An Image Inpainting Approach to Short-term Load Forecasting

Yanzhu Liu, Shreya Dutta, Adams Wai Kin Kong and Chai Kiat Yeo

***Abstract*—In current power systems, electrical energy is generated whenever there is a demand for it. Therefore, load forecasting, which estimates the active load in advance, is imperative for power system planning and operations. Based on the time horizon, load forecasting is classified as very short-term (below one day), short-term (a day to two weeks), medium-term (two weeks to three years) and long-term (over three years). This paper focuses on the short-term forecasting. The complex multi-level seasonality of load series (e.g., the load in a given hour is not only dependent on load in the previous hour, but also on the previous day's load in the same hour, and on the previous week's load in the same day-of-the-week and hour) makes this task challenging, especially when the load data is represented in 1d numerical series. However, in multi-channel images, the patterns in spatial neighbourhood of one channel and the patterns in the neighbourhood along the channel dimension are able to be captured by 3d image processing operations. Hence, this study proposes to transform electrical load data from 1d series to 3d images and transform the problem from future series forecasting to missing patch inpainting. Furthermore, it proposes a recurrent neural network to model the temporal trends in the series by convolutional operations on the spatial neighbourhood in the images. One advantage of the proposed method is that it supports the load prediction for a new location based on multiple related load series nearby, benefiting from the properties that similar visual patterns can be shared between different images converted from different series. Experimental results demonstrate the effectiveness of the proposed method on the PJM and London smart meter benchmark and show the capability of inferring the future load from related series if there is a lack of history.**

***Index Terms*—Deep learning, short-term load forecasting, univariate time-series**

I. Introduction

ELECTRIC load forecasting plays a crucial role in power system planning and operations [1]. In terms of forecasting horizon, it can be classified into four categories [2]: very short-term (seconds to several hours), short-term (a day to two weeks), medium-term (weeks up to three years) and long-term (over three years). Very short-term load forecasting is used for power system transients and dynamics; medium-term load forecasting helps scheduling of fuel supplies and maintenance program and long-term forecasting is necessary for strategic planning of power networks [3]. Short-term load forecasting is very important for power system planning and operations, it provides the basis for power system planning to avoid over and under utilization of generating capacity and guides the power system operations such as automatic generation control and security assessment [4] [5] [6] [7]. Moreover, short-term electricity forecasting helps in optimizing energy usage in urban buildings, realizing energy-saving operations and improving deployment strategies to avoid electricity outages during peak times [8].

In terms of forecasting object, electric load forecasting can be classified either based on the load levels of power system, such as system load, distribution feeder load and building & residential load, or based on the area's population and living standard of people, such as domestic load, commercial load, industrial load and agriculture load. However, recent rapid development of AI research provides a load type-agnostic solution. AI models are designed to automatically learn the underlying pattern of load series from historical data, which highly depends on the data availability. Before enough historical load records are accumulated, most data-driven forecasting models are not applicable. If a model has powerful generalization capability, other load data with similar characteristics will be useful to supervise it. Generalization refers to extracting underlying information that is shared among multiple related load series and is suitable to apply on the target series.

Besides historical data, electric load is subject to various exogenous factors, such as weather conditions, calendar effect and types of consumers. *Multivariate models* in load forecasting integrate these variables for prediction. *Univariate models* are those that do not rely on any exogenous variables. It has been reported that for short-term forecasting, the accuracies of multivariate models are generally better than univariate models [2]. However, it cannot be guaranteed that exogenous data is always available and reliable. The proposed model in this paper is a univariate model that does not make use of any information beyond historical series.

The challenge of short-term load forecasting arises from the complex multi-level seasonality of load series. For example, the load in a given hour is not only dependent on the load in the previous hour, but also on the previous day's load in the same hour, and on the previous week's load in the same day-of-the-week and hour. From the data perspective, the only input of univariate load forecasting model is a 1d series of numbers. The models have to capture the multi-level seasonality from long look-back horizons along the 1d series. However, if the series numbers are properly mapped to pixels of an image, a week-long half-hourly series (i.e., 7*48 time stamps) corresponds to a tiny patch with 7*48 pixels. Therefore, longer pattern of series is able to be explored in local regions of

This work was done when Y. Liu worked at Nanyang Technological University. She is now with Institute for Infocomm Research, A-Star, Singapore 138632. (email: Liu_Yanzhu@i2r.a-star.edu.sg)

S. Dutta is with Energy Research Institute @ Nanyang Technological University (ERI@N), Singapore, 639798

A.W.K. Kong and C. K. Yeo are with School of Computer Science and Engineering, Nanyang Technological University, Singapore, 639798

Shreya: main suggestion is to emphasise that this algorithm can be adopted for short term load forecasting at any level of the power system and that it has been tested for both system and household data. Hence, irrespective of the objective of the forecast and therefore the type of electrical load, this technique can be utilized.

images. And under proper re-organization (e.g., Figure 1), the multi-level seasonality can be encoded in spatial patterns, which motivates the authors to use a computer vision approach to solve the short-term load forecasting problem.

In computer vision, image inpainting is a task to fill missing pixels in a damaged image. If the series are transformed into images and a section of the future values to be predicted is transformed as an unobserved patch, estimating future values in the load series, in fact, becomes similar to generating pixels in the missing area of the images. To adapt image inpainting models for load forecasting, this paper firstly encodes a series including historical energy meter values and a length of unknown future values to a multi-channel image, each channel of which is built from series values of one week period and the values to be predicted are kept as a missing patch in the image. Based on that, a recurrent neural network (RNN) integrated with 3D convolutions in the memory cells is proposed to generate the missing patches. It takes a τ-channel image with an unobserved patch in the last channel as input and produces a 1-channel image with the filled patch as output. The recurrent network models the sequential property between channels of images. The 3d convolution operations encapsulated in LSTM cell capture shift-invariant local patterns in spatial neighbourhoods of both intra and inter channels. Because the proposed network is learnt from instances converted from segments of load series, the training set can be constructed by segments of multiple historical series. For a newly installed sensor or a new building with little history, multiple load series of sensors located in nearby regions can be used as historical data to train the proposed network. If the few historical values of the target sensor are able to construct a testing instance, the forecasting can be queried from the network by inputting the testing instance. Due to similar visual characteristics can be shared between different images, a model capturing the visual patterns of an image can be applied to related images. This property provides the basis that the information learnt by the proposed model from related load series can be applied to the others.

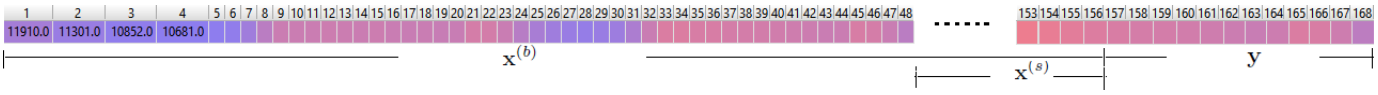
The rest of this paper is organized as follows. Section II reviews the literature of load forecasting. Section III describes the proposed method and its network architecture. Section IV reports the experimental results, and section V concludes the paper.

II. Related work

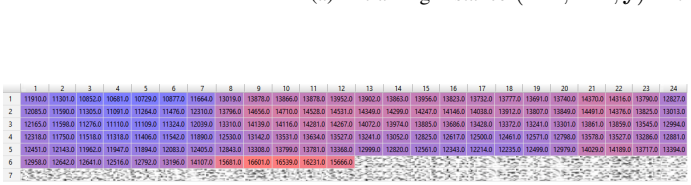
Numerous approaches and their variants are developed for short-term load forecasting [2]. Many of them are based on the following classical techniques. 1) *Regression analysis* estimates the relationship between load as the dependent variable and independent variables such as weather conditions and events. The prominent approaches include multiple linear regression [9], semi-parametric additive regression [10], and fuzzy regression [11]. 2) *Statistical time series* techniques, represented by exponential smoothing [12] and ARIMA (Auto-regressive Integrated Moving Average) [13], are univariate models which do not rely on any exogenous variables. ARIMA models can be generalized to include explanatory factors. 3) *Machine learning* techniques, represented by SVR (support vector regression) [14] and ANNs (artificial neural networks) [5], learn the nonlinear mapping from load and exogenous variables to future load values in the supervised fashion. Generally speaking, regression analysis and machine learning based models are multivariate models, while statistical time series models are univariate. Because electric load is highly dependent on weather conditions, multivariate models incorporating the explanatory variables have potential advantages in terms of accuracy over univariate models [2] [15]. In contrast, due to no additional information requirements, univariate models can be widely and fairly applied on more datasets with only historical load series. It is worth mentioning that Hong et al. [2] provided a comprehensive review on them. Readers can refer to it for more details.

In recent years, deep learning technique has achieved great success in wide areas. Several deep neural network approaches have also been developed for short-term load forecasting. Singh et al. [17] provided a very recent review. From the network architecture perspective, three predominant networks for load forecasting are Stacked Auto-encoder [18], Recurrent Neural Network (RNN) [19] and Convolution Neural Network (CNN) [20]. From the learning strategy perspective, two different supervised manners are applied: end-to-end training, which directly learns from raw load series and outputs the predicted series in the future horizon by the neural networks [19] [21]; and hybrid training, where neural layers work as automatic feature extractor capturing the nonlinear abstract features from load series linked with another forecaster (e.g., SVR, fuzzy regressor or another type of neural network) to produce the estimated series based on the extracted features [22] [23]. From the data format perspective, the input of the neural networks can be 1d univariate load series, 2d transformed image-like grids, and 2d original multivariate series which can be presented as a matrix in which rows are for the time dimension and columns are for the load and exogenous variables. The output of the neural networks is either 1d future load series, or one real number for the load value in the immediate next time stamp mainly used in multi-step forecasting.

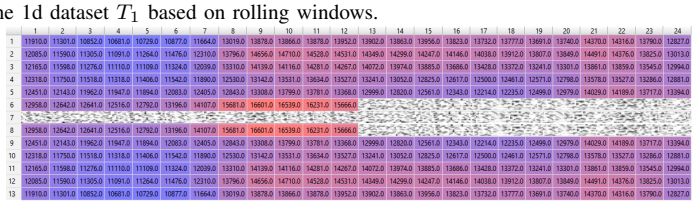
Generally speaking, CNNs consist of convolutional layers performing the convolutional operation on input data and fully-connected layers computing the final output. Because CNNs are powerful to capture local correlation and extract abstract features, but not specially designed for sequential data, most of the CNNs based load forecasting approaches only keep convolutional layers for feature extraction and replace fully-connected layers by other forecasters. As CNNs are developed for grid data, if univariate load series data are viewed as 1d grids, CNNs can be used for short-term load forecasting with 1d convolution [24]. Plenty of CNNs based approaches have been proposed to transform load series into images and perform 2d convolutional operation on them. The outputs of convolutional layers are the extracted features, which are passed into a predictor such as multilayer neural network, K-means cluster and so on [23] [25] [26] to get the final estimation. As recurrent networks are designed for processing sequential data, LSTM has been used as the forecaster inte-



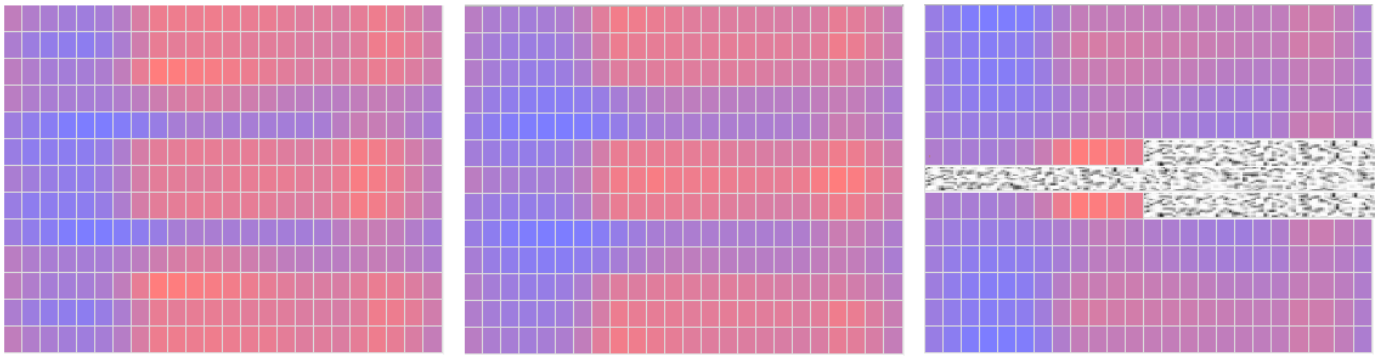
(a) A training instance $(\mathbf{x}^{(b)}, \mathbf{x}^{(s)}, \mathbf{y})$ in the 1d dataset T_1 based on rolling windows.



(b) A transformed image $\mathbf{x}'$ with $h = 7, m = 24, H = 36, d = 1$.



(c) A transformed image $\dot{\mathbf{x}}$ with size $13 \times 24 \times 1$.



(d) An example image with size $13 \times 24 \times 3$ transformed from hourly electric consumption dataset [16] on three weeks period

Fig. 1: Examples of the transformed dataset set.

grated with the 2d convolutional layers [27], [28] [29] [30].

The proposed neural network in this paper is a LSTM equipped with 3d convolution, but it is very different from the convolution and RNN hybrid approaches in the literature. It takes τ-channel images ($\tau > 3$) as inputs and produces 1-channel images as outputs, which is trained end-to-end. The recurrent component models the sequential property among the channels of images, not among the series values at time stamps as traditional RNN based forecasting models. The Eidetic-3d [31] cell is used as the LSTM cell. The output of an Eidetic-3d cell at a certain layer for a time is computed from cell input, hidden state, cell memory and global memory. All the four inputs are filtered with 3d convolutional operations before being passed into the gates. Therefore, 3d convolution filters are encapsulated inside the LSTM cell, but not the separate convolutional layers installed on top of the whole RNN as existing load forecasting approaches.

The proposed approach is motivated by image inpainting idea. In computer vision, image inpainting or image completion aims to fill missing region with visually realistic and semantically reasonable content. With power of rich representation capability and being able to automatically extract abstract knowledge, deep learning techniques reduce barriers of interdisciplinary applications. Under reconstruction of electric load series to images with unknown future series as a missing patch, inpainting models are possible to provide accurate performance for load forecasting. In literature, numerous deep inpainting models have been proposed [32][33] [34] [35]. An extensive review about them is offered by Zhen et al. in [36]. Deep neural networks (DNNs) based approaches are the state-of-the-art [37] [38] for image inpainting. Yeh et al. [39] proposed a constrained semantic image inpainting model to generate the missing content by conditioning on training images. Yu et al. [40] introduced a contextual attention mechanism into a generative adversarial network to explicitly utilize known surrounding background information as references to generate missing patches. In [41], they extended inpainting model to free shape of missing regions and made use of guidance learnt from millions of unlabeled images. However, to the best knowledge of the authors, there is no existing work using such techniques on time series forecasting, because they are very different tasks on the different data types. However, deep learning, which is able to automatically extract abstract features provides an opportunity to apply the techniques across domains.

III. Image Based Energy Consumption Series Forecasting

For the sake of clear presentation, the problem setting and notations are given first. Given a length-S observed series history $[y_1, y_2, \cdots, y_S]$ and a forecast horizon H ($H \ll S$), the task is to predict the vector of future values $[y_{S+1}, y_{S+2}, \cdots, y_{S+H}]$, where $y_i \in \mathbb{R}$ is the energy consumption at the i-th time step of the series. This study focuses on higher frequency series than daily, e.g., in hour and half hour. The term "image" in this paper refers to as a multi-channel matrix in $\mathbb{R}^{h \times w \times d}$, where h, w are height and width, and d is the number of channels and not necessarily equals to 3 like a RGB color image. The term "frame" refers to a slice of an image along the channel dimension, i.e., it is a matrix in $\mathbb{R}^{h \times w}$.

A. Time series encoding

Following the strategy of deep learning methods in traditional time series forecasting, the training set is constructed

TABLE I: Table of Notations

Symbol	Domain	Description
S	$\mathbb{R}$	Length of observed historical series
H	$\mathbb{R}$	Forecast horizon
$\mathbf{x}^{(b)}$	$\mathbb{R}^{L_b}$	$\mathbf{x}^{(b)} = [x_1, x_2, \cdots, x_{L_b}]$, long input series with length L_b
$\mathbf{x}^{(s)}$	$\mathbb{R}^{L_s}$	$\mathbf{x}^{(s)} = [x_1, x_2, \cdots, x_{L_s}]$, short input series with length L_s
$\mathbf{y}$	$\mathbb{R}^H$	Ground truth of forecast series (i.e., $\mathbf{y} = [y_1, y_2, \cdots, y_H]$)
$\mathbf{z}$	$\mathbb{R}^H$	Noise series (i.e., $\mathbf{z} = [z_1, z_2, \cdots, z_H]$)
m	$\mathbb{R}$	Frequency per day of the target series (e.g., $m = 24$ in hourly series and $m = 48$ in half-hourly series)
h	$\mathbb{R}$	Height of the images transformed by Eq.1
d	$\mathbb{R}$	The number of channels of the images transformed from $\mathbf{x}^{(b)}$
d'	$\mathbb{R}$	The number of channels of the images transformed from $\mathbf{x}^{(s)}$
τ	$\mathbb{R}$	The number of image channels inputted into LSTM cell
$\mathbf{x}'$	$\mathbb{R}^{h \times m \times d}$	Image transformed by Eq.1
$\dot{\mathbf{x}}^{(b)}$	$\mathbb{R}^{2h-1 \times m \times d}$	Image transformed from $\mathbf{x}^{(b)}$ by Eq.2
$\dot{\mathbf{x}}^{(s)}$	$\mathbb{R}^{2h-1 \times m \times d'}$	Image transformed from $\mathbf{x}^{(s)}$ by Eq.2

based on rolling windows. To encode both short-term and long-term temporal information, an instance is built as a pair of sub-series determined by a big and a small window respectively. Let the big window size $L_b = 7 \times m \times d - H$, where m is the frequency per day of the series, (e.g., $m = 24$ in hourly series and $m = 48$ in half-hourly series) and $L_b < S$. Thus, a sub-series with size L_b consists of contiguous time stamp values within d weeks. Let the small window size $L_s = h \times m \times d' - H$, where $h = \lceil \frac{H}{m} \rceil$ and $\lceil \cdot \rceil$ is the ceiling function. Thus, a sub-series with size L_s consists of contiguous time stamp values within $h \times d'$ days. Denote $T_1 = \{(X_b, X_s, Y)\}$ as the training set with n labeled instances: $X_b = \{\mathbf{x}_i^{(b)} | \mathbf{x}_i^{(b)} = [y_{i-(L_b-1)}, y_{i-(L_b-2)}, \cdots, y_{i-1}, y_i], L_b < i \leq S\}$, $X_s = \{\mathbf{x}_i^{(s)} | \mathbf{x}_i^{(s)} = [y_{i-(L_s-1)}, y_{i-(L_s-2)}, \cdots, y_{i-1}, y_i], L_s < i \leq S\}$, and $Y = \{\mathbf{y}_i | \mathbf{y}_i = [y_{i+1}, y_{i+2}, \cdots, y_{i+H}], i + H < S\}$. Therefore, for the i-th instance in the training set, $(\mathbf{x}_i^{(b)}, \mathbf{x}_i^{(s)})$ is the pair of input series and $\mathbf{y}_i$ is its label series. To simplify the notations, when referring to any input series, we omit the subscript i for $\mathbf{x}_i^{(b)}$, $\mathbf{x}_i^{(s)}$ and $\mathbf{y}_i$, and rewrite $\mathbf{x}^{(b)} = [x_1, x_2, \cdots, x_{L_b}]$, $\mathbf{x}^{(s)} = [x_1, x_2, \cdots, x_{L_s}]$ and $\mathbf{y} = [y_1, y_2, \cdots, y_H]$. The target of the problem is to predict $\mathbf{y}^*$ for a testing input $(\mathbf{x}^{*(b)}, \mathbf{x}^{*(s)})$. Figure 1a shows an example of an instance $(\mathbf{x}^{(b)}, \mathbf{x}^{(s)}, \mathbf{y})$ in this traditional 1d time series dataset T_1, where $m = 24$, $d = 1$, $d' = 6$ and $H = 12$. The numbers in the rectangles are energy values, and the brighter color indicates higher consumption in that time stamp.

Each instance $(\mathbf{x}^{(b)}, \mathbf{x}^{(s)}, \mathbf{y})$ in T_1 will be encoded into two images based on the same strategy. Let $\mathbf{x}$ denote the input series with the window size $L = h \times m \times d$, (i.e., $\mathbf{x} = \mathbf{x}^{(b)}$ if $h = 7$ and $\mathbf{x} = \mathbf{x}^{(s)}$ if $h = \lceil \frac{H}{m} \rceil$). To keep the unknown values in the forecasting horizon as a patch in the transformed image, a length-H random noise series $\mathbf{z} = [z_1, z_2, \cdots, z_H]$ is appended to each input series $\mathbf{x}$, and Eq. 1 is used to produce the input image $\mathbf{x}' \in \mathbb{R}^{h \times m \times d}$:

$$\mathbf{x}'_{p,q,r} = \begin{cases} x_{(r-1)\times m \times h + (p-1) \times m + q} & \text{if } (r-1) \times m \times h \\ & +(p-1) \times m + q \leq L \\ z_{(r-1)\times m \times h + (p-1) \times m + q - L} & \text{otherwise} \end{cases} \tag{1}$$

where $p \in \{1, 2, \cdots, h\}$ and $q \in \{1, 2, \cdots, m\}$ are the indexes of rows and columns respectively and $r \in \{1, 2, \cdots, d\}$ is the index of channels. Denote $I_{\mathbf{x}} = \{(p, q, r) | (r-1) \times m \times h + (p-1) \times m + q \leq L\}$ as the index set of the region with $\mathbf{x}$ and $I_{\mathbf{z}} = \{(p, q, r) | (r-1) \times m \times h + (p-1) \times m + q > L\}$ as the index set of the region with $\mathbf{z}$. According to Eq. 1, the input series $\mathbf{x}$ with appended $\mathbf{z}$ is arranged in left-to-right then top-to-bottom order as d matrices of $h \times m$ size. m is the number of time slots per day and therefore, e.g. for $\mathbf{x}^{(b)}$, such $h \times m$ $(h = 7)$ matrix consists of contiguous time series values within one week. These matrices are concatenated together as a d-channel image $\mathbf{x}'$ and $\mathbf{z}$ is the last H pixels in the last channel if $H < h \times m$, which is the unobserved patch to be predicted. If $d = 1$, there is only one frame in the image, and Figure 1b shows its 2d representation in a transformed image with $h = 7, m = 24, H = 36$.

Based on the above reorganization strategy, a window of $7 \times m \times d - H$ series is constructed into an image with the size of $7 \times m$ and d-channel. The images in the new training set are very tiny and narrow compared to most image application datasets where CNN is applicable. Moreover, the targets we aim to predict are the values in $I_{\mathbf{z}}$ region, which are located at the boundary of the images and hence there are no spatial neighbors from future time stamps. Therefore, a simple trick is used to augment the images further by Eq. 2:

$$\dot{\mathbf{x}}_{p,q,r} = \begin{cases} \mathbf{x}'_{p,q,r} & p \leq h, r \leq d \\ \mathbf{x}'_{2h-p,q,r} & p > h, r \leq d \end{cases} \tag{2}$$

where $\dot{\mathbf{x}} \in \mathbb{R}^{2h-1 \times m \times d}$. According to Eq. 2, each frame of the image is flipped down along the bottom boundary as a mirror reflection, as shown in Figure 1c. $\dot{I}_{\mathbf{x}}$ and $\dot{I}_{\mathbf{z}}$ are the index sets of pixels with original values from $\mathbf{x}$ and $\mathbf{z}$ respectively. Note that $\mathbf{y}$ consists of the corresponding ground truth of pixels in $\dot{I}_{\mathbf{z}}$. This constructed image training set is denoted as $T_2 = \{(\dot{\mathbf{x}}^{(b)}, \dot{\mathbf{x}}^{(s)}, \mathbf{y})\}$.

As per the traditional rolling window based time series forecasting problem, the testing input is the segment in the last time window, i.e., the testing input $\mathbf{x}^{*(b)} = [y_{S-L_b+1}, y_{S-L_b+2}, \cdots, y_S]$, $\mathbf{x}^{*(s)} = [y_{S-L_s+1}, y_{S-L_s+2}, \cdots, y_S]$, and the testing target $\mathbf{y}^* = [y_{S+1}, y_{S+2}, \cdots, y_{S+H}]$ in the original series. Following the same transform strategy, the testing instance $(\mathbf{x}^{*(b)}, \mathbf{x}^{*(s)}, \mathbf{y}^*)$ is converted to $(\dot{\mathbf{x}}^{*(b)}, \dot{\mathbf{x}}^{*(s)}, \mathbf{y}^*)$ by using Eq. 1 and Eq. 2. Table I lists the notations defined in this paper.

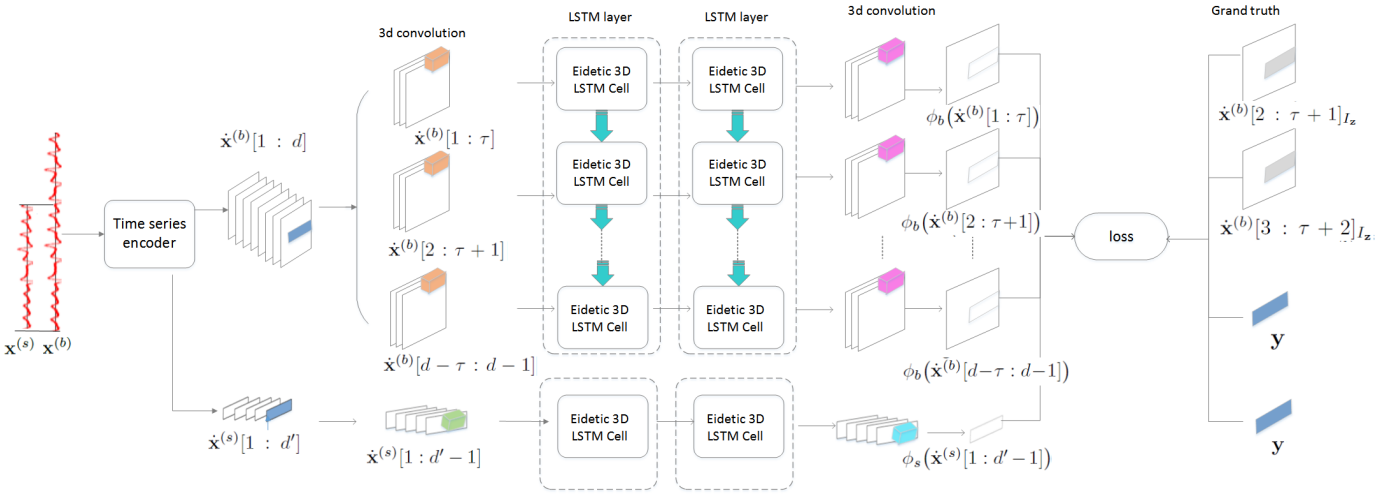


Fig. 2: The architecture of tsRNN. The notations in the figure are defined in Table I.

The reason why univariate time series are suitable to be explicitly organized as images in this way is because of the specific characteristics of energy consumption data: it has more obvious seasonality than time series in other domains. In the transformed image $\dot{\mathbf{x}}^{(b)}$, pixels in adjacent columns (e.g., $\dot{\mathbf{x}}_{p,q-1,r}$ and $\dot{\mathbf{x}}_{p,q,r}$) correspond to contiguous time slots of the same day (e.g., 10am and 11am in day q); pixels in adjacent rows (e.g., $\dot{\mathbf{x}}_{p-1,q,r}$ and $\dot{\mathbf{x}}_{p,q,r}$) correspond to the same time slots of consecutive days (e.g., 10am in day $p-1$ and day p), and pixels in adjacent channels (e.g., $\dot{\mathbf{x}}_{p,q,r-1}$ and $\dot{\mathbf{x}}_{p,q,r}$) correspond to the same time slots of consecutive weeks (e.g., Monday 10am in week $r-1$ and week r). Therefore, the techniques used to capture relationships in the spatial neighborhood can be adapted to extract features for the temporal pattern in the original time series. Figure 1d provides an image example which is transformed from a three-week hourly energy consumption series [16]. The periodic characteristic of energy consumption is visualized in the figure with the brighter color indicating more consumption in the time stamp.

B. A recurrent neural network for patch generation

Based on the time series encoding in section III-A, the problem setting of the univariate energy consumption can be stated as an image inpainting problem: given an image training set $T_2 = \{(\dot{\mathbf{x}}^{(b)}, \dot{\mathbf{x}}^{(s)}, \mathbf{y})\}$, the task is to learn a model to predict $\mathbf{y}^*$ in the unobserved patch located by $\dot{I}_{\mathbf{z}}$ for a testing input $(\dot{\mathbf{x}}^{*(b)}, \dot{\mathbf{x}}^{*(s)})$. A recurrent neural network is proposed to generate the unobserved patches. The proposed architecture - recurrent neural network for time series forecasting (tsRNN) is shown in Figure 2. tsRNN consists of two LSTMs integrated together by the objective function, one of which extracts long-term temporal features on $\mathbf{x}^{(b)}$, and the other explores short-term features on $\mathbf{x}^{(s)}$. For the sake of convenience, denote the frames of any image $\mathbf{x} \in \mathbb{R}^{h \times w \times d}$ from channels a to c as $\mathbf{x}[a:c]$, where $a, c \in \{1, \cdots, d\}$ and $a < c$. Thus, $\mathbf{x}$ is equivalent to $\mathbf{x}[1:d]$. As shown in Figure 2, time series $\mathbf{x}^{(b)}, \mathbf{x}^{(s)}$ are transformed into two images $\dot{\mathbf{x}}^{(b)}[1:d]$ and $\dot{\mathbf{x}}^{(s)}[1:d']$ with noise patches highlighted in blue by the time series encoder described in section III-A first, and then $\dot{\mathbf{x}}^{(b)}[1:d]$ is split into $d-\tau$ τ-channel images $\dot{\mathbf{x}}^{(b)}[1:\tau], \dot{\mathbf{x}}^{(b)}[2:\tau+1], \cdots, \dot{\mathbf{x}}^{(b)}[d-\tau:d-1]$, where τ is the predefined temporal depth as per the standard recurrent neural networks. For each τ-channel image $\dot{\mathbf{x}}^{(b)}[1:\tau], \cdots, \dot{\mathbf{x}}^{(b)}[d-\tau:d-2]$, the patch region located by $I_{\mathbf{z}}$ in the frames next to the last channel (i.e., $\dot{\mathbf{x}}^{(b)}[2:\tau+1]_{I_{\mathbf{z}}}, \dot{\mathbf{x}}^{(b)}[3:\tau+2]_{I_{\mathbf{z}}}, \cdots$ respectively) is the ground truth. For the last image $\dot{\mathbf{x}}^{(b)}[d-\tau:d-1]$, $\mathbf{y}$ is the ground truth. As shown in the last column in Figure 2, grey regions are ground truth patches and blue regions are the final target regions we want to generate. In tsRNN, only one input image $\dot{\mathbf{x}}^{(s)}[1:d'-1]$ is used for $\dot{\mathbf{x}}^{(s)}[1:d']$.

As discussed in section III-A, due to the periodic property of energy consumption data, dependencies exist between pixels in different frames of these transformed images. Long short-term memory (LSTM) is a powerful tool to capture the time dependencies. Wang et al. [31] proposed a new LSTM cell (Eidetic-3d) for video future frame generation, which is a memory cell interacting with its historical records via a gate-controlled self-attention module. We equip Eidetic-3d as the basic cell in tsRNN. As shown in Figure 2, tsRNN consists of two LSTMs. The τ-channel images $\dot{\mathbf{x}}^{(b)}[1:\tau]$ to $\dot{\mathbf{x}}^{(b)}[d-\tau:d-1]$ are input into the top LSTM in order. Each of them is propagated through Eidetic-3d cell layer by layer, and the cells in the same layer (indicated by dotted rectangles in the figure) share weights. Similarly, $\dot{\mathbf{x}}^{(s)}[1:d']$ is fed into the bottom LSTM. A following 3d convolutional layer is used to output feature maps as the generated frames $\phi_b\big(\dot{\mathbf{x}}^{(b)}[1:\tau]\big), \phi_b\big(\dot{\mathbf{x}}^{(b)}[2:\tau+1]\big), \cdots, \phi_b\big(\dot{\mathbf{x}}^{(b)}[d-\tau:d-1]\big)$, and $\phi_s\big(\dot{\mathbf{x}}^{(s)}[1:d'-1]\big)$. The generated patches located by the index set $I_{\mathbf{z}}$ (highlighted by grey rectangles in the figure) are supervised by the loss function in Eq. 3:

Algorithm 1 Pseudo code of learning in tsRNN

Input: An energy consumption series $[y_1, y_2, \cdots, y_S]$, where $y_i \in \mathbb{R}$, the series frequency m, and the forecast horizon H.
Output: Predictions of future values $\tilde{\mathbf{y}}^* = [\tilde{y}_{S+1}, \tilde{y}_{S+2}, \cdots, \tilde{y}_{S+H}]$.

```
Initialize integers d, d' and τ (τ < d). Initialize a noise vector z = [z_1, z_2, ..., z_H].
Build the training set T_1 = {(x^(b), x^(s), y)} from [y_1, ..., y_S] based on the definition.
Transform T_1 = {(x^(b), x^(s), y)} to T_2 = {(ẋ^(b), ẋ^(s), y)} based on Eq. 1 and Eq. 2.
for epoch = 1 to MAX_epoch do
    Shuffle T_2 and divide it into mini-batches T_s.
    for each T_s do
        Forward propagate instances through the three layers of tsRNN as shown in Figure 2.
        Calculate loss by Eq. 3.
        Backward propagate and update the weights.
    end for
end for
Construct the testing instance (ẋ^(b)*, ẋ^(s)*, y*) from ([y_{S-L_b+1}, y_{S-L_b+2}, ..., y_S], [y_{S-L_s+1}, y_{S-L_s+2}, ..., y_S], [y_{S+1}, y_{S+2}, ..., y_{S+H}]).
Forward propagate (ẋ^(b)*, ẋ^(s)*), and output ỹ* = φ_s(ẋ^(s)*[1 : d' − 1])_{I_z}.
```

$$\mathcal{L} = \frac{1}{2} \sum_{(\dot{\mathbf{x}}^{(b)}, \dot{\mathbf{x}}^{(s)}, \mathbf{y}) \in T_2} \Big(||\phi_s\big(\dot{\mathbf{x}}^{(s)}[1 : d' - 1]\big)_{I_\mathbf{z}} - \mathbf{y}||_2^2 + \sum_{i=1}^{d-\tau-1} ||\phi_b\big(\dot{\mathbf{x}}^{(b)}[i : i + \tau - 1]\big)_{I_\mathbf{z}} - \dot{\mathbf{x}}^{(b)}[i + 1 : i + \tau]_{I_\mathbf{z}}||_2^2 + ||\phi_b\big(\dot{\mathbf{x}}^{(b)}[d - \tau : d - 1]\big)_{I_\mathbf{z}} - \mathbf{y}||_2^2 \Big) \tag{3}$$

where $\phi_b(\cdot)$ and $\phi_s(\cdot)$ are the mapping functions representing the layer operations of the top LSTM and the bottom LSTM in tsRNN respectively, and operator $[\cdot]_{I_\mathbf{z}}$ collects values at each position (p, q, r) in the index set $I_\mathbf{z} = \{(p, q, r)\}$ in order of increasing p, q, r. $||\cdot||_2^2$ is the L_2 norm.

The learning of tsRNN is summarized in Algorithm 1. The input is the history series $[y_1, y_2, \cdots, y_S]$ with frequency m and the horizon H to be predicted. The training set T_2 is built from this series based on the horizon H. Given different forecasting horizon H, different images are constructed correspondingly due to the different $I_\mathbf{z}$ even if the same history series is provided. Figure 2 shows an example that H is shorter than one week. However, tsRNN supports longer horizons without any changes, in that case the generated frames and ground truth images are more than one channel.

IV. Experimental results

The proposed tsRNN is comprehensively evaluated on benchmarks of two different types of power load - system load and residential load with different frequencies and different value scales. The first benchmark has two sub datasets of system load, which is used to evaluate the predictive performance of tsRNN compared with the baseline methods for different forecasting horizons. The second benchmark is residential load, which is used to evaluate the generalization capability of tsRNN. Furthermore, since the data-driven univariate load forecasting methods predict future series only based on historical records, which do not rely on any other information related to load types, these two benchmarks with different types of load are also used to evaluate the performances of load type-agnostic models. The benchmarks used in the experiments are summarized as follows:

- PJM system load data [16] is collected from the website of PJM Interconnection LLC, which is a regional transmission organization in the United States operating an electricity transmission system serving 14 states of US. The experiments are performed on two sub datasets - AEP and Dayton, which record the electricity consumption in Michigan and a part of Ohio served by the company AEP and Dayton under PJM respectively. Both consist of one hourly series from 2 Oct 2004 00:00 to 2 Aug 2018 23:00 with 121,272 time steps in megawatts (MW). The min and max consumption values in AEP series and Dayton series are (9581.0, 25695.0) and (982.0, 3746.0).
- London residential load data [42] is collected from London smart meter data store, which contains the energy consumption readings of 5,567 households in 112 blocks between November 2011 and February 2014 in kilowatt-hours (kWh). This benchmark consists of 5,567 half-hourly series, and the starting and ending dates are not the same for all series. The min and max consumption values of different series are different, and all the consumption values are in the range of [0 to 10] kWh.

A. Comparing to baseline methods

To comprehensively evaluate the predictive performance of tsRNN, statistical univariant time series forecasting methods, traditional machine learning methods and deep learning methods are compared on the PJM benchmark. The baseline methods include seven traditional models: Seasonal Auto Regressive Integrated Moving Average (SARIMA, [43]), Exponential Smoothing (ExpSm, [44]), Theta [45], Trigonometric Box-Cox ARMA Trend Seasonal (TBATS, [46]), and Dynamic

TABLE II: Evaluation metrics

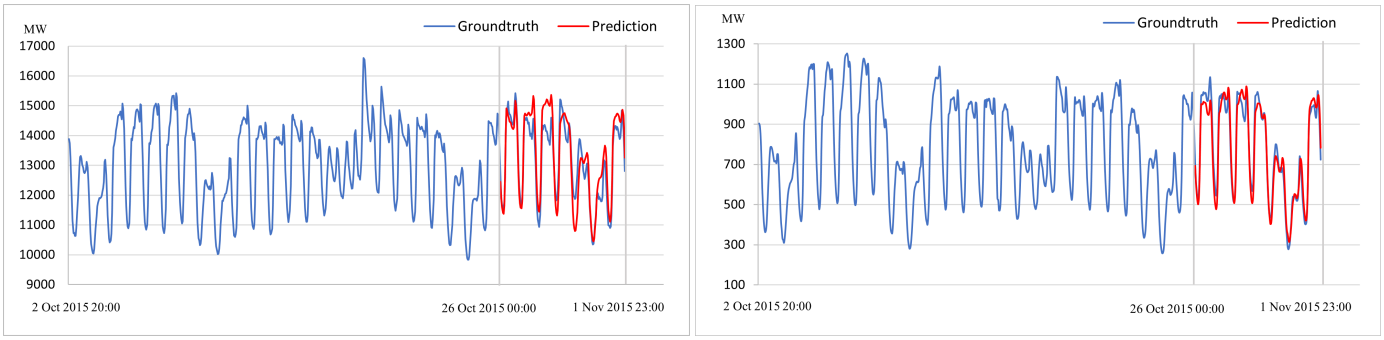
Name	Equations
RMSE (root mean squared error)	$\sqrt{\frac{1}{H}\sum_{i=1}^{H}(y_i - \tilde{y}_i)^2}$
MAE (mean absolute error)	$\frac{1}{H}\sum_{i=1}^{H}\lvert y_i - \tilde{y}_i\rvert$
MAPE (mean absolute percent error)	$\frac{1}{H}\sum_{i=1}^{H}\lvert\frac{y_i - \tilde{y}_i}{y_i}\rvert$
sMAPE (symmetric mean absolute percent error)	$\frac{1}{H}\sum_{i=1}^{H}\frac{2\times\lvert y_i - \tilde{y}_i\rvert}{\lvert y_i\rvert + \lvert\tilde{y}_i\rvert}$
ND (normalized deviation)	$\frac{\sum_{i=1}^{H}\lvert y_i - \tilde{y}_i\rvert}{\sum_{i=1}^{H}\lvert y_i\rvert}$

TABLE III: Results on PJM - AEP.

	$h = 48$					$h = 168$				
	RMSE	MAE	MAPE	sMAPE	ND	RMSE	MAE	MAPE	sMAPE	ND
SARIMA	.057±.04	.047±.03	.151±.11	.146±.09	.138±.09	.113±.07	.094±.06	.433±.33	.311±.17	.302±.19
ExpSm	.095±.06	.077±.05	.235±.14	.226±.10	.228±.13	.126±.06	.101±.05	.431±.30	.322±.12	.332±.17
CatBoost	.056±.03	.046±.03	.149±.13	.139±.08	.138±.09	.089±.04	.073±.04	.323±.28	.259±.11	.237±.12
PooledRegression	.057±.04	.048±.03	.152±.13	.146±.09	.141±.09	.090±.04	.074±.04	.333±.30	.259±.11	.239±.12
Theta	.111±.04	.096±.04	.381±.20	.312±.11	.290±.10	.134±.05	.113±.04	.655±.47	.386±.12	.371±.12
TBATS	.096±.05	.080±.05	.272±.17	.263±.19	.240±.14	.194±.21	.161±.17	.771±.95	.491±.35	.528±.56
DHR	.210±.08	.179±.07	.648±.34	618±.20	.536±.17	.223±.06	.186±.05	1.051±.82	.618±.15	.630±.24
MLP	.122±.08	.108±.08	.375±.31	.322±.20	.327±.25	.127±.06	.107±.06	.437±.28	.367±.16	.340±.16
DeepAR	.086±.05	.069±.05	.196±.12	.220±.14	.197±.10	.092±.04	.076±.04	.380±.35	.262±.12	.256±.15
N-BEATS	.055±.03	.046±.03	**.143±.12**	.140±.08	**.133±.07**	.088±.04	.072±.04	.304±.25	.257±.10	.231±.10
Transformer	.073±.04	.061±.04	.204±.17	.180±.10	.188±.12	.138±.06	.120±.06	.417±.16	.503±.19	.371±.12
WaveNet	**.053±.03**	**.044±.03**	.145±.12	**.134±.07**	.134±.08	.088±.04	.073±.03	.363±.32	.257±.12	.251±.14
LSTM	.114±.04	.100±.07	.346±.25	.303±.17	.294±.19	.121±.05	.102±.05	.493±.34	.352±.12	.335±.15
tsRNN-2d-half	.075±.03	.065±.03	.227±.14	.199±.07	.191±.08	.135±.04	.120±.04	.655±.05	.400±.18	.444±.27
tsRNN-2d	.065±.03	.055±.03	.190±.13	.174±.08	.168±.08	.135±.04	.120±.04	.657±.05	.401±.18	.445±.27
tsRNN	.060±.03	.051±.03	.176±.14	.157±.08	.159±.09	**.083±.04**	**.069±.03**	**.294±.25**	**.246±.09**	**.224±.09**

TABLE IV: Results on PJM - Dayton.

	$h = 48$					$h = 168$				
	RMSE	MAE	MAPE	sMAPE	ND	RMSE	MAE	MAPE	sMAPE	ND
SARIMA	.053±.04	.044±.03	.110±.08	.110±.08	.108±.07	.113±.07	.093±.06	.310±.23	.247±.14	.254±.17
ExpSm	.069±.04	.057±.03	.137±.07	.139±.07	.140±.08	.103±.05	.084±.05	.266±.16	.225±.10	.230±.12
CatBoost	.051±.03	.042±.03	.100±.06	.100±.06	.101±.07	.076±.03	.062±.03	.178±.09	.175±.07	.165±.07
PooledRegression	.053±.03	.044±.03	.105±.07	.106±.07	.106±.07	.080±.04	.066±.04	.198±.12	.183±.09	.177±.09
Theta	.103±.04	.090±.04	.248±.08	.233±.07	.222±.07	.124±.05	.105±.04	.365±.19	.290±.09	.287±.10
TBATS	.094±.05	.079±.05	.208±.12	.204±.13	.194±.11	.168±.15	.140±.13	.464±.47	.394±.36	.392±.38
DHR	.194±.07	.165±.06	.415±.13	.464±.14	.403±.11	.202±.06	.168±.05	.552±.22	.461±.10	.463±.13
MLP	.094±.05	.082±.05	.217±.15	.206±.12	.204±.13	.103±.04	.087±.04	.251±.11	.255±.11	.233±.09
DeepAR	.067±.03	.056±.03	.185±.13	.171±.08	.168±.09	.088±.05	.072±.04	.236±.16	.197±.10	.200±.12
N-BEATS	.055±.03	.046±.03	.152±.13	.141±.07	.137±.08	.076±.04	.062±.03	.184±.12	.176±.08	.166±.08
Transformer	.064±.03	.054±.03	.176±.14	.173±.09	.160±.09	.127±.05	.110±.05	.296±.08	.360±.13	.293±.09
WaveNet	**.050±.03**	**.041±.03**	**.098±.06**	**.098±.06**	**.099±.06**	.114±.07	.097±.07	.303±.21	.241±.12	.270±.18
LSTM	.088±.05	.077±.05	.196±.11	.190±.09	.185±.10	.096±.04	.080±.04	.238±.10	.220±.08	.212±.08
tsRNN-2d-half	.062±.03	.052±.02	.136±.06	.130±.05	.129±.05	.100±.03	.086±.03	.264±.12	.237±.08	.242±.10
tsRNN-2d	.058±.03	.049±.02	.127±.06	.122±.05	.122±.06	.099±.03	.086±.03	.263±.12	.236±.08	.242±.10
tsRNN	.056±.03	.047±.03	.123±.07	.117±.06	.119±.07	**.071±.03**	**.058±.03**	**.176±.09**	**.163±.07**	**.159±.07**



(a) AEP series segment from 02 Oct 2015 to 01 Nov 2015.

(b) DAYTON series segment from 02 Oct 2015 to 01 Nov 2015.

Fig. 3: Examples of series and predictions of tsRNN.

Harmonic Regression (DHR, [47]), CatBoost [48], Pooled Regression [49]; and six deep neural network methods: multi-layer perception (MLP) and long short-term memory (LSTM), DeepAR [50], N-BEATS [51], WaveNet [52] and Transformer [53]. In the first category, CatBoost and Pooled Regression are global methods, which predict the future series based on all training series samples. However, other four statistical baselines are local methods, which predict the future values of the target series only based on its own historical series. In the deep learning category, WaveNet and Transformer are CNN-based methods; MLP and N-BEATS are fully-connected networks; and LSTM and DeepAR are recurrent networks. Six metrics [54] are used as performance indexes. Table II lists their definitions, where $\tilde{y_i}$ is the prediction for i-th time stamp in the forecasting horizon, y_i is the ground truth and H is the horizon length.

AEP and Dayton data, each of which contains 121,272 continuous time steps, are split into a history segment for training and a future segment for testing based on the 8:2 ratio (i.e., the first 98,720 time stamps are for training and the remaining 24,192 slots of 144-week steps are for testing). For all deep learning methods - tsRNN, MLP, LSTM, DeepAR, N-BEATS, WaveNet and Transformer - the segment $[y_1 \cdots y_S]$ (S=98,720) is used to construct the training sets based on rolling windows, and the 144-week testing segment $[y_{S+1} \cdots y_{|\mathbf{y}|}]$ ($|\mathbf{y}|$ is the total number of time stamps in $\mathbf{y}$) is split into

144 segments $\{[y_{S+i*168-L_b}, \cdots, y_{S+i*168}]|i = 0, \cdots, 143\}$ and converted into the testing samples $\{(\mathbf{x}_i^{(b)*}, \mathbf{x}_i^{(s)*}, \mathbf{y}_i^*)|i = 0, \cdots, 143\}$. All the methods are compared for two forecasting horizons: $h = 48$ (two days) and $h = 168$ (one week). For each h, only one model of tsRNN is trained on $[y_1 \cdots y_S]$, which is applied to predict for all 144 testing samples, likewise for all deep learning baselines. However, local statistical methods, including SARIMA, ExpSm, Theta, TBATS, and DHR, are only able to predict the future slots next to the end of training slots. For example, if the forecasting target is $[y_{S+1} \cdots y_{S+h}]$, the training segment must end at y_S. As a result, to obtain the predictions for testing samples, 144 models of local methods are learnt from $[y_1 \cdots y_S], [y_1 \cdots y_{S+168}], [y_1 \cdots y_{S+2\times168}], \cdots, [y_1 \cdots y_{|\mathbf{y}|-168}]$ respectively. This is a limitation of them compared to deep learning methods.

TABLE V: Hyper-parameters of the deep learning models for the PJM benchmark

	$h = 48$	$h = 168$
tsRNN		
#layers	2 Eidetic LSTM layers	
#nodes	4 cells with [2,3,3] conv (1st layer) 4 cells with [2,3,3] conv (2nd layer)	
window size	$d = 5$, $d' = 5$, $\tau = 2$	
minibatch size	192	
#iterations	10000	
learning rate	0.0001	
MLP		
#layers	2 fully-connected layers + Relu	
#nodes	224 neurons (1st layer) 192 neurons (2nd layer)	
window size	128	256
minibatch size	64	
#iterations	75,000	
learning rate	0.001	
LSTM		
#layers	1 LSTM layer + Relu + Dropout 1 fully-connected layer	
#nodes	128 units(1st layer) 48 neurons(2nd layer)	192 units(1stlayer) 168 neurons(2nd layer)
window size	128	256
minibatch size	64	
#iterations	75,000	
learning rate	0.001	

By using grid search on $\{0, 1, 2\}$ to set the optimal combination for the six standard parameters of SARIMA (auto-regressive, integrated, moving average, seasonal auto-regressive, seasonal integrated and seasonal moving average), it is observed that for the first ten models learnt from $[\mathbf{y}_1 \cdots \mathbf{y}_S], [\mathbf{y}_1 \cdots \mathbf{y}_{S+1}], \cdots, [\mathbf{y}_1 \cdots \mathbf{y}_{S+10}]$, the optimal combinations of all models are [1,1,1,1,1,1]. Therefore, in the experiments, the six parameters of SARIMA are set as 1. For standard MLP and LSTM models, five sets of features are extracted. For instance, for the time stamp i, the features include history consumption values $[y_{i-L} \cdots y_i]$ within a window L, consumption differences between every two adjacent time stamps $[y_{i-L+1} - y_{i-L}, \cdots, y_i - y_{i-1}]$, codes indicating whether the consumption values are increasing (i.e., 1 if $y_{j+1} > y_j$; -1 if $y_{j+1} < y_j$; 0 otherwise. $j = i - L, \cdots, i$), one binary code indicating whether the time stamp i is in a workday, 24 one-hot codes indicating that the time stamp i is in which hour of the day (e.g., if the time stamp i belongs to 00:00-00:59, the codes are [1 0 0 0 0 0 ...]). Settings of other statistical baselines follow the default parameters in Monash Time Series Forecasting Archive [55].

The hyper-parameters used in the experiments including window size, architecture information and the training parameters of MLP, LSTM and tsRNN are listed in Table V. Hyper-parameters of other baselines follow the default settings in Monash Time Series Forecasting Archive [55]. Table III and IV summarize the comparison results on the PJM benchmark. Due to the different value scales of AEP and Dayton, the training and testing segments are normalized by $\frac{y_i - min}{max - min}$ where min and max are minimum and maximum values of the training segments, and the prediction results are computed on the normalized data. The numbers in Table III and Table IV are the mean errors on the 144 testing samples and $\pm$ represents standard deviation. The best performances are highlighted in bold, and the second best results are underlined.

As shown in the tables, for the forecasting horizon $h = 48$, WaveNet and N-BEATS perform the best on AEP dataset. For Dayton dataset, WaveNet is the leading model and CatBoost follows behind. However, when horizon $h = 168$, the performance of WaveNet drops significantly compared to that on the shorter horizon. Similar characteristic is observed on local statistical methods (e.g., SARIMA) which are good at short future forecasting, but their performances change a lot with increasing length of the forecasting horizon. The proposed tsRNN achieves the best results for the forecasting horizon $h = 168$ on both AEP and Dayton datasets, and its performance is affected less by the length of forecasting horizon. Figure 3 shows examples from AEP and Dayton dataset. The blue curves are the true series data, and red curves are the one week-ahead predictions performed by tsRNN. It can be observed that the main errors occur at the time stamps when the electricity load values change significantly.

B. Ablation Studies

To evaluate the benefits of the data reconstruction and augmentation, the proposed method is tested on every intermediate transformed data shown in Figure 1. For the original one-dimensional series data (i.e., Figure 1a) before reconstruction, the proposed idea degenerates to a standard LSTM, whose performance has been reported in Tables III and IV. Because tsRNN is proposed for 3D data in terms of two key aspects: the supervised learning is to infer frame d based on frames 1 to $d-1$ of the images; and 3D convolution kernel works on adjacent multiple frames, it is not able to be performed directly on 2D reconstructions. Therefore, we replicate the target frame (as shown in Figure 1b and Figure 1c) to enable the above 3D operations, i.e, in Figure 2, $\dot{\mathbf{x}}^{(b)}[1:d]$ is with $d = 2$ and $\dot{\mathbf{x}}^{(b)}[1] = \dot{\mathbf{x}}^{(b)}[2]$. Under this replication, all frames of $\dot{\mathbf{x}}^{(s)}[1:d]$ with $\dot{\mathbf{x}}^{(s)}[1] = \dot{\mathbf{x}}^{(s)}[2]$ in Figure 2 are the noises replacing the target regions. Hence, the part of loss on $\dot{\mathbf{x}}^{(s)}$ of Eq.3 is omitted in the experiments for 2D data.

tsRNN-2d-half in Tables III and IV is the tsRNN model performed on the replicated 2D data of Figure 1b, and tsRNN-

2d is that on Figure 1c. As indicated by the results of all metrics for all four dataset settings in Tables III and IV, tsRNN-2d-half and tsRNN-2d perform much better than LSTM but worse than tsRNN, which verified the benefits of the data reconstruction. Moreover, comparing to tsRNN-2d-half, the inputs of tsRNN-2d enjoy spatial augmentation by flipping. For 48-horizon forecasting on both datasets, tsRNN-2d outperforms tsRNN-2d-half benefiting from the data augmentation. However, for 168-horizon forecasting, tsRNN-2d and tsRNN-2d-half produce very similar results, but perform much worse than tsRNN, which demonstrates that when models are out of capability to capture the underlying patterns, the spatial augmentation of data is not able to help much.

C. Generalization capability of tsRNN

To evaluate the generalization capability of tsRNN, the London benchmark is divided on blocks as training/testing partition (blocks 1-90 for training and blocks 91-112 for testing). The energy consumption of each household in a block is stored as one time series, and therefore, in this experiment, the models are learnt on multiple series from the training blocks and applied on the series from the testing blocks. As the starting and ending time of different series are different, we use all slots from all training series to construct the transformed images, and use the last horizon ($h = 48, 168$) of each testing series as the forecasting target. The aims of this experiment are to evaluate the capability of a method to model the common patterns of energy consumption of households in these blocks based on an assumption that these blocks have high similarity in terms of energy consumption.

The number of households in the training blocks (i.e., blocks 1-90) is 4,397, and the average number of training samples of tsRNN to be constructed from one household series is 500 by the encoder of Section III-A. Therefore, the number of training samples is more than 2,100,000. To deal with the large scalability of this dataset, in the training phase of tsRNN, for each epoch, 100 households are randomly selected and those series are converted to images and split as minibatches to train the model. When one epoch is finished, another 100 random households are chosen. There are abnormal points in the series as the values in a whole day are zeros. In the preprocessing, the days with more than 15 zero values are removed from the training series. When there are zeros in the testing series, and the corresponding MAPE and ND will be infinity for any predictions (see the definitions in Table II). Therefore, the three matrices RMSE, MAE and sMAPE are evaluated in this experiment. Consumption values are normalized by minimum and maximum values of each series separately using $\frac{y_i - min}{max - min}$. Table VI shows the results. Benefiting from making use of series from the training blocks, tsRNN outperforms all baselines significantly. It shows that given time series under the assumption of high similarity, tsRNN is able to capture their common characteristics and generalized to other related series. The architecture and hyper-parameters of deep learning models in this experiments are listed in Table VII.

In the same way as the experiments in Section IV-A, separate SARIMA and ExpSm models are learnt for each of the testing series (the number of testing series is 1,038),

TABLE VII: Hyper-parameters of the deep learning models for the London benchmark

	$h = 48$	$h = 168$
tsRNN		
#layers	2 Eidetic LSTM layers	
#nodes	4 cells with [2,3,3] conv (1st layer) 4 cells with [2,3,3] conv (2nd layer)	
window size	$d = 5$, $d' = 5$, $\tau = 2$	
minibatch size	64	
#iterations	1000	
learning rate	0.0001	
MLP		
#layers	2 fully-connected layers + Relu	
#nodes	128 neurons(1st layer) 96 neurons(2nd layer)	224 neurons(1st layer) 192 neurons(2nd layer)
window size	128	256
minibatch size	48	
#iterations	100,000	
learning rate	0.001	
LSTM		
#layers	1 LSTM layer + Relu + Dropout 1 fully-connected layer	
#nodes	128 units(1st layer) 48 neurons(2nd layer)	192 units(1stlayer) 168 neurons(2nd layer)
window size	128	256
minibatch size	64	
#iterations	85,000	
learning rate	0.001	

whereas only one deep learning model is trained from all of the training series. For tsRNN and the other deep learning baselines, following the traditional strategy, the segment before the target horizon in the testing series is used to construct the testing input. As SARIMA and ExpSm are only able to predict the future stamps based on the previous history in the same time series, in each testing series, the exact segment converted to the testing input of deep learning models is used to learn the SARIMA and ExpSm model. Therefore, for different forecasting horizon $h = 48$ and $h = 168$, the segments in a same testing series used to learn SARIMA and ExpSm models are different. For example, given a testing series $\mathbf{y}$, if $h = 48$, the segment used to learn SARIMA and ExpSm is $[y_{|\mathbf{y}|-48-L_b} \cdots y_{|\mathbf{y}|-48}]$; if $h = 168$, the segment will be $[y_{|\mathbf{y}|-168-L_b} \cdots y_{|\mathbf{y}|-168}]$. From the experimental results listed in Table VI, we can observe that the performances of SARIMA and ExpSm for $h = 48$ and $h = 168$ are close, which is obviously different from the phenomenon in the experiments of Section IV-A. In those experiments, the predictions for the next 48 or 168 time stamps come from the same SARIMA and ExpSm model learnt from the same segment. Therefore, it can be concluded that the performances of SARIMA and ExpSm are highly affected by the length of training segment, particularly when the length of training segment is not significantly longer then the forecasting horizon.

V. Conclusion

Benefiting from the rich representation capability of images, this paper tackles the long-term prediction and generalization challenges of short-term load forecasting problem by transforming a consumption series into multi-channel images. Based on the transformation, a forecasting problem for univariant load series is converted to an image inpainting problem.

TABLE VI: Results on the London benchmark

	$h = 48$			$h = 168$		
	RMSE	MAE	sMAPE	RMSE	MAE	sMAPE
SARIMA	.126±.07	.089±.05	.806±.27	.126±.08	.085±.07	.835±.29
ExpSm	.124±.08	.087±.06	.844±.36	.131±.23	.093±.21	.900±.37
MLP	.139±.19	.092±.09	.805±.26	.133±.14	.089±.05	.824±.25
LSTM	.139±.19	.093±.09	.809±.27	.132±.14	.088±.05	.819±.25
tsRNN	**.094±.05**	**.063±.04**	.711±.27	**.102±.05**	**.067±.03**	.718±.24

This paper proposes an adapted recurrent deep neural network with 3d convolutions to generate the unobserved patch in the transformed images, which is the prediction of future time stamps of the consumption series. Experimental results have shown its effectiveness and generalization capabilities compared to both statistical models and standard deep learning methods.

ACKNOWLEDGMENT

The authors would like to thank Siemens Smart Infrastructure, Singapore for providing their domain expertise in the field of building energy and Energy Research Institute @ Nanyang Technological University (ERI@N) to conduct this research as a part of the EcoCampus Flagship Programme which focused on campus-wide sustainability.

REFERENCES

[1] T. Hong *et al.*, "Energy forecasting: Past, present, and future," *Foresight: The International Journal of Applied Forecasting*, no. 32, pp. 43–48, 2014.

[2] T. Hong and S. Fan, "Probabilistic electric load forecasting: A tutorial review," *International Journal of Forecasting*, vol. 32, no. 3, pp. 914–938, 2016.

[3] N. Phuangpornpitak and W. Prommee, "A study of load demand forecasting models in electric power system operation and planning," *GMSARN International Journal*, vol. 10, no. 2016, pp. 19–24, 2016.

[4] W. Xifan and J. R. McDonald, "Modern power system planning," 1994.

[5] H. S. Hippert, C. E. Pedreira, and R. C. Souza, "Neural networks for short-term load forecasting: A review and evaluation," *IEEE Transactions on power systems*, vol. 16, no. 1, pp. 44–55, 2001.

[6] K. Metaxiotis, A. Kagiannas, D. Askounis, and J. Psarras, "Artificial intelligence in short term electric load forecasting: a state-of-the-art survey for the researcher," *Energy conversion and Management*, vol. 44, no. 9, pp. 1525–1534, 2003.

[7] T. Hong, *Short term electric load forecasting*. North Carolina State University, 2010.

[8] S. Humeau, T. K. Wijaya, M. Vasirani, and K. Aberer, "Electricity load forecasting for residential customers: Exploiting aggregation and correlation between households," in *2013 Sustainable internet and ICT for sustainability (SustainIT)*. IEEE, 2013, pp. 1–6.

[9] N. Charlton and C. Singleton, "A refined parametric model for short term load forecasting," *International Journal of Forecasting*, vol. 30, no. 2, pp. 364–368, 2014.

[10] S. Fan and R. J. Hyndman, "Short-term load forecasting based on a semi-parametric additive model," *IEEE Transactions on Power Systems*, vol. 27, no. 1, pp. 134–141, 2011.

[11] V. Hinojosa and A. Hoese, "Short-term load forecasting using fuzzy inductive reasoning and evolutionary algorithms," *IEEE Transactions on power systems*, vol. 25, no. 1, pp. 565–574, 2010.

[12] J. W. Taylor, "Short-term electricity demand forecasting using double seasonal exponential smoothing," *Journal of the Operational Research Society*, vol. 54, no. 8, pp. 799–805, 2003.

[13] S.-J. Huang and K.-R. Shih, "Short-term load forecasting via arma model identification including non-gaussian process considerations," *IEEE Transactions on power systems*, vol. 18, no. 2, pp. 673–679, 2003.

[14] B.-J. Chen, M.-W. Chang *et al.*, "Load forecasting using support vector machines: A study on eunite competition 2001," *IEEE transactions on power systems*, vol. 19, no. 4, pp. 1821–1830, 2004.

[15] M. Singh and R. Maini, "Various electricity load forecasting techniques with pros and cons," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8, no. 6, pp. 220–229, 2020.

[16] https://www.kaggle.com/robikscube/hourly-energy-consumption.

[17] K. Nilakanta Singh, R. Singh *et al.*, "A review on deep learning models for short-term load forecasting," *Applications of Artificial Intelligence and Machine Learning*, pp. 705–721, 2021.

[18] L. Wang, Z. Zhang, and J. Chen, "Short-term electricity price forecasting with stacked denoising autoencoders," *IEEE Transactions on Power Systems*, vol. 32, no. 4, pp. 2673–2681, 2016.

[19] H. Shi, M. Xu, and R. Li, "Deep learning for household load forecasting—a novel pooling deep rnn," *IEEE Transactions on Smart Grid*, vol. 9, no. 5, pp. 5271–5280, 2017.

[20] Z. Deng, B. Wang, Y. Xu, T. Xu, C. Liu, and Z. Zhu, "Multi-scale convolutional neural network with time-cognition for multi-step short-term load forecasting," *IEEE Access*, vol. 7, pp. 88 058–88 071, 2019.

[21] Ö. F. Ertugrul, "Forecasting electricity load by a novel recurrent extreme learning machines approach," *International Journal of Electrical Power & Energy Systems*, vol. 78, pp. 429–435, 2016.

[22] C. Tong, J. Li, C. Lang, F. Kong, J. Niu, and J. J. Rodrigues, "An efficient deep model for day-ahead electricity load forecasting with stacked denoising auto-encoders," *Journal of Parallel and Distributed Computing*, vol. 117, pp. 267–273, 2018.

[23] X. Dong, L. Qian, and L. Huang, "Short-term load forecasting in smart grid: A combined cnn and k-means clustering approach," in *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*. IEEE, 2017, pp. 119–125.

[24] K. Amarasinghe, D. L. Marino, and M. Manic, "Deep neural networks for energy load forecasting," in *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*. IEEE, 2017, pp. 1483–1488.

[25] I. Koprinska, D. Wu, and Z. Wang, "Convolutional neural networks for energy time series forecasting," in *2018 international joint conference on neural networks (IJCNN)*. IEEE, 2018, pp. 1–8.

[26] L. Li, K. Ota, and M. Dong, "Everything is image: Cnn-based short-term electrical load forecasting for smart grid," in *2017 14th International Symposium on Pervasive Systems, Algorithms and Networks & 2017 11th International Conference on Frontier of Computer Science and Technology & 2017 Third International Symposium of Creative Computing (ISPAN-FCST-ISCC)*. IEEE, 2017, pp. 344–351.

[27] C. Tian, J. Ma, C. Zhang, and P. Zhan, "A deep neural network model for short-term load forecast based on long short-term memory network and convolutional neural network," *Energies*, vol. 11, no. 12, p. 3493, 2018.

[28] W. He, "Load forecasting via deep neural networks," *Procedia Computer Science*, vol. 122, pp. 308–314, 2017.

[29] M. Cai, M. Pipattanasomporn, and S. Rahman, "Day-ahead building-level load forecasts using deep learning vs. traditional time-series techniques," *Applied energy*, vol. 236, pp. 1078–1088, 2019.

[30] L. Han, Y. Peng, Y. Li, B. Yong, Q. Zhou, and L. Shu, "Enhanced deep networks for short-term and medium-term load forecasting," *IEEE Access*, vol. 7, pp. 4045–4055, 2018.

[31] Y. Wang, L. Jiang, M.-H. Yang, L.-J. Li, M. Long, and L. Fei-Fei, "Eidetic 3d lstm: A model for video prediction and beyond," in *The 7th International Conference on Learning Representations (ICLR)*, 2019.

[32] K. Nazeri, E. Ng, T. Joseph, F. Qureshi, and M. Ebrahimi, "Edgeconnect: Structure guided image inpainting using edge prediction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, Oct 2019.

[33] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, "Image inpainting for irregular holes using partial convolutions," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 85–100.

[34] Y. Wang, X. Tao, X. Qi, X. Shen, and J. Jia, "Image inpainting via generative multi-column convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[35] Z. Yan, X. Li, M. Li, W. Zuo, and S. Shan, "Shift-net: Image inpainting via deep feature rearrangement," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 1–17.

[36] Z. Qin, Q. Zeng, Y. Zong, and F. Xu, "Image inpainting based on deep learning: A review," *Displays*, p. 102028, 2021.

[37] Y. Zeng, J. Fu, H. Chao, and B. Guo, "Learning pyramid-context encoder network for high-quality image inpainting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 1486–1494.

[38] C. Zheng, T.-J. Cham, and J. Cai, "Pluralistic image completion," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 1438–1447.

[39] R. A. Yeh, C. Chen, T. Yian Lim, A. G. Schwing, M. Hasegawa-Johnson, and M. N. Do, "Semantic image inpainting with deep generative models," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5485–5493.

[40] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Generative image inpainting with contextual attention," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 5505–5514.

[41] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. Huang, "Free-form image inpainting with gated convolution," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019, pp. 4471–4480.

[42] https://www.kaggle.com/jeanmidev/smart-meters-in-london.

[43] J. Durbin and S. J. Koopman, *Time series analysis by state space methods*. Oxford university press, 2012.

[44] R. Hyndman, A. B. Koehler, J. K. Ord, and R. D. Snyder, "Forecasting with exponential smoothing: the state space approach." Springer Science & Business Media, 2008.

[45] V. Assimakopoulos and K. Nikolopoulos, "The theta model: a decomposition approach to forecasting," *International journal of forecasting*, vol. 16, no. 4, pp. 521–530, 2000.

[46] A. M. D. Livera, R. J. Hyndman, and R. D. Snyder, "Forecasting time series with complex seasonal patterns using exponential smoothing," *Journal of the American Statistical Association*, vol. 106, no. 496, pp. 1513–1527, 2011.

[47] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2018.

[48] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features," *arXiv preprint arXiv:1706.09516*, 2017.

[49] J. Trapero, N. Kourentzes, and R. Fildes, "On the identification of sales forecasting models in the presence of promotions," *Journal of the Operational Research Society*, vol. 66, p. 299–307, 2014.

[50] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "Deepar: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020.

[51] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-beats: Neural basis expansion analysis for interpretable time series forecasting," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=r1ecqn4YwB

[52] A. Borovykh, S. Bohte, and K. Oosterlee, "Conditional time series forecasting with convolutional neural networks," in *Lecture Notes in Computer Science/Lecture Notes in Artificial Intelligence*, 2017, pp. 729–730.

[53] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[54] A. Botchkarev, "Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology," *arXiv preprint arXiv:1809.03006*, 2018.

[55] R. Godahewa, C. Bergmeir, G. I. Webb, R. J. Hyndman, and P. Montero-Manso, "Monash time series forecasting archive," *arXiv preprint arXiv:2105.06643*, 2021.