Class Project STAT 618: Bayesian Neural Network for Time-Series Forecasting

Billy Hernawan
Civil and Environmental Engineering Department
TAMU

Abstract

Deterministic time-series models lacks any form of uncertainty quantification and assumes that there is no noise in the observed data used to train these models. Bayesian Neural Network (BNN) provides uncertainty quantification through Bayesian inference and leverages the powerful Feedforward Neural Network (FNN) to make multi-step-ahead prediction in a time-series data. In this study, an implementation of the algorithm that features a singular Markov Chain Monte Carlo (MCMC) sampler with Langevin-gradient proposal was completed in Julia and the model was trained on a air quality time-series data. Preliminary convergence analysis on the model parameters was conducted and prediction over the next 100 days was assessed.

1 Background

The use of deep Neural Networks (e.g., Long Short-Term Memory (LSTM) & Gated Recurrent Unit (GRU)) in time-series forecasting have shown promising results [1] [2]. Nonetheless, conventional Neural Networks suffer from its inability to quantify uncertainty in model prediction. Bayesian paradigm offers a natural approach to quantify uncertainty in the parameters of the models and thus, the prediction. Most Bayesian inference on parameters of any forward model (e.g., Neural Networks & Finite Element Method) leverages Gaussian random-walk MCMC approach. However, chain convergence to a stationary distribution (i.e., posterior distribution $\pi(x)$) may be difficult to retrieve as high-dimensional parameter spaces often lead to slow convergence and poor mixing. Many previous studies [3] [4] have summarized and reported the difficulty in convergence analysis despite the many previously proposed methods. To circumvent this limitation, gradient-based Monte Carlo methods [5] have been proposed to accelerate convergence in a high-dimensional problem. This study adopts previous works by Chandra et al. [6] [7] where the authors implemented a BNN using parallel tempering with Langevin-gradient Monte Carlo to predict highly volatile stock price time-series.

2 Motivation

Due to the inherent capability of Bayesian paradigm in quantifying uncertainty in model's prediction, this study implemented the algorithm outlined in Chandra & He [6] to predict a seasonal-oriented air quality time series data set (i.e., time-series data that shows similar periodical behavior). More over, Chandra & He's [6] original implementation was done in Python, the programming language of choice within the machine learning community, and this study translated and implemented the algorithm in Julia. The most compelling reason to a Julia implementation is the faster computational speed compared to Python. This is mainly because Julia is a just-in-time (JIT) compiled language similar to C++ while Python is a interpreted language with a Global Interpreter Lock (GIL) that limits code operation to a single thread. The objective of this study is,

(a) to assess the prediction on a seasonal time-series using the proposed algorithm,
(b) to investigate if a single sampler using the proposed sampling technique can retrieve multi-modal parameter distribution in high-dimensional problem.

3 Methodology

3.1 Bayesian Neural Network

A Neural Network (NN) can be defined as,

$$f : \mathbb{R}^p \to \mathbb{R}^m$$

For the submission of Final Project for STAT 618 Spring 2023.

where $\mathbb{R}^p$ is the feature vector and $\mathbb{R}^m$ is the response vector. In the original implementation of the algorithm, Chandra & He [6] applied the proposed algorithm to a highly-volatile stock market data set with no observed seasonality. In order to illustrate the working mechanism of the proposed algorithm, the authors deemed appropriate to use a fully-connected dense NN for the highly-volatile stock market data set presented in Chandra & He [6]. A Bayesian Neural Network is a probabilistic approach to handle noise in the observed data by finding an appropriate statistical model that can describe the approximate distribution of the BNN parameters (i.e., weights and biases) [8]. Figure 1 illustrates one of the weights distribution given a set of training data.

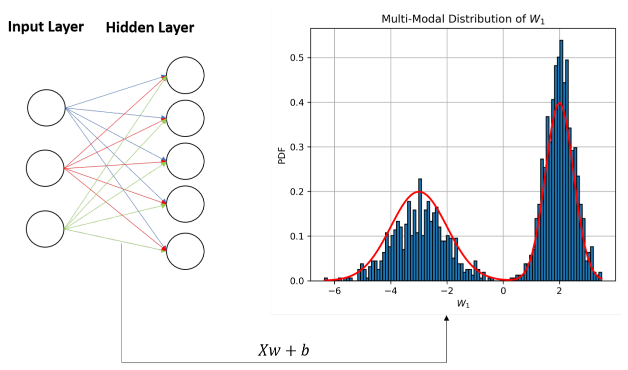

Figure 1: Posterior distribution of one of the weights (i.e., parameters of NN) after training through Bayesian inference. The proposed method can also learn multi-modal distribution

In this study, the likelihood function is defined as,

$$p(\boldsymbol{y}_s|\boldsymbol{\theta}) = \frac{-1}{(2\pi\tau^2)^{S/2}} \exp\left(\frac{-1}{2\tau^2}\sum_{t\in S}(\boldsymbol{y}_t - f(\bar{\boldsymbol{x}}_t))^2\right)$$

and a prior function of,

$$p(\boldsymbol{\theta}) \propto \frac{1}{(2\pi\sigma^2)^{L/2}} \times \exp\left(\frac{-1}{2\sigma^2}\left(\sum_{t\in S}^{M}\theta\right)\right) \times \tau^{2(1+v_1)} \exp\left(\frac{-v_2}{\tau^2}\right)$$

where τ is noise as model's parameter, S is training instances, v_1 & v_2 are user-defined parameters and M is the number of network parameters.

3.2 Langevin-Gradient Monte Carlo

Langevin-Gradient Monte Carlo (LGMC) features algorithm to generate samples from a probability distribution of interest ($\pi(x)$) by simulating the Langevin equation which is a continuous-time Markov process that satisfies the following stochastic differential equation (SDE):

$$dX_t = -\nabla V(x)dt + \sqrt{2}dB_t$$

where $-\nabla V(x)dt$ presents a drift term and dB_t denotes time derivative of standard Brownian motion [9]. In other words, LGMC seeks to move towards the target distribution (i.e., $\pi(x)$) by using gradient but with induced stochasticity (e.g, Stochastic Gradient Descent with Noise). Compared to the traditional Gaussian random-walk approach and the more popular gradient-based Hamiltonian Monte Carlo (HMC), LGMC features a hybrid of random-walk and gradient updating to sample parameters in the proposal space.

Chandra & He [6] implemented the LGMC hydrid approach by featuring a one-step gradient over Gaussian noise. The implementation is highlighted below,

Proposal sample:

$$\theta_r^p \sim q(\theta|\theta_r^k)$$

where the superscript k corresponds to the previous sampled parameters, the subscript r corresponds to the number of replica sampler (see section 3.3) and the superscript p corresponds to the proposed samples at current step.

The one-step gradient with Gaussian noise is implemented as following:

$$\boldsymbol{\theta}^p \sim \mathcal{N}(\bar{\boldsymbol{\theta}}^k, \Sigma_\theta)$$
$$\bar{\boldsymbol{\theta}}^k = \bar{\boldsymbol{\theta}}^k + r\nabla E(\theta^k)$$
$$E = \sum_{i\in S}(y_i - f(\bar{x}_i))^2$$
$$\nabla E(\boldsymbol{\theta}^k) = \left(\frac{\partial E}{\partial\theta_1}, \frac{\partial E}{\partial\theta_2}, ..., \frac{\partial E}{\partial\theta_L}\right)$$

3.3 Parallel Tempering

Parallel tempering features multiple/replica sampler that can be implemented in a parallel computing environment and is homogenous to ensemble learning technique. One of the main advantages of using LGMC with replica sampler is its capability to sample multi-modal distribution of parameters, particularly for a high-dimensional problem and non-linear forward model. This is achieved by assigning a temperature (i.e., analogous to weights) on each replica sampler that has a value between 0 and 1. Higher temperature has higher probability to accept weaker proposals and vice-versa. A collection of samples from R replicas can be defined as,

$$\boldsymbol{\Theta} = (\theta^1, ..., \theta^R)$$

with assigned temperature,

$$T \in [0, 1]$$

and posterior distribution retrieved by each replica sampler as,

$$p(\theta|\boldsymbol{D})^T$$

Lower temperature level guides the posterior distribution to be closer to a uniform stationary distribution that is analogous to a global exploration strategy of the parameter space while larger temperature level leads to the true posterior distribution and is analogous to a local exploration strategy. An illustration of multiple replica samplers with various temperature is shown in Figure 2.

After a defined number of interval swap (i.e., a user-defined hyper-parameter), replica swap is conducted across the replicas and is determined by Metropolis-Hasting acceptance criterion. However, due to time constraint, replica sampler was not implemented in the current Julia implementation.

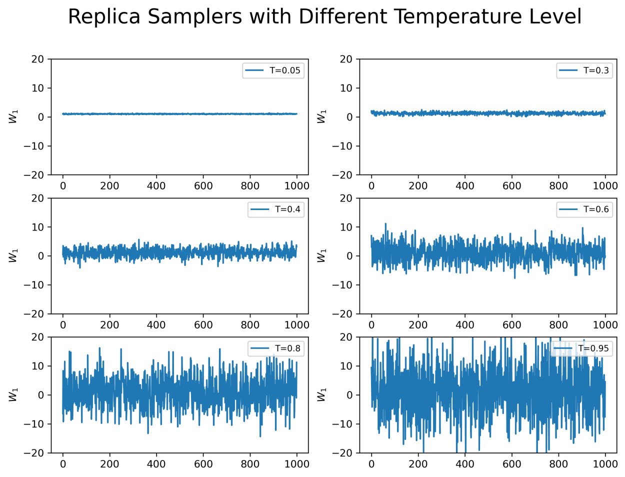

Figure 2: Illustration of an ensemble of replica samplers with various temperature level.

3.4 State-space Reconstruction of Time Series

The objective of the BNN is to make multi-step prediction $y \in \mathbb{R}^p$ based on previous input of $x \in \mathbb{R}^m$. Generally, $p \neq m$ but in the current implementation $p = m$ (i.e., to predict 5 time-step ahead, the model needs 5 known values of previous time-steps). Figure 3 depicts how a Neural Network takes the input data and predicts the output.

Given a specific length of a time-series (N), an embedding dimension (m) and a time-lag (T) can be specified to reconstruct time series as input-output vector. For the very first instance,

$$\bar{\boldsymbol{x}}_1 = [x_1, x_2, x_3, ..., x_m]$$

$$\boldsymbol{y}_1 = [x_{m+1}, x_{m+2}, x_{m+3}, ..., x_{m+n}]$$

Naturally, for all the remaining instance as the window is moved across the entire time-series,

$$\bar{\boldsymbol{x}}_t = [x_{1+(t-1)T}, x_{2+(t-1)T}, x_{3+(t-1)T}, ..., x_{m+(t-1)T}]$$

$$\boldsymbol{y}_t = [x_{m+1+(t-1)T}, x_{m+2+(t-1)T}, x_{m+3+(t-1)T}, ..., x_{m+n+(t-1)T}]$$

In the current implementation, the time-lag was chosen to be 1 (i.e., predict consecutive time-state) and the embedding dimension and desired output control the size of the input and output layer.

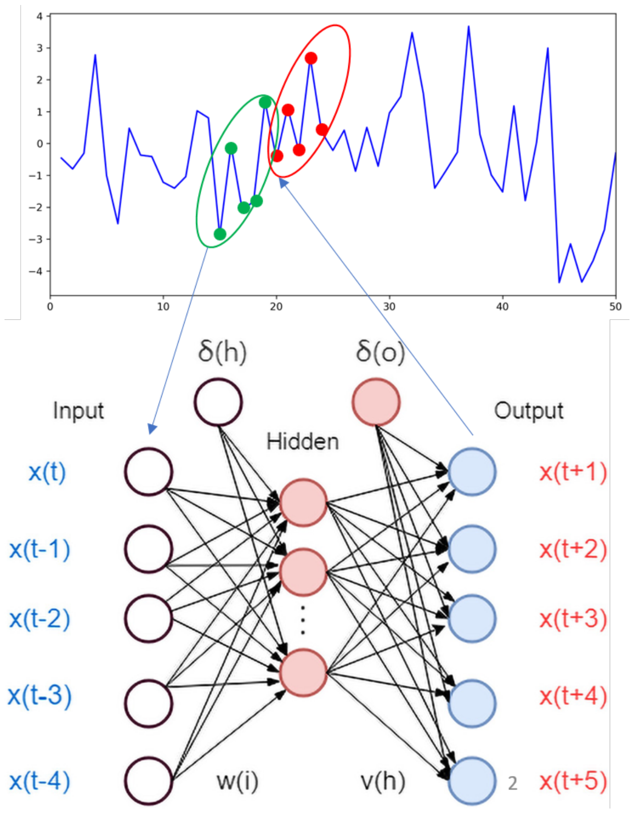

Figure 3: Illustration of Neural Network taking a vector of size 5 and outputing a vector of size 5.

4 Implementation

4.1 Data Set

For the seasonal time-series data set that is studied in this article is the 'Air Quality in Madrid (2001-2018)'. This data set was selected because it contained clear observed seasonality from the year 2001 to 2018. The data set contains various gasses and particulate matter reading from various weather stations across Madrid, Spain. The data was aggregated to give a representative daily value and create a time series of various gasses. The data is plotted in Figure 4.

The data set contains 217,871 rows with readings of SO_2, CO, NO, NO_2, $PM25$, O_3 and many others. The training and validation data set ranges from 2001-01-01 to 2014-08-27 and 2014-08-28 to 2017-05-20 (around 70% and 20% split) and the test data set ranges from 2017-05-21 to 2018-05-01. Although the proposed model is to specifically work on a uni-variate time-series, the Bayesian inference methodology can be implemented to a multi-variate time-series by altering the model architecture to take input of multiple vectors. Figure 5 shows the correlation structure between the features of the data set.

4.2 Implementation

A single-sampler based on LGMC was implemented in Julia using a fully-connected dense network with a topology of $[5, 10, 5]$ (i.e., 1 hidden layer of the size 10) and 300,000 samples were taken during the simulation.

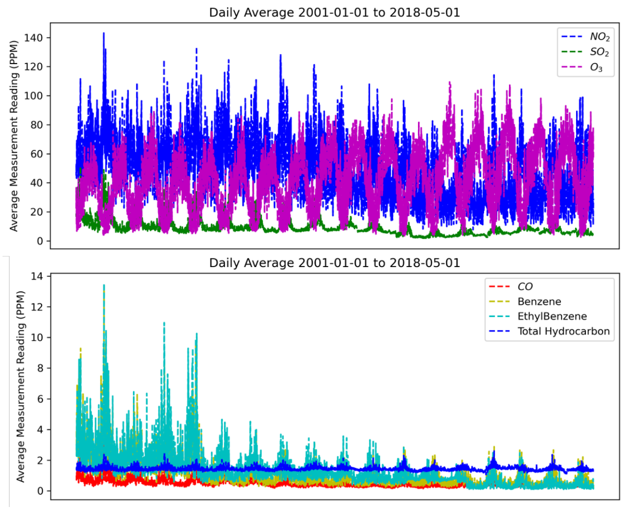

Figure 4: Daily Average of Various Gasses from 2001 to 2018 in Madrid, Spain.

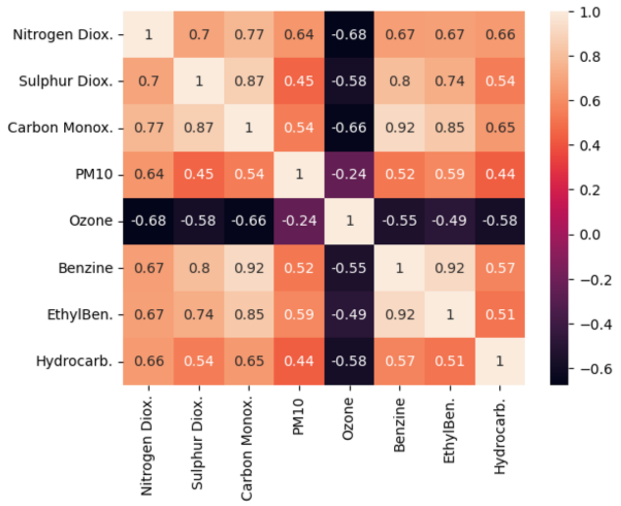

Figure 5: Correlation Matrix of Some of the Features in the Data Set.

The train and validation data set are of the shape $x_{train} \in \mathbb{R}^{4991x10}$ and $x_{validate} \in \mathbb{R}^{991x10}$ and there are a total of 115 parameters that the BNN requires (e.g., $w_1 \in \mathbb{R}^{5x10}$, $b_1 \in \mathbb{R}^{1}0$, $w_2 \in \mathbb{R}^{10x5}$, $b_2 \in \mathbb{R}^{5}$). Sigmoid activation function was selected for the current implementation and the wall-clock train time was 4.5 hours trained on Intel(R) Core(TM) i5-10500. The code implementation can be found here: `https://github.com/hernawa2/bayesian_neural_network`.

5 Result

The cumulative mean and cumulative standard deviation is shown in Figure 6. Upon initial observation, it seems that all the 115 parameters reached convergence (i.e., reach stationary distribution) as its cumulative mean and standard deviation are asymptotic. Nonetheless, for high-dimensional problem, additional diagnostics [2] on convergence should be conducted. The burn-in was determined to be after 250,000 samples and the posterior distributions were approximated from the remaining 50,000 samples.

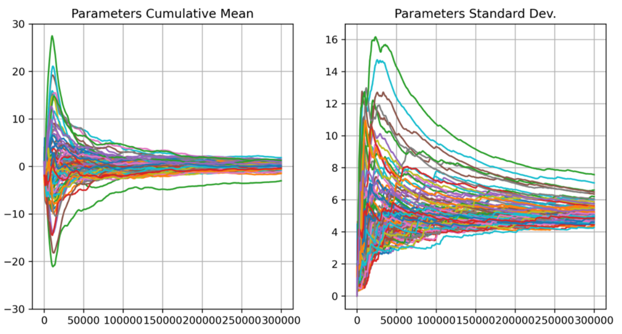

Figure 6: Cumulative Mean and Standard Deviation of All of 115 Parameters.

The posterior distribution of randomly-picked parameters is illustrated in Figure 7. Initial inspection suggests that a singular sampler was able to sample multi-modal distribution. However, further testings and simulations are needed to draw concrete conclusion on the ability of a singular LGMC to sample multi-modal distribution.

The Bayesian point estimates in the posterior distribution of each parameters were retrieved and used to retrieve mean model prediction on NO_2 highlighted by the red line in Figure 8 for the next 100 days after the train & validation data set. Multiple realizations (i.e., samples the parameters and pass it to the FNN multiple times to create a band of prediction) from the posterior distribution of parameters were created. The model realizations that had the biggest average distance from the actual NO_2 reading were determined to be the uncertainty in the model prediction portrayed by the green line in Figure 8.

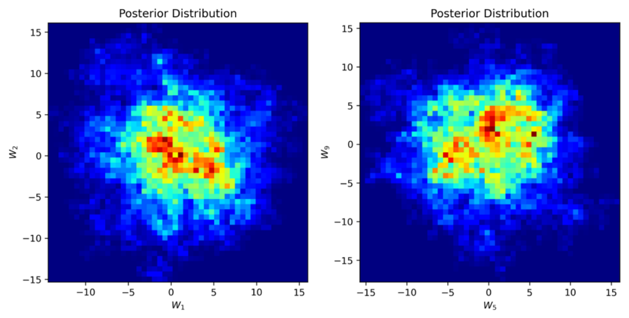

Figure 7: Posterior Distribution of Some of the Calibrated Weights.

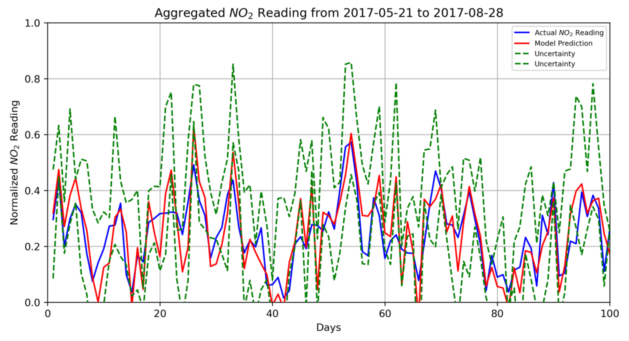

Figure 8: Model Prediction on NO_2 for the Next 100 Days Past the Training & Validation Data.

Another prediction on SO_2 using a specific model trained on SO_2 time-series data set was also attempted. Figure 9 shows the actual reading and the model prediction.

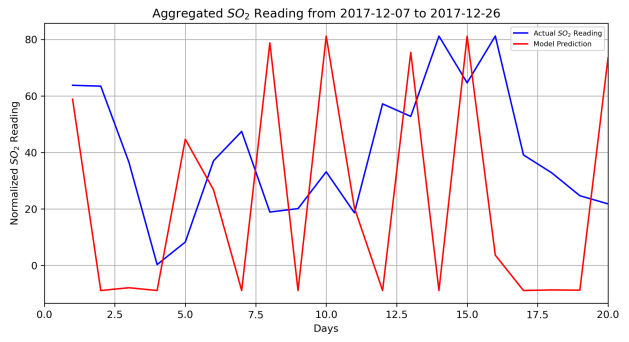

Figure 9: Model Prediction on SO_2 for the Next 20 Days Past the Training & Validation Data.

6 Conclusion

The modeling approach described above can be useful for non-seasonal time-series forecasting as illustrated in Chandra & He [6]. Nonetheless, it is important to ensure that the seasonal information is represented in the model to retrieve better prediction. In the current approach, a fully-dense connected network does not have the capability to retain the seasonal information. One way to improve the modeling approach is to leverage other popular specialized time-series NN such as RNN with LSTM for historical information retention and utilize the proposed sampling technique to quantify uncertainty.

References

[1] K. Sako, B.N. Mpinda, P.C. Rodrigues (2022). Neural Networks for Financial Time Series Forecasting. *Entropy* **24**(5):657.

[2] X. Song, Y. Liu, L. Xue, J. Wang, J. Zhang, J. Wang, L. Jiang, Z. Cheng (2020). Time-Series Well Performance Prediction based on Long Short-Term Memory (LSTM) Neural Network Model. *Journal of Petroleum Science and Engineering* **186**:106682.

[3] M.K. Song and B. Carlin (1996). Markov Chain Monte Carlo Convergence Diagnostics: A Comparative Review. *Journal of the American Statistical Association* **91**434:883-904.

[4] V. Roy (2020). Convergence Diagnostics for Markov Chain Monte Carlo. *Annual Review of Statistics and Its Application* **7**:387-412.

[5] M. Girolami and B. Calderhead (2011). Riemann Manifold Langevin and Hamiltonian Monte Carlo Methods. *Journal of the Royal Statistical Society Series B: Statistical Methodology* **73**2:123-214.

[6] R. Chandra and Y. He (2021). Bayesian Neural Networks for Stock Price Forecasting before and during COVID-19 Pandemic. *PLoS ONE* **16**(7):e0253217.

[7] R. Chandra, K. Jain, R.V. Deo and S. Cripps (2019). Langevin-Gradient Parallel Tempering for Bayesian Neural Learning. *Neurocomputing* **359**:315–326.

[8] L.V. Jospin, H. Laga, F. Boussaid, W. Buntine and M. Bennamoun (2022). Hands-On Bayesian Neural Networks - A Tutorial for Deep Learning Users. *IEEE Computational Intelligence Magazine* **17**2:29–48.

[9] F. Luan, S. Zhao, K. Bala and I. Gkioulekas (2020). Langevin Monte Carlo Rendering with Gradient-based Adaptation. *ACM Trans. Graph.* **39**4:140.

[10] Decide Soluciones. Air Quality in Madrid (2001-2018). URL: `https://www.kaggle.com/datasets/decide-soluciones/air-quality-madrid`.