Deep Uncertainty Quantification: A Machine Learning Approach for Weather Forecasting

Bin Wang
Centre for Artificial Intelligence,
University of Technology Sydney
Southwest Jiaotong University
bin.wang-7@student.uts.edu.au

Jie Lu*
Centre for Artificial Intelligence,
University of Technology Sydney
jie.lu@uts.edu.au

Zheng Yan
Centre for Artificial Intelligence,
University of Technology Sydney
yan.zheng@uts.edu.au

Huaishao Luo
Southwest Jiaotong University
infocom525@gmail.com

Tianrui Li*
Institute of Artificial Intelligence,
Southwest Jiaotong University
Chengdu 611756, China
trli@swjtu.edu.cn

Yu Zheng†
JD Intelligent Cities Research
JD Intelligent Cities Business Unit
Beijing, China
msyuzheng@outlook.com

Guangquan Zhang
Centre for Artificial Intelligence,
University of Technology Sydney
guangquan.zhang@uts.edu.au

ABSTRACT

Weather forecasting is usually solved through numerical weather prediction (NWP), which can sometimes lead to unsatisfactory performance due to inappropriate setting of the initial states. In this paper, we design a data-driven method augmented by an effective information fusion mechanism to learn from historical data that incorporates prior knowledge from NWP. We cast the weather forecasting problem as an end-to-end deep learning problem and solve it by proposing a novel negative log-likelihood error (NLE) loss function. A notable advantage of our proposed method is that it simultaneously implements single-value forecasting and uncertainty quantification, which we refer to as *deep uncertainty quantification* (DUQ). Efficient deep ensemble strategies are also explored to further improve performance. This new approach was evaluated on a public dataset collected from weather stations in Beijing, China. Experimental results demonstrate that the proposed NLE loss significantly improves generalization compared to mean squared error (MSE) loss and mean absolute error (MAE) loss. Compared with NWP, this approach significantly improves accuracy by 47.76%, which is a state-of-the-art result on this benchmark dataset. The preliminary version of the proposed method won 2nd place in an online competition for daily weather forecasting [1].

CCS CONCEPTS

· **Applied computing → Environmental sciences**.

KEYWORDS

Urban computing; weather forecasting; deep learning; uncertainty quantification

ACM Reference Format:
Bin Wang, Jie Lu, Zheng Yan, Huaishao Luo, Tianrui Li, Yu Zheng, and Guangquan Zhang. 2019. Deep Uncertainty Quantification: A Machine Learning Approach for Weather Forecasting. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330704

1 INTRODUCTION

Meteorological elements, such as temperature, wind and humidity, profoundly affect many aspects of human livelihood [3, 11]. They provide analytical support for issues related to urban computing such as traffic flow prediction, air quality analysis, electric power generation planning and so on [26]. The most common method currently utilized in meteorology is the use of physical models to simulate and predict meteorological dynamics known as numerical weather prediction, or NWP. The advantage of NWP is that it is based on the numerical solution of atmospheric hydro thermo dynamic equations and is able to obtain high prediction accuracy if the initial solution is appropriately chosen. However, NWP may not be reliable due to the instability of these differential equations [20]. With the growing availability of meteorological big data, researchers have realized that introducing data-driven approaches

* Jie Lu and Tianrui Li are the correspondence authors of this paper.
† Yu Zheng is also affiliated with Xidian University.

[1] *AI Challenger 2018* https://challenger.ai/competition/wf2018. For English speakers, kindly click on the top right corner on the website.

into meteorology can achieve considerable success. Several machine learning methods have been applied to weather forecasting [4, 5, 17]. The merit of data-driven methods is that they can quickly model patterns through learning to avoid solving complex differential equations. Nevertheless, learning from historical observations alone requires big data and a tedious amount of feature engineering to achieve satisfying performance, which presented us with the following challenge. Could we combine the advantages of NWP and machine learning to make a more efficient and effective solution? At the same time, single-value (i.e. point estimation) forecasting lacks credibility and flexibility for numerous types of human decision. Could we provide more information to indicate the prediction interval based on high-quality uncertainty quantification? This paper aims to introduce a unified deep learning method to address these problems through end-to-end learning. In particular, we will predict multiple meteorological variables across different weather stations at multiple future steps. The proposed approach has several advantages: efficient data pre-processing, end-to-end learning, high accuracy, uncertainty quantification and easy-to-deploy which makes it have considerable practical significance. The contributions of this work are summarized as follows:

(1) It proposes an effective deep model and information fusion mechanism to handle weather forecasting problems. To the best of our knowledge, this is the first machine learning method which combines historical observations and NWP for weather forecasting. Data and source codes will be released and can be used as a benchmark for researchers to study machine learning in the meteorology field[2].
(2) It establishes effective assumptions and constructs a novel negative log-likelihood error (NLE) loss function. Unlike Bayesian deep learning (BDL), deep uncertainty quantification (DUQ) can be seamlessly integrated with current deep learning frameworks such as Tensorflow and Pytorch. It can be directly optimized via backpropagation (BP). Our experiments show that compared with typical mean squared error (MSE) and mean absolute error (MAE) loss, training by NLE loss significantly improves the generalization of point estimation. This phenomenon has never been reported in previous researches.
(3) Besides precise point estimation, DUQ simultaneously inferences the sequential prediction interval. This attractive feature has not been studied well in previous deep learning research for time series forecasting. It can be applied to various time series regression scenarios.
(4) It explores efficient deep ensemble strategies. The experimental results demonstrate that the ensemble solution significantly improves accuracy.

The rest of the paper is structured as follows. We discuss related works in Section II and introduce our method in Section III. In Section IV, we discuss experiments and performance analysis . Last, we conclude with a brief summary and shed light on valuable future works in Section VI.

[2] Released codes: https://github.com/BruceBinBoxing/Deep_Learning_Weather_Forecasting

2 RELATED WORKS

Weather Forecasting Weather forecasting has been well studied for more than a century. Most contemporary weather forecasting relies on the use of NWP approaches to simulate weather systems using numerical methods [9, 14, 20]. Some researchers have addressed weather forecasting as a purely data-driven task using ARIMA [1], SVM [16], forward neural network [21], etc. These shallow models explore only a few variables, which may not capture the spatio-temporal dynamics of diverse meteorological variables. Deep learning has also shown promise in the field of weather prediction. The study in [5] first adopted an auto-encoder to reduce and capture non-linear relationships between variables, and then trained a multi-layer perceptron for prediction. In [4], a deep hybrid model was proposed to jointly predict the statistics of a set of weather-related variables. The study in [18] formulated precipitation nowcasting as a spatio-temporal sequence forecasting problem and proposed convolutional LSTM to handle it. However, these purely data-driven models are limited in that: 1) they all ignore important prior knowledge contained in NWP, which may not capture the spatio-temporal dynamics of diverse meteorological variables; 2) some need tedious feature engineering, such as extracting seasonal features as inputs and kernel selection, which seems contrary to the end-to-end philosophy of deep learning; 3) all lack the flexibility of uncertainty quantification.

Deep Learning Although deep learning for regression has achieved great success and benefits from the powerful capability of learning representation, solutions like [23, 25, 27] only focus on point estimation and there is a substantial gap between deep learning and uncertainty quantification.

Uncertainty Quantification For ease of explaining uncertainty in regression scenario, let us only consider the equation: $\hat{Y} = f(X) + \epsilon$, where statistically $f(X)$ is the mean estimation (predictable point estimation) of the learned machine learning model and is also called the epistemic part. Its uncertainty comes from *model variance* denoted by σ_m^2; ϵ is the irreducible noise, also named the aleatoric part. The reason it exists is because there are unobtained explanatory variables or unavoidable random factors, so it is called *data variance*. Due to the difficulty of expressing ϵ with a deterministic equation, data variance is usually modeled by a Gaussian distribution with zero mean and a variance σ_d^2 (Central Limit Theorems). If σ^2 does not change, it is a homoskedastic problem, otherwise it is regarded as heteroskedastic. Then the total variance $\sigma^2 = \sigma_d^2 + \sigma_m^2$. The learning process is usually implemented by maximum likelihood estimation, which will learn the estimated $\hat{\sigma}^2$.

Uncertainty quantification can provide more reference information for decision-making and has received increased attention from researchers in recent years [6] . However, most uncertainty quantification methods are based on shallow models and do not take advantages of deep learning. Deep models can automatically extract desirable representations, which is very promising for high-quality uncertainty quantification. To this end, Bayesian deep learning (BDL), which learns a distribution over weights, is currently the most popular technique [22]. Nevertheless, BDL has a prominent drawback in that it requires significant modification, adopting variational inference (VI) instead of back-propagation (BP), to train

deep models. Consequently, BDL is often more difficult to implement and computationally slower. An alternative solution is to incorporate uncertainty directly into the loss function and directly optimize neural networks by BP [7, 12, 13]. This still suffers from certain limitations. 1) Regression is solved as a *mapping* problem rather than *curve fitting*, hence this method cannot naturally be applied to multi-step time series forecasting [13, 15]. 2) The output only consider a single dimension. If it is to be extended to multiple dimensions or for multi-step time series forecasting, the method must be based on effective and reasonable assumptions. 3) Only a shallow forward neural network is used for illustration and the superior performance of deep learning is not explored.

The proposed DUQ addresses these limitations by combining deep learning and uncertainty quantification to forecast multi-step meteorological time series. It can quantify uncertainty, fuse multi-source information, implement multi-out prediction, and can take advantage of deep models. Meantime, it is optimized directly by BP.

3 OUR METHOD

3.1 Problem Statement

Let us say we have historical meteorological observations from a chosen number of weather stations and a preliminary weather forecast from NWP. For each weather station, we concern weather forecasting to approximate ground truth in the future. We define this formally below:

3.1.1 Notations. For a weather station s, we are given:

(1) Historical observed meteorological time series $\mathbf{E}(t) = [e_1(t), e_2(t), \ldots, e_{N_1}(t)] \in \mathrm{R}^{N_1}$, where the variable e_i is one type of meteorological element, for $t = 1, \ldots, T_E$.
(2) Another feature series consist of forecasting timesteps, station ID and NWP forecasting, i.e., $\mathbf{D}(t) = [d_1(t), d_2(t), \ldots, d_{N_2}(t)] \in \mathrm{R}^{N_2}$, where the variable $d_i(t)$ is one of N_2 features, for $t = T_E + 1, \ldots, T_E + T_D$, and T_D is the required number of forecasting steps.
(3) Ground truth of target meteorological variables denoted as $\mathbf{Y}(t) = [y_1(t), y_2(t), \ldots, y_{N_3}(t)] \in \mathrm{R}^{N_3}$, where the variable $y(t)$ is one of N_3 target variables, for $t = T + 1, T_E + 2, \ldots, T_E + T_D$ and its estimation denoted as $\hat{\mathbf{Y}}(t)$.
(4) Then we define:
$\mathbf{E}_{T_E} = [\mathbf{E}(1), \mathbf{E}(2), \ldots, \mathbf{E}(T_E)] \in \mathrm{R}^{T_E \times N_1}$
$\mathbf{D}_{T_D} = [\mathbf{D}(T_E + 1), \mathbf{D}(T_E + 2), \ldots, \mathbf{D}(T_E + T_D)] \in \mathrm{R}^{T_D \times N_2}$
$\mathbf{X}_{T_D} = [\mathbf{E}_{T_E}; \mathbf{D}_{T_D}]$
$\mathbf{Y}_{T_D} = [\mathbf{Y}(T_E + 1), \mathbf{Y}(T_E + 2), \ldots, \mathbf{Y}(T_E + T_D)] \in {}^{T}\mathrm{R}^{N_D \times 3}$.

3.1.2 Task Definition. Given $\mathbf{X}_{T_D}$, the point estimation will predict $\hat{\mathbf{Y}}_{T_D}$ to approximate $\mathbf{Y}_{T_D}$ as far as possible. The prediction interval $[\hat{\mathbf{Y}}^L_{T_D}, \hat{\mathbf{Y}}^U_{T_D}]$ will ensure $\mathbf{Y}_{TD} \in [\hat{\mathbf{Y}}^L_{T_D}, \hat{\mathbf{Y}}^U_{T_D}]$ (element-wise) with the predefined tolerance probability. The prediction interval will cover the ground truth with at least the expected tolerance probability.

This research was driven by a real-world weather forecasting competition. For feasible comparison, it focuses on a set time period, i.e., from 3:00 intraday to 15:00 (UTC) of the next day, hence $T_D = 37$. The target variables include temperature at 2 meters (t2m), relative humidity at 2 meters (rh2m) and wind at 10 meters (w10m), hence $N_3 = 3$. The proposed method can be easily extended for any time interval prediction and more target variables.

3.2 Information Fusion Methodology

Data exploration analysis provides insights for the motivation and methodology of information fusion. Fig. 1 shows the variation of three target meteorological variables over the past three years. It can seen that only temperature reflects a strong seasonal variation, while relative humidity and wind speed are subjected to much noise. Based on this observation, methods that extract seasonal features from historical data may not provide the best results, since weather changes too dramatically [1, 4]. Frequent concept drift cause long-term historical meteorological data lack value [8]. One conclusion summarizes that *"For many time series tasks only a few recent time steps are required"*[2]. On the other hand, NWP is a relatively reliable forecasting method, but inappropriate initial states can introduce undesirable error bias. To address this, we propose a balanced fusion methodology:

- First, *only recent* observations, i.e., $\mathbf{E}_{T_E}$ should be adopted for *modeling recent meteorological dynamics*.
- Second, a wise NWP fusion strategy should incorporate NWP forecasting *at a counterpart forecasting timestep* to easily correcting bias in NWP. Conversely, an unwise fusion strategy that is not carefully designed may absorb NWP which is not conducive to capturing important NWP signals. Hence we incorporate NWP forcasting into $\mathbf{D}_{T_D}$ rather than into $\mathbf{E}_{T_E}$ or hidden coding (see Fig. 3).

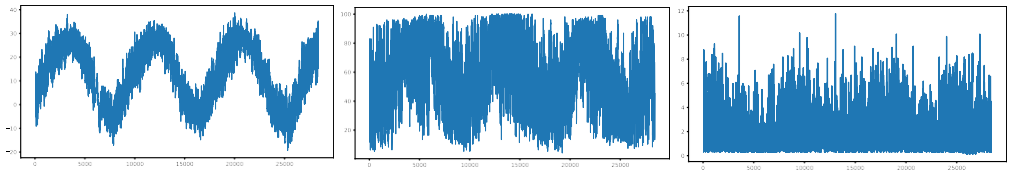
Fig. 2 aggregates historical statistics of mean (solid line) and 90% confidence interval (shade area) for 10 stations from 3:00 intraday to 15:00 (UTC). We find that: 1) There exists obvious difference of mean and variance statistics, e.g., the mean value of station-ID 7 follows a different trend compared with other stations. 2) Every hour at every station has different meteorological characteristics of mean and variance. To address this, we will introduce station ID and time ID into $\mathbf{D}_{T_D}$.

3.3 Data Preprocessing

3.3.1 Missing values. There are two kind of missing values, i.e. *block missing* (one-day data lost) and *local missing* (local non-continuous time series), which vary in severity. For block missing [24], we just delete the data of those days from the dataset. For local missing data, we use linear interpolation to impute missing values. Taking the training set as an example, we delete 40 days with block missing values from a total of 1188 days, leaving the training data from 1148 (1188-40) days.

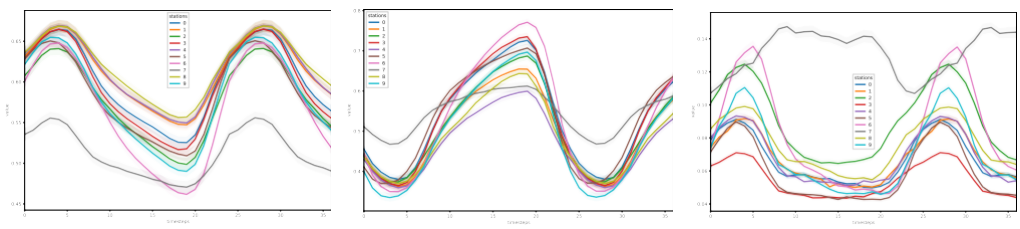
3.3.2 Normalization of Continuous Variables. Continuous variables without normalization sometimes result in training failure for deep leaning, so we use min-max normalization to normalize each continuous feature into [0, 1]. In the evaluation, we re-normalize the predicted values back to the normal scale.

3.3.3 Category Variables. There are two category variables, i.e. *Timesteps ID* and *Station ID*. Rather than hard-coding, such as one-hot or sin-cosine coding, we code them by embedding, which has achieved better performance than hard-coding [10].



(a) t2m (b) rh2m (c) w10m

Figure 1: Three historical target variable series from 03/01/2015-05/31/2018. They show a strong seasonal variation of t2m, a weak seasonal variation of 2-meter rh2m , and almost no seasonal variation of w10m.



(a) t2m (b) rh2m (c) w10m

Figure 2: The variation of mean (solid line) and 90% confidence interval (shaded area) for 10 stations during the target forecasting time zone (3:00 intraday to 15:00 of the next day) from 03/01/2015-05/31/2018. Values of Y-axis are following normalization.

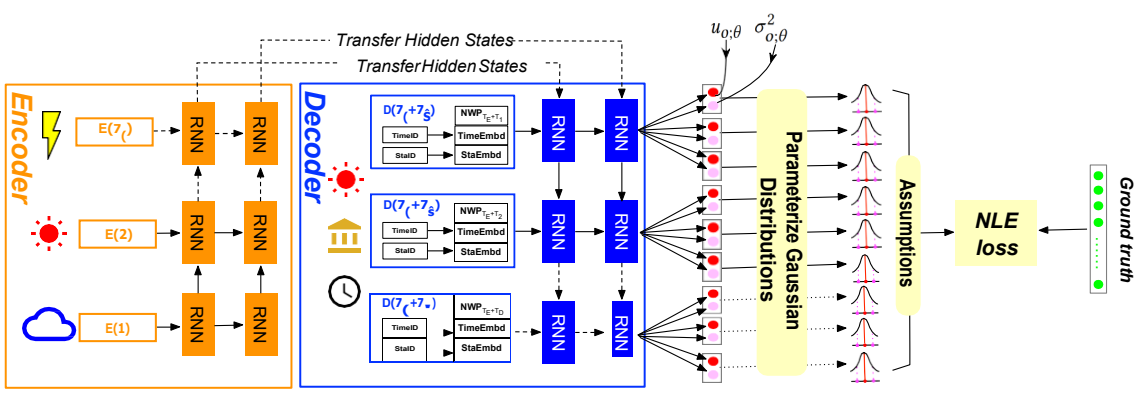


Figure 3: DUQ for sequential point estimation and prediction interval.

3.3.4 Input/Output Tensors. Lastly, we load data from all stations and dates and reshape it to three tensors as follows:

- (I, T_E, S, N_1), (I, T_D, S, N_2) (i.e., input tensors).
- (I, T_D, S, N_3) (i.e., ground truth tensor).

Note that I is the date index and S is the station index. When drawing training samples, we first draw integer date $i \in I$ and station $s \in S$. We can then index by i, s from these three tensors and obtain one training instance $\mathbf{X}^{i,s} = [\mathbf{E}^{i,s}_{T_E}; \mathbf{D}^{i,s}_{T_D}]$ and $\mathbf{Y}^{i,s}_{T_D}$ abbreviated as $\mathbf{X}_{T_D} = [\mathbf{E}_{T_E}; \mathbf{D}_{T_D}]$ and $\mathbf{Y}_{T_D}$ for brevity. The advantage of organizing data in this four-tuple style is that we can conveniently index the data via the specific dimension for hindsight inspection and consideration of scalability. For example, we can index specific dates and stations for later transfer learning research. Readers can refer to the instantiated example *Parameter Settings for Reproducibility* in Section V for deeper understanding.

3.4 Model Architecture

The proposed DUQ is based on sequence-to-sequence (seq2seq, also a.k.a Encoder-Decoder). Its detailed formula is not discussed here. Readers can refer to [19] for more detail. There are already many high-performance variants for different tasks, but most of them focus on making improvements from the structural perspective to make point estimation more precise. We first incorporate sequential uncertainty quantification for weather forecasting into the architecture presented in Fig. 3. The encoder first extracts latent representations $\mathbf{c}$ from the observed feature series $\mathbf{E}_{T_E}$:

$$\mathbf{c} = Enc(\mathbf{E}_{T_E}; \theta_1)$$

where $\mathbf{c}$ captures the current meteorological dynamics and is then transferred to form the initial state of the decoder. Based on the memory of $\mathbf{c}$, the decoder absorbs $\mathbf{D}_{T_D}$ including station identity

(StaID), forecasting time identity (TimeID), and NWP forecasting. Two embedding layers will be introduced for StaID and TimeID respectively to automatically learn the embedding representations. This architecture will generate sequential point estimation $\mathbf{u}_{T_D}$ used as $\hat{\mathbf{Y}}_{T_D}$ to predict $\mathbf{Y}_{T_D}$ as well as the variance $\hat{\sigma}^2_{T_D}$ utilized to estimate $[\hat{\mathbf{Y}}^L_{T_D}, \hat{\mathbf{Y}}^U_{T_D}]$:

$$\hat{\sigma}^2_{T_D}, \hat{\mathbf{Y}}_{T_D} = Dec(\mathbf{c}, \mathbf{D}_{T_D}; \theta_2)$$

where θ_1 and θ_2 are learnable parameters. We use $f(\cdot)$ to represent the combination of Encoder-Decoder and use $\mathbf{X}_{T_D} = [\mathbf{E}_{T_D}; \mathbf{D}_{T_D}]$, which can then be regarded as:

$$\hat{\sigma}^2_{T_D}, \hat{\mathbf{Y}}_{T_D} = f(\mathbf{X}_{T_D})$$

3.5 Learning Phase

DUQ predicts two values at each timestep corresponding to the predicted mean and variance to parameterize the Gaussian distributions [3]. The NLE is calculated for the Gaussian distributions, which must be based on reasonable hypotheses. Three mild but experimentally effective assumptions are proposed (degree of effectiveness can be seen in the experimental results in Table 5):

(1) Each day and each station are independent. This assumption ensures it is reasonable that the number of all training samples can be regard as $I \times S$. Based on this, we can minimize the negative log-likelihood error loss:

$$NLE = -\sum_{i=1}^{I}\sum_{s=1}^{S} p_\theta(\mathbf{Y}_{T_D}|\mathbf{X}_{T_D})$$

(2) Each target variable and each timestep at one specific station are *conditionally independent* given $\mathbf{X}_{T_D}$. Based on this, we can further decompose $p_\theta(\mathbf{Y}_{T_D}|\mathbf{X}_{T_D})$ by the product rule and transform it via log operation as:

$$p_\theta(\mathbf{Y}_{T_D}|\mathbf{X}_{T_D}) = \prod_{o}^{N_3}\prod_{t}^{T_E+T_D} p_\theta(y_o(t)|\mathbf{X}_t)$$

$$log\, p_\theta(\mathbf{Y}_{T_D}|\mathbf{X}_{T_D}) = \sum_{o}^{N_3}\sum_{t}^{T_E+T_D} log\, p_\theta(y_o(t)|\mathbf{X}_t)$$

(3) The target variables satisfy multivariate independent Gaussian distribution and σ is a function of the input features, i.e., $\mathbf{Y}_{T_D} \sim N(u_\theta(\mathbf{X}_{T_D}), \sigma_\theta(\mathbf{X}_{T_D}))$. Based on this assumption, the final loss is:

$$NLE = -\sum_{i}^{I}\sum_{s}^{S}\sum_{o}^{N_3}\sum_{t}^{T_E+T_D} log\, p_\theta(y_o(t)|\mathbf{X}^{i,s}_t)$$

$$= \sum_{i}^{I}\sum_{s}^{S}\sum_{o}^{N_3}\sum_{t}^{T_E+T_D} \frac{log\sigma^2_{o;\theta}(\mathbf{X}^{i,s}_t)}{2} + \frac{(y_o(t) - u_{o;\theta}(\mathbf{X}^{i,s}_t))^2}{2\sigma^2_{o;\theta}(\mathbf{X}^{i,s}_t)} + C$$

[3] We enforce the positivity constraint on the variance by passing the second output through the $softplus$ function $log(1 + exp(\cdot))$ and add a minimum variance (e.g. 10^{-6}) for numerical stability.

where C is a constant which can be omitted during training. $y_o(t)$ is the ground truth of a target variable o at timestep t of the counterpart station s on the day i, $\sigma^2_{o;\theta}(\mathbf{X}^{i,s}_t)$ and $u_{o;\theta}(\mathbf{X}^{i,s}_t)$ are respectively the variance and mean of a Gaussian distribution parameterized by DUQ. The aim of the entire learning phase is to minimize NLE. Optimizing by deep models can easily lead to overfitting on the training set, therefore it is necessary to implement early-stopping on the validation set. Algorithm 1 outlines the procedure for the learning phase.

Algorithm 1: Algorithm for learning

Input : Input tensors: (I, T_E, S, N_1), (I, T_D, S, N_2);
Output tensor: (I, T, S, N_3);
Maximum iterations;
Tolerance iterations for early-stopping;
Output : Learned DUQ model

```
// Learning phase
Initialize all learnable parameters θ in DUQ
repeat
    B ← ∅
    while each training datum n (1 ≤ n ≤ BatchSize) do
        // Format training data samples
        Draw a random integer i ~ uniform(0, I)
        Draw a random integer s ~ uniform(0, S)
        Index by i, s from input and output tensors and get one
          training sample (X_{T_D}, Y_{T_D})
        Put this sample into B
    end
    Update θ via BP by minimizing the NLE loss on B
until stopping criteria are met;
```

3.6 Inference Phase

After training, we can implement statistical inference for an input $\mathbf{X}_{T_D}$ by:

$$u_\theta(\mathbf{X}_{T_D}), \sigma^2_\theta(\mathbf{X}_{T_D}) = f(\mathbf{X}_{T_D})$$

where $u_\theta(\mathbf{X}_{T_D})$ is statistically the mean estimation i.e., $\hat{\mathbf{Y}}_{T_D}$ given $\mathbf{X}_{T_D}$, which will be adopted for forecasting and $\sigma^2_\theta(\mathbf{X}_{T_D})$ is statistically the variance estimation, i.e., $\hat{\sigma}^2_{T_D}$ given $\mathbf{X}_{T_D}$. Recall our assumption that $\hat{\mathbf{Y}}_{T_D}$ satisfies Gaussian distribution, so upper bound $\hat{\mathbf{Y}}^U_{T_D}$ and lower bound $\hat{\mathbf{Y}}^L_{T_D}$ can be inferenced as $\hat{\mathbf{Y}}^U_{T_D} = \hat{\mathbf{Y}}_{T_D} + \lambda\hat{\sigma}_{T_D}$ and $\hat{\mathbf{Y}}^L_{T_D} = \hat{\mathbf{Y}}_{T_D} - \lambda\hat{\sigma}_{T_D}$, where $\hat{\sigma}_{T_D}$ is the standard deviation and, λ should be determined according to the pre-defined $1 - z$. In this research, $1 - z = 0.9$ thus λ is set to 1.65 according to the z-score of Gaussian distribution. Algorithm 2 gives the inference procedure.

3.7 Ensemble Methodology

We adopt a simple but efficient principle for ensemble: each single model is a DUQ-based model initialized with specified nodes. The ensemble point estimation is the averaged point estimation of all DUQ-based models, which is scalable and easily implementable.

3.8 Evaluation Metrics

3.8.1 Point Estimation Measurement. We first calculate the root mean squared error ($RMSE$) for each objective variable from S=10

Algorithm 2: Algorithm for inference

Input : $\mathbf{X}_{T_D}$, z;

Output: $\hat{\mathbf{Y}}^L_{T_D}$, $\hat{\mathbf{Y}}^U_{T_D}$, $\hat{\mathbf{Y}}_{T_D}$, $\hat{\sigma}_{T_D}$;

// Inference phase

$\hat{\mathbf{Y}}_T(\ , \hat{\sigma}_T) = f_\theta(\mathbf{X}_T)$

// determine λ by z from z-score of Gaussian distribution

$\hat{\mathbf{Y}}^L_{T_D} = \hat{\mathbf{Y}}_{T_D} - \lambda\hat{\sigma}_{T_D}$

$\hat{\mathbf{Y}}^U_{T_D} = \hat{\mathbf{Y}}_{T_D} + \lambda\hat{\sigma}_{T_D}$

de-normalize $\hat{\mathbf{Y}}^L_{T_D}, \hat{\mathbf{Y}}^U_{T_D}, \mathbf{Y}_{T_D}$ for real-world evaluation.

stations for daily evaluation.

$$RMSE_{obj} = \sqrt{\frac{\sum_{s=1}^{S}\sum_{t=T_E+1}^{T_E+T_D}(y^s_o(t) - \hat{y}^s_o(t))^2}{S \cdot T_D}}$$

where $y^s_o(t)$ and $\hat{y}^s_o(t)$ are respectively the ground truth and the predicted value of the objective variable obj (i.e., $t2m$, $rh2m$ or $w10m$ in this paper) of the station s at time t.

$$RMSE_{day} = \frac{RMSE_{t2m} + RMSE_{rh2m} + RMSE_{w10m}}{N_3}$$

$RMSE_{day}$ is the ultimate RMSE criterion in the experimental reports for each day. $RMSE_{avg}$ is the average $RMSE_{day}$ over all days. To demonstrate the improvement over the classic NWP method, we employ the following evaluation using the associated skill score (SS, the higher the better):

$$SS_{obj} = 1 - \frac{RMSE_{obj_ml}}{RMSE_{obj_nwp}}$$

where $RMSE_{obj_nwp}$ is the $RMSE_{obj}$ calculated by the NWP method and $RMSE_{obj_ml}$ is calculated from the prediction made of machine learning models.

$$SS_{day} = \frac{SS_{t2m} + SS_{rh2m} + SS_{w10m}}{N_3}$$

SS_{day} is the ultimate SS criterion in experimental reports for every day. SS_{avg} is the average SS_{day} over all days, which is also the ultimate rank score in the online competition.

3.8.2 Prediction Interval Measurement. To evaluate the prediction interval, we introduce the metric called prediction interval coverage probability (PICP). First, an indicator tensor $\mathbf{B} \in \mathbb{R}^{S \times T_D \times N_3}$ is defined. Each Boolean variable $b^{s,t,o} \in [0, 1]$ represents whether the objective variable o at the predicted time step t at the station s has been captured by the estimated prediction interval.

$$b^{s,t,o} = \begin{cases} 1, \text{ if } & \hat{y}^L_{s,o}(t) \le y_{s,o}(t) \le \hat{y}^U_{s,o}(t) \\ 0, \text{ else.} & \end{cases}$$

The total number of captured data points for the objective variable is defined as C_{obj},

$$C_{obj} = \sum_{s=1}^{S}\sum_{t=T_E+1}^{T_E+T_D} b^{s,t,o}$$

Then $PICP_{obj}$ for the objective variable is defined as:

$$PICP_{obj} = \frac{C_{obj}}{S \cdot T_D}$$

Ideally, $PICP_{obj}$ should be equal to or greater than the predefined value i.e., $1 - z = 0.9$ where z is the significant level and is set to 0.1 in our experiments.

4 EXPERIMENTS AND PERFORMANCE ANALYSIS

4.1 Baselines

SARIMA Seasonal autoregressive integrated moving average is a benchmark model for univariate time series, where parameters are chosen using AIC (Akaike information criterion).

SVR Support vector regression is a non-linear support vector machine for regression estimation.

GBRT Gradient boosting regression tree is an ensemble method for regression tasks and is widely used in practice.

DUQ$_{50}$ is one layer GRU-based seq2seq with 50 hidden nodes. The loss function is *NLE*.

DUQ$_{50,50}$ is two layers GRU-based seq2seq with 50 hidden nodes of each layer. The loss function is *NLE*.

DUQ$_{200}$ is one layer GRU-based seq2seq with 200 hidden nodes. The loss function is *NLE*.

DUQ$_{300,300}$ is two layers GRU-based seq2seq with 300 hidden nodes of each layer. The loss function is *NLE*.

DUQ$_{noNWP}$ is the same as DUQ$_{300,300}$ except that NWP forecasting (i.e. **NWP** of $\mathbf{D}_{T_D}$, refer to Fig. 3) is masked by zero values.

DUQ$_{noOBS}$ is the same as DUQ$_{300,300}$ except that the observation features (i.e. $\mathbf{E}_{T_E}$) are masked by zero values.

Seq2Seq$_{MSE}$ is the same as DUQ$_{300,300}$ except that the loss function is MSE.

Seq2Seq$_{MAE}$ is the same as DUQ$_{300,300}$ except that the loss function is MAE.

DUQ$_{Esb3}$ ensembles three DUQ models (i.e., DUQ$_{300-300}$, DUQ$_{200-200}$, DUQ$_{100,100}$) for online evaluation. This method achieved 2nd place in the online competition.

DUQ$_{Esb10}$ ensembles 10 DUQ models with different architecture to explore the effectiveness of the ensemble. It ensembles DUQ$_{300,300}$, DUQ$_{310,310}$, ..., DUQ$_{390,390}$ (increasing at 10-neuron intervals).

Model$_{1st}$ achieves the best SS_{avg} during online comparison. According to the on-site report, the author also adopted a complicated stacking and ensemble learning strategy.

4.2 Experimental Environments

The experiments were implemented on a GPU server with Quadro P4000 GPU and Keras programming environment (Tensorflow backend).

4.3 Parameter Settings for Reproducibility

The *batch size* is set to 512. The *embedding dimension* of each embedding layer is set to 2. Since we adopted an early-stopping strategy, it was not necessary to set the *epoch* parameter. Instead, we set the number of *maximum iterations* to a relatively large number of 10000 to take sufficient batch iterations into consideration. The *validation*

interval (vi) is set to 50 meaning that for every 50 iterations, we will test our model on validation set and calculate the validation loss. We set the *early-stopping tolerance* (est) to 10, meaning that if the validation loss over 10 continuous iterations did not decrease, training would be stopped early. We defined the *validation times* (vt) when early-stopping was triggered, hence the *total iterations* (ti) can be calculated by ti=vt×vi. For the prediction interval, z was set to 0.1, $1-z=0.9$ thus λ is set to 1.65 according to the z-score of Gaussian distribution.

We set $N_1=9, N_2=31, N_3=3, T_E=28, T_D=37$ to preprocess the original dataset. After preprocessing, the final dataset shape was as shown below.

For the training set:

- Encoder inputs: (1148, 28, 10, 9)
- Decoder inputs: (1148, 37, 10, 31)
- Decoder outputs: (1148, 37, 10, 3)

For the validation set:

- Encoder inputs: (87, 28, 10, 9)
- Decoder inputs: (87, 37, 10, 31)
- Decoder outputs: (87, 37, 10, 3)

For the test set on each day:

- Encoder inputs: (1, 28, 10, 9)
- Decoder inputs: (1, 37, 10, 31)
- Decoder outputs: (1, 37, 10, 3)

The meaning of each number is explained as follows: we acquired data from 1148 days for the training set and data from 87 days for the validation set. Because our evaluation is based on online daily forecasting, the test day index is 1. Number 28 is a hyperparameter, meaning that the previous 28 hours of observations were used to model recent meteorological dynamics. Number 37 was set according to the specified forecasting steps for the next 37 hours. Number 9 is the dimension of observed meteorological variables. Number 31 (dimension of decoder inputs) consists of concatenating *Timesteps ID* and *Station ID* into 29-dimension of NWP forecasting (2+29=31). Number 3 is the ground truth number for 3 target variables. The size of the final training set is 1148*10=11480. The size of validation set is 87*10=870, which is used for early-stopping. The size of test set on each day is 1*10=10.

4.4 Performance analysis

Table 3 presents the evaluation by *SS* score based on rolling forecasting, with incremental data releasd on a daily basis for nine days to mimic real-world forecasting processes.

4.4.1 Effect of information fusion. Comparing $\text{DUQ}_{300-300}$ with DUQ_{noNWP} validates the effectiveness of fusing NWP forecasting. Comparing $\text{DUQ}_{300-300}$ with DUQ_{noOBS} validates the effectiveness of modeling recent meteorological dynamics. Comparing pure NWP with DUQ_{noOBS} illustrates the performance of NWP alone can be further improved by deep learning method. A more noteworthy observation is that DUQ_{noNWP} without NWP information still performs better than pure NWP, which exhibits the superiority of DUQ for modeling meteorological data. These comparisons are powerful proof that modeling with NWP or OBS alone is not good enough, and that comprehensive information fusion is a better solution, which is a valuable reference for the meteorological industry.

4.4.2 Effect of deep learning. On average, the deep learning-based models (DUQ and Seq2Seq) perform better than the non-deep learning models (SARIMA, SVR, GBRT). Comparing DUQ_{50} and DUQ_{50-50} validates the influence of deeper layers. Comparing DUQ_{50}, DUQ_{200}, and $\text{DUQ}_{300-300}$ validates the effectiveness of nodes under the same number of layers.

4.4.3 Effect of loss function. A notable result is that $\text{DUQ}_{300-300}$ trained by NLE loss performs much better than Seq2Seq_{MSE} (MSE loss) and Seq2Seq_{MAE} (MAE loss). In order to empirically understand the reasons for better generalization when trained by NLE, we calculated the *ti* when early-stopping was triggered, as shown in Table 1. It can be seen that $\text{DUQ}_{300-300}$ requires more iterations to converge. A reasonable interpretation is that NLE loss jointly implements two tasks i.e., mean optimization and variance optimization, which need more iterations to converge. This joint optimization may to some extent play a regularization role and help each other out of the local minimum. It may therefore require more iterations to converge and may have better generalization. We believe that this phenomenon deserves the attention of researchers, and that it should be proved by theory in follow-up work.

Table 1: Iterations when early-stopping triggered

Methods	ti (vt× vi)
$\text{DUQ}_{300-300}$	2900 (58*50)
Seq2Seq_{MSE}	2100 (42*50)
DUQ_{noNWP}	1950 (39*50)
Seq2Seq_{MAE}	1850 (37*50)
DUQ_{noOBS}	1450 (29*50)

4.4.4 Effect of ensemble. The ensemble model DUQ_{Esb3} was used in the online competition [4]. DUQ_{Esb10} achieved the best SS_{avd}, which indicates that ensemble with more DUQ models would provide a better solution.

4.4.5 Significance of T-test. Because real-world forecasting only covers nine days, we implemented a *one-tail paired T-test* with significance level $sid = 0.25$ to ensure that our results were statistically significant. The column *P-value* between DUQ_{Esb10} and others shows that each T-test has been passed which means that our method DUQ_{Esb10} is significantly better than any other baseline under the specified significance level. For the single model, we also implemented a T-test between $\text{DUQ}_{300-300}$ and other baselines including DUQ_{noOBS}, DUQ_{noNWP}, Seq2Seq_{MSE}, Seq2Seq_{MAE} to ensure that $\text{DUQ}_{300-300}$ had significant effectiveness, which is shown in Table. 2.

4.4.6 Evaluation by RMSE. We also evaluated all methods by $RMSE_{avd}$ as shown in Table 4. Since $RMSE_{avd}$ and SS_{avd} do not have a fully linear relationship, the counterpart assessment does not reach the

[4] Readers can refer to https://challenger.ai/competition/wf2018 to check our online scores which are consistent with DUQ_{Esb3} during Day 3-Day 9. During Day 1 and Day 2 of the online competition, DUQ_{Esb3} had not been developed. In this paper we re-evaluate DUQ_{Esb3} offline on Day 1 and Day 2. This is also why DUQ_{Esb3} with SS_{avd}=0.4673 is better than the Model_{1st} (0.4671) but was only awarded 2nd place online.

Table 2: T-test among single models for $DUQ_{300-300}$

	P-value on $RMSE_{av\partial}$	P-value on $SS_{av\partial}$
DUQ_{300_300}	-	-
$DUQ_{noNW\ P}$	0.00	0.00
DUQ_{noOBS}	0.00	0.00
$Seq2Seq_{MSE}$	0.00	0.00
$Seq2Seq_{MAE}$	0.03	0.08

optimum at the same time while DUQ_{Esb10} still achieves the best $RMSE_{av\partial}$. The related P-value also indicates that DUQ_{Esb10} is significantly better than other baselines under $si\partial = 0.25$. Because online evaluation does not release the RMSE ranking, the RMSE of $Model_{1st}$ place is not shown in Table 4.

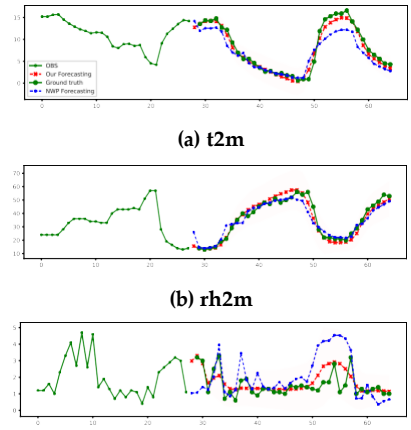
4.4.7 Discuss instability of weather forecasting. Due to meteorological instability and variability, no single model can achieve the best scores every day - not even the ensemble method. Sometimes a single model can achieve the highest SS_{day} score, such as DUQ_{200} on Day 2 and DUQ_{50_50} on Day 7. Overall, however, the ensemble method DUQ_{Esb10} achieves the greatest score benefit from the stability of ensemble learning. The instability of meteorological elements also reflects the need for a prediction interval.

4.4.8 Quantity of prediction interval. An effective prediction interval should satisfy that $PICP_{obj}$ is equal to or greater than the pre-defined $1 - z = 90\%$. Table 5 shows the results. In particular, the $PICP_{rh2m}$ on Day 3 seems far below expectations. The main reason is that forecasting of $rh2m$ is not accurate on that day. The online competition scores of all contestants were particular low on that day. Generally, our approach meets the requirement $PICP_{av\partial} \geq 1 - z = 90\%$.

4.4.9 Quality of prediction interval. We take the model DUQ_{300_300} to visualize the quality of the prediction interval. Fig. 4 illustrates a forecasting instance at one station on a competition day. In each sub-figure, the left green line is the observed meteorological value during the previous 28 hours, the right green line is the ground truth, the blue line is the NWP prediction, the red line is DUQ_{300_300} prediction and the red shaded area is the 90% prediction interval. A noticeable observation is that the prediction interval does not become wider over time, instead, it presents that the width of the middle part is narrower than both ends particularly for t2m and rh2m (deep learning with point estimation alone will not reveal these insights for operators.). A reasonable explanation is that meteorological elements largely change during the daytime and become more stable during night time. Having this prediction interval would provide more information for travel/production planning than only point prediction. Another noteworthy point is that because w10m fluctuates sharply, it is more difficult to forecast point estimate precisely, and the prediction interval tends to be wider than t2m and rh2m.

5 CONCLUSIONS AND FUTURE WORKS

This paper addresses the real-world problem in weather forecasting which has a profound impact on our daily life, by introducing a new deep uncertainty quantification (DUQ) method. A novel loss function called negative log-likelihood error (NLE) was designed to train the prediction model, which is capable of simultaneously inferring sequential point estimation and prediction interval. A noteworthy



(a) t2m

(b) rh2m

(c) w10m

Figure 4: A test sample at one station is chosen to visualize the forecasting of 3 target variables in the future 37 hours. We can see that all predicted points fall into the prediction interval given by $1 - z = 90\%$.

experimental phenomenon reported in this paper is that training by NLE loss significantly improves the generalization of point estimation. This may provide practitioners with new insights to develop and deploy learning algorithms for related problems such as time series regression. Based on the proposed method and an efficient deep ensemble strategy, state-of-the-art performance on a real-world benchmark dataset of weather forecasting was achieved. The overall method was developed in Keras and was flexible enough to be deployed in the production environment. The data and source codes will be released and can be used as a benchmark for researchers and practitioners to investigate weather forecasting. Future works will be directed towards architecture improvement (e.g., attention mechanism), automatic hyperparameter-tuning, and theoretical comparison between NLE and MSE/MAE.

ACKNOWLEDGMENTS

This work was supported by the Australian Research Council (No. DP190101645), and the Natural Science Foundation of China (Nos. 61773324, 61573292). The first author would like to personally thank Junbo Zhang and Xiuwen Yi for helpful discussions.

REFERENCES

[1] Ling Chen and Xu Lai. 2011. Comparison between ARIMA and ANN models used in short-term wind speed forecasting. In *Power and Energy Engineering Conference (APPEEC), 2011 Asia-Pacific*. IEEE, 1–4.
[2] Felix A Gers, Douglas Eck, and Jürgen Schmidhuber. 2002. Applying LSTM to time series predictable through time-window approaches. In *Neural Nets WIRN Vietri-01*. Springer, 193–200.
[3] Tilmann Gneiting and Adrian E Raftery. 2005. Weather forecasting with ensemble methods. *Science* 310, 5746 (2005), 248–249.
[4] Aditya Grover, Ashish Kapoor, and Eric Horvitz. 2015. A deep hybrid model for weather forecasting. In *International Conference on Knowledge Discovery and Data Mining*. ACM, 379–386.
[5] Emilcy Hernández, Victor Sanchez-Anguix, Vicente Julian, Javier Palanca, and Néstor Duque. 2016. Rainfall prediction: A deep learning approach. In *International Conference on Hybrid Artificial Intelligence Systems*. Springer, 151–162.
[6] Abbas Khosravi, Saeid Nahavandi, Doug Creighton, and Amir F Atiya. 2011. Comprehensive review of neural network-based prediction intervals and new

Table 3: The *SS* performance of different methods on 9 days. The column P-value compare the best performing method DUQ_{Esb10} with other methods, using one-tail paired T-test.

Method	SS_{day1}	SS_{day2}	SS_{day3}	SS_{day4}	SS_{day5}	SS_{day6}	SS_{day7}	SS_{day8}	SS_{day9}	$SS_{av\partial}$	P-value
SARIMA	0.1249	-1.4632	-0.2417	-0.4421	-0.2631	-0.2301	0.0630	0.2015	-0.4579	-0.3010	0.00
SVR	-0.7291	-0.6342	-0.1999	-0.5918	-1.1230	-0.8568	-0.6154	-0.5123	-0.5807	-0.6492	0.00
GBRT	0.0221	0.1318	-0.0086	-0.0396	-0.0960	0.0067	0.0772	0.0859	0.0000	0.0199	0.00
DUQ_{50}	0.4813	0.4833	0.2781	0.3053	0.4277	0.4853	0.4609	0.4987	0.2647	0.4095	0.00
DUQ_{50_50}	0.4847	0.4969	0.3088	0.4012	0.4302	0.5051	**0.5656**	0.5502	0.3239	0.4518	0.00
DUQ_{200}	0.5278	**0.5088**	0.2890	0.3797	0.4479	0.5358	0.4961	0.5235	0.3478	0.4507	0.02
DUQ_{300_300}	0.5220	0.5002	0.3352	0.4067	0.4474	0.5289	0.5324	0.5463	0.3047	0.4582	0.00
DUQ_{noNWP}	0.2348	0.2992	0.0081	0.2440	0.1630	0.3125	0.2660	0.3003	-0.1599	0.1853	0.00
DUQ_{noOBS}	0.4694	0.4744	0.2624	0.3447	0.3925	0.4588	0.4756	0.4901	0.3150	0.4092	0.00
$Seq2Seq_{MSE}$	0.4978	0.3934	0.2860	0.3960	0.3965	0.4842	0.4820	0.5138	0.3192	0.4188	0.00
$Seq2Seq_{MAE}$	0.5314	0.4346	0.2671	0.3980	0.4610	0.5391	0.4711	0.5565	0.2999	0.4399	0.00
DUQ_{Esb3}	0.5216	0.4951	0.3358	0.4050	0.4627	0.5359	0.5350	0.5664	**0.3479**	0.4673	0.04
DUQ_{Esb10}	**0.5339**	0.4940	**0.3516**	0.4355	0.4600	0.5575	0.5581	**0.5776**	0.3298	**0.4776**	-
$Model_{1st}$	0.4307	0.4847	0.3088	**0.4572**	**0.5019**	**0.5753**	0.5345	0.5726	0.3384	0.4671	0.24

Table 4: The *RMSE* performance of different methods on 9 days. Since *RMSE* and *SS* are not fully linear relationship, the counterpart assessment does not reach the optimal at the same time.

Method	$RMSE_{day1}$	$RMSE_{day2}$	$RMSE_{day3}$	$RMSE_{day4}$	$RMSE_{day5}$	$RMSE_{day6}$	$RMSE_{day7}$	$RMSE_{day8}$	$RMSE_{day9}$	$RMSE_{av\partial}$	P-value
NWP	7.5923	9.7276	5.1079	5.7335	6.1542	7.5239	6.3647	7.0457	5.8819	6.7924	0.00
SARIMA	7.3017	16.5954	7.2964	8.9968	8.0726	8.1440	7.1722	5.8466	8.1795	8.6228	0.00
SVR	8.1788	10.0436	5.9310	7.1662	7.7576	8.4064	7.9094	8.4749	7.5431	7.9346	0.00
GBRT	6.5551	8.0321	5.6892	6.1422	6.1083	6.5687	5.9449	6.7122	5.9573	6.4122	0.00
DUQ_{50}	3.0432	4.5256	4.1858	4.5445	3.0069	3.4912	3.8087	3.2299	4.5545	3.8211	0.00
DUQ_{50_50}	2.9013	4.2442	4.0203	3.6641	3.2275	3.2062	**2.6428**	2.8175	4.3089	3.4481	0.00
DUQ_{200}	2.8128	**4.0400**	4.1624	4.0732	2.8580	2.8086	3.2870	3.1963	4.3326	3.5079	0.03
DUQ_{300_300}	2.7168	4.0615	3.8866	3.7977	2.8083	2.9211	2.8012	2.9784	**4.3308**	3.3669	0.16
DUQ_{noNWP}	5.0371	5.5370	5.0529	4.6819	4.1385	4.4716	5.7058	5.8346	7.6805	5.3489	0.00
DUQ_{noOBS}	3.2170	4.8604	4.4150	4.1303	3.5896	3.8239	3.4992	3.2031	4.3008	3.8933	0.00
$Seq2Seq_{MSE}$	3.1328	5.0769	4.1400	3.8426	3.2040	3.3142	3.3027	3.1785	4.6426	3.7594	0.00
$Seq2Seq_{MAE}$	2.7272	5.0933	4.2837	4.0184	2.7888	3.0029	3.7165	2.7935	4.6509	3.6750	0.00
DUQ_{Esb3}	2.8000	4.4338	**3.7054**	3.7886	2.8566	2.7890	2.7979	2.8011	4.3310	3.3670	0.05
DUQ_{Esb10}	**2.7027**	4.3341	3.7999	**3.5743**	**2.7627**	**2.6874**	2.7799	**2.7402**	4.3949	**3.3085**	-

Table 5: PICP on every day

$PICP_{obj}$	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	$PICP_{av\partial}$
$PICP_{t2m}$	0.9513	0.9351	0.8918	0.8891	0.9594	0.9648	0.9027	0.8945	0.9351	**0.9249**
$PICP_{rh2m}$	0.9945	0.8702	0.7648	0.8729	0.9621	0.9621	0.9243	0.9243	0.9135	**0.9099**
$PICP_{w10m}$	0.9567	0.9567	0.9081	0.9594	0.9675	0.9621	0.9378	0.9648	0.9540	**0.9519**

advances. *TNN* 22, 9 (2011), 1341–1356.

[7] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Neural Information Processing Systems*. 6402–6413.

[8] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang. 2018. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering* (2018), 1–1. https://doi.org/10.1109/TKDE.2018.2876857

[9] G Marchuk. 2012. *Numerical methods in weather prediction*. Elsevier.

[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Neural Information Processing Systems*. 3111–3119.

[11] Allan H Murphy. 1993. What is a good forecast? An essay on the nature of goodness in weather forecasting. *Weather and Forecasting* 8, 2 (1993), 281–293.

[12] David A Nix and Andreas S Weigend. 1994. Estimating the mean and variance of the target probability distribution. In *International Conference on Neural Networks*, Vol. 1. 55–60.

[13] Tim Pearce, Mohamed Zaki, Alexandra Brintrup, and Andy Neely. 2018. High-Quality prediction intervals for deep learning: A distribution-free, ensembled approach. In *International Conference on Machine Learning*, Vol. 80. 4075–4084.

[14] Lewis Fry Richardson. 2007. *Weather prediction by numerical process*. Cambridge University Press.

[15] Stephen Roberts, Michael Osborne, Mark Ebden, Steven Reece, Neale Gibson, and Suzanne Aigrain. 2013. Gaussian processes for time-series modelling. *Philosophical Transactions of the Royal Society A* 371, 1984 (2013), 20110550.

[16] Nicholas I Sapankevych and Ravi Sankar. 2009. Time series prediction using support vector machines: a survey. *IEEE Computational Intelligence Magazine* 4, 2 (2009).

[17] Navin Sharma, Pranshu Sharma, David Irwin, and Prashant Shenoy. 2011. Predicting solar generation from weather forecasts using machine learning. In *International Conference on Smart Grid Communications*. IEEE, 528–533.

[18] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Neural Information Processing Systems*. 802–810.

[19] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. 2015. Unsupervised learning of video representations using LSTMs. In *International Conference on Machine Learning*. 843–852.

[20] MA Tolstykh and AV Frolov. 2005. Some current problems in numerical weather prediction. *Izvestiya Atmospheric and Oceanic Physics* 41, 3 (2005), 285–295.

[21] Cyril Voyant, Marc Muselli, Christophe Paoli, and Marie-Laure Nivet. 2012. Numerical weather prediction (NWP) and hybrid ARMA/ANN model to predict global radiation. *Energy* 39, 1 (2012), 341–355.

[22] Hao Wang and Dit-Yan Yeung. 2016. Towards Bayesian deep learning: A framework and some existing methods. *IEEE Transactions on Knowledge and Data Engineering* 28, 12 (2016), 3395–3408.

[23] Xiuwen Yi, Junbo Zhang, Zhaoyuan Wang, Tianrui Li, and Yu Zheng. 2018. Deep Distributed Fusion Network for Air Quality Prediction. In *International Conference on Knowledge Discovery and Data Mining*. ACM, 965–973.

[24] Xiuwen Yi, Yu Zheng, Junbo Zhang, and Tianrui Li. 2016. ST-MVL: Filling Missing Values in Geo-sensory Time Series Data. In *International Joint Conference on Artificial Intelligence*.

[25] Junbo Zhang, Yu Zheng, Dekang Qi, Ruiyuan Li, Xiuwen Yi, and Tianrui Li. 2018. Predicting citywide crowd flows using deep spatio-temporal residual networks. *Artificial Intelligence* 259 (2018), 147–166.

[26] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. 2014. Urban computing: Concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 5, 3 (2014), 38.

[27] Zhengyi Zhou and David S Matteson. 2015. Predicting ambulance demand: A spatio-temporal kernel approach. In *International Conference on Knowledge Discovery and Data Mining*. ACM, 2297–2303.