# Applying SVGD to Bayesian Neural Networks for Cyclical Time-Series Prediction and Inference

**Xinyu Hu**[1,2†] **Paul Szerlip**[2] **Theofanis Karaletsos**[2] **Rohit Singh**[2]

[1]Columbia University, New York, NY [2]Uber AI Labs, San Francisco, CA

xh2194@cumc.columbia.edu, {pas,theofanis,rohits}@uber.com

## 1 Introduction

Accurate and robust prediction of time-series data has shown meaningful impact in various applications [1]. For example, at Uber, predicting rider demand accurately benefits supply planning and resource allocation. Inaccurate predictions and confidence miscalibrations in the estimated predictions can lead to suboptimal decision making which may further result in either over or under supply. Ultimately, in an industrial setting such miscalibrations can result in extra cost to the company or to the customers. However, it is challenging to predict quantities like demand accurately due to potentially unknown exogenous variables that cause anomalous patterns and contribute to prediction variability. Although, there are many popular and successful recurrent or modified convolutional network models for capturing time dynamics [2, 3], they typically are trained using maximum likelihood and suffer from overconfidence. Moreover, such point estimates are typically insufficient to quantify prediction variability. While there are important successes using frequentist ensembles [4], the Bayesian framework is a natural choice for modeling the prediction uncertainty and interpreting the estimates. Recently, many attempts have been made to adapt existing Bayesian techniques to model neural networks [5–10], referred to as Bayesian neural networks (BNNs). Variational inference (VI) is often used to approximate the posterior distribution over parameters efficiently [7]. One particular posterior approximation for BNNs is the Monte Carlo Dropout [5, 11] and has been applied to time-series forecasting as well [12]. However, both accuracy and scalability of VI depend on the particular approximating distribution. In this work, we employ Stein variational gradient descent (SVGD), which is a generalized nonparametric VI algorithm for approximating continuous distributions [13]. SVGD has the advantage of not requiring knowledge of the explicit form of the posterior distribution and provides a theoretically guaranteed weak convergence of the samples [14]. By assuming independent prior distributions and using the radial basis function (RBF) kernel, SVGD is fast and scalable to large neural networks and offers an elegant and efficient solution for forecasting quantities like rider demand while also modeling the prediction uncertainty.

We propose a regression-based BNN model to predict spatiotemporal quantities like hourly rider demand with calibrated uncertainties. The main contributions of this paper are (i) A feed-forward deterministic neural network (DetNN) architecture that predicts cyclical time series data with sensitivity to anomalous forecasting events; (ii) A Bayesian framework applying SVGD to train large neural networks for such tasks, capable of producing time series predictions as well as measures of uncertainty surrounding the predictions. Experiments show that the proposed BNN reduces average estimation error by 10% across 8 U.S. cities compared to a fine-tuned multilayer perceptron (MLP), and 4% better than the same network architecture trained without SVGD.

## 2 Bayesian neural network

The proposed neural network consists of an encoder to learn the hidden features and a decoder to predict time series, as shown in Figure 1. The outcome of interest is a vector of continuous variables $\boldsymbol{y}$. The input features are denoted as $\boldsymbol{x}$. The parameter of the model $\boldsymbol{\theta} = (\boldsymbol{w}, \boldsymbol{\Sigma})$ consists of the neural network parameter $\boldsymbol{w}$ and the noise covariance matrix $\boldsymbol{\Sigma}$. The regression model is specified as:

$$\boldsymbol{y}|\boldsymbol{\theta} = f_{\boldsymbol{w}}(\boldsymbol{x}) + \boldsymbol{\epsilon}, \tag{1}$$

[†] This work was done while the first author was employed at Uber AI Labs, San Francisco, USA.

Third workshop on Bayesian Deep Learning (NeurIPS 2018), Montréal, Canada.

where $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \boldsymbol{\Sigma})$. In equation (1), $f_{\boldsymbol{w}}(\boldsymbol{x})$ denotes the output of the neural network. The predicted sequence is modeled independently across time points through the neural network. The correlation among the time points could be modeled through a structured $\boldsymbol{\Sigma}$, but in our experiments, $\boldsymbol{\Sigma}$ is assumed to be a $d \times d$ positive definite diagonal matrix for simplicity and computational efficiency. The $i$th outcome $\boldsymbol{y}_i$, is a vector of length $d$ and is assumed to be independently but not identically sampled from a multivariate Gaussian distribution $N(f_{\boldsymbol{w}}(\boldsymbol{x}_i), \boldsymbol{\Sigma})$ for $i = 1, \ldots, N$.

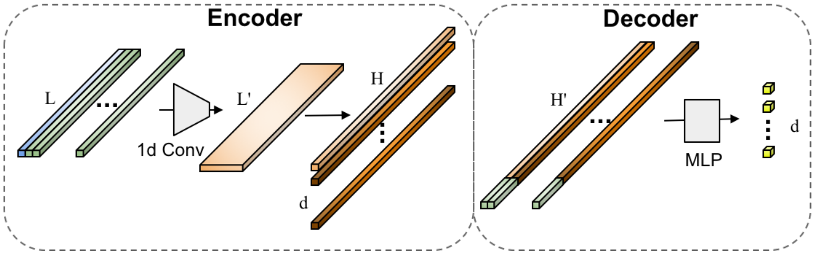


Figure 1: Neural network structure: in the encoder, the blue(leftmost) bar indicates the time series of interest to predict, for example hourly rider demand; the green bars(after the first bar) indicate the sequential location information, for example, one-hot encoded vector indicating if the hour of the day or the day of the week is a special time window like a holiday we need to pay attention to. The inputs are passed into the 1d convolutional layers to learn the hidden features. $d$ parallel linear functions are used to map from the same hidden features to $d$ reconstructed features. In the decoder, those reconstructed features are concatenated with the sequential location information at the prediction hour given the information is known in advance. Then the learned features are passed into a MLP one by one independently to predict the outcome.

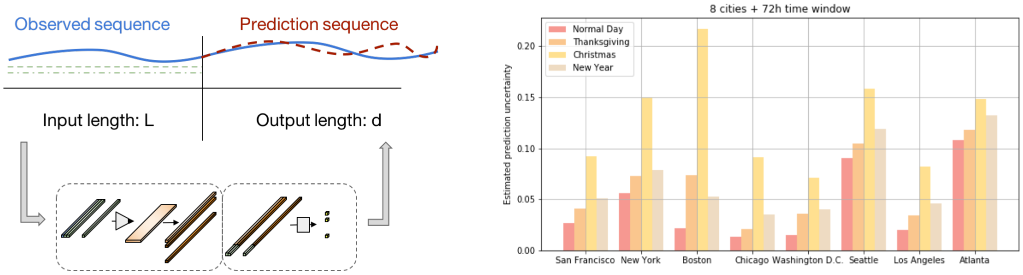
A Bayesian framework is imposed on the model (1) by assigning prior distributions to the model parameters. The prior of the neural network parameters is given by $\boldsymbol{w}|\alpha \sim N(\mathbf{0}, \alpha^{-1}\boldsymbol{I})$, $\alpha \sim \Gamma(a_0, b_0)$; the prior of the noise covariance $\boldsymbol{\Sigma} = diag(\sigma_1^2, \ldots, \sigma_d^2)$, $\sigma_i^{-2} \sim \Gamma(a_1, b_1)$ for $i = 1, \ldots, d$. $\boldsymbol{w}$ and $\boldsymbol{\Sigma}$ are esimated to maximzie the joint log-likelihood with different learning rates.

During training, $n$ such neural networks are built via SVGD. When a new data point is passed into the trained network, $n$ posterior samples of $\boldsymbol{\theta}$ are obtained for inference. The predicted outcome is estimated as $\mathbb{E}(\boldsymbol{y}) = \mathbb{E}_{\boldsymbol{\theta}}(\mathbb{E}(\boldsymbol{y}|\boldsymbol{\theta}))$. The prediction variability is decomposed into three sources: model uncertainty, model misspecification and inherent noise. Assuming there is no misspecification, the prediction variability can be estimated through $n$ SVGD samples by $Cov(\boldsymbol{y}) = Cov_{\boldsymbol{\theta}}(\mathbb{E}(\boldsymbol{y}|\boldsymbol{\theta})) + \mathbb{E}_{\boldsymbol{\theta}}(Cov(\boldsymbol{y}|\boldsymbol{\theta}))$, where $Cov_{\boldsymbol{\theta}}(\mathbb{E}(\boldsymbol{y}|\boldsymbol{\theta}))$ represents the model uncertainty and $\mathbb{E}_{\boldsymbol{\theta}}(Cov(\boldsymbol{y}|\boldsymbol{\theta}))$ represents the inherent noise. Under the assumption of diagonal noise covariance, constructing a credible region is equivalent to constructing a credible interval at each dimension. The $\alpha$-level credible interval is estimated as $[\hat{y} - z_{\alpha/2}\hat{\eta}, \hat{y} + z_{\alpha/2}\hat{\eta}]$, where $z_{\alpha/2}$ is the upper $\alpha/2$ quantile of a standard Gaussian, $\hat{y} = \frac{1}{n}\sum_{i=1}^{n} f_{\hat{\boldsymbol{w}}_i}(x)$, $\hat{\eta} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{\sigma}_i^2 + f_{\hat{\boldsymbol{w}}_i}(x)^2) - (\frac{1}{n}\sum_{i=1}^{n} f_{\hat{\boldsymbol{w}}_i}(x))^2}$.

The detailed BNN via SVGD algorithm is shown in the Appendix. In all experiments, an RBF kernel $k(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) = exp(-\frac{1}{h}||\boldsymbol{\theta}_i - \boldsymbol{\theta}_j||_2^2)$ is used with the bandwidth $h = H^2/logn$ where H is the median of the pairwise distances between the SVGD samples. The bandwidth is changed adaptively over iterations. The Stein operator depends on the target posterior only through the score function $\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|\mathcal{D}) = \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}, \mathcal{D})$, where $\mathcal{D} = (x_i, y_i)_{i=1}^{N}$. Thus the exact posterior distribution is not required to be represented explicitly to generate approximate samples from it. To calculate the gradient of $\log p(\boldsymbol{\theta}|\mathcal{D})$, we need all the training data. But during training mini-batches of size $b$ are passed into the neural networks. This is fixed by approximating the data likelihood by $\log p(\boldsymbol{\theta}, \mathcal{D}) \approx \log p_0(\boldsymbol{\theta}) + \frac{N}{b}\sum_{k=1}^{b} \log p(\mathcal{D}_k|\boldsymbol{\theta})$, where $p_0(\boldsymbol{\theta})$ is the prior distribution of $\boldsymbol{\theta}$.

## 3 Experiments

We predict the hourly rider demand across 8 U.S. cities along with quantified prediction variability. The data used in the experiment is the hourly number of completed trips at Uber from 2014 to 2018 among 8 U.S. cities. The dataset is split sequentially into 50%/25%/25% train, validation, and test data and preprocessed to fit the Gaussian assumption. The hourly demand data exhibits a strong 24-hour cyclical pattern with jitters around some special time windows. For example, the demand drops during Thanksgiving and rises dramatically after New Year's eve. To handle the important time windows, extra one-hot encoded channels are added to the input to the convolutional layers. As illustrated in Figure 2 (a), the input of the model consists of an hourly demand sequence and several sequential location sequences indicating the hour of the day, day of the week etc., the output is the predicted demand sequence. The difference of the prediction variability of a 72-hour window around holidays and a non-holiday using the previous 144-hour input is shown in Figure 2 (b). The estimated variability is always higher around holidays, especially around Christmas, compared to the one around a normal day in all 8 cities, meaning that the BNN model is less confident about predicting a holiday than predicting a non-holiday, as expected.



(a) Overview

(b) Estimated prediction variability

Figure 2: (a) is an overview of the prediction process: the blue line indicates the demand sequence, the green dotted lines indicate the one-hot encoded sequential location input, the red dashed line indicates the predicted sequence; (b) shows the estimated variability when predicting 72 hours around holidays and a non-holiday on the test data in 8 cities using a BNN model with 30 particle samples.

Table 1: WMAPE comparison for MLP, DetNN and BNN.

| WMAPE | MLP | DetNN | BNN-10 | BNN-30 | BNN-50 |
|---|---|---|---|---|---|
| San Francisco | 0.0718 | 0.0678 | 0.0658 | 0.0657 | 0.066 |
| New York City | 0.0773 | 0.0747 | 0.0763 | 0.0743 | 0.0743 |
| Boston | 0.0823 | 0.079 | 0.0778 | 0.077 | 0.0768 |
| Chicago | 0.0935 | 0.084 | 0.0807 | 0.0802 | 0.0795 |
| Washington D.C. | 0.079 | 0.0742 | 0.0758 | 0.0737 | 0.0737 |
| Seattle | 0.0822 | 0.0813 | 0.0777 | 0.0772 | 0.077 |
| Los Angeles | 0.0792 | 0.0703 | 0.0655 | 0.0647 | 0.065 |
| Atlanta | 0.0933 | 0.0877 | 0.0825 | 0.0805 | 0.0813 |
| Average | 0.0823 | 0.0774 | 0.0753 | 0.0741 | 0.0742 |

The performance of the BNN model with 10, 30 and 50 particle samples, referred to as BNN-10, BNN-30 and BNN-50, is shown in Table 1. With only one SVGD sample, a reasonably well maximum a posteriori estimate can be obtained. The sample size in the experiment is chosen arbitrarily as a balance of prediction performance and computational efficiency. The input sequence length is 144 hours, the output sequence length is 6 hours. The weighted mean absolute percentage error (WMAPE) $\sum_{i=1}^{N}|y_i - \hat{y}_i| / \sum_{i=1}^{N}|y_i|$, where $y$ and $\hat{y}$ are the true and predicted outcome, is used as the evaluation metric. Table 1 shows a summary of averaged WMAPE across the 6-hour prediction window. As performance benchmarks, we also show the results of a MLP model and a DetNN model which has the same network structure as the BNN. The hyper-parameters are tuned separately for each model. Averaging across all cities, DetNN achieves 6% decrease in WMAPE from MLP, and BNN-30 achieves 4% decrease from DetNN. Bayesian inference of parameters using SVGD further improves the DetNN performance with an additional benefit of quantified prediction variability.

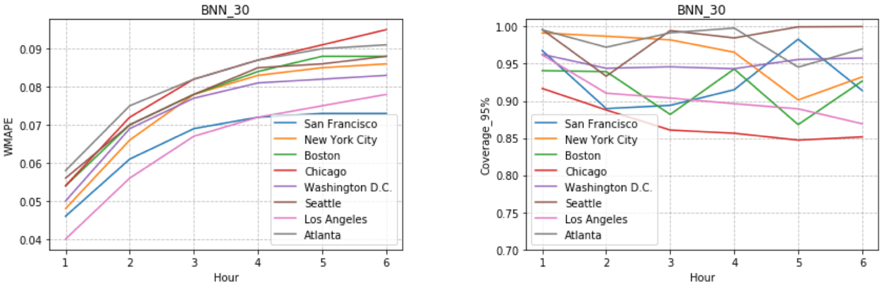


Figure 3: Estimated WMAPE (left) and 95% coverage probability (right) with a 6-hour prediction window.

Figure 3 shows the estimated WMAPE and 95% coverage probability from BNN-30 over a 6-hour prediction window. 95% coverage probability means the percentage that the true value is within the 95% credible band. The WMAPE increases when predicting further, but the coverage probability does not necessarily decrease. Even if the point estimation is not good enough, the BNN model could report low confidence by having a high variability around the estimation, thus facilitating better informed supply allocation.

## 4 Discussion

We have proposed a particular neural network architecture aimed at spatiotemporal modeling with cyclical components applied to the example of estimating demand, which is an important problem in the ridesharing space. We furthermore perform Bayesian inference on the proposed model using a variant of SVGD that gives us promising performance gains. Our experimental results indicate the advantage of Bayesian estimation using SVGD for our model, which encourages further investigation into the issue of modeling uncertainty for industrial scale problems. There remain interesting research questions to be investigated further. For example, in the future, we will explore different correlation structures to model time series data and investigate the use of more structured prior distributions instead of the independent prior assumption we are currently making.

## References

[1] Martin Längkvist, Lars Karlsson, and Amy Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, 2014.

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

[3] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*, 2017.

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

[5] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

[6] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027, 2016.

[7] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

[8] José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869, 2015.

[9] Christos Louizos and Max Welling. Multiplicative normalizing flows for variational bayesian neural networks. *arXiv preprint arXiv:1703.01961*, 2017.

[10] Theofanis Karaletsos, Peter Dayan, and Zoubin Ghahramani. Probabilistic meta-representations of neural networks. *arXiv preprint arXiv:1810.00555*, 2018.

[11] Yingzhen Li and Yarin Gal. Dropout inference in bayesian neural networks with alpha-divergences. *arXiv preprint arXiv:1703.02914*, 2017.

[12] Lingxue Zhu and Nikolay Laptev. Deep and confident prediction for time series at uber. In *Data Mining Workshops (ICDMW), 2017 IEEE International Conference on*, pages 103–110. IEEE, 2017.

[13] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances In Neural Information Processing Systems*, pages 2378–2386, 2016.

[14] Qiang Liu. Stein variational gradient descent as gradient flow. In *Advances in neural information processing systems*, pages 3115–3123, 2017.

# Appendix

**Algorithm 1** BNN via SVGD

```
Input: data $\mathcal{D}$, step size $e$, target joint likelihood $p(\mathcal{D}, \boldsymbol{\theta})$
Initialize: n neural networks
for each batch do
    feed forward to compute $\log p(\mathcal{D}, \boldsymbol{\theta}_i)$ for $i = 1, \ldots, n$.
    backpropagate to calculate $\nabla_{\boldsymbol{\theta}_i} \log p(\mathcal{D}, \boldsymbol{\theta}_i)$ for $i = 1, \ldots, n$.
    update $\boldsymbol{\theta}_i$ for $i = 1, \ldots, n$: $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i + \frac{e}{n} \sum_{j=1}^{n} [k(\boldsymbol{\theta}_j, \boldsymbol{\theta}_i) \nabla_{\boldsymbol{\theta}_j} \log p(\mathcal{D}, \boldsymbol{\theta}_j) + \nabla_{\boldsymbol{\theta}_j} k(\boldsymbol{\theta}_j, \boldsymbol{\theta}_i)]$
end for
```