Forecasting the Evolution of Hydropower Generation

Fan Zhou*
University of Electronic Science and Technology of China, Chengdu, China
fan.zhou@uestc.edu.cn

Liang Li
University of Electronic Science and Technology of China, Chengdu, China
liang.li@std.uestc.edu.cn

Kunpeng Zhang
University of Maryland, College park
kpzhang@umd.edu

Goce Trajcevski
Iowa State University, Ames
gocet25@iastate.edu

Fuming Yao
Dadu River Hydropower Development Co., Ltd, China Energy Investment Co.

Ying Huang
Dadu River Hydropower Development Co., Ltd, China Energy Investment Co.

Ting Zhong
University of Electronic Science and Technology of China, Chengdu, China

Jiahao Wang
University of Electronic Science and Technology of China, Chengdu, China

Qiao Liu
University of Electronic Science and Technology of China, Chengdu, China

ABSTRACT

Hydropower is the largest renewable energy source for electricity generation in the world, with numerous benefits in terms of: environment protection (near-zero air pollution and climate impact), cost-effectiveness (long-term use, without significant impacts of market fluctuation), and reliability (quickly respond to surge in demand). However, the effectiveness of hydropower plants is affected by multiple factors such as reservoir capacity, rainfall, temperature and fluctuating electricity demand, and particularly their complicated relationships, which make the prediction/recommendation of station operational output a difficult challenge. In this paper, we present DeepHydro, a novel stochastic method for modeling multivariate time series (e.g., water inflow/outflow and temperature) and forecasting power generation of hydropower stations. DeepHydro captures temporal dependencies in co-evolving time series with a new conditioned latent recurrent neural networks, which not only considers the hidden states of observations but also preserves the uncertainty of latent variables. We introduce a generative network parameterized on a continuous normalizing flow to approximate the complex posterior distribution of multivariate time series data, and further use neural ordinary differential equations to estimate the continuous-time dynamics of the latent variables constituting the observable data. This allows our model to deal with the discrete observations in the context of continuous dynamic systems, while being robust to the noise. We conduct extensive experiments on real-world datasets from a large power generation company consisting of cascade hydropower stations. The experimental results demonstrate that the proposed method can effectively predict the power production and significantly outperform the possible candidate baseline approaches.

*Corresponding author.

CCS CONCEPTS

• **Computing methodologies → Artificial intelligence**; • **Applied computing → Enterprise information systems**; *Industry and manufacturing*; *Decision analysis*; *Forecasting*.

KEYWORDS

Hydropower; ordinary differential equations; multivariate time series; normalizing flow; Bayesian learning

ACM Reference Format:
Fan Zhou, Liang Li, Kunpeng Zhang, Goce Trajcevski, Fuming Yao, Ying Huang, Ting Zhong, Jiahao Wang, and Qiao Liu. 2020. Forecasting the Evolution of Hydropower Generation. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining USB Stick (KDD '20), August 23–27, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3394486.3403337

1 INTRODUCTION

One major consequence of the increase of the world's population is a growing consumption of (and high demand for) energy – in particular, electricity. An important ramification of dealing with the energy crisis in the context of electrical energy is the negative environmental impacts – most notably, the air pollution because of the power plants that burn fossil fuels, such as coal or natural gas. Renewable energy – e.g., sunlight, wind, water, tides – that can be naturally replenished, has become a viable approach towards addressing the energy crisis, and with much smaller adverse impacts. Hydropower, as the largest single source of the renewable power, derives energy from rivers and/or large reservoirs, and has played an increasingly important role in supplying electricity, especially in countries that are deficient in other resources. Compared to unreliable and intermittent power such as wind and solar, hydropower can provide sustainable and controllable energy, capable of quickly responding to surges of demand in the grid. Hydropower plants utilize river/reservoir waters that have a height differential – typically using a dam to provide slopes – to drive turbine generators, converting the kinetic energy to electricity. In 2018, hydroelectricity accounted for about 7% and 16% of total utility-scale electricity generation in U.S. and China[1], respectively.

Notwithstanding the multiple advantages, the output of a hydropower station or a cascade of stations is affected by many factors, such as river/reservoir inflows, temperature, seasonal demand, abrupt demands, gross domestic product (GDP), electricity price, etc. As an example, in general, the hydroelectricity generation in the summer is significantly higher than in the winter due to the abundant precipitation in summer. Complementary to this, the increased water inflow in rainy seasons is not fully leveraged for power generation because of the water level regulation function of dams – i.e., the reservoirs need to abandon surplus water to make sure that water level is within certain safety limits, to prevent disasters such as flood and landslide. Similarly, when the water inflow decreases, the generation capacity of stations are not fully utilized, e.g., only part of water turbines are running in the winter due to lack of liquid water in conjunction with the minimum water level that needs to be maintained to ensure the reservoir ecology. However, a significant factor in these complex interactions is that the power generation companies need to maximize their profits which, in turn, requires coordinating the power generation of their hydropower plants. Put it simply, the amount of generated power multiplied with the electricity price is something that needs to be considered jointly with maintaining water storage for future use and ensuring ecological safety.

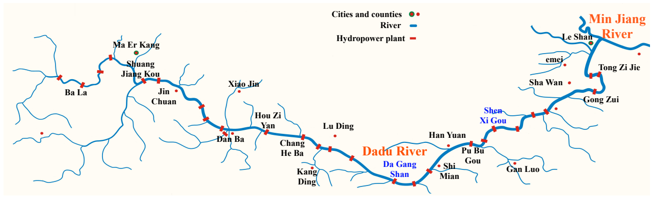


Figure 1: Power stations distribution.

Application Domain & Challenges. Our actual application domain is illustrated in Figure 1 – which shows the map of hydropower stations on the Dadu River, a tributary of the Yangtze River in Sichuan Province. The total length of the mainstream of the Dadu River is 1,062km, along which 28 cascade hydropower stations are distributed with a total installed power generation capacity of about 27.08 gigawatts. The annual power generation of these stations is 115.8 billion kWh, which accounts for about 10% of hydropower in China. The details of the current status of the deployment are discussed in the Appendix.

Modeling the electricity generation of hydropower stations while taking into account a variety of factors has been considered as an important research topic with high societal impact. It can benefit both electricity generation enterprises and smart grid [2, 3, 11, 24, 28]. If power stations can foresee the demand and combine it with their generating capabilities, they would be able to balance the generation among stations and maximize the profit. Concomitantly, state grid could learn to better allocate load requirements and improve the efficiency and security of power delivery. Previous research has mainly focused on macroscopic analysis to model and predict the power consumption demand of smart grid [3, 11, 28]. Much fewer studies have been conducted on fine-grained power plants generation prediction. This, however, can enable higher resource utilization efficiency and, in turn, improve load balance of smart grid – which is at the heart of the motivation for our work. Due to the interplay of various uncertain factors, modeling and forecasting the hydropower generation is a nontrivial task, and it needs to effectively consider several contexts.

Hydropower data can be both high-volume and received in real time, and its representation is often composed of multivariate time-series. In addition to the directly-related contexts such as grid demands, on-grid power, water level along with inflow/outflow – there are certain natural and societal factors that affect the models of hydropower generation. For instance, weather conditions and seasons determine how much water can be used for generating electricity. To provide a stable output, regular and abundant supply of high-quality water is required, often happening in precipitation abundant seasons (e.g., summer). In contrast, water level and flow can be lower due to morainal or ice-scoured soil in winter – as dams are typically built in rural areas with the presence of lakes and/or falls, generally located in rugged mountain regions or glaciated areas. Furthermore, social factors such as electricity price and economic activities have influential impact on the power generation. For example, the on-grid price of different plants is *not* the same, which is determined by the construction cost and the capital cost. In practice, the older the dam, the lower the cost and, consequently, the price – e.g., the sale price of electricity in Pu Bu Gou (built in 2000s) is quite higher than that of Gong Zui Dam (built in 1970s; cf. Figure 1). Also, the sale price is not constant but changes within a certain range, e.g., the price in December is higher than in August – which is a natural outcome of seasonal changes and can be understood by a game between state grid and power generation enterprise, both of which aim to maximize their profit. These economical factors are often mixed up with other factors – e.g., the distance between the dam and industrial/commercial centers and the cost of transmission – all of which are confounding factors for modeling and predicting the hydropower generation.

Our contributions. To address the aforementioned challenges, we present **DeepHydro** – a novel framework modeling the stochastic multivariate time series and the temporal dependencies of latent variables for predicting the power generation of a hydro-plant. Specifically, we design a novel conditioned latent recurrent neural networks model – CL-RNN – to capture the sequential patterns, while maintaining both temporal dependencies and stochasticity of latent variables. We further present a network with *continuous* normalizing flow to approximate the complex distribution of latent variables, allowing efficient variational inference and density estimation. Rather than performing *discrete* transformation as in [22, 29, 34], we borrow the idea of *instantaneous* transformation [4] to avoid computationally intensive determinant computation by solving ordinary differential equations (ODEs). This not only improves the accuracy of posterior approximation with more multi-modal and flexible distribution, but also significantly reduces the number of parameters required. To generalize the probabilistic latent variables to continuous-time dynamics that is natural in power generation, we introduce an ODE extrapolation decoder

[1]https://www.hydropower.org/statusreport

parameterized on another network, which defines a continuous generative process of state transition and bridges the time gaps between discrete observations. The latter enables our method to handle the scenarios when asymmetric measurements happen, e.g., due to transmission delay or sensor failure in a hydro-plant. By incorporating external factors with a fusion network, DeepHydro accounts for both co-evolving patterns of time series (e.g., temperature and water inflow/outflow) and the implicit knowledge (e.g., electricity price) for forecasting the power generation. In sum, our main contributions are as follows:

- We present a novel deep Bayesian learning based model, DeepHydro, to predict the power generation of large-scale hydro-plants. To enable modeling and forecasting, DeepHydro explicitly handles the temporal dependence among stochastic variables, as well as hidden states – to learn accurate and robust representations of multivariate time series corresponding to industrial hydropower generation data.
- We propose to learn the evolution of latent representation with continuous normalizing flow, which preserves the continuous-time characteristics of time series data with unbiased stochastic estimate of the density. By solving the ODE initial-value problem for variable extrapolation, we can optimize our model using variational inference, while better reflecting the continuous dynamical system of hydropower generation.
- We conduct experiments on real-world datasets from a large hydropower generation company demonstrating that DeepHydro improves the power generation prediction performance compared to the state-of-the-art time series learning approaches. To foster enthusiasm on industrial power data mining and efficient renewable energy use, and enable reproducibility, our code and datasets of experiments are publicly available[2].

2 RELATED WORK

Recent advances in deep neural networks, especially recurrent neural networks (RNNs) and their variants [5, 10], have facilitate many deep learning models on classifying and forecasting time series [8, 13, 18, 21, 25, 31]. RNNs such as Long Short-Term Memory (LSTM) [10] and Gated Recurrent Unit (GRU) [5] are dominant building blocks for modeling time series data, owing to the ability of RNNs on modeling non-linear temporal patterns. However, most of these models have focused on different applications, such as traffic prediction [18], sequential recommendation [17] and anomaly detection [13], by capturing long-short term dependency among time series.

There exist efforts that focused on using temporal convolutions for time-series forecasting [1, 25], which are easier to train and more efficient. However, these models are naturally deterministic and unable to effectively model the uncertainty of latent variables. To address this issue, stochastic methods have been incorporated in recent research [9, 29, 32, 34]. All these works capture stochastic latent states using Variational AutoEncoder (VAE), while maintaining uncertainty in the underlying space and allowing posterior inference and sampling conditioned on observations. Nevertheless, they either suffer from biased inference inherent in vanilla VAE [9, 32], or are confronted with high computational cost [29, 34].

As a special kind of (multivariate) time series, electricity generation/demand mining has received significant attention due to its practical value in industry and social consumption. Most of the existing works [3, 11, 19, 24, 28] focus on smart grid load prediction, however, few efforts exist that have focused on hydropower generation. The most related work to this paper is [2], which uses dynamic Bayesian network, combined with a multi-time-scale coupling operation, to model the correlations among time series and predict the hydropower generation. However, their model is limited to small-scale dataset due to the computational bottleneck of probabilistic graphical models. In addition, the proposed methods are evaluated on a smaller hydro-plant (Tankeng Dam) with an installed capacity of 600 megawatts (MW). By contrast, Dagangshan Dam, one of the hydro-plants in our cascade stations, has a total installed capacity of 2,600 MW.

What separates our model is that it is built upon two key techniques: normalizing flow [22] and neural ODEs [4]. Normalizing flows (NF) refer to a family of generative models (see [20] for a comprehensive review) for probabilistic modeling and posterior Inference. Instead of directly applying NF, we introduce a continuous NF to fit the dynamic systems of hydropower. This is inspired by recent works [4, 23] which treat neural networks as dynamic models with a continuum of hidden states, and reformulate the forward pass of neural networks as the solution of the initial value problem in ODE. In addition, to model the time series data as in [4, 23], we also equip our model with the ability to learn the influence of factors, the temporal dependence of latent variables, as well as the decoder of the hidden states of RNN, by solving different ODEs. In this vein, our model initiates the attempt to mine temporal knowledge and the data fusion in an ODE solving manner. In addition to providing a new perspective of forecasting time series, our model is more robust on data representation and can dynamically balance the computational overhead and efficiency, based on the different components.

3 PRELIMINARIES

We now discuss the settings and define the problem more formally, followed by a discussion of the basic background.

3.1 Contexts and Problem Definition

The electricity generated by hydropower plants is not fully on-grid – i.e., typically there are two outlets of water-turbine generator: on-grid power and the auxiliary power for plant self-use. In practice, on-grid power is allocated by the state grid and is more irregular, while self-usage power is relatively stable. A strong indicator of the power generation is the reservoir water level changes. As shown in Figure 2, the water inflow volume roughly equals the sum of the generation flow and the discarded (outflow) water. Also, the hydro-plants are not independent but are cascaded and need to be coordinated together – recall that there are 28 plants in the Dadu River (cf. Figure 1). Moreover, the inflow of a particular reservoir is determined by both the precipitation and the upper stream of previous plants, as well as other small creeks. As mentioned in Section 1, there are many other time-series data, such as temperature

[2]https://github.com/Anewnoob/DeepHydro

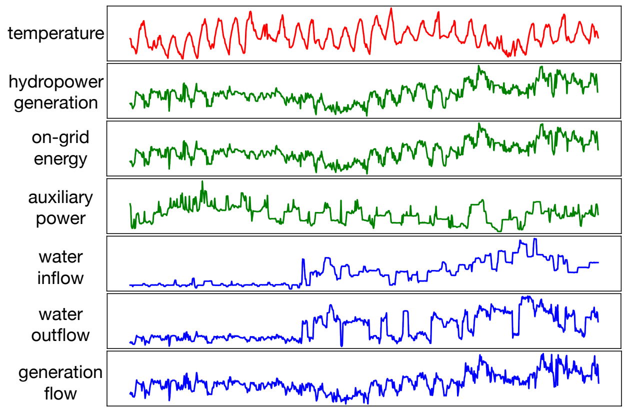


Figure 2: An example of 7-metric multivariate time series in one-month (April 8, 2018 – May 8, 2018) from Dagangshan Dam (not normalized, for better visualization)

(affecting the residential electricity consumption), gross industrial output value, etc. that can affect the power demand – all of which should be properly considered in modeling hydropower generation.

Our core data consists of a sequence of multivariate hydropower observations $\mathbf{X}$. Each observation $\mathbf{x}_t$ at time t consists of electricity power $\mathbf{v}_t$ and water flow $\mathbf{w}_t$. As illustrated in Figure 2, the electricity, defined as $\mathbf{v}_t = \{v_t^1, v_t^2, v_t^3\}$, contains total power generation, on-grid power and auxiliary power. The water flow, defined as $\mathbf{w}_t = \{w_t^1, w_t^2, w_t^3\}$, is composed of water inflow, outflow and generation flow. Thus, each observation $\mathbf{x}_t \in \mathbb{R}^P$ at time t is a vector with a dimension of $P = 6$.

We also consider three broader factors: (1) The temporal factor τ, as one of ancillary information, consists of HourOfDay, DayOfWeek and WeekdayOrWeekend. (2) The natural factor includes water flow and temperature. (3) The social factor that we incorporate is the electricity price. Each of them can significantly affect the robustness of the model and, consequently, the prediction performance. We use $\mathbf{e}$ to denote these three external factors.

Definition 3.1. **Hydropower generation forecasting**. Given a series of N past observations $\mathbf{X}_{t-1} = \{\mathbf{x}_{t-N}, \mathbf{x}_{t-N+1}, \cdots, \mathbf{x}_{t-1}\} \in \mathbb{R}^{N\times P}$, as well as external factors $\mathbf{e}_t$ at time t, the task of *hydropower generation forecasting* (HGF) is to learn a model $\mathcal{F}$ to predict the volume of hydropower generation $\hat{\mathbf{v}}_t$ at time t ($t > N$), formally defined as:

$$\hat{\mathbf{v}}_t = \mathcal{F}(\mathbf{X}_{t-1}|\mathbf{e}_t; \Omega) \tag{1}$$

where Ω denotes the parameters.

3.2 Background

3.2.1 Latent Variable Models. Variational Autoencoder (VAE) [15] combined with RNN models [5, 10], provides a possibility of learning the internal dependence of sequential data while preserving the distribution of latent variables $\mathbf{z}$. It has been widely used in modeling various time series data in the literature [9, 29, 32, 34]. Though facilitating coherent latent space learning, the learned representation of time series data is limited to the last hidden state $\mathbf{h}$ of the RNN encoder. In addition, these models usually make a strong assumption that the posterior can be decomposed into multiple independent factors. Since the variational inference needs to search the optimal posterior approximation within a parametric family of distributions, the models are not flexible enough to match the true posterior. This issue inspired normalizing flows [22], which learns an accurate posterior density $q_\phi(\mathbf{z}|\mathbf{X})$ of latent variables by stacking a series of invertible transformations as $\mathbf{z}^k = \beta^k\left(\beta^{k-1}\left(\ldots\beta^1\left(\mathbf{z}^0\right)\right)\right)$, where each β is an invertible transformation and ϕ are parameters.

3.2.2 Neural Ordinary Differential Equations. A recent work [4] considered the parameter updating in neural networks – which was previously done by backpropagation – as the process of solving ODEs. From the perspective of numerical methods, the discrete layers of neural networks (e.g., the hidden states $\mathbf{h}_t$ of the RNNs) can be regarded as an Euler discretization of a differential equation:

$$\frac{d\mathbf{h}(t)}{dt} = f_\omega(\mathbf{h}(t), t), \quad \text{where} \quad \mathbf{h}(t) = \mathbf{h}_t,$$
$$\mathbf{h}(t_1) = \mathbf{h}(t_0) + \int_{t_0}^{t_1} f_\omega(\mathbf{h}(t), t)dt, \tag{2}$$

where neural network f_ω is parameterized by ω specifying the continuous dynamics of the hidden states. By regarding the parameter update process of the neural ODE block as solving ODEs with numerical methods such as Euler, Runge–Kutta and adjoint method, it is possible for us to obtain the hidden states $\mathbf{h}(t)$ at *any* desired moment.

4 ARCHITECTURE & METHODOLOGY

Figure 3 overviews the architecture of our proposed method: DeepHydro. Due to the specifics of the hydropower flow data, e.g., large fluctuation range (from zero to full-load) and multivariate time series, we first pre-process the data to obtain the model inputs, denoted as $\mathbf{X}$. Overall, DeepHydro consists of three main components: (a) hydropower inference network; (b) external feature extraction network; and (c) multi-feature fusion predictor. Among them, (a) hydropower inference network is based on an encoder-decoder framework, which learns the latent representation of $\mathbf{X}$ based on variational inference. In the encoder, conditioned latent RNN (CL-RNN) is designed to transform the input $\mathbf{X}_{t-1}$ from observations to high-level representations including hidden states and latent variables in the z-space. Subsequently, we exploit a continuous normalizing flow to transform the latent variable $\mathbf{z}_{t-1}$ obtained by CL-RNN to a more accurate non-Gaussian posterior density $q_\phi(\mathbf{z}_{t-1}|\mathbf{X}_{t-1}, \mathbf{z}_{t-2})$. In the decoder, we first restructure $\mathbf{X}_{t-1}$ from $\mathbf{z}_{t-1}$ and then leverage an ODE network to extrapolate the latent variable $\mathbf{z}_t$ which generalizes the stochastic representation in previous steps to infer the latent variables at time t. For the external factor extraction network (EFEN), the natural and social factors such as water flow, temperature and electricity price are extracted with another ODE network. Finally, the multivariate hydropower is predicted by the predictor which is a simple MLP performing regression task combining the information output by CL-RNN $\mathbf{g}_{t-1}$, external factors $\mathbf{e}_t$ and ODE decoder $\mathbf{z}_t$. In the following, we detail the DeepHydro, while highlighting the advantages of our method compared to related models.

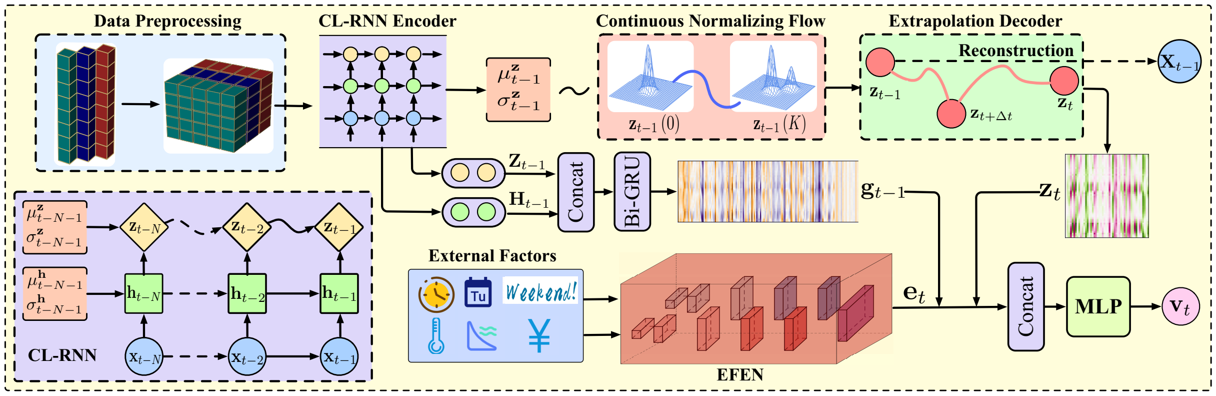

Data Preprocessing
CL-RNN Encoder
Continuous Normalizing Flow
Extrapolation Decoder
Reconstruction
$\mu^{\mathbf{z}}_{t-1}$ $\sigma^{\mathbf{z}}_{t-1}$
$\mathbf{z}_{t-1}(0)$
$\mathbf{z}_{t-1}(K)$
$\mathbf{z}_{t-1}$ $\mathbf{z}_{t+\Delta t}$ $\mathbf{z}_t$
$\mathbf{X}_{t-1}$
$\mathbf{Z}_{t-1}$
$\mathbf{H}_{t-1}$
Concat
Bi-GRU
$\mathbf{g}_{t-1}$
$\mathbf{Z}_t$
External Factors
Tu
Weekend!
¥
EFEN
$\mathbf{e}_t$
Concat
MLP
$\mathbf{v}_t$
CL-RNN
$\mathbf{z}_{t-N}$ $\mathbf{z}_{t-2}$ $\mathbf{z}_{t-1}$
$\mathbf{h}_{t-N}$ $\mathbf{h}_{t-2}$ $\mathbf{h}_{t-1}$
$\mathbf{x}_{t-N}$ $\mathbf{x}_{t-2}$ $\mathbf{x}_{t-1}$

Figure 3: Overall architecture of the proposed DeepHydro.

4.1 Hydropower Inference Network

One of the major challenges of hydropower generation forecasting is how to capture the informative temporal correlations and establish inherent relationship between various time series data. Towards that, we design a hydropower inference network based on an encoder-decoder framework, to learn both hidden states of observations and uncertainty regarding the hidden states. The inference network is composed by three modules, i.e., a Conditioned Latent RNN (CL-RNN) encoder, a continuous normalizing flow network, and an extrapolation ODE decoder.

4.1.1 Conditioned Latent RNN. CL-RNN acts as the hydropower data encoder. As shown in the lower left corner of Figure 3, CL-RNN feeds the inputs from discrete data space to obtain a continuous z-space representation. Different from traditional RNN encoders in which the latent variable $\mathbf{z}$ only relies on the last hidden state $\mathbf{h}$, CL-RNN captures the temporal dependencies among latent variables $\mathbf{Z}_{t-1} = \{\mathbf{z}_{t-N}, \mathbf{z}_{t-N+1}, \cdots, \mathbf{z}_{t-1}\}$, as well as the discrete h-space learned by GRU cells. In addition, we use numerical ODE solver to update the parameters inherent in RNNs.

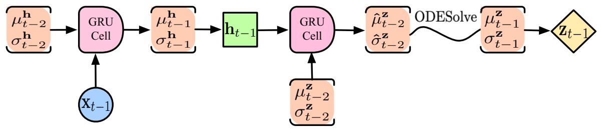


Figure 4: Detailed architecture of one-timestep in CL-RNN.

Figure 4 depicts the detailed structure of CL-RNN within one-time step. With the previous mean and variance $\left(\mu^{\mathbf{h}}_{t-2}, \sigma^{\mathbf{h}}_{t-2}\right)$ of hidden state $\mathbf{h}_{t-2}$ and hydropower input $\mathbf{x}_{t-1}$ at time $t-1$, we first feed them into the GRU cell to generate $\left(\mu^{\mathbf{h}}_{t-1}, \sigma^{\mathbf{h}}_{t-1}\right)$, as well as the hidden state $\mathbf{h}_{t-1}$ at time $t-1$. Then, in order to study the inherent relationships of multivariate hydropower flow data, we add another GRU cell, which takes $\left(\mu^{\mathbf{z}}_{t-2}, \sigma^{\mathbf{z}}_{t-2}\right)$ and $\mathbf{h}_{t-1}$ as the inputs, for latent representation learning. Subsequently, the second GRU cell output $\hat{\mu}^{\mathbf{z}}_{t-2}$ is taken as the initial value passing by a numerical ODE solver – where we use *Euler* method as its numerical solution – to produce the latent variable $\mu^{\mathbf{z}}_{t-1}$ at time $t-1$. As a result, CL-RNN completes a continuous latent representation learning conditioned on previous latent variables and hidden states, which is expected to capture stochastic temporal dependencies among observations while solving the problem of discrete recurrent networks from a dynamic system view. The above process is summarized as follows:

$$\begin{aligned}
\mu^{\mathbf{h}}_{t-1}, \sigma^{\mathbf{h}}_{t-1} &= \mathrm{GRUCell}\left(\mu^{\mathbf{h}}_{t-2}, \sigma^{\mathbf{h}}_{t-2}, \mathbf{x}_{t-1}\right),\\
\mathbf{h}_{t-1} &= \mu^{\mathbf{h}}_{t-1},\\
\hat{\mu}^{\mathbf{z}}_{t-2}, \hat{\sigma}^{\mathbf{z}}_{t-2} &= \mathrm{GRUCell}\left(\mu^{\mathbf{z}}_{t-2}, \sigma^{\mathbf{z}}_{t-2}, \mathbf{h}_{t-1}\right),\\
\mu^{\mathbf{z}}_{t-1} &= \mathrm{ODESolve}\left(f_\omega, \hat{\mu}^{\mathbf{z}}_{t-2}, t-2, t-1\right),\\
\sigma^{\mathbf{z}}_{t-1} &= \hat{\sigma}^{\mathbf{z}}_{t-2},\\
\mathbf{z}_{t-1} &= \mu^{\mathbf{z}}_{t-1},
\end{aligned} \tag{3}$$

where the ODE solver in CL-RNN is Euler; $t-2$ and $t-1$ denote the start time and end time of integration; and for GRU cell, we have the following formula (for the first GRU cell in Figure 4):

$$\begin{aligned}
\mathbf{i} &= \left[\mu^{\mathbf{h}}_{t-2}, \sigma^{\mathbf{h}}_{t-2}, \mathbf{x}_{t-1}\right],\\
\mathbf{o} &= \left[\mu^{\mathbf{h}}_{t-2}\mathbf{R}(\mathbf{i}), \sigma^{\mathbf{h}}_{t-2}\mathbf{R}(\mathbf{i}), \mathbf{x}_{t-1}\right],\\
\hat{\mu}, \hat{\sigma} &= \tanh\left(\mathbf{w_o o} + \mathbf{b_o}\right),\\
\mu^{\mathbf{h}}_{t-1} &= \left(1-\mathbf{U}(\mathbf{i})\right)\hat{\mu} + \mathbf{U}(\mathbf{i})\mu^{\mathbf{h}}_{t-2},\\
\sigma^{\mathbf{h}}_{t-1} &= \left|\left(1-\mathbf{U}(\mathbf{i})\right)|\hat{\sigma}| + \mathbf{U}(\mathbf{i})\sigma^{\mathbf{h}}_{t-2}\right|,
\end{aligned} \tag{4}$$

where $\mathbf{R}$ and $\mathbf{U}$ denote reset gate (determining which important information of current inputs should be remembered) and update gate (deciding how much previous information to save), respectively; $\mathbf{w_o}$ and $\mathbf{b_o}$ denote the learnable parameters and bias. Towards reproducibility, we provide a pseudo-code for the CL-RNN in Algorithm 1.

Until now, the mean $\mu^{\mathbf{z}}_{t-1}$ and the variance $\sigma^{\mathbf{z}}_{t-1}$ of latent variable $\mathbf{z}_{t-1}$ are available – which means that we can directly sample $\mathbf{z}_{t-1}$ from the learned distribution $q_\phi\left(\mathbf{z}_{t-1}|\mathbf{X}_{t-1}, \mathbf{z}_{t-2}\right)$ using the reparameterization trick [15]: $\mathbf{z}_{t-1} = \mu^{\mathbf{z}}_{t-1} + \sigma^{\mathbf{z}}_{t-1} * \epsilon$, where ϵ are samples from a standard Gaussian $\epsilon \sim \mathcal{N}(0, I)$.

4.1.2 Continuous Normalizing Flow. However, the real posterior distribution $q_\phi\left(\mathbf{z}_{t-1}|\mathbf{X}_{t-1}, \mathbf{z}_{t-2}\right)$ may not follow a typical Gaussian, which means we may impose a strong regulation of the latent variables from CL-RNN. To let our model approximate the true posterior distribution, we introduce a continuous normalizing flow (CNF) inspired by [3], which simplifies the computation of the change in $\mathbf{z}$ and its log densities to transform $q_\phi(\mathbf{z}_{t-1}|\mathbf{X}_{t-1}, \mathbf{z}_{t-2})$ in a continuous way. Let $\mathbf{z}_{t-1}(0) = \mathbf{z}_{t-1}$ be the initial value, and differential function β_ψ, parameterized by ψ, be uniformly *Lipschitz* continuous in both $\mathbf{z}$ and time k:

$$\frac{d\mathbf{z}_{t-1}(k)}{dk} = \beta_\psi(\mathbf{z}(k), k), \tag{5}$$

which describes a continuous-in-time transformation of $\mathbf{z}(k)$. According to the theorem of *instantaneous change of variables* [4], the change in log densities $\log q_\phi(\mathbf{z}_{t-1}(K))|\mathbf{X}_{t-1})$ also follows a differential equation:

$$\frac{d \log q_\phi(\mathbf{z}_{t-1}(k)|\mathbf{X}_{t-1})}{dk} = -\operatorname{tr}\left(\frac{\partial \beta_\psi}{\partial \mathbf{z}(k)}\right), \tag{6}$$

where tr denotes the trace operation – which replaces the intensive determinant computation in normalizing flows [22].

Then, the latent variables $\mathbf{z}_{t-1}(K)$ after time-length K and its log densities $\log q_\phi(\mathbf{z}_{t-1}(K)|\mathbf{X}_{t-1})$ can be computed as:

$$\mathbf{z}_{t-1}(K) = \mathbf{z}_{t-1}(0) + \int_0^K \beta_\psi(\mathbf{z}_{t-1}(k), k)dk, \tag{7}$$

$$\log q_\phi(\mathbf{z}_{t-1}(K)|\mathbf{X}_{t-1}) = \log q_\phi(\mathbf{z}_{t-1}(0)|\mathbf{X}_{t-1}) - \int_0^K \operatorname{tr}\left(\frac{\partial \beta_\psi}{\partial \mathbf{z}(k)}\right), \tag{8}$$

where we let $K = 1$ in DeepHydro – which, however, can be arbitrarily set as $K > 1$ for more transformations. Now, we can train the hydropower inference network by maximizing the evidence lower bound (ELBO) as follows:

$$\begin{aligned}\text{ELBO}(\theta, \phi) = \\ &\mathbb{E}_{q_\phi} \log\left[p_\theta(\mathbf{X}_{t-1}|\mathbf{z}_{t-1}(K))\right] + \mathbb{E}_{q_\phi} \log\left[p_\theta(\mathbf{z}_{t-1}(K))\right] \\ &- \mathbb{E}_{q_\phi}\left[q_\phi(\mathbf{z}_{t-1}(0)|\mathbf{X}_{t-1})\right] + \int_0^K \operatorname{tr}\left(\frac{\partial \beta_\psi}{\partial \mathbf{z}(k)}\right).\end{aligned} \tag{9}$$

After CNF, a more accurate non-Gaussian posterior density $q_\phi(\mathbf{z}_{t-1}(K)|\mathbf{X}_{t-1})$ can be obtained, resulting in a more informative latent representation $\mathbf{z}_{t-1}$ at time $t-1$.

4.1.3 Extrapolation Decoder. At this point, we have obtained probability density regarding $\mathbf{z}_{t-1}$ at time $t-1$, which generalizes the latent variables from previous observations $\mathbf{X}_{t-1}$. To predict the power at t, the latent representation at t is desirable. Furthermore, we would like to endow the inference network with the capability of autoregressively capturing conditioned dependencies and inferring future density of stochastic variables. Towards this goal, we propose an extrapolation network as the decoder, which infers the future latent variable $\mathbf{z}_t$ at time t by solving the ODEs:

$$\mathbf{z}_t = \text{ODESolve}(g_\nu, \mathbf{z}_{t-1}(K), t-1, t) \tag{10}$$

where g_ν denotes the differentiable neural network in ODE blocks with parameters ν. Here we use Dopris method [7] as the numerical solution in the extrapolation decoder.

4.2 External Factors Extraction Network

Recall that in addition to the hydropower generation data itself, the natural and social factors provide key information that influences power generation. As mentioned, the generation is strongly correlated to electricity demand and power consumption, which is highly seasonal and periodical in time of day (e.g., working hours vs. family hours), week and weekend, months/seasons (e.g., winter vs. summer). Similarly, the water flows (e.g., water inflow and water outflow) are always associated with generating power or water level adjustment. Moreover, the hotter (colder) the summer (winter), the more electricity people use, which directly determines the amount of electricity generated. Lastly, the social/economic factors such as the electricity price can also potentially affect the hydropower generation, i.e., the higher the price, the more profit that is driving hydro-plants to generate more electricity. In this work, we collectively refer to these influences as external features and design an external factor extraction network (EFEN), as shown in the lower middle part of Figure 3, for feature extraction and fusion.

In EFEN, we first embed the categorical temporal features (HourofDay and DayofWeek) to a low-level dimension. Then we let all external factors $\mathbf{e}$ pass by a 2-layer fully connected network (FCN), which gives an output with 128 dimensions at time t – and ELU [6] is used as the non-linear activation function. Rather than simply concatenating different types of feature vectors as most of existing works [16, 18], we append an ODE block to extract the different types of features that are usually entangled with each other (e.g., temporal, sequential and categorical features should be merged). The rationale behind this choice lies in the observation that FCN alone is too shallow to capture the non-liner interactions among different features. In contrast, the factors that pass our EFEN – an *infinite* deep network structure – can effectively impose real impacts on the prediction. Here we use fixed-adams as the ODE solver. We now obtain the learned external factor representation $\mathbf{e}_t$ that could be used for generated power prediction.

4.3 Hydropower Generation Predictor

Now, forecasting the generated hydropower is straightforward. Specifically, we combine the external factors features $\mathbf{e}_t$ extracted by EFEN, global temporal features $\mathbf{g}_{t-1}$ captured via CL-RNN, and latent representations $\mathbf{z}_t$ output by extrapolation decoder, for prediction. For $\mathbf{g}_{t-1}$, we first concatenate the sequence of $\mathbf{Z}_{t-1} = \{\mathbf{z}_{t-N}, \mathbf{z}_{t-N+1}, \cdots, \mathbf{z}_{t-1}\}$ and $\mathbf{H}_{t-1} = \{\mathbf{h}_{t-N}, \mathbf{h}_{t-N+1}, \cdots, \mathbf{h}_{t-1}\}$, which are outputs of CL-RNN. Then, the intermediate variables are fed into a Bi-GRU network for global temporal dependencies extraction. Finally, we use a simple MLP to predict the final hydropower generation prediction $\hat{\mathbf{v}}_t$ with three major representations:

$$\hat{\mathbf{v}}_t = \text{MLP}([\mathbf{z}_t, \mathbf{e}_t, \mathbf{g}_{t-1}]) \tag{11}$$

Objective: In this work, we aim to predict the hydropower generation at time t with proposed DeepHydro by minimizing the mean squared error between the real hydropower generation $\mathbf{v}_t$ and the predicted value $\hat{\mathbf{v}}_t$ while maximizing the ELBO in Eq.(9). Therefore, the loss function is defined as:

$$\mathcal{L}(\Omega) = \|\mathbf{v}_t - \hat{\mathbf{v}}_t\|_2^2 - \text{ELBO}(\theta, \phi). \tag{12}$$

The details of the training procedure of DeepHydro are summarized in Algorithm 2 in the Appendix.

5 EXPERIMENTS

We now present the results from our empirical evaluations of DeepHydro against several state-of-the-art baselines on time series forecasting, to demonstrate its superiority in terms of prediction performance. We also investigate the effectiveness of different components of DeepHydro and present intuitive explanations of the model performance.

5.1 Experimental Settings

Table 1: Statistics of datasets. Some social/economic information such as sale price are masked.

Dataset	DGS	PDS
Time span	P1:1/1/2017–12/31/2017 P2:1/1/2018–12/31/2018	P1:1/1/2017–12/31/2017 P2:1/1/2018–12/31/2018
Data type	Multivariate time series	Multivariate time series
Time interval	1 hour	1 hour
power generation (kwh)	[0.0, 2591.2]	[0.0, 668.0]
Water flow		
Water inflow	[0.0, 4640.0]	[0.0, 11579.0]
Water outflow	[55.1, 5220.0]	[53.8, 6490.0]
Generation flow	[55.1, 1730.0]	[55.1, 2480.0]
External Factors (meteorology, time and sale price)		
temperature/°C	[-24.4 21.7]	[-24.4 21.7]
HourOfDay	[0.0, 24.0]	[0.0, 24.0]
DayOfWeek	[1, 7]	[1, 7]
WeekdayOrWeekend	[0,1]	[0,1]

5.1.1 Datasets. To compare the performance of different methods, we conduct experiments on datasets from two large hydropower generation stations, i.e., Dagangshan Dam and Shenxigou Dam.

Recall that Figure 1 showed the map of hydropower stations on the Dadu River, a tributary of the Yangtze River in Sichuan Province, and that the basic cartographic properties and statistics of power generation were discussed in Section 3.1. The detailed description of the complete datasets that we used is shown in Table 1. One can observe that each dataset contains two types of multivariate time series data – hydropower flow generation and water flow information. As mentioned, the temporal dimension and some additional external factors that profoundly influence the hydropower generation (e.g., temperature and electricity price) are also modeled in our method. We split each dataset into two different periods (P1 and P2), both spanning one year. We use the first 41 weeks for training and the last 11 weeks for testing.

- **DGS** is collected from the Dagangshan Dam – an arch dam on the Dadu River in Shimian County, which houses a hydroelectric power station with 4 x 650 MW generators for a total installed capacity of 2,600 MW. This data contains the power generation spanning two years.
- **PDS** is gathered from the Shenxigou Hydropower plant in Hanyuan County, which has a total of 660 MW (4 x 165 MW) installed capacity. This data also contains the hydropower generation for two years.

5.1.2 Baselines. We compare DeepHydro with the following 9 benchmarks that are typically used for time series forecasting:

- **Historical Average (HA)**: models historical data average of a certain time period T to predict the hydropower generation in the next time step. In this paper, we set $T = 7$.
- **ARIMA**: is a well-known time series model that combines autogressive (AR) and moving average (MA) for prediction.
- **SARIMA**: is formed by including additional seasonal terms in the ARIMA model, which has been widely used in grid load prediction [3, 28].
- **LSTM**: is a RNN-based model, capturing the information of long-short term dependency with a vanilla LSTM, which has been commonly used for time series forecasting in the literature [27, 30, 31].
- **Bi-GRU**: is a bidirectional GRU model that concatenates the forward and backward hidden states for prediction.
- **Bi-GRU-ATT**: incorporates attention mechanism into the Bi-GRU network [5] which can selectively weight the important information and dependency, and has been used as building blocks for time series prediction and sequential recommendation models [17, 33, 35].
- **GRU-VAE**: aims at reconstructing the input from the latent variable of observations by VAE, which employs GRU as its encoder and decoder. This method has been used for time series modeling in [26, 32].
- **PlanarVAE**: learns the non-Gaussian posterior density with planar NF [22] to obtain more robust latent representation than vanilla VAE. The main architecture of modeling time series follows the GRU-VAE. This method has been used for sequential recommendation [34] and time series anomaly detection [29].
- **LatentODE** [23]: generalizes RNNs to have continuous-time hidden dynamics defined by ODEs. It considers latent representation $\mathbf{z}$ as a time series variable in RNN, and therefore can handle arbitrary time gaps between observations, and explicitly model the irregular sampled time series.

5.1.3 Variants. We implemented three variants of DeepHydro to investigate the effect of its different components:

- **DeepHydro-NF**: replaces Continuous NF with a planar NF in DeepHydro to learn the latent space distribution.
- **DeepHydro-RNN**: uses classic RNN such as LSTM as encoder instead of CL-RNN.
- **DeepHydro-NE**: removes external factor extraction network, i.e., it makes prediction purely based on the multivariate time series data.

5.1.4 Metrics. We evaluate all methods with three widely used metrics for time series prediction: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE), defined as: RMSE $= \sqrt{\frac{1}{M}\sum_{i=1}^{M}(\hat{\mathbf{v}}_t - \mathbf{v}_t)^2}$, MAE $= \frac{1}{M}\sum_{i=1}^{M}|\hat{\mathbf{v}}_t - \mathbf{v}_t|$, MAPE $= \frac{1}{M}\sum_{i=1}^{M}|\frac{\hat{\mathbf{v}}_t - \mathbf{x}_t}{\mathbf{v}_t}|$, where $\hat{\mathbf{v}}_t$ and $\mathbf{v}_t$ are predicted and ground-truth hydropower generation volume at time t, and M is the collection of testing samples.

5.2 Experimental Results

The details of our evaluation follow.

5.2.1 Performance Comparison. Table 2 reports the performance of all methods on the two datasets, from which we have the following observations. First, traditional methods such as HA, ARIMA and SARIMA perform poorly because they cannot capture nonlinear

Table 2: Performance comparisons on DGS and PDS over two different time spans. Pair t-test was performed for statistical significance of the results ($p < 0.005$).

Datasets	DGS						PDS					
Time span	P1			P2			P1			P2		
Metrics / Method	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE
HA	491.8	390.1	0.823	533.8	434.9	0.963	139.3	113.6	0.590	139.3	114.6	0.446
ARIMA	242.5	165.4	0.733	278.7	227.6	0.882	72.1	59.5	0.322	74.8	63.6	0.283
SARIMA	239.4	161.8	0.735	273.3	220.5	0.884	71.3	58.1	0.318	73.4	62.8	0.273
LSTM	200.3	143.9	0.631	235.4	182.3	0.839	51.6	35.8	0.208	53.3	37.2	0.182
Bi-GRU	197.8	137.6	0.605	233.2	174.6	0.702	50.8	35.5	0.201	52.6	36.7	0.179
Bi-GRU-ATT	193.8	136.7	0.580	232.0	172.2	0.643	50.1	34.4	0.204	51.9	36.6	0.177
GRU-VAE	195.3	138.6	0.621	232.1	173.2	0.669	50.2	34.2	0.201	52.1	36.4	0.178
PlanarVAE	192.6	135.5	0.578	230.7	170.4	0.630	49.1	33.9	0.198	50.8	35.5	0.173
LatentODE	190.3	134.3	0.581	227.1	167.5	0.634	48.6	33.1	0.201	50.3	35.1	0.171
DeepHydro	**144.1**	**106.4**	**0.412**	**182.2**	**133.4**	**0.537**	**35.2**	**25.9**	**0.169**	**38.8**	**28.7**	**0.153**

temporal dependencies among multivariate power generation and fail to leverage useful external factors.

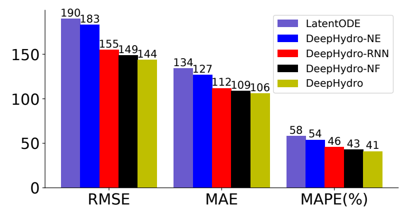
Second, modeling electricity power with simply deep recurrent networks, e.g., LSTM, Bi-GRU, Bi-GRU-ATT, does not yield good prediction performance, due to the incapability of modeling stochasticity of electricity power.

Third, the effect of other stochastic approaches such as GRU-VAE and PlanarVAE are limited because the autoregressive models (LSTM or GRU) are powerful enough to decode the entire time series, resulting in uselessness of stochastic latent factors, as being observed in other sequence modeling tasks [12, 34]. LatentODE, which usually performs the second best, is capable of defining a generative process over time series. However, it is originally designed for irregularly-sampled data with time decay factors which are not the case for electricity power – the generated power is uniformly and timely measured in modern hydropower stations.

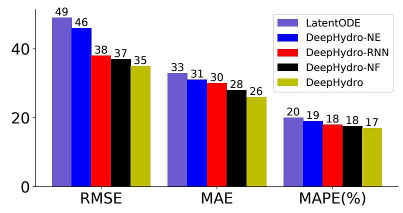
Fourth, DeepHydro significantly outperforms all baselines on all metrics in both datasets, demonstrating the effectiveness of DeepHydro in dealing with multivariate time series data. For example, DeepHydro yields 27.6%, 21.8%, and 15.9% improvement over the second best approach in terms of RMSE, MAE, and MAPE on PDS-P1, respectively.

Finally, it is worthwhile to note that our method can adapt to hydro-plants of different scales and possibly reap more performance gain on data with larger ranges. For example, the performance gain of DeepHydro on larger scale hydro-plant DGS (P1) increases by 29.1% in terms of MAPE.

5.2.2 *Ablation Study.* Figure 5 shows the effect of the different building blocks of DeepHydro. Clearly, external factors play an important role on power prediction, which is verified by the significant drop of performance of DeepHydro-NE. The advantage of our latent model CL-RNN can be observed by the reduced performance of DeepHydro-RNN. This result indicates that modeling the temporal dependence among latent variables (in addition to hidden states) benefits model expressiveness, i.e., capturing more informative signals from historical stochastic variables. Moreover, replacing continuous NF with planar NF causes performance degradation.



(a) Ablation-DGS-P1.



(b) Ablation-PDS-P1.

Figure 5: The ablation analysis on two datasets.

This reveals the effect of continuous transformation of stochastic variables, which, arguably, better reflects the natural continuous-time dynamics of industrial time series data of electricity power.

Table 3: Influence of external factors.

External Factors	RMSE (%)	MAE(%)	MAPE(%)
Water flow	18.3	14.9	15.2
Temporal information	11.1	4.6	5.0
Temperature	8.2	4.2	4.8
Electricity price	6.5	3.2	4.3

5.2.3 *Influence of Different Factors.* We investigated the influence of different factors by individually removing them from DeepHydro and present the results in Table 3 – each value indicating the performance gain (i.e., reduced errors) if incorporating corresponding factor back into DeepHydro. The water flow (both inflow and outflow) is the strongest indicator of power generation prediction – as expected, since the hydropower electricity is generated by the water. Furthermore, temporal information, temperature and electricity price are also correlated to hydropower generation, e.g., seasonal fluctuation, residential and industrial demand, etc., all of which should be taken into account for fine-grained power forecasting.

6 CONCLUSION

We presented DeepHydro – a novel framework for power generation forecasting. Our model is built upon a new designed RNN that captures the temporal dependencies among latent variables reflecting the distribution of observations. DeepHydro is a normalizing flow based generative approach capable of alleviating the agnostic posterior estimation problem. We also provide an alternative view of optimizing time series models through treating the discrete layers of neural networks as solving ODEs, which could better approximate the dynamics system and continuous industrial data such as hydropower electricity. Experiments conducted on real-world hydro-plant datasets prove the effectiveness of our method. *Our DeepHydro model has been successfully deployed on the hydropower data mining platform*, and is continuously learning with new hydropower data. Currently, we are exploring better optimization methods to improve power forecasting for a series of cascaded hydro-plants in the company, which is more challenging since we need to globally coordinate the data among stations. In addition, generalizing our method to other time series tasks such as electricity price movement prediction and precipitation/water flow forecasting are topics of particular interest that could further improve the power generation forecasting and marketing management.

ACKNOWLEDGMENT

This work was supported by National Natural Science Foundation of China (Grant No.61602097, U19B2028 and 61772117), and NSF grants CNS 1646107.

REFERENCES

[1] Anastasia Borovykh, Sander Bohte, and Cornelis W Oosterlee. 2017. Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691* (2017).

[2] Juan Chen and Ping-An Zhong. 2019. A multi-time-scale power prediction model of hydropower station considering multiple uncertainties. *Science of The Total Environment* 677 (2019), 612–625.

[3] Pudi Chen, Shenghua Liu, Chuan Shi, Bryan Hooi, Bai Wang, and Xueqi Cheng. 2018. NeuCast: Seasonal Neural Forecast of Power Grid Time Series. In *IJCAI*. 3315–3321.

[4] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. 2018. Neural ordinary differential equations. In *NeuIPS*.

[5] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[6] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *arXiv preprint arXiv:1511.07289* (2015).

[7] John R Dormand and Peter J Prince. 1980. A family of embedded Runge-Kutta formulae. *Journal of computational and applied mathematics* 6, 1 (1980), 19–26.

[8] Chenyou Fan, Yuze Zhang, Yi Pan, Xiaoyue Li, Chi Zhang, Rong Yuan, Di Wu, Wensheng Wang, Jian Pei, and Heng Huang. 2019. Multi-Horizon Time Series Forecasting with Temporal Attention Learning. In *KDD*. 2527–2535.

[9] Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. 2016. Sequential neural models with stochastic layers. In *NIPS*. 2199–2207.

[10] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[11] Bryan Hooi, Hyun Ah Song, Amritanshu Pandey, Marko Jereminov, Larry Pileggi, and Christos Faloutsos. 2018. Streamcast: Fast and online mining of power grid time sequences. In *SDM*. SIAM, 531–539.

[12] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *ICML*. 1587–1596.

[13] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. 2018. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *KDD*. 387–395.

[14] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv: 1412.6980* (2014).

[15] Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *ICLR*.

[16] Yuxuan Liang, Kun Ouyang, Lin Jing, Sijie Ruan, Ye Liu, Junbo Zhang, David S Rosenblum, and Yu Zheng. 2019. UrbanFM: Inferring Fine-Grained Urban Flows. In *KDD*. 3132–3142.

[17] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: short-term attention/memory priority model for session-based recommendation. In *KDD*. 1831–1839.

[18] Zheyi Pan, Yuxuan Liang, Weifeng Wang, Yong Yu, Yu Zheng, and Junbo Zhang. 2019. Urban traffic prediction from spatio-temporal data using deep meta learning. In *KDD*. 1720–1730.

[19] Yue Pang, Bo Yao, Xiangdong Zhou, Yong Zhang, Yiming Xu, and Zijing Tan. 2018. Hierarchical Electricity Time Series Forecasting for Integrating Consumption Patterns Analysis and Aggregation Consistency.. In *IJCAI*. 3506–3512.

[20] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. 2019. Normalizing Flows for Probabilistic Modeling and Inference. *arXiv preprint arXiv:1912.02762* (2019).

[21] Syama Sundar Rangapuram, Matthias W. Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. 2018. Deep State Space Models for Time Series Forecasting. In *NeurIPS*. 7796–7805.

[22] Danilo Rezende and Shakir Mohamed. 2015. Variational Inference with Normalizing Flows. In *ICML*. 1530–1538.

[23] Yulia Rubanova, Tian Qi Chen, and David K Duvenaud. 2019. Latent Ordinary Differential Equations for Irregularly-Sampled Time Series. In *NeurIPS*. 5321–5331.

[24] Tárik S Salem, Karan Kathuria, Heri Ramampiaro, and Helge Langseth. 2019. Forecasting intra-hour imbalances in electric power systems. In *AAAI*, Vol. 33. 9595–9600.

[25] Rajat Sen, Hsiang-Fu Yu, and Inderjit S Dhillon. 2019. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. In *NeurIPS*. 4838–4847.

[26] Maximilian Sölch, Justin Bayer, Marvin Ludersdorfer, and Patrick van der Smagt. 2016. Variational inference for on-line anomaly detection in high-dimensional time series. *arXiv preprint arXiv:1602.07109* (2016).

[27] Dongjin Song, Ning Xia, Wei Cheng, Haifeng Chen, and Dacheng Tao. 2018. Deep r-th Root of Rank Supervised Joint Binary Embedding for Multivariate Time Series Retrieval. *KDD* (2018), 2229–2238.

[28] Hyun Ah Song, Bryan Hooi, Marko Jereminov, Amritanshu Pandey, Larry Pileggi, and Christos Faloutsos. 2017. PowerCast: Mining and forecasting power grid sequences. In *ECML-PKDD*. 606–621.

[29] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. 2019. Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network. In *KDD*. 2828–2837.

[30] Jingyuan Wang, Yang Zhang 0032, Ke Tang, Junjie Wu, and Zhang Xiong. 2019. AlphaStock: A Buying-Winners-and-Selling-Losers Investment Strategy using Interpretable Deep Reinforcement Attention Networks. *KDD* (2019), 1900–1908.

[31] Jingyuan Wang, Ze Wang, Jianfeng Li, and Junjie Wu. 2018. Multilevel Wavelet Decomposition Network for Interpretable Time Series Analysis. In *KDD*. 2437–2446.

[32] Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, et al. 2018. Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. In *WWW*. 187–196.

[33] Yumo Xu and Shay B Cohen. 2018. Stock Movement Prediction from Tweets and Historical Prices. *ACL* (2018), 1970–1979.

[34] Ting Zhong, Zijing Wen, Fan Zhou, Goce Trajcevski, and Kunpeng Zhang. 2020. Session-based Recommendation via Flow-based Deep Generative Networks and Bayesian Inference. *Neurocomputing* (2020).

[35] Ali Ziat, Edouard Delasalles, Ludovic Denoyer, and Patrick Gallinari. 2017. Spatio-Temporal Neural Networks for Space-Time Series Forecasting and Relations Discovery. In *ICDM*. 705–714.

A ALGORITHMS

Algorithm 1 describes the details of CL-RNN while the training procedure of DeepHydro is outlined in Algorithm 2.

```
Algorithm 1: CL-RNN
Input: multivariate hydropower data: $\mathbf{X}_{t-1}$, means $\mu^{\mathbf{h}}_{t-N-1}$
       and $\mu^{\mathbf{z}}_{t-N-1}$, variances $\sigma^{\mathbf{h}}_{t-N-1}$ and $\sigma^{\mathbf{z}}_{t-N-1}$,
       differential function $f_\omega$.
Output: $\mathbf{Z}_{t-1} = \{\mathbf{z}_{t-N}, \mathbf{z}_{t-N+1}, \cdots, \mathbf{z}_{t-1}\}$,
        $\mathbf{H}_{t-1} = \{\mathbf{h}_{t-N}, \mathbf{h}_{t-N+1}, \cdots, \mathbf{h}_{t-1}\}$ and $(\mu^{\mathbf{z}}_{t-1}, \sigma^{\mathbf{z}}_{t-1})$.
1  Initialize $\mu^{\mathbf{h}}_{t-N-1} = \mathbf{0}, \sigma^{\mathbf{h}}_{t-N-1} = \mathbf{0}, \mu^{\mathbf{z}}_{t-N-1} = \mathbf{0}, \sigma^{\mathbf{z}}_{t-N-1} = \mathbf{0}$;
   foreach $i$ in $[t-N, \cdots, t-1]$ do
2      $\mu^{\mathbf{h}}_i, \sigma^{\mathbf{h}}_i, \mathbf{h}_i = \text{GRUCell}\left(\mu^{\mathbf{h}}_{i-1}, \sigma^{\mathbf{h}}_{i-1}, \mathbf{x}_i\right)$;
3      $\hat{\mu}^{\mathbf{z}}_{i-1}, \hat{\sigma}^{\mathbf{z}}_{i-1}, \mathbf{z}_{i-1} = \text{GRUCell}\left(\mu^{\mathbf{z}}_{i-1}, \sigma^{\mathbf{z}}_{i-1}, \mathbf{h}_i\right)$;
4      $\mathbf{z}_i = \text{ODESolve}\left(f_\omega, \mathbf{z}_{i-1}, i-1, i\right)$;
5      $\mu^{\mathbf{z}}_i = \mathbf{z}_i, \sigma^{\mathbf{z}}_i = \hat{\sigma}^{\mathbf{z}}_{i-1}$;
6      Save hidden state $\mathbf{h}_i$ and latent variable $\mathbf{z}_i$;
7  end
8  Concatenate all hidden states $\mathbf{h}_{t-N:t-1}$ to $\mathbf{H}_{t-1}$;
9  Concatenate all latent variables $\mathbf{z}_{t-N:t-1}$ to $\mathbf{Z}_{t-1}$;
10 return $\mathbf{Z}_{t-1}, \mathbf{H}_{t-1}$ and $\left(\mu^{\mathbf{z}}_{t-1}, \sigma^{\mathbf{z}}_{t-1}\right)$.
```

```
Algorithm 2: DeepHydro
Input: multivaraiate hydropower data $\mathbf{X}_{t-1}$; external
       factors $\mathbf{e}_t$;
Output: predicted hydropower generation $\hat{\mathbf{v}}_t$.
1  Initialize all parameters $\Omega$ of DeepHydro;
2  while not converged do
3      Obtain $\mathbf{Z}_{t-1}, \mathbf{H}_{t-1}$ and $(\mu^{z}_{t-1}, \sigma^{z}_{t-1})$ by Algorithm 1;
4      Sample $\mathbf{z}_{t-1}(0) = \mu^{z}_{t-1} + \sigma^{z}_{t-1} \times \epsilon, \epsilon \sim \mathcal{N}(0, I)$;
5      Obtain $\mathbf{z}_{t-1}(K)$ using CNF by maximizing ELBO (Eq.(9));
6      Reconstruct $\mathbf{X}_{t-1}$ with latent variable $\mathbf{z}_{t-1}(K)$;
7      Obtain $\mathbf{z}_t$ with extrapolation decoder, cf. Eq.(10);
8      Obtain $\mathbf{g}_{t-1}$ via a Bi-GRU with input $[\mathbf{Z}_{t-1}, \mathbf{H}_{t-1}]$;
9      Obtain external features $\mathbf{e}_t$ with EFEN;
10     Predict hydropower generation $\hat{\mathbf{v}}_t$ with Eq.(11);
11     Update $\Omega$ by maximizing the objective in Eq.(12).
12 end
```

B DATA PREPROCESSING

The range of hydropower generation and water flow value are wide (cf. Table 1). A straightforward method is to utilize the Min-Max normalization to regularize the value into a range of [0, 1] during training stage and re-scale the predicted value back to the real value during testing. We segment the time series into fixed-length sequences with length N ($N = 1 \times 24 \times 7$), i.e., a sequence consists of the power generated in 1 week with observation interval of 1 hour. Therefore, each batch of training data has the shape $\mathbf{X}_{t-1} \in \mathbb{R}^{B \times N \times P}$, where B denotes the batch size. In addition, we adopt *Lagrange interpolation polynomial* method to interpolate the anomalous/missing value appeared in the data. Moreover, some temporal feature (e.g., HourOfDay and DayOfweek) are embedded into the low-dimensional vectors for training.

C IMPLEMENTATION NOTES

We implement DeepHydro and baselines using PyTorch1.1.0 and Python3.6 on Ubuntu 16.04 OS with a single NVIDIA GeForce GTX 2080ti GPU. The batch size is by default set to 128. The learning rate is set to 0.0001 at the beginning, and then decayed 50% every 20 epochs. ADAM optimizer [14] is used with the setting of $\beta_1 = 0.9$ and $\beta_2 = 0.999$. For all methods, we run 150 epochs in total for training the model and then verify the model on testing data.

The ODE solvers in CL-RNN was implemented by the *Euler* method, while in Continuous NF and extrapolation decoder we used *Dopris* solver [7]. In the external factor extraction network, *fixed-adams* is employed to solve the ODEs. These choices are based on the compromises between computational precision and model efficiency, which means other sophisticated numerical methods such as *adjoint* method, *Runge–Kutta*, *midpoint*, *tsit5* and *adaptive-heun* can be easily used to replace the methods used in our experiments. All of these ODE solvers can be easily implemented with the open source package[3].

[3]https://github.com/rtqichen/torchdiffeq