7] A. A. Benczúr, L. Kocsis, and R. Pálovics, "Online machine learning in big data streams," 2018, *arXiv:1802.05872*. [Online]. Available: http://arxiv.org/abs/1802.05872

[8] D. Wang, B. Liu, P.-N. Tan, and L. Luo, "OMuLeT: Online multi-lead time location prediction for hurricane trajectory forecasting," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 963–970.

[9] D. Lopez-Paz, "Gradient episodic memory for continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6467–6476.

[10] S. Ruping, "Incremental learning with support vector machines," in *Proc. IEEE Int. Conf. Data Mining*, Nov./Dec. 2001, pp. 641–642.

[11] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "ICaRL: Incremental classifier and representation learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2001–2010.

[12] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari, "End-to-end incremental learning," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 233–248.

[13] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with A-GEM," 2018, *arXiv:1812.00420*. [Online]. Available: http://arxiv.org/abs/1812.00420

[14] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 2990–2999.

[15] N. Dehak, R. Dehak, J. R. Glass, D. A. Reynolds, and P. Kenny, "Cosine similarity scoring without score normalization techniques," in *Proc. Odyssey*, 2010, p. 15.

[16] X. Liu, L. Golab, W. Golab, I. F. Ilyas, and S. Jin, "Smart meter data analytics: Systems, algorithms, and benchmarking," *ACM Trans. Database Syst.*, vol. 42, no. 1, p. 2, 2017.

[17] D. B. Rawat and C. Bajracharya, "Detection of false data injection attacks in smart grid communication systems," *IEEE Signal Process. Lett.*, vol. 22, no. 10, pp. 1652–1656, Oct. 2015.

[18] P.-N. Tan, *Introduction to Data Mining*. London, U.K.: Pearson, 2018.

[19] J. Arroyo and C. Maté, "Forecasting histogram time series with k-nearest neighbours methods," *Int. J. Forecasting*, vol. 25, no. 1, pp. 192–207, Jan. 2009.

[20] M. P. Wand, "Data-based choice of histogram bin width," *Amer. Statistician*, vol. 51, no. 1, pp. 59–64, Feb. 1997.

[21] G. Valentini and T. G. Dietterich, "Bias-variance analysis of support vector machines for the development of SVM-based ensemble methods," *J. Mach. Learn. Res.*, vol. 5, pp. 725–775, Jul. 2004.

[22] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, and M. Isard, "Tensorflow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Operating Syst. Design Implement. (OSDI)*, 2016, pp. 265–283.

[23] M. Kim, S. Park, J. Lee, Y. Joo, and J. Choi, "Learning-based adaptive imputation methodwith kNN algorithm for missing power data," *Energies*, vol. 10, no. 10, p. 1668, 2017.

CHANGHA LEE received the B.S. degree in electronic engineering from Hanyang University, Seoul, South Korea, in 2018, and the M.S. degree in electronic engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2020, where he is currently pursuing the Ph.D. degree. Since 2018, he has been a member of the Network and Computing Laboratory, KAIST. His current research interests include deep learning acceleration platform, online learning, and energy prediction service platform.

SEONG-HWAN KIM (Associate Member, IEEE) received the B.S. degree in media and communications engineering from Hanyang University, Seoul, South Korea, in 2012, and the integrated master's Ph.D. degree in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2020. He is currently a Postdoctoral Researcher with the Network and Computing Laboratory, KAIST. His research interests include cloud brokering systems, deep learning, and energy service platform and others.

CHAN-HYUN YOUN (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in electronics engineering from Kyungpook National University, Daegu, South Korea, in 1981 and 1985, respectively, and the Ph.D. degree in electrical and communications engineering from Tohoku University, Sendai, Japan, in 1994. Since 1997, he has been a Professor with the School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea. From 2013 to 2017, he was an Associate Vice-President of the Office of Planning and Budgets, KAIST. His research interests include distributed high performance computing systems, cloud computing systems, edge computing systems, deep learning framework, and others. He received the Best Paper Award from CloudComp 2014, and wrote a book on *Cloud Broker and Cloudlet for Workflow Scheduling*, Springer, in 2017. He was a General Chair of the 6th EAI International Conference on Cloud Computing (Cloud Comp 2015), KAIST, in 2015. He was a Guest Editor of the IEEE Wireless Communications, in 2016.