TRustworthy Uncertainty Propagation for Sequential Time-Series Analysis in RNNs

Dimah Dera, Sabeen Ahmed, Nidhal C. Bouaynaya, and Ghulam Rasool,

Abstract—The massive time-series production through the Internet of Things and digital healthcare requires novel data modeling and prediction. Recurrent neural networks (RNNs) are extensively used for analyzing time-series data. However, these models are unable to assess prediction uncertainty, which is particularly critical in heterogeneous and noisy environments. Bayesian inference allows reasoning about predictive uncertainty by estimating the posterior distribution of the parameters. The challenge remains in propagating the high-dimensional distribution through the sequential, non-linear layers of RNNs, resulting in mode collapse leading to erroneous uncertainty estimation and exacerbating the gradient explosion problem. This paper proposes a TRustworthy Uncertainty propagation for Sequential Time-series analysis (TRUST) in RNNs by introducing a Gaussian prior over network parameters and estimating the first two moments of the Gaussian variational distribution using the evidence lower bound. We propagate the variational moments through the sequential, non-linear layers of RNNs using the first-order Taylor approximation. The propagated covariance of the predictive distribution captures uncertainty in the output decision. The extensive experiments using ECG5000 and PeMS-SF classification and weather and power consumption prediction tasks demonstrate 1) significant robustness of TRUST-RNNs against noise and adversarial attacks and 2) self-assessment through the uncertainty that increases significantly with increasing noise.

Index Terms—Recurrent neural networks, uncertainty propagation, variational inference, gated recurrent units and long short-term memory networks.

1 Introduction

Time-series analysis and prediction are essential areas of machine learning (ML). Many critical applications rely on time-series analysis, including earthquake prediction, economic forecasting and healthcare monitoring. Recurrent neural networks (RNNs) are specialized artificial neural networks designed to process and learn from sequential and time-series data [1], [2]. Popular variants of RNNs include long short-term memory (LSTM) and gated recurrent unit (GRU) networks [3], [4], [5]. LSTMs and GRUs employ multiple gates in their architecture to control the information flow and overcome the vanishing gradients problem of traditional RNNs [4]. These models have shown remarkable success in dealing with time dependencies of the sequential data and have achieved promising performance in various time-series classification and prediction tasks [1], [2], [3], [5]. Applications that demonstrate the success of LSTMs and GRUs include power consumption estimation [6], [7], weather forecasting [8], healthcare diagnoses and disease prognoses [9], [10].

RNNs, like other traditional artificial neural networks, use training data to learn point estimates of network parameters by minimizing a loss function—a measure of discrepancy between a ground truth and model predictions. During testing, the learned network's parameters are used to provide deterministic predictions for any new data examples. RNN architectures do not provide an estimation of uncertainty (or confidence) in the learnable parameters or model predictions. However, acknowledging the level of uncertainty in the model's parameters and predictions is critical for high-stake applications, e.g., financial prediction, legal decision-making and medical diagnosis. Consider examples of detecting heart failure or life-threatening arrhythmias by analyzing the electrocardiogram (ECG) signal [10] or monitoring smart grids, gas pipelines or aircraft engines [11], [12], [13]. Missing a prediction due to artifacts in the time-series signals could endanger human lives and significantly impact productivity. Uncertainty in models' decisions may serve as a warning and allow users to explore alternative solutions, such as recommending additional diagnostic tests and thus preventing tragic health, financial or societal damage due to highly uncertain decisions.

Quantifying uncertainty in model predictions can justify the behavior of a model under input data distribution shift (due to predictions in noisy environments) and improves the generalization on the out-of-distribution inputs [14]. Moreover, the vulnerability of neural networks to adversarial attacks—i.e., crafted imperceptible (to humans) noise that misleads ML algorithms to make erroneous output—has raised concerns and even halted the deployment of RNNs and their variants in healthcare or safety-critical applications [15], [16], [17]. The uncertainty or model confidence provides valuable information to users for detecting and tackling adversarial attacks [14].

Bayesian formulation facilitates a mathematically grounded approach for estimating uncertainty in deep neural networks (DNNs) [14]. In the Bayesian settings, we define a prior distribution over the network parameters and estimate their posterior distribution using Bayes' rule after observing the training data. The predictive distribution of any new data example can then be computed by marginalizing out the network parameters. The variance (or variance-covariance matrix in the multivariate case) of the predictive distribution provides a quantitative measure of uncertainty associated with the model's prediction. However, due to the non-linear structure of DNNs and the high dimensionality

- *Dimah Dera is with the Chester F. Carlson Center for Imaging Science, Rochester Institute of Technology, NY 14623. E-mail: dimah.dera@ieee.org.*
- *Sabeen Ahmed is with the Electrical Engineering Department, the University of South Florida, FL 33620. E-mail: ahmeds5@usf.edu.*
- *Nidhal C. Bouaynaya is with the Electrical and Computer Engineering Department, Rowan University, NJ 08028. E-mail: bouaynaya@rowan.edu.*
- *Ghulam Rasool is with the Machine Learning Department, Moffit Cancer Center, FL 33612. Email: ghulam.rasool@moffitt.org.*

of the parameter space, the exact Bayesian inference on the parameters is intractable. Variational inference is an effective and extensively used method in the literature for approximating the posterior distribution of DNNs' parameters [18], [19], [20], [21].

The variational inference turns the probability density estimation problem into an optimization problem by approximating the unknown true posterior with a parameterized distribution, which is generally a Gaussian distribution, by minimizing the Kullback-Leibler (KL) divergence between the approximate and the true distribution. The objective function is known as the evidence lower bound (ELBO) [14], [18], [19], [20], [21], [22], [23], [24], [25]. The recent state-of-the-art Bayesian DNNs that used variational inference for estimating uncertainty focused on fully connected and convolutional neural networks (CNNs), such as Bayes-by-backprop (BBB) [18], Bayes-CNN [19], Dropout-CNN [20] and PremiUm-CNN [14]. Quantifying uncertainty in RNNs using Bayesian deep learning has endured multiple challenges, including propagating the variational distribution through the recurrent architecture of RNNs, which may result in 1) mode collapse and catastrophic variance underestimation, which lead to erroneous estimation of the uncertainty [26], [27]; and 2) exacerbation of the well-known gradient explosion problem in RNNs [28], [29].

1.1 Our Contributions

In this paper, we propose new sequential machine learning models (i.e., new Bayesian recurrent neural networks) that are robust to artifacts, noise, and adversarial attacks. These models can quantify uncertainty in the predictions and self-assess their performance. We start by defining a Gaussian distribution as a prior over RNN parameters. Later, the variational posterior distribution is estimated by minimizing the evidence lower bound (ELBO) objective function. In the proposed Bayesian framework, the learnable network parameters are random variables defined with a probability distribution function. Therefore, all operations at each layer of an RNN, LSTM or GRU are derived, considering the parameters as random variables. These operations include (1) inner products between two random vectors; (2) Hadamard products between two random vectors; and (3) non-linear transformations operating over random vectors, such as hyperbolic tangent (Tanh), sigmoid and rectified Linear Unit (ReLU). We use the first-order Taylor series to approximate the mean and covariance matrix of the variational distribution after the non-linear activation functions. At the network's output, the mean vector represents the prediction or classification decision, and the variance-covariance matrix reflects the uncertainty associated with the output decision. We evaluate the models' performance for sequential time-series analysis. More specifically, the contributions are summarized as follows.

1) Introduce TRustworthy Uncertainty propagation for Sequential Time-series analysis (TRUST) framework for RNNs and their variants, including LSTMs and GRUs. We adopt powerful and computationally efficient statistical frameworks from sequential Bayesian estimation used for tracking probability distributions in non-linear dynamical systems [30].
2) Establish the mathematical foundations for propagating the first two moments (mean and covariance matrix) of the variational probability distribution through non-linear layers of various RNN, LSTM, and GRU architectures for analyzing time-series data and quantifying uncertainty in the networks' predictions.
3) Perform an extensive performance evaluation and analysis on a variety of benchmark supervised classification and prediction time-series datasets. Our experiments include weather and power consumption as prediction tasks [31], [32], and PeMS-SF and ECG5000 datasets as classification tasks [33].
4) Demonstrate significant robustness of the proposed TRUST models (compared to the state-of-the-art Bayesian and deterministic models) against random noise and adversarial attacks. The TRUST robustness is achieved without significantly increasing the number of parameters or computational cost at the inference time.
5) Exploit the uncertainty information at the TRUST models output (in the form of the variance-covariance matrix of the predictive distribution) to self-assess the performance degradation and failure under noisy conditions and adversarial attacks.

The remainder of this paper is organized as follows. Section 2 presents a review of the related work in the area of Bayesian RNNs. Section 3 explains the proposed TRUST framework in detail for RNNs, LSTMs and GRUs. Section 4 elaborates on the experimental settings used to evaluate the proposed TRUST models compared to the state-of-the-art Bayesian and deterministic homologs. Experimental results are presented and discussed in Section 5. Finally, we conclude this paper in Section 6.

2 Related Work

Recently, Fortunato *et al*. extended BBB to RNNs (BBB-RNNs) by introducing a fully factorized Gaussian distribution over network parameters and formulating the ELBO objective function for a truncated sequence of an unrolled RNN [21]. The authors introduced a hierarchical posterior distribution over the parameters to allow the networks to adapt locally to batches of data. Gal and Ghahramani extended dropout-CNN [22] formulation to RNNs, termed variational RNNs or VAR-RNNs [23]. The dropout in RNNs was implemented by removing the same recurrent network units at each time step and randomly eliminating inputs, outputs, and recurrent connections. Molchanov *et al*. applied sparse variational dropout (SparseVD) to RNNs with unbounded dropout rates as a Bayesian compression approach that induced sparsity during the training of RNNs [34]. Lobacheva *et al*. extended SparseVD to GRU and LSTM models [35]. Later, Goel and Bajpai used VAR-LSTM to quantify uncertainty in forecasting global sales in hotel businesses [36], and Zhu and Laptev used the same network for time-series anomaly detection at Uber [37]. Gan *et al*. applied stochastic gradient Markov Chain Monte Carlo to learn uncertainty in RNNs' weights by adding gradient noise during training and model averaging when testing [38]. Rangapuram *et al*. introduced a probabilistic time series forecasting approach that combined linear state space models with a parametrized learned recurrent neural network. The forecast distribution was presented in terms of Monte Carlo samples [39]. Salinas *et al*. proposed a DeepAR (DAR) model that made probabilistic forecasts using Monte Carlo samples to compute consistent quantile estimates for all sub-ranges in the prediction horizon [40]. DAR model performed a univariate forecast using an encoder-decoder LSTM architecture. The decoder consists of a fully-connected layer with two outputs, one for the mean and one for the standard deviation, followed by a softplus activation function. DAR model parameters were learned by optimizing a log-likelihood loss function [40].

These state-of-the-art uncertainty quantification techniques in RNNs and their variants rely on a frequentist probability approach or adopt dropout to quantify uncertainty [21], [23], [36], [37], [38]. They follow Monte Carlo (MC) sampling by drawing one

random sample from the variational distribution and passing it forward through the network layers. The moments of the variational distribution are generally *not propagated* through recurrent layers or various operations within the RNN network. At test time, the uncertainty in the model prediction is always estimated by performing multiple passes (MC sampling) through the trained model and computing the sample variance of the predictions. Moreover, such measures of uncertainty in the model output have not been extensively analyzed under noisy conditions, out-of-distribution inputs, or adversarial attacks, nor there was a variance-covariance versus Signal-to-Noise-Ratio (SNR) analysis to evaluate the proposed measure of uncertainty.

There are other methods in the literature for estimating confidence intervals of point predictions, including ensemble and frequentist approaches [41], [42], [43], [44]. These ensemble and frequentist RNNs are built by creating multiple perturbed versions of the original RNN by re-sampling the optimal parameters. These methods are usually post-hoc, that is, the RNNs are trained using deterministic settings, ignoring the computation of uncertainty or confidence during the training phase. We direct the reader to survey articles for further details on estimating confidence intervals of point predictions [45], [46].

3 Proposed Method

3.1 Mathematical Foundations of TRUST Models

This section provides the mathematical foundations of TRUST models, including RNNs, LSTMs and GRUs. We adopt variational inference and propagate the first two moments, i.e., the mean vector and the variance-covariance matrix of the (multivariate) variational distribution $q_\phi(\mathbf{\Omega})$ through RNN layers, including non-linear activation functions. We derive mathematical relations for various operations within one cell of an RNN and extend the framework to LSTM and GRU cells. By propagating the mean and the covariance of the variational distribution, we obtain the mean and the covariance of the predictive distribution $p(\mathbf{y}^*|\mathbf{X}^*, \mathcal{D})$ at the network output for any test sample $(\mathbf{X}^*, \mathbf{y}^*)$. The mean of $p(\mathbf{y}^*|\mathbf{X}^*, \mathcal{D})$ represents the network's prediction, while the covariance matrix reflects the uncertainty in the prediction.

3.2 Bayesian Approximation and Variational Inference

We consider an RNN with L stacked layers assuming the parameters (weights and biases) are random variables and are shared across recurrent states. The parameters are represented by $\mathbf{\Omega} = \{\mathcal{W}^{(l)}\}_{l=1}^{L}$, with biases augmented in weight matrices. The training dataset $\mathcal{D}$ is a set of N sequences, $\mathcal{D} = \{\mathbf{X}^{(n)}, \mathbf{y}^{(n)}\}_{n=1}^{N}$, where $\mathbf{X}^{(n)} \in \mathbb{R}^{\tau \times K}$ with τ denoting the length of the sequences and K is the input vector size (number of features) and $\mathbf{y}^{(n)} \in \mathbb{R}^{C}$ with C representing the number of classes for classification tasks or the number of output neurons for prediction tasks.

We introduce a Gaussian distribution as a prior probability distribution over the network parameters $\mathbf{\Omega} \sim p(\mathbf{\Omega})$. We assume the parameters are independent across layers to 1) extract uncorrelated features across different network layers and 2) develop a feasible optimization problem, as estimating the joint distribution of all layers is mathematically intractable in large ML models. Given the training sequences $\mathcal{D}$ and the prior distribution $p(\mathbf{\Omega})$, the estimation of the posterior distribution $p(\mathbf{\Omega}|\mathcal{D})$ is typically intractable. The variational inference approximates the true posterior $p(\mathbf{\Omega}|\mathcal{D})$ with a parametrized variational distribution $q_\phi(\mathbf{\Omega})$. The optimization problem is formulated by minimizing the Kullback-Leibler (KL) divergence, $\mathrm{KL}\left[q_\phi(\mathbf{\Omega})\|p(\mathbf{\Omega}|\mathcal{D})\right]$, which is equivalent to optimizing the evidence lower bound (ELBO) objective (or loss) function $\mathfrak{L}(\phi; \mathcal{D})$ using gradient decent update rule during training of the Bayesian neural network [47],

$$\mathfrak{L}(\phi; \mathcal{D}) = -E_{q_\phi(\mathbf{\Omega})}\left\{\log p(\mathcal{D}|\mathbf{\Omega})\right\} + \mathrm{KL}\left[q_\phi(\mathbf{\Omega})\|p(\mathbf{\Omega})\right]. \quad (1)$$

The ELBO loss function consists of two parts: 1) the negative expected log-likelihood of the training data given the network parameters; and 2) the regularization term, which is defined by the KL-divergence between the proposed variational distribution $q_\phi(\mathbf{\Omega})$ and the prior distribution $p(\mathbf{\Omega})$.

3.3 Variational Moments Propagation in TRUST-RNNs

The operations in an RNN cell include a matrix-vector multiplication followed by an element-wise non-linear activation function, e.g., hyperbolic tangent (Tanh). Fig. 1 illustrates the propagation of moments of the variational distribution through the recurrent states with an expanded view of operations performed within one RNN cell. The current state, $\mathbf{s}^{(t)}$ at time t, is given by:

$$\mathbf{s}^{(t)} = \psi(\mathbf{U}\mathbf{x}^{(t)} + \mathbf{W}\mathbf{s}^{(t-1)}), \quad (2)$$

where $\mathbf{x}^{(t)} \in \mathbb{R}^{K\times 1}$ is the input vector at the time step t, $\mathbf{U} \in \mathbb{R}^{H\times K}$ is the input-hidden weight matrix, $\mathbf{W} \in \mathbb{R}^{H\times H}$ is the recurrent weight matrix, $\mathbf{s}^{(t-1)} \in \mathbb{R}^{H\times 1}$ is the hidden state at time $t-1$, H is the number of hidden units and ψ is the non-linear activation function (Tanh in this case).

We concatenate the matrices $\mathbf{U}$ and $\mathbf{W}$ as one large matrix, $\boldsymbol{\mathcal{W}} = \begin{bmatrix}\mathbf{U} & \mathbf{W}\end{bmatrix}$. Similarly, the vectors $\mathbf{x}^{(t)}$ and $\mathbf{s}^{(t-1)}$ are concatenated as a column vector, $\tilde{\mathbf{x}} = \begin{bmatrix}\mathbf{x}^{(t)} & \mathbf{s}^{(t-1)}\end{bmatrix}^T$, where T represents the transpose operation. We eliminate the super-script t from $\tilde{\mathbf{x}}$ to simplify the notations. Equation (2) is re-written as:

$$\mathbf{s}^{(t)} = \psi(\mathbf{b}), \quad \text{where} \quad \mathbf{b} = \boldsymbol{\mathcal{W}}\,\tilde{\mathbf{x}}. \quad (3)$$

Let $\mathbf{w}_i^T \in \mathbb{R}^{1\times(K+H)}$ be the i^{th} row of the matrix $\boldsymbol{\mathcal{W}}$, where $i = 1, \ldots, K+H$. We introduce a prior, i.e., Gaussian distribution,

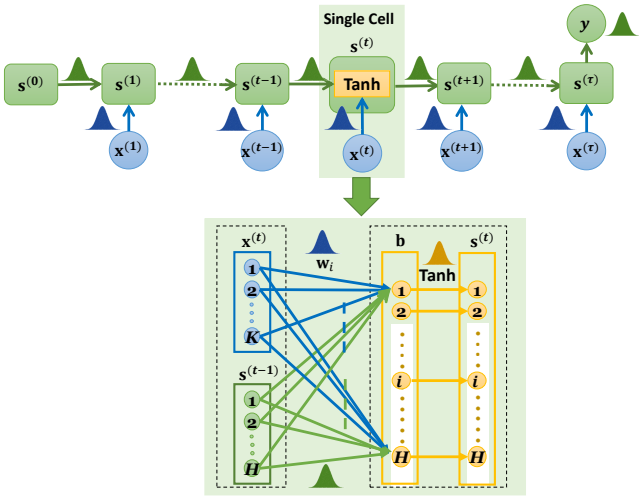


Fig. 1. A schematic layout of the proposed TRUST-RNN with the variational moments propagation (represented by the bell-shaped distribution). The bottom box shows an expanded view of operations performed inside a single RNN cell. The blue, green, and yellow colors represent the input, hidden state, and the non-linear activation function, respectively. In the proposed settings, the weights $\mathbf{w}_i$, the recurrent states $\mathbf{s}^{(t)}$, the output of the inner product $\mathbf{b}$ (between weights and the concatenated state and input) are all random vectors.

over the weight vector $\mathbf{w}_i$. The variational posterior will also be a Gaussian distribution, $\mathbf{w}_i \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{w}_i}, \boldsymbol{\Sigma}_{\mathbf{w}_i})$ where the mean and covariance are estimated by optimizing the ELBO objective function (1) during training of the network. The input $\mathbf{x}^{(t)}$ and the state $\mathbf{s}^{(t)}$ are assumed to be mutually independent random vectors with the mean and covariance defined as $\mathbf{x}^{(t)} \sim (\boldsymbol{\mu}_{\mathbf{x}^{(t)}}, \boldsymbol{\Sigma}_{\mathbf{x}^{(t)}})$ and $\mathbf{s}^{(t-1)} \sim (\boldsymbol{\mu}_{\mathbf{s}^{(t-1)}}, \boldsymbol{\Sigma}_{\mathbf{s}^{(t-1)}})$, respectively. Although $\mathbf{x}^{(t)}$ and $\mathbf{s}^{(t)}$ may not necessarily follow Gaussian distributions, we assume that their distribution functions can be approximated by a mean vector and a variance-covariance matrix. We further assume that the weight vectors $\mathbf{w}_i$ are independent of each other and also independent of the input $\mathbf{x}^{(t)}$ and the state $\mathbf{s}^{(t)}$. The mean and the covariance of the concatenated random vector $\tilde{\mathbf{x}}$ are then given by:

$$\boldsymbol{\mu}_{\tilde{\mathbf{x}}} = \begin{bmatrix} \boldsymbol{\mu}_{\mathbf{x}^{(t)}} & \boldsymbol{\mu}_{\mathbf{s}^{(t-1)}} \end{bmatrix}^T, \qquad \boldsymbol{\Sigma}_{\tilde{\mathbf{x}}} = \begin{bmatrix} \boldsymbol{\Sigma}_{\mathbf{x}^{(t)}} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_{\mathbf{s}^{(t-1)}} \end{bmatrix}. \quad (4)$$

Every element of $\mathbf{b}$ in (3) is the result of an inner product between two independent random vectors $\mathbf{w}_i$ and $\tilde{\mathbf{x}}$, that is, $b_i = \mathbf{w}_i^T \tilde{\mathbf{x}}$. The mean and the covariance of the resulting random vector $\mathbf{b}$ are derived using Proposition 1. The proof of Proposition 1 is in Appendix A in the supplementary materials.

Proposition 1. (mean and covariance propagation through an inner product of two independent random vectors)

$$\mu_{b_i} = \boldsymbol{\mu}_{\mathbf{w}_i}^T \boldsymbol{\mu}_{\tilde{\mathbf{x}}}, \quad (5)$$

$$\boldsymbol{\Sigma}_{\mathbf{b}} = \begin{cases} \mathrm{tr}\left(\boldsymbol{\Sigma}_{\mathbf{w}_i} \boldsymbol{\Sigma}_{\tilde{\mathbf{x}}}\right) + \boldsymbol{\mu}_{\mathbf{w}_i}^T \boldsymbol{\Sigma}_{\tilde{\mathbf{x}}} \boldsymbol{\mu}_{\mathbf{w}_j} + \boldsymbol{\mu}_{\tilde{\mathbf{x}}}^T \boldsymbol{\Sigma}_{\mathbf{w}_i} \boldsymbol{\mu}_{\tilde{\mathbf{x}}}, & i = j \\ \boldsymbol{\mu}_{\mathbf{w}_i}^T \boldsymbol{\Sigma}_{\tilde{\mathbf{x}}} \boldsymbol{\mu}_{\mathbf{w}_j}^T, & i \neq j \end{cases}$$

where $i, j = 1, \cdots, K+H$, and tr represents the trace operator.

Non-linear Activation Function: The non-linear activation function ψ operates element-wise on $\mathbf{b}$. The mean and the covariance at the output of ψ are approximated using the first-order Taylor series [48] as:

$$\boldsymbol{\mu}_{\mathbf{s}^{(t)}} \approx \psi(\boldsymbol{\mu}_{\mathbf{b}}), \quad \boldsymbol{\Sigma}_{\mathbf{s}^{(t)}} \approx \boldsymbol{\Sigma}_{\mathbf{b}} \odot \left(\nabla\psi(\boldsymbol{\mu}_{\mathbf{b}})\, \nabla\psi(\boldsymbol{\mu}_{\mathbf{b}})^T\right), \quad (6)$$

where ∇ represents the gradient of the function ψ with respect to $\mathbf{b}$ and $\odot$ is the Hadamard product. The Equations in (6) hold true for any non-linear activation function that operates element-wise, including Tanh, sigmoid, or rectified Linear Unit (ReLU).

3.4 Variational Moments Propagation in TRUST-LSTMs

3.4.1 The Structure of an LSTM

The LSTM network was introduced by [49] to overcome the vanishing gradient problem in RNNs. An LSTM cell consists of four gates, i.e., input, forget, output, and gate gates and an additional state, referred to as the cell state or memory cell $\mathbf{c}^{(t)}$. The four gates, along with the two states ($\mathbf{c}^{(t)}$ and $\mathbf{s}^{(t)}$), control the flow of information inside the LSTM cell and help avoid gradient vanishing/exploding problem. The input gate $\mathbf{i}^{(t)}$ controls the information from the gate gate, $\mathbf{g}^{(t)}$, that is read into the cell state $\mathbf{c}^{(t)}$. The forget gate $\mathbf{f}^{(t)}$ removes the content of the cell state and the output gate $\mathbf{o}^{(t)}$ reads the output from the cell state. Similar to the RNN cell, the input and hidden state vectors are concatenated together, $\tilde{\mathbf{x}} = \begin{bmatrix} \mathbf{x}^{(t)} & \mathbf{s}^{(t-1)} \end{bmatrix}^T$. The input, forget, output and gate gates perform the following operations:

$$\begin{aligned} \mathbf{i}^{(t)} &= \psi_s(\,\tilde{\mathbf{i}}\,), \quad \text{where } \tilde{\mathbf{i}} = \boldsymbol{\mathcal{W}}^{(\mathbf{i})}\, \tilde{\mathbf{x}}, \\ \mathbf{f}^{(t)} &= \psi_s(\,\tilde{\mathbf{f}}\,), \quad \text{where } \tilde{\mathbf{f}} = \boldsymbol{\mathcal{W}}^{(\mathbf{f})}\, \tilde{\mathbf{x}}, \\ \mathbf{o}^{(t)} &= \psi_s(\,\tilde{\mathbf{o}}\,), \quad \text{where } \tilde{\mathbf{o}} = \boldsymbol{\mathcal{W}}^{(\mathbf{o})}\, \tilde{\mathbf{x}}, \\ \mathbf{g}^{(t)} &= \psi(\,\tilde{\mathbf{g}}\,), \quad \text{where } \tilde{\mathbf{g}} = \boldsymbol{\mathcal{W}}^{(\mathbf{g})}\, \tilde{\mathbf{x}}, \end{aligned} \quad (7)$$

where ψ_s and ψ refer to the sigmoid and Tanh activation functions and $\boldsymbol{\mathcal{W}}^{(\mathbf{i})}$, $\boldsymbol{\mathcal{W}}^{(\mathbf{f})}$, $\boldsymbol{\mathcal{W}}^{(\mathbf{o})}$ and $\boldsymbol{\mathcal{W}}^{(\mathbf{g})}$ are the weight matrices of the input, forget, output and gate gates, respectively. Note that we have dropped the super-script t from the vectors $\tilde{\mathbf{i}}$, $\tilde{\mathbf{f}}$, $\tilde{\mathbf{o}}$, and $\tilde{\mathbf{g}}$ to simplify the mathematical notations. At time t, the cell state $\mathbf{c}^{(t)}$ and the hidden state $\mathbf{s}^{(t)}$ are updated as follows:

$$\mathbf{c}^{(t)} = \mathbf{f}^{(t)} \odot \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \odot \mathbf{g}^{(t)}, \quad (8)$$

$$\mathbf{s}^{(t)} = \mathbf{o}^{(t)} \odot \psi(\mathbf{c}^{(t)}). \quad (9)$$

3.4.2 TRUST-LSTM

Fig. 2 presents the TRUST-LSTM network with detailed operations performed inside one cell. We first consider $(\mathbf{w}_i^{\mathbf{i}})^T, (\mathbf{w}_i^{\mathbf{f}})^T, (\mathbf{w}_i^{\mathbf{o}})^T$ and $(\mathbf{w}_i^{\mathbf{g}})^T \in \mathbb{R}^{1\times(K+H)}$, which represent i^{th} rows of the matrices $\boldsymbol{\mathcal{W}}^{(\mathbf{i})}$, $\boldsymbol{\mathcal{W}}^{(\mathbf{f})}$, $\boldsymbol{\mathcal{W}}^{(\mathbf{o})}$, and $\boldsymbol{\mathcal{W}}^{(\mathbf{g})}$, respectively, where $i = 1, \ldots, K+H$. We introduce Gaussian prior distributions over the weight vectors $\mathbf{w}_i^{\mathbf{i}}$, $\mathbf{w}_i^{\mathbf{f}}$, $\mathbf{w}_i^{\mathbf{o}}$, and $\mathbf{w}_i^{\mathbf{g}}$. The variational distributions are given by: $\mathbf{w}_i^{\mathbf{i}} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{w}_i^{\mathbf{i}}}, \boldsymbol{\Sigma}_{\mathbf{w}_i^{\mathbf{i}}})$, $\mathbf{w}_i^{\mathbf{f}} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{w}_i^{\mathbf{f}}}, \boldsymbol{\Sigma}_{\mathbf{w}_i^{\mathbf{f}}})$, $\mathbf{w}_i^{\mathbf{o}} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{w}_i^{\mathbf{o}}}, \boldsymbol{\Sigma}_{\mathbf{w}_i^{\mathbf{o}}})$ and $\mathbf{w}_i^{\mathbf{g}} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{w}_i^{\mathbf{g}}}, \boldsymbol{\Sigma}_{\mathbf{w}_i^{\mathbf{g}}})$. The weight vectors are assumed to be independent of each other and independent of the input $\tilde{\mathbf{x}}$. We write the individual elements of the vectors $\tilde{\mathbf{i}}$, $\tilde{\mathbf{f}}$, $\tilde{\mathbf{o}}$, and $\tilde{\mathbf{g}}$ in (7) as follows:

$$\begin{aligned} \tilde{i}_i &= (\mathbf{w}_i^{\mathbf{i}})^T\, \tilde{\mathbf{x}}, \qquad \tilde{f}_i = (\mathbf{w}_i^{\mathbf{f}})^T\, \tilde{\mathbf{x}}, \\ \tilde{o}_i &= (\mathbf{w}_i^{\mathbf{o}})^T\, \tilde{\mathbf{x}}, \qquad \tilde{g}_i = (\mathbf{w}_i^{\mathbf{g}})^T\, \tilde{\mathbf{x}}. \end{aligned} \quad (10)$$

Each of these elements is the result of an inner product between two independent random vectors (weights-inputs vectors). Therefore, the mean and the covariance of $\tilde{\mathbf{i}}$, $\tilde{\mathbf{f}}$, $\tilde{\mathbf{o}}$ and $\tilde{\mathbf{g}}$ are derived using Proposition 1.

Gates Activation Functions in LSTM: The mean and variance-covariance matrices at the output of the non-linear activation functions ψ_s and ψ in (7) are derived using the first-order Taylor series approximation [48],

$$\begin{aligned} \boldsymbol{\mu}_{\mathbf{i}^{(t)}} &\approx \psi_s(\boldsymbol{\mu}_{\tilde{\mathbf{i}}}), \quad \boldsymbol{\Sigma}_{\mathbf{i}^{(t)}} \approx \boldsymbol{\Sigma}_{\tilde{\mathbf{i}}} \odot \left(\nabla\psi_s(\boldsymbol{\mu}_{\tilde{\mathbf{i}}})\, \nabla\psi_s(\boldsymbol{\mu}_{\tilde{\mathbf{i}}})^T\right), \\ \boldsymbol{\mu}_{\mathbf{f}^{(t)}} &\approx \psi_s(\boldsymbol{\mu}_{\tilde{\mathbf{f}}}), \quad \boldsymbol{\Sigma}_{\mathbf{f}^{(t)}} \approx \boldsymbol{\Sigma}_{\tilde{\mathbf{f}}} \odot \left(\nabla\psi_s(\boldsymbol{\mu}_{\tilde{\mathbf{f}}})\, \nabla\psi_s(\boldsymbol{\mu}_{\tilde{\mathbf{f}}})^T\right), \\ \boldsymbol{\mu}_{\mathbf{o}^{(t)}} &\approx \psi_s(\boldsymbol{\mu}_{\tilde{\mathbf{o}}}), \quad \boldsymbol{\Sigma}_{\mathbf{o}^{(t)}} \approx \boldsymbol{\Sigma}_{\tilde{\mathbf{o}}} \odot \left(\nabla\psi_s(\boldsymbol{\mu}_{\tilde{\mathbf{o}}})\, \nabla\psi_s(\boldsymbol{\mu}_{\tilde{\mathbf{o}}})^T\right), \\ \boldsymbol{\mu}_{\mathbf{g}^{(t)}} &\approx \psi(\boldsymbol{\mu}_{\tilde{\mathbf{g}}}), \quad \boldsymbol{\Sigma}_{\mathbf{g}^{(t)}} \approx \boldsymbol{\Sigma}_{\tilde{\mathbf{g}}} \odot \left(\nabla\psi(\boldsymbol{\mu}_{\tilde{\mathbf{g}}})\, \nabla\psi(\boldsymbol{\mu}_{\tilde{\mathbf{g}}})^T\right). \end{aligned} \quad (11)$$

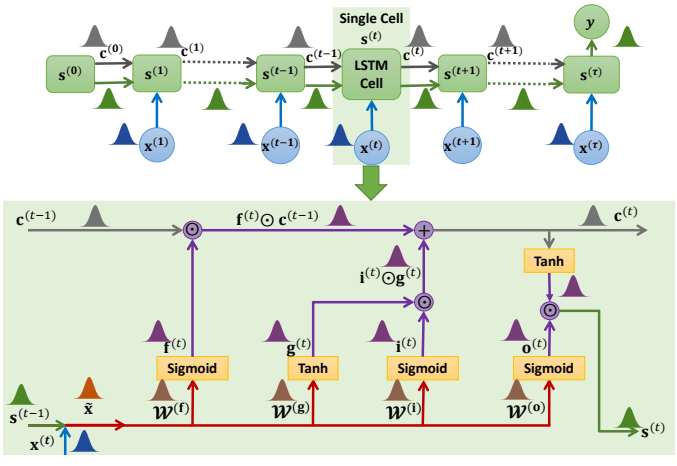


Fig. 2. A schematic layout of TRUST-LSTM with a detailed view of the operations in an LSTM cell. The following color coding is used: blue color for the input $\{\mathbf{x}^{(t)}\}_{t=1}^{\tau}$, green color for the hidden state $\{\mathbf{s}^{(t)}\}_{t=1}^{\tau}$, red color for the concatenated input and hidden state $\tilde{\mathbf{x}} = \begin{bmatrix} \mathbf{x}^{(t)} & \mathbf{s}^{(t-1)} \end{bmatrix}^T$, purple color for outputs of gates, grey color for the cell memory/state $\{\mathbf{c}^{(t)}\}_{t=1}^{\tau}$, and orange color for the activation functions.

Cell State/Memory $\mathbf{c}^{(t)}$: The derivation of the mean and covariance of the cell state $\mathbf{c}^{(t)}$ requires propagating the moments through the Hadamard product. We rewrite (8) as:

$$\mathbf{c}^{(t)} = \tilde{\mathbf{c}}_1 + \tilde{\mathbf{c}}_2, \quad \text{where} \quad \tilde{\mathbf{c}}_1 = \mathbf{f}^{(t)} \odot \mathbf{c}^{(t-1)} \quad \text{and} \quad \tilde{\mathbf{c}}_2 = \mathbf{i}^{(t)} \odot \mathbf{g}^{(t)}. \tag{12}$$

The three gates, $\mathbf{f}^{(t)}$, $\mathbf{g}^{(t)}$, $\mathbf{i}^{(t)}$, and the cell state, $\mathbf{c}^{(t-1)}$, are assumed to be uncorrelated to each other, resultantly $\tilde{\mathbf{c}}_1$ and $\tilde{\mathbf{c}}_2$ are also uncorrelated. Under this condition, the mean and covariance of $\tilde{\mathbf{c}}_1$ (and similarly for $\tilde{\mathbf{c}}_2$) are derived according to Proposition 2 (the proof is in Appendix B in the supplementary materials).

Proposition 2. (mean and variance-covariance propagation through a Hadamard product)

$$\begin{aligned} \boldsymbol{\mu}_{\tilde{\mathbf{c}}_1} &= \boldsymbol{\mu}_{\mathbf{f}^{(t)}} \odot \boldsymbol{\mu}_{\mathbf{c}^{(t-1)}}, \\ \boldsymbol{\Sigma}_{\tilde{\mathbf{c}}_1} &= \boldsymbol{\Sigma}_{\mathbf{f}^{(t)}} \odot \boldsymbol{\Sigma}_{\mathbf{c}^{(t-1)}} + \mathbf{diag}(\boldsymbol{\mu}_{\mathbf{f}^{(t)}})\, \boldsymbol{\Sigma}_{\mathbf{c}^{(t-1)}}\, \mathbf{diag}(\boldsymbol{\mu}_{\mathbf{f}^{(t)}}) \\ &\quad + \mathbf{diag}(\boldsymbol{\mu}_{\mathbf{c}^{(t-1)}})\, \boldsymbol{\Sigma}_{\mathbf{f}^{(t)}}\, \mathbf{diag}(\boldsymbol{\mu}_{\mathbf{c}^{(t-1)}}), \end{aligned} \tag{13}$$

where $\mathbf{diag}(\boldsymbol{\mu}_{\mathbf{f}^{(t)}})$ represents the diagonal matrix whose entries are given by the column vector $\boldsymbol{\mu}_{\mathbf{f}^{(t)}}$. Finally, the mean and covariance of the cell state $\mathbf{c}^{(t)}$ are derived as follows:

$$\boldsymbol{\mu}_{\mathbf{c}^{(t)}} = \boldsymbol{\mu}_{\tilde{\mathbf{c}}_1} + \boldsymbol{\mu}_{\tilde{\mathbf{c}}_2}, \qquad \boldsymbol{\Sigma}_{\mathbf{c}^{(t)}} = \boldsymbol{\Sigma}_{\tilde{\mathbf{c}}_1} + \boldsymbol{\Sigma}_{\tilde{\mathbf{c}}_2}. \tag{14}$$

Hidden State $\mathbf{s}^{(t)}$ *in LSTM*: To find the mean and variance-covariance of the hidden state, we rewrite (9) as follows:

$$\mathbf{s}^{(t)} = \mathbf{o}^{(t)} \odot \tilde{\mathbf{s}}^{(t)}, \qquad \text{where} \quad \tilde{\mathbf{s}}^{(t)} = \psi(\mathbf{c}^{(t)}). \tag{15}$$

The moments after the element-wise non-linear transformation in (15) are approximated using Taylor series,

$$\begin{aligned} \boldsymbol{\mu}_{\tilde{\mathbf{s}}^{(t)}} &\approx \psi(\boldsymbol{\mu}_{\mathbf{c}^{(t)}}), \\ \boldsymbol{\Sigma}_{\tilde{\mathbf{s}}^{(t)}} &\approx \boldsymbol{\Sigma}_{\mathbf{c}^{(t)}} \odot \left(\nabla\psi(\boldsymbol{\mu}_{\mathbf{c}^{(t)}})\, \nabla\psi(\boldsymbol{\mu}_{\mathbf{c}^{(t)}})^T\right). \end{aligned} \tag{16}$$

Then, the moments of the hidden state $\mathbf{s}^{(t)}$ in (15) are derived using Proposition 2 (propagation through the Hadamard product).

3.5 Variational Moments Propagation in TRUST-GRUs

3.5.1 The Structure of a GRU

The GRU network, introduced by [50], has a simpler internal structure than LSTM, i.e., two gates, a reset gate and an update gate. The reset gate $\mathbf{r}^{(t)}$ filters out previously retained irrelevant information from the hidden layer, and the update gate $\mathbf{z}^{(t)}$ controls the information added to the hidden state. The input and the hidden state from the previous time step are concatenated, $\tilde{\mathbf{x}} = \left[\mathbf{x}^{(t)} \;\; \mathbf{s}^{(t-1)}\right]^T$, and linearly combined with weight matrices. Using sigmoid activation functions ψ_s, the outputs of reset and update gates are given as the following:

$$\begin{aligned} \mathbf{r}^{(t)} &= \psi_s(\,\tilde{\mathbf{r}}\,), \quad \text{where} \quad \tilde{\mathbf{r}} = \boldsymbol{\mathcal{W}}^{(\mathbf{r})}\, \tilde{\mathbf{x}}, \\ \mathbf{z}^{(t)} &= \psi_s(\,\tilde{\mathbf{z}}\,), \quad \text{where} \quad \tilde{\mathbf{z}} = \boldsymbol{\mathcal{W}}^{(\mathbf{z})}\, \tilde{\mathbf{x}}, \end{aligned} \tag{17}$$

where $\boldsymbol{\mathcal{W}}^{(\mathbf{r})}$ and $\boldsymbol{\mathcal{W}}^{(\mathbf{z})}$ are weight matrices of the reset and update gates, respectively. Before we calculate the current hidden state $\mathbf{s}^{(t)}$, we need to build a candidate hidden state $\mathbf{h}^{(t)}$ using the concatenation of the input $\mathbf{x}^{(t)}$ with $[\mathbf{s}^{(t-1)} \odot \mathbf{r}^{(t)}]$, that is:

$$\hat{\mathbf{x}} = \left[\mathbf{x}^{(t)} \quad \mathbf{s}^{(t-1)} \odot \mathbf{r}^{(t)}\right]^T, \tag{18}$$

$$\mathbf{h}^{(t)} = \psi(\,\tilde{\mathbf{h}}\,), \quad \text{where} \quad \tilde{\mathbf{h}} = \boldsymbol{\mathcal{W}}^{(\mathbf{h})}\, \hat{\mathbf{x}}, \tag{19}$$

where $\boldsymbol{\mathcal{W}}^{(\mathbf{h})}$ is the weight matrix of the candidate hidden state. Now, the hidden state $\mathbf{s}^{(t)}$ is given by the following:

$$\mathbf{s}^{(t)} = \mathbf{z}^{(t)} \odot \mathbf{s}^{(t-1)} + (1 - \mathbf{z}^{(t)}) \odot \mathbf{h}^{(t)}. \tag{20}$$

3.5.2 TRUST-GRU

Fig. 3 presents the general structure of a TRUST-GRU network and operations performed within one GRU cell. Similar to TRUST-LSTM, we derive the propagation of the mean and the variance-covariance matrix of the variational posterior distribution function through a single GRU cell.

Consider $(\mathbf{w}_i^{\mathbf{r}})^T, (\mathbf{w}_i^{\mathbf{z}})^T$ and $(\mathbf{w}_i^{\mathbf{h}})^T \in \mathbb{R}^{1\times(K+H)}$ to be the i^{th} rows of the matrices $\boldsymbol{\mathcal{W}}^{(\mathbf{r})}$, $\boldsymbol{\mathcal{W}}^{(\mathbf{z})}$ and $\boldsymbol{\mathcal{W}}^{(\mathbf{h})}$, respectively, where $i = 1, \ldots, K+H$. We introduce a Gaussian distribution as the prior over the weight vectors $\mathbf{w}_i^{\mathbf{r}}$, $\mathbf{w}_i^{\mathbf{z}}$, and $\mathbf{w}_i^{\mathbf{h}}$. The variational distributions are given as: $\mathbf{w}_i^{\mathbf{r}} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{w}_i^{\mathbf{r}}}, \boldsymbol{\Sigma}_{\mathbf{w}_i^{\mathbf{r}}})$, $\mathbf{w}_i^{\mathbf{z}} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{w}_i^{\mathbf{z}}}, \boldsymbol{\Sigma}_{\mathbf{w}_i^{\mathbf{z}}})$, and $\mathbf{w}_i^{\mathbf{h}} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{w}_i^{\mathbf{h}}}, \boldsymbol{\Sigma}_{\mathbf{w}_i^{\mathbf{h}}})$. In our settings, the weight vectors are assumed to be independent of each other and of the input $\tilde{\mathbf{x}}$ and $\hat{\mathbf{x}}$. Every element of the random vectors $\tilde{\mathbf{r}}$ and $\tilde{\mathbf{z}}$ in (17) can be written as an inner product between two independent random vectors, such as $\tilde{r}_i = (\mathbf{w}_i^{\mathbf{r}})^T\, \tilde{\mathbf{x}}$ and $\tilde{z}_i = (\mathbf{w}_i^{\mathbf{z}})^T\, \tilde{\mathbf{x}}$. Therefore, the mean and covariance of the random vectors $\tilde{\mathbf{r}}$ and $\tilde{\mathbf{z}}$ are derived using Proposition 1.

Gates Activation Functions in GRU: We approximate the mean and covariance at the output of the non-linear activation function ψ_s in the reset and update gates, given in (17), using the first-order Taylor series as:

$$\begin{aligned} \boldsymbol{\mu}_{\mathbf{r}^{(t)}} &\approx \psi_s(\boldsymbol{\mu}_{\tilde{\mathbf{r}}}), \quad \boldsymbol{\Sigma}_{\mathbf{r}^{(t)}} \approx \boldsymbol{\Sigma}_{\tilde{\mathbf{r}}} \odot \left(\nabla\psi_s(\boldsymbol{\mu}_{\tilde{\mathbf{r}}})\, \nabla\psi_s(\boldsymbol{\mu}_{\tilde{\mathbf{r}}})^T\right), \\ \boldsymbol{\mu}_{\mathbf{z}^{(t)}} &\approx \psi_s(\boldsymbol{\mu}_{\tilde{\mathbf{z}}}), \quad \boldsymbol{\Sigma}_{\mathbf{z}^{(t)}} \approx \boldsymbol{\Sigma}_{\tilde{\mathbf{z}}} \odot \left(\nabla\psi_s(\boldsymbol{\mu}_{\tilde{\mathbf{z}}})\, \nabla\psi_s(\boldsymbol{\mu}_{\tilde{\mathbf{z}}})^T\right). \end{aligned} \tag{21}$$

Hidden State $\mathbf{s}^{(t)}$ *in GRU*: We start with the candidate state $\mathbf{h}^{(t)}$ and the concatenation operation as defined in (18) and (19). By introducing $\mathbf{a} = \mathbf{s}^{(t-1)} \odot \mathbf{r}^{(t)}$, we can derive the mean and the covariance of $\mathbf{a}$ using Proposition 2. Thus, the mean and the covariance matrix of $\hat{\mathbf{x}}$ in (18) are given as follows:

$$\boldsymbol{\mu}_{\hat{\mathbf{x}}} = \begin{bmatrix} \boldsymbol{\mu}_{\mathbf{x}^{(t)}} & \boldsymbol{\mu}_{\mathbf{a}} \end{bmatrix}^T, \qquad \boldsymbol{\Sigma}_{\hat{\mathbf{x}}} = \begin{bmatrix} \boldsymbol{\Sigma}_{\mathbf{x}^{(t)}} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_{\mathbf{a}} \end{bmatrix}. \tag{22}$$

The mean and covariance matrix of $\mathbf{h}^{(t)}$ as defined in (19) are derived using Proposition 1 and then the first-order Taylor

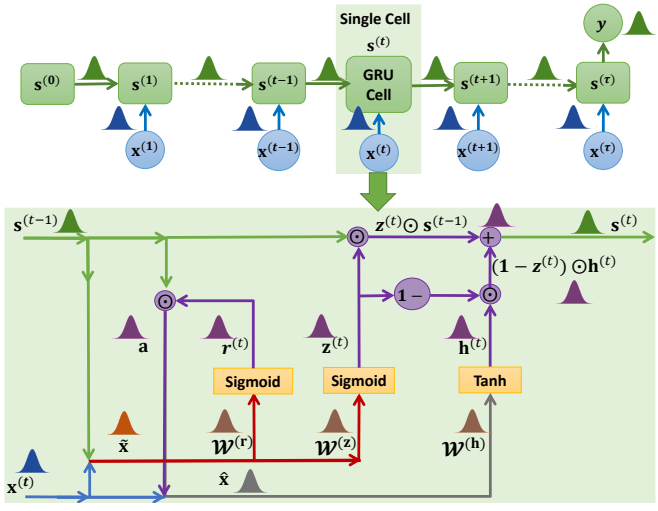


Fig. 3. A schematic layout of the proposed TRUST-GRU. One GRU cell is expanded in the bottom view. The following color scheme is used: blue for the input vector $\mathbf{x}^{(t)}$, green for the hidden states $\mathbf{s}^{(t)}$, red for the concatenated input-state vector $\tilde{\mathbf{x}} = \left[\mathbf{x}^{(t)} \;\; \mathbf{s}^{(t-1)}\right]^T$, gray for the concatenated vector $\hat{\mathbf{x}} = \left[\mathbf{x}^{(t)} \quad \mathbf{s}^{(t-1)} \odot \mathbf{r}^{(t)}\right]^T$, purple for the output of Hadamard product between two vectors or an output of a non-linear activation function, and orange for the activation functions.

series approximation (following (21)). Finally, we rewrite the hidden state $\mathbf{s}^{(t)}$ (defined in (20)) as, $\mathbf{s}^{(t)} = \tilde{\mathbf{s}}_1 + \tilde{\mathbf{s}}_2$,, where $\tilde{\mathbf{s}}_1 = \mathbf{z}^{(t)} \odot \mathbf{s}^{(t-1)}$ and $\tilde{\mathbf{s}}_2 = (1 - \mathbf{z}^{(t)}) \odot \mathbf{h}^{(t)}$. The mean and the covariance of $\tilde{\mathbf{s}}_1$ and $\tilde{\mathbf{s}}_2$ are derived using Proposition 2. Since $\tilde{\mathbf{s}}_1$ and $\tilde{\mathbf{s}}_2$ are correlated, the mean and the variance-covariance matrix of the current hidden state $\mathbf{s}^{(t)}$ are derived as follows:

$$\begin{aligned}\boldsymbol{\mu}_{\mathbf{s}^{(t)}} &= \boldsymbol{\mu}_{\tilde{\mathbf{s}}_1} + \boldsymbol{\mu}_{\tilde{\mathbf{s}}_2}, \\ &= \boldsymbol{\mu}_{\mathbf{z}^{(t)}} \odot \boldsymbol{\mu}_{\mathbf{s}^{(t-1)}} + (1 - \boldsymbol{\mu}_{\mathbf{z}^{(t)}}) \odot \boldsymbol{\mu}_{\mathbf{h}^{(t)}}. \end{aligned} \tag{23}$$

$$\begin{aligned}\boldsymbol{\Sigma}_{\mathbf{s}^{(t)}} &= \boldsymbol{\Sigma}_{\tilde{\mathbf{s}}_1} + \boldsymbol{\Sigma}_{\tilde{\mathbf{s}}_2} + \boldsymbol{\Sigma}_{\tilde{\mathbf{s}}_1\tilde{\mathbf{s}}_2} + \boldsymbol{\Sigma}^T_{\tilde{\mathbf{s}}_1\tilde{\mathbf{s}}_2}, \\ &= \boldsymbol{\Sigma}_{\mathbf{z}^{(t)}} \odot \boldsymbol{\Sigma}_{\mathbf{s}^{(t-1)}} + \mathbf{diag}(\boldsymbol{\mu}_{\mathbf{z}^{(t)}})\, \boldsymbol{\Sigma}_{\mathbf{s}^{(t-1)}}\, \mathbf{diag}(\boldsymbol{\mu}_{\mathbf{z}^{(t)}}) \\ &+ \mathbf{diag}(\boldsymbol{\mu}_{\mathbf{s}^{(t-1)}})\, \boldsymbol{\Sigma}_{\mathbf{z}^{(t)}}\, \mathbf{diag}(\boldsymbol{\mu}_{\mathbf{s}^{(t-1)}}) + \boldsymbol{\Sigma}_{\mathbf{z}^{(t)}} \odot \boldsymbol{\Sigma}_{\mathbf{h}^{(t)}} \\ &+ \mathbf{diag}(1 - \boldsymbol{\mu}_{\mathbf{z}^{(t)}})\, \boldsymbol{\Sigma}_{\mathbf{h}^{(t)}}\, \mathbf{diag}(1 - \boldsymbol{\mu}_{\mathbf{z}^{(t)}}) \\ &+ \mathbf{diag}(\boldsymbol{\mu}_{\mathbf{h}^{(t)}})\, \boldsymbol{\Sigma}_{\mathbf{z}^{(t)}}\, \mathbf{diag}(\boldsymbol{\mu}_{\mathbf{h}^{(t)}}) - \boldsymbol{\Sigma}_{\mathbf{z}^{(t)}} \odot \boldsymbol{\mu}_{\mathbf{s}^{(t-1)}}\, \boldsymbol{\mu}^T_{\mathbf{h}^{(t)}} \\ &- \boldsymbol{\Sigma}_{\mathbf{z}^{(t)}} \odot \boldsymbol{\mu}_{\mathbf{h}^{(t)}}\, \boldsymbol{\mu}^T_{\mathbf{s}^{(t-1)}}, \end{aligned} \tag{24}$$

where $\boldsymbol{\Sigma}_{\tilde{\mathbf{s}}_1\tilde{\mathbf{s}}_2}$ is the cross-covariance matrix of the two random vectors $\tilde{\mathbf{s}}_1$ and $\tilde{\mathbf{s}}_1$ (detailed derivation of $\boldsymbol{\Sigma}_{\tilde{\mathbf{s}}_1\tilde{\mathbf{s}}_2}$ is in Appendix C in the supplementary materials).

3.6 Model Output and Loss Function

For the prediction tasks, the output of the network (RNN, LSTM, or GRU) consists of a fully-connected layer, $\tilde{\mathbf{y}} = \mathbf{W}^{(y)}\, \mathbf{s}^{(T)}$, where $\mathbf{W}^{(y)}$ is the weight matrix. We use Proposition 1 to propagate the mean and the variance of the output.

In the case of classification problems, the network's output layer has a softmax function φ, i.e., $\hat{\mathbf{y}} = \varphi(\tilde{\mathbf{y}})$ after the fully-connected layer. Even though the softmax function does not operate element-wise, we can use the Taylor series to approximate the mean $\boldsymbol{\mu}_{\hat{\mathbf{y}}}$ and the covariance matrix $\boldsymbol{\Sigma}_{\hat{\mathbf{y}}}$ as follows [30]:

$$\boldsymbol{\mu}_{\hat{\mathbf{y}}} \approx \varphi(\boldsymbol{\mu}_{\tilde{\mathbf{y}}}), \quad \boldsymbol{\Sigma}_{\hat{\mathbf{y}}} \approx \mathbf{J}_\varphi \boldsymbol{\Sigma}_{\tilde{\mathbf{y}}} \mathbf{J}^T_\varphi, \tag{25}$$

where $\mathbf{J}_\varphi$ is the Jacobian matrix of the softmax function φ evaluated at $\boldsymbol{\mu}_{\tilde{\mathbf{y}}}$.

In the ELBO loss function in (1), we assume a diagonal covariance matrix for the initial variational distribution and use the first-order Taylor series to approximate the expectation. Thus, the expected log-likelihood in (1) is written as:

$$E_{q_\phi(\boldsymbol{\Omega})}\{\log p(\mathcal{D}|\boldsymbol{\Omega})\} \approx \tag{26}$$
$$-\frac{1}{2N}\sum_{i=1}^{N}\sum_{t=1}^{\tau}\Big[\log(|\boldsymbol{\Sigma}^i_{\hat{\mathbf{y}}_t}|) + (\mathbf{y}^i_t - \boldsymbol{\mu}^i_{\hat{\mathbf{y}}_t})^T(\boldsymbol{\Sigma}^i_{\hat{\mathbf{y}}_t})^{-1}(\mathbf{y}^i_t - \boldsymbol{\mu}^i_{\hat{\mathbf{y}}_t})\Big],$$

where N refers to independently and identically distributed (iid) data points (i.e., sequences of data observations), $\mathbf{y}^i_t$ is the ground truth output of the i^{th} data sequence ($t = 1, \ldots, \tau$ time steps) and $\boldsymbol{\mu}^i_{\hat{\mathbf{y}}_t}$ and $\boldsymbol{\Sigma}^i_{\hat{\mathbf{y}}_t}$ are the mean and the covariance matrix of the predicted output at time t, i.e., $\hat{\mathbf{y}}_t$, of the TRUST model.

The regularization term in (1) is the KL-divergence between two multivariate Gaussian distributions, i.e., the variational posterior distribution and the prior distribution, defined over the network parameters for all hidden units H in the RNN, LSTM, or GRU network and the fully-connected layers [51]. KL regularization terms for TRUST-RNN, TRUST-LSTM and TRUST-GRU, are explained in Appendix D in the supplementary materials.

RNNs, LSTMs and GRUs can be set up in many different ways depending upon the problem, and the dataset, such as one-to-one, one-to-many, many-to-one, many-to-many, and many-to-many with a bottleneck (encoder-decoder type) [52]. The expected log-likelihood in the ELBO loss function presented in (26) is applicable for all arrangements. For the one-to-one and many-to-one arrangements, we have $t = \tau$. The propagation of the mean and the variance-covariance, as well as the KL divergence regularization terms do not require any modification for handling different output arrangements. Moreover, the mean and covariance propagation through the TRUST models presented above can be easily extended to different variations of LSTM and GRU models, including stacked and bidirectional architectures [53], [54], [55], [56], [57], [58], [59], [60], [61], [62].

3.7 Back-propagation Through Time (BPTT)

The back-propagation operation, referred to as BPTT in sequence-based models, involves computing the gradient of the ELBO loss function $\mathfrak{L}(\phi; \mathcal{D})$ with respect to the variational parameters ϕ. BPTT for the TRUST models is performed using libraries such as PyTorch, TensorFlow or JAX. For the TRUST models, the set of parameters ϕ includes the mean vectors and variance-covariance matrices of weights and biases in RNN, LSTM or GRU networks and in the output layer. Finally, the gradient $\nabla_\phi \mathfrak{L}(\phi; \mathcal{D})$ is used to update the parameters ϕ using the gradient descent update rule.

4 Experiments

This section explains the experimental settings used to evaluate the proposed TRUST-LSTMs and TRUST-GRUs, against the state-of-the-art Bayesian and deterministic homologs. We exclude the simple RNN architecture (i.e., without the gates structure) from the simulation because it is well-known that simple RNNs suffer from vanishing and exploding gradient problems due to long-term dependencies, which make RNNs very impractical [28], [63], [64], [65], [66]. We focus on classification and univariate prediction (or forecasting) tasks in the experiments. The simulation can be easily extended to multivariate forecasting cases and other time-series tasks. In our experiments, TRUST models are compared with Bayes-by-backprop recurrent neural networks (BBB-RNNs) [21], variational or VAR-RNNs [23], DeepAR (DAR) [40] and deterministic (DET-RNNs) in both LSTM and GRU configurations using four different datasets. The original DAR model architecture in the literature includes only the LSTM network for prediction tasks [40]. Thus, we compare with DAR-LSTM in our simulation of prediction datasets. The experiments include training, validating and testing 34 different neural networks, i.e., four models (TRUST, BBB, VAR and DET), two configurations (LSTM and GRU), four datasets and the DAR model on the two prediction datasets. The datasets include weather and power consumption datasets for the prediction tasks [31], [32], and ECG5000 and PeMS-SF for the classification tasks [33]. Appendix E in the supplementary materials provides a detailed description of the datasets. Table 1 presents the architectural hyper-parameters, including the number of layers, the number of hidden units in each layer, and the batch size for each dataset. These hyper-parameters are the same for all five models (TRUST, BBB, VAR, DAR, and DET) and both configurations (LSTM and GRU). Adam algorithm is used as the optimizer [67] with a decaying learning rate (LR) and polynomial schedule [68]. The number of epochs, initial and final learning rate (LR), and the KL weighting factors are presented for TRUST models in Table 1 (columns 5 - 8). For all other models, we fine-tune these four hyper-parameters to ensure the convergence of each model to its best performance. The KL weighting factor for

the PeMS-SF dataset is set to 10^{-3} for the TRUST-LSTM model and 0.01 for the TRUST-GRU model[1].

4.1 Robustness Analysis

We establish the robustness of TRUST models against additive Gaussian noise and adversarial attacks compared to BBB, VAR, DAR, and DET models. The robustness analysis is performed on the trained and validated models for respective datasets. First, we evaluate the performance of each model on clean test data (without noise). Then, different Gaussian or adversarial noise levels are added to the test examples to evaluate each model's performance.

Three levels of Gaussian noise are used, low, medium, and high, determined by the noise required to introduce sufficient distortion in the test examples. We use standard deviation (SD) to define the noise levels for each dataset (Table 2).

The adversarial examples are generated using the fast gradient sign method (FGSM) and the basic iterative method (BIM) [69], [70]. We use untargeted attacks for the prediction tasks and both targeted and untargeted attacks for the classification tasks. We use three levels of severity for both types of adversarial attacks. The severity of attacks is defined using ϵ as given below [70]:

$$\mathbf{X}^{\text{FGSM}} = \mathbf{X} + \epsilon \ \text{sign}\big[\nabla_{\mathbf{X}} J(\mathbf{X}, y_{true})\big], \tag{27}$$

$$\mathbf{X}^{\text{BIM}}_{k+1} = \text{Clip}_{\mathbf{X},\epsilon}\Big\{\mathbf{X}^{\text{BIM}}_{k}\alpha \ \text{sign}\big(\nabla_{\mathbf{X}} J(\mathbf{X}^{\text{BIM}}_{k}, y_{true})\big)\Big\}, \tag{28}$$

where J is the ELBO objective function in (1), α is the step-size, k is the number of iterations for the BIM attack, and $\mathbf{X}^{\text{BIM}}_0 = \mathbf{X}$. We choose $\alpha = 1$ and set the maximum number of iteration to 100. The clip operation in (28) is $\|\alpha \ \text{sign}(\nabla_{\mathbf{X}} J(\mathbf{X}^{\text{BIM}}_{k}, y_{true}))\| < \epsilon$ and $\|\mathbf{X}^{\text{BIM}}_{k} + \alpha \ \text{sign}(\nabla_{\mathbf{X}} J(\mathbf{X}^{\text{BIM}}_{k}, y_{true}))\| \in [0, 1]$. Table 2 provides ϵ values for the three levels of attacks applied to the test examples of each dataset.

4.2 Variance-vs-SNR Analysis

The proposed TRUST models provide the output prediction and uncertainty information simultaneously in the form of the predictive distribution's mean and variance-covariance matrix. The analysis of uncertainty under noisy conditions (when the networks are subject to Gaussian noise or adversarial attacks) provides insights into the network's performance after deployment and possible detection of models' failure due to changes in the input.

We perform a detailed analysis of the predictive variance for TRUST, BBB, VAR and DAR models at various levels of Gaussian noise and adversarial attacks. The variance analysis involves, first, testing the trained models on clean test data and then gradually increasing the noise level (Gaussian or adversarial) in the test data. The amount of noise at each level is measured using the signal-to-noise ratio (SNR). The average predictive variance is calculated for all the test examples at each noise level.

For the prediction tasks, the outputs of the TRUST models consist of a predictive mean value and a single variance value for each sample at the next time step. In contrast, for the classification tasks, the outputs contain a predictive mean vector and a variance-covariance matrix that shows uncertainty in all classes (diagonal elements) and the correlation between different classes (off-diagonal elements). We use the variance, i.e., the diagonal value corresponding to the predicted class, for our analysis.

The predictive variance for BBB and VAR models is the sample variance between different predictions using 20 forward passes (MC samples) through the respective network for each test example. We subtract the average predictive variance at zero noise (clean test examples) from the variance values at each noise level. The resulting average predictive variance values are plotted against the respective SNR values to produce *variance-vs-SNR* curves.

The variance-vs-SNR curves in Section 5 are interpreted from right to left. The average predictive variance for test data, having very high SNR (low noise), is plotted as a point on the extreme right side of the graph. The addition of noise leads to a decrease in the SNR values, moving from right to left. The extreme left point presents average variance at the lowest SNR (highest noise).

[1]Source code available at https://github.com/dimahdera/TRUST-RNNs.git

TABLE 1
Datasets and hyperparameters used in the experiments.

Dataset	No. of Layers	Hidden units	Batch size	No. of epochs	Initial LR	Final LR	KL Weight Factor
Weather	2	30	200	50	10^{-3}	10^{-6}	10^{-3}
Power	4	100	14	200	10^{-4}	10^{-5}	10^{-4}
ECG5000	2	200	50	100	10^{-3}	10^{-5}	10^{-4}
PeMS-SF	2	400	10	600	10^{-4}	10^{-6}	10^{-3} / 0.01

TABLE 2
The levels of Gaussian noise (standard deviation (SD)) and the severity of adversarial attacks (ϵ) used in the experiments for all four datasets.

Dataset	Gaussian Noise (SD)			Adversarial Attacks (ϵ)		
	Low	Med	High	Low	Med	High
Weather	0.05	0.1	0.2	0.05	0.1	0.2
Power consumption	0.05	0.1	0.2	0.05	0.1	0.2
ECG5000	0.1	0.3	0.5	0.001	0.05	0.07
PeMS-SF	0.05	0.1	0.2	10^{-4}	0.003	0.005

4.3 Statistical Analysis

We perform statistical analysis to establish whether TRUST models perform significantly better as compared to BBB, VAR, DAR, and DET models. Given the non-normal nature of the data, i.e., RMSE and classification accuracy, we use the two-sided Wilcoxon signed-rank test to perform pair-wise comparisons between TRUST and other models, e.g., TRUST-GRU vs. BBB-GRU or TRUST-LSTM vs. VAR-LSTM. We highlight the statistically significant differences in RMSE or classification accuracy of TRUST models compared to BBB, VAR, DAR, or DET models using a $(\dagger)$ or a $(\star)$. The former symbol shows statistical significance at the level of 99% and the latter at 95%.

In the variance-vs-SNR analysis, we aim to identify the noise level that results in a statistically significant increase in the predictive variance. Therefore, we perform pair-wise comparisons between the average predictive variance at each noise level and the variance at zero noise (clean test data) using the Wilcoxon signed-rank test. The significance level is 95%, and the point of the significant increase in the predictive variance is marked with a $(*)$ on the variance-vs-SNR curves.

5 Results and Discussion

5.1 Performance Analysis and Robustness

This section evaluates the performance and establishes the robustness of the proposed TRUST models compared to BBB, VAR, DAR, and DET models. The Root Mean Square Error (RMSE) demonstrates the performance metric of the models on the prediction tasks, and the classification accuracy is the metric for the classification tasks.

5.1.1 Prediction Tasks

Table 3 presents RMSE values of the TRUST, BBB, VAR, and DET models in both LSTM and GRU configurations and the DAR-LSTM model for the weather and power consumption datasets. The RMSE values are reported for the test data before and after adding three levels of Gaussian noise and adversarial attacks. Lower RMSE values demonstrate better performance. We observe that increasing the level of Gaussian noise or adversarial attacks results in an increase in the RMSE for all models. However, at higher noise levels, TRUST models perform significantly better than all other models. The bold font refers to the model that significantly outperforms all tested models.

Weather dataset: In Table 3(a), we observe that for the noise-free case, the proposed TRUST-LSTM/GRU models perform significantly better than VAR, DAR, and DET models yet slightly worse than BBB models. However, as the level of Gaussian noise increases, the TRUST models maintain significantly lower RMSE. On the other hand, TRUST models significantly outperform all other models ($p < .01$ for all pair-wise comparisons) for higher levels of FGSM and BIM adversarial noise.

Power consumption dataset: Table 3(b) shows that TRUST-LSTM significantly outperforms all other models ($p < .01$) when Gaussian noise or adversarial attack is added to the test set. Yet, TRUST-GRU performs significantly better than all other models, only at high levels of Gaussian noise and adversarial attacks.

5.1.2 Classification Tasks

Table 4 presents classification accuracy for TRUST, BBB, VAR, and DET models in both LSTM and GRU configurations for PeMS-SF and ECG5000 datasets. The classification accuracy is measured on the clean test data and after adding three levels of Gaussian and adversarial noise (using targeted and untargeted settings) to the test data. For the targeted adversarial attacks, the target class is class label 3. We notice that there are no statistical differences between the performance of TRUST and other models on the clean test dataset. However, when Gaussian and particularly targeted/untargeted adversarial noise is added to the test data, the TRUST models significantly outperform BBB, VAR, and DET models. The bold font in Table 4 refers to the model that significantly outperforms all tested models.

TABLE 3
Test RMSE of TRUST-LSTM/GRU compared to BBB, VAR, DAR, and DET models. All models are tested using (1) noise-free test data, (2) three levels of Gaussian noise, and (3) two types of adversarial attacks, FGSM and BIM, with three levels of attack severity.

Noise		LSTM Models					GRU Models			
		TRUST	BBB	VAR	DAR	DET	TRUST	BBB	VAR	DET
(a) Weather Dataset										
No noise		.024	**.021**†	.034†	.032†	.025	.028	**.026**†	.037†	.039†
Gaussian	Low	.028	**.026**†	.036†	.034†	.029	**.031**	**.031**	.039†	.047†
	Med	**.036**	**.036**	.041†	.037	.037	**.04**	.042†	.042†	.062†
	High	**.063**	.065†	.066†	.064†	.067†	.063	.07†	**.056**†	.105†
FGSM	Low	**.058**	.102†	.091†	**.058**	.083†	**.059**	.083†	.095†	.135†
	Med	**.105**	.179†	.146†	**.105**	.142†	**.104**	.142†	.152†	.226†
	High	**.199**	.302†	.243†	.217†	.248†	**.197**	.255†	.252†	.387†
BIM	Low	**.059**	.102†	.091†	**.059**	.083†	**.059**	.083†	.096†	.135†
	Med	**.105**	.179†	.146†	.161	.142†	**.105**	.143†	.154†	.226†
	High	**.2**	.302†	.243†	.220†	.248†	**.198**	.256†	.253†	.387†
(b) Power Consumption Dataset										
No noise		.508	.511†	.5	.544†	**.496**†	.508	.519†	.497†	**.495**†
Gaussian	Low	**.546**	.583†	.55*	.582†	.573†	.566	.591†	**.542**†	.585†
	Med	**.619**	.745†	.67†	.673†	.75†	**.653**	.755†	**.651**	.798†
	High	**.802**	1.172†	1.012†	.867†	1.2†	**.862**	1.211†	.968†	1.386†
FGSM	Low	**.788**	.937†	.844†	.857†	.957†	.877	.938†	**.812**†	.999†
	Med	**1.01**	1.315†	1.143†	1.141†	1.349†	**1.101**	1.315†	**1.093**	1.404†
	High	**1.353**	1.973†	1.666†	1.589†	1.993†	**1.368**	1.958†	1.612†	2.107†
BIM	Low	**.768**	.937†	.836†	.902†	.946†	.857	.949†	**.817**†	.999†
	Med	**1.001**	1.326†	1.134†	1.165†	1.348†	**1.107**	1.343†	**1.097**	.1.416†
	High	**1.319**	1.998†	1.663†	1.547†	2.054†	**1.306**	2.039†	1.612†	2.131†

5.2 Uncertainty and Self-Assessment

We use the predictive variance, at the output of the TRUST models, as a quantitative metric to assess their performance without any additional data processing or computational burden. We refer to this as *self-assessment* since the TRUST models can ascertain whether their predictions are trustworthy based on the variance information. For example, high variance reflects high uncertainty or low confidence in the prediction.

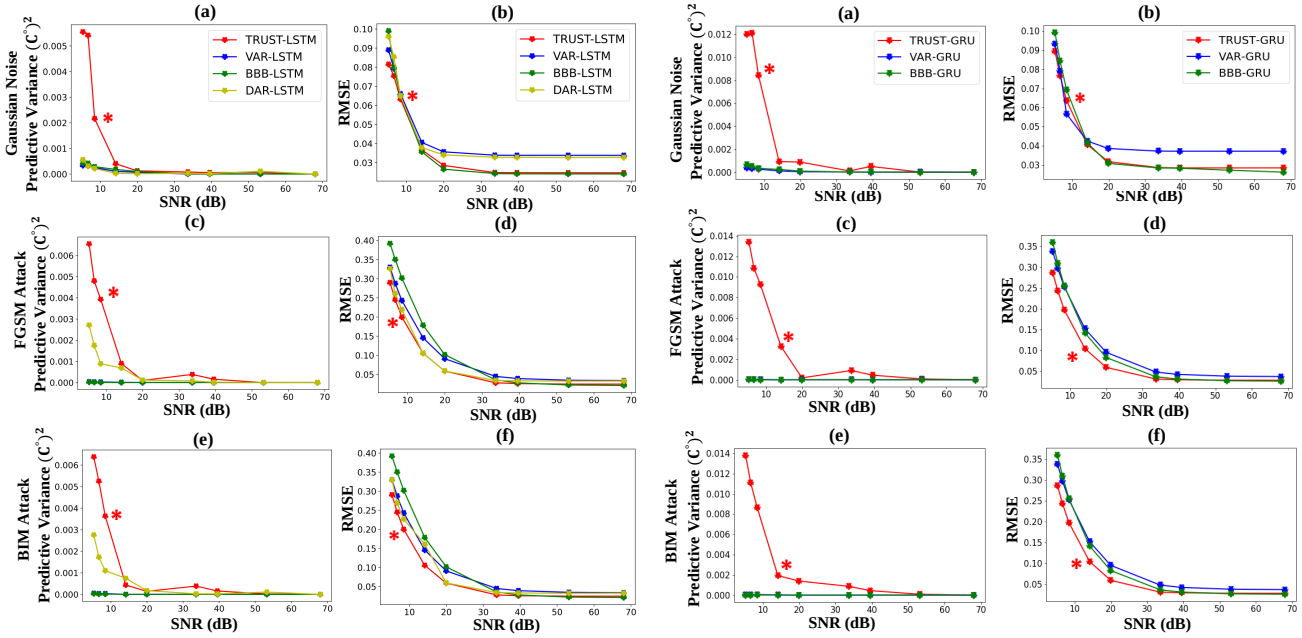
5.2.1 Prediction Tasks

Figs. 4(i) and 5(i) present variance-vs-SNR (left sub-figures) and RMSE-vs-SNR (right sub-figures) for TRUST, BBB, VAR, and DAR LSTMs using weather and power consumption datasets, respectively. Figs. 4(ii) and 5(ii) present similar plots for TRUST, BBB, and VAR GRUs. We interpret variance-vs-SNR curves from right to left in all plots. The $(*)$ in the variance plots (left sub-figures) indicates the significant increase ($p < .05$) in the average predictive variance and the corresponding SNR value. In the RMSE plots (right sub-figures), the $(*)$ refers to the corresponding RMSE values when the variance becomes significantly higher.

We notice that with the increasing noise level (or equivalently decreasing SNR) for both Gaussian and adversarial noise, the RMSE values of all models increase. However, TRUST models maintain lower RMSE, and the average predictive variance of the TRUST models increases significantly. The TRUST models can use this significant increase in the predictive variance to assess their own performance (self-assessment). Predictive variance in BBB, VAR, and DAR models does not show such behavior.

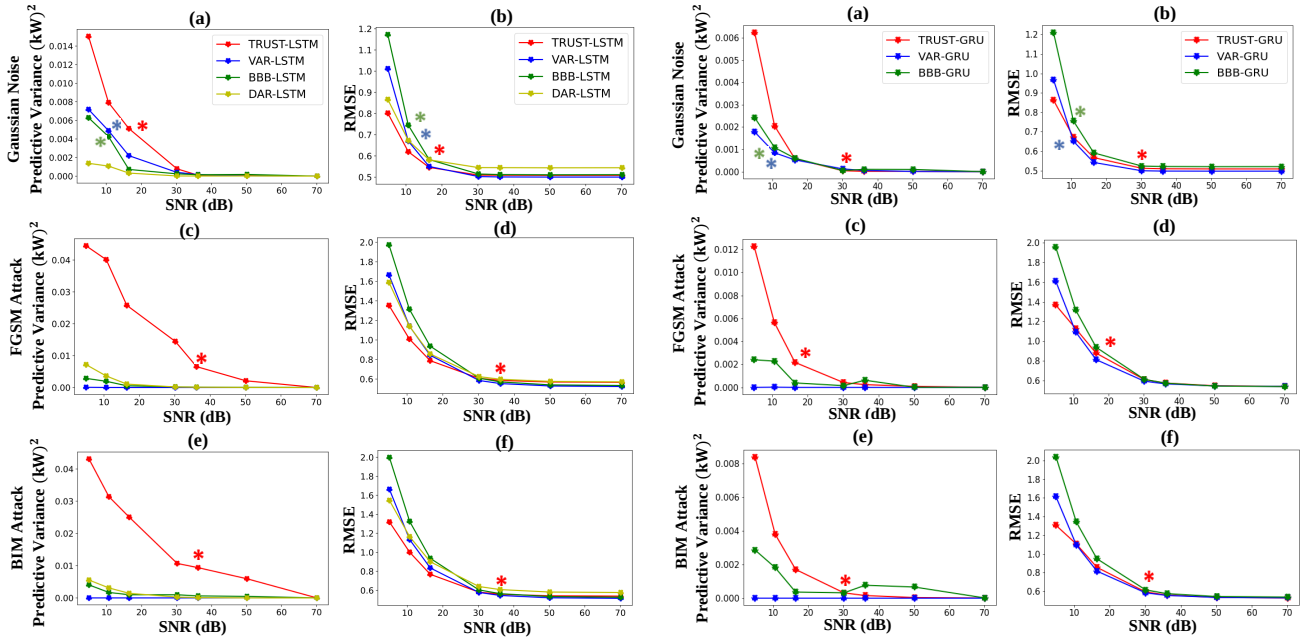
Weather dataset: With a decreasing SNR, we observe that the TRUST-LSTM variance significantly increases at SNR ≤ 8 dB for the Gaussian noise (Fig. 4(i)(a)) and for both types of adversarial attacks, FGSM (Fig. 4(i)(c)) and BIM (Fig. 4(i)(e)). The right sub-figures in Fig. 4(i) show that the RMSE increases from 0.024 at high SNR values to 0.063 for Gaussian noise (Fig. 4(i)(b)) and 0.2 for both FGSM and BIM (Fig. 4(i)(d and f)) at the $(*)$ point. Thus, when the TRUST models are less accurate (higher RMSE values), they become uncertain (significant increase in the variance). The DAR variance increases at high levels of adversarial attacks (SNR ≤ 4 dB), failing to detect the attacks at multiple earlier levels.

In Fig. 4(ii), the significant increase in the TRUST-GRU variance is observed at SNR ≤ 8 dB for the Gaussian noise (Fig. 4(ii)(a)) and at SNR ≤ 14 dB for the FGSM (Fig. 4(ii)(c)) and BIM (Fig. 4(ii)(e)). The right sub-figures in Fig. 4(ii) show that the RMSE of TRUST-GRU increases from 0.028 at high SNR values to 0.063 for Gaussian noise (Fig. 4(ii)(b)) and 0.11 for both FGSM and BIM (Fig. 4(ii)(d and f)). The variance-vs-SNR curves of BBB and VAR models (Fig. 4(i) and 4(ii)) do not demonstrate a significant increase in the predictive variance with the increasing levels of Gaussian noise or severity of adversarial attacks.



(i) LSTM Configuration (ii) GRU Configuration

Fig. 4. Weather Dataset: The average predictive variance and RMSE, both plotted against SNR for TRUST, BBB, and VAR models in both (i) LSTM and (ii) GRU configurations and DeepAR (DAR) LSTM. Various levels of noise (Gaussian, FGSM or BIM) are added to the test data, and the SNR values are calculated and plotted on the x-axes in all sub-figures. The variance values are averaged over all test samples. Interpreting sub-figures from right to left: we observe a significant increase (indicated by $*$) in the predictive variance of TRUST models with increasing noise levels or severity of adversarial attacks (or equivalently decreasing SNR); however, BBB, VAR and DAR models do not demonstrate such behavior.



(i) LSTM Configuration (ii) GRU Configuration

Fig. 5. Power Consumption Dataset: The average predictive variance and RMSE, both plotted against SNR for TRUST, BBB, and VAR models in both (i) LSTM and (ii) GRU configurations and DeepAR (DAR) LSTM. Various levels of noise (Gaussian, FGSM or BIM) are added to the test data, and the SNR values are calculated and plotted on the x-axes in all sub-figures. The variance values are averaged over all test samples. Interpreting sub-figures from right to left: we observe a significant increase (indicated by $*$) in the predictive variance of TRUST models with increasing noise levels or severity of adversarial attacks (or equivalently decreasing SNR). BBB and VAR variance shows an increase when Gaussian noise is added; however, no significant increase in the variance is noticed when models are subject to FGSM or BIM attacks.

Power consumption dataset: The TRUST-LSTM variance significantly increases at SNR $\leq$ 18 dB for the Gaussian noise (Fig. 5(i)(a)) and at SNR $\leq$ 36 dB for both FGSM (Fig. 5(i)(c)) and BIM (Fig. 5(i)(e)). The RMSE increases from 0.5 at high SNR values to 0.54 for Gaussian noise (Fig. 5(i)(b)) and 0.6 for both FGSM and BIM attacks (Fig. 5(i)(d and f)).

In Fig. 5(ii), the significant increase in the TRUST-GRU variance is observed at SNR $\leq$ 30 dB for the Gaussian noise

TABLE 4
The classification accuracy of TRUST, BBB, VAR, and DET models. All models are tested using (1) noise-free test data, (2) three levels of Gaussian noise, and (3) two types of adversarial attacks, FGSM and BIM (targeted / untargeted), each with three levels of attack severity.

Noise		LSTM Models				GRU Models			
		TRUST	BBB	VAR	DET	TRUST	BBB	VAR	DET
(a) PeMS-SF Dataset									
No noise		87.1	87.2	87.1	86.8	88.7	86.5	88.8	88.8
Gaussian	Low	**87.1**	84.4	85.1	81.7⋆	**86.5**	77.4†	83.6	83.5
	Med	**84.1**	74.5†	79.5	80.0	**76.5**	45.6†	76.2	71.2
	High	**64.1**	42.9†	59.3	60.0	**52.4**	19.6†	52.3	47.0
FGSM (T)	Low	87.1	86.8	**87.2**	84.7	88.7	81.8†	87.6⋆	**88.8**⋆
	Med	**49.4**	32.0†	37.1	30.0†	**40.0**	26.4†	24.7†	7.6†
	High	**43.5**	18.1†	29.4†	12.9†	**33.5**	11.9†	14.7†	5.3†
FGSM (U)	Low	85.3	85.5	**87.1**	83.5	**87.1**	78.9†	82.9	83.5
	Med	**31.8**	23.2†	22.4 ⋆	19.4†	**17.6**	16.6	8.8⋆	11.1⋆
	High	**31.2**	10.1†	19.4 ⋆	10.6†	**7.6**	5.2	7.0	5.1
BIM (T)	Low	87.1	86.1	**87.2**	84.7	88.7	81.4†	87.6⋆	**88.8**⋆
	Med	**49.4**	31.8†	37.0	30.0†	**40.0**	25.6†	24.6†	7.6†
	High	**43.5**	17.6†	29.2†	12.7†	**33.2**	11.5†	14.6†	5.2†
BIM (U)	Low	85.2	84.6	**87.1**	83.5	**87.1**	80.5†	82.9	83.5
	Med	**31.7**	22.5†	22.3⋆	19.4†	**17.5**	16.4	8.8⋆	11.1⋆
	High	**31.2**	8.8†	19.4⋆	10.5†	**7.6**	4.9⋆	7.1	5.0⋆
(b) ECG5000 Dataset									
No noise		93.2	92.4	**95.2**	**95.2**	93.8	92.6	93.2	**96.2**
Gaussian	Low	**93.0**	90.8	92.4	92.0	90.6	93.3	93.6	93.6
	Med	**82.0**	80.7	80.1	79.4	**87.2**	82.2 ⋆	85.6	78.6⋆
	High	**73.0**	68.9⋆	70.4	65.2†	**76.4**	67.7⋆	73.4	64.8⋆
FGSM (T)	Low	92.0	92.2	**94.5**	93.6	91.6	92.9	91.7	92.8
	Med	**87.0**	56.7†	68.8†	8.0†	**87.2**	45.9†	64.6†	28.9†
	High	**84.2**	4.2†	8.8†	2.6†	**85.9**	8.6†	16.2†	2.2†
FGSM (U)	Low	92.2	92.8	93.3	93.8	92.4	93.6	91.3	90.7
	Med	**41.2**	40.7†	40.2⋆	40.2	**50.2**	36.9⋆	27.4⋆	10.2⋆
	High	**2.4**	2.2	1.4⋆	1.0⋆	**40.0**	3.7†	3.8†	3.4†
BIM (T)	Low	92.0	93.2	**94.5**	93.6	91.5	92.6	91.6	92.7
	Med	**87.0**	56.2†	68.2†	8.0†	**87.1**	45.8†	64.0†	28.5†
	High	**84.0**	4.0†	8.7†	2.6†	**85.5**	8.6†	16.1†	2.2†
BIM (U)	Low	92.2	92.8	93.3	**93.8**	92.3	**93.4**	91.2	90.5
	Med	**41.2**	40.1⋆	40.2⋆	40.1	**50.1**	36.5⋆	27.2⋆	10.0⋆
	High	**2.4**	2.2	1.4⋆	1.0⋆	**39.9**	3.6†	3.7†	3.3†

and BIM attack (Fig. 5(ii)(a and e)) and at SNR $\leq$ 16 dB for the FGSM (Fig. 5(ii)(c)). The RMSE of TRUST-GRU increases from 0.5 at high SNR values to 0.51 for Gaussian noise (Fig. 5(ii)(b)), 0.88 for FGSM (Fig. 5(ii)(d)) and 0.56 for BIM (Fig. 5(ii)(f)). We also notice that the average predictive variance for BBB and VAR models increases significantly with decreasing SNR for the Gaussian noise only (Fig. 5(i)(a) and 5(ii)(a)). However, the TRUST-LSTM/GRU models are more sensitive to noise, i.e., they detect the noise at higher SNR values than their BBB and VAR homologs. The variance-vs-SNR curves of BBB, VAR and DAR models do not demonstrate a significant increase in the predictive variance with the increasing severity of FGSM or BIM adversarial attacks (Fig. 5(i)(c and e) and 5(ii)(c and e)).

5.2.2 Classification Tasks

Figs. 6(i) and 6(ii) present the predictive variance (left sub-figures) and the classification accuracy (right sub-figures) plotted against SNR in the LSTM configuration for PeMS-SF and ECG5000.

PeMS-SF dataset: With a decreasing SNR, the TRUST-LSTM variance significantly increases at SNR $\leq$ -8 dB for Gaussian noise (Fig. 6(i)(a)) and SNR $\leq$ 23 dB for both targeted and untargeted adversarial attacks, FGSM (Fig. 6(i)(c and e)) and BIM (Fig. 6(i)(g and i)). The right sub-figures in Fig. 6(i) show that the TRUST-LSTM classification accuracy declines from 87% at high SNR values to $\sim 64\%$ for Gaussian noise, $\sim 43\%$ for targeted FGSM and BIM, and $\sim 31\%$ for untargeted FGSM and BIM.

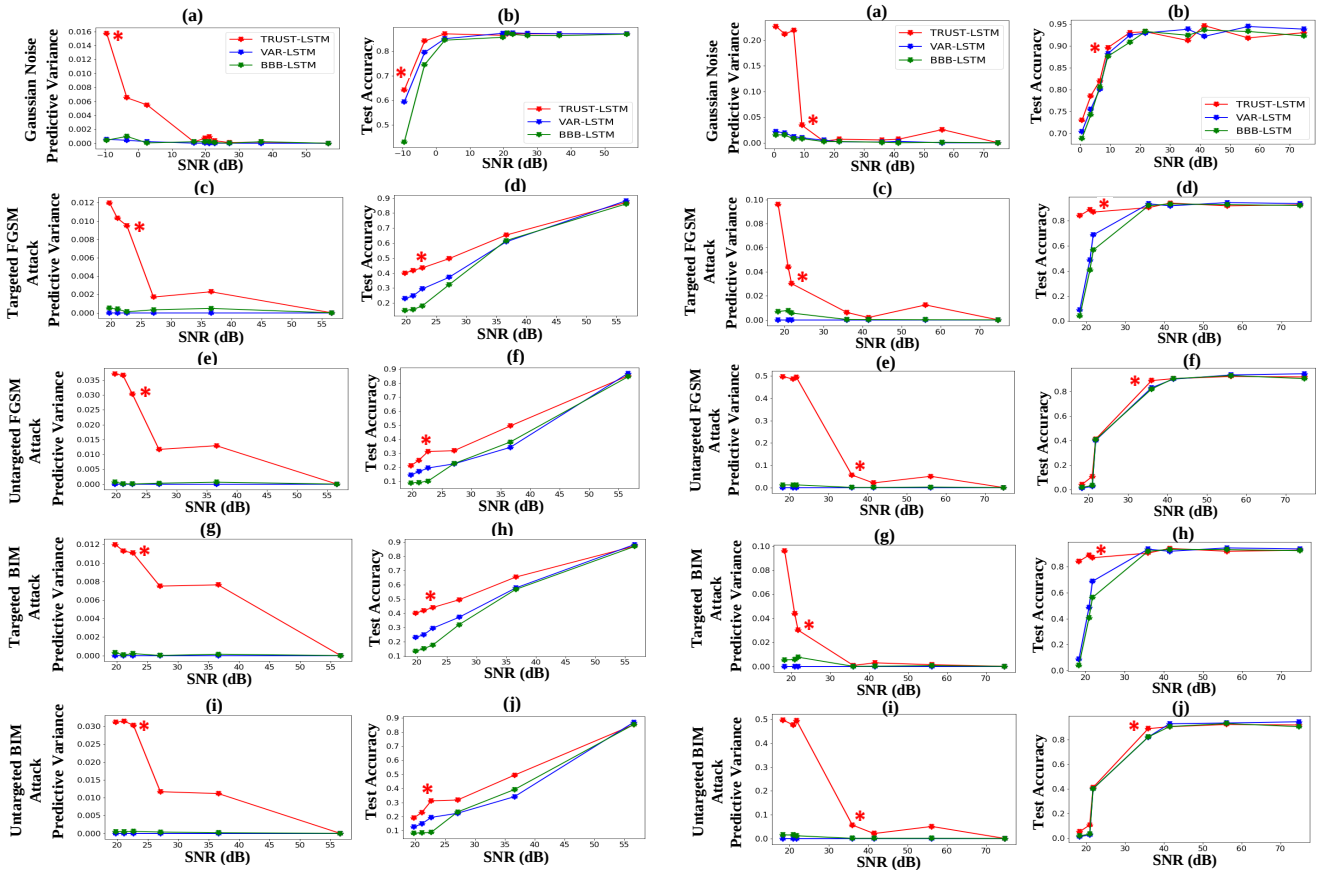
Although the classification accuracy of BBB and VAR LSTMs severely declines with the low SNR values (for both Gaussian and adversarial noise), the variance-vs-SNR curves (Fig. 6(i)) do not demonstrate a significant increase in the predictive variance.

ECG5000 dataset: With a decreasing SNR, we observe that the TRUST-LSTM variance significantly increases at SNR $\leq$ 8 dB for Gaussian noise (Fig. 6(ii)(a)) and SNR $\leq$ 22 dB for targeted FGSM and BIM (Fig. 6(ii)(c and g)). For untargeted FGSM and BIM, the TRUST-LSTM variance significantly increases at SNR $\leq$ 36 dB (Fig. 6(ii)(e and i)). The right sub-figures in Fig. 6(ii) show that the TRUST-LSTM classification accuracy declines from 93% at high SNR values to $\sim 90\%$ under Gaussian noise (Fig. 6(ii)(b)), $\sim 84\%$ under targeted or untargeted FGSM and BIM at the $(*)$ point (Fig. 6(ii)(d, f, h and j)). The variance-vs-SNR curves of BBB and VAR models (Fig. 6(ii)) do not demonstrate a significant increase in the predictive variance with the increasing levels of Gaussian noise or adversarial attacks. The variance-vs-SNR and classification accuracy-vs-SNR curves of TRUST, BBB and VAR models in the GRU configuration are provided in Appendix F in the supplementary materials.

5.3 Computational Comparison

The computational demand of the proposed TRUST models is comparable to that of DET and VAR homologs and significantly less than that of BBB and DAR models. Since the prior variational distribution has a diagonal covariance matrix, TRUST models add a single parameter (the variance) for every weight vector in a fully-connected (FC) layer. Thus, for an FC layer with H hidden units, the number of additional parameters is H. While for an LSTM or GRU layer, the number of additional parameters is $H\times$ the number of weight matrices. The first-order Taylor series approximation of the mean and the variance-covariance matrix after non-linear activation functions operates without additional parameters.

Table 5 presents the total number of parameters for one LSTM layer (4 gates), one GRU layer (3 gates), and one FC layer for TRUST, BBB, DAR, and VAR/DET models. We assume that the total number of hidden units is $H = 100$, the input vector size is $K = 30$, and the number of output neurons (or classes for classification problems) is $C = 5$. The increase in the number of parameters for TRUST models is $\approx 1.5\%$ for the LSTM/GRU layers and $\approx 1\%$ for the FC layer compared to DET/VAR models. The increase in the number of parameters of TRUST models is significantly lower than that of the BBB models, which require twice the number of parameters compared to DET/VAR models. The number of parameters for the DAR-LSTM model is comparable to the number of DET-LSTM parameters with an additional FC layer with two outputs for the mean and the standard deviation. The VAR models have the same number of parameters as DET models because VAR models use dropout approximation to estimate the variational distribution without adding parameters.



(i) PeMS-SF Dataset - LSTMs (ii) ECG5000 Dataset - LSTMs

Fig. 6. The average predictive variance and classification accuracy, both plotted against SNR for TRUST, BBB, and VAR LSTMs for (i) PeMS-SF and (ii) ECG5000 datasets. Various noise levels (Gaussian, FGSM, and BIM, targeted and untargeted) are added to the test data, and the SNR values are calculated and plotted on the x-axes in all sub-figures. The variance values are averaged over all test samples. The $(*)$ indicates the statistical increase in the predictive variance when the noise increases or the SNR decreases.

Table 6 presents inference time measured in seconds and calculated using all the test examples for each one of the five datasets. We notice that the inference time of the TRUST models is higher than that of DET models but generally lower than the inference time of BBB, VAR and DAR models. The BBB, VAR, and DAR models require Monte Carlo sampling at the test time to compute average prediction and sample variance, which increases the inference time. Therefore, the robustness of TRUST models is maintained without a notable increase in the computational complexity and inference time, which allows the proposed models to be applicable to real-world applications.

5.4 Discussion

This paper introduces uncertainty estimation in deep sequence models, RNNs, LSTMs, and GRUs, focusing on time-series analysis. We propose the TRUST models based on Bayesian density propagation. TRUST-LSTMs and GRUs propagate the first two moments of the variational posterior distribution of the models' parameters and estimate the uncertainty in models' predictions via the variance of the predictive distribution.

A detailed analysis of the proposed TRUST models is performed using five different time-series prediction and classification datasets. The performance of TRUST models is compared with the state-of-the-art LSTM/GRU networks, including BBB, VAR, DAR, and DET, under various levels of Gaussian noise and two different types of adversarial attacks, i.e., FGSM and BIM in both targeted and untargeted settings.

Our analysis reveals that TRUST models maintain their performance and outperform other models (BBB, VAR, DAR, and DET) when subject to Gaussian noise or adversarial attacks. Furthermore, the predictive variance of the TRUST models has shown a statistically significant increase when the noise is high and the performance of models starts to decrease. In general, we do not observe an increase in the average predictive variance for

TABLE 5
Total number of parameters of an LSTM layer (4 gates), a GRU layer (3 gates) and an output FC layer for TRUST, BBB, VAR, DAR, and DET models. The number of hidden units is $H = 100$, the size of the input vector is $K = 30$ and the number output neurons is $C = 5$.

Model	Layer type	Parameter Calculation	Total
TRUST	LSTM	$4 \times K \times H + 4 \times H \times H + 8 \times H$	52,800
	GRU	$3 \times K \times H + 3 \times H \times H + 6 \times H$	39,600
	FC	$H \times C + C$	505
BBB	LSTM	$2 \times (4 \times K \times H + 4 \times H \times H)$	104,000
	GRU	$2 \times (3 \times K \times H + 3 \times H \times H)$	78,000
	FC	$2 \times (H \times C)$	1,000
DAR	LSTM	$4 \times K \times H + 4 \times H \times H$	52,000
	FC	$H \times H + 2 \times (H \times C)$	11,000
DET VAR	LSTM	$4 \times K \times H + 4 \times H \times H$	52,000
	GRU	$3 \times K \times H + 3 \times H \times H$	39,000
	FC	$H \times C$	500

TABLE 6
The inference time in seconds for all models in both LSTM and GRU configurations, calculated using clean test examples.

Dataset	LSTM Models					GRU Models			
	TRUST	BBB	VAR	DAR	DET	TRUST	BBB	VAR	DET
Weather	6.3	28.1	21.7	21.8	1.6	4.5	24.4	21.2	2.2
Power	19.9	43.5	28.8	22.3	2.1	22.8	35.4	33.1	1.9
PeMS-SF	14.0	31.0	21.8	-	1.7	12.6	26.8	20.3	1.6
ECG5000	5.7	2.9	0.8	-	0.4	3.1	2.7	0.8	0.3

BBB, VAR or DAR models, which indicates that the variance at the output of BBB, VAR and DAR models does not capture the uncertainty in the model decision. We believe that propagating moments of the variational distribution through the network layers transmits vital information about the strong and weak features of the data from the hidden states to the output layer. The second moment (i.e., the variance) of the variational distribution over the parameters acts as a filter on the input features and modulates these features according to their importance. This additional filtering of features via the variance of the variational distribution forces the predictive variance to statistically increase when these features are corrupted with noise or adversarial attacks.

6 Conclusion

This paper proposes a new framework for TRustworthy Uncertainty propagation for Sequential Time-series analysis in recurrent neural networks, i.e., RNN, LSTM, and GRU networks, named TRUST-RNNs. TRUST models consider network parameters as random variables and approximate the variational posterior distribution by minimizing the evidence lower bound, which allows for estimating the output uncertainty. TRUST-RNNs propagate the first two moments of the variational distribution through the sequential network layers. The extensive experiments on various time-series classification and prediction datasets have demonstrated statistically significant robustness against Gaussian noise and adversarial attacks compared to the state-of-the-art Bayesian and deterministic RNNs. We have also shown that propagating uncertainty in RNNs and their variants inherently results in self-aware models that can assess their own performance and produce a statistically significant increase in the output uncertainty (measured by the predictive variance) in the presence of Gaussian noise and particularly adversarial attacks (targeted or untargeted).

Acknowledgments

The work was supported by the National Science Foundation Awards CRII-2153413, ECCS-1903466 and OAC-2008690. The authors gratefully acknowledge the financial support by the Lockheed Martin Corporation, including the insightful discussions with Sanipa K. Arnold, Integrated Warfare Systems and Sensors Artificial Intelligence/Machine Learning Strategy Lead, and Jeff Cammerata, Lead Member Engineering Staff. We are also grateful to UK EPSRC support through EP/T013265/1 project NSF-EPSRC: ShiRAS - Towards Safe and Reliable Autonomy in Sensor Driven Systems.

References

[1] T. Guo, Z. Xu, X. Yao, H. Chen, K. Aberer, and K. Funaya, "Robust online time series prediction with recurrent neural networks," in *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. Ieee, 2016, pp. 816–825.

[2] C. Choi, "Time series forecasting with recurrent neural networks in presence of missing data," Master's thesis, UiT The Arctic University of Norway, 2018.

[3] G. V. Houdt, C. Mosquera, and G. Nápoles, "A review on the long short-term memory model," *Artificial Intelligence Review*, pp. 1–27, 2020.

[4] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: LSTM cells and network architectures," *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[5] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[6] H. Hippert, C. Pedreira, and R. Souza, "Neural networks for short-term load forecasting: a review and evaluation," *IEEE Transactions on Power Systems*, vol. 16, no. 1, pp. 44–55, 2001.

[7] M. Rana, I. Koprinska, and A. Troncoso, "Forecasting hourly electricity load profile using neural networks," in *International Joint Conference on Neural Networks (IJCNN)*, 2014, pp. 824–831.

[8] Q. Zhang, H. Wang, J. Dong, G. Zhong, and X. Sun, "Prediction of sea surface temperature using long short-term memory," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 10, pp. 1745–1749, 2017.

[9] J. Gao, H. Zhang, P. Lu, and Z. Wang, "An effective LSTM recurrent network to detect arrhythmia on imbalanced ECG dataset," *Journal of Healthcare Engineering*, 2019.

[10] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals," *circulation*, vol. 101, no. 23, pp. e215–e220, 2000.

[11] F. Eichinger, P. Efros, S. Karnouskos, and K. Böhm, "A time-series compression technique and its application to the smart grid," *The VLDB Journal*, vol. 24, 04 2014.

[12] S. S. Aljameel, D. M. Alomari, S. Alismail, F. Khawaher, A. A. Alkhudhair, F. Aljubran, and R. M. Alzannan, "An anomaly detection model for oil and gas pipelines using machine learning," *Computation*, vol. 10, no. 8, 2022.

[13] S. Ning, J. Sun, C. Liu, and Y. Yi, "Applications of deep learning in big data analytics for aircraft complex system anomaly detection," *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, vol. 235, no. 5, pp. 923–940, 2021.

[14] D. Dera, N. C. Bouaynaya, G. Rasool, R. Shterenberg, and H. M. Fathallah-Shaykh, "PremiUm-CNN: Propagating uncertainty towards robust convolutional neural networks," *IEEE Transactions on Signal Processing*, vol. 69, pp. 4669–4684, 2021.

[15] F. Karim, S. Majumdar, and H. Darabi, "Adversarial attacks on time series," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 10, pp. 3309–3320, 2021.

[16] M. G. Abdu-Aguye, W. Gomaa, Y. Makihara, and Y. Yagi, "Detecting adversarial attacks in time-series data," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020, pp. 3092–3096.

[17] G. R. Mode and K. A. Hoque, "Adversarial examples in deep learning for multivariate time series regression," in *IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, 2020, pp. 1–10.

[18] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning, (ICML)*, vol. 37, 2015, pp. 1613–1622.

[19] K. Shridhar, F. Laumann, A. Llopart Maurin, and M. Liwicki, "Bayesian convolutional neural networks," *arXiv preprint arXiv:1806.05978*, 2018.

[20] Y. Gal and Z. Ghahramani, "Bayesian convolutional neural networks with Bernoulli approximate variational inference," in *Proceedings of 4th International Conference on Learning Representations, (ICLR) workshop track*, 2016.

[21] M. Fortunato, C. Blundell, and O. Vinyals, "Bayesian recurrent neural networks," *arXiv preprint arXiv:1704.02798*, 2017.

[22] Y. Gal, "Uncertainty in deep learning," Ph.D. dissertation, University of Cambridge, 2016.

[23] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," *Proceedings of the 29th International Conference on Neural Information Processing Systems, (NIPS)*, vol. 29, pp. 1019–1027, 2016.

[24] D. Dera, G. Rasool, and N. Bouaynaya, "Extended variational inference for propagating uncertainty in convolutional neural networks," in *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, 2019, pp. 1–6.

[25] D. Dera, G. Rasool, N. C. Bouaynaya, A. Eichen, S. Shanko, J. Cammerata, and S. Arnold, "Bayes-SAR Net: Robust SAR image classification with uncertainty estimation using Bayesian convolutional neural network," in *2020 IEEE International Radar Conference (RADAR)*, 2020, pp. 362–367.

[26] T. Bengtsson, P. Bickel, and B. Li, "Curse-of-dimensionality revisited: Collapse of the particle filter in very large scale systems," *Probability and Statistics: Essays in Honor of David A. Freedman*, p. 316–334, 2008.

[27] C. Snyder, T. Bengtsson, P. Bickel, and J. Anderson, "Obstacles to high-dimensional particle filtering," *Monthly Weather Review*, vol. 136, no. 12, pp. 4629–4640, 2008.

[28] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on International Conference on Machine Learning, (ICML)*, 2013, pp. 1310–1318.

[29] S. Kanai, Y. Fujiwara, and S. Iwamura, "Preventing gradient explosions in gated recurrent units," in *Proceedings of the 30th International Conference on Neural Information Processing Systems, (NIPS)*, 2017.

[30] D. Simon, *Optimal State Estimation: Kalman, H Infinity, and Non-linear Approaches*. Wiley-Interscience, 2006.

[31] W. S. Beutenberg, "Max planck institute for biogeochemistry: Weather time series dataset," 2020. [Online]. Available: https://www.bgc-jena.mpg.de/wetter/

[32] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[33] T. Bagnall, "Time series classification. created by William Vickers," 2021. [Online]. Available: http://www.timeseriesclassification.com/dataset.php

[34] D. Molchanov, A. Ashukha, and D. Vetrov, "Variational dropout sparsifies deep neural networks," in *Proceedings of the 34th International Conference on International Conference on Machine Learning, (ICML)*, 2017, pp. 2498–2507.

[35] E. Lobacheva, N. Chirkova, and D. Vetrov, "Bayesian sparsification of gated recurrent neural networks," in *Proceedings of the Workshop on Compact Deep Neural Networks with industrial applications, NIPS*, 2018.

[36] S. Goel and R. Bajpai, "Impact of uncertainty in the input variables and model parameters on predictions of a long short term memory (LSTM) based sales forecasting model," *Machine Learning and Knowledge Extraction*, vol. 2, pp. 256–270, 2020.

[37] L. Zhu and N. Laptev, "Deep and confident prediction for time series at uber," in *IEEE International Conference on Data Mining Workshops (ICDMW)*, Nov 2017.

[38] Z. Gan, C. Li, C. Chen, Y. Pu, Q. Su, and L. Carin, "Scalable Bayesian learning of recurrent neural networks for language modeling," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL, Vancouver, Canada, July 30 - August 4, Volume 1*, 2017, pp. 321–331.

[39] S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski, "Deep state space models for time series forecasting," in *Proceedings of the 31st International Conference on Neural Information Processing Systems, (NIPS)*, vol. 31, 2018.

[40] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "DeepAR: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020.

[41] A. Alaa and M. Van Der Schaar, "Frequentist uncertainty in recurrent neural networks via blockwise influence functions," in *Proceedings of the 37th International Conference on International Conference on Machine Learning, (ICML)*, 2020, pp. 175–190.

[42] S. Krstanovic and H. Paulheim, "Ensembles of recurrent neural networks for robust time series forecasting," in *International Conference on Innovative Techniques and Applications of Artificial Intelligence*. Springer, 2017, pp. 34–46.

[43] B. Wang, J. Lu, Z. Yan, H. Luo, T. Li, Y. Zheng, and G. Zhang, "Deep uncertainty quantification: A machine learning approach for weather forecasting," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

[44] T. Song, W. Ding, H. Liu, J. Wu, H. Zhou, and J. Chu, "Uncertainty quantification in machine learning modeling for multi-step time series forecasting: Example of recurrent neural networks in discharge simulations," *Water*, vol. 12, no. 3, 2020.

[45] A. Khosravi, S. Nahavandi, D. Creighton, and A. F. Atiya, "Comprehensive review of neural network-based prediction intervals and new advances," *IEEE Transactions on neural networks*, vol. 22, no. 9, pp. 1341–1356, 2011.

[46] H. D. Kabir, A. Khosravi, M. A. Hosen, and S. Nahavandi, "Neural network-based uncertainty quantification: A survey of methodologies and applications," *IEEE access*, vol. 6, pp. 36 218–36 234, 2018.

[47] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[48] A. Papoulis and S. U. Pillai, *Probability, Random Variables, and Stochastic Processes*, 4th ed. McGraw-Hill Higher Education, 2002.

[49] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[50] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," in *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*, 2014.

[51] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[52] A. Karpathy, "The unreasonable effectiveness of recurrent neural networks," May 2015. [Online]. Available: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

[53] F. Gers and J. Schmidhuber, "Recurrent nets that time and count," *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, vol. 3, pp. 189–194 vol.3, 2000.

[54] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Computation*, vol. 12, pp. 2451–2471, 1999.

[55] Y. Lu and F. M. Salem, "Simplified gating in long short-term memory (LSTM) recurrent neural networks," in *IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2017, pp. 1601–1604.

[56] X. SHI, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. WOO, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Proceedings of the 28th International Conference on Neural Information Processing Systems, (NIPS)*, 2015.

[57] K. Yao, T. Cohn, K. Vylomova, K. Duh, and C. Dyer, "Depth-gated LSTM," *arXiv preprint arXiv:1508.03790*, 2015.

[58] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Gated feedback recurrent neural networks," in *International conference on machine learning*, 2015, pp. 2067–2075.

[59] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," *arXiv preprint arXiv:1503.00075*, 2015.

[60] P. Liu, X. Qiu, J. Chen, and X.-J. Huang, "Deep fusion LSTMs for text semantic matching," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016, pp. 1034–1043.

[61] R. Dey and F. M. Salem, "Gate-variants of gated recurrent unit (GRU) neural networks," in *IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, 2017, pp. 1597–1600.

[62] A. H. Mirza, "Variants of combinations of additive and multiplicative updates for GRU neural networks," in *IEEE 26th Signal Processing and Communications Applications Conference (SIU)*, 2018, pp. 1–4.

[63] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[64] F. Gers, "Long short-term memory in recurrent neural networks," Ph.D. dissertation, Verlag nicht ermittelbar, 2001.

[65] J. Zhang, T. He, S. Sra, and A. Jadbabaie, "Why gradient clipping accelerates training: A theoretical justification for adaptivity," in *Proceedings of 7th International Conference on Learning Representations, (ICLR)*, 2019.

[66] A. H. Ribeiro, K. Tiels, L. A. Aguirre, and T. Schön, "Beyond exploding and vanishing gradients: analysing RNN training using attractors and smoothness," in *International Conference on Artificial Intelligence and Statistics*, 2020, pp. 2370–2380.

[67] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of 3th International Conference on Learning Representations, (ICLR)*, 2015.

[68] M. Abadi, *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/schedules/PolynomialDecay

[69] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proceedings of 3rd International Conference on Learning Representations, (ICLR)*, 2015.

[70] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Proceedings of 5th International Conference on Learning Representations, (ICLR) workshop track*, 2017.

[71] M. Cuturi, "Fast global alignment kernels," in *Proceedings of the 28th international conference on machine learning (ICML)*, 2011.

[72] T. Gneiting and A. E. Raftery, "Strictly proper scoring rules, prediction, and estimation," *Journal of the American statistical Association*, vol. 102, no. 477, pp. 359–378, 2007.

[73] G. Biau and B. Patra, "Sequential quantile prediction of time series," *IEEE Transactions on Information Theory*, vol. 57, pp. 1664–1674, 2011.

Dimah Dera is an Assistant Professor in the Chester F. Carlson Center for Imaging Science at the Rochester Institute of Technology. She received her Ph.D. and M.S. in Electrical and Computer Engineering and M.A. in Mathematics from Rowan University. Dimah received the National Science Foundation (NSF) Computer and Information Science and Engineering Research Initiation Initiative (CRII) and NSF Research Experiences for Undergraduates (REU) supplement awards in 2022 for her current research focusing on robust machine learning and time-series analysis. She also received the Faculty Excellence Award in 2023 and the Fred W. and Frances H. Rusteberg Endowment Fellowship Award in 2022. She won the Best Student Paper Award at the 2019 IEEE International Workshop on Machine Learning for Signal Processing (MLSP'19) and the Runner-up Best Paper Award at the 2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM'15). She is the recipient of the NJ Health Foundation Research Grant award (2021), the IEEE Philadelphia Sections Benjamin Franklin Key Award (2021), the STEM Innovator to Watch Award from the NJ Tech Council (2018) and the NSF iREDEFINE Professional Development Award (2017).

Sabeen Ahmed is a Ph.D. candidate in the Electrical Engineering department of the University of South Florida. She has work experience of fifteen years in the telecom technology industry, including leadership roles. Her focus of research is Machine Learning and Deep Learning. Her work on improving the performance and credibility of deep learning models in oncological settings. Her paper titled; 'Failure Detection in Deep Neural Networks for Medical Imaging' was recently published in Frontiers in Medical Technology. She has an accepted tutorial paper on Transformers for time series in Circuits, Systems, and Signal Processing.

Nidhal Carla Bouaynaya holds a Ph.D. in Electrical and Computer Engineering (ECE) and an M.S. in Pure Mathematics from the University of Illinois at Chicago. She is a Professor of ECE and the Director of Rowan's Machine and Artificial Intelligence Virtual Reality Center (MAIVRC). She is currently serving as the Associate Dean for Research and Graduate Studies at the Henry M. Rowan College of Engineering. Her research interests are in Statistical Learning and Mathematical Optimization. She co-authored more than 100 refereed journal articles, book chapters, and conference proceedings, such as IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Signal Processing Letters, IEEE Signal Processing Magazine, and PLOS Medicine. Bouaynaya won numerous Best Paper Awards. The most recent was at the 2019 IEEE International Workshop on Machine Learning for Signal Processing.

Ghulam Rasool is an Assistant Member in the Department of Machine Learning at the H. Lee Moffitt Cancer Center and Research Institute, Tampa, FL. He received a BS in Mechanical Engineering from the National University of Sciences and Technology (NUST), Pakistan, in 2000, an M.S. in Computer Engineering from the Center for Advanced Studies in Engineering (CASE), Pakistan, in 2010, and a Ph.D. in Systems Engineering from the University of Arkansas at Little Rock in 2014. He was a post-doctoral fellow with the Rehabilitation Institute of Chicago and Northwestern University from 2014 to 2016. Before joining Moffitt, he was an Assistant Professor at the Department of Electrical and Computer Engineering at Rowan University. His current research focuses on building trustworthy multimodal machine learning and artificial intelligence model for cancer diagnosis, treatment planning, and risk assessment. His research efforts are currently funded by two National Science Foundation awards (NSF) awards. Previously his research was supported by the National Institute of Health (NIH), U.S. Department of Education, NSF, the New Jersey Health Foundation (NJHF), Google, NVIDIA, and Lockheed Martin, Inc