energies

MDPI

Article

Deep Neural Network Based Demand Side Short Term Load Forecasting †

Seunghyoung Ryu [1], Jaekoo Noh [2] and Hongseok Kim [1,*]

[1] Department of Electronic Engineering, Sogang University, 35 Baekbeom-ro, Mapo-gu, Seoul 121-742, Korea; shryu@sogang.ac.kr

[2] Software Center, Korea Electric Power Corporation (KEPCO), 105 Munji Road, Yuseong-Gu, Daejeon 305-760, Korea; jknoh@kepco.co.kr

* Correspondence: hongseok@sogang.ac.kr; Tel.: +82-2-705-7989

† This paper is an extended version of our paper published in Proceedings of the 2016 IEEE International Conference on Smart Grid Communications (SmartGridComm), Sydney, Australia, 6–9 November 2016.

Academic Editor: José C. Riquelme
Received: 15 July 2016; Accepted: 16 December 2016; Published: 22 December 2016

Abstract: In the smart grid, one of the most important research areas is load forecasting; it spans from traditional time series analyses to recent machine learning approaches and mostly focuses on forecasting aggregated electricity consumption. However, the importance of demand side energy management, including individual load forecasting, is becoming critical. In this paper, we propose deep neural network (DNN)-based load forecasting models and apply them to a demand side empirical load database. DNNs are trained in two different ways: a pre-training restricted Boltzmann machine and using the rectified linear unit without pre-training. DNN forecasting models are trained by individual customer's electricity consumption data and regional meteorological elements. To verify the performance of DNNs, forecasting results are compared with a shallow neural network (SNN), a double seasonal Holt–Winters (DSHW) model and the autoregressive integrated moving average (ARIMA). The mean absolute percentage error (MAPE) and relative root mean square error (RRMSE) are used for verification. Our results show that DNNs exhibit accurate and robust predictions compared to other forecasting models, e.g., MAPE and RRMSE are reduced by up to 17% and 22% compared to SNN and 9% and 29% compared to DSHW.

Keywords: short-term load forecasting; deep neural network; deep learning; rectified linear unit (ReLU); exponential smoothing; smart grid; restricted Boltzmann machine (RBM); pre-training

1. Introduction

Load forecasting is one of the main research areas in the smart grid and can be classified depending on its target forecasting ranges from minutes to years. Among these, short-term load forecasting (STLF), which focuses on predicting the load of hours or days ahead, secures attention in the smart grid because of its wide applicability to demand side management, energy storage operation, peak load reduction, etc. One of the main characteristics of the smart grid is bidirectional communications, which breaks down the border between electricity generation and consumption. Bidirectional communications can be realized with the deployment of advanced metering infrastructure (AMI), which is a prerequisite of many smart grid services, such as demand response (DR), targeted dynamic tariffs, load monitoring, outage detection and restoration, the customer information system, etc. [1,2].

However, the collapse of the border between generation and consumption increases the complexity of the grid. Now, consumers can produce electricity from renewable sources, such as photovoltaic cells and wind turbines. Furthermore, by using an energy storage system (ESS), consumers can actively shape their power demand. In addition to these changes, deregulation and privatization

of the electricity market produce different aspects than the traditional market. In the literature, load forecasting is vitally important for the electricity industry in the deregulated electricity market. Furthermore, accurate models for electric load forecasting are also essential to the operation and planning of a utility company [3]. In addition, DR is considered as one of the lease expensive resources available for operating the system according to the new paradigm under deregulation [4], and thus, forecasting the DR resource is important in the DR market. We also note that accurate demand forecasting can encourage customers to participate in DR programs and to receive monetary rewards from the utility company [5]. Meanwhile, in the deregulated retail market, various end-user services, such as customer load management, require load forecasting to attract more customers. These characteristics imply that the role of consumers would be more significant than before, and thus, the scope of forecasting is shifted from the macroscopic view to the microscopic view.

There are many research activities on forecasting models from classical time series analysis to recent machine learning approaches. Time series analysis [6], one of the most popular load forecasting methods, includes the autoregressive integrated moving average (ARIMA), exponential smoothing and the Kalman filter. In [7,8], the authors developed a double seasonal Holt–Winters model (exponential smoothing) that showed better results than other models like ARIMA or neural networks.

Another branch of load forecasting is to use an artificial neural network (ANN) model [9,10]. Since ANN can model nonlinearity, it is widely used for various forecasting applications. For example, ANN-based STLF models are well summarized in [11]. It is known that network structure plays an important role in ANN because feature information about the model, including the forecasting period or what variables are used, is reflected in the structure of neural networks. The most common type of ANNs is a multilayer perceptron (MLP) that forecasts a load profile using previous load data (e.g., [12,13]). So far, many ANN models have been published in the literature and showed adequate forecasting results. In [14], an ANN-based hourly load forecasting model was proposed for a household application. In [15], electricity price is considered as an input parameter for load forecasting in real-time pricing markets. Furthermore, neural network models could be combined with other methods: fuzzy logic [16], wavelet transform [17] or both [18]. Some literature uses different methods in training neural networks. In [19], a continuous genetic algorithm was proposed to train a neural network-based STLF model. Other types of neural networks, such as a self-organizing map (SOM), are also used for prediction and classification [20]. In [5,21], the two-stage adaptive prediction model based on SOM and K-means clustering was used for a buildings and apartments dataset.

A deep neural network (DNN) is an ANN with more layers than the typical three layers of MLP. The deep structure increases the feature abstraction capability of neural networks. Recently, the progress of the Internet of Things (IoT) and big data enables the adoption of DNNs in diverse research fields. However, to the best of our knowledge, there are only a few DNN-based electricity load forecasting models; in [22], DNN is used for time series wind forecasting, and DNN with the discriminative pre-training method is used for time series electricity load forecasting in [23]. In [24], a deep belief network (DBN) was used as a part of ensemble learning. They used support vector regression with time series forecasting results of DBN. Our approach differs from the above in that: (1) We adopt the DNN forecasting model for various types of load consumption data of the individual demand side level; (2) We forecast daily load profiles the day ahead; (3) We compare two different DNNs: The restricted Boltzmann machine (RBM) pre-training method and DNN with rectified linear unit (ReLU).

We summarize our key contributions as follows. First, we apply deep learning to demand side STLF and propose a DNN-based STLF framework. Specifically, we train DNNs in two different ways: pre-training RBM and using ReLU without pre-training. With using ReLU, DNN can be easily trained and performs better than the shallow neural network (SNN) with one hidden layer. Second, we carefully investigate the problem of training DNNs with customer load data. Overfitting may occur during DNN training if the volume of available individual load data is relatively small considering the size of the neural network. Our experimental results show that DNN can be well

trained with customer's three-year load data. Third, we test the DNN-based STLF model using various load databases of the demand side level and for the aggregated case. DNN models are trained for 40 big-sized industrial customers and the aggregated consumption of high usage customers in Korea. We compare DNN forecasting results with the typical three-layered SNN, seasonal ARIMA and the double seasonal Holt–Winters (DSHW) model. Numerical results indicate that the DNN-based STLF model has lower mean absolute percentage error (MAPE) and relative root mean square error (RRMSE) by up to 17% and 22% compared to SNN and by 9% and 29% compared to DSHW.

The rest of this paper is organized as follows. In Section 2, basic ANNs and DNNs will be introduced. In Section 3, we propose a DNN-based framework, and the implementation issues of forecasting model will be followed in Section 4. Experimental results and the conclusion will be given in Sections 5 and 6, respectively.

2. Methodology

Recently, DNNs have shown tremendous progress in many research fields, such as acoustic modeling, natural language processing and image recognition. The layout of ANNs had remained shallow due to the problems of training DNNs, such as vanishing gradient, overfitting and lack of computational power. In 2006, Hinton et al. published a seminal paper about deep learning [25], and after that, deep learning developed rapidly. With many hidden layers, DNNs have the ability to capture highly abstracted features from training data. Since load profiles have nonlinear characteristics among various factors that affect the shapes of load patterns, it is reasonable to use DNN as a forecast model. In this paper, we adopt two different DNN models to learn complicated relations between weather variables, dates and previous consumptions for individual customers. Then, the DNN produces a day-ahead prediction of the 24-h load profile according to past observations.

2.1. Artificial Neural Networks

An artificial neuron is a mathematical model imitating a biological neuron of the central nervous system. It shows nonlinear reactions according to input signals. Figure 1 shows the structure of the artificial neuron and explains the mechanism of the neuron's nonlinear reactions according to the inputs.

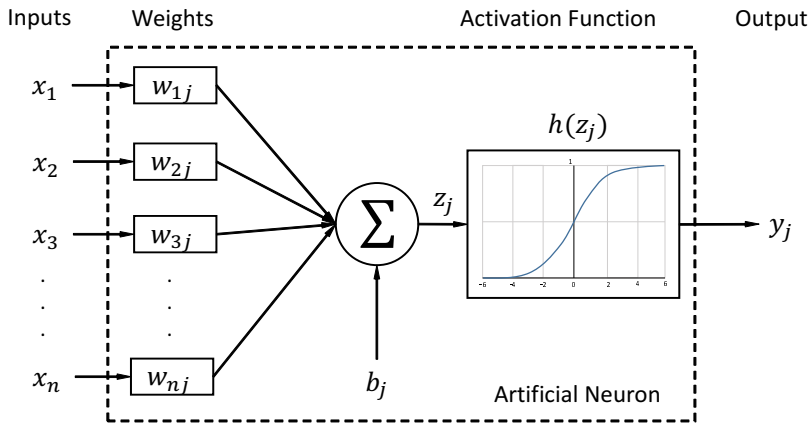

Figure 1. Structure of the artificial neuron.

The artificial neuron consists of inputs, bias, weights, output and an activation function. A typical neuron j with the number of n inputs is as depicted in Figure 1. According to the shape of the activation function, the output could be construed as the intensity of reaction or the turn-on probability of the neuron.

ANN is a system of connected artificial neurons, and it has the ability to model any arbitrary nonlinear function [26]. A typical structure of ANN is based on multilayer perceptron (MLP), depicted in Figure 2. The MLP has a three-layered structure; input, hidden and output layers. Each layer consists of neurons without intra-layer connections. However, between the layers, there exist full weighted connections among neurons. The elements of MLP are as follows. By using a superscript l to denote a layer, the output vector $\mathbf{y}$ of the layer l is:

$$\mathbf{y}^l = \mathbf{h}(\mathbf{z}^l) = \mathbf{h}((\mathbf{W}^l)^T \mathbf{y}^{l-1} + \mathbf{b}^l) \tag{1}$$

where $\mathbf{h}(\cdot)$ represents a nonlinear vector activation function, $\mathbf{W}^l = [w_{ij}^l]$ and $\mathbf{b}^l$ represent a weight matrix and a bias vector, respectively. The structure implies that the MLP has the power of manipulating the input space by adjusting weight matrices between layers. The main process of building MLP is to calculate appropriate weight matrices that produce the desired output corresponding to given data. The back propagation (BP) algorithm is used to update weight matrices to minimize the error between the target value and the value produced by MLP. Then, w_{ij}^l is updated by:

$$\Delta w_{ij}^l = -\eta \frac{\partial E}{\partial w_{ij}^l} \tag{2}$$

where η represents a learning rate and E is the error. Δw_{ij}^l can be derived by the chain rule [27].

In the BP algorithm, error signal δ propagates backwards to calculate Δw_{ij}^l. In SNN, δ propagates well to the input layer. However, in DNN, since the gradient of typical activation functions is zero in the saturation area, δ also goes to zero as it propagates toward the input layer. This problem of training DNNs is called vanishing gradient and made it hard to train DNNs.

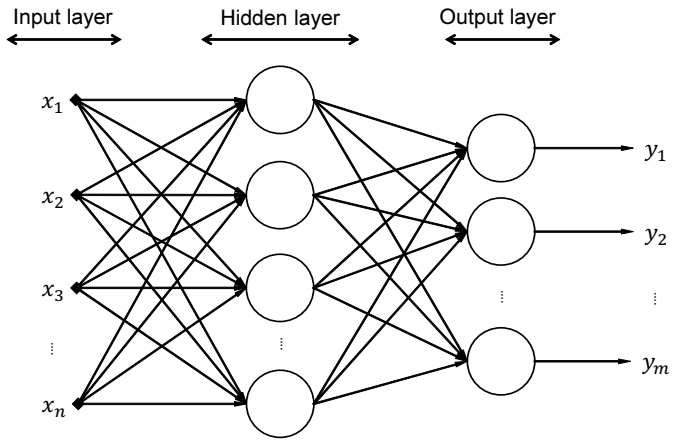

Figure 2. Multilayer perceptron.

2.2. Deep Neural Networks

We adopt two techniques to overcome the difficulties of training the deep structure.

2.2.1. RBM Pre-Training

The first method is to pre-train the DNN by stacking RBMs, which is known as DBN. It is known that neural networks with multiple hidden layers can be well trained by the BP algorithm when weights are initialized by stacked RBMs [25]. Furthermore, unsupervised pre-training leads the network to better local minima that support better generalization [28].

RBM is a neural network that has only two layers, called the visible and hidden layers. Typical RBM has binary nodes in each layer denoted as $\mathbf{v} = \{v_i\}$ and $\mathbf{h} = \{h_j\}$, respectively. For any (v_i, h_j)

pair, there exists a bidirectional connection, but no intra-layer connection. The probability to have a specific $(\mathbf{v}, \mathbf{h})$ configuration is determined by its energy $E(\mathbf{v}, \mathbf{h})$ in the form of:

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v}, \mathbf{h})} \tag{3}$$

where $Z = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}$ is a normalization factor. With bias vectors $\mathbf{b_v}$ for visible, $\mathbf{b_h}$ for hidden and weight matrix $\mathbf{W}$ between $\mathbf{v}$ and $\mathbf{h}$, the energy of RBM is defined as [29]:

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{b}_\mathbf{v}^T \mathbf{v} - \mathbf{b}_\mathbf{h}^T \mathbf{h} - \mathbf{v}^T \mathbf{W} \mathbf{h} \tag{4}$$

Since there are no intra-layer connections, the states of the visible and hidden layer are conditionally independent of each other. Thus, the conditional distributions $p(\mathbf{v}|\mathbf{h})$ and $p(\mathbf{h}|\mathbf{v})$ are:

$$p(\mathbf{v}|\mathbf{h}) = \prod_i^{N_\mathbf{v}} p(v_i|\mathbf{h}) \tag{5}$$

$$p(\mathbf{h}|\mathbf{v}) = \prod_j^{N_\mathbf{h}} p(h_j|\mathbf{v}) \tag{6}$$

where $N_\mathbf{v}$ and $N_\mathbf{h}$ are the numbers of visible nodes and hidden nodes. Then, the probability of the specific node turning on is:

$$p(h_j = 1|\mathbf{v}) = \sigma(\mathbf{b}_{\mathbf{h},j} + \mathbf{v}^T \mathbf{W}_j) \tag{7}$$

$$p(v_i = 1|\mathbf{h}) = \sigma(\mathbf{b}_{\mathbf{v},i} + \mathbf{h}^T (\mathbf{W}^T)_i) \tag{8}$$

where $\sigma(x) = 1/(1 + e^{-x})$ and $\mathbf{W}_j$ represents the j-th column vector of the matrix $\mathbf{W}$. RBM is then trained to maximize the log probability of $\mathbf{v}$ over given training data. The contrastive divergence algorithm is widely known to update $\mathbf{W}$. It performs k-step Gibbs sampling to generate a reconstructed image of the given training data and updates $\mathbf{W}$ with:

$$\Delta w_{ij} = \epsilon(\langle \mathbf{v}, \mathbf{h} \rangle_{data} - \langle \mathbf{v}, \mathbf{h} \rangle_{recon}) \tag{9}$$

where ϵ is a learning rate, and the angle brackets are expectations under the distribution of the subscript [29]. Geometrically, $\Delta\mathbf{W}$ is the direction of lowering/increasing the energy of training/reconstructed data.

The RBM pre-training method initializes DNN with weights of the stacked RBMs that are already trained with the given data. The RBM stacking process is described in Figure 3. First, train the RBM with training data. Then, its upper RBM is trained from the hidden layer of the previous RBM. The hidden layer of the lower RBM becomes the visible layer of its new upper RBM. In this manner, the learning and stacking procedure iterates until it has the desired network structure. After unsupervised RBM pre-training, supervised fine-tuning is performed with the BP algorithm. Because the network already knows the feature of the given training data, the pre-training method helps to train DNN.

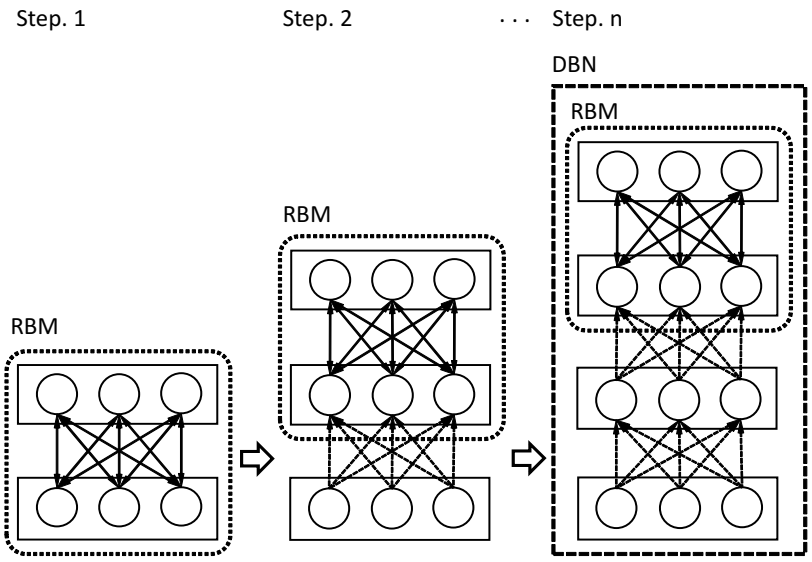

Figure 3. Stacking RBM.

2.2.2. ReLU

Another method to train the DNN is much simpler than pre-training: using ReLUs instead of typical sigmoid units [30]. The ReLU has a rectifier $h(z) = \max(0, z)$ as an activation function that is described in Figure 4. Since the gradient of the rectifier is 1 if $z \geq 0$ and 0 otherwise, error signal δ is still alive when it propagates backward to the lowest layer, and DNN can be trained well with the BP algorithm without the vanishing gradient problem. Furthermore, using ReLU has an advantage in training time due to the characteristic of its gradient.

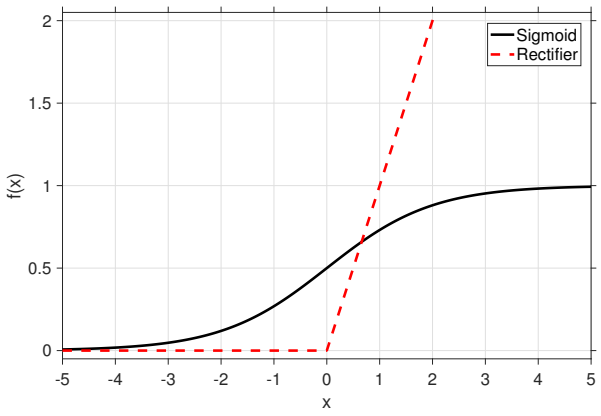

Figure 4. Activation functions.

3. Proposed Framework

In this section, we describe the proposed load forecasting framework using DNNs as depicted in Figure 5. The framework is based on the knowledge discovery in databases (KDD) process. Figure 5 could be divided into three major parts: data processing module, DNN training module and DNN forecasting module.

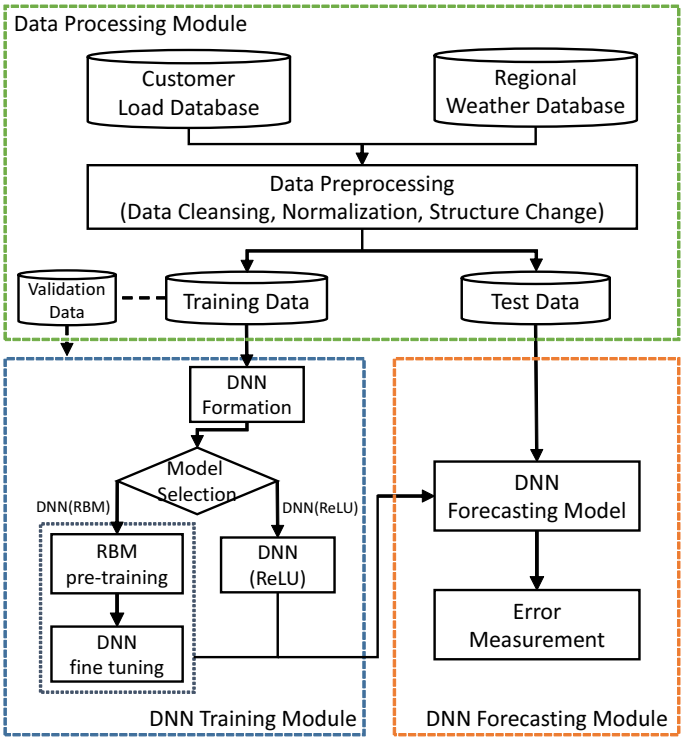

Figure 5. The proposed deep neural network (DNN) short-term load forecasting framework.

3.1. Data Processing

The upper part of Figure 5 shows the data processing module. First, we acquire demand side hourly load data provided by Korea Electric Power Corporation (KEPCO). Each customer in the database has information about time, power consumptions, province and industrial category in accordance with the Korea Standard Industrial Classification (KSIC). Customers consist of dominant power contractors in their province and industrial category. Unlike large-scale aggregated electricity consumption, demand side loads show various load patterns. For this study, we select eight representative industrial categories among KSIC with five randomly-selected customers per category. Figure 6 shows the box and whiskers plot of the normalized consumptions of the eight selected categories. Each customer is trained with the local weather parameters depending on his/her location; Figure 7 shows an example of five weather parameters during three years for the training and testing period. Note that because Korea is a small country and located in a mid-latitude temperate climate zone, the weather parameters are roughly similar to Figure 7 across customers.

The representative load pattern per category can be inferred by the median, which is drawn as the red line in a box. For example, Figure 6a has a typical load patten of a rise during daytime, except lunch time, and a decrease at night. We also use weather data to forecast load profiles. From the regional weather database of the National Climate Data Service System (NCDSS), we select five parameters after examining the correlation analysis result. Daily average temperature, humidity, solar radiations, cloud cover and wind speed are used as input data.

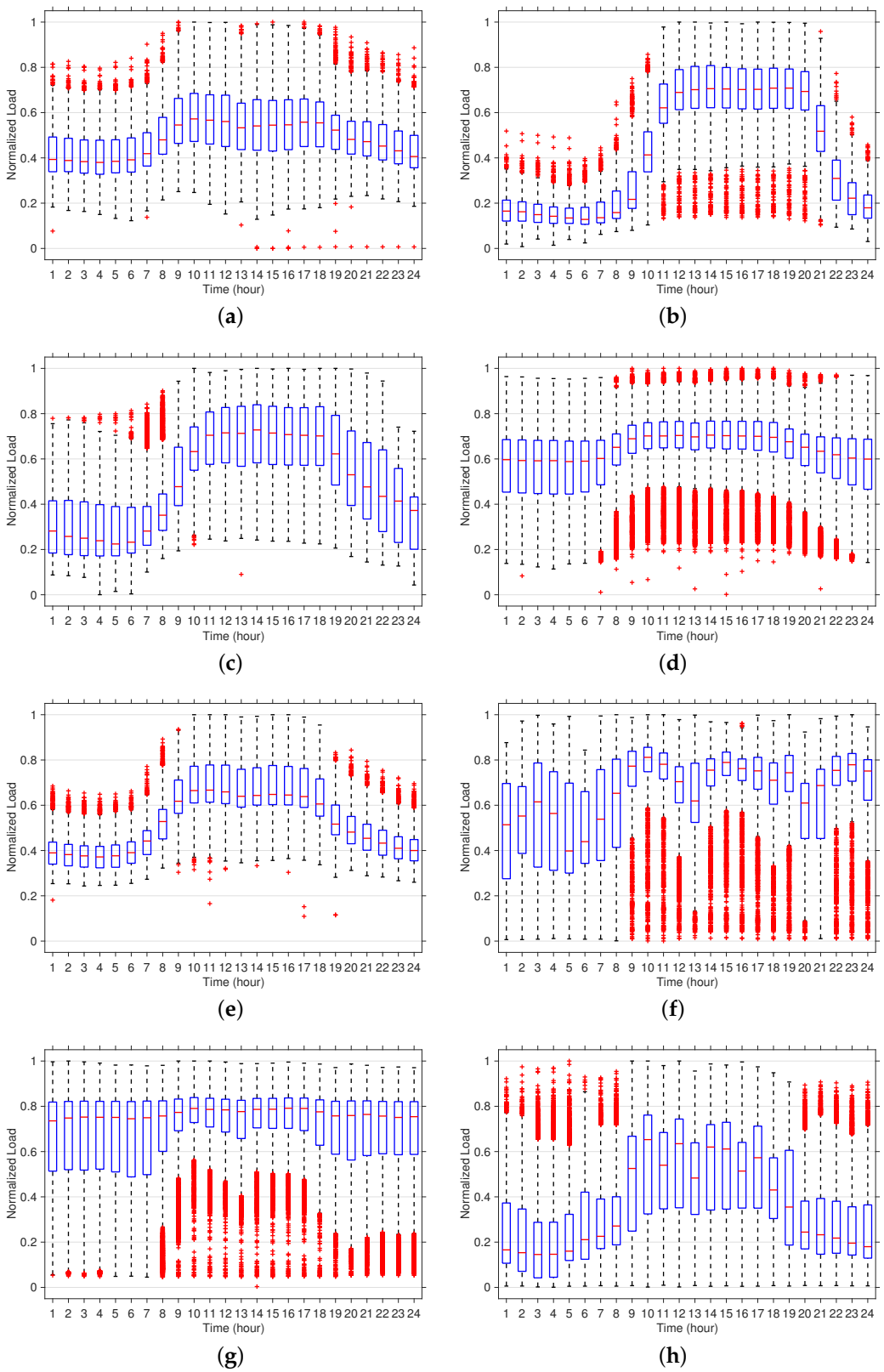

Figure 6. Box plot of normalized daily electricity consumptions: (**a**) Public administration; (**b**) Retail business; (**c**) R&D services; (**d**) Networking business; (**e**) Healthcare; (**f**) Vehicle and trailer manufacturing industry; (**g**) Electronic component and computer manufacturing industry; (**h**) Other manufacturing industries.

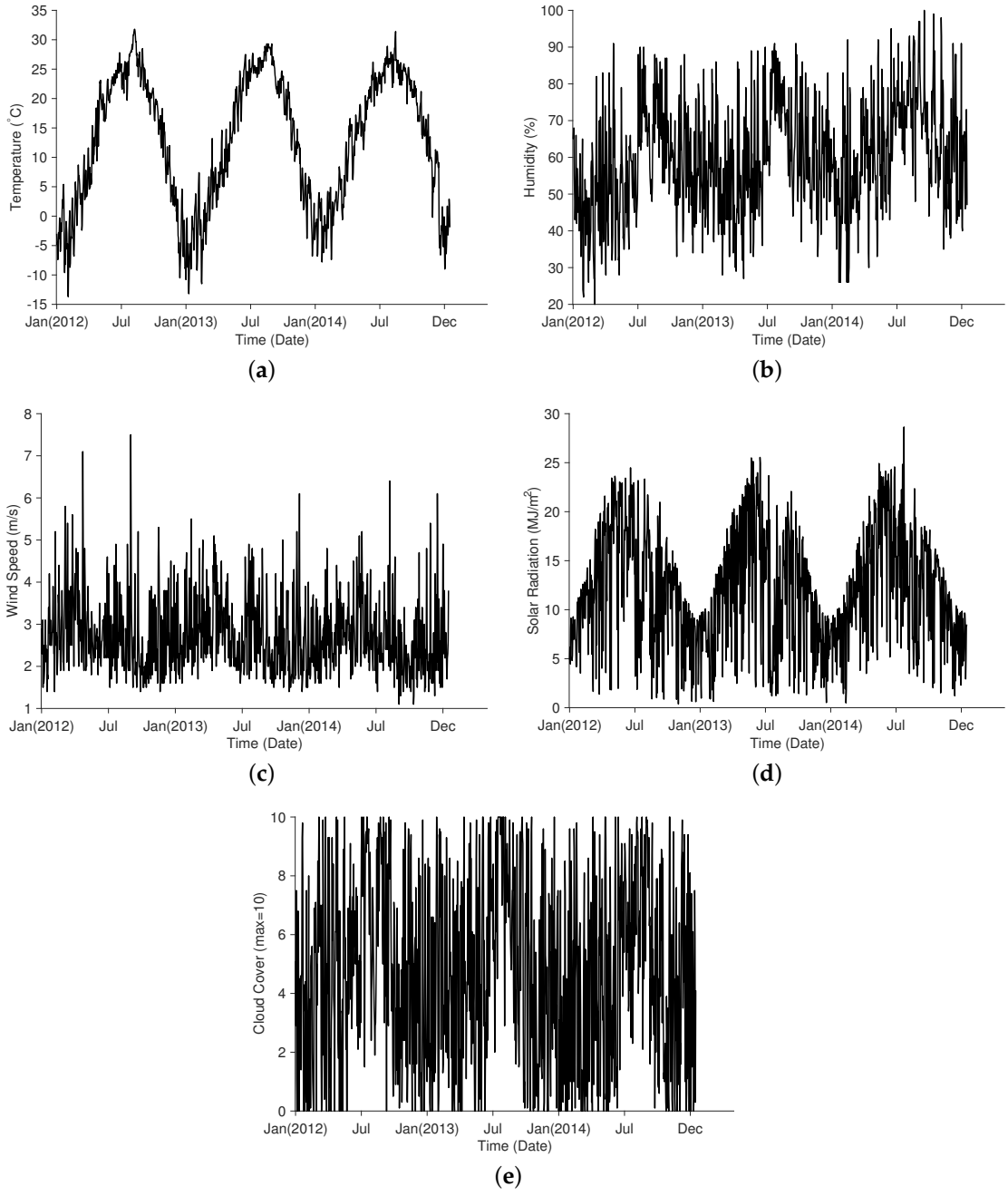

Figure 7. Weather variation in Korea: (**a**) Average temperature; (**b**) Humidity; (**c**) Wind speed; (**d**) Solar radiation; (**e**) Cloud cover.

Next, load and weather data go through the data preprocessing step. Data preprocessing contains data cleansing, normalization and structure change. We find that load measurements have some defective data, such as missing values and zero measured loads. Defects are replaced with the average of previous loads. After data cleansing, all load and weather data are normalized with the maximum value. If training data have large values, the weight matrix needs to be extremely small to make the weighted sum within the appropriate input range of the sigmoid activation function described in Figure 4. However, small weights make it hard to use the BP algorithm, so we normalize the database first and denormalize the forecasting result to obtain valid load predictions.

The third part of preprocessing is restructuring the given database into training data and test data. The training data contain input variables of the neural network and the desired output, i.e., labels to

calculate prediction errors. For example, the label of daily load forecasting model is a 24-h normalized load profile. Input variables could be any parameters that affect electricity consumption. The input data consist of past electricity consumptions, weather parameters for the target date, season, month and date indicators. With a time window index n that determines the period of previous electricity consumption for prediction, the composition of training data is described in Table 1.

In the experiments, we focus on forecasting load profiles of business days. Thus, from Table 1, weekend data are excluded from the input. In addition, we select the number of days in a window, $n = 5$, to use past load data up to the same weekday of the previous week to reflect weekly seasonality. Thus, a total of 133 inputs is used for forecasting.

Test data have an identical structure to the training data. This is used for evaluating the accuracy of the proposed forecasting model. Practical predictions are easily obtained by replacing test data with past observations for target forecasting dates. We construct validation data from the training data and exploit validation errors as indicators for selecting proper parameters.

Table 1. Training data configuration.

Parameter	Configuration	Number of Parameters
$y_1, y_2, \ldots, y_{24}$	label, 24-h normalized load profile	24
$x_1, x_2, \ldots, x_{24n}$	past n normalized load profile	$24 \cdot n$
$x_{24 \cdot n+1}, \ldots, x_{26 \cdot n}$	date information of past n days (day of the week, weekday indicator)	$2 \cdot n$
$x_{26 \cdot n+1}, \ldots, x_{26 \cdot n+5}$	weather parameters correspond to label (temperature, humidity, wind speed, solar radiation, cloud cover)	5
$x_{26 \cdot n+6}, \ldots, x_{26 \cdot n+9}$	date information of label (season, month, day of the week, weekday indicator)	4

3.2. Training and Forecasting

After obtaining test and training data by the data processing module, training data come into the DNN training module in the lower left part of Figure 5. The DNN training module is the main part of this framework, and the implementation details are given in Section 4. We first determine the structure of DNN, e.g., the numbers of hidden layers and neurons. The numbers of neurons in input and output layers are fixed to the dimension of the training data and the period of the forecast load profile, respectively. Each neuron in the output layer produces a prediction value for the specific hour. Once the DNN is created, we select a way to train the DNN (RBM pre-training or ReLU), and the DNN learns nonlinear relations between load profiles and past observations.

The last part of the framework is the forecasting module. We trained our DNN with training data; the DNN structure is fixed, and then, we used the trained DNN to predict the testing day. For the three weeks of test days without the weekend, to predict the day d, the input is taken from the days $d - 5$ to $d - 1$. Then, d changes from $d = 1$ to $d = 15$. Thus, our model predicts the daily load profile one by one like a sliding window. Of course, our model can be modified to be updated on the run using batch retraining; periodically, DNN is retrained using the same procedure when new data are available for training, e.g., once every few weeks.

DNN provides 24-h loads of the target forecasting date according to the input parameters listed in Table 1. Since neurons are fully connected, all hours of previous n days (in the experiment, five days) affect each output value. Thus, our model learns the nonlinear relations of inter-day loads and intra-day loads during training and forecasts the next whole day based on trained relations. Forecasting model accuracy is evaluated by test data. The MAPE and RRMSE are used for the error measure.

4. Implementation of DNNs

We construct DNNs using R with the "darch" package [31] and train with various customer's electricity consumption data.

4.1. Size of Data

There could be some issues of using DNNs as a forecasting model if the size of load data is limited. DNNs show remarkable performance in image recognition and speech recognition with a huge training database. For example, the ImageNet large-scale visual recognition challenge in 2014 has more than four hundred thousand training images. Using pre-training is not necessary with large training data. However, the volume of load forecasting data is limited because gathering demand side load data just started recently. We could obtain roughly 750 daily load profiles per customer for three years, except holidays. There are a few possible ways to increase the size of training data, such as adding distorted data [32], which was used in image classification, or using other customer's training data having a similar load profile. However, there is no guarantee that training with other customer's data helps with better forecasting. In contrast, the number of parameters in DNN may outnumber training data. This size-related problem could lead to overfitting that memorizes the whole training data rather than learning. Therefore, we need to consider the proper DNN structure and overfitting.

4.2. Structure of DNNs

The structure of DNNs is mainly determined by the numbers of layers and neurons. The numbers of input neurons and output neurons are automatically fixed by the training data and forecasting period. In [19], the authors used a heuristic method to choose the number of hidden neurons. They gradually increased the number of neurons and compared the errors. The heuristics may work with the typical three-layer MLP because there are only two possible actions: increase or decrease. However, in the case of DNNs, it is not feasible to test all possible combinations of layers and neurons, since we need to consider not only the width of hidden layers, but also the depth. Therefore, we determine the structure by trial and error with adding or subtracting 50 neurons for each layer and, finally, select four hidden layers and 150 neurons per each layer. Hidden layers have either a sigmoid unit for RBM pre-training or ReLU, and the output layer has linear units to construct load prediction. The proposed DNN structure is described in Figure 8. In the experiment, we train DNNs with a mini-batch and resilient backpropagation [33]. In addition, we train DNNs without non-business days to focus on the load profile of business days, and a time window of five is used.

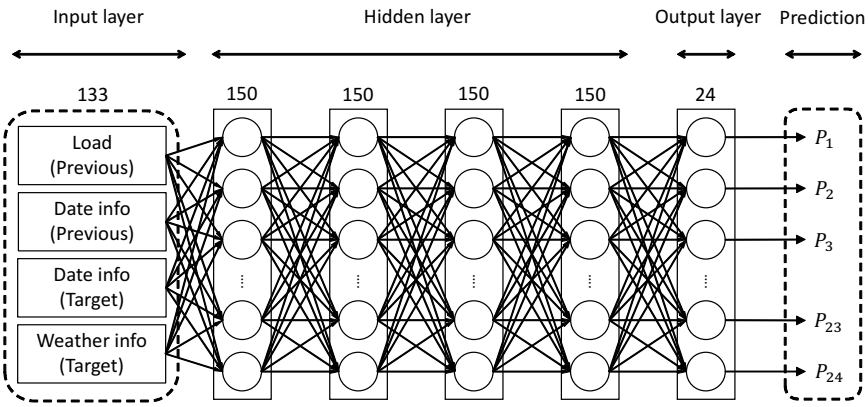

Figure 8. The structure of DNN.

4.3. Resolving Overfitting

Overfitting can be recognized by observing a learning curve that describes the error rate according to the number of epochs. An epoch is a period of time that the network learns all training data once. When overfitting occurs, there exists a certain point where the test error starts to increase while the training error still decrease. This means that the model memorizes the given training data, but it cannot predict well in a real situation. Thus, too much training with the complex forecasting model leads to bad generalization. There are several methods to prevent overfitting, such as dropout [34] or early stopping. However, when we observe the learning curve of DNN forecasting models, we did not observe overfitting to occur; the test error also gradually decreases as the training error does so in both the pre-training and ReLU models. An example of the customer's learning curve is depicted in Figure 9. Since customers have different learning curves, we apply *k*-fold cross-validation to determine stopping criterion. From this study, the size of the electricity consumption data seems less critical, but it may require further verification with different network structures and/or when more data are available later.

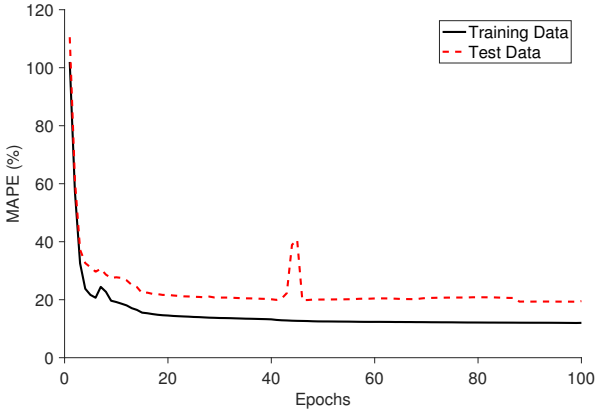

Figure 9. An example of the customer learning curve.

Another interesting point is that the training error of the RBM model mostly drops first, followed by the DNN model without pre-training and the ReLU model as shown in Figure 10. This implies that pre-training with RBM expedites the speed of convergence because of starting from better initial weights. However, the errors of all three models eventually approach similar values as the training continues.

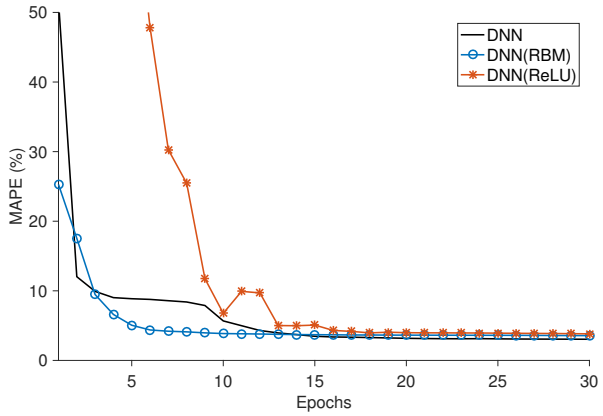

Figure 10. Training error comparison among DNN models.

5. Experimental Results with Case Studies

In this section, experimental results of STLF by DNNs are given. MAPE and RRMSE are used to measure the accuracy of daily predictions, and daily MAPE and RRMSE are defined as:

$$MAPE_{daily} = \frac{1}{T}\sum_{i=1}^{T}\frac{|R_i - F_i|}{R_i} \tag{10}$$

$$RRMSE_{daily} = \sqrt{\frac{1}{T}\sum_{i=1}^{T}(R_i - F_i)^2}/R_{avg} \tag{11}$$

where R_i and F_i represent real and forecast consumption at hour i, respectively, and T is the number of time units in a day. R_{avg} refers to the average of daily values. Experiments are focused on forecasting load profiles of working days. Forecasting results of DNNs are compared with SNN, seasonal ARIMA (we denote ARIMA for representing seasonal ARIMA for the rest of the results) and the DSHW model in order to verify DNNs as a forecasting model of individual customers. Instead of double seasonal ARIMA, we used DSHW, which shows better forecasting accuracy than double seasonal ARIMA in [8]. Time series models (seasonal ARIMA and DSHW) are created for each daily load profile prediction with the past eight weeks [7].

5.1. Case 1: Single Load Type

We first present the result about one selected customer and analyze the prediction results of four seasons. The customer who is located in Busan belongs to the networking business category and has typical daily load patterns with seasonal variation. Figure 11 shows the customer's electricity consumptions of 2012 to 2014. There exists a significant load increase in summer and a slight increase in winter during the daytime. Maximum, average and minimum hourly electricity consumptions are 2667, 1642 and 641 kWh, respectively. We predict load profiles of three weeks in the middle of each season. Seasonal predictions are described in Figure 12. The daily load profile is zoomed in the box. ARIMA with daily seasonality shows the worst predictions; their load shapes are somewhat similar to real loads, but ARIMA could not reflect the recent trend. The DSHW model considers both daily and weekly seasonality. It shows better accuracy than the ARIMA model. However, DSHW sometimes shows unusual patterns, like in Figure 12b,c, which seem to be the imprint of past abnormal consumptions. In contrast, the predictions by the two proposed DNN models are more stable than the ARIMA and DSHW models. When there exists a gap between real and DNN predictions, other models exhibit wider gaps. However, it is hard to find the opposite cases.

In all four seasons, DNN predictions follow the recent trend and real load patterns well. The average errors of all seasons are shown in Table 2. As can be seen, two DNN forecasting models show better accuracy than the SNN, DSHW and ARIMA models overall. DNN with pre-training shows the best accuracy on average by 3.2% and 4.1% in terms of MAPE and RRMSE. MAPE decreases by 27% compared to SNN and 12% compared to DSHW. Due to the average gap of predictions by ARIMA, the MAPE of DNN is much lower than ARIMA, i.e., a 69% decrease. Furthermore, DNN models predict well without being affected by seasonal variation.

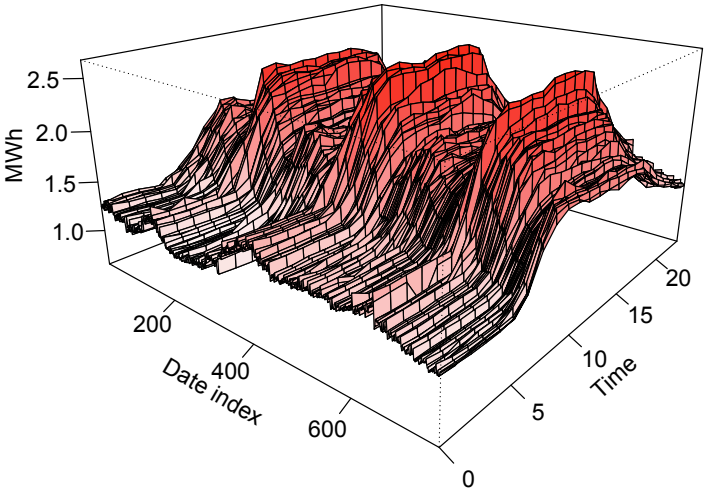

Figure 11. An example of a customer's daily and yearly load pattern.

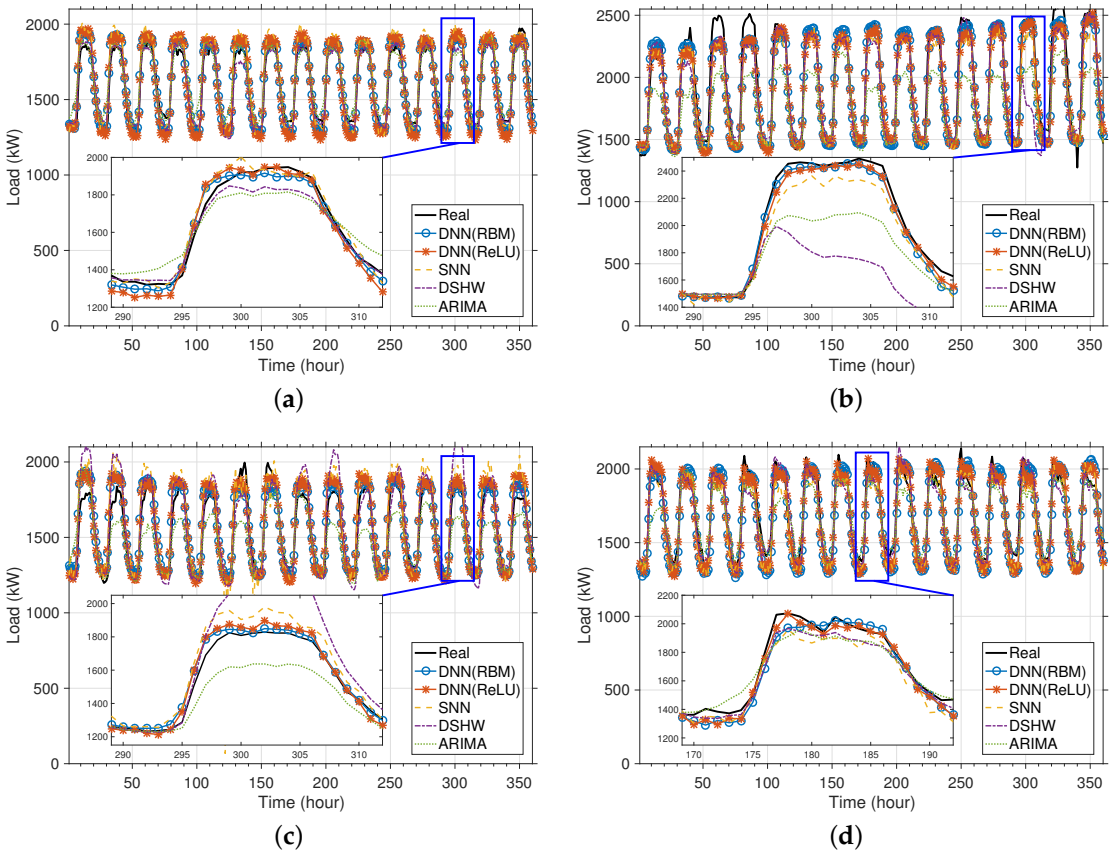

Figure 12. Seasonal prediction results: (**a**) spring; (**b**) summer; (**c**) fall; (**d**) winter. SNN, shallow neural network; DSHW, double seasonal Holt–Winters.

Table 2. Seasonal error table of the selected customer.

Season	MAPE (%)					RRMSE (%)				
	RBM	**ReLU**	**SNN**	**DSHW**	**ARIMA**	**RBM**	**ReLU**	**SNN**	**DSHW**	**ARIMA**
Spring [1]	2.66	3.77	3.44	2.01	9.45	3.18	4.24	4.30	2.54	13.8
Summer [2]	2.88	3.36	4.20	4.50	5.86	4.12	4.68	5.92	7.93	7.16
Fall [3]	3.49	3.84	6.01	5.04	13.43	4.52	5.00	8.48	7.34	26.55
Winter [4]	3.76	2.82	3.78	3.04	11.16	4.58	3.50	4.72	4.06	15.36
Average	3.20	3.45	4.36	3.65	9.97	4.10	4.36	5.86	5.47	15.61

[1] 7∼25.04.2014, [2] 7∼25.07.2014, [3] 13∼31.10.2014, [4] 6∼24.01.2014.

5.2. Case 2: Various Load Types

To identify the generalized performance of the proposed DNNs, we apply the DNNs to a large set of load databases where there are eight industrial categories (public administration, retail business, R&D services, networking business, healthcare, vehicle and trailer manufacturing industry, electronic component and computer manufacturing industry and other manufacturing industries), and five customers are randomly selected in each category. We target forecasting daily load profiles of 10 weeks from 13 September to 19 December 2014. Considering the highly inaccurate result of ARIMA in Case 1 and the burden of model selection, we exclude ARIMA in Case 2.

The error results of 40 customers are shown in Table 3. MAPE and RRMSE are used for error measurements, and abnormal days with more than 100% MAPE are not counted in the result. As described in the table, both DNN models show accurate forecasting result even though SNN or DSHW sometimes exhibits better result than the DNNs. Overall, the average MAPE of DNN with ReLU decreases by 17% and 9% compared to the SNN and DSHW, respectively. The decrease in the RRMSE is more drastic: 22% and 29% lower than that of the SNN and DSHW. For further analysis, we compare the daily errors of whole customers. Figure 13a,b shows the cumulative distribution function (CDF) of daily RRMSEs and MAPEs. In terms of RRMSE and MAPE CDFs, the two DNN models are always better than the SNN model. Since DNNs have more computational power, this result may not be surprising. Compared with DSHW, the two DNN models and DSHW exhibit similar performance in a small error region. However, there exists a crossover point beyond which DNNs surpass the DSHW model. Roughly, when errors are under 3%, the performance of DSHW is better than that of DNNs, but after that, DNNs begin to be better than DSHW; DNNs predict more reliably, even if loads are uncertain with high prediction errors. These results imply the robustness of DNN models. Between two DNN models, there is no significant difference in accuracy. Since the RBM pre-training model needs more computational power and time, the DNN with ReLU would be a better choice for customer load forecasting.

Table 3. Error table of various customers in 8 industries.

Industry	User ID	Peak Load (kW)	MAPE (%)				RRMSE (%)			
			RBM	ReLU	SNN	DSHW	RBM	ReLU	SNN	DSHW
Public administration	1	2784	12.53	12.82	11.93	15.82	15.68	15.94	14.98	21.90
	2	6487	8.64	8.69	8.39	11.02	11.18	10.92	10.62	15.66
	3	2012	4.72	5.06	4.52	3.82	6.36	6.03	6.13	6.00
	4	3890	14.79	16.25	17.46	19.82	20.14	21.96	21.42	30.26
	5	8743	7.87	7.54	8.54	8.26	9.50	9.19	12.74	11.26
		Average	9.71	10.07	10.17	11.75	12.57	12.81	13.18	17.02
Retail business	6	3358	11.27	8.57	9.07	6.64	8.92	9.11	9.40	8.81
	7	7646	10.52	10.12	12.26	9.84	14.56	13.95	21.13	16.74
	8	5194	9.12	8.87	13.50	12.24	11.10	10.73	17.15	25.46
	9	7268	10.26	11.36	16.99	12.88	12.37	13.29	16.29	22.23
	10	7214	6.45	7.47	8.14	7.82	9.74	10.19	10.54	14.35
		Average	9.52	9.28	11.99	9.88	11.34	11.45	14.90	17.52
R&D services	11	8845	3.53	3.32	4.48	3.43	4.90	4.21	6.23	4.83
	12	2740	11.13	7.53	14.47	6.29	12.74	8.94	24.86	9.55
	13	8221	5.72	4.42	5.41	5.46	5.76	4.33	5.40	5.73
	14	1276	6.64	6.14	9.21	5.70	9.42	8.04	32.71	8.65
	15	891	10.03	9.38	10.31	9.80	12.71	13.56	14.76	16.21
		Average	7.41	6.16	8.78	6.14	9.11	7.82	16.79	8.99
Networking business	16	878	2.18	2.14	2.65	2.34	3.91	3.84	4.17	4.12
	17	1481	2.51	2.04	5.72	2.10	3.37	2.76	8.12	3.65
	18	2667	2.76	3.20	4.61	3.42	3.72	4.19	6.43	5.50
	19	477	7.51	7.64	9.33	6.22	10.14	10.37	11.89	8.23
	20	10,022	1.78	1.38	6.88	0.68	2.331	1.95	15.81	1.05
		Average	3.35	3.28	5.84	2.95	4.67	4.62	9.28	4.51
Healthcare	21	2193	4.69	5.01	7.76	6.00	6.17	6.54	20.69	8.53
	22	6937	3.60	3.82	4.54	5.01	4.97	5.21	5.97	6.67
	23	2603	3.81	5.45	5.70	6.16	5.18	6.44	8.80	9.05
	24	4110	3.21	2.95	4.40	3.76	4.36	3.66	7.09	5.33
	25	2932	5.48	6.34	6.24	7.82	7.32	7.96	7.56	11.37
		Average	4.16	4.71	5.73	5.75	5.60	5.96	10.02	8.19
Vehicle and trailer manufacturing industry	26	23,138	12.44	14.51	12.71	9.67	14.47	15.13	12.80	16.84
	27	4132	14.09	17.58	20.10	18.40	12.81	12.66	14.15	24.55
	28	130,133	7.94	10.16	9.80	16.74	10.95	13.19	12.93	28.18
	29	38,674	14.65	12.86	15.52	16.74	17.35	15.32	18.78	27.35
	30	128,257	3.35	4.12	3.90	6.96	4.82	5.47	5.07	14.41
		Average	10.49	11.85	12.41	13.70	12.08	12.35	12.75	22.27
Electronic component and computer manufacturing industry	31	28,690	5.73	5.76	6.95	4.78	7.98	7.87	9.44	6.86
	32	602	18.37	18.79	22.05	16.12	16.70	21.70	20.64	31.63
	33	112,630	2.72	1.19	2.17	1.11	3.32	1.90	2.88	1.91
	34	8607	3.33	1.84	2.76	1.54	3.88	2.24	4.75	2.07
	35	299,880	1.58	1.43	2.01	1.40	2.07	2.00	2.57	1.89
		Average	6.35	5.80	7.19	4.99	6.79	7.14	8.06	8.87
Other manufacturing industries	36	5721	12.89	13.81	15.50	19.10	13.52	14.41	16.36	26.17
	37	832	33.14	33.65	45.21	40.73	24.43	25.25	29.11	53.15
	38	8202	21.48	21.15	24.01	25.94	28.34	30.76	31.54	44.55
	39	1164	15.60	16.19	16.31	12.00	31.79	32.38	31.93	20.61
	40	1948	15.58	13.51	13.95	15.79	16.07	14.13	14.20	20.23
		Average	19.74	19.66	23.00	22.71	22.83	23.39	24.63	32.94
Total	40 users	Average	8.84	8.85	10.64	9.73	10.62	10.69	13.70	15.04

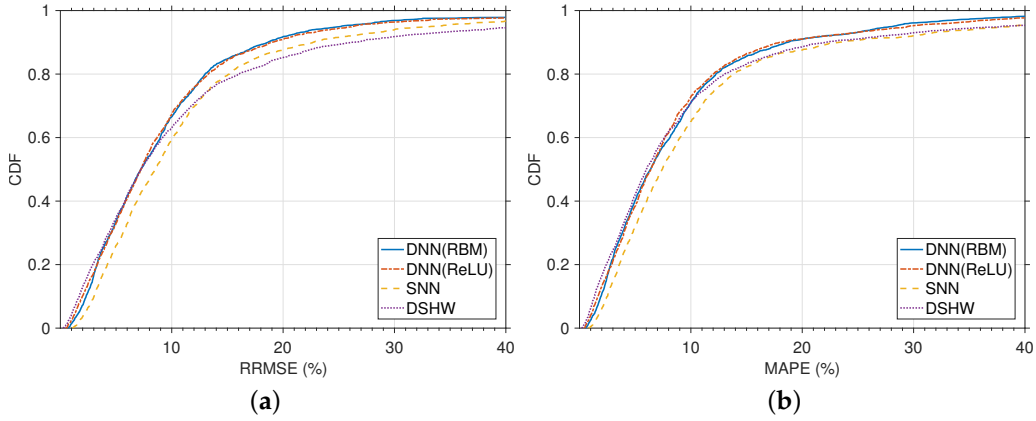

Figure 13. CDF of daily errors: (**a**) RRMSE; (**b**) MAPE.

5.3. Case 3: Aggregated Load

From Case 2, each customer's errors are around 10% on average. One may think the errors are higher than the results from the literature [9,23]. The difference is that previous research is focused on large-scale load consumption, such as at the level of substations or power systems. It should be noted that individual-level load forecasting shows higher errors up to 30% [35]. By aggregating customer's electricity consumption, error drops significantly. Figure 14 represents three years' aggregated loads with more than 500 customers with high usage. Load consumption varies in GW units, and weekly seasonality can be observed. Figure 15 describes the forecasting results of the five models. With aggregation, we obtain the average MAPE of 2.27% by DNN with pre-training and 2.19% by DNN with ReLU during the same periods of Case 2. In contrast, the MAPEs of SNN, DSWH and ARIMA are 2.55%, 2.98%, and 3.29% on average, respectively. The average MAPE and RRMSE are described in Table 4. The result confirms that aggregation helps to reduce prediction errors by reducing inherent variability [35], and the proposed DNN models also work well with large-scale load consumptions.

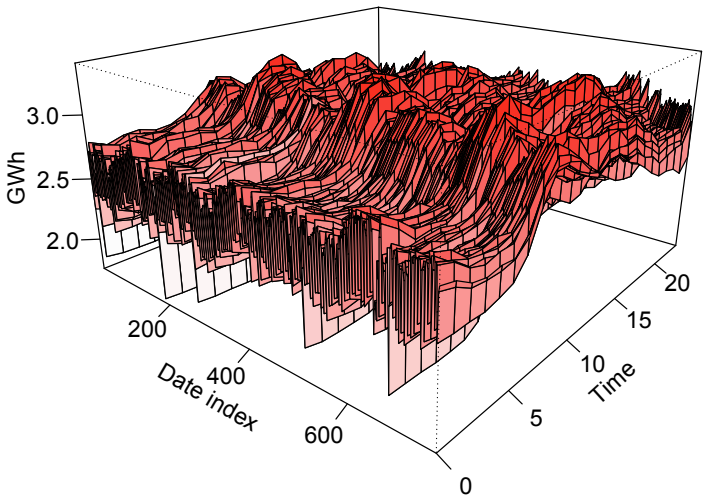

Figure 14. An example of the aggregated daily and yearly load pattern.

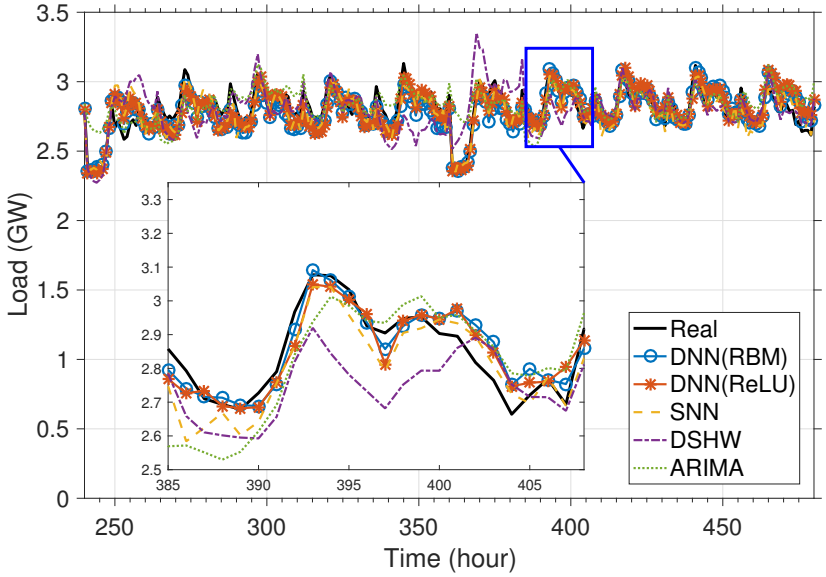

Figure 15. Predictions of the aggregated load.

Table 4. Error table of the aggregated load.

	MAPE (%)					RRMSE (%)				
	RBM	**ReLU**	**SNN**	**DSHW**	**ARIMA**	**RBM**	**ReLU**	**SNN**	**DSHW**	**ARIMA**
Average	2.27	2.19	2.98	2.55	3.29	2.91	2.76	3.70	3.35	4.21

6. Conclusions

In this paper, we applied deep learning to demand side STLF and proposed a DNN-based STLF framework. The proposed framework contains data processing, DNN training and DNN forecasting step and predicts the 24-h load pattern day-ahead based on weather, date and past electricity consumptions. Specifically, we compared two different DNNs; pre-training RBM and using ReLU without pre-training. Then, we investigated the problems of training DNNs, e.g., overfitting. Our extensive experiments show that the two proposed approaches, DNN with pre-training and DNN with ReLU, can be trained well with up to three years of customer load data without overfitting. Finally, we applied the proposed framework to a large-scale set of customers' data. DNN models were trained for 40 dominant industrial customers and exhibited better performance compared to other forecasting models, such as ARIMA, SNN and DSHW. The numerical results indicate that the DNN-based STLF model has less MAPE and RRMSE by up to 17% and 22% than SNN and 9% and 29% than DSHW. We found that DNN with ReLU is easy to train and shows accurate and robust predictions, and thus, it can be firstly applied for the demand side STLF model.

The results of this paper can be further extended in several directions. We are currently gathering nation-wide data and, thus, expect to train the DNN better when big data are available. Then, it would be possible to explore a better DNN structure, e.g., considering the convolutional neural network as a time series forecasting model. In addition, analyzing the conditions (e.g., load type, load pattern, date, etc.) when DNN is accurate would be helpful to select a proper forecasting model for individual customers.

Although our extensive experimental results may not necessarily guarantee that DNNs surpass all other forecasting models, our study is meaningful in the sense that it sheds light on the study of DNN for STLF.

Acknowledgments: This research was supported by the Korea Electric Power Corporation (KEPCO) and the Korea Electrical Engineering & Science Research Institute. (Grant Number R14XA02-50), as well as by the KEPCO (CX72166553-R16DA17) of the Republic of Korea.

Author Contributions: Seunghyoung Ryu designed the algorithm, performed the simulations, and prepared the manuscript as the first author. Jaekoo Noh assisted the project and managed to obtain the demand side data from KEPCO. Hongseok Kim led the project and research. All authors discussed the simulation results and approved the publication.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ipakchi, A.; Albuyeh, F. Grid of the future. *IEEE Power Energy Mag.* **2009**, *7*, 52–62.
2. Farhangi, H. The path of the smart grid. *IEEE Power Energy Mag.* **2010**, *8*, 18–28.
3. Feinberg, E.A.; Genethliou, D. Load forecasting. In *Applied Mathematics for Restructured Electric Power Systems*; Springer: Berlin, Germany, 2005; pp. 269–285.
4. Albadi, M.H.; El-Saadany, E. A summary of demand response in electricity markets. *Electr. Power Syst. Res.* **2008**, *78*, 1989–1996.
5. Park, S.; Ryu, S.; Choi, Y.; Kim, J.; Kim, H. Data-driven baseline estimation of residential buildings for demand response. *Energies* **2015**, *8*, 10239–10259.
6. Hagan, M.T.; Behr, S.M. The time series approach to short term load forecasting. *IEEE Trans. Power Syst.* **1987**, *2*, 785–791.

7. Taylor, J.W. Short-term electricity demand forecasting using double seasonal exponential smoothing. *J. Oper. Res. Soc.* **2003**, *54*, 799–805.
8. Taylor, J.W.; de Menezes, L.M.; McSharry, P.E. A comparison of univariate methods for forecasting electricity demand up to a day ahead. *Int. J. Forecast.* **2006**, *22*, 1–16.
9. Park, D.C.; El-Sharkawi, M.; Marks, R.; Atlas, L.; Damborg, M. Electric load forecasting using an artificial neural network. *IEEE Trans. Power Syst.* **1991**, *6*, 442–449.
10. Hernandez, L.; Baladrón, C.; Aguiar, J.M.; Carro, B.; Sanchez-Esguevillas, A.J.; Lloret, J. Short-term load forecasting for microgrids based on artificial neural networks. *Energies* **2013**, *6*, 1385–1408.
11. Hippert, H.S.; Pedreira, C.E.; Souza, R.C. Neural networks for short-term load forecasting: A review and evaluation. *IEEE Trans. Power Syst.* **2001**, *16*, 44–55.
12. Bakirtzis, A.G.; Petridis, V.; Kiartzis, S.; Alexiadis, M.C. A neural network short term load forecasting model for the Greek power system. *IEEE Trans. Power Syst.* **1996**, *11*, 858–863.
13. Lu, C.; Wu, H.T.; Vemuri, S. Neural network based short term load forecasting. *IEEE Trans. Power Syst.* **1993**, *8*, 336–342.
14. Rodrigues, F.; Cardeira, C.; Calado, J. The daily and hourly energy consumption and load forecasting using artificial neural network method: A case study using a set of 93 households in Portugal. *Energy Procedia* **2014**, *62*, 220–229.
15. Chen, H.; Cañizares, C.; Singh, A. ANN-based short-term load forecasting in electricity markets. In Proceedings of the 2001 IEEE Power Engineering Society Winter Meeting, Columbus, OH, USA, 28 January–1 February 2001; Volume 2, pp. 411–415.
16. Papadakis, S.; Theocharis, J.; Kiartzis, S.; Bakirtzis, A. A novel approach to short-term load forecasting using fuzzy neural networks. *IEEE Trans. Power Syst.* **1998**, *13*, 480–492.
17. Bashir, Z.; El-Hawary, M. Applying wavelets to short-term load forecasting using PSO-based neural networks. *IEEE Trans. Power Syst.* **2009**, *24*, 20–27.
18. Kodogiannis, V.S.; Amina, M.; Petrounias, I. A clustering-based fuzzy wavelet neural network model for short-term load forecasting. *Int. J. Neural Syst.* **2013**, *23*, doi:10.1142/S012906571350024X.
19. Shayeghi, H.; Shayanfar, H.; Azimi, G. Intelligent neural network based STLF. *Int. J. Comput. Syst. Sci. Eng.* **2009**, *4*, 17–27.
20. Fan, S.; Chen, L. Short-term load forecasting based on an adaptive hybrid method. *IEEE Trans. Power Syst.* **2006**, *21*, 392–401.
21. Park, S.; Ryu, S.; Choi, Y.; Kim, H. A framework for baseline load estimation in demand response: Data mining approach. In Proceedings of the 2014 IEEE International Conference on Smart Grid Communications (SmartGridComm), Venice, Italy, 3–6 November 2014; pp. 638–643.
22. Dalto, M.; Matusko, J.; Vasak, M. Deep neural networks for ultra-short-term wind forecasting. In Proceedings of the 2015 IEEE International Conference on Industrial Technology (ICIT), Seville, Spain, 17–19 March 2015; pp. 1657–1663.
23. He, W. Deep neural network based load forecast. *Comput. Model. New Technol.* **2014**, *18*, 258–262.
24. Qiu, X.; Zhang, L.; Ren, Y.; Suganthan, P.N.; Amaratunga, G. Ensemble deep learning for regression and time series forecasting. In Proceedings of the 2014 IEEE Symposium on Computational Intelligence in Ensemble Learning (CIEL), Orlando, FL, USA, 9–12 December 2014; pp. 21–26.
25. Hinton, G.E.; Salakhutdinov, R.R. Reducing the dimensionality of data with neural networks. *Science* **2006**, *313*, 504–507.
26. Leshno, M.; Lin, V.Y.; Pinkus, A.; Schocken, S. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Netw.* **1993**, *6*, 861–867.
27. Abu-Mostafa, Y.S.; Magdon-Ismail, M.; Lin, H.T. *Learning from Data*; AMLBook: Berlin, Germany, 2012.
28. Erhan, D.; Bengio, Y.; Courville, A.; Manzagol, P.A.; Vincent, P.; Bengio, S. Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.* **2010**, *11*, 625–660.
29. Hinton, G. *A Practical Guide to Training Restricted Boltzmann Machines*; Springer: Berlin, Germany, 2012; pp. 599–619.
30. Maas, A.L.; Hannun, A.Y.; Ng, A.Y. Rectifier nonlinearities improve neural network acoustic models. In Proceedings of the International Machine Learning Society, Atlanta, GA, USA, 16–21 June 2013; Volume 30, p. 1.

31. Drees, M. Implementierung und Analyse von Tiefen Architekturen in R. Master's Thesis, Fachhochschule Dortmund, Dortmund, Germany, 2013.
32. Simard, P.Y.; Steinkraus, D.; Platt, J.C. Best practices for convolutional neural networks applied to visual document analysis. In Proceedings of the 2003 7th International Conference on Document Analysis and Recognition (ICDAR), Edinburgh, UK, 3–6 August 2003.
33. Riedmiller, M.; Braun, H. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In Proceedings of the 1993 IEEE International Conference On Neural Networks, San Francisco, CA, USA, 28 March–1 April 1993; pp. 586–591.
34. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
35. Sevlian, R.; Rajagopal, R. Short term electricity load forecasting on varying levels of aggregation. *Statistics* **2014**, arXiv:1404.0058.