Supporting Information for "Exploring Bayesian deep learning for weather forecasting with the Lorenz 84 system"

Yang Liu[1,2], Jisk Attema[1], Wilco Hazeleger[3]

[1]Netherlands eScience Center, Amsterdam, the Netherlands

[2]Meteorology and Air Quality Group, Wageningen University, Wageningen, the Netherlands

[3]Faculty of Geosciences, Utrecht University, Utrecht, the Netherlands

Contents of this file

1. Text S1 to S5
2. Figures S1 to S2

Introduction This supplementary material includes additional information regarding BDL and the variational inference method for training the Bayesian neural networks, namely the Bayes by Backprop. It also includes a brief introduction of the local re-parameterization trick and the evaluation metrics. A detailed explanation of the lead time dependent forecasts and numerical configurations of training and forecasting with BayesLSTM is provided.

S1. BDL and Bayes by Backprop In this section, we elaborate on BDL and Bayes by Backprop in detail. Deep neural networks are equipped with multiple layers of fixed weights to extract the multiple levels of abstraction in the training data (LeCun et al., 2015). These neural networks are deterministic and prone to overfitting and forecasts with over-confidence (Blundell et al., 2015). In order to incorporate uncertainty estimation during training and forecasting, we need to transform the deterministic neural network into a probabilistic structure and therefore we seek help from the Bayes' theorem.

To begin with, we have a training set $D = (x_i, y_i)_i$ which consists of input data $x \in \mathbb{R}^d$ and output data $y \in \mathbb{R}^d$. We aim for a probabilistic neural network $p(y|x, w)$ with a distributed weight w. To obtain the weight distribution, we apply the Bayesian inference and we will get the expression:

$$p(w|x, y) = \frac{p(y|x, w)p(w)}{p(x, y)} \tag{1}$$

By substituting $p(x, y)$ according to the law of total probability, we can obtain:

$$p(w|x, y) = \frac{p(y|x, w)p(w)}{\int p(y|x, w)p(w)dw} \tag{2}$$

The integral in the denominator requires coverage of all possible values of w, which is computationally prohibitive. This makes the true posterior probability distribution intractable (Blundell et al., 2015; Shridhar et al., 2018, 2019). To solve this, various approximation methods (e.g. maximum-a-posteriori scheme, variational inference schemes) were studied in the past (Graves, 2011; Shridhar et al., 2018). Among all of them, variational inference schemes are widely embraced and they work well for a wide range of BDL applications.

So far, three popular variational inference schemes are always used to approximate the intractable posterior: Monte Carlo Markov Chain (MCMC), Monte Carlo dropout (MCD) and Bayes by Backprop (BBB).

MCMC is a family of methods that combines stochastic variational inference and Monte Carlo approaches (Salimans et al., 2015). Unlike variational inference in general which takes a random draw from a simple variational distribution and keeps optimizing the distribution, MCMC methods subsequently apply a stochastic transition operator to the random draw, so as to cover the exact posterior distribution. As long as the iterations are sufficient, MCMC can approximate the exact posterior well. However, we do not know if the iterations are sufficient and this procedure is always very costly.

An alternative is MCD, which uses dropout to perform variational inference where the variational distribution comes from a Bernoulli distribution. Hence it is also known as Variational inference with Bernoulli Distribution (Gal & Ghahramani, 2015, 2016). This method makes use of dropout in each layer of neural network during training as well as testing and thus the process is equivalent to sampling from a Bernoulli distribution and provides a measure of uncertainty. Nevertheless, it provides a rough approximation of the target posterior and the control of uncertainty is limited by the complexity of the neural network.

Given the drawback of MCMC and MCD, in this study, we choose a more functional and affordable approach, BBB, to approximate the posterior. This method is a back-propagation (BP) compatible variational inference approach that estimates a density function with a known distribution and progressively update it with BP (Blundell et

al., 2015; Shridhar et al., 2018). More details about BBB can be found in the section Methodology in the main body of the paper.

S2. Local re-parameterization method With the implementation of BBB and optimization function, it seems our BayesLSTM is ready for training. However, the whole procedure is not ready for back-propagation (BP) due to the stochastic node including $\mathcal{N}(\theta|\mu, \sigma^2)$ in our chosen variational distribution. In order to enable the optimization of the parameters of Gaussian posterior with BP, we introduce the local reparameterization method, which translates the global uncertainty in the weights into a form of local uncertainty (Kingma et al., 2015; Shridhar et al., 2019). In this case, we replace $\mathcal{N}(\theta|\mu, \sigma^2)$ with $w = \mu + \sigma * \epsilon$ where $\epsilon \sim \mathcal{N}(0, 1)$. More information about the local reparameterization method can be found in the related literature (Kingma & Welling, 2013; Kingma et al., 2015; Shridhar et al., 2018, 2019).

S3. Evaluation metrics In order to evaluate these ensemble forecasts, we include continuous ranked probability score (CRPS), root mean square error (RMSE) and Euclidean distance.

CRPS is a popular verification tool for the assessment of probabilistic forecast systems. It is used to evaluate an ensemble forecast against a single deterministic observation and it has the following form (Gneiting et al., 2005):

$$CRPS(F, v) = \int_{-\inf}^{\inf} [F(r) - H(r - v)]^2 dr \tag{3}$$

with F the predictive cumulative density function (CDF), v the verifying observation, r the threshold value, and $H(r-v)$ the Heaviside function which takes the value 0 when $r < v$ and otherwise 1.

RMSE score is also included in this study and it is defined in the following way so as to work with the probabilistic forecast results:

$$RMSE = \frac{1}{N}\Sigma_{n=1}^{N}\frac{1}{T}\Sigma_{t=1}^{T}\sqrt{(x_{pred} - x_{obs})} \tag{4}$$

with N the number of ensemble members, T the number of time steps, x_{pred} and x_{obs} the predicted and observed value.

Euclidean distance (EuD) is used to evaluate the similarity between trajectories, which is expressed as:

$$EuD = \frac{1}{N}\Sigma_{N=1}^{N}\frac{1}{T}\Sigma_{T=1}^{T}\sqrt{(x_{pred} - x_{obs})^2 + (y_{pred} - y_{obs})^2 + (z_{pred} - z_{obs})^2} \tag{5}$$

S4. Estimate uncertainty with BDL Typically, three types of uncertainties are considered: (1) Uncertainties in the initial conditions (2) Necessary approximations and corresponding uncertainties in the construction of a numerical model of the real atmosphere (3) Uncertainties posed by external forcing and boundaries. The last one is often ignored in an operational weather forecast system, but relevant for climate studies and local forecasts. Kendall and Gal (2017) explained how the model uncertainty and initial condition uncertainty are addressed by BDL respectively. Given the nature of the forward process of BDL, model uncertainty is addressed by the BDL via placing a prior distribution over a

model's weight, which is already explained by the prior dependent part $KL[q(w|\theta)||p(w)]$ during the training of BDL when minimizing the KL divergence (Kendall & Gal, 2017).

The representation of initial condition uncertainty with BDL is less straightforward. During training, the BayesLSTMs approach output $y_j \in \mathbb{R}^d$ with an input $x_j \in \mathbb{R}^d$ and weight distribution w_j, as $P(y_j|x_j, w_j)$. However, there is always noise ϵ_j in the input fields (initial conditions) and this propagates through the network and affects the output, thus $P(y_j|x_j, w_j, \epsilon_j)$. With the help of Bayes rule, we bring in the pre-knowledge of model weight distribution as prior and search for the best weight distribution as posterior that is able to achieve the testing set $\hat{x_j}$ and $\hat{y_j}$ following $P(\hat{y_j}|\hat{x_j}) = \mathbb{E}_{P(w_j|D)}[P(\hat{y_j}|\hat{x_j}, w_j)]$. To simplify the problem, we assume a Gaussian distribution for the prior and the weight distribution then can be solved via maximum a posteriori (MAP). Due to the use of the BBB approach, we need to solve the KL divergence and the knowledge of observation is brought in via the data-dependent term $\mathbb{E}_{q(w|\theta)}[log\ p(D|w)]$ in equation 3 in the paper, and finally the $-log\ p(D|w^{(j)})$ in equation 4. It then can be reformulated as:

$$\mathbb{L}(\theta, D) = -\frac{1}{N}\sum_{j=1}^{N} log\ p(y_j|f^{\hat{w_j}}(x_j, \epsilon_j)) \tag{6}$$

If we assume the error in the output is Gaussian, then $-logp(y_j|f^{\hat{w_j}}(x_j, \epsilon_j)) \propto \frac{1}{2\sigma^2}||y_j - f^{\hat{w_j}}(x_j)||^2 + \frac{1}{2}log\sigma^2$. The shape of this Gaussian probability is determined by the variance σ, which is in this case the deviation between the ground truth in reality and the uncertainty corresponding to the noise in the input fields ϵ_j. As a result, the data dependent loss evolves as:

$$\mathbb{L}_{NN}(\theta, D) = \frac{1}{N}\sum_{j=1}^{N}\frac{1}{2\sigma(\epsilon_j)^2}||y_j - f^{\hat{w}_j}(x_j)||^2 + \frac{1}{2}log\ \sigma(\epsilon_j)^2 \quad (7)$$

with $\sigma(\epsilon_i)$ indicating the variance as a function of noise in the input fields, which represents the uncertainty in the initial conditions. In other words, uncertainty in the initial condition is modeled through the neural networks by generating a distribution over the output of the model. Note that this is recognized as heteroscedastic and aleatoric uncertainty in machine learning and it does not exist in a non-Bayesian (deterministic) neural network since this observation noise parameter ϵ is fixed as part of the model's weight decay and therefore neglected after training (Kendall & Gal, 2017). So far, our analysis is based on the hypothesis of linear propagation of expectation with variational inference in the LSTM (Fortunato et al., 2017). However, it should be noted that this becomes different when forecasting more than one step ahead since there is also uncertainty coming from the forecasts of previous time steps.

S5. Lead time dependent forecast and numerical configurations The whole numerical processes of training the BayesLSTM and making lead time dependent forecasts with BayesLSTM are described in details in this section. We generate time sequences of variable X, Y and Z with modified Lorenz 84 model, which including 1500 time steps in total. We perform sequence-to-one forecast with BayesLSTM and the first 1300 time steps are used for training. During training, in each epoch we pass 3 x 1300 points to the BayesLSTM model. The model samples its weight distribution at each time step (with 3 points X, Y and Z passed to the model) and after 1300 iterations we perform back-propagation to update the BayesLSTM model. The procedure will be repeated until

the maximum epochs are reached (or the early stop module is called, depending on the setup).

The validation set consists of 200 time steps. The forecasting procedure is the same as the training process, except that there is no back-propagation and we perform lead time dependent forecasts (Liu et al., 2020). At each forecast time step t_n, we use the Lorenz model output (observation) to initialize the model and make forecasts, which can be described by the equation below:

$$(X, Y, Z)_{pred[t_n]} = BayesLSTM((X, Y, Z)_{obs[t_0,t_1,t_2,...,t_{n-1}]}) \tag{8}$$

with $(X, Y, Z)_{pred[t_n]}$ denotes the forecast X, Y and Z at time step t_n, and $(X, Y, Z)_{obs[t_m]}$ represents the Lorenz model output of X, Y and Z at time step t_m with $m < n$. It is still sequence-to-one prediction, which means data at time step t_0 to t_{n-1} will be passed to the BayesLSTM model to make the forecast for the time step t_n.

For one more lead time step t_{n+1}, the forecast is based on both the model output from t_0 to t_{n-1} and the forecast of time step t_n, and therefore it can be expressed as:

$$(X, Y, Z)_{pred[t_{n+1}]} = BayesLSTM((X, Y, Z)_{obs[t_0,t_1,t_2,...,t_{n-1}]} + (X, Y, Z)_{pred[t_n]}) \tag{9}$$

It can be noticed that starting from the time step t_{n+1} (the second lead time step), the forecast quality also depends on the forecasts of previous time steps. This helps us explore how far the BayesLSTM can predict the Lorenz system, which corresponds to the actual predictability of a weather and climate system.

References

Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015). Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*.

Fortunato, M., Blundell, C., & Vinyals, O. (2017). Bayesian recurrent neural networks. *arXiv preprint arXiv:1704.02798*.

Gal, Y., & Ghahramani, Z. (2015). Bayesian convolutional neural networks with bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158*.

Gal, Y., & Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning* (pp. 1050–1059).

Gneiting, T., Raftery, A. E., Westveld III, A. H., & Goldman, T. (2005). Calibrated probabilistic forecasting using ensemble model output statistics and minimum crps estimation. *Monthly Weather Review*, *133*(5), 1098–1118.

Graves, A. (2011). Practical variational inference for neural networks. In *Advances in neural information processing systems* (pp. 2348–2356).

Kendall, A., & Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems* (pp. 5574–5584).

Kingma, D. P., Salimans, T., & Welling, M. (2015). Variational dropout and the local reparameterization trick. In *Advances in neural information processing systems* (pp. 2575–2583).

Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint*

arXiv:1312.6114.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436–444.

Liu, Y., Bogaardt, L., Attema, J., & Hazeleger, W. (2020). Extended range arctic sea ice forecast with convolutional long-short term memory networks. *Monthly Weather Review*. (under review)

Salimans, T., Kingma, D., & Welling, M. (2015). Markov chain monte carlo and variational inference: Bridging the gap. In *International conference on machine learning* (pp. 1218–1226).

Shridhar, K., Laumann, F., & Liwicki, M. (2018). Uncertainty estimations by softplus normalization in bayesian convolutional neural networks with variational inference. *arXiv preprint arXiv:1806.05978*.

Shridhar, K., Laumann, F., & Liwicki, M. (2019). A comprehensive guide to bayesian convolutional neural network with variational inference. *arXiv preprint arXiv:1901.02731*.

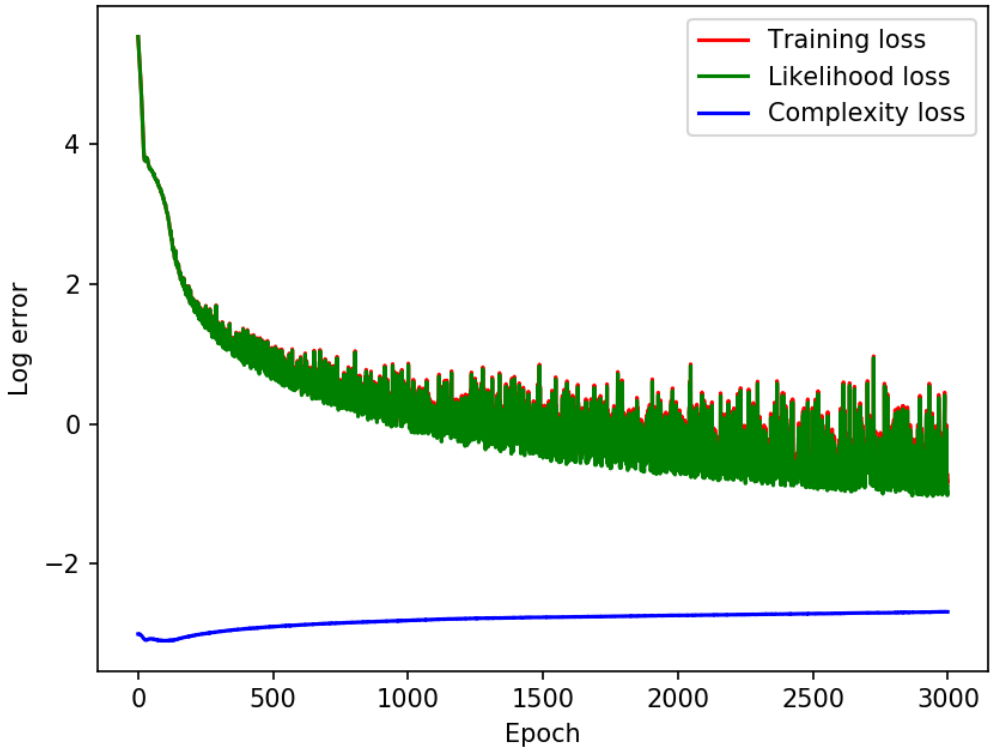

Figure S1. Training loss in the logarithmic form. The likelihood loss and complexity loss are included separately.

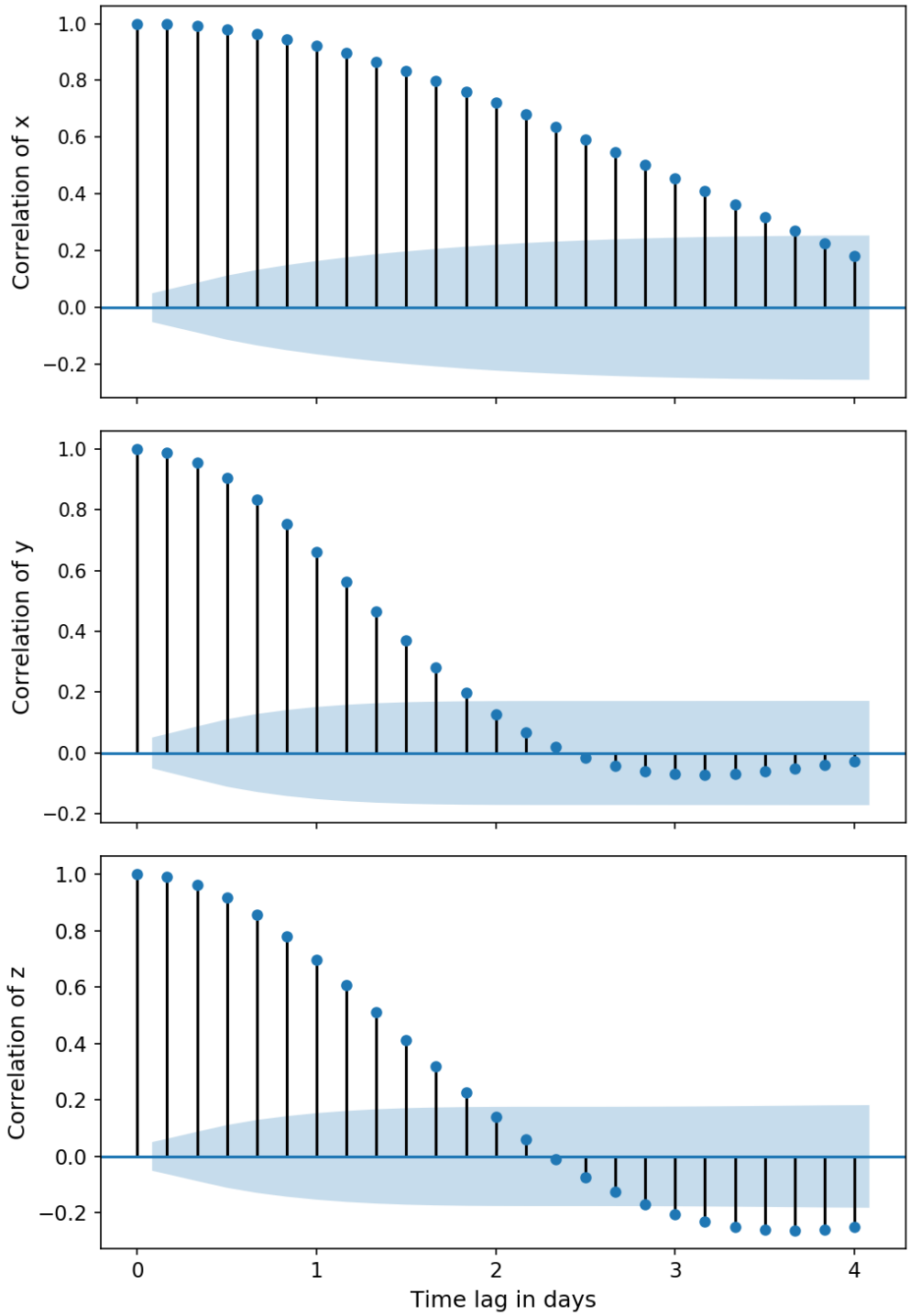

Figure S2. Autocorrelation of each variable of the Lorenz 84 model output. The blue shades indicate the p-value in null hypothesis significance testing.