DeepPIPE: A Distribution-free Uncertainty Quantification Approach for Time Series Forecasting

Bin Wang[a,c], Tianrui Li[a,b,*], Zheng Yan[c], Guangquan Zhang[c], Jie Lu[c]

[a] *Institute of Artificial Intelligence, School of Information Science and Technology, Southwest Jiaotong University, Chengdu 611756, China*
[b] *National Engineering Laboratory of Integrated Transportation Big Data Application Technology, Southwest Jiaotong University, Chengdu 611756, China*
[c] *Centre for Artificial Intelligence, University of Technology Sydney*

Abstract

Time series forecasting is a challenging task as the underlying data generating process is dynamic, nonlinear, and uncertain. Deep learning such as LSTM and auto-encoder can learn representations automatically and has attracted considerable attention in time series forecasting. However, current approaches mainly focus on point estimation, which leads to the inability to quantify uncertainty. Meantime, existing deep uncertainty quantification methods suffer from various limitations in practice. To this end, this paper presents a novel end-to-end framework called deep prediction interval and point estimation (DeepPIPE) that simultaneously performs multi-step point estimation and uncertainty quantification for time series forecasting. The merits of this approach are threefold: first, it requires no prior assumption on the distribution of data noise; second, it utilizes a novel hybrid loss function that improves the accuracy and stability of forecasting; third, it is only optimized by back-propagation algorithm, which is time friendly and easy to be implemented. Experimental results demonstrate that the proposed approach achieves state-of-the-art performance on three real-world datasets.

Keywords: Deep Learning, Uncertainty Quantification, Time Series, Forecasting

1. Introduction

Time series are observations of a dynamic system collected sequentially over time [1]. To forecast the future values, historical t-step observations $\mathbf{X}_{1:t} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t] \in R^{t \times d}$ with each $\mathbf{x}_i \in R^d$ and d being the feature dimensions, are analyzed to build a model that depicts the underlying dynamic of the nonlinear system. The model is then utilized to predict the most possible series $\hat{\mathbf{y}}_{t+1:t+S} = [\hat{y}_{t+1}, \hat{y}_{t+2}, ..., \hat{y}_{t+S}] \in R^S$ to approximate the ground truth series $\mathbf{y}_{t+1:t+S}$ in the future S steps, which can be formally described as below:

$$\hat{\mathbf{y}}_{t+1:t+S} = \underset{\mathbf{y}_{t+1:t+S}}{\operatorname{argmax}} Pr(\mathbf{y}_{t+1:t+S} | \mathbf{X}_{1:t}) \tag{1}$$

or denoted by the machine learning model $f(\cdot)$:

$$\hat{\mathbf{y}}_{t+1:t+S} = f(\mathbf{X}_{1:t}) \tag{2}$$

The forecasted $\hat{\mathbf{y}}_{t+1:t+S}$ is termed as *point estimation*, a.k.a, single-value forecasting. Previous studies of deep learning for time series forecasting have achieved great success, especially on point estimation problems [2, 3, 4, 5, 6, 7]. Nonetheless, point estimation omits uncertainty, which cannot provide sufficient information for safe and optimal decision-making [8]. For example, crowd prediction and management without uncertainty quantification will lack warning information for managers to take early deployment measures [9]. In financial trading, if a prediction algorithm only forecasts a single value, it will not be sufficient to derive a safe trading policy in that the market can be high-risk and volatile [6]. Concerning many automated systems, various safety issues may arise beyond expectation [10], hence letting the automated systems sense high uncertain scenarios to stop operations and seek experts for intervention will be the safe strategy. *Uncertainty quantification* has therefore received increased attention in the research community [11, 12]. Through the uncertainty quantification method, researchers proposed LSTM-based auto-encoder scheme for anomaly detection [13]. In order to formalize uncertainty quantification in the scenarios of time series forecasting, let us first consider the equation of one-step point estimation by setting $n = 1$ in formula (2) and it achieves:

$$\hat{y}_{t+1} = f(\mathbf{X}_{1:t}) \tag{3}$$

Although $\hat{y}_{t+1}$ is regarded as the prediction for the ground truth y_{t+1}, a basic view of statistical machine learning is that y_{t+1} is difficult to be forecasted perfectly and there often exists irreducible noise ϵ_{t+1} in y_{t+1}, which is expressed as:

$$y_{t+1} = f(\mathbf{X}_{1:t}) + \epsilon_{t+1} \tag{4}$$

The philosophy of uncertainty quantification is to predict a prediction interval (PI) $[\hat{y}^L_{t+1}, \hat{y}^U_{t+1}]$ to bound y_{t+1} to satisfy

$$Pr_{t+1} = Pr[\hat{y}^L_{t+1} \leq y_{t+1} \leq \hat{y}^U_{t+1}] \geq P_c, \tag{5}$$

*Corresponding author
Email address: trli@swjtu.edu.cn (Tianrui Li)

where P_c is the predefined confidence level and Pr_{t+1} is called prediction interval coverage probability (PICP). It can be inferred that if $\hat{y}^U_{t+1}$ is extremely large and in the meantime $\hat{y}^L_{t+1}$ is excessively small, Pr_{t+1} can be always equal to 100%, which is nonsensical for users. An admirable uncertainty quantification is when formula (5) holds, smaller interval width, i.e., $\hat{y}^U_{t+1} - \hat{y}^L_{t+1}$, is better. The equations (3), (4) and (5) can be extended for the other time step $t + S$.

This study aims to utilize one deep learning model to forecast multi-step point estimation and prediction interval simultaneously in time series scenarios. The contributions of this paper are summarized as below:

1. It proposes an end-to-end deep learning method to solve sequential point estimation and uncertainty quantification simultaneously for time series forecasting. More formally, it is an integrated model $g(\cdot)$ with the specified P_c to predict $\hat{\mathbf{y}}, \hat{\mathbf{y}}^L$ and $\hat{\mathbf{y}}^U \in R^S$, i.e., $\hat{\mathbf{y}}_{t+1:t+S}, \hat{\mathbf{y}}^L_{t+1:t+S}, \hat{\mathbf{y}}^U_{t+1:t+S} = g(\mathbf{X}_{1:t}|P_c)$.
2. It is distribution-free for modeling ϵ_{t+1}, which means it implicitly learns the distribution of the irreducible noise ϵ_{t+1} instead of imposing an explicit distribution assumption. This property makes it very consistent with the real-world environment where ϵ_{t+1} usually does not obey mathematically analytic distribution like Gaussian distribution, which has been validated in our experiments.
3. It is trained by a designed hybrid loss function. We experimentally display that it can achieve better generalization accuracy and variance by training with that loss function. Moreover, it only utilizes back-propagation for training, which makes it friendly implemented by popular deep learning libraries, such as TensorFlow and PyTorch.

The proposed model is also flexible and can be enhanced by being combined with popular modules such as attention mechanisms and residual connections.

The rest of the paper is organized as follows. In Section II, we discuss the related works; in Section III, we introduce the proposed methodology DeepPIPE; in Section IV, we show the experimental results on three real-world datasets; and finally, some conclusions and future works are given in Section V.

2. Related works

Time series forecasting in nature is a problem of dynamic modeling, which usually takes on nonlinear or chaotic properties. Detecting chaotic behavior from a time series has been proposed [14]. Although it has been found that chaotic systems are ubiquitous, most of them do not have explicit dynamical equations and can be only modeled through the observed time series [15]. In recent years, with deep learning reaching fever pitch in data science, various types of deep neural networks were introduced to model time series forecasting [16, 17, 18, 19, 20]. Most of these studies focused on point estimation, and have devised effective network architectures such as sequence-to-sequence [21], attention mechanism [22], embedding [23], which can shed enlightenment on their combination with uncertainty quantification.

Uncertainty quantification is constantly a hot research topic. A pioneering neural-network-based method is MVE, which utilized the neural network to predict mean and variance to parameterize Gaussian distribution [24]. This method has been progressively combined with advanced network architectures such as echo-state network [25] and deep forward neural networks [26]. However, MVE relies on a strong assumption that the irreducible noise ϵ_{t+1} obeys Gaussian distribution and performs poorly when the assumption is too inconsistent with reality. To this end, two distribution-free neural-network-based methods are especially proposed. Method LUBE constructed a non-convex loss function and hence used simulated annealing for optimization [27]. Method DeepHQ is optimized based on gradient descent-based algorithms [28]. Nonetheless, both LUBE and DeepHQ can only implement prediction interval without point estimation and conduct regression for cross-sectional data rather than time-series data, which cannot be directly applied to time series scenarios [29]. A Bayesian method was proposed for modeling time series [30]. However, Bayesian methodology required Markov chain Monte Carlo (MCMC) or variational inference (VI) as the optimization algorithm, and when combined with deep learning as Bayesian deep learning (BDL) [31], it causes an efficient bottleneck of implementation. Non-Bayesian deep learning methods hence have been proposed to address the issue of BDL. A non-Bayesian study mathematically proved that Monte Carlo Dropout (MC-Dropout) could be operated as Bayesian approximation [32] hence can be taken as a solution based on the back-propagation algorithm. The following research extended MC-Dropout for modeling time series based on recurrent neural networks [33]. Another branch, i.e., non-Bayesian approach, is to design novel loss function to depicts uncertainty quantification when modeling time series. Method DeepMVE, inspired by MVE and sequence-to-sequence architecture, was proposed based on the assumption that the irreducible noise ϵ_{t+s} obeyed Gaussian distribution [34]. However, this method may overestimate the lower and upper bounds when the actual noise is non-Gaussian. DE-RNN [12] was proposed which did not require the distribution assumption of ϵ_{t+s}, but DE-RNN are limited in that: 1) It needs discretization to transform regression as a classification problem, which asks users to denote the interval width at the expense of information loss; 2) It adopts the Monte Carlo procedure for multi-step forecasting, which is time-consuming; 3) It evaluates the aspect of PICP alone but ignores the quality of interval width.

The DeepPIPE thus is proposed to bridge the current research gap for time series forecasting. To the best of our knowledge, it is the first methodology that can overcome all the limitations existing in previous studies, including DeepHQ, DeepMVE, LUBE, Bayesian deep learning, MC-Dropout, and DE-RNN. Its key characteristics are summarized in Table 1.

3. Proposed methodology

DeepPIPE is based on Encoder-Decoder (a.k.a sequence-to-sequence) [21]. Both encoder and decoder consist of LSTM which can address gradient vanishing problem compared with

Table 1: Summarized key differences between DeepPIPE and previous methods. PE and PI represents point estimation and prediction interval, respectively.

V.S.		**DeepPIPE**
DeepHQ	PI	PI & PE
DeepMVE	distribution assumption	distribution free
LUBE	simulated annealing	back propagation
BDL	Bayesian inference	non-Bayesian method
MC-Dropout	Monte Carlo	non-Monte Carlo
DE-RNN	need discretization	non-discretization

vanilla RNN [35]. The formula of the LSTM cell is:

$$\begin{aligned} i_t &= \sigma\big(\mathbf{x}_t U^i + h_{t-1} W^i\big) \\ f_t &= \sigma\big(\mathbf{x}_t U^f + h_{t-1} W^f\big) \\ o_t &= \sigma\big(\mathbf{x}_t U^o + h_{t-1} W^o\big) \\ \tilde{C}_t &= \tanh\big(\mathbf{x}_t U^z + h_{t-1} W^z\big) \\ C_t &= \sigma\big(f_t \odot C_{t-1} + i_t \odot \tilde{C}_t\big) \\ h_t &= tanh(C_t) \odot o_t \end{aligned} \tag{6}$$

where

- $\mathbf{x}_t$ defines the input vectors at the timestep t,
- i_t defines the input gates, f_t defines the forget gates and o_t defines the output gates,
- $\tilde{C}_t$ defines the temporary hidden state which is computed based on the current input and the previous hidden state. C_t defines the internal memory.
- h_t defines the output hidden states, computed by multiplying the memory with the output gate. h_{t-1} defines the hidden states at the previous time step,
- W and U define the learnable weights,
- σ and $tanh$ define sigmoid and tanh activation function respectively,
- $\odot$ defines the element-wise product.

The encoder $E(\cdot)$ first encodes input series $\mathbf{X}_{1:t}$ to context vectors $\mathbf{c}$:

$$\mathbf{c} = E(\mathbf{X}_{1:t}; \theta_1) \tag{7}$$

$\mathbf{c}$ is then transferred to form the initial state of decoder $D(\cdot)$ which decodes $\mathbf{c}$ and consequently generates the sequential prediction interval and point estimation simultaneously:

$$\hat{\mathbf{y}}_{t+1:t+S}, \hat{\mathbf{y}}^L_{t+1:t+S}, \hat{\mathbf{y}}^U_{t+1:t+S} = D(\mathbf{c}; \theta_2) \tag{8}$$

where θ_1 and θ_2 are learnable parameters. For each time step $t+s$, $\hat{y}_{t+s}, \hat{y}^L_{t+s}$ $y_{t+s}$and $\hat{y}^U_{t+s}$ are calculated through below transformation:

$$\begin{aligned} \hat{y}_{t+s} &= h_{t+s} U^p + B^p \\ \hat{y}^L_{t+s} &= h_{t+s} U^l + B^l \\ \hat{y}^U_{t+s} &= h_{t+s} U^u + B^u \end{aligned} \tag{9}$$

where U and B are learnable parameters shared among total time steps. The architecture of DeepPIPE is illustrated in Fig. 1, where the *Assumptions* and *Hybrid loss function* are explained in the next section.

3.1. Hybrid loss function

In order to equip DeepPIPE with the capability of point estimation and prediction interval, the training loss function L consists of two parts, where L_{PE} is to penalize the loss of point estimation and L_{PI} is to penalize the loss of prediction interval. The hyper-parameter β balances the influence between point estimation and prediction interval. Formally, it is defined as below:

$$L = \beta L_{PE} + L_{PI} \tag{10}$$

Particularly, L_{PE} is denoted as below:

$$L_{PE} = \frac{1}{NS} \sum_{n=1}^{N} \sum_{s=1}^{S} v_s \cdot |y^{(n)}_{t+s} - \hat{y}^{(n)}_{t+s}| \tag{11}$$

where N is the training batch size, S is the number of forecasted time steps, v_s is a weight factor for the time step s, $y^{(n)}_{t+s}$ and $\hat{y}^{(n)}_{t+s}$ are the ground truth and our prediction at the timestep $t+s$, respectively.

To learn prediction interval, L_{PI} is inspired from the study [28] and constructed as below:

$$L_{PI} = \lambda \frac{N}{(1-P_c)P_c} max(0, (P_c - PICP))^2 + MPIW \tag{12}$$

The advanced intuitions of the formula (12) is, during training process, if PICP is lower than the expected confidence level P_c, a penalty loss will be imposed through the operation $max(0, (P_c - PICP))^2$. The width of the prediction interval is meanwhile accomplished as small as possible by plus MPIW. N corresponds to the training batch size. The expression $\lambda \frac{N}{(1-P_c)P_c}$ is the derivative which can be referred to the original paper [28] for the details. In particular, λ is a weighting hyper-parameter that controls the loss balance between PICP and MPIW. To address multi-step forecasting in time series scenarios, PICP is denoted as:

$$PICP = \frac{c}{N \cdot S} \tag{13}$$

where $c = \sum_{n=1}^{N} \sum_{s=1}^{S} b^{(n)}_{t+s}$ is the total number of *captured data points*. Specially, we say a forecasting value $y^{(n)}_{t+s}$ (value of the nth training sample at the time step $t+s$) captured, shall satisfy:

$$b^{(n)}_{t+s} = \begin{cases} 1, & \text{if} \quad \hat{y}^{(n),L}_{t+s} \le y^{(n)}_{t+s} \le \hat{y}^{(n),U}_{t+s} \\ 0, & \text{else.} \end{cases}$$

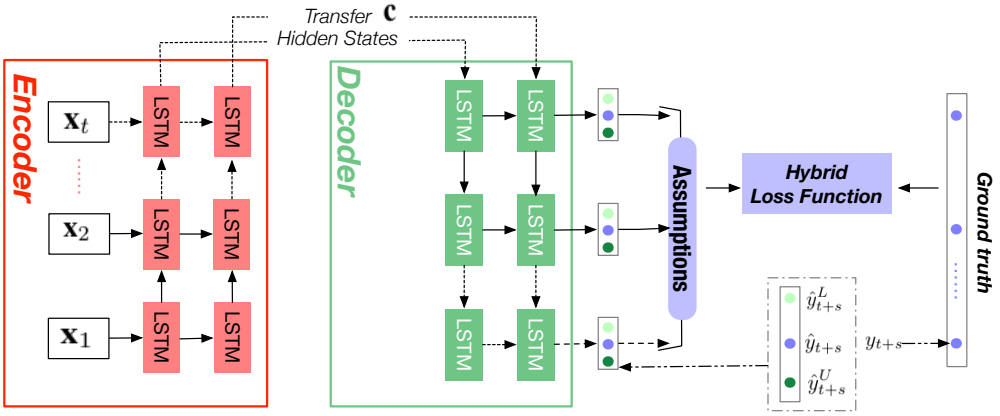


Figure 1: Framework of DeepPIPE. The encoder $E(\cdot)$ first encodes input series $\mathbf{X}_{1:t}$ to hidden states $\mathbf{c}$ which is transferred to form the initial state of decoder $D(\cdot)$. $D(\cdot)$ then decodes $\mathbf{c}$ and consequently generates the sequential prediction interval and point estimation simultaneously. The proposed hybrid loss function based on related assumptions is adopted to train the whole network.

where $\hat{y}_{t+s}^{(n),L}$ and $\hat{y}_{t+s}^{(n),U}$ are the forecasted lower and upper bound for the the nth training sample at the time step $t+s$. MPIW is defined as:

$$MPIW = \frac{1}{c}\sum_{n=1}^{N}\sum_{s=1}^{S} w_s \cdot (\hat{y}_{t+s}^{(n),U} - \hat{y}_{t+s}^{(n),L}) \cdot b_{t+s}^{(n)} \quad (14)$$

where w_s is a weight factor for the time step s. Note that $\frac{1}{c}$ indicates that only the captured forecasting values are considered for the calculation of MPIW. Due to the existence of operation $max(\cdot)$ in (12), applying the Boolean $b_{t+s}^{(n)}$ in formulas (13) and (14) can cause the L_{PI} not to be differentiable for back-propagation algorithm. To this end, $b_{t+s}^{(n)}$ is substituted with a softened version as below:

$$b_{t+s}^{(n)} = \sigma(s \cdot (\hat{y}_{t+s}^{(n),U} - y_{t+s}^{(n)})) \cdot \sigma(s \cdot (y_{t+s}^{(n)} - \hat{y}_{t+s}^{(n),L})) \quad (15)$$

where σ is the sigmoid activation function and s is a hyper-parameter factor for softing. Accordingly, the softened $b_{t+s}^{(n)}$ is adopted in formulas (13) and (14).

3.1.1. Explanation of assumptions

The formula (12) in the original study [28] does not consider the time dimension; hence, it can not be applied directly for modeling time series. To address this problem, we expand it to be compatible with the sequence-to-sequence model based on assumptions as follows.

- Different time steps $t+s$ have an equal weight factor as one. That is to say, in formula (11) and (14), in this research, v_s and w_s are set to 1.
- Although this research only considers the single variable forecasting, when DeepPIPE is adopted for multiple variables forecasting, the weight factors of different variables can be set according to users' experience.

3.2. Algorithm

Algorithm 1 outlines the DeepPIPE training procedure. We first construct the data pairs from historical time-series data. A sliding window with sliding step one is applied to the raw time series. For the timestep i, we pair $\mathbf{X}_{1:t}$ and $\mathbf{y}_{t+1:t+S}$ as a training sample, which is shown by lines 1-6. The model is then trained through batch gradient descending and back-propagation algorithm to minimize the loss function L, which is illustrated by lines 7-11.

Algorithm 1: DeepPIPE Training Algorithm

Input : Historical input time series: $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_I$;
Historical target time series: $y_1, y_2, ..., y_I$;
Input window size t;
Output window size S;
Confidence level P_c (0.9 in our experiments).
Output: Learned DeepPIPE model
// Format the training dataset by sliding window
1 $\mathcal{D} \leftarrow \emptyset$
2 **while** *all available timestamp i* $(0 \le i \le I-t-S)$ **do**
3 $\quad \mathbf{X}_{1:t} = [\mathbf{x}_{1+i}, \mathbf{x}_{2+i}, ..., \mathbf{x}_{t+i}]$
4 $\quad \mathbf{y}_{t+1:t+S} = [y_{t+1+i}, y_{t+2+i}, ..., y_{t+S+i}]$
5 $\quad$ put a data sample $(\mathbf{X}_{1:t}, \mathbf{y}_{t+1:t+S})$ into $\mathcal{D}$
6 **end**
// Split dataset $\mathcal{D}$ into training set $\mathcal{D}_1$, validation set $\mathcal{D}_2$, and test set $\mathcal{D}_3$.
// Train
7 Initialize all trainable parameters θ_1, θ_2 in DeepPIPE.
8 **repeat**
9 $\quad$ randomly select a batch of samples $\mathcal{B}$ from $\mathcal{D}_1$
10 $\quad$ update parameters θ_1, θ_2 by minimizing the loss function with $\mathcal{B}$
11 **until** *stopping criteria are met*;

4. Experiments

In this section, we first introduce the datasets, experimental settings, evaluation metrics, evaluation metrics, and baselines. Afterward, we conduct a performance analysis for the experimental results.

4.1. Datasets

Three open real-world datasets *appliances energy* and *air quality* are used in our experiments. Fig. 2 displays examples of three time-series datasets. The detail of each dataset is introduced as follows.

4.1.1. Multivariate datasets

Appliances energy UCI dataset was collected for 4.5 months with 10-min frequency, having a total 19737 records [36][1]. One-hot encoding is utilized to transform the category variables (e.g., DayOfWeek and TimeOfDay), and the final feature dimensions are 36 for each time step, and the dimension *appliances energy consumption* is the target variable.

Air quality dataset was collected for the air quality forecasting research [37][2]. This dataset is collected from 00:00 01/May/2014 to 22:00 30/Apr/2015 at 1-hour frequency, a totally of 8759 records. Unfortunately, there are serious missing value problems in the majority of air monitoring stations. We finally select the station *Yangjiang* as our data source, which has the least missing values. We have six pollutants including $PM_{2.5}$, PM_{10}, NO_2, CO, O_3, and SO_2 to build previous feature series and take the future $PM_{2.5}$ as the target variable.

4.1.2. Univariate dataset

Brent oil price dataset contains daily Brent oil prices from 17th of May 1987 until the 30th of September 2019 at 1-day frequency [3], totally 8216 records.

4.2. Experimental settings

4.2.1. Preprocessing

We use min-max normalization to normalize the continuous features into [0, 1] and utilize linear interpolation for missing data imputation. In the evaluation, we rescale the predicted values back to the normal scale.

[1]Data link: https://archive.ics.uci.edu/ml/datasets/Appliances+energy +prediction. We investigated that there was a technical misuse in the original research paper [36]. In the source codes, the authors shuffled training/testing data partitioning, which was a misuse of time-series data. We split the data in chronological order without shuffle operation. Hence the MAE in our experimental results and in the original paper are not comparable because of forecasting implemented on the different test datasets.

[2]Data link: http://urban-computing.com/data/Data-1.zip

[3]Data link: https://www.kaggle.com/mabusalah/brent-oil-prices

4.2.2. Hyper-parameter settings

For a fair comparison, all deep learning methods are configured with the same hyper-parameter as follows. In particular, we set $\lambda = 2$, $\beta = 1.5$, $s = 160$ and $P_c = 0.9$ in formula (12) and (15). We set the hidden nodes of each layer as 64. We set the batch size as 256. We set the time window size of input feature series as $t = 12, 6, 7$ for datasets appliances energy, air quality, and Brent oil price respectively. For the single forecasting, we set $S = 1$, which indicates to predict the next value once time. For the multi-step forecasting, we set $S = 3$, which means to forecast the following three values once time. Financial time-series prediction is a challenging task, and long-term forecasting is often unhelpful, hence 1-step-ahead setting has become the first consideration [38]. Our experiments follow the 1-step-ahead setting for the oil price prediction. All datasets are formatted according to the Algorithm 1. The resultant dataset for training, validation, and test are outlined in Table 2.

4.2.3. Optimization method

Adam [39] is adopted as the optimization method to learn the parameters with learning rate 0.001, and early-stopping is implemented on the validation set to avoid overfitting.

4.2.4. Experimental environment

All models are trained on a server with Quadro P4000 GPU. The programming environment is Python 3.6 and Tensorflow.

Table 2: Information of three real-world datasets. 1/3-step prediction indicate that datasets are formatted for *n*-step-ahead prediction.

Dataset	Feature dimensions	Size training	validation	test
1-step prediction				
Brent Oil Price	1	5743	1228	1229
Appliances Energy	36	13312	1480	4931
Air Quality	6	7089	788	876
3-step prediction				
Appliances Energy	36	13311	1479	4931
Air Quality	6	7087	788	876

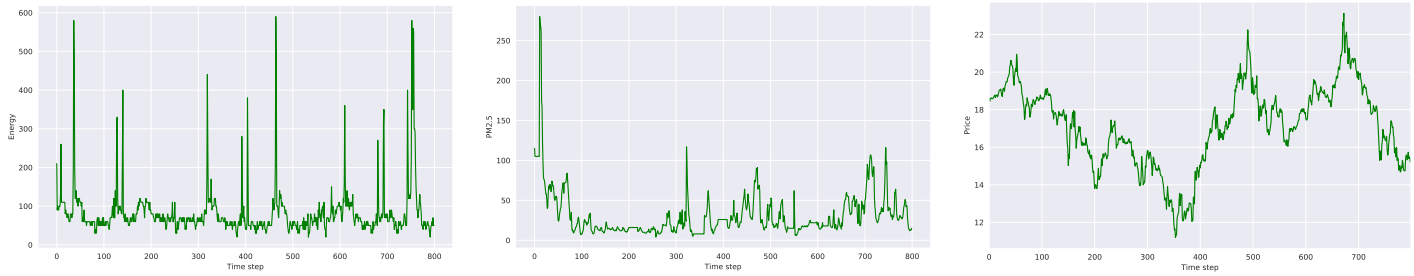
4.3. Evaluation metrics

4.3.1. Point estimation measurement

During test, MAE is calculated to evaluate the predictive error of point estimation as below:

$$MAE = \frac{1}{NS}\sum_{n=1}^{N}\sum_{s=1}^{S}|y_{t+s}^{(n)} - \hat{y}_{t+s}^{(n)}| \tag{16}$$

where $y_{t+s}^{(n)}$ and $\hat{y}_{t+s}^{(n)}$ are the ground truth and the prediction for the test sample n at the timestep $t + s$, respectively.



(a) Clear cycle in appliances energy dataset (b) Numerous abrupt points in air quality dataset (c) Complicated dynamics in oil price dataset

Figure 2: Examples of three time-series datasets. Each dataset illustrates the first 800-step time series.

4.3.2. Uncertainty quantification measurement

PICP in formula (13) and MPIW in formula (14) are taken as the measurements of uncertainty quantification.

4.4. Baselines

As mentioned that LUBE [11] requires heuristic algorithms and BDL [31] requires variational inference for optimization. Both two methods not only cannot be seamlessly integrated with popular gradient-based frameworks, such as Tensorflow. Although MC-Dropout [33] only needs back-propagation during training, it requires Monte Carlo by repeating forward calculation for thousands of times to inference prediction interval and point estimation. Moreover, the above methods are time-consuming and do not show significant improvement compared to the MVE method according to previous researches [11, 26]. In this respect of pragmatism, their shortcomings have already made them lose their advantage compared with DeepPIPE, and thus, we compare DeepPIPE with below baselines.

- **DeepMVE** adopts the Encoder-Decoder architecture extended from MVE model [34]. Although MVE loss can be easily integrated into deep learning for uncertainty quantification, a strong assumption is that the ground truth $y_{t+s} \sim N(\hat{y}_{t+s}, \sigma_{t+s})$, and the interval width can be calculated by looking up the z-score table of Gaussian distribution according to the pre-defined P_c. In our experiments, IW_{t+s} is calculated by $2 * 1.64 * \sigma_{t+s}$ because of P_c set to 0.9. The learning process is essentially by maximum likelihood estimation. It has the same hyper-parameter settings with DeepPIPE.

- **DeepHQ** has the same hyper-parameter settings with DeepPIPE except that its loss function only includes prediction interval part L_{PI}. It can also be regarded as a sequence-to-sequence extension from the previous study [28]. This baseline aims to illustrate the effectiveness of the designed hybrid loss function, which optimizes L_{PE} and L_{PI} simultaneously.

- **ShallowPIPE** has the same hyper-parameter settings with DeepPIPE except only having one hidden layer. This baseline aims to show the influence of the layer number when compared with DeepPIPE.

Since weights initialization can influence the performance of deep models. In our experiments, each deep model was trained up to five times with different random seeds. The test results are reported with the expression as *mean* $\pm$ *standard deviation*. In addition, non-deep learning baselines include:

- **ARIMA** (Autoregressive Integrated Moving Average) is a popular method for univariate time series forecasting. It was implemented by *statsmodels.tsa.arima_model.ARIMA*.

- **VAR** (Vector Auto Regression) is one of the most commonly used parametric methods for time series forecasting. It was implemented by *statsmodels.tsa.api.VAR*.

- **SVR** (Support Vector Regression) is a category of SVM for regression tasks. It was implemented by the Python package *sklearn.svm.SVR*.

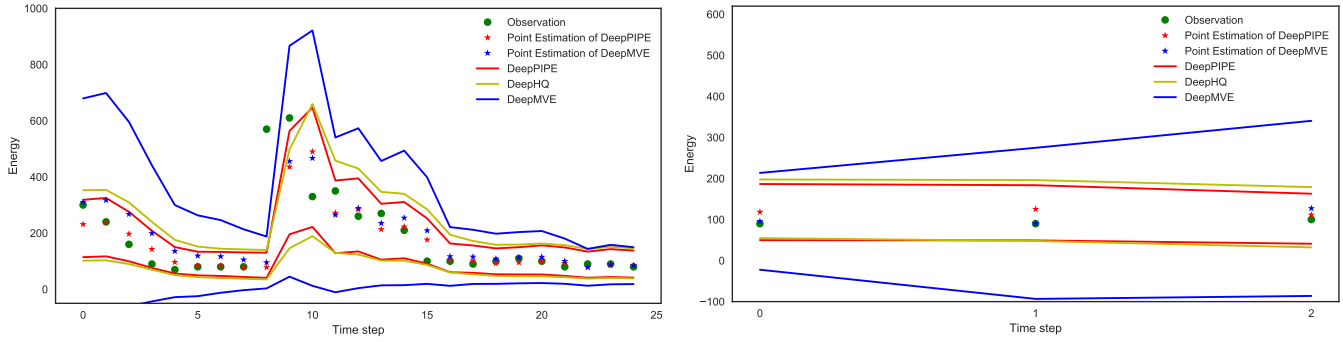
- **GBDT** (Gradient Boosting Decision Tree) is a popular boosting method that ensembles a set of weak decision trees in order. It was implemented by the Python package *sklearn.ensemble.GradientBoostingRegressor*.

4.5. Performance analysis

Table. 4 and 5 report the experimental results. Fig.3 and Fig.4 exhibit the visualization of forecasting examples. We conduct analysis from aspects as follows.

4.5.1. Performance of point estimation

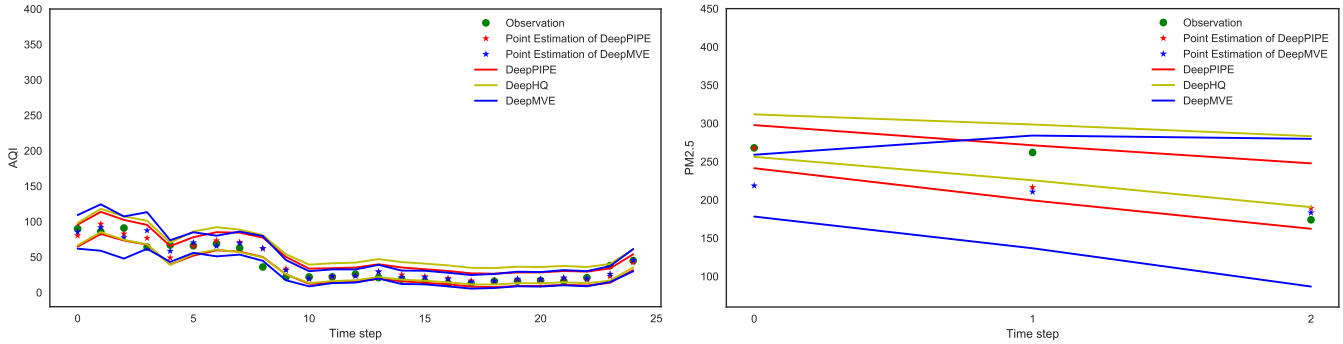
Our top-line conclusions are that DeepPIPE almost achieved the best MAE measurement on three datasets for both 1-step and 3-step forecasting. We can see that on appliances energy dataset, it achieved 34.99$\pm$0.61 for 1-step prediction and 41.77$\pm$1.03 for 3-step prediction. For the air quality dataset, it achieved 5.25$\pm$0.09 for 1-step prediction and 7.69$\pm$0.03 for a 3-step prediction. DeepHQ has no ability for point estimation, so the relevant results were left out. For the 1-step forecasting of Brent oil price, ARIMA achieved the best 1.14$\pm$0.00, whereas DeepPIPE attained 1.22$\pm$0.17. The result is not impossible because the financial dynamic is highly complicated, and the classic baseline ARIMA can sometimes be strong and perform better. It is noticeable that DeepPIPE has obvious advantages when modeling prediction interval, which is analyzed as below.



(a) 25 test examples of 1-step-ahead forecasting.

(b) One example of end-to-end 3-step-ahead forecasting

Figure 3: Predictive examples of DeepPIPE on appliances energy dataset. DeepPIPE achieved better MPIW significantly than DeepMVE.



(a) 25 test examples of 1-step-ahead forecasting.

(b) One example of end-to-end 3-step-ahead forecasting

Figure 4: Forecasting examples of DeepPIPE on air quality dataset. DeepPIPE achieved better MPIW significantly than DeepMVE.

4.5.2. *Performance of prediction interval*

All methods have achieved the predefined confidence level of P_c, which indicated that all PICPs were greater than 0.9. However, under this premise, the performance of MPIW had obvious distinctions. Methods like ARIMA, VAR, GBDT, DeepMVE significantly overestimated MPIW. Our explanation for this was that these models assumed that the ground truth obeyed Gaussian distribution, which might be far different from the realistic data distribution. The observations that are consistent with our explanation are that DeepHQ, ShallowPIPE, and DeepPIPE, which all are distribution-free, accomplished smaller MPIW. Notably, DeepPIPE accomplished the best MPIW performance 106.96±2.39 (1-step) and 116.52±1.69 (3-step) on appliances energy dataset, 22.79±0.47 (1-step) and 29.95±0.12 (3-step) on air quality dataset and 3.26±0.28 on Brent oil price dataset. The results of SVR were omitted, since *sklearn.svm.SVR* has not provided the function for the prediction interval.

4.5.3. *Performance of stability*

Deep models trained with various initialized parameters can lead to different forecasting results. To address the randomness problem, we implemented each deep model with different random seeds up to five times and calculated the standard deviation expressed via $\pm X$ to reflect the forecasting stability. In our experiments, DeepPIPE achieved the least standard deviation, and it illustrated that DeepPIPE had a stable generalization. Although methods including ARIMA, VAR, SVR, and GBDT, were configured by certain hyper-parameters leading to deterministic solutions, thus the resultant standard deviations equal to zeros, they usually have poor performances on MAE and MPIW.

4.5.4. *Effect of hybrid loss function*

On appliances energy dataset for 1-step forecasting, DeepHQ achieved MPIW as 115.75±5.71 while DeepPIPE achieved smaller MPIW as 106.96±2.39. For 3-step forecasting, DeepHQ achieved MPIW as 119.20±3.82 while DeepPIPE achieved smaller MPIW as 116.52±1.69. For the air quality dataset for 1-step forecasting, DeepHQ achieved MPIW as 23.71±0.61 while DeepPIPE achieved smaller MPIW as 22.79±0.47. For 3-step forecasting, DeepHQ achieved MPIW as 31.02±0.57 while DeepPIPE achieved smaller MPIW as 29.95±0.12. On the Brent oil price dataset, DeepHQ achieved MPIW as 3.72±0.12 while DeepPIPE achieved smaller MPIW as 3.26±0.28. These observations indicated that combining L_{PI} and L_{PE} rather only L_{PE} as loss function indeed improved the performance of prediction interval. Our explanation for this was that it benefited from the multi-task learning, which can promote consistency and coherence for point estimation and prediction interval.

4.5.5. Effect of deep learning

By comparing DeepPIPE with ShallowPIPE, we can see that stacked two layers helped DeepPIPE improve its performance for both point estimation and prediction interval on three datasets. For example, on appliances energy dataset, ShallowPIPE achieved MAE as 36.31±0.85, MPIW as 113.54±6.70, while DeepPIPE achieved smaller MAE as 34.99±0.61, MPIW as 106.96±2.39. The experimental results indicated the proposed DeepPIPE was compatible with the philosophy of deep learning. We believe its performance can be improved further when combined with other advanced deep learning techniques, such as attention mechanism and memory network.

4.5.6. The advantage of distribution-free assumption

As shown in Fig. 3 and Fig. 4, a noteworthy observation was that DeepMVE forecasted the point estimation located at the central location of the prediction interval, which was the consistent result of the Gaussian assumption. However, the point estimation of DeepPIPE does not necessarily locate at the central location between the prediction interval. This observation proves the superiority of DeepPIPE to model sensible distribution from the observed data.

5. Conclusions and future works

In this paper, we proposed a deep learning approach called DeepPIPE for time series forecasting. A novel loss function has been designed to cast the problems as multi-task learning, thus makes DeepPIPE be capable of implementing prediction interval and point estimation simultaneously. We have qualitatively analyzed its superiority over previous models, that is, DeepPIPE can be trained by back-propagation algorithm, which can be seamlessly deployed in popular deep learning libraries and is an admirable hallmark compared with LUBE, BDL, and MC-Dropout, of which they require heuristic algorithms, variational inference, and Monte Carlo, respectively. Furthermore, different from DeepMVE, it can automatically learn the distribution of ϵ_{t+s} without prior distribution assumptions. It also overcomes the mentioned drawbacks of DE-RNN. To the best of our knowledge, this is the first methodology that can overcome all the limitations existing in previous studies, including DeepHQ, DeepMVE, LUBE, Bayesian deep learning, MC-Dropout, and DE-RNN. We also quantitatively show its considerable performance and forecasting stability on three real-world datasets. For the future works, we will explore DeepPIPE in real-world applications, such as energy forecasting [40] and electrocardiogram analysis [41]. We also aim to incorporate advanced techniques, including attention mechanisms and memory networks, to further improve its performance.

6. Acknowledgments

This work was supported by the Natural Science Foundation of China (Nos. 61773324, 61573292, 61976247) and the Australian Research Council (No. DP190101645).

References

[1] S. Choudhary, G. Hiranandani, S. K. Saini, Sparse decomposition for time series forecasting and anomaly detection, in: SIAM International Conference on Data Mining, 2018, pp. 522–530.

[2] J. Wang, Q. Gu, J. Wu, G. Liu, Z. Xiong, Traffic speed prediction and congestion source exploration: A deep learning method, in: International Conference on Data Mining, 2016, pp. 499–508.

[3] R. Yu, Y. Li, C. Shahabi, U. Demiryurek, Y. Liu, Deep learning: A generic approach for extreme condition traffic forecasting, in: SIAM International Conference on Data Mining, 2017, pp. 777–785.

[4] J. Zhang, Y. Zheng, D. Qi, R. Li, X. Yi, T. Li, Predicting citywide crowd flows using deep spatio-temporal residual networks, Artificial Intelligence 259 (2018) 147–166.

[5] T. Guo, A. Bifet, N. Antulov-Fantulin, Bitcoin volatility forecasting with a glimpse into buy and sell orders, in: International Conference on Data Mining, 2018, pp. 989–994.

[6] L. Zhang, C. Aggarwal, G.-J. Qi, Stock price prediction via discovering multi-frequency trading patterns, in: International Conference on Knowledge Discovery and Data Mining, 2017, pp. 2141–2149.

[7] H. Li, Y. Shen, Y. Zhu, Stock price prediction using attention-based multi-input LSTM, in: Asian Conference on Machine Learning, 2018, pp. 454–469.

[8] X. Zeng, J. Lu, Decision support systems with uncertainties in big data environments, Knowledge-Based Systems 143 (2018) 327.

[9] D. Helbing, D. Brockmann, T. Chadefaux, K. Donnay, U. Blanke, O. Woolley-Meza, M. Moussaid, A. Johansson, J. Krause, S. Schutte, et al., Saving human lives: What complexity science and information systems can contribute, Journal of Statistical Physics 158 (3) (2015) 735–781.

[10] M. Perc, M. Ozer, J. Hojnik, Social and juristic challenges of artificial intelligence, Palgrave Communications 5 (1) (2019) 1–7.

[11] A. Khosravi, S. Nahavandi, D. Creighton, A. F. Atiya, Comprehensive review of neural network-based prediction intervals and new advances, Transactions on Neural Networks 22 (9) (2011) 1341–1356.

[12] K. Yeo, I. Melnyk, N. Nguyen, E. K. Lee, DE-RNN: Forecasting the probability density function of nonlinear time series, in: International Conference on Data Mining, 2018, pp. 697–706.

[13] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, G. Shroff, Lstm-based encoder-decoder for multi-sensor anomaly detection, arXiv preprint arXiv:1607.00148.

[14] S. Kodba, M. Perc, M. Marhl, Detecting chaos from a time series, European Journal of Physics 26 (1) (2005) 205–215.

[15] Z. Liu, Chaotic time series analysis, Mathematical Problems in Engineering 2010 (2010) 1–31.

[16] Y. Zhao, Y. Shen, Y. Zhu, J. Yao, Forecasting wavelet transformed time series with attentive neural networks, in: International Conference on Data Mining, 2018, pp. 1452–1457.

[17] C. Chen, K. Li, S. G. Teo, G. Chen, X. Zou, X. Yang, R. C. Vijay, J. Feng, Z. Zeng, Exploiting spatio-temporal correlations with multiple 3d convolutional neural networks for citywide vehicle flow prediction, in: International Conference on Data Mining, 2018, pp. 893–898.

[18] A. Ziat, E. Delasalles, L. Denoyer, P. Gallinari, Spatio-temporal neural networks for space-time series forecasting and relations discovery, in: International Conference on Data Mining, 2017, pp. 705–714.

[19] M. Qiu, P. Zhao, K. Zhang, J. Huang, X. Shi, X. Wang, W. Chu, A short-term rainfall prediction model using multi-task convolutional neural networks, in: International Conference on Data Mining, 2017, pp. 395–404.

[20] D. Deng, C. Shahabi, U. Demiryurek, L. Zhu, Situation aware multi-task learning for traffic prediction, in: International Conference on Data Mining, 2017, pp. 81–90.

[21] Y. Xiao, J. Cai, Y. Yang, H. Zhao, H. Shen, Prediction of microrna subcellular localization by using a sequence-to-sequence model, in: International Conference on Data Mining, 2018, pp. 1332–1337.

[22] Y. Yuan, G. Xun, F. Ma, Y. Wang, N. Du, K. Jia, L. Su, A. Zhang, Muvan: A multi-view attention network for multivariate temporal data, in: International Conference on Data Mining, 2018, pp. 717–726.

[23] T. Chen, H. Yin, H. Chen, L. Wu, H. Wang, X. Zhou, X. Li, Tada: trend alignment with dual-attention multi-task recurrent neural networks for sales prediction, in: International Conference on Data Mining, 2018, pp. 49–58.

Table 3: Experimental results on Brent oil price dataset. DeepPIPE achieved best on MAE and MPIW.

Model	MAE	PICP	MPIW
1-step prediction			
ARIMA	1.14±0.00	1.00±0.00	3.45±0.00
SVR	1.23±0.00	-	-
GBDT	1.38±0.00	0.98±0.00	4.54±0.00
DeepMVE	1.21±0.28	0.96±0.02	4.04±0.40
DeepHQ	-	0.99±0.03	3.42±0.12
ShallowPIPE	1.18±0.21	0.99±0.01	3.38±0.33
DeepPIPE	**1.02±0.17**	1.00±0.00	**3.26±0.28**

Table 4: Experimental results on appliances energy dataset. DeepPIPE achieved best on MAE and MPIW.

Model	MAE	PICP	MPIW
1-step prediction			
ARIMA	36.17±0.00	1.00±0.00	233.05±0.00
VAR	36.91±0.00	1.00±0.00	232.21±0.00
SVR	41.06±0.00	-	-
GBDT	36.70±0.00	1.00±0.00	173.84±0.00
DeepMVE	38.52±2.35	0.95±0.02	152.33±5.61
DeepHQ	-	0.93±0.00	115.75±5.71
ShallowPIPE	36.31±0.85	0.93±0.01	113.54±6.70
DeepPIPE	**34.99±0.61**	0.93±0.00	**106.96±2.39**
3-step prediction			
ARIMA	42.68±0.00	1.00±0.00	279.85±0.00
VAR	44.23±0.00	1.00±0.00	275.39±0.00
SVR	44.67±0.00	-	-
GBDT	43.51±0.00	1.00±0.00	204.23±0.00
DeepMVE	45.53±3.21	0.95±0.02	141.97±18.06
DeepHQ	-	0.93±0.00	119.20±3.82
ShallowPIPE	48.17±1.71	0.93±0.01	123.67±2.52
DeepPIPE	**41.77±1.03**	0.93±0.01	**116.52±1.69**

[24] D. A. Nix, A. S. Weigend, Estimating the mean and variance of the target probability distribution, in: International Conference on Neural Networks, 1994, pp. 55–60.

[25] W. Yao, Z. Zeng, C. Lian, Generating probabilistic predictions using mean-variance estimation and echo state network, Neurocomputing 219 (2017) 536–547.

[26] B. Lakshminarayanan, A. Pritzel, C. Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles, in: Neural Information Processing Systems, 2017, pp. 6402–6413.

[27] A. Khosravi, S. Nahavandi, D. Creighton, A. F. Atiya, Lower upper bound estimation method for construction of neural network-based prediction intervals, Transactions on Neural Networks 22 (3) (2010) 337–346.

[28] T. Pearce, M. Zaki, A. Brintrup, A. Neely, High-quality prediction intervals for deep learning: A distribution-free, ensembled approach, in: International Conference on Machine Learning, 2018.

[29] S. Roberts, M. Osborne, M. Ebden, S. Reece, N. Gibson, S. Aigrain, Gaussian processes for time-series modelling, Philosophical Transactions of the Royal Society A 371 (1984) (2013) 20110550.

[30] M. Johnson, A. Willsky, Stochastic variational inference for bayesian time series models, in: International Conference on Machine Learning, 2014, pp. 1854–1862.

[31] M. Fortunato, C. Blundell, O. Vinyals, Bayesian recurrent neural networks, arXiv preprint arXiv:1704.02798.

[32] Y. Gal, Z. Ghahramani, Dropout as a bayesian approximation: Representing model uncertainty in deep learning, in: International Conference on Machine Learning, 2016, pp. 1050–1059.

[33] L. Zhu, N. Laptev, Deep and confident prediction for time series at uber, in: International Conference on Data Mining Workshops, 2017, pp. 103–110.

[34] B. Wang, J. Lu, Z. Yan, H. Luo, T. Li, Y. Zheng, G. Zhang, Deep uncertainty quantification: A machine learning approach for weather forecasting, in: International Conference on Knowledge Discovery and Data Mining, 2019, pp. 2087–2095.

[35] J. Wang, V. W. Zheng, Z. Liu, K. C.-C. Chang, Topological recurrent neural network for diffusion prediction, in: International Conference on Data Mining, 2017, pp. 475–484.

[36] L. M. Candanedo, V. Feldheim, D. Deramaix, Data driven prediction models of energy use of appliances in a low-energy house, Energy and Buildings 140 (2017) 81–97.

[37] Y. Zheng, X. Yi, M. Li, R. Li, Z. Shan, E. Chang, T. Li, Forecasting fine-grained air quality based on big data, in: International Conference on Knowledge Discovery and Data Mining, 2015, pp. 2267–2276.

[38] S. T. A. Niaki, S. Hoseinzade, Forecasting s&p 500 index using artificial neural networks and design of experiments, Journal of Industrial Engineering International 9 (1) (2013) 1.

[39] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.

[40] T. Hong, J. Xie, J. Black, Global energy forecasting competition 2017:

Table 5: Experimental results on air quality dataset. DeepPIPE achieved best on MAE and MPIW.

Model	MAE	PICP	MPIW
1-step prediction			
ARIMA	5.35±0.00	1.00±0.00	26.30±0.00
VAR	5.40±0.00	1.00±0.00	27.10±0.00
SVR	5.50±0.00	-	-
GBDT	5.49±0.00	-	25.89±0.00
DeepMVE	5.47±0.29	0.93±0.00	23.69±0.79
DeepHQ	-	0.93±0.00	23.71±0.61
ShallowPIPE	5.65±0.34	0.93±0.00	23.62±0.66
DeepPIPE	**5.25±0.09**	0.93±0.01	**22.79±0.47**
3-step prediction			
ARIMA	7.85±0.00	1.00±0.00	37.46±0.00
VAR	7.94±0.00	1.00±0.00	36.01±0.00
SVR	8.00±0.00	-	-
GBDT	8.03±0.00	-	34.52±0.00
DeepMVE	7.80±0.24	0.93±0.01	32.31±0.82
DeepHQ	-	0.91±0.00	31.02±0.57
ShallowPIPE	8.01±0.27	0.91±0.00	30.42±0.58
DeepPIPE	**7.69±0.03**	0.91±0.00	**29.95±0.12**

Hierarchical probabilistic load forecasting, International Journal of Forecasting 35 (4) (2019) 1389–1399.

[41] M. Perc, Nonlinear time series analysis of the human electrocardiogram, European Journal of Physics 26 (5) (2005) 757.