entropy

MDPI

Article

Bayesian Recurrent Neural Network Models for Forecasting and Quantifying Uncertainty in Spatial-Temporal Data

Patrick L. McDermott [1,*] and Christopher K. Wikle [2]

[1] Jupiter Intelligence, Boulder, CO 80302, USA
[2] Department of Statistics, University of Missouri, Columbia, MO 65211, USA; wiklec@missouri.edu
* Correspondence: plmyt7@gmail.com

Received: 29 December 2018; Accepted: 12 February 2019; Published: 15 February 2019

Abstract: Recurrent neural networks (RNNs) are nonlinear dynamical models commonly used in the machine learning and dynamical systems literature to represent complex dynamical or sequential relationships between variables. Recently, as deep learning models have become more common, RNNs have been used to forecast increasingly complicated systems. Dynamical spatio-temporal processes represent a class of complex systems that can potentially benefit from these types of models. Although the RNN literature is expansive and highly developed, uncertainty quantification is often ignored. Even when considered, the uncertainty is generally quantified without the use of a rigorous framework, such as a fully Bayesian setting. Here we attempt to quantify uncertainty in a more formal framework while maintaining the forecast accuracy that makes these models appealing, by presenting a Bayesian RNN model for nonlinear spatio-temporal forecasting. Additionally, we make simple modifications to the basic RNN to help accommodate the unique nature of nonlinear spatio-temporal data. The proposed model is applied to a Lorenz simulation and two real-world nonlinear spatio-temporal forecasting applications.

Keywords: recurrent neural network; Bayesian machine learning; nonlinear dynamical models; long-lead forecasting; spatial-temporal

1. Introduction

Nonlinear and quasilinear spatio-temporal data can be found throughout the engineering, biological, geophysical and social sciences. Some examples of such processes include animal or robotic interactions, local economic forecasting, river flow forecasting, visual motion capture, and radar precipitation reflectivity nowcasting, to name a few. The nonlinearity in these systems makes forecasting and quantifying uncertainty difficult from both a modeling and computational perspective. While statistical forecasting of univariate nonlinear time-series processes is relatively well-developed [1,2], nonlinear multivariate systems have seen much less progress in statistics. Dynamical spatial-temporal models (DSTMs) are multivariate systems that have the added challenge of characterizing interactions between different scales of variability while simultaneously facing the curse-of-dimensionality that is exacerbated for nonlinear parametric spatio-temporal models, e.g., [3].

Some more recent nonlinear DSTMs in the statistical literature include threshold or regime switching models, e.g., [4,5], agent (individual)-based models, e.g., [6], general quadratic nonlinear (GQN) models [7], analog models [8], and mechanistic nonlinear models [9]. While such models have shown success for particular systems, more flexible models are often needed for highly nonlinear systems with complex latent relationships. Furthermore, with only a few exceptions, it can be quite difficult to explicitly specify the nonlinearities in these systems. One exception includes using

physically motivated models such as stochastic partial differential equations, e.g., [7,9,10], although this requires some *a priori* knowledge of the dynamics in the system. Due to the many challenges associated with modeling nonlinear spatio-temporal processes, much of the statistical development of these models has lagged behind other disciplines such as applied mathematics, dynamical systems, and machine learning.

One of the many appealing aspects of machine learning methods is their ability to extract salient features and relationships from complex high-dimensional data, particularly for forecasting and classification. Spatio-temporal processes are a strong candidate for machine learning methods due to the complex interactions and high-dimensionality that are ubiquitous in these processes. While there have been past attempts to apply machine learning methods, such as feed-forward neural networks, e.g., [11] and deep learning models, e.g., [12] to nonlinear spatial-temporal processes, the explicit accounting for dynamics in these processes has been less of a focus. Moreover, although feed-forward neural networks provide a convenient framework for modeling multivariate processes, they are not designed to explicitly capture time-sequential dynamical interactions between variables. As noted in the dynamical systems literature, explicitly modeling the dynamics is often paramount to successfully forecasting such systems. Recurrent neural networks (RNNs) represent a machine learning model with the potential to effectively model the nonlinear dynamics in multivariate sequential systems such as spatio-temporal processes.

First popularized in the 1980s, RNNs fell out of favor, in part, because of the so-called "vanishing gradient" problem that makes these models extremely difficult to estimate with back-propagation. More recently, as deep learning models have gained in popularity, solutions such as "long-short-term-memory" (LSTM) networks [13,14] have helped mitigate this vanishing gradient problem. As RNNs have become more manageable from an estimation perspective, they have increasingly been used to model complicated sequential forecasting problems such as visual object tracking [15], speech recognition [16], and text generation [17], just to name a few. Simultaneously, RNNs have also seen a rise in usage in the dynamic systems literature due to their ability to replicate complex attractor dynamics that are often present in chaotic systems [18]. Thus, RNNs provide a black-box method that can capture dynamical relationships for problems where it is either difficult to specify these relationships *a priori* or little information is available on the specific form of these relationships. Importantly, RNNs fill this void by providing a mechanism for capturing complex sequential relationships between variables, thus providing a modeling tool for a broad set of dynamical problems.

As RNNs have become more prevalent, a variant of the original RNN model, referred to as an *echo-state network* (ESN) [19] has become a staple in the dynamical systems literature for solving nonlinear forecasting problems. ESNs are extremely appealing because they retain much of the forecast accuracy of an RNN at a fraction of the computational cost. In essence, ESNs simulate randomly the parameters that make up the hidden states of a RNN (see below), thus reducing the problem to a traditional regression type problem. Although the methodology described here is more closely related to the RNN framework than the ESN, we do borrow and discuss ideas from the ESN literature to motivate choices pertaining to the proposed model. For a spatial-temporal example of an ESN model see [20]. Conversely, LSTMs represent a variant of the RNN that use a gated structure to control how fast or slow certain structures are remembered. While the presented methodology does not consider LSTMs, due to the similar issues of dependent parameters in both the LSTM and traditional RNN framework, many of the ideas presented here could be extended into the LSTM framework.

Despite the broad size and overall scope of the RNN literature, these models are almost always presented without considering uncertainty. The few attempts at quantifying uncertainty are generally presented in an *ad hoc* fashion, without a formal probability based framework. For example, recent work such as [21] use less rigorous methods such as dropout (see Section 2.3) to quantify uncertainty. Conversely, as previously discussed, the statistical literature on modeling of nonlinear dynamical spatio-temporal systems does consider uncertainty quantification but is not well-developed,

especially compared to its linear counterpart. We address both of these issues by proposing a Bayesian spatial-temporal RNN model in which the forecasting strength of a traditional RNN is preserved, while also producing comprehensive uncertainty measures. In particular, we introduce a RNN model within a fully Bayesian framework that accounts for uncertainty in both parameters and data in a rigorous fashion. Furthermore, the data generating process and associated error process (e.g., measurement error) are rarely accounted for in the RNN literature. Accounting for these errors is often critical in the spatio-temporal literature, see [10]. The proposed Bayesian framework allows one to take advantage of the forecasting ability of the RNN model while also rigorously accounting for both the data generating process and other error processes associated with the data.

The application of MCMC methods to neural networks dates back to the seminal work of [22]. Since the work of [22] there have been limited attempts to use Bayesian MCMC methods with other neural network type models (e.g., RNNs, LSTMs). While others have used Bayesian modeling within the RNN framework, e.g., [23–25], to our knowledge this is the first fully Bayesian RNN trained with traditional Markov Chain Monte Carlo (MCMC) methods. The additional transition parameters in an RNN (see Section 2.1) introduce an extra layer of dependence, and thus sampling difficulties, that require advanced sampling techniques beyond what is presented in previous works such as [22]. Additionally, the gradient based nature of many previous Bayesian sampling methods for traditional neural networks, e.g., [22], require additional modifications to deal with the "vanishing gradient" problem discussed above. To assist with this problem we introduce additional expansion parameters within a traditional MCMC framework to help break some of this dependence between parameters.

By using MCMC methods, the proposed model and algorithm can more accurately measure uncertainty compared to traditional optimization methods or variational Bayesian methods. Although Bayesian methods such as stochastic gradient MCMC (SG-MCMC) have recently shown promise as an estimation tool for high-dimensional RNNs, i.e., [25], these stochastic gradient algorithms typically require the partitioning of the data to create so-called mini-batches. Spatio-temporal models often involve explicit dependencies between data points in space and/or time along with hierarchical relationships. Therefore, it may be difficult or impossible to partition the data in this way (especially when considering spatial dependence), which may make such stochastic algorithms prohibitive for spatial-temporal problems that are considered within a rigorous statistical framework.

We introduce multiple extensions to the traditional RNN model at both the data and latent stage of the model, with the dual aim of facilitating estimation and improving the forecasting ability of the model. The proposed extensions incorporate mechanisms from both the ESN and dynamical systems literature. Furthermore, we regularize the parameters in the model by proposing priors that help mitigate over-parameterization, inspired by traditional ESN models. Similar to traditional RNNs, fitting an RNN within a fully Bayesian framework presents a multitude of computational issues. To assist with computation, we propose using dimension reduction to deal with high-dimensional spatial-temporal processes. In addition, within a MCMC paradigm, we borrow the idea of including expansion parameters in the model from the data augmentation literature, e.g., [26–28], to assist with sampling the highly dependent parameters that make up an RNN.

Section 2 describes the proposed Bayesian spatio-temporal RNN model, along with various modeling details. Next, Section 3 goes through the specifics of the MCMC algorithm developed to implement the model. In the beginning of Section 4 the choices made for the setup of the model are described in detail. Section 4 continues with a simulated multiscale Lorenz dynamical system example, followed by a long-lead sea surface temperature (SST) forecasting problem and a United States (U.S.) state-level unemployment rate application. Finally, we end with a concluding discussion on the approach, along with possible future extensions in Section 5.

2. Spatio-Temporal Recurrent Neural Network

2.1. Traditional Recurrent Neural Network

Suppose we are interested in the n_y-dimensional spatial-temporal response vector $\mathbf{Y}_t$ at time t with corresponding input vector $\mathbf{x}_t$ of dimension n_x, with one being the first element of $\mathbf{x}_t$ corresponding to an intercept term (or bias term). Then, the traditional RNN model for multivariate data, e.g., [29], is defined as follows for $t = 1, \ldots, T$:

$$\text{data stage:} \qquad \mathbf{Y}_t = g(\mathbf{V}\mathbf{h}_t), \tag{1}$$

$$\text{hidden stage:} \qquad \mathbf{h}_t = g_h(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t), \tag{2}$$

where $\mathbf{h}_t$ is a n_h-dimensional vector of hidden state variables, $\mathbf{W}$ is a square $n_h \times n_h$ weight matrix, $\mathbf{U}$ is a $n_h \times n_x$ weight matrix, and $\mathbf{V}$ is a $n_y \times n_h$ weight matrix. The function $g(\cdot)$ is an activation function that creates a mapping between the response and the hidden states, and $g_h(\cdot)$ denotes the activation function for the hidden layer. For a continuous response vector, $g(\cdot)$ is simply the identity function, although this setup can also handle categorical data by allowing $g(\cdot)$ to be the softmax function. Nonlinearity is induced in the RNN model through the form of $g_h(\cdot)$, which is typically defined to be the hyperbolic tangent function (as is assumed throughout this article).

The square weight matrix $\mathbf{W}$ in (2) can be thought of analogously to a transition matrix in a typical vector autoregressive (VAR) model. That is, $\mathbf{W}$ models the latent dynamic connections between the various hidden states. Thus, underlying nonlinear interactions between variables or locations can effectively be modeled within this framework through $\mathbf{W}$. Having a mechanism to capture these interactions is often vital when modeling nonlinear spatio-temporal processes, e.g., [3]. Critically, the hidden states extract and supply salient hidden dynamic features from the data. Ideally, the hidden states will represent a general set of dynamical patterns from the input data, thus allowing the $\mathbf{V}$ parameters to appropriately weight these patterns. While the RNN defined in (1) and (2) has shown success at forecasting a variety of different systems, the model lacks any explicit error terms, and thus, does not contain a mechanism to formally account for uncertainty in the data, model, or parameters.

2.2. Bayesian Spatio-Temporal Recurrent Neural Network

In this section we introduce the Bayesian spatio-temporal RNN, referred to hereafter as the BAST-RNN model. Borrowing the notation introduced in the previous section, the BAST-RNN model is defined as follows:

$$\text{data stage:} \qquad \mathbf{Y}_t = \boldsymbol{\mu} + \mathbf{V}_1\mathbf{h}_t + \mathbf{V}_2\mathbf{h}_t^2 + \boldsymbol{\epsilon}_t, \quad \boldsymbol{\epsilon}_t \sim \text{Gau}(\mathbf{0}, \mathbf{R}), \tag{3}$$

$$\text{hidden stage:} \qquad \mathbf{h}_t = g_h\left(\frac{\delta}{|\lambda_w|}\mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\widetilde{\mathbf{x}}_t\right), \tag{4}$$

where $\boldsymbol{\mu}$ is a n_y-dimensional spatial intercept vector, $\mathbf{V}_1$ and $\mathbf{V}_2$ are each $n_y \times n_h$ output weight matrices, and the initial hidden state is set such that $\mathbf{h}_0 \equiv \mathbf{0}$. Here, we assume that $\mathbf{R} \equiv \sigma_\epsilon^2\mathbf{I}$, but note that when necessary, additional temporal or spatial structure can be modeled through the covariance matrix $\mathbf{R}$ (such an additional structure is not needed for the applications presented here). The hidden state parameter, λ_w, represents the largest eigenvalue of the matrix $\mathbf{W}$ and δ is a scaling parameter with a Unif$(0, 1)$ prior. By dividing $\mathbf{W}$ by $|\lambda_w|$ and restricting δ, we ensure the spectral radius of $\mathbf{W}$ is at most one. When the spectral radius of $\mathbf{W}$ exceeds one, the model may exhibit unstable behavior [19], and restricting the spectral radius in this fashion is common in the ESN literature, since $\mathbf{W}$ is not estimated in the ESN model. We should note that this scaling does not guarantee stationarity. We find that including δ in the model provides extra flexibility while providing stability for the hidden states. It is important to note that given the parameters δ, $\mathbf{W}$,$\mathbf{U}$, the initial condition $\mathbf{h}_0$, and input vectors, $\widetilde{\mathbf{x}}_t$ (see below), the hidden states are known and thus, do not need to be directly estimated. A so-called "leaking-rate" parameter is often included in the ESN framework when defining (4). Similar to [20],

we did not find this additional parameter useful for spatio-temporal forecasting, and therefore omitted it from the BAST-RNN model specification.

Along with scaling $\mathbf{W}$, we also extend the traditional RNN model by allowing for additional nonlinearity in (3) through $\mathbf{h}_t^2 \equiv (h_{t,1}^2, \ldots, h_{t,n_h}^2)'$. By including a nonlinear mapping between the response and $\mathbf{h}_t$, the proposed model can capture more nonlinear behavior and accommodate more extreme responses, see [20]. This nonlinear transformation of the hidden states provides additional useful covariates (and thus patterns) for predicting the process of interest. It may also be useful to include higher order interactions between the $\mathbf{h}_t$'s, although such interactions are not helpful for the applications described below. The major difference between the BAST-RNN and an ESN model (for example, the similar ESN model presented in [20]) concerns the estimation of the hidden states that make up the BAST-RNN. Using both input and output information, within a Bayesian framework, to estimate these parameters, the BAST-RNN requires far fewer hidden states (i.e., n_h) than a typical ESN model.

We borrow the idea of embedding the input from the dynamical systems literature as introduced by [30], to define the input vector in (4) as:

$$\widetilde{\mathbf{x}}_t{}' = \{\mathbf{x}_t', \mathbf{x}_{t-\tilde{\tau}}', \ldots, \mathbf{x}_{t-m\tilde{\tau}}'\}', \tag{5}$$

where $\tilde{\tau}$ is usually referred to as the "embedding lag" and m the "embedding length", thus leading to a $(m+1)n_x + 1$ dimensional input vector (assuming the first element of $\widetilde{\mathbf{x}}_t{}'$ corresponds to an intercept term). By embedding the process of interest, the proposed model utilizes all of the recent trajectory of the system, opposed to a single instance in time. Other statistical nonlinear spatio-temporal forecasting methods, e.g., [8,20], have shown that embedding the process of interest can lead to more accurate forecasts and quantifiably better uncertainty measures.

2.3. BAST-RNN Prior Distributions

The presented BAST-RNN model is comprised of multiple high-dimensional parameter weight matrices, resulting in an over-parameterized model. This problem is not unique to the BAST-RNN, and is often a criticism of RNNs and feed-forward neural networks in general. Due to the prevalence of this over-parameterization problem, many solutions have been proposed in the machine learning literature. All of the applications presented below contain training datasets that would be considered small in terms of number of samples compared to traditional machine learning problems. Issues concerning over-fitting can be exacerbated when dealing with training sets that contain a small number of samples. It is not uncommon to have such examples in the spatio-temporal literature. Thus, properly addressing issues related to over-fitting is vital to the success of the proposed methodology.

More recently, a method known as dropout [31,32] has shown promise as a tool to deal with over-parameterized weight matrices, thereby helping to prevent over-fitting. In essence, dropout creates a type of "hard" regularization by removing entire hidden units (and therefore weight parameters) during training. Similarly, ESN models deal with over-parameterized weight matrices by randomly setting a large percentage of parameters in the weight matrices to zero and then drawing the remaining parameters from a bounded or constrained distribution, see [20]. These are just two of the many proposed solutions for regularizing the over-parameterized weight matrices that make up neural network models. For statistical models, addressing this problem is similarly vital to help prevent over-fitting. Therefore, we propose regularization priors for the BAST-RNN model (see below) that borrow ideas from both the dropout and ESN method of regularization in a rigorous fashion.

Allowing for many possible sparse networks is a strength of both the dropout and ESN regularization methods. As discussed throughout the Bayesian machine learning literature, e.g., [25,33], the natural modeling averaging implicit in the fully Bayesian paradigm acts in a similar way to produce a model averaging effect across many potential networks. Generally in a Bayesian framework, model averaging is induced by using one of the many available priors in the Bayesian variable selection literature,

see [34] for an overview. For example, stochastic search variable selection (SSVS) priors [35] represent an effective tool for shrinking parameter values infinitesimally close to zero. In general, SSVS priors consist of a mixture of two distributions, in which one of the distributions shrinks the parameter value near (or to) zero, while the other distribution in the mixture is more vague and allows the parameter to be non-zero.

Although the traditional SSVS prior uses Gaussian distributions, i.e., [36] for the weight matrices $\mathbf{W}$ and $\mathbf{U}$, we replace these Gaussian distributions with a truncated normal to create a "hard" constraint (see (6) and (7) below). As has been previously noted in the Bayesian neural network literature, i.e., [37], the parameters at the top level of the model (i.e., $\mathbf{V}_1\mathbf{h}_t$ and $\mathbf{V}_2\mathbf{h}_t^2$ for the BAST-RNN model in (3)) are not identifiable. By using truncated normal distributions, we are in some sense constraining the contribution of each weight matrix $\mathbf{W}$ and $\mathbf{U}$ towards the $\mathbf{h}_t$'s. While helping to partly alleviate this identifiability problem, we also find that using truncated normals helps improve mixing when performing MCMC estimation. Finally, as discussed in [37], this non-identifiability is not an issue when parameters are given proper priors and interest is only in prediction, as is the case here.

Using the SSVS framework described above, each element of the weight matrix $\mathbf{W} = \{w_{i,\ell}\}$, for $i = 1, \ldots, n_h$ and $\ell = 1, \ldots, n_h$, is given the following prior distribution:

$$w_{i,\ell} = \gamma_{i,\ell}^{w} \mathrm{TN}_{[-a_w, a_w]}(0, \sigma_{w,0}^2) + (1 - \gamma_{i,\ell}^{w}) \mathrm{TN}_{[-a_w, a_w]}(0, \sigma_{w,1}^2), \tag{6}$$

where $\sigma_{w,0}^2 \gg \sigma_{w,1}^2$ and the notation $\mathrm{TN}_{[-a_w, a_w]}$ denotes a truncated normal distribution, truncated between $-a_w$ and a_w. Moreover, $\gamma_{i,\ell}^{w}$ represents an indicator variable with prior, $\gamma_{i,\ell}^{w} \sim \text{Bernoulli}(\pi_w)$, such that π_w can be thought of as the prior probability of including $w_{i,\ell}$ in the model. An analogous prior is used for each element of $\mathbf{U} = \{u_{i,r}\}$ for $r = 1, \ldots, (m+1)n_x + 1$, such that:

$$u_{i,r} = \gamma_{i,r}^{u} \mathrm{TN}_{[-a_u, a_u]}(0, \sigma_{u,0}^2) + (1 - \gamma_{i,r}^{u}) \mathrm{TN}_{[-a_u, a_u]}(0, \sigma_{u,1}^2), \tag{7}$$

where $\gamma_{i,r}^{u} \sim \text{Bernoulli}(\pi_u)$ and $\sigma_{u,0}^2 \gg \sigma_{u,1}^2$. As described in Section 4.2, both hyper-parameters π_w and π_u are set to small values in order to regularize many of the parameters in the model (since $\sigma_{w,1}^2$ and $\sigma_{u,1}^2$ are set to very small values, as detailed in Appendix A). The priors defined in (6) and (7) mimic aspects of the regularization produced from using dropout or the ESN model by similarly removing or (nearly) zeroing out many of the model parameters, but in a more formal framework. While developing the proposed model we also considered other popular Bayesian variable selection priors such as the Lasso prior [38] and the horseshoe prior [39]. We found the proposed SSVS priors provided the most flexibility with our approach and provided a similar mechanism as the dropout method in helping to prevent over-fitting. Note, there are many other variations of the original SSVS prior, such as the spike-and-slab LASSO [40], that could also be utilized within the presented methodology.

Next, the parameters matrices $\mathbf{V}_1$ and $\mathbf{V}_2$ are given traditional SSVS priors with Gaussian distributions (see Appendix A for the full details). Although a L_2 (ridge) penalty is typically used for estimating the $\mathbf{V}$ matrices in the ESN model, we found this penalty to be less flexible. To finish the specification of the model, the spatial intercept is given the Gaussian prior, $\boldsymbol{\mu} \sim \text{Gau}(\mathbf{0}, \sigma_\mu^2 \mathbf{I})$, and the variance parameter σ_ϵ^2 is given the inverse-gamma prior, $\sigma_\epsilon^2 \sim \text{IG}(\alpha_\epsilon, \beta_\epsilon)$. See Appendix A for the specific values of the presented hyper-parameters and Section 4.2 for a further discussion on certain hyper-parameter choices.

2.4. Dimension Reduction

With the rise of machine learning and high-dimensional methods has come the increase in size of spatio-temporal data sets. In most cases, this increase in size can be attributed to the number of spatial locations (or grid-points) in a given data set, rather than the number of time points. When n_y or n_x (or both) is large, the BAST-RNN model can quickly become computationally prohibitive. For example, with more locations, each step of the MCMC algorithm (in particular, Metropolis-Hastings steps) will become more computationally costly. Secondly, with more locations it may be necessary to increase

the value of n_h, thus increasing the size of all the weight parameter matrices in the model. A common solution to this problem in the spatio-temporal dynamical modeling literature is to use some form of dimension reduction, i.e., [10]. This is primarily because there is a great deal of dependence in the spatial dimension and the underlying dynamics live on a much lower dimensional manifold than the dimension of the spatial data. It is common then that a first encoding step is conducted to project the spatio-temporal data into lower dimensional space, and then the complex dynamics are modeled from this lower dimensional space.

There is a great deal of flexibility when selecting a dimension reduction method for high-dimensional spatio-temporal processes. Depending on the application, any number of methods can be selected from linear methods such as wavelets, splines, or principal components, or nonlinear methods such as Laplacian eigenmaps [41], restricted Boltzmann machines [42], or diffusion maps [43], just to name a few. To describe how dimension reduction can be used with the BAST-RNN model, suppose we let $\mathbf{Z}_t$ be a n_z-dimensional observed response vector at time t. Then, for linear dimension reduction, $\mathbf{Z}_t$ can be decomposed such that $\mathbf{Z}_t \approx \mathbf{\Phi}\mathbf{Y}_t$, where $\mathbf{\Phi}$ is a $n_z \times n_b$ basis function matrix and $\mathbf{Y}_t$ is a n_b-dimensional vector of basis coefficients. Importantly, we assume that $n_b \ll n_z$, thus, $\mathbf{Y}_t$ provides a lower-dimensional set of variables (expansion coefficients) with which our model can be built. For example, the proposed BAST-RNN model can be re-formulated using the basis coefficients as follows:

$$\text{data model:} \qquad \mathbf{Z}_t = \mathbf{\Phi}\mathbf{Y}_t + \boldsymbol{\nu}_t \;\; \boldsymbol{\nu}_t \sim \text{Gau}(\mathbf{0}, \mathbf{\Sigma}_\nu), \tag{8}$$

$$\text{process model:} \qquad \mathbf{Y}_t = \boldsymbol{\mu} + \mathbf{V}_1\mathbf{h}_t + \mathbf{V}_2\mathbf{h}_t^2 + \boldsymbol{\epsilon}_t \;\; \boldsymbol{\epsilon}_t \sim \text{Gau}(\mathbf{0}, \mathbf{R}, \tag{9}$$

where the error term $\boldsymbol{\nu}_t$ helps account for the truncation error caused by using a reduced dimension. For some applications it may be more important than others to include the error term (i.e., $\boldsymbol{\nu}_t$) in (8). Although the forecasting applications examined below do not include this truncation error term, the proposed framework allows for the potential to account for such uncertainty.

3. Computation: Parameter Expansion MCMC

Similar to non-Bayesian RNN estimation, the nonlinearity and dependence structures in the BAST-RNN model present unique estimation and computational challenges. Both the $\mathbf{W}$ and $\mathbf{U}$ weight matrices in the BAST-RNN model are particularly difficult to estimate due to the fact that both are within the nonlinear activation function, along with the many dependencies that exist between these two matrices. This dependence occurs since, given the embedded input ($\widetilde{\boldsymbol{x}}_t$ above) and δ, the hidden states in (4) are completely determined by $\mathbf{W}$ and $\mathbf{U}$. Thus, as $\mathbf{W}$ and $\mathbf{U}$ change, so do the values of the hidden states. Importantly, since $\mathbf{W}$ weights the hidden states, the parameter values of $\mathbf{W}$ are highly dependent on the specific values of the hidden states and by proxy, the values of $\mathbf{U}$.

Parameter expansion data augmentation (PXDA), e.g., [26–28], is a method developed for missing data problems in which mixing for MCMC algorithms is difficult due to dependencies between parameters. While the parameter expansion in the PXDA algorithm is generally applied to missing data, we borrow the parameter expansion idea and apply it to the sampling of the $\mathbf{W}$ matrix (we did not find it necessary to use this same technique on the $\mathbf{U}$ matrix), since $\mathbf{W}$ directly weights the hidden states. In essence, parameter expansion MCMC (PX-MCMC) introduces extra parameters (referred to as the expansion parameters) into the model to create extra randomness. For example, suppose for a given iteration of the MCMC algorithm we sampled $\mathbf{W}$ and then $\mathbf{U}$. Instead of moving from $\mathbf{W}$ to $\mathbf{U}$ (i.e., $\mathbf{W} \rightarrow \mathbf{U}$), the expanded parameter is used to create an intermediate step such that $\mathbf{W} \rightarrow \mathbf{W}^* \rightarrow \mathbf{U}$. That is, $\mathbf{W}^*$ is a randomly transformed version of $\mathbf{W}$, thus helping to break some of the dependence between weight matrices $\mathbf{W}$ and $\mathbf{U}$ [28] (refers to this randomness as a "shake-up" of the parameters). Without this extra randomness, samples for the weight matrices $\mathbf{W}$ and $\mathbf{U}$ quickly become degenerate.

By introducing this intermediate step, the mixing in the MCMC algorithm greatly improves for both the $\mathbf{W}$ and $\mathbf{U}$ matrix. The amount of randomness used to transform $\mathbf{W}$ into $\mathbf{W}^*$ can be thought

of similarly to the learning rate parameter used in traditional stochastic gradient descent (SGD) or SG-MCMC algorithms for machine-learning problems. Typically, a learning rate parameter is used in SGD algorithms to determine how fast or slow the weights in a given model are learned. For example, in SG-MCMC, at each iteration when the model parameters are updated, a standard multivariate Gaussian distribution multiplied by a learning rate parameter is added to the updated parameter values, i.e., [25]. Analogous to the learning rate for SGD, the extra randomness induced by $\mathbf{W}^*$ allows the algorithm to better search the entire parameter space, thus improving the mixing of the algorithm.

To more rigorously describe the PX-MCMC algorithm, we need to define additional notation. Suppose we introduce the expansion parameter matrix $\boldsymbol{\alpha}$, where $\boldsymbol{\alpha} = \{\alpha_{i,\ell}\}$ for $i = 1, \ldots, n_h$ and $\ell = 1, \ldots, n_h$, and $\boldsymbol{\alpha} \subset \mathbf{A}$, where $\mathbf{A} \in \mathbb{R}^{n_h^2}$. Next, we define the transformation $t_{\boldsymbol{\alpha}} : \mathbf{W} \longrightarrow \mathbf{W}$, where we require $t_{\boldsymbol{\alpha}}$ to be a one-to-one differentiable function and denote the Jacobian for this transformation as $J_{\boldsymbol{\alpha}}(\mathbf{W})$. Let $\boldsymbol{\Gamma}_{V_1}, \boldsymbol{\Gamma}_{V_2}, \boldsymbol{\Gamma}_U$, and $\boldsymbol{\Gamma}_W$ denote all of the indicator variables for the SSVS priors corresponding to the respective weight matrices in (3) and (4). We define Θ to be all of the parameters in the model not associated with $\mathbf{W}$; that is, $\Theta \equiv \{\boldsymbol{\mu}, \mathbf{V}_1, \mathbf{V}_2, \boldsymbol{\Gamma}_{V_1}, \boldsymbol{\Gamma}_{V_2}, \mathbf{U}, \boldsymbol{\Gamma}_U, \delta, \sigma_\epsilon^2\}$. Furthermore, let $\mathbf{Y}_{1:T} \equiv \{\mathbf{Y}_1, \ldots, \mathbf{Y}_T\}$ and $\widetilde{\mathbf{x}}_{1:T} \equiv \{\widetilde{\mathbf{x}}_1, \ldots, \widetilde{\mathbf{x}}_T\}$. Finally, we define the likelihood of the model (before the introduction of the expansion parameter matrix $\boldsymbol{\alpha}$) using the following slight abuse of notation $\prod_{t=1}^{T} [\mathbf{Y}_t \mid \Theta, \mathbf{W}, \boldsymbol{\Gamma}_W, \widetilde{\mathbf{x}}_t] = [\mathbf{Y}_{1:T} \mid \Theta, \mathbf{W}, \boldsymbol{\Gamma}_W, \widetilde{\mathbf{x}}_{1:T}]$, where $[\cdot]$ denotes a distribution.

Now, we outline the PX-MCMC for the BAST-RNN model; note, we leave the detailed description of the presented algorithm for Appendix B. Using the notation defined above, one can show (see Appendix B) the following relationship for the joint posterior of $\mathbf{W}$ and $\boldsymbol{\Gamma}_W$:

$$[\mathbf{W}, \boldsymbol{\Gamma}_W \mid \Theta, \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}] = \int_{\mathbf{A}} [t_{\boldsymbol{\alpha}}(\mathbf{W}), \boldsymbol{\Gamma}_W \mid \Theta, \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}] \, |J_{\boldsymbol{\alpha}}(\mathbf{W})| \, [\boldsymbol{\alpha}] \, d\boldsymbol{\alpha}. \tag{10}$$

To sample from the integral in (10), we assume $\mathbf{W}', \boldsymbol{\Gamma}_W \sim [t_{\boldsymbol{\alpha}}(\mathbf{W}), \boldsymbol{\Gamma}_W \mid \Theta, \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}]$. We will take $\mathbf{W} = t_{\boldsymbol{\alpha}}^{-1}(\mathbf{W}')$, thus allowing for the joint sampling of $\mathbf{W}$ and $\boldsymbol{\Gamma}_W$, leading to Step 1 in Algorithm 1. Next, the draw from $[\boldsymbol{\alpha}]$ is denoted as $\boldsymbol{\alpha}_0$, thus Step 2 in Algorithm 1.

Algorithm 1 PX-MCMC algorithm.

1. Sample $\mathbf{W}, \boldsymbol{\Gamma}_W$ from: $[\mathbf{W}, \boldsymbol{\Gamma}_W \mid \Theta, \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}] \propto$
   $[\mathbf{Y}_{1:T} \mid \Theta, \mathbf{W}, \boldsymbol{\Gamma}_W, \widetilde{\mathbf{x}}_{1:T}] \, [\mathbf{W} \mid \boldsymbol{\Gamma}_W] \, [\boldsymbol{\Gamma}_W]$.
2. Generate $\alpha_{0,i,\ell} \sim \text{Gau}(0, \sigma_\alpha^2)$ for $i = 1, \ldots, n_h$ and $\ell = 1, \ldots, n_h$.
3. Transform $\widetilde{\mathbf{W}} = t_{\boldsymbol{\alpha}_0}^{-1}(\mathbf{W})$.
4. Sample $\boldsymbol{\alpha}$ from: $[\boldsymbol{\alpha} \mid t_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}}), \boldsymbol{\Gamma}_W, \Theta, \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}] \propto$
   $[\mathbf{Y}_{1:T} \mid t_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}}), \boldsymbol{\Gamma}_W, \Theta, \boldsymbol{\alpha}, \widetilde{\mathbf{x}}_{1:T}] \, [t_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}}) \mid \boldsymbol{\Gamma}_W, \boldsymbol{\alpha}] \, [\boldsymbol{\alpha}] \, |J_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}})|$.
5. Sample Θ from: $[\Theta \mid t_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}}), \boldsymbol{\Gamma}_W, \boldsymbol{\alpha}, \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}] \propto [\mathbf{Y}_{1:T} \mid t_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}}), \Theta, \widetilde{\mathbf{x}}_{1:T}] \, [\Theta]$.

We assume $\boldsymbol{\alpha} \sim \text{Gau}(\mathbf{0}, \sigma_\alpha^2 \mathbf{I})$ for the BAST-RNN model implementation, where the prior variance σ_α^2 can be thought of as analogous to the learning rate parameter used in many machine learning estimation algorithms. There is a great deal of flexibility with regards to the particular distribution used for $[\boldsymbol{\alpha}]$, i.e., [28], and its choice should depend on the particular model and application. Letting $\widetilde{\mathbf{W}} \equiv t_{\boldsymbol{\alpha}_0}^{-1}(\mathbf{W})$, we can sample $\boldsymbol{\alpha}$ and Θ using the following full-conditional distributions (see Appendix B for further details):

$$\begin{aligned}
&[\boldsymbol{\alpha} \mid t_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}}), \boldsymbol{\Gamma}_W, \Theta, \mathbf{Y}_{1:T}, , \widetilde{\mathbf{x}}_{1:T}] \propto \\
&[\mathbf{Y}_{1:T} \mid t_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}}), \boldsymbol{\Gamma}_W, \Theta, \boldsymbol{\alpha}, \widetilde{\mathbf{x}}_{1:T}][t_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}}) \mid \boldsymbol{\Gamma}_W, \boldsymbol{\alpha}] \, [\boldsymbol{\alpha}] \, |J_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}})|, \qquad (11) \\
&[\Theta \mid t_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}}), \boldsymbol{\Gamma}_W, \boldsymbol{\alpha}, \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}] \quad \propto [\mathbf{Y}_{1:T} \mid t_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}}), \Theta, \widetilde{\mathbf{x}}_{1:T}] \, [\Theta]. \qquad (12)
\end{aligned}$$

Taking the previous three equations together, we can form the PX-MCMC algorithm given in Algorithm 1. For the sake of brevity, we leave the specific full-conditional distributions for all the model parameters for Appendix C.

4. Applications

We begin by discussing the specifics of model implementation, including comparison metrics and methods, the MCMC setup, and specific hyper-parameter choices. We then present the analysis of a simulated multiscale Lorenz data set from the Lorenz dynamical system. In addition, the setup and results of a Pacific SST long-lead forecasting problem are given, followed by an application to state-level unemployment data in the U.S.

4.1. Validation Measures and Alternative Models

Since the stated goal of developing the BAST-RNN model is to produce accurate forecasts with realistic uncertainty bounds, we evaluate the model in terms of both mean squared prediction error (MSPE) and continuous ranked probability score (CRPS). Both measures are only calculated for out-of-sample values, since the focus of the model is on forecasting. For our purposes, the MSPE is defined as the average squared difference between the out-of-sample forecasts and true out-of-sample values across all time periods and spatial locations. Moreover, for a predictive CDF F and true out-of-sample realization h, CRPS is defined as e.g., [44]:

$$\text{CRPS}(F, h) = \int_{\mathbb{R}} (F(r) - \mathbb{1}\{r \geq h\})^2 dr. \tag{13}$$

The usefulness of CRPS lies in its ability to both quantify the accuracy and distribution of a forecast, thus producing a principled (proper scoring rule) measure of how well a model quantifies uncertainty, i.e., [45]. In all the applications presented below, after the model is trained on in-sample data, out-of-sample forecasts are generated successively at the given lead time. We define lead time as the temporal difference between the input and the response. These successive forecasts are made by repeatedly plugging in the inputs for a given lead time to get out-of-sample forecasts, using the posterior samples from the BAST-RNN.

For the sake of comparison, we also evaluated the ensemble quadratic ESN (E-QESN) model from [20] for all of the applications below. The E-QESN model presents a strong comparison model since it can also quantify uncertainty and shares much of the same flexibility as the BAST-RNN model. Few other methods share the E-QESN's ability to produce forecasts with uncertainty quantification at such a low computational and implementation cost. Unlike the BAST-RNN, the E-QESN is not presented within a Bayesian framework and thus produces less theoretical sound uncertainty estimates.

We also compared to a model referred to as the linear DSTM, e.g., [10] (Chapter 7), defined here as:

$$Y_{t,i} = \sum_{j=1}^{n_y} a_{ij} Y_{t-1,j} + \zeta_{t,i}^{(l)}, \tag{14}$$

for each location $Y_{t,i}$, where $\{a_{i,j}\}$ are weight parameters and $\zeta_{t,i}^{(l)}$ is a spatially referenced noise term, such that $\boldsymbol{\zeta}_t^{(l)} \sim \text{Gau}(\mathbf{0}, \boldsymbol{\Sigma}_{\zeta_l})$. Finally, we compared to the GQN model discussed above [7]. For the results presented below, the GQN model is defined as:

$$Y_{t,i} = \sum_{j=1}^{n_y} a_{ij} Y_{t-1,j} + \sum_{k=1}^{n_y} \sum_{\ell=1}^{n_y} b_{i,k,\ell} Y_{t-1,k} Y_{t-1,\ell} + \zeta_{t,i}^{(q)}, \tag{15}$$

where $\boldsymbol{\zeta}_t^{(q)} \sim \text{Gau}(\mathbf{0}, \boldsymbol{\Sigma}_{\zeta_q})$. Both $\boldsymbol{\Sigma}_{\zeta_l}$ and $\boldsymbol{\Sigma}_{\zeta_q}$ are estimated empirically using the residuals from the training period. Although both the GQN and linear DSTM can be formulated as Bayesian models,

such formulations are not pursued here. Instead, forecast distributions are calculated through a Monte Carlo approach for both models. While this is not an exhaustive list of comparison methods, these methods represent much of the state-of-the art in statistical spatial-temporal modeling and nonlinear spatial-temporal forecasting. Note, although not pursued here, there are many other spatio-temporal nonlinear models from the applied mathematical literature that could also be applied to the applications presented here, such as the stochastic mode reduction models of [46] and the empirical model reduction methods from [47].

4.2. BAST-RNN Implementation Details

Note, the implementation settings discussed here are used for all of the presented applications, with slight deviations for specific applications as discussed below. The BAST-RNN model is implemented using the PX-MCMC algorithm (Algorithm 1), sampling 100,000 iterations with the first 25,000 iterations treated as burn-in, while thinning the samples such that every fifth post burn-in sample was retained. We monitored convergence by examining the trace plots for the parameters in the model along with the posterior forecasts (a sample of such trace plots can be found in Appendix D). The number of hidden units (n_h) is set to 20. We found this number of hidden units balanced computational cost and forecast accuracy in that larger numbers of hidden units produced similar results in terms of forecast accuracy, but substantially slowed the algorithm. Although not pursued here, the number of hidden units could be varied by using advanced computational methods such as reversible jump MCMC [48]. Selection of the parameters for the embedded input, defined in (5), is conducted by using cross-validation, over an application specific grid with the E-QESN model, see ([20], for a detailed description of this procedure). As suggested by [49], both the input and response are scaled by their respective means and standard deviations.

While we leave the specific hyper-parameter values used in the prior distributions to Appendix A, we will briefly discuss these choices. Specifically for the parameter weight matrices that make up the hidden units (i.e., $\mathbf{W}$ and $\mathbf{U}$), the hyper-parameters π_w and π_u (as defined in Section 2.3) are set to small values to encourage sparseness and prevent overfitting. In particular, these hyper-parameters are set such that $\pi_w > \pi_u$, since the matrix $\mathbf{U}$ is weighting the data; we found this specification helped prevent overfitting to the in-sample data. Moreover, a_w and a_u are both set to small values so that $a_w = a_u$, as to follow the common practice in machine learning of bounding parameter values to prevent particular hidden states (and by proxy particular in-sample patterns) from dominating, and thus, avoid overfitting. As discussed in [49], when parameter values are unbounded in the RNN framework the hidden states become extremely nonlinear to the point that the results can become unstable.

4.3. Simulation: Multiscale Lorenz-96 Model

Many RNN methods in the literature use the classic three-variable Lorenz model from [50] to evaluate forecasting ability, e.g., [51,52]. Due to the chaotic and nonlinear behavior of the Lorenz model, this system produces data resembling a realistic nonlinear forecasting problem, but it has an unrealistically low state dimension (three) and is not spatially referenced. Here, evaluation of the BAST-RNN model is applied to a less cited, but more spatially interesting Lorenz model [53], often referred to as the Lorenz-96 model, which explicitly includes spatial locations and structure. In particular, we consider a more complicated extension of the Lorenz-96 model, the *multiscale Lorenz-96* model, that contains interacting large-scale and small-scale processes, where the large-scale locations are directly influenced by neighboring small-scale locations and vice-versa, e.g., [54–56].

While multiple parameterizations exist for the multiscale Lorenz-96 model, we use the following parameterization from [55] (note, the superscript L is used throughout to signify variables from the Lorenz-96 model):

$$\frac{dx^L_{k_L}}{dt} = x^L_{k_L-1}(x^L_{k_L+1} - x^L_{k_L-2}) - x^L_{k_L} + \tilde{F} + \frac{h_x}{J_L}\sum_{j_L} y^L_{j_L,k_L} + \eta^{(1)}_{k_L},$$
$$\frac{dy^L_{j_L,k_L}}{dt} = \frac{1}{\epsilon_L}[y^L_{j_L+1,k_L}(y^L_{j_L-1,k_L} - y^L_{j_L+2,k_L}) - y^L_{j_L,k_L} + h_y x^L_{k_L}], \quad (16)$$

for $j_L = 1, \ldots, J_L$ and $k_L = 1, \ldots, K_L$ (for notational convenience the subscript t has been suppressed from (16)). The state variable $x^L_{k_L}$ denotes the process at a large-scale process location, with each large-scale location having J_L corresponding small-scale locations denoted by the process $y^L_{j_L,k_L}$. Each of the large-scale locations can be thought of as equally spaced spatial variables on a one-dimensional circular spatial domain such that $x^L_{K_L+1} = x^L_1$ (i.e., periodic boundary conditions). A given set of small-scale locations corresponding to a particular large-scale location is defined with a similar spatial domain and boundary condition.

The parameter $\tilde{F}$ in (16) denotes a forcing parameter, while ϵ_L controls the time-scale separation between the large and small-scale processes, $\eta^{(1)}_{k_L}$ is an additive independent Gaussian noise term such that $\eta^{(1)}_{k_L} \overset{iid}{\sim} \text{Gau}(0, \sigma^2_{\eta_1})$, with $\sigma^2_{\eta_1} = 1$, and h_x, h_y control how much the large and small-scale locations influence each other, respectively. For the analysis using the BAST-RNN model, we simulate from the full model but treat the small-scale locations as unobserved and evaluate the BAST-RNN only on the large-scale locations, thus creating a difficult but realistic nonlinear spatio-temporal forecasting problem. After burn-in, 400 time periods are retained from the the multiscale Lorenz-96 model, with the last 75 time periods treated as out-of-sample. The data are simulated with a time step of $\Delta = 0.05$ using an Euler solver. We use the same parameter values as [55] to simulate the data: $K_L = 18, J_L = 20, \tilde{F} = 10, \epsilon_L = 0.5, h_x = -1$, and $h_y = 1$. In order to create a more statistically-oriented forecasting problem, Gaussian white noise error is added to each large-scale realization, so that $z^L_{k_L} = x^L_{k_L} + \eta^{(2)}_{k_L}$, where $\eta^{(2)}_{k_L} \overset{iid}{\sim} \text{Gau}(0, \sigma^2_{\eta_2})$, with $\sigma^2_{\eta_2} = (2.5)^2$. In addition, the forecasting problem is made slightly more nonlinear by setting the lead time to three periods (i.e., the input and response are separated by three periods). Along with using the implementation settings detailed in Section 4.2, the embedded input parameters $\tilde{\tau} = 2$ and $m = 4$ are used.

Posterior mean forecasts and prediction intervals (P.I.s) for the BAST-RNN model with the multiscale Lorenz-96 data are shown for six locations in Figure 1. Note that because a low signal-to-noise ratio was used to simulate the data, the true signal is substantially corrupted by the additive noise (as shown by the blue dotted line used to represent the true signal of the process in Figure 1). Despite the high level of noise, the model recovers much of the signal for the six locations shown in Figure 1. Moreover, it appears that many of the true values of the process are captured by the 95% P.I.s. Across all 18 large-scale locations in the simulated data, 91.9% of the true values are contained within the 95% P.I.s, while only 72.8% of the true values are contained within the intervals produced by the E-QESN model.

A more detailed comparison of the BAST-RNN model and the three models described in Section 4.1 can be found in Table 1. In particular, Table 1 shows the BAST-RNN outperforming the other three competing models by producing lower values for the MSPE. It is not entirely surprising that the BAST-RNN and E-QESN model outperformed the other two less flexible models considering the level of nonlinearity in the simulated data. The different methods of estimating the hidden state parameters may account for the BAST-RNN outperforming the E-QESN in terms of MSPE. In addition, compared to the E-QESN model, Table 1 also shows the BAST-RNN model produces superior uncertainty measures based on a lower CRPS. Overall, these results simultaneously demonstrate the ability of the BAST-RNN model to accurately forecast the trajectory of the states in a nonlinear process, while also giving robust measures of uncertainty.

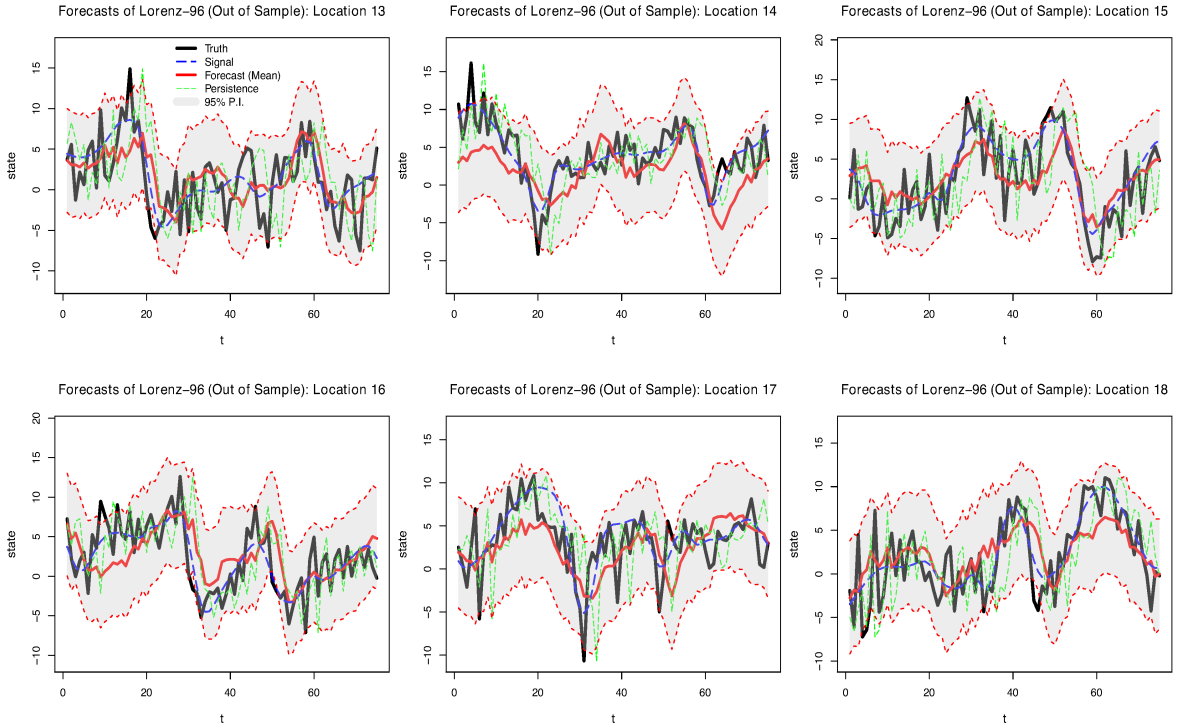

Figure 1. Posterior out-of-sample summaries for six of the 18 large-scale locations from the simulated multiscale Lorenz-96 data over 75 periods. The black line in each plot represents the true simulated value (data), while the red line denotes the forecasted posterior mean from the BAST-RNN model. The blue dashed line denotes the true signal of the process, defined to be the value of the large-scale locations in Equation (16) before the additive error, $\text{Gau}(0, \sigma^2_{\eta_1})$, is applied. The shaded grey area in each plot signifies the 95% posterior prediction intervals. The dotted green line denotes a persistence forecast (i.e., $\hat{Y}_{i,t+\tau} = Y_{i,t}$).

Table 1. Comparison of the four forecasting methods for the simulated multiscale Lorenz-96 data in terms of mean squared prediction error (MSPE) and continuous ranked probability score (CRPS), with a lead time of three periods. Both metrics are calculated over all out-sample periods and locations.

Model	MSPE	CRPS
BAST-RNN	13.08	154.37
E-QESN	13.91	168.61
GQN	14.85	172.50
Lin. DSTM	15.11	166.60

4.4. Application: Long-Lead Tropical Pacific SST Forecasting

Tropical Pacific SST is one of the largest sources of variability affecting global climate, e.g., see the overview in [57]. Changes in SST at various time-scales contribute to extreme weather events across the globe, from hurricanes to severe droughts, as well as related impacts (e.g., waterfowl migration). Therefore, accurate long-lead SST forecasts are vital for many resource managers. Of particular interest when considering SST is the anomalous warming (El Niño) and cooling (La Niña) of the Pacific ocean, referred to collectively as the El Niño Southern Oscillation (ENSO) phenomena.

The focus of our analysis is on the ENSO phenomena that occurred during 2015 and 2016. Besides being one of the most extreme ENSO events on record, many forecasting methods that were effective for past ENSO cycles failed to accurately forecast the 2015–2016 cycle, i.e., [57,58]. As described in [59], there are currently a suite of both deterministic and statistical methods for forecasting SST, with the statistical models often performing as well or better than the deterministic

models. A summary of the deterministic models used to forecast SST can be found in works such as [60] and [61]. Some nonlinear statistical models that have shown success for the ENSO forecasting problem include a general quadratic nonlinear (GQN) model, i.e., [7], a switching Markov model [4], and classic neural network models, i.e., [62]; for a more expansive list of nonlinear SST models see [20]. It is important to note that to our knowledge, this is the first RNN method applied to the ENSO forecasting problem with a formal mechanism for quantifying uncertainty. Furthermore, the BAST-RNN model is used to produce out-of-sample forecasts and P.I.s with a lead time of six months for the 2015–2016 ENSO cycle. Due to the lead time and intensity of the 2015–2016 ENSO cycle, this presents a difficult nonlinear forecasting problem.

The SST forecasting application uses monthly data over a spatial domain covering 29° S–29° N latitude and 124° E–70° W longitude, with a resolution of 2° × 2°, leading to 2229 oceanic spatial locations. The data set is available from the publicly available extended reconstruction sea surface temperature (ERSST) data (http://iridl.ldeo.columbia.edu/SOURCES/.NOAA/) provided by National Ocean and Atmospheric Administration (NOAA) and covers a period from 1970 through 2016. As is common in the climatology literature, the SST data are converted into anomalies by subtracting the monthly climatological means calculated (in this case) over the period 1981–2010, for each spatial location. Furthermore, when constructing ENSO forecasting methods, it is common to evaluate the performance of a given method using the univariate summary measure for ENSO referred to as the Niño 3.4 index. Much of the variability in the ENSO phenomena is contained in the Niño 3.4 region (i.e., 5° S–5° N, 120°–70° W), so for our purposes, the Niño 3.4 index is simply the average SST anomaly over all locations in this region for a given month (see Figure 3).

Training of the model is implemented using Algorithm 1 and the setup from Section 4.2 with data from January 1970–August 2014, while out-of-sample forecasts were made every two months for a period from February 2015–December 2016 (i.e., the 2015–2016 ENSO cycle) with a lead time of six months. Using the description in Section 2.4, dimension reduction is conducted using empirical orthogonal functions (EOFs), also referred to as spatial-temporal principal components, see Chapter 5 of [10], on both the input and response. The first 10 EOFs, which account for over 80% of the variability in the data, are retained for both the input and response. This same number of EOFs has been used in multiple previous SST studies, i.e., [4,63]. Note, the first two EOFs account for almost 57% of the variation in the data. Due to this, some of the variables associated with these EOFs were given higher values for π_u (see Appendix A for the specific values and a more detailed discussion of these choices). Once again, the embedded input parameters are selected using the E-QESN model such that $\tilde{\tau} = 6$ and $m = 4$.

Comparison of the forecasting ability of the BAST-RNN model and the E-QESN model for the entire spatial domain can be seen in Figure 2 for October 2015. Occurring directly before the peak of the 2015–2016 ENSO cycle (see Figure 3), October 2015 represents an important month from the most recent ENSO cycle. Overall, both methods capture much of the warm intensity trend, with the BAST-RNN model forecasting a slightly higher intensity (especially for the Niño 3.4 region) compared to the E-QESN method. Although both methods appear to produce P.I.s with similar upper bounds, the P.I's for the BAST-RNN are narrower, suggesting the model is more confident in its prediction of a warm period. Importantly, the highest intensity true values from the Niño 3.4 region for October 2015 are contained within the 95% P.I.s for the BAST-RNN model.

Furthermore, the BAST-RNN model is evaluated in terms of the previously described Niño 3.4 index in Figure 3. Much of the overall temporal nonlinear trend of the 2015–2016 ENSO cycle is captured by the BAST-RNN model, as shown in Figure 3, with nearly all of the true values contained within the 95% P.I.s. We should note, like the vast majority of ENSO forecasting models, the forecast mean from the BAST-RNN model also underestimates the peak of the ENSO cycle. Considering the 2015–2016 ENSO cycle was one of the most extreme ENSO cycles of recent record, i.e., [58], it is not entirely surprising that most models underestimated the peak of the cycle. However, it is important to

reiterate that the out-of-sample forecast P.I.s for the BAST-RNN model still suggested the possibility of a large warming event during the true peak, unlike many other ENSO forecast models.

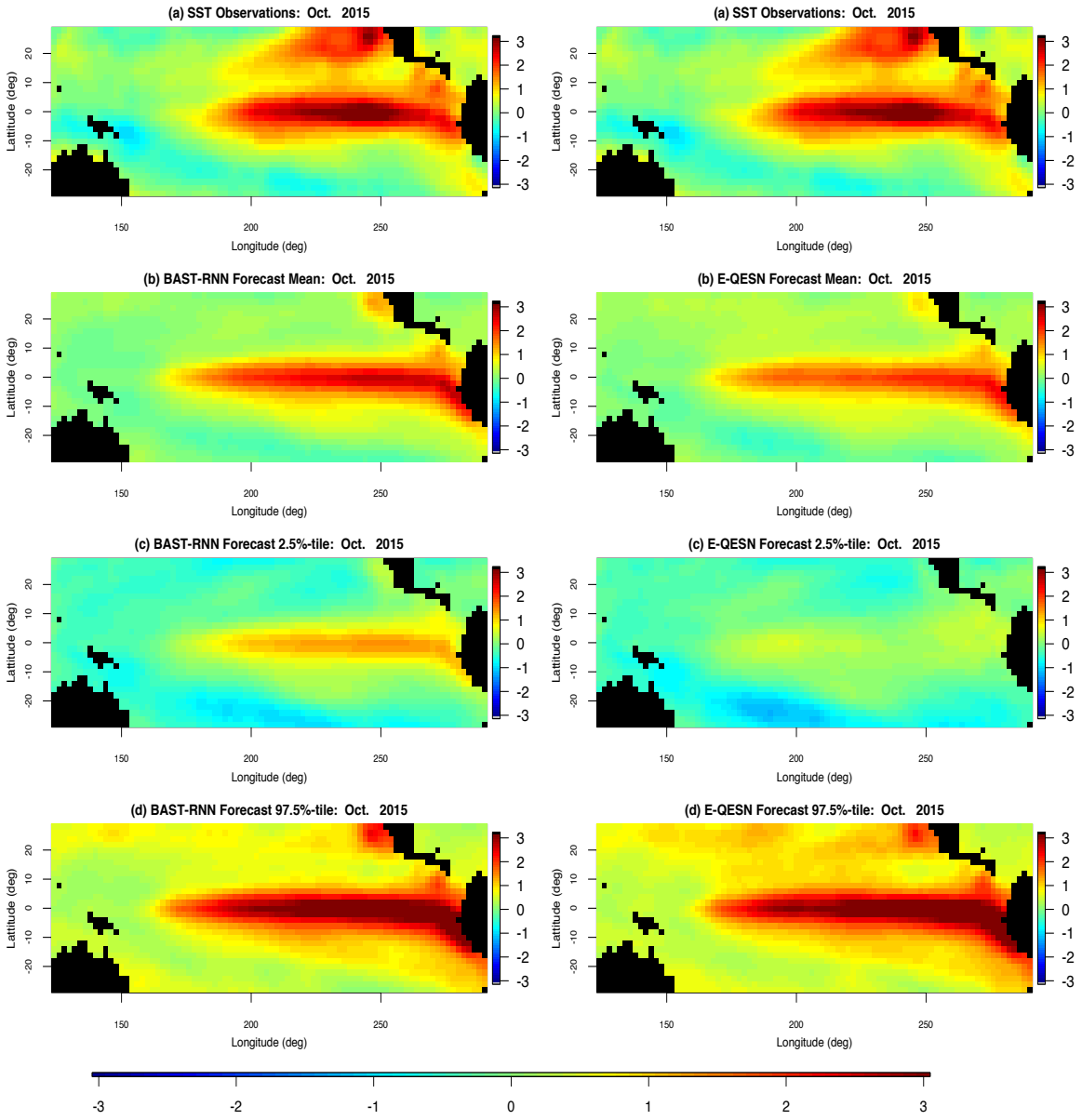

Figure 2. Spatial posterior summaries of sea surface temperature (SST) anomalies for all 2229 oceanic spatial locations in the SST long-lead forecasting application for October 2015. The left column shows results from the BAST-RNN model, while the right column contains results from the competing E-QESN model for the same period. (**a**) The true SST for October 2015. (**b**) Posterior mean out-of-sample forecasts for the BAST-RNN model and mean out-of-sample forecasts over all ensembles for the E-QESN model. (**c**) Lower 2.5% point wise P.I.s for the respective forecasting method. (**d**) Upper 97.5% point wise P.I.s for the respective forecasting method. All plots are in units of degree Celsius.

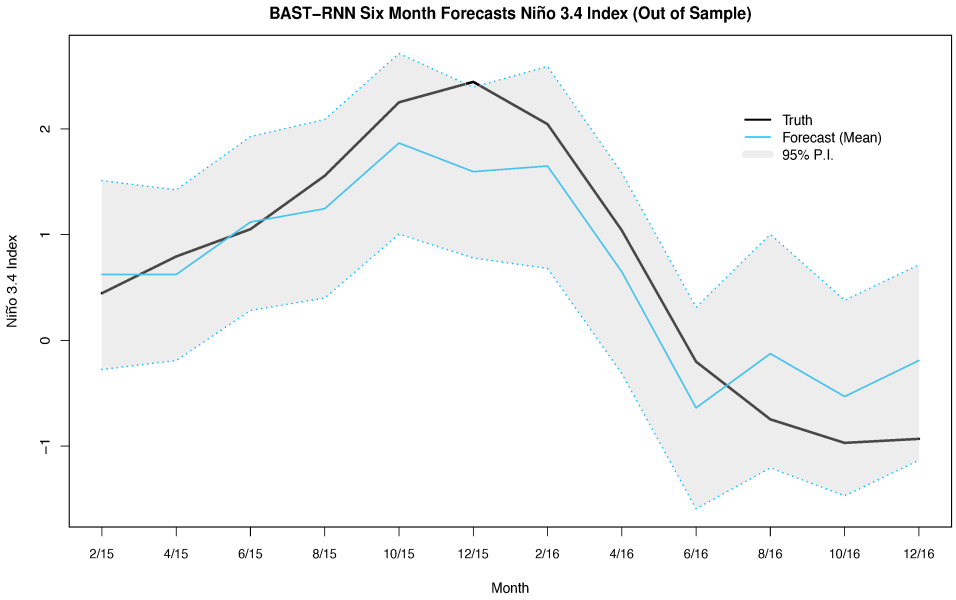

Figure 3. Summary of the posterior results with the BAST-RNN model for the Niño 3.4 index. For a given month, the Niño 3.4 index is defined as the average SST over all locations in the Niño 3.4 region (5° S–5° N, 120°–70° W). The solid black lines denotes the true Niño 3.4 index for a given month during the 2015–2016 cycle. Posterior mean out-of-sample forecasts from the BAST-RNN model are denoted by the light blue line. The grey shaded area represents the 95% P.I.s from the BAST-RNN for the Niño 3.4 index. All values are given in units of degree Celsius.

Once again we evaluate the performance of the BAST-RNN against the three comparison models described above. Throughout the 2015–2016 ENSO cycle, Table 2 shows the BAST-RNN as a more accurate long-lead forecasting model than the other three models. The BAST-RNN model greatly outperforms the other models over the Niño 3.4 region, illustrating the model's ability to forecast nonlinear dynamics. Moreover, Table 2 also shows the BAST-RNN model has the lowest CRPS over all 2229 locations in the application, thus providing useful uncertainty information across the entire spatial domain. By producing sensible uncertainty metrics for events six months into the future, the BAST-RNN model gives resource managers advanced information on which informed decisions can be made. Considering the widespread impact SST has on the global climate, such reliable information is invaluable from both a scientific and economical perspective.

Table 2. Summary metrics for each of the four methods evaluated for the long-lead SST forecasting application. Overall, MSPE denotes the MSPE calculated over all out-of-sample periods and all oceanic locations. The column labeled CRPS denotes the CRPS calculated over all locations and out-of-sample time periods. The columns Niño 3.4 MSPE and Niño 3.4 CRPS denote the MSPE and CRPS, respectively, over all locations in the Niño 3.4 region and all out-of-sample periods.

Model	Overall MSPE	Niño 3.4 MSPE	CRPS	Niño 3.4 CRPS
BAST-RNN	0.253	0.223	3.437	0.318
E-QESN	0.272	0.319	3.455	0.408
GQN	0.309	0.619	3.924	0.538
Lin. DSTM	0.328	0.785	3.752	0.699

4.5. Application: U.S. State-Level Unemployment Rate

Finally, the BAST-RNN model was applied to forecasting state unemployment rates in the Midwest of the U.S. Previous research by [64,65] have shown neural network models to be successful for forecasting national unemployment. A lesser studied, but equally important, component of unemployment forecasting is the spatio-temporal problem of forecasting state-level rates. Moreover,

while linear models have shown success at forecasting unemployment rates at short lead times, nonlinear models generally produce more accurate results at longer lead times, i.e., [64,66]. The BAST-RNN can account for the nonlinearity present for a longer lead forecast, while also incorporating the dependence between unemployment rates in near-by states.

Compared to the previous two applications (Lorenz system and SST), the U.S. unemployment rate is a much slower moving process (i.e., compare Figures 3 and 4 where each displays approximately one (quasi) cycle of the processes of interest, and note that Figure 4 covers a temporal span twice as long as Figure 3). For example, there have been many fewer U.S. unemployment cycles over the past 40 years compared to ENSO cycles. Due to this difference in the rate of the dynamical process, smaller values for the hyper-parameters a_w and a_u are used for the unemployment application (see Appendix A for the specific values and a more detailed discussion). By using smaller values for a_w and a_u the model has more memory of recent past values, i.e., [19], which is necessary for slower moving processes. Similar to the two previous applications, the embedding parameters were selected with the E-QESN model, with the model selecting $\tilde{\tau} = 0$ and $m = 0$ (i.e., $\widetilde{\mathbf{x}}_t{}' = \mathbf{x}_t$ in (5)).

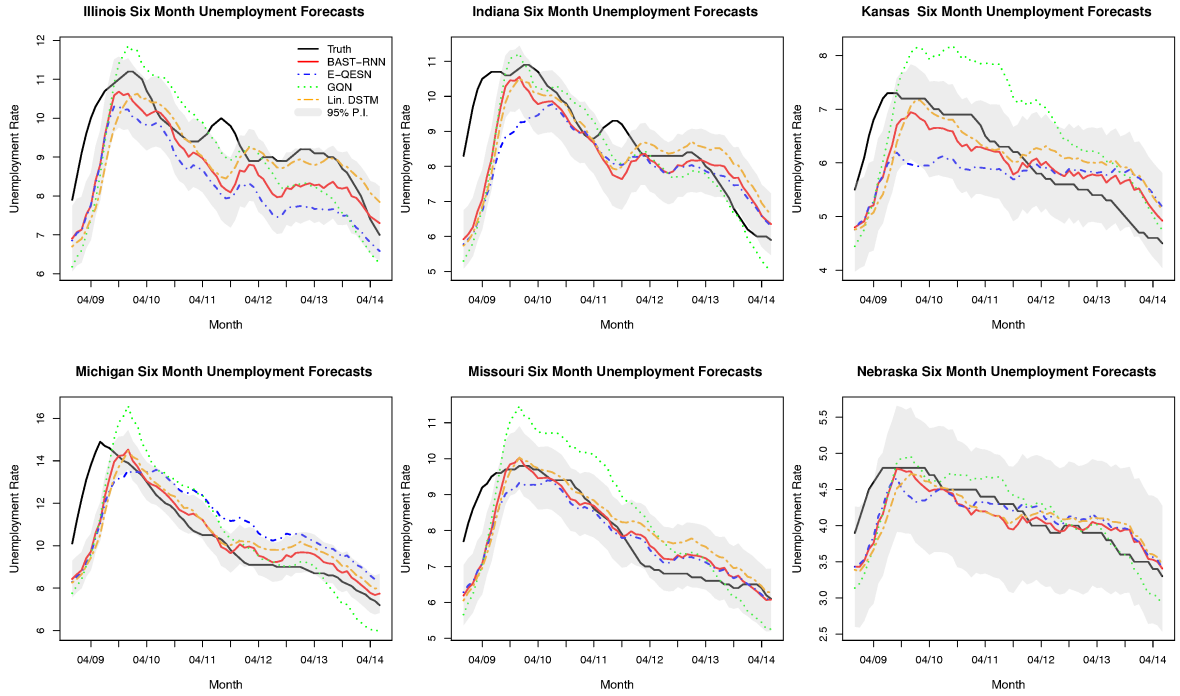

Figure 4. Posterior out-of-sample summaries and comparison methods for six of the 12 states in the U.S. state-level unemployment application. The observed unemployment rate is represented by the black line. The red line denotes the posterior mean from the BAST-RNN model, while the dotted blue, green, and orange line represent the ensemble quadratic ESN (E-QESN), general quadratic nonlinear (GQN), and Linear Dynamical spatial-temporal model (DSTM), respectively. The shaded grey area in each plot signifies the 95% posterior prediction intervals associated with the BAST-RNN model.

Seasonally adjusted monthly unemployment data were obtained from the publicly available Federal Reserve Bank of St. Louis (http://research.stlouisfed.org/fred2), for a period starting in January 1976 for the 12 states that make up the U.S. Census Bureau's Midwest Region [67]. The period from December 2008 through June 2014 was designated as the out-of-sample period to evaluate the performance of the model. This period represents the most recent unemployment cycle caused by the Great Recession that started in 2008 and provides a nonlinear time series to assess the model. Using a lead time of six months, each model was trained on data from January 1976 through May 2008. The results in Table 3 show the BAST-RNN model to be a more accurate forecast model with higher quality uncertainty measures than the three competing models. From Figure 4 it is clear that all four

models struggle with identifying the start of the unemployment cycle in 2009, with the BAST-RNN model generally recovering to accurately predict the states with peaks later in the cycle. Across the six states displayed in Figure 4 (selected to represent a range of different unemployment cycles in the region), the BAST-RNN model appears to be the most accurate in terms of forecasting the decrease in the unemployment rate during the recovery that followed the 2008 recession.

Table 3. Comparison for the four forecasting methods for the U.S. state-level unemployment data in terms of mean squared prediction error (MSPE) and continuous ranked probability score (CRPS), with a lead time of six months. Both metrics are calculated over all out-of-sample periods and states.

Model	MSPE	CRPS
BAST-RNN	0.612	27.11
E-QESN	0.965	36.66
GQN	0.964	37.32
Lin. DSTM	0.865	33.60

5. Discussion and Conclusions

The results of all three applications presented above demonstrate the potential of using machine learning methods within a Bayesian modeling framework for forecasting nonlinear spatio-temporal processes. While many methods struggled with forecasting the 2015–2016 ENSO cycle, the BAST-RNN model forecasted much of the overall cycle correctly, especially when accounting for forecast uncertainty. Additionally, the BAST-RNN model was able to forecast the correct nonlinear trajectory for the multiscale Lorenz data despite a considerable amount of noise. With regards to both forecast accuracy and quantification of uncertainty, the BAST-RNN model was superior to the three competing models, over a reasonably long out-of-sample temporal span across three different applications.

Placing popular machine learning methods, such as RNNs, within a more rigorous statistical framework allows for more thorough uncertainty quantification, while also providing a useful framework for building more complicated models. That is, the proposed BAST-RNN model provides a first step towards more hierarchical machine learning methods that account for sources of variation at multiple levels. High-dimensional real-world processes often contain multiple layers of interconnected uncertainties and these uncertainties can more easily be untangled and formally modeled within the proposed modeling framework. Conversely, even the most precise uncertainty quantification methods are of diminishing value if they are not flexible enough to accurately forecast the process of interest. Thus, by combining the forecasting ability of the RNN model with the rigor of Bayesian modeling, the proposed methodology provides a powerful tool for modelers.

The proposed model can be used for a broad range of forecasting problems (as seen by the variety of applications analyzed here) both in its current form and with relatively minor modifications. For example, the model can easily be extended to account for different types of response data such as binary or count data. Moreover, the BAST-RNN is flexible enough to deal with varying degrees of nonlinearity, whereas past statistical nonlinear forecasting models may fail with higher levels of nonlinearity. The applications shown above provide evidence of this flexibility with the model producing accurate results for both quasilinear processes (SST and unemployment) and a highly nonlinear process (Lorenz process).

Other extensions of the model include letting the parameters associated with the embedded input and the number of hidden units vary, which could more accurately account for the uncertainty associated with these choices. Putting the model within a fully Bayesian hierarchical framework is another possible extension. Moreover, when adding additional hidden layers to the model it may be necessary to incorporate more computational efficient methods to improve scalability, possibly borrowing ideas from the ESN literature. It is also likely that other forms of dimension reduction may be useful when considering the BAST-RNN model for other applications. In particular, incorporating the nonlinearity or dynamics of the process explicitly in the selected dimension reduction method

could be important for some applications. For large data sets, where dimension reduction is not appropriate, it may be necessary to combine the presented computational framework with other computational methods such as Langevin dynamics, i.e., [68].

Author Contributions: Computational code was written by P.L.M.. Moreover, P.L.M. and C.K.W. both contributed to the model design and manuscript preparation.

Funding: This work was partially supported by the U.S. National Science Foundation (NSF) and the U.S. Census Bureau under NSF grant SES-1132031, funded through the NSF-Census Research Network (NCRN) program. In addition, partial support was provided by NSF grant DMS-1811745.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Specification of Priors

Below is a list of prior distributions for all of the parameters in the BAST-RNN model, along with specific values for each of the hyper-parameters used in the model.

Each element in the weight matrix $\mathbf{W}$ is given the following prior distribution:

$$w_{i,\ell} = \gamma^w_{i,\ell}\text{TN}_{[-a_w,a_w]}(0,\sigma^2_{w,0}) + (1-\gamma^w_{i,\ell})\text{TN}_{[-a_w,a_w]}(0,\sigma^2_{w,1}), \text{ for } \quad \gamma^w_{i,\ell} \sim \text{Bernoulli}(\pi_w),$$

where $\sigma^2_{w,0} = (1,000)^2$, $\sigma^2_{w,1} = 0.001$, $a_w = 0.20$, and $\pi_w = 0.20$.

Each element in the weight matrix $\mathbf{U}$ is given the following prior distribution:

$$u_{i,r} = \gamma^u_{i,r}\text{TN}_{[-a_u,a_u]}(0,\sigma^2_{u,0}) + (1-\gamma^u_{i,r})\text{TN}_{[-a_u,a_u]}(0,\sigma^2_{u,1}), \text{ for } \quad \gamma^u_{i,r} \sim \text{Bernoulli}(\pi_u),$$

where $\sigma^2_{u,0} = (1,000)^2$, $\sigma^2_{u,1} = 0.0005$, $a_u = 0.20$, and $\pi_u = 0.025$.

Each element in the weight matrix $\mathbf{V}_1$ is given the following prior distribution:

$$v_{1,k,i} = \gamma^v_{1,k,i}\text{Gau}(0,\sigma^2_{v_1,0}) + (1-\gamma^v_{1,k,i})\text{Gau}(0,\sigma^2_{v_1,1}), \text{ for } \quad \gamma_{1,k,i} \sim \text{Bernoulli}(\pi_{v_1}),$$

where $\sigma^2_{v_1,0} = 10, \sigma^2_{v_1,1} = 0.01$, and $\pi_{v_1} = 0.50$.

Each element in the weight matrix $\mathbf{V}_2$ is given the following prior distribution:

$$v_{2,k,i} = \gamma^v_{2,k,i}\text{Gau}(0,\sigma^2_{v_2,0}) + (1-\gamma^v_{2,k,i})\text{Gau}(0,\sigma^2_{v_2,1}), \text{ for } \quad \gamma_{2,k,i} \sim \text{Bernoulli}(\pi_{v_2}),$$

where $\sigma^2_{v_2,0} = 0.5$, $\sigma^2_{v_2,1} = 0.05$, and $\pi_{v_2} = 0.25$.

Finally, $\boldsymbol{\alpha} \sim \text{Gau}(\mathbf{0}, \sigma^2_\alpha \mathbf{I})$, where $\sigma^2_\alpha = (.10)^2$, $\quad \boldsymbol{\mu} \sim \text{Gau}(\mathbf{0}, \sigma^2_\mu \mathbf{I})$, where $\sigma^2_\mu = 100$, $\quad \delta \sim \text{Unif}(0,1)$, $\sigma^2_\epsilon \sim \text{IG}(\alpha_\epsilon, \beta_\epsilon)$, where $\alpha_\epsilon = 1$ and $\beta_\epsilon = 1$.

For the SST application, π_u was set to 0.05 for all of the $\mathbf{U}$ parameters associated with the first EOF as well as $\mathbf{U}$ parameters associated with non-lagged inputs corresponding to the second EOF. Overall, the first two EOFs account for almost 57% of the variation in the SST data with the first EOF accounting for 46% of the overall variation by itself, thus suggesting higher prior probabilities for these inputs to be included in the model.

As discussed in the main text, the U.S. state-level unemployment application involved a much slower moving process than the SST and Lorenz system examples. In particular, this can be seen by comparing Figures 3 and 4, where Figure 3 displays an approximately completed ENSO cycle, and Figure 4 shows part of an unemployment cycle, while Figure 4 covers a temporal span twice as long as Figure 3. To account for this difference, the hyper-parameters a_w and a_u were adjusted for the unemployment application such that, $a_u = 0.05$ and $a_w = 0.05$.

To justify these prior choices we simulated from (4) with different values of a_w and a_u, while setting the input to zero for every period after the first period. As shown in Figure A1, this procedure allowed us to examine the memory of the hidden units for different values of a_w and a_u. The results in Figure A1 show that for smaller values of a_w and a_u, the hidden units have more memory (i.e.,

the original signal dies off slowly). Therefore smaller values of a_w and a_u are more appropriate for the slower moving unemployment example, which requires more memory of the recent trajectory of the process. It is important to note that the focus of Figure A1 is on how quickly the original signal is forgotten for values of a_w and a_u rather than on the displayed dynamical speeds.

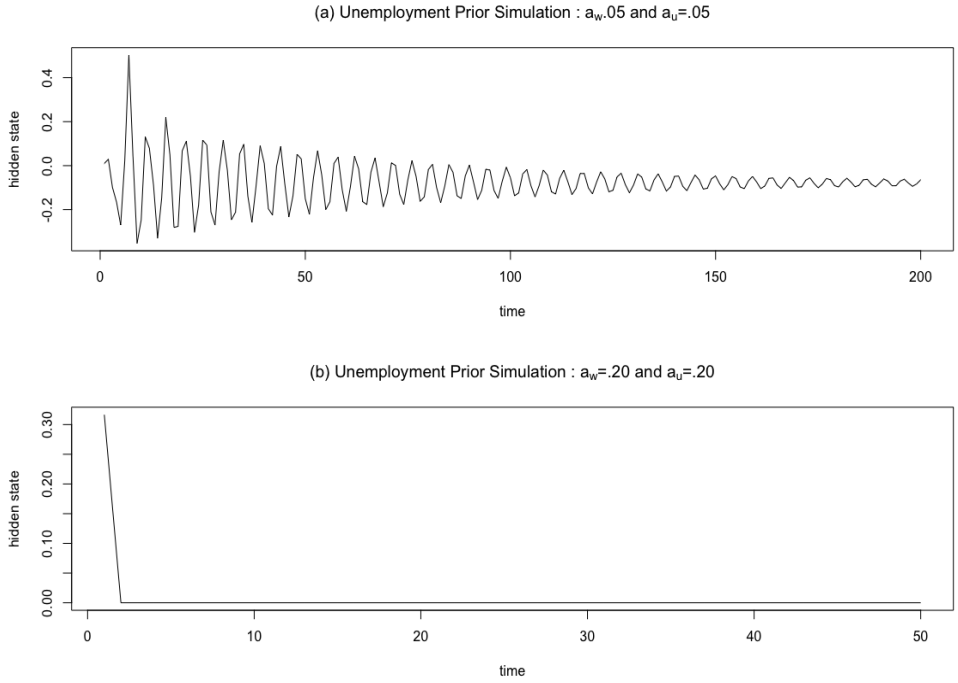

Figure A1. Prior simulation for the unemployment application to demonstrate the difference in memory of the hidden units, $\mathbf{h}_t$, for different values of a_w and a_u. (**a**) The first 200 periods from a simulation from (4) with $a_w = 0.05$ and $a_u = 0.05$, for a fixed input. (**b**) The first 50 periods from a simulation from (4) with $a_w = 0.20$ and $a_u = 0.20$, for a fixed input. Together, (**a**,**b**) illustrate how smaller values of a_w and a_u are more appropriate for slower moving processes that require more memory, such as the unemployment application.

Appendix B. Details of Algorithm 1

Suppose we introduce the expansion parameter $\boldsymbol{\alpha}$ where $\boldsymbol{\alpha} = \{\alpha_{i,\ell}\}$ for $i = 1, \ldots, n_h$ and $\ell = 1, \ldots, n_h$, and $\boldsymbol{\alpha} \subset \mathbf{A}$ where $\mathbf{A} \in \mathbb{R}^{n_h^2}$. Define the following function $t_{\boldsymbol{\alpha}} : \mathbf{W} \longrightarrow \mathbf{W}$, where we require that $t_{\boldsymbol{\alpha}}$ is a one-to-one differentiable function. Let Θ represent all of the parameters in the model not associated with $\mathbf{W}$, such that $\Theta \equiv \{\mathbf{V}_1, \mathbf{V}_2, \boldsymbol{\mu}, \boldsymbol{\Gamma}_{V_1}, \boldsymbol{\Gamma}_{V_2}, \mathbf{U}, \boldsymbol{\Gamma}_U, \delta, \sigma_\epsilon^2\}$. Next, let $\mathbf{Y}_{1:T} \equiv \{\mathbf{Y}_1, \ldots, \mathbf{Y}_T\}$ and $\widetilde{\mathbf{x}}_{1:T} \equiv \{\widetilde{\mathbf{x}}_1, \ldots, \widetilde{\mathbf{x}}_T\}$.

We define the likelihood of the model (before the introduction of the expansion parameter matrix $\boldsymbol{\alpha}$) using the following slight abuse of notation $\prod_{t=1}^{T} [\mathbf{Y}_t \mid \Theta, \mathbf{W}, \boldsymbol{\Gamma}_W, \widetilde{\mathbf{x}}_t] = [\mathbf{Y}_{1:T} \mid \Theta, \mathbf{W}, \boldsymbol{\Gamma}_W, \widetilde{\mathbf{x}}_{1:T}]$, with the notation $[\cdot]$ denoting a distribution. We assume that $\boldsymbol{\alpha}$ is only dependent on $\Theta, \mathbf{W}, \boldsymbol{\Gamma}_W$, and $\mathbf{Y}_{1:T}$ through the transformation $t_{\boldsymbol{\alpha}}$ and independent of these values otherwise.

The function $t_{\boldsymbol{\alpha}}$ is defined as follows: $t_{\boldsymbol{\alpha}}(\mathbf{W}) = \{t_{\alpha_{i,\ell}}(w_{i,\ell})\} = \{\kappa(w_{i,\ell} - \alpha_{i,\ell})\}$, where: $\kappa(q_\kappa) = -a + \frac{2a}{1+e^{-q_\kappa}}$, thus ensuring $q_\kappa \in [-a, a]$. The Jacobian for the transformation $t_{\boldsymbol{\alpha}}(\mathbf{W})$, is defined as $J_{\boldsymbol{\alpha}}(\mathbf{W}) = \frac{\partial}{\partial \mathbf{W}} t_{\boldsymbol{\alpha}}(\mathbf{W}) = \prod_{i=1}^{n_h} \prod_{\ell=1}^{n_h} \frac{\partial t_{\alpha_{i,\ell}}(w_{i,\ell})}{\partial w_{i,\ell}}$, while the Jacobian for the transformation $t_{\boldsymbol{\alpha}}^{-1}(\mathbf{W})$, is defined as $\widetilde{J}_{\boldsymbol{\alpha}}(\mathbf{W}) = \frac{\partial}{\partial \mathbf{W}} t_{\boldsymbol{\alpha}}^{-1}(\mathbf{W}) = \prod_{i=1}^{n_h} \prod_{\ell=1}^{n_h} \frac{\partial t^{-1}_{\alpha_{i,\ell}}(w_{i,\ell})}{\partial w_{i,\ell}}$.

1. Sample $\mathbf{W}$ and $\mathbf{\Gamma}_W$ as follows:

$$[\mathbf{W}, \mathbf{\Gamma}_W \mid \Theta, \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}] = \frac{[\Theta, \mathbf{W}, \mathbf{\Gamma}_W \mid \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}]}{[\Theta \mid \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}]} \tag{A1}$$

$$= \frac{\int_{\mathbf{A}} [\Theta, \mathbf{W}, \mathbf{\Gamma}_W, \boldsymbol{\alpha} \mid \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}] d\boldsymbol{\alpha}}{[\Theta \mid \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}]} \tag{A2}$$

$$= \frac{\int_{\mathbf{A}} [\Theta, \mathbf{W}, \mathbf{\Gamma}_W \mid \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}, \boldsymbol{\alpha}] [\boldsymbol{\alpha} \mid \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}] d\boldsymbol{\alpha}}{[\Theta \mid \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T},]} \tag{A3}$$

$$= \frac{\int_{\mathbf{A}} [\Theta, \mathbf{W}, \mathbf{\Gamma}_W \mid \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}] [\boldsymbol{\alpha}] d\boldsymbol{\alpha}}{[\Theta \mid \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T},]} \tag{A4}$$

$$= \frac{\int_{\mathbf{A}} [\Theta, t_{\boldsymbol{\alpha}}(\mathbf{W}), \mathbf{\Gamma}_W \mid \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}] \, |J_{\boldsymbol{\alpha}}(\mathbf{W})| \, [\boldsymbol{\alpha}] \, d\boldsymbol{\alpha}}{[\Theta \mid \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}]} \tag{A5}$$

$$= \int_{\mathbf{A}} [t_{\boldsymbol{\alpha}}(\mathbf{W}), \mathbf{\Gamma}_W \mid \Theta, \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}] \, |J_{\boldsymbol{\alpha}}(\mathbf{W})| \, [\boldsymbol{\alpha}] \, d\boldsymbol{\alpha}, \tag{A6}$$

As stated in the main text, to sample from this integral, we assume $\mathbf{W}', \mathbf{\Gamma}_W \sim [t_{\boldsymbol{\alpha}}(\mathbf{W}), \mathbf{\Gamma}_W \mid \Theta, \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}]$. We will take $\mathbf{W} = t_{\boldsymbol{\alpha}}^{-1}(\mathbf{W}')$, thus allowing for the joint sampling of $\mathbf{W}$ and $\mathbf{\Gamma}_W$. This result leads to step 1 of Algorithm 1 and defining $\boldsymbol{\alpha}_0$ as the simulated value from $[\boldsymbol{\alpha}]$ leads to step 2. Note the procedure described here closely follows the procedure outlined directly below Equation (1.4.3) in [27]. The assumption stated above that $\boldsymbol{\alpha}$ is only dependent on $\Theta, \mathbf{W}, \mathbf{\Gamma}_W, \widetilde{\mathbf{x}}_{1:T}$ and $\mathbf{Y}_{1:T}$ through the transformation $t_{\boldsymbol{\alpha}}$ is utilized when going from (A3) to (A4).

2. Sample Θ and $\boldsymbol{\alpha}$, as follows:

$$[\Theta, \boldsymbol{\alpha} \mid \mathbf{W}, \mathbf{\Gamma}_W, \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}] = \frac{[\Theta, \boldsymbol{\alpha}, \mathbf{W}, \mathbf{\Gamma}_W, \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}]}{[\mathbf{W}, \mathbf{\Gamma}_W, \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}]} \tag{A7}$$

$$= \frac{[\Theta, \boldsymbol{\alpha}, t_{\boldsymbol{\alpha}_0}^{-1}(\mathbf{W}), \mathbf{\Gamma}_W, \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}] \, |\widetilde{J}_{\boldsymbol{\alpha}_0}(\mathbf{W})|}{[t_{\boldsymbol{\alpha}_0}^{-1}(\mathbf{W}), \mathbf{\Gamma}_W, \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}] \, |\widetilde{J}_{\boldsymbol{\alpha}_0}(\mathbf{W})|} \tag{A8}$$

$$= \frac{[\Theta, \boldsymbol{\alpha}, t_{\boldsymbol{\alpha}_0}^{-1}(\mathbf{W}), \mathbf{\Gamma}_W, \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}]}{[t_{\boldsymbol{\alpha}_0}^{-1}(\mathbf{W}), \mathbf{\Gamma}_W, \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}]} \tag{A9}$$

$$\propto [\Theta, \boldsymbol{\alpha}, \widetilde{\mathbf{W}}, \mathbf{\Gamma}_W, \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}] \tag{A10}$$

$$= [\Theta, \boldsymbol{\alpha}, t_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}}), \mathbf{\Gamma}_W, \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}] \, |J_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}})| \tag{A11}$$

$$\propto [\mathbf{Y}_{1:T} \mid t_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}}), \mathbf{\Gamma}_W, \Theta, \boldsymbol{\alpha}, \widetilde{\mathbf{x}}_{1:T}] \, [\Theta]$$
$$\times [t_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}}) \mid \mathbf{\Gamma}_W, \boldsymbol{\alpha}] \, [\boldsymbol{\alpha}] \, |J_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}})| \tag{A12}$$

Above in (A10), $\widetilde{\mathbf{W}}$ is defined as $\widetilde{\mathbf{W}} \equiv t_{\boldsymbol{\alpha}_0}^{-1}(\mathbf{W})$. Going from (A11) to (A12), we assume $\boldsymbol{\alpha}$ is conditionally independent of $\mathbf{\Gamma}_W$, $t_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}})$ and $\boldsymbol{\alpha}$ are independent of $\widetilde{\mathbf{x}}_{1:T}$, and Θ is conditionally independent of $\boldsymbol{\alpha}, t_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}}), \mathbf{\Gamma}_W$, and $\widetilde{\mathbf{x}}_{1:T}$. Finally, the full-conditional distributions for Θ and $\boldsymbol{\alpha}$ are as follows:

$$[\boldsymbol{\alpha} \mid t_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}}), \mathbf{\Gamma}_W, \Theta, \mathbf{Y}_{1:T}, , \widetilde{\mathbf{x}}_{1:T}] \propto [\mathbf{Y}_{1:T} \mid t_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}}), \mathbf{\Gamma}_W, \Theta, \boldsymbol{\alpha}, \widetilde{\mathbf{x}}_{1:T}]$$
$$\times [t_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}}) \mid \mathbf{\Gamma}_W, \boldsymbol{\alpha}] \, [\boldsymbol{\alpha}] \, |J_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}})| \tag{A13}$$

$$[\Theta \mid t_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}}), \mathbf{\Gamma}_W, \boldsymbol{\alpha}, \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}] \propto [\mathbf{Y}_{1:T} \mid t_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}}), \mathbf{\Gamma}_W, \Theta, \boldsymbol{\alpha}, \widetilde{\mathbf{x}}_{1:T}] \, [\Theta] \tag{A14}$$

$$\propto [\mathbf{Y}_{1:T} \mid t_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}}), \Theta, \widetilde{\mathbf{x}}_{1:T}] \, [\Theta] \tag{A15}$$

These two full conditionals lead directly to steps 4 and 5 of Algorithm 1, respectively, where $\boldsymbol{\alpha}$ is sampled using Metropolis-Hasting steps and Θ is sampled using Gibbs and Metropolis-Hasting steps (see Appendix C).

Algorithm 1 PX-MCMC algorithm.

1. Sample $\mathbf{W}, \boldsymbol{\Gamma}_W$ from: $[\mathbf{W}, \boldsymbol{\Gamma}_W \mid \Theta, \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}] \propto$ $[\mathbf{Y}_{1:T} \mid \Theta, \mathbf{W}, \boldsymbol{\Gamma}_W, \widetilde{\mathbf{x}}_{1:T}]\,[\mathbf{W} \mid \boldsymbol{\Gamma}_W]\,[\boldsymbol{\Gamma}_W]$.
2. Generate $\alpha_{0,i,\ell} \sim \text{Gau}(0, \sigma^2_\alpha)$ for $i = 1, \ldots, n_h$ and $\ell = 1, \ldots, n_h$.
3. Transform $\widetilde{\mathbf{W}} = t^{-1}_{\boldsymbol{\alpha}_0}(\mathbf{W})$.
4. Sample $\boldsymbol{\alpha}$ from: $[\boldsymbol{\alpha} \mid t_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}}), \boldsymbol{\Gamma}_W, \Theta, \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}] \propto$ $[\mathbf{Y}_{1:T} \mid t_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}}), \boldsymbol{\Gamma}_W, \Theta, \boldsymbol{\alpha}, \widetilde{\mathbf{x}}_{1:T}]\,[t_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}}) \mid \boldsymbol{\Gamma}_W, \boldsymbol{\alpha}]\,[\boldsymbol{\alpha}]\,|J_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}})|$.
5. Sample Θ from: $[\Theta \mid t_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}}), \boldsymbol{\Gamma}_W, \boldsymbol{\alpha}, \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}] \propto [\mathbf{Y}_{1:T} \mid t_{\boldsymbol{\alpha}}(\widetilde{\mathbf{W}}), \Theta, \widetilde{\mathbf{x}}_{1:T}]\,[\Theta]$.

Appendix C. Full-Conditionals for the BAST-RNN Model

The full-conditional distributions for all of the parameters in the BAST-RNN model are detailed in this Appendix. To ease the notation we define:

$$\widetilde{\Theta} \equiv \{\boldsymbol{\mu}, \mathbf{V}_1, \mathbf{V}_2, \boldsymbol{\Gamma}_{V_1}, \boldsymbol{\Gamma}_{V_2}, \mathbf{W}, \boldsymbol{\Gamma}_W, \boldsymbol{\alpha}, \boldsymbol{\alpha}_0, \mathbf{U}, \boldsymbol{\Gamma}_U, \delta, \sigma^2_\epsilon\},$$

and borrowing the notational convention from [69], let $\widetilde{\Theta}_{-w_{i,\ell}} = \widetilde{\Theta} \cap \{w_{i,\ell}\}^c$, such that the notation "c" denotes the compliment. Thus, $\widetilde{\Theta}_{-w_{i,\ell}}$ denotes the collection of all of the parameters in $\widetilde{\Theta}$ except for $w_{i,\ell}$. A similar notation can be used for all of the other parameters in the model.

We will use the notation $\Phi(\cdot)$ to denote the cumulative distribution function for the Gaussian distribution. Next, let Υ_k be a $(2n_h + 1) \times (2n_h + 1)$ diagonal matrix with the first diagonal element corresponding to σ^2_μ, the next n_h diagonal entries corresponding to $\gamma^v_{1,k,i}\sigma^2_{v_1,0} + (1 - \gamma^v_{1,k,i})\sigma^2_{v_1,1}$ (for $i = 1, \ldots, n_h$), and the last n_h diagonal entries corresponding to $\gamma^v_{2,k,i}\sigma^2_{v_2,0} + (1 - \gamma^v_{2,k,i})\sigma^2_{v_2,1}$ (for $i = 1, \ldots, n_h$). Probability distribution functions for the Gaussian priors associated with $v_{1,k,i}$ and $v_{2,k,i}$ are denoted by $\phi^{v_1}(\cdot)$ and $\phi^{v_2}(\cdot)$ (as defined in Appendix A), respectively.

Finally, the vector $\widetilde{\mathbf{h}}_t$ is defined as $\widetilde{\mathbf{h}}_t \equiv (1, \mathbf{h}'_t, \mathbf{h}^{2'}_t)'$, such that $\widetilde{\mathbf{h}}_{1:T}$ is a $(2n_h + 1) \times T$ matrix. Throughout, we will let $\mathbf{g}_t \equiv \boldsymbol{\mu} + \mathbf{V}_1\mathbf{h}_t + \mathbf{V}_2\mathbf{h}^2_t$ to reduce the amount of notation. The BAST-RNN model is defined by the following full-conditional distributions:

- $$[w_{i,\ell}, \gamma^w_{i,\ell} \mid \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}, \widetilde{\Theta}_{-\{w_{i,\ell}, \gamma^w_{i,\ell}\}}] \propto \prod_{t=1}^{T} \exp\left(\frac{-(\mathbf{Y}_t - \mathbf{g}_t)'(\mathbf{Y}_t - \mathbf{g}_t)}{2\sigma^2_\epsilon}\right)$$
$$\times \left(\frac{\gamma^w_{i,\ell}\exp\left(\frac{-w^2_{i,\ell}}{2\sigma^2_{w,0}}\right)}{\Phi(\frac{a_w}{\sigma_{w,0}}) - \Phi(\frac{-a_w}{\sigma_{w,0}})} + \frac{(1-\gamma^w_{i,\ell})\exp\left(\frac{-w^2_{i,\ell}}{2\sigma^2_{w,1}}\right)}{\Phi(\frac{a_w}{\sigma_{w,1}}) - \Phi(\frac{-a_w}{\sigma_{w,1}})}\right) \times \left(\pi_w^{\gamma^w_{i,\ell}} + (1 - \pi_w)^{1-\gamma^w_{i,\ell}}\right),$$
for $i = 1, \ldots, n_h$ and $\ell = 1, \ldots, n_h$.
- $$[\alpha_{i,\ell} \mid \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}, \widetilde{\Theta}_{-\alpha_{i,\ell}}] \propto \prod_{t=1}^{T} \exp\left(\frac{-(\mathbf{Y}_t - \mathbf{g}_t)'(\mathbf{Y}_t - \mathbf{g}_t)}{2\sigma^2_\epsilon}\right)$$
$$\times \left(\frac{\gamma^w_{i,\ell}\exp\left(\frac{-\{t_{\alpha_{i,\ell}}(\bar{w}_{i,\ell})\}^2}{2\sigma^2_{w,0}}\right)}{\Phi(\frac{a_w}{\sigma_{w,0}}) - \Phi(\frac{-a_w}{\sigma_{w,0}})} + \frac{(1-\gamma^w_{i,\ell})\exp\left(\frac{-\{t_{\alpha_{i,\ell}}(\bar{w}_{i,\ell})\}^2}{2\sigma^2_{w,1}}\right)}{\Phi(\frac{a_w}{\sigma_{w,1}}) - \Phi(\frac{-a_w}{\sigma_{w,1}})}\right) \times \exp\left(\frac{-\alpha^2_{i,\ell}}{2\sigma^2_\alpha}\right) \times \left(\frac{2a_w\exp(-\tilde{w}_{i,\ell}+\alpha_{i,\ell})}{(1+\exp(-\tilde{w}_{i,\ell}+\alpha_{i,\ell}))^2}\right),$$
for $i = 1, \ldots, n_h$ and $\ell = 1, \ldots, n_h$.
- $$[u_{i,r}, \gamma^u_{i,r} \mid \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}, \widetilde{\Theta}_{-\{u_{i,r}, \gamma^u_{i,r}\}}] \propto \prod_{t=1}^{T} \exp\left(\frac{-(\mathbf{Y}_t - \mathbf{g}_t)'(\mathbf{Y}_t - \mathbf{g}_t)}{2\sigma^2_\epsilon}\right)$$
$$\times \left(\frac{\gamma^u_{i,r}\exp\left(\frac{-u^2_{i,r}}{2\sigma^2_{u,0}}\right)}{\Phi(\frac{a_u}{\sigma_{u,0}}) - \Phi(\frac{-a_u}{\sigma_{u,0}})} + \frac{(1-\gamma^u_{i,r})\exp\left(\frac{-u^2_{i,r}}{2\sigma^2_{u,1}}\right)}{\Phi(\frac{a_u}{\sigma_{u,1}}) - \Phi(\frac{-a_u}{\sigma_{u,1}})}\right) \times \left(\pi_u^{\gamma^u_{i,r}} + (1 - \pi_u)^{1-\gamma^u_{i,r}}\right),$$

for $i = 1, \ldots, n_h$ and $\ell = 1, \ldots, n_h$.

- $[\delta \mid \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}, \widetilde{\Theta}_{-\delta}] \propto \prod_{t=1}^{T} \exp\left(\frac{-(\mathbf{Y}_t - \mathbf{g}_t)'(\mathbf{Y}_t - \mathbf{g}_t)}{2\sigma_\epsilon^2} \right) \times I_{[0,1]}(\delta)$
- $[\mu_{1,k}, \mathbf{V}_{1,k}, \mathbf{V}_{2,k} \mid \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}, \widetilde{\Theta}_{-\{\mu_{1,k}, \mathbf{V}_{1,k}, \mathbf{V}_{2,k}\}}] \propto$
  $\text{Gau}\left(\left(\frac{1}{\sigma^2 \epsilon} \widetilde{\mathbf{h}}_{1:T} \widetilde{\mathbf{h}}'_{1:T} + \Upsilon_k^{-1} \right)^{-1} \frac{1}{\sigma^2 \epsilon} \widetilde{\mathbf{h}}_{1:T} \mathbf{Y}_{1:T,k}, \left(\frac{1}{\sigma^2 \epsilon} \widetilde{\mathbf{h}}_{1:T} \widetilde{\mathbf{h}}'_{1:T} + \Upsilon_k^{-1} \right)^{-1} \right)$,
  for $k = 1, \ldots, n_h$.
- $[\gamma^v_{1,k,i} \mid \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}, \widetilde{\Theta}_{-\{\gamma^v_{1,k,i}\}}] \propto \text{Bernoulli}\left(\frac{\phi^{v_1}(v_{1,k,i} \mid \gamma^v_{1,k,i}=1)}{\phi^{v_1}(v_{1,k,i} \mid \gamma^v_{1,k,i}=1) + \phi^{v_1}(v_{1,k,i} \mid \gamma^v_{1,k,i}=0)} \right)$,
  for $i = 1, \ldots, n_h$ and $k = 1, \ldots, n_y$.
- $[\gamma^v_{2,k,i} \mid \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}, \widetilde{\Theta}_{-\{\gamma^v_{2,k,i}\}}] \propto \text{Bernoulli}\left(\frac{\phi^{v_2}(v_{2,k,i} \mid \gamma^v_{2,k,i}=1)}{\phi^{v_2}(v_{2,k,i} \mid \gamma^v_{2,k,i}=1) + \phi^{v_2}(v_{2,k,i} \mid \gamma^v_{2,k,i}=0)} \right)$,
  for $i = 1, \ldots, n_h$ and $k = 1, \ldots, n_y$.
- $[\sigma_\epsilon^2 \mid \mathbf{Y}_{1:T}, \widetilde{\mathbf{x}}_{1:T}, \widetilde{\Theta}_{-\{\sigma_\epsilon^2\}}] \propto \text{IG}(\frac{Tn_y}{2} + \alpha_\epsilon, \frac{1}{2} \sum_{t=1}^{T} (\mathbf{Y}_t - \mathbf{g}_t)'(\mathbf{Y}_t - \mathbf{g}_t) + \beta_\epsilon)$.

All the parameters in $\widetilde{\Theta}$ are sampled in the order provided above, with $\mathbf{W}, \mathbf{\Gamma}_W, \boldsymbol{\alpha}, \mathbf{U}, \mathbf{\Gamma}_U$, and δ requiring Metropolis-Hasting steps, while $\mathbf{V}_1, \mathbf{V}_2, \boldsymbol{\mu}, \mathbf{\Gamma}_{V_1}, \mathbf{\Gamma}_{V_2}$, and σ_ϵ^2 are sampled with Gibbs steps.

Appendix D. Trace Plots for the BAST-RNN Model

Displayed below are various trace plots from the Lorenz-96 simulated example. The convergence for the other applications (not shown here) was similar.

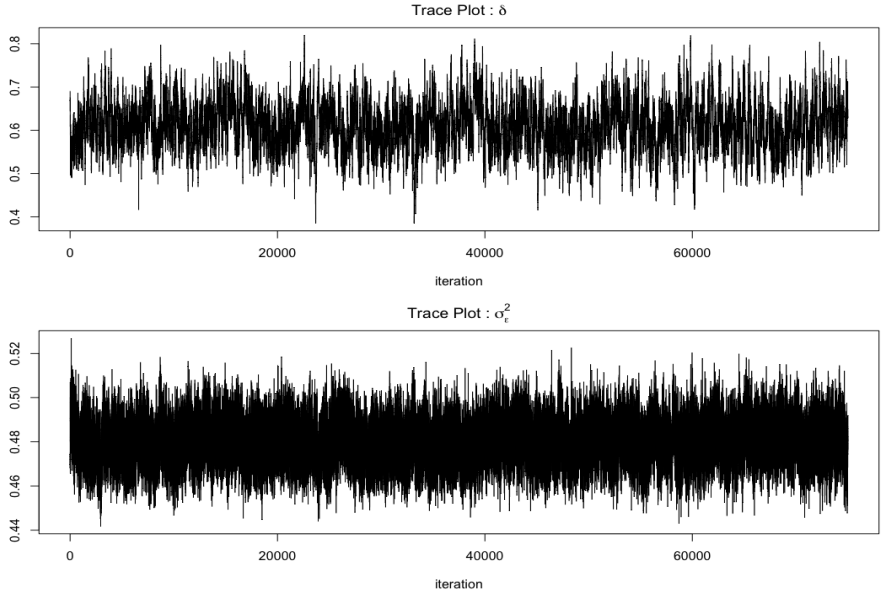

Figure A2. Trace plots for the parameters δ and σ_ϵ^2 for the Lorenz-96 simulated example.

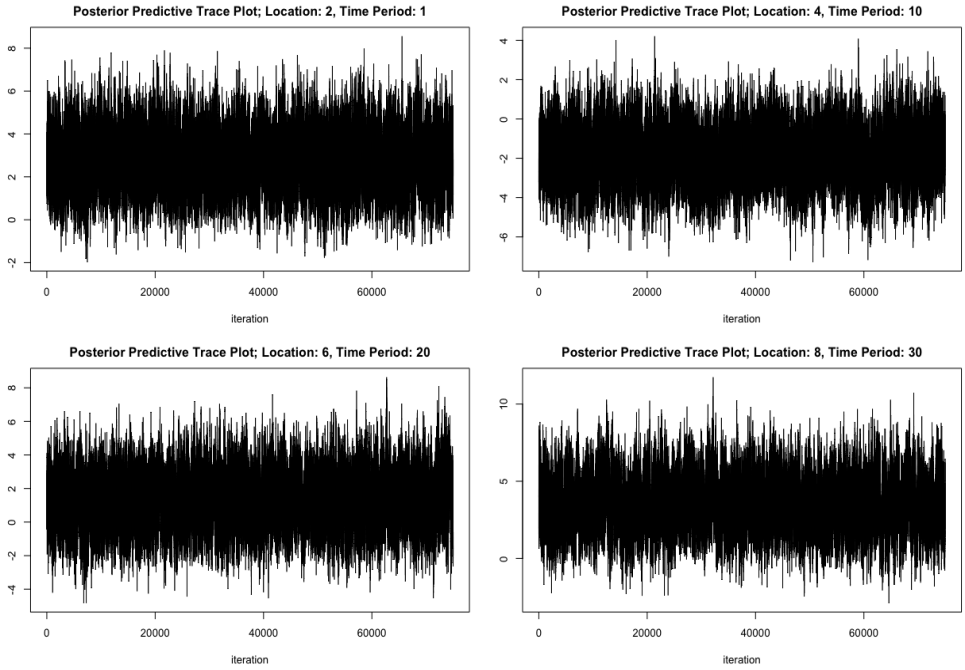

Figure A3. A sample of four trace plots for the posterior predictions from the Lorenz-96 simulated example.

References

1. Fan, J.; Yao, Q. *Nonlinear Time Series*; Springer: Berlin, Germany, 2005.
2. Billings, S.A. *Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains*; John Wiley & Sons: Hoboken, NJ, USA, 2013.
3. Wikle, C. Modern perspectives on statistics for spatio-temporal data. *Wiley Interdiscip. Rev. Comput. Stat.* **2015**, *7*, 86–98. [CrossRef]
4. Berliner, L.M.; Wikle, C.K.; Cressie, N. Long-lead prediction of Pacific SSTs via Bayesian dynamic modeling. *J. Clim.* **2000**, *13*, 3953–3968. [CrossRef]
5. Wu, G.; Holan, S.H.; Wikle, C.K. Hierarchical Bayesian spatio-temporal Conway–Maxwell Poisson models with dynamic dispersion. *J. Agric. Biol. Environ. Stat.* **2013**, *18*, 335–356. [CrossRef]
6. Hooten, M.B.; Wikle, C.K. Statistical agent-based models for discrete spatio-temporal systems. *J. Am. Stat. Assoc.* **2010**, *105*, 236–248. [CrossRef]
7. Wikle, C.K.; Hooten, M.B. A general science-based framework for dynamical spatio-temporal models. *Test* **2010**, *19*, 417–451. [CrossRef]
8. McDermott, P.L.; Wikle, C.K. A model-based approach for analog spatio-temporal dynamic forecasting. *Environmetrics* **2016**, *27*, 70–82. [CrossRef]
9. Richardson, R.A. Sparsity in nonlinear dynamic spatiotemporal models using implied advection. *Environmetrics* **2017**, *28*, e2456. [CrossRef]
10. Cressie, N.; Wikle, C. *Statistics for Spatio-Temporal Data*; John Wiley & Sons: New York, NY, USA, 2011.
11. Tang, B.; Hsieh, W.W.; Monahan, A.H.; Tangang, F.T. Skill comparisons between neural networks and canonical correlation analysis in predicting the equatorial Pacific sea surface temperatures. *J. Clim.* **2000**, *13*, 287–293. [CrossRef]
12. Dixon, M.F.; Polson, N.G.; Sokolov, V.O. Deep Learning for Spatio-Temporal Modeling: Dynamic Traffic Flows and High Frequency Trading. *arXiv* **2017**, arXiv:1705.09851.
13. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
14. Graves, A.; Liwicki, M.; Fernández, S.; Bertolami, R.; Bunke, H.; Schmidhuber, J. A novel connectionist system for unconstrained handwriting recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2009**, *31*, 855–868. [CrossRef] [PubMed]
15. Ning, G.; Zhang, Z.; Huang, C.; He, Z.; Ren, X.; Wang, H. Spatially supervised recurrent convolutional neural networks for visual object tracking. *arXiv* **2016**, arXiv:1607.05781.

16. Yildiz, I.B.; von Kriegstein, K.; Kiebel, S.J. From birdsong to human speech recognition: Bayesian inference on a hierarchy of nonlinear dynamical systems. *PLoS Comput. Biol.* **2013**, *9*, e1003219. [CrossRef] [PubMed]
17. Graves, A. Generating sequences with recurrent neural networks. *arXiv* **2013**, arXiv:1308.0850.
18. Jaeger, H. *The "Echo State" Approach to Analysing and Training Recurrent Neural Networks-with an Erratum Note*; German National Research Center for Information Technology GMD Technical Report: Bonn, Germany, 2001; Volume 148.
19. Lukoševičius, M.; Jaeger, H. Reservoir computing approaches to recurrent neural network training. *Comput. Sci. Rev.* **2009**, *3*, 127–149. [CrossRef]
20. McDermott, P.L.; Wikle, C.K. An Ensemble Quadratic Echo State Network for Nonlinear Spatio-Temporal Forecasting. *STAT* **2017**, *6*, 315–330. [CrossRef]
21. van der Westhuizen, J.; Lasenby, J. Bayesian LSTMs in medicine. *arXiv* **2017**, arXiv:1706.01242.
22. Neal, R.M. Bayesian Learning for Neural Networks. Ph.D. Thesis, University of Toronto, Toronto, ON, Canada, 1994.
23. Chatzis, S.P. Sparse Bayesian Recurrent Neural Networks. In *Machine Learning and Knowledge Discovery in Databases*; Springer: Berlin, Germany, 2015; pp. 359–372.
24. Chien, J.T.; Ku, Y.C. Bayesian recurrent neural network for language modeling. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *27*, 361–374. [CrossRef] [PubMed]
25. Gan, Z.; Li, C.; Chen, C.; Pu, Y.; Su, Q.; Carin, L. Scalable Bayesian Learning of Recurrent Neural Networks for Language Modeling. *arXiv* **2016**, arXiv:1611.08034.
26. Liu, J.S.; Wu, Y.N. Parameter expansion for data augmentation. *J. Am. Stat. Assoc.* **1999**, *94*, 1264–1274. [CrossRef]
27. Hobert, J.P.; Marchev, D. A theoretical comparison of the data augmentation, marginal augmentation and PX-DA algorithms. *Ann. Stat.* **2008**, *36*, 532–554. [CrossRef]
28. Hobert, J.P. The data augmentation algorithm: Theory and methodology. *Handbook of Markov Chain Monte Carlo*; Chapman & Hall/CRC: London, UK, 2011; pp. 253–293.
29. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Gated feedback recurrent neural networks. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 Junly 2015; pp. 2067–2075.
30. Takens, F. Detecting strange attractors in turbulence. *Lect. Notes Math.* **1981**, *898*, 366–381.
31. Srivastava, N.; Hinton, G.E.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
32. Polson, N.; Sokolov, V. Deep Learning: A Bayesian Perspective. *Bayesian Anal.* **2017**, *12*, 1275–1304. [CrossRef]
33. MacKay, D.J. A practical Bayesian framework for backpropagation networks. *Neural Comput.* **1992**, *4*, 448–472. [CrossRef]
34. O'Hara, R.B.; Sillanpää, M.J. A review of Bayesian variable selection methods: What, how and which. *Bayesian Anal.* **2009**, *4*, 85–117. [CrossRef]
35. George, E.I.; McCulloch, R.E. Variable selection via Gibbs sampling. *J. Am. Stat. Assoc.* **1993**, *88*, 881–889. [CrossRef]
36. George, E.I.; McCulloch, R.E. Approaches for Bayesian variable selection. *Stat. Sin.* **1997**, *7*, 339–373.
37. Ghosh, M.; Maiti, T.; Kim, D.; Chakraborty, S.; Tewari, A. Hierarchical Bayesian neural networks: An application to a prostate cancer study. *J. Am. Stat. Assoc.* **2004**, *99*, 601–608. [CrossRef]
38. Park, T.; Casella, G. The bayesian lasso. *J. Am. Stat. Assoc.* **2008**, *103*, 681–686. [CrossRef]
39. Carvalho, C.M.; Polson, N.G.; Scott, J.G. The horseshoe estimator for sparse signals. *Biometrika* **2010**, *97*, 465–480. [CrossRef]
40. Ročková, V.; George, E.I. The spike-and-slab lasso. *J. Am. Stat. Assoc.* **2018**, *113*, 431–444. [CrossRef]
41. Belkin, M.; Niyogi, P. Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering. In Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic (NIPS'01), Vancouver, BC, Canada, 3–8 December 2001; pp. 585–591.
42. Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), Haifa, Israel, 21–24 June 2010; pp. 807–814.
43. Coifman, R.; Lafon, S. Diffusion maps. *Appl. Comput. Harmon. Anal.* **2006**, *21*, 5–30. [CrossRef]
44. Matheson, J.E.; Winkler, R.L. Scoring rules for continuous probability distributions. *Manag. Sci.* **1976**, *10*, 1087–1096. [CrossRef]

45. Gneiting, T.; Katzfuss, M. Probabilistic forecasting. *Annu. Rev. Stat. Appl.* **2014**, *1*, 125–151. [CrossRef]
46. Majda, A.J.; Timofeyev, I.; Vanden-Eijnden, E. Systematic strategies for stochastic mode reduction in climate. *J. Atmos. Sci.* **2003**, *60*, 1705–1722. [CrossRef]
47. Kravtsov, S.; Kondrashov, D.; Ghil, M. Multilevel regression modeling of nonlinear processes: Derivation and applications to climatic variability. *J. Clim.* **2005**, *18*, 4404–4424. [CrossRef]
48. Green, P.J. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika* **1995**, *82*, 711–732. [CrossRef]
49. Lukoševičius, M. A practical guide to applying echo state networks. In *Neural Networks: Tricks of the Trade*; Springer: Berlin, Germany, 2012; pp. 659–686.
50. Lorenz, E.N. Deterministic nonperiodic flow. *J. Atmos. Sci.* **1963**, *20*, 130–141. [CrossRef]
51. Ma, Q.L.; Zheng, Q.L.; Peng, H.; Zhong, T.W.; Xu, L.Q. Chaotic time series prediction based on evolving recurrent neural networks. In Proceedings of the 2007 International Conference on Machine Learning and Cybernetics, Hong Kong, China, 19–22 August 2007; Volume 6, pp. 3496–3500.
52. Chandra, R.; Zhang, M. Cooperative coevolution of Elman recurrent neural networks for chaotic time series prediction. *Neurocomputing* **2012**, *86*, 116–123. [CrossRef]
53. Lorenz, E.N. Predictability: A problem partly solved. In Proceedings of the Seminar on Predictability, Reading, UK, 4–8 September 1995; Volume 1.
54. Wilks, D.S. Effects of stochastic parametrizations in the Lorenz'96 system. *Quart. J. R. Meteorol. Soc.* **2005**, *131*, 389–407. [CrossRef]
55. Chorin, A.J.; Lu, F. Discrete approach to stochastic parametrization and dimension reduction in nonlinear dynamics. *Proc. Natl. Acad. Sci. USA* **2015**, *112*, 9804–9809. [CrossRef]
56. Grooms, I.; Lee, Y. A framework for variational data assimilation with superparameterization. *Nonlinear Processes Geophys.* **2015**, *22*, 601–611. [CrossRef]
57. Hu, S.; Fedorov, A.V. The extreme El Niño of 2015–2016: The role of westerly and easterly wind bursts, and preconditioning by the failed 2014 event. *Clim. Dyn.* **2017**, 1–19. [CrossRef]
58. L'Heureux, M.L.; Takahashi, K.; Watkins, A.B.; Barnston, A.G.; Becker, E.J.; Di Liberto, T.E.; Gamble, F.; Gottschalck, J.; Halpert, M.S.; Huang, B.; et al. Observing and predicting the 2015-16 El Niño. *Bull. Am. Meteorol. Soc.* **2017**, *98*, 1363–1382. [CrossRef]
59. Barnston, A.G.; Tippett, M.K.; L'Heureux, M.L.; Li, S.; DeWitt, D.G. Skill of real-time seasonal ENSO model predictions during 2002–2011: Is our capability increasing? *Bull. Am. Meteorol. Soc.* **2012**, *93*, 631–651. [CrossRef]
60. Barnston, A.G.; He, Y.; Glantz, M.H. Predictive skill of statistical and dynamical climate models in SST forecasts during the 1997–1998 El Niño episode and the 1998 La Niña onset. *Bull. Am. Meteorol. Soc.* **1999**, *80*, 217–243. [CrossRef]
61. Jan van Oldenborgh, G.; Balmaseda, M.A.; Ferranti, L.; Stockdale, T.N.; Anderson, D.L. Did the ECMWF seasonal forecast model outperform statistical ENSO forecast models over the last 15 years? *J. Clim.* **2005**, *18*, 3240–3249. [CrossRef]
62. Tangang, F.T.; Tang, B.; Monahan, A.H.; Hsieh, W.W. Forecasting ENSO events: A neural network–extended EOF approach. *J. Clim.* **1998**, *11*, 29–41. [CrossRef]
63. Gladish, D.W.; Wikle, C.K. Physically motivated scale interaction parameterization in reduced rank quadratic nonlinear dynamic spatio-temporal models. *Environmetrics* **2014**, *25*, 230–244. [CrossRef]
64. Liang, F. Bayesian neural networks for nonlinear time series forecasting. *Stat. Comput.* **2005**, *15*, 13–29. [CrossRef]
65. Sharma, S.; Singh, S. Unemployment rates forecasting using supervised neural networks. In Proceedings of the 2016 6th International Conference Cloud System and Big Data Engineering (Confluence), Noida, India, 14–15 January 2016; pp. 28–33.
66. Teräsvirta, T.; Van Dijk, D.; Medeiros, M.C. Linear models, smooth transition autoregressions, and neural networks for forecasting macroeconomic time series: A re-examination. *Int. J. Forecast.* **2005**, *21*, 755–774. [CrossRef]
67. Jones, N.A.; Smith, A.S. *The Two or More Races Population, 2000*; US Department of Commerce, Economics and Statistics Administration, US Census Bureau: Washington, DC, USA, 2001; Volume 8.

68. Welling, M.; Teh, Y.W. Bayesian learning via stochastic gradient Langevin dynamics. In Proceedings of the 28th International Conference on Machine Learning (ICML-11), Bellevue, WA, USA, 28 June–2 July 2011; pp. 681–688.
69. Bradley, J.R.; Wikle, C.K.; Holan, S.H. Bayesian spatial change of support for count-valued survey data with application to the american community survey. *J. Am. Stat. Assoc.* **2016**, *111*, 472–487. [CrossRef]