ARTICLE IN PRESS

International Journal of Forecasting xxx (xxxx) xxx

Contents lists available at ScienceDirect

International Journal of Forecasting

journal homepage: www.elsevier.com/locate/ijforecast

Coupling LSTM neural networks and state-space models through analytically tractable inference

Van-Dai Vuong *, Luong-Ha Nguyen, James-A. Goulet

Department of Civil, Geologic and Mining Engineering, Polytechnique Montréal, 2500 chemin de Polytechnique, Montréal, Québec, H3T 1J4, Canada

ARTICLE INFO

Keywords:
Bayesian inference
Probabilistic method
Long short-term memory
State-space models
Time series forecasting
Hybrid model

ABSTRACT

Long short-term memory (LSTM) neural networks and state-space models (SSMs) are effective tools for time series forecasting. Coupling these methods to exploit their advantages is not a trivial task because their respective inference procedures rely on different mechanisms. In this paper, we present formulations that allow for analytically tractable inference in Bayesian LSTMs and the probabilistic coupling between Bayesian LSTMs and SSMs. This is enabled by using analytical Gaussian inference as a single mechanism for inferring both the LSTM's parameters as well as the posterior for the SSM's hidden states. We show through several experimental comparisons that the resulting hybrid model retains the interpretability feature of SSMs, while exploiting the ability of LSTMs to learn complex seasonal patterns with minimal manual setups.

1. Introduction

Time series forecasting plays a key role in various industries such as finance, transportation, energy, infrastructure and healthcare. Recurrent neural networks (RNNs) (Rumelhart et al., 1986) and especially the long short-term memory (LSTM) architecture (Hochreiter & Schmidhuber, 1997) have been shown to be effective tools for time series forecasting problems (Hewamalage et al., 2021). The main strength of neural networks (NNs) is their ability to automatically identify complex patterns while requiring minimal manual setups (Januschowski et al., 2020). Nevertheless, without making assumptions about the data, the results from NNs are typically difficult to interpret (Rangapuram et al., 2018). By contrast, state-space models (SSMs) (Durbin & Koopman, 2001) provide a probabilistic framework for decomposing time series into interpretable patterns such as level, trend and seasonality.

In this paper, we propose a new method that allows coupling Bayesian LSTMs with SSMs to obtain a hybrid model that retains the interpretability feature of SSMs, while exploiting the ability of LSTMs to learn complex patterns automatically. Coupling these methods is not a trivial task because their respective inference procedures rely on different mechanisms. SSMs are probabilistic models which rely on Bayesian inference, whereas LSTMs typically optimize their parameters using backpropagation (Rumelhart et al., 1986) and gradient descent (GD). As a result, one needs to choose between the two mechanisms when coupling these models. Typically, existing hybrid models including those proposed by Smyl (2020) and Rangapuram et al. (2018) have relied on backpropagation and GD. In this paper, we explore the probabilistic coupling between LSTMs and SSMs, while using Bayesian inference for inferring both the LSTM's parameters as well as the posterior for SSM's hidden states.

Recently, the Tractable Approximate Gaussian Inference (TAGI) method (Goulet et al., 2021) has allowed analytical Bayesian inference in neural networks. The performance of TAGI has been demonstrated on feedforward

* Corresponding author.
E-mail address: van-dai.vuong@polymtl.ca (V.-D. Vuong).

https://doi.org/10.1016/j.ijforecast.2024.04.002

Please cite this article as: V.-D. Vuong, L.-H. Nguyen and J.-A. Goulet, Coupling LSTM neural networks and state-space models through analytically tractable inference. International Journal of Forecasting (2024), https://doi.org/10.1016/j.ijforecast.2024.04.002.

neural networks (FNNs) (Goulet et al., 2021), convolutional neural networks (CNNs), generative adversarial networks (GANs) (Nguyen & Goulet, 2021), and reinforcement learning (RL) (Nguyen & Goulet, 2022), yet it has never been tested for RNNs due to the lack of a mathematical framework to do so. In this paper, we present the mathematical formulations which enable using the TAGI method with the LSTM architecture for obtaining analytically the posterior mean vectors and diagonal covariance matrices for the latent variables and model parameters. Because all variables in this TAGI-LSTM are now Gaussians, this new framework can thus be directly coupled with SSMs.

The paper is organized as follows: Section 2 reviews Bayesian RNNs along with hybrid models which couple RNNs with SSMs. Section 3 reviews the TAGI method along with the LSTM architecture which are the foundations for this work. Section 4 first presents the mathematical formulations that have been developed to apply TAGI to the LSTM architecture, and then introduces the probabilistic coupling between TAGI-LSTM and SSMs. Finally, Section 5 compares our methods with other benchmark models on both simulated and real datasets.

The following notation is used throughout the manuscript; x: lowercases denote variables; $\boldsymbol{x}$: bold lowercases denote vectors; X: italic uppercases denote random variables; $\boldsymbol{X}$: bold italic uppercases denote vectors or matrices of random variables; $\mathbf{X}$: bold upright uppercases denote deterministic matrices; $\boldsymbol{y}_{1:t}$: denote observations from 1 to t; $\boldsymbol{\mu}$ and $\mathbb{E}[\cdot]$: mean vectors; $\boldsymbol{\Sigma}$: covariance matrices; $\boldsymbol{X}_{t|t-1} \sim \mathcal{N}(\boldsymbol{x}_t; \boldsymbol{\mu}^{\boldsymbol{X}}_{t|t-1}, \boldsymbol{\Sigma}^{\boldsymbol{X}}_{t|t-1}) \equiv \mathcal{N}(\boldsymbol{\mu}^{\boldsymbol{X}}_{t|t-1}, \boldsymbol{\Sigma}^{\boldsymbol{X}}_{t|t-1})$: prior probability density functions (PDFs) for the random variables $\boldsymbol{X}_t$ given data $\boldsymbol{y}_{1:t-1}$ with $\boldsymbol{\mu}^{\boldsymbol{X}}_{t|t-1} \equiv \mathbb{E}[\boldsymbol{X}_t|\boldsymbol{y}_{1:t-1}]$ and $\boldsymbol{\Sigma}^{\boldsymbol{X}}_{t|t-1} \equiv \text{cov}(\boldsymbol{X}_t|\boldsymbol{y}_{1:t-1})$; $\boldsymbol{X}_{t|t} \sim \mathcal{N}(\boldsymbol{x}_t; \boldsymbol{\mu}^{\boldsymbol{X}}_{t|t}, \boldsymbol{\Sigma}^{\boldsymbol{X}}_{t|t})$: posterior PDFs for the random variables $\boldsymbol{X}_t$ given data $\boldsymbol{y}_{1:t}$; and $\boldsymbol{X}_{t|\mathrm{T}} \sim \mathcal{N}(\boldsymbol{x}_t; \boldsymbol{\mu}^{\boldsymbol{X}}_{t|\mathrm{T}}, \boldsymbol{\Sigma}^{\boldsymbol{X}}_{t|\mathrm{T}})$: smoothed estimates for the random variables $\boldsymbol{X}_t$ given data $\boldsymbol{y}_{1:\mathrm{T}}$ where T is the last training time step.

2. Related works

In this section, we review Bayesian recurrent neural networks (RNNs) along with hybrid models combining RNNs with state-space models (SSMs).

2.1. Bayesian recurrent neural networks

Bayesian NNs place a prior distribution over the parameters $f(\boldsymbol{\theta})$, and the goal is to find the posterior distribution $f(\boldsymbol{\theta}|\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x}$ are the input covariates and $\boldsymbol{y}$ are the observed responses. This posterior is widely acknowledged as being intractable for NNs (Goodfellow et al., 2016), thus methods such as the Laplace approximation (MacKay, 1992) and variational inference (VI) (Hernandez-Lobato & Adams, 2015; Sun et al., 2017; Wu et al., 2019) have been proposed to approximate it. Chien and Ku (2016) developed an approximation for the Hessian matrix in order to use the Laplace approximation in Bayesian RNNs. Fortunato et al. (2019) applied the variational scheme Bayes by Backprop (Blundell et al., 2015) to RNNs for estimating the parameters' posterior distribution. In a general context, VI has been shown to have a limited scalability to large datasets (Hernandez-Lobato & Adams, 2015). A part of the reason for it is that multiple passes through data are required for optimization, and there are typically more parameters in the VI-based RNNs than the deterministic ones sharing the same architecture.

Gal and Ghahramani (2016a) have approached Bayesian RNNs in a different way. They showed that performing Monte Carlo dropout (MC-dropout) and weight decay in a deterministic RNN trained with backpropagation and GD enables providing accurate predictive uncertainties. Creating Bayesian RNNs by applying MC-dropout to a deterministic RNN is straightforward, yet it cannot quantify the posterior for the parameters. In this approach, dropout can either be applied to only the non-recurrent connections (Pham et al., 2014; Zaremba et al., 2014), to the LSTM cell states (Moon et al., 2015), or to all of the input, output and recurrent connections (Gal & Ghahramani, 2016a).

The abovementioned Bayesian RNN methods rely on backpropagation and GD to optimize their parameters. In this paper, we provide a mathematical framework that allows applying the TAGI method to the LSTM architecture for quantifying the parameters' posterior distribution analytically using Bayesian inference.

2.2. Hybrid models

Some studies have established hybrid models combining RNNs and SSMs by using the former ones to either model nonlinear SSM's transition and/or observation equations, or define the SSM's parameters. Krishnan et al. (2017), Zheng et al. (2017) and Zaheer et al. (2017) have used RNNs to model the nonlinear transition of the hidden states, allowing their models to be non-Markovian. However, the common limitation among methods using NNs to parametrize the SSM's nonlinear dynamic models is that the marginal log-likelihood function is intractable so that they rely on either Monte Carlo (MC) methods (Zheng et al., 2017) or variational inference (Fraccaro et al., 2016; Krishnan et al., 2017) to approximate this function in order to update their neural network parameters. Fraccaro et al. (2016) and Rangapuram et al. (2018) have kept the SSM's transition and observation models linear which allows performing exact inference. The DeepState method (Rangapuram et al., 2018) uses a global LSTM to learn from the whole dataset and a local linear SSM for each time series. The global LSTM outputs the SSM's time-varying parameters defining the model matrices for each local SSM. In a different approach, Smyl (2020) used the deterministic exponential smoothing equations as a data processing tool for a LSTM. The level and seasonality components are estimated for each time series, and are used to normalize and deseasonalize data on-the-fly. However, the use of the deterministic form of exponential smoothing leads to point estimates for the baseline level and trend components.

The existing methods mentioned above have already attempted to couple SSMs with the deterministic RNNs. In this paper, we explore another direction by providing a probabilistic coupling between Bayesian LSTMs and SSMs.

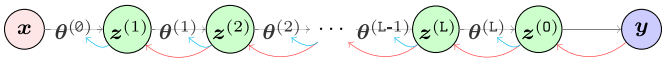

Fig. 1. Graphical representation of a feedforward neural network. Black arrows represent the network forward connections; red and cyan arrows represent the inference directions for the hidden states and for the parameters, respectively.

3. Background

In this section, we review the theoretical basics of the LSTM architecture and the fundamentals of the TAGI method.

3.1. Long short-term memory neural network

In the context of univariate time series, dependencies are the relations between the current observation and the past ones. RNNs typically experience vanishing and exploding gradients which limit their capacity to learn long-term dependencies (Goodfellow et al., 2016). The long short-term memory (LSTM) network (Hochreiter & Schmidhuber, 1997) is a special type of RNN designed to address this problem. LSTMs use gating mechanisms to automatically select and store the dependency information in their memory. There are two kinds of memory in a LSTM cell, the hidden states $\boldsymbol{h}$ encode the short-term dependency information, whereas the cell states $\boldsymbol{c}$ store the long-term one. A LSTM cell consists in four gates including the forget, input, output and candidate gates, $\{\boldsymbol{f}, \boldsymbol{i}, \boldsymbol{o}, \tilde{\boldsymbol{c}}\} \in \mathbb{R}^{\mathtt{N}}$, the cell states $\boldsymbol{c} \in \mathbb{R}^{\mathtt{N}}$, and the hidden states $\boldsymbol{h} \in \mathbb{R}^{\mathtt{N}}$, where $\mathtt{N}$ is the number of hidden units. All operations in a LSTM cell can be summarized by the following LSTM recursive equations

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}^f \boldsymbol{x}_t + \boldsymbol{U}^f \boldsymbol{h}_{t-1} + \boldsymbol{b}^f), \tag{1a}$$

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}^i \boldsymbol{x}_t + \boldsymbol{U}^i \boldsymbol{h}_{t-1} + \boldsymbol{b}^i), \tag{1b}$$

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}^o \boldsymbol{x}_t + \boldsymbol{U}^o \boldsymbol{h}_{t-1} + \boldsymbol{b}^o), \tag{1c}$$

$$\tilde{\boldsymbol{c}}_t = \tanh(\boldsymbol{W}^c \boldsymbol{x}_t + \boldsymbol{U}^c \boldsymbol{h}_{t-1} + \boldsymbol{b}^c), \tag{1d}$$

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t, \tag{1e}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t), \tag{1f}$$

where $\boldsymbol{x}_t \in \mathbb{R}^{\mathtt{M}}$ is the input vector, $\{\boldsymbol{W}^f, \boldsymbol{W}^i, \boldsymbol{W}^o, \boldsymbol{W}^c\} \in \mathbb{R}^{\mathtt{M}\times\mathtt{N}}$ are the weight matrices for the input, $\{\boldsymbol{U}^f, \boldsymbol{U}^i, \boldsymbol{U}^o, \boldsymbol{U}^c\} \in \mathbb{R}^{\mathtt{N}\times\mathtt{N}}$ are the weight matrices for the hidden states, $\{\boldsymbol{b}^f, \boldsymbol{b}^i, \boldsymbol{b}^o, \boldsymbol{b}^c\} \in \mathbb{R}^{\mathtt{N}}$ are the bias vectors, with $\mathtt{M}$ being the input vector's length, $\sigma(\cdot)$ is the logistic sigmoid function and $\tanh(\cdot)$ is the hyperbolic tangent function, and $\odot$ denotes the element-wise multiplication operation.

A LSTM cell takes the input $\boldsymbol{x}_t$ and the hidden states $\boldsymbol{h}_{t-1}$ from the previous time step as inputs (Eqs. (1a)–(1d)), and outputs the hidden states $\boldsymbol{h}_t$ (Eq. (1f)) which are then used to make final predictions. The cell and the hidden states, $\boldsymbol{c}_t$ and $\boldsymbol{h}_t$, are updated at every time step in order to retain relevant information which is useful for current and future predictions. The updating process for the cell states $\boldsymbol{c}_t$ consists of two steps, and is done using Eq. (1e). The first step is to discard irrelevant information from the previous cell states $\boldsymbol{c}_{t-1}$, by performing the element-wise product $\boldsymbol{f}_t \odot \boldsymbol{c}_{t-1}$. The second step consists in including the new information into the cell states $\boldsymbol{c}_t$ by adding the element-wise product $\boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t$. After updating the cell states $\boldsymbol{c}_t$, the hidden states $\boldsymbol{h}_t$ are updated using Eq. (1f).

3.2. Tractable approximate Gaussian inference

Tractable Approximate Gaussian Inference (TAGI) (Goulet et al., 2021) is an analytically tractable framework for Bayesian neural networks consisting in two main steps: forward and backward. The forward step propagates the uncertainties from the input layer and the network parameters up to the output layer. The backward step employs a layer-wise recursive procedure to infer analytically the parameters $\boldsymbol{\theta}$ and the hidden states $\boldsymbol{Z}$ from the observations $\boldsymbol{y}$.

Fig. 1 presents a compact graphical representation for a feedforward neural network where $\boldsymbol{x}$ is the input vector; $\boldsymbol{z}^{(j)}$ and $\boldsymbol{\theta}^{(j)} = \{\boldsymbol{W}^{(j)}, \boldsymbol{B}^{(j)}\}$, $\forall j = 1 : \mathtt{L}$, are the hidden states and parameters of the j^{th} hidden layer; $\boldsymbol{z}^{(0)}$ is the output vector; $\boldsymbol{y}$ are the observations; and $\mathtt{L}$ is the number of hidden layers. TAGI considers the network's input, output, hidden states as well as parameters as Gaussian random variables such that $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}_X, \boldsymbol{\Sigma}_X)$, $\boldsymbol{Z} \sim \mathcal{N}(\boldsymbol{\mu}_Z, \boldsymbol{\Sigma}_Z)$, and $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}_\theta, \boldsymbol{\Sigma}_\theta)$.

The j^{th} hidden layer contains the hidden units $\boldsymbol{z}^{(j)}$ and their activation units $\boldsymbol{a}^{(j)}$ such that $\boldsymbol{a}^{(j)} = \phi(\boldsymbol{z}^{(j)})$, where $\phi(\cdot)$ is the activation function. The forward step starts by passing information from the input layer to the first hidden layer using

$$\boldsymbol{Z}^{(1)} = \boldsymbol{W}^{(0)} \boldsymbol{X} + \boldsymbol{B}^{(0)}, \tag{2}$$

$$\boldsymbol{A}^{(1)} = \phi(\boldsymbol{Z}^{(1)}). \tag{3}$$

Eq. (2) involves two types of operations including additions and multiplications of Gaussian random variables. Because TAGI assumes that the hidden states $\boldsymbol{Z}^{(1)}$ are Gaussians, these two operations must also result in Gaussian random variables, even if the multiplication operation does not lead to exact results. In order to overcome this problem, TAGI relies on the *Gaussian multiplication approximation* (GMA) to model the product of two Gaussian random variables by a Gaussian distribution whose exact mean and variance are calculated analytically as presented in Appendix A. In addition, the output moments for the nonlinear activation function $\phi(\cdot)$ in Eq. (3) cannot be evaluated analytically; TAGI uses the local linearization of activation functions $\tilde{\phi}(\cdot)$ to obtain the output moments as detailed in Appendix B. Using these approximations allows calculating analytically the mean vectors and covariance matrices defining the PDFs for the hidden and activation units. Similarly, going from a j^{th} layer to the subsequent $j + 1^{th}$ is done using the same Eqs. (2) and (3), while replacing the input $\boldsymbol{X}$ by the activation units $\boldsymbol{A}^{(j)} \sim \mathcal{N}(\boldsymbol{\mu}_A^{(j)}, \boldsymbol{\Sigma}_A^{(j)})$, and $\boldsymbol{\theta}^{(0)}$ by the parameters $\boldsymbol{\theta}^{(j)} \sim \mathcal{N}(\boldsymbol{\mu}_\theta^{(j)}, \boldsymbol{\Sigma}_\theta^{(j)})$ of the j^{th} layer. This allows propagating the uncertainties from the input $\boldsymbol{X}$ and the parameters $\boldsymbol{\theta}$ through the network up to the output layer $\boldsymbol{Z}^{(0)}$.

The relation between the output layer $\boldsymbol{z}^{(0)}$ and the observations $\boldsymbol{y}$ is defined by the observation equation as

$$\boldsymbol{y} = \boldsymbol{z}^{(0)} + \boldsymbol{v}, \quad \boldsymbol{v} : \boldsymbol{V} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_{\boldsymbol{V}}), \tag{4}$$

where $\boldsymbol{v}$ denotes the observation error. The moments of the predictive distribution $\boldsymbol{Y} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{Y}}, \boldsymbol{\Sigma}_{\boldsymbol{Y}})$ can be obtained following $\boldsymbol{\mu}_{\boldsymbol{Y}} = \boldsymbol{\mu}_{\boldsymbol{Z}}^{(0)}$, $\boldsymbol{\Sigma}_{\boldsymbol{Y}} = \boldsymbol{\Sigma}_{\boldsymbol{Z}}^{(0)} + \boldsymbol{\Sigma}_{\boldsymbol{V}}$.

In the context of TAGI, inferring the hidden states and parameters means estimating the conditional probability distributions $f(\boldsymbol{z}|\boldsymbol{y}) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{Z}|\boldsymbol{y}}, \boldsymbol{\Sigma}_{\boldsymbol{Z}|\boldsymbol{y}})$ and $f(\boldsymbol{\theta}|\boldsymbol{y}) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}|\boldsymbol{y}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}|\boldsymbol{y}})$. The backward step first computes the posterior for the output $\boldsymbol{Z}^{(0)}$ using the Gaussian conditional equations

$$\begin{aligned} f(\boldsymbol{z}^{(0)} \mid \boldsymbol{y}) &= \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{Z}^{(0)}|\boldsymbol{y}}, \boldsymbol{\Sigma}_{\boldsymbol{Z}^{(0)}|\boldsymbol{y}}), \\ \boldsymbol{\mu}_{\boldsymbol{Z}^{(0)}|\boldsymbol{y}} &= \boldsymbol{\mu}_{\boldsymbol{Z}^{(0)}} + \boldsymbol{\Sigma}_{\boldsymbol{Y}\boldsymbol{Z}^{(0)}}^{\intercal} \boldsymbol{\Sigma}_{\boldsymbol{Y}}^{-1} (\boldsymbol{y} - \boldsymbol{\mu}_{\boldsymbol{Y}}), \\ \boldsymbol{\Sigma}_{\boldsymbol{Z}^{(0)}|\boldsymbol{y}} &= \boldsymbol{\Sigma}_{\boldsymbol{Z}^{(0)}} - \boldsymbol{\Sigma}_{\boldsymbol{Y}\boldsymbol{Z}^{(0)}}^{\intercal} \boldsymbol{\Sigma}_{\boldsymbol{Y}}^{-1} \boldsymbol{\Sigma}_{\boldsymbol{Y}\boldsymbol{Z}^{(0)}}, \end{aligned} \tag{5}$$

where $\boldsymbol{\Sigma}_{\boldsymbol{Y}\boldsymbol{Z}^{(0)}} = \boldsymbol{\Sigma}_{\boldsymbol{Z}^{(0)}}$. This is depicted by the red arrow from $\boldsymbol{y}$ to $\boldsymbol{z}^{(0)}$ in Fig. 1. The layer-wise recursive inference procedure is then recursively applied to infer the hidden states and parameters of each layer from the last to the first layer as presented by the red and cyan arrows in Fig. 1. For maintaining the linear complexity with respect to the number of parameters in the network, TAGI assumes diagonal covariance matrices for the hidden states and the parameters, and relies on the conditional independence assumptions of hidden units between layers, i.e., $\boldsymbol{Z}^{(j-1)} \perp\!\!\!\perp \boldsymbol{Z}^{(j+1)} | \boldsymbol{z}^{(j)}$. The posterior distribution for the hidden states of each layer $\boldsymbol{Z}^{(j)}$ is estimated analytically using

$$\begin{aligned} f(\boldsymbol{z}^{(j)}|\boldsymbol{y}) &= \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{Z}|\boldsymbol{y}}^{(j)}, \boldsymbol{\Sigma}_{\boldsymbol{Z}|\boldsymbol{y}}^{(j)}), \\ \boldsymbol{\mu}_{\boldsymbol{Z}|\boldsymbol{y}}^{(j)} &= \boldsymbol{\mu}_{\boldsymbol{Z}}^{(j)} + \mathbf{J}_{\boldsymbol{Z}} (\boldsymbol{\mu}_{\boldsymbol{Z}|\boldsymbol{y}}^{(j+1)} - \boldsymbol{\mu}_{\boldsymbol{Z}}^{(j+1)}), \\ \boldsymbol{\Sigma}_{\boldsymbol{Z}|\boldsymbol{y}}^{(j)} &= \boldsymbol{\Sigma}_{\boldsymbol{Z}}^{(j)} + \mathbf{J}_{\boldsymbol{Z}} (\boldsymbol{\Sigma}_{\boldsymbol{Z}|\boldsymbol{y}}^{(j+1)} - \boldsymbol{\Sigma}_{\boldsymbol{Z}}^{(j+1)}) \mathbf{J}_{\boldsymbol{Z}}^{\intercal}, \\ \mathbf{J}_{\boldsymbol{Z}} &= \mathrm{cov}(\boldsymbol{Z}^{(j)}, \boldsymbol{Z}^{(j+1)}) (\boldsymbol{\Sigma}_{\boldsymbol{Z}}^{(j+1)})^{-1}, \end{aligned} \tag{6}$$

where $\mathrm{cov}(\boldsymbol{Z}^{(j)}, \boldsymbol{Z}^{(j+1)})$ is the covariance between the hidden states of the j^{th} and $j+1^{th}$ layers. In parallel, the parameters $\boldsymbol{\theta}^{(j)}$ can be inferred using the same Eq. (6) where the variable $\boldsymbol{Z}^{(j)}$ is replaced by $\boldsymbol{\theta}^{(j)}$. The calculations for obtaining the covariances $\mathrm{cov}(\boldsymbol{Z}^{(j)}, \boldsymbol{Z}^{(j+1)})$ and $\mathrm{cov}(\boldsymbol{\theta}^{(j)}, \boldsymbol{Z}^{(j+1)})$, as well as other quantities in Eq. (6) have been detailed by Goulet et al. (2021).

A weakly informative prior is employed when initializing the hyper-parameters $\{\boldsymbol{\mu}_{\boldsymbol{\theta}}^{(\emptyset)}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{(\emptyset)}\}$. In order to learn the parameters efficiently, TAGI repeats the inference over multiple epochs, analogously to the empirical Bayes approach (Efron, 2010). This means that the posterior hyper-parameters $\{\boldsymbol{\mu}_{\boldsymbol{\theta}|\boldsymbol{y}}^{(i)}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}|\boldsymbol{y}}^{(i)}\}$ at the i^{th} epoch are used as the prior hyper-parameters for the next $i+1^{th}$ epoch.

TAGI can theoretically be used with any existing NN architecture to create the corresponding Bayesian neural network. However, this has never been done for RNNs. In this paper, we provide the mathematical formulations to use TAGI with the LSTM architecture.

4. Methods

In this section, we first present the analytically tractable TAGI-LSTM neural network, then introduce the probabilistic coupling between it and SSMs to create TAGI-LSTM/SSM hybrid models.

4.1. TAGI-LSTM

This section introduces the mathematical formulations for the TAGI-LSTM model where the LSTM's parameters and hidden states presented in Section 3.1 are inferred analytically using the TAGI method presented in Section 3.2. In order to apply the TAGI method to LSTM, we employ the same architecture formulations presented in Eq. (1) to calculate the gates $\{\boldsymbol{f}, \boldsymbol{i}, \tilde{\boldsymbol{c}}, \boldsymbol{o}\}$, the cell states $\boldsymbol{c}$, and the hidden states $\boldsymbol{h}$, and we consider them, along with the network parameters $\boldsymbol{\theta}$, as Gaussian random variables. In order to maintain the linear computational complexity of the TAGI method, we need to rely on the same independence assumption employed by Goulet et al. (2021), that is considering a diagonal covariance structure for all LSTM's gates, hidden states, cell states, and parameters. Fig. 2a shows a graphical representation of a LSTM cell, whereas Fig. 2b presents the graph for an example of stacked TAGI-LSTM network having an input layer containing the covariates $\boldsymbol{x}$, two LSTM layers, and a fully connected output layer. The observations $\boldsymbol{y}$ are related to the outputs $\boldsymbol{z}^{(0)}$ through Eq. (4).

4.1.1. Forward step

At a time step t, we denote the marginal prior knowledge for the hidden states of a LSTM layer given the past data $\boldsymbol{y}_{1:t-1}$ by the Gaussian PDF $\boldsymbol{H}_{t|t-1} \sim \mathcal{N}(\boldsymbol{\mu}_{t|t-1}^{\boldsymbol{H}}, \boldsymbol{\Sigma}_{t|t-1}^{\boldsymbol{H}})$, where $\boldsymbol{\mu}_{t|t-1}^{\boldsymbol{H}} \equiv \mathbb{E}[\boldsymbol{H}_t|\boldsymbol{y}_{1:t-1}]$, and $\boldsymbol{\Sigma}_{t|t-1}^{\boldsymbol{H}} \equiv \mathrm{cov}(\boldsymbol{H}_t|\boldsymbol{y}_{1:t-1})$. In the forward step, we want to pass information from the input covariates $\boldsymbol{X}_t$ through the LSTM layers, up to the output layer. This corresponds to estimating the prior knowledge for the hidden and cell states of each LSTM layer, $\boldsymbol{H}_{t|t-1}^{(j)}$ and $\boldsymbol{C}_{t|t-1}^{(j)}$, as well as the prior $\boldsymbol{Z}_{t|t-1}^{(0)} \sim \mathcal{N}(\boldsymbol{\mu}_{t|t-1}^{\boldsymbol{Z}^{(0)}}, \boldsymbol{\Sigma}_{t|t-1}^{\boldsymbol{Z}^{(0)}})$ for the hidden states of the output layer.

In order to pass information through a LSTM layer, we first need to obtain the prior knowledge for the four LSTM gates. In the probabilistic context where all quantities are modelled by Gaussian random variables, we define the hidden states for the forget gate as

$$\boldsymbol{Z}_t^f = \boldsymbol{W}_t^f \boldsymbol{X}_t + \boldsymbol{U}_t^f \boldsymbol{H}_{t-1} + \boldsymbol{B}_t^f.$$

Each hidden state $Z_{i,t}^f$ (at time step t, and cell i) for the forget gate can be obtained as

$$Z_{i,t}^f = \boldsymbol{W}_{i,t}^f \boldsymbol{X}_t + \boldsymbol{U}_{i,t}^f \boldsymbol{H}_{t-1} + B_{i,t}^f, \tag{7}$$

where $i = 1 : \mathtt{N}$, $\boldsymbol{W}_{i,t}^f \in \mathbb{R}^{1\times \mathtt{M}}$, $\boldsymbol{U}_{i,t}^f \in \mathbb{R}^{1\times \mathtt{N}}$ are row weight matrices, $\mathtt{N}$ being the LSTM cell's size, and $\mathtt{M}$ is the input's size. We employ diagonal covariance structures for the PDFs of the input covariates $\boldsymbol{X}_t$ and the hidden states $\boldsymbol{H}_{t-1}$. Under this assumption, $Z_{i,t}^f$ is the sum of $\mathtt{M}+\mathtt{N}$ independent products and a bias term as shown in Eq. (7). Goulet et al. (2021) have shown that when adding a large number of independent product terms, the

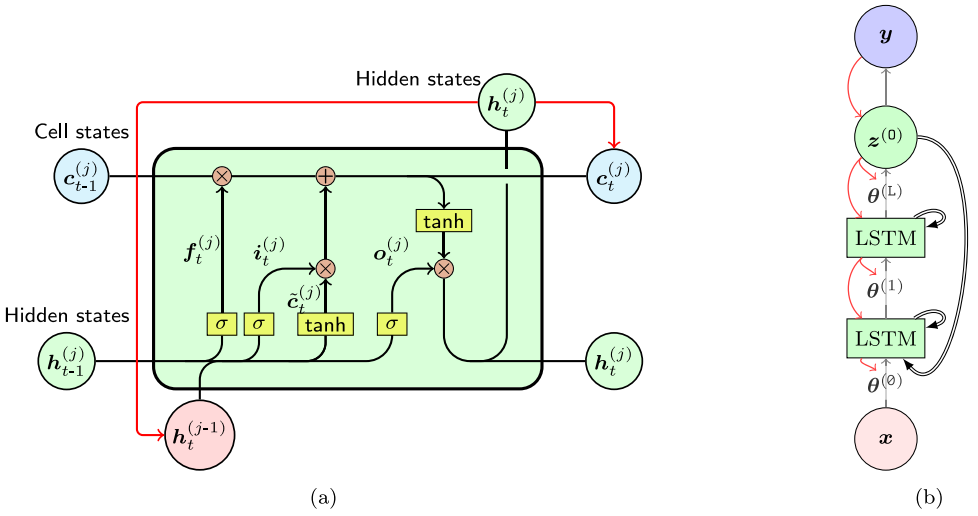

Fig. 2. (a) Graphical representation of a LSTM cell. Red arrows represent the inference procedure to update the $j-1$th LSTM layer from the subsequent jth LSTM layer. (b) Graphical representation of a stacked TAGI-LSTM network. Black arrows represent the network's forward connections, red arrows represent the layer-wise inference paths, and double arrows represent the recurrent connections.

correlation between the resulting pairs of output hidden states tends to zero such that assuming $Z^f_{i,t} \perp\!\!\!\perp Z^f_{j,t}$ is valid. Therefore, the hidden states for the forget gate $\boldsymbol{Z}^f_t$ are modelled by a Gaussian PDF having a diagonal covariance matrix, and the first two moments for its units are computed from

$$\begin{aligned}\mathbb{E}[Z^f_{i,t|t-1}] &= \mathbb{E}[\boldsymbol{W}^f_{i,t|t-1}\boldsymbol{X}_{t|t-1}] \\ &\quad +\mathbb{E}[\boldsymbol{U}^f_{i,t|t-1}\boldsymbol{H}_{t-1|t-1}] + \mathbb{E}[B^f_{i,t|t-1}], \\ \text{var}(Z^f_{i,t|t-1}) &= \text{var}(\boldsymbol{W}^f_{i,t|t-1}\boldsymbol{X}_{t|t-1}) \\ &\quad +\text{var}(\boldsymbol{U}^f_{i,t|t-1}\boldsymbol{H}_{t-1|t-1}) + \text{var}(B^f_{i,t|t-1}),\end{aligned}$$

where the means and variances of the product terms $\boldsymbol{W}^f_{i,t|t-1}\boldsymbol{X}_{t|t-1}$ and $\boldsymbol{U}^f_{i,t|t-1}\boldsymbol{H}_{t-1|t-1}$ are calculated exactly using the GMA equations given in Appendix A.

From Eq. (1a), we apply the locally linearized sigmoid activation function $\tilde{\sigma}(\cdot)$ to the hidden states of the forget gate to estimate its output values $\boldsymbol{F}_t = \tilde{\sigma}(\boldsymbol{Z}^f_t)$. As a result, the forget gate $\boldsymbol{F}_t$ also has a diagonal covariance matrix such that $F_{i,t} \perp\!\!\!\perp F_{j,t}$. The equations for obtaining its mean vector and covariance matrix are presented in Appendix B. Similarly, the prior knowledge for the other LSTM gates can be estimated using the same procedure. Because the equations used to calculate other gates $\{\boldsymbol{I}, \tilde{\boldsymbol{C}}, \boldsymbol{O}\}$ involve the same operations, i.e., the sum of several independent product terms, we can extend the independence assumption not only to these gates but also between all LSTM gates such that $I_{i,t} \perp\!\!\!\perp I_{j,t}$, $\tilde{C}_{i,t} \perp\!\!\!\perp \tilde{C}_{j,t}$, $O_{i,t} \perp\!\!\!\perp O_{j,t}$, and $\boldsymbol{F}_t \perp\!\!\!\perp \boldsymbol{I}_t \perp\!\!\!\perp \tilde{\boldsymbol{C}}_t \perp\!\!\!\perp \boldsymbol{O}_t$. As presented in Eq. (1e), the calculations of the cell states only involve element-wise operations from independent components. Therefore, all the cell states $C_{i,t}$ can be considered as independent and are obtained by

$$C_{i,t} = F_{i,t}C_{i,t-1} + I_{i,t}\tilde{C}_{i,t}.$$

Because Eqs. (1a)–(1d) which are used to estimate the LSTM gates $\{\boldsymbol{F}_t, \boldsymbol{I}_t, \tilde{\boldsymbol{C}}_t, \boldsymbol{O}_t\}$ do not involve the cell states $\boldsymbol{C}_{t-1}$, there is no direct information path between these gates and the cell states $\boldsymbol{C}_{t-1}$ other than through the hidden states $\boldsymbol{H}_{t-1}$. Therefore, the LSTM gates at time step t and the cell states at time step $t-1$ are conditionally independent given the hidden states $\boldsymbol{h}_{t-1}$, that is, $\boldsymbol{F}_t \perp\!\!\!\perp \boldsymbol{I}_t \perp\!\!\!\perp \tilde{\boldsymbol{C}}_t \perp\!\!\!\perp \boldsymbol{O}_t \perp\!\!\!\perp \boldsymbol{C}_{t-1} | \boldsymbol{h}_{t-1}$. Under this conditional independence assumption, we can apply the GMA equations to obtain the mean and variance for each cell state following

$$\begin{aligned}\mathbb{E}[C_{i,t|t-1}] &= \mathbb{E}[F_{i,t|t-1}] \cdot \mathbb{E}[C_{i,t-1|t-1}] \\ &\quad +\mathbb{E}[I_{i,t|t-1}] \cdot \mathbb{E}[\tilde{C}_{i,t|t-1}], \\ \text{var}(C_{i,t|t-1}) &= \text{var}(F_{i,t|t-1}) \cdot \text{var}(C_{i,t-1|t-1}) \\ &\quad +\text{var}(F_{i,t|t-1}) \cdot \mathbb{E}[C_{i,t-1|t-1}]^2 \\ &\quad +\text{var}(C_{i,t-1|t-1}) \cdot \mathbb{E}[F_{i,t|t-1}]^2 \\ &\quad +\text{var}(I_{i,t|t-1}) \cdot \text{var}(\tilde{C}_{i,t|t-1}) \\ &\quad +\text{var}(I_{i,t|t-1}) \cdot \mathbb{E}[\tilde{C}_{i,t|t-1}]^2 \\ &\quad +\text{var}(\tilde{C}_{i,t|t-1}) \cdot \mathbb{E}[I_{i,t|t-1}]^2.\end{aligned}$$

From Eq. (1f), the LSTM hidden states are obtained from $H_{i,t} = O_{i,t} \cdot \tilde{\tanh}(C_{i,t})$, where $\tilde{\tanh}(\cdot)$ is the locally linearized hyperbolic tangent activation function. Following the same reasoning used for the cell states, the hidden states $H_{i,t}$ are also considered as independent. Their mean and variance are estimated by

$$\begin{aligned}\mathbb{E}[H_{i,t|t-1}] &= \mathbb{E}[O_{i,t|t-1}] \cdot \mathbb{E}[\tilde{\tanh}(C_{i,t|t-1})], \\ \text{var}(H_{i,t|t-1}) &= \text{var}(O_{i,t|t-1}) \cdot \text{var}(\tilde{\tanh}(C_{i,t|t-1})) \\ &\quad +\text{var}(O_{i,t|t-1}) \cdot \mathbb{E}[\tilde{\tanh}(C_{i,t|t-1})]^2 \\ &\quad +\text{var}(\tilde{\tanh}(C_{i,t|t-1})) \cdot \mathbb{E}[O_{i,t|t-1}]^2.\end{aligned}$$

The hidden states for the fully-connected output layer are obtained from the hidden states of the last (L^{th}) LSTM layer using

$$Z^{(0)}_{i,t} = \boldsymbol{W}^{(\text{L})}_{i,t}\boldsymbol{H}^{(\text{L})}_t + B^{(\text{L})}_{i,t}. \tag{8}$$

Under the independence assumption, the mean and variance of each hidden state are given by

$$\begin{aligned}\mathbb{E}[Z^{(0)}_{i,t|t-1}] &= \mathbb{E}[\boldsymbol{W}^{(L)}_{i,t|t-1}\boldsymbol{H}^{(L)}_{t|t-1}] + \mathbb{E}[B^{(L)}_{i,t|t-1}],\\ \text{var}(Z^{(0)}_{i,t|t-1}) &= \text{var}(\boldsymbol{W}^{(L)}_{i,t|t-1}\boldsymbol{H}^{(L)}_{t|t-1}) + \text{var}(B^{(L)}_{i,t|t-1}).\end{aligned}$$

4.1.2. Backward step

The forward step presented in Section 4.1.1 can be regarded as sending information from the input layer to the output layer. In the backward step, we want to send the information in the opposite direction, from the output layer back to the input layer, in order to update the prior knowledge that has been obtained during the forward pass. Therefore, the objective of this step is to estimate the posterior PDFs for the hidden states and parameters of each layer in the network. For that, we rely on the conditional independence assumption between the hidden states of different layers in order to apply the layer-wise inference procedure, allowing to update the hidden states and parameters simultaneously within a same layer. This is essential for maintaining the computational tractability of the TAGI method. The different steps of the inference are depicted by the red arrows in Fig. 2b.

For the output layer, we compute the posterior $\boldsymbol{Z}^{(0)}_{t|t} \sim \mathcal{N}(\boldsymbol{\mu}^{\boldsymbol{Z}^{(0)}}_{t|t}, \boldsymbol{\Sigma}^{\boldsymbol{Z}^{(0)}}_{t|t})$ using the Gaussian conditional equations given in Eq. (5). This is analogous to updating the output $\boldsymbol{Z}^{(0)}$ of a TAGI-FNN as presented in Section 3.2. For the j^{th} LSTM layer, the posterior $\boldsymbol{H}^{(j)}_{t|t} \sim \mathcal{N}(\boldsymbol{\mu}^{\boldsymbol{H}^{(j)}}_{t|t}, \boldsymbol{\Sigma}^{\boldsymbol{H}^{(j)}}_{t|t})$ for its hidden states is estimated following

$$\begin{aligned} f(\boldsymbol{h}^{(j)}_{t|t}) &= \mathcal{N}(\boldsymbol{\mu}^{\boldsymbol{H}^{(j)}}_{t|t}, \boldsymbol{\Sigma}^{\boldsymbol{H}^{(j)}}_{t|t}),\\ \boldsymbol{\mu}^{\boldsymbol{H}^{(j)}}_{t|t} &= \boldsymbol{\mu}^{\boldsymbol{H}^{(j)}}_{t|t-1} + \mathbf{J}_{\boldsymbol{H}}(\boldsymbol{\mu}^{\boldsymbol{H}^{(j+1)}}_{t|t} - \boldsymbol{\mu}^{\boldsymbol{H}^{(j+1)}}_{t|t-1}),\\ \boldsymbol{\Sigma}^{\boldsymbol{H}^{(j)}}_{t|t} &= \boldsymbol{\Sigma}^{\boldsymbol{H}^{(j)}}_{t|t-1} + \mathbf{J}_{\boldsymbol{H}}(\boldsymbol{\Sigma}^{\boldsymbol{H}^{(j+1)}}_{t|t} - \boldsymbol{\Sigma}^{\boldsymbol{H}^{(j+1)}}_{t|t-1})\mathbf{J}^{\intercal}_{\boldsymbol{H}},\\ \mathbf{J}_{\boldsymbol{H}} &= \text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{H}^{(j+1)}_{t|t-1})(\boldsymbol{\Sigma}^{\boldsymbol{H}^{(j+1)}}_{t|t-1})^{-1}, \end{aligned} \tag{9}$$

where $\text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{H}^{(j+1)}_{t|t-1})$ is the covariance between the hidden states of the j^{th} layer and the $j+1^{th}$. Note that $\boldsymbol{H}^{(j)}_{t}$ is indirectly related to $\boldsymbol{H}^{(j+1)}_{t}$ through the LSTM gates $\{\boldsymbol{F}^{(j+1)}_t, \boldsymbol{I}^{(j+1)}_t, \tilde{\boldsymbol{C}}^{(j+1)}_t, \boldsymbol{O}^{(j+1)}_t\}$ as presented in Eqs. (1a)–(1d). To obtain $\boldsymbol{H}^{(j)}_{t|t}$, we directly calculate the covariance matrix $\text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{H}^{(j+1)}_{t|t-1})$, bypassing the LSTM gates as shown by the red arrow from $\boldsymbol{h}^{(j)}_t$ to $\boldsymbol{h}^{(j-1)}_t$ shown in Fig. 2a. The posterior $\boldsymbol{\theta}^{(j)}_{t|t} \sim \mathcal{N}(\boldsymbol{\mu}^{\boldsymbol{\theta}^{(j)}}_{t|t}, \boldsymbol{\Sigma}^{\boldsymbol{\theta}^{(j)}}_{t|t})$ for the parameters can be estimated using the same Eq. (9), while replacing the variable $\boldsymbol{H}^{(j)}$ by $\boldsymbol{\theta}^{(j)}$. Note that the last LSTM layer (L^{th}) is connected with the output layer so that when applying Eq. (9) for inferring its hidden states and parameters, we need to calculate the covariances $\text{cov}(\boldsymbol{H}^{(L)}_{t|t-1}, \boldsymbol{Z}^{(0)}_{t|t-1})$ and $\text{cov}(\boldsymbol{\theta}^{(L)}_{t|t-1}, \boldsymbol{Z}^{(0)}_{t|t-1})$. The detailed formulations for the covariances $\text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{H}^{(j+1)}_{t|t-1})$ and $\text{cov}(\boldsymbol{\theta}^{(j)}_{t|t-1}, \boldsymbol{H}^{(j+1)}_{t|t-1})$ are given in Appendix C, whereas the ones for $\text{cov}(\boldsymbol{H}^{(L)}_{t|t-1}, \boldsymbol{Z}^{(0)}_{t|t-1})$ and $\text{cov}(\boldsymbol{\theta}^{(L)}_{t|t-1}, \boldsymbol{Z}^{(0)}_{t|t-1})$ have been detailed by Goulet et al. (2021).

When performing inference for a LSTM layer, we need to update the cell states in addition to updating the hidden states and parameters. To this end, we estimate the posterior $\boldsymbol{C}^{(j)}_{t|t} \sim \mathcal{N}(\boldsymbol{\mu}^{\boldsymbol{C}^{(j)}}_{t|t}, \boldsymbol{\Sigma}^{\boldsymbol{C}^{(j)}}_{t|t})$ for the cell states based on the knowledge from the hidden states of a same LSTM layer using the same Rauch-Tung-Striebel (RTS) procedure (Rauch et al., 1965) used in Eq. (9) where the variable $\boldsymbol{H}^{(j)}$ is replaced by $\boldsymbol{C}^{(j)}$, and $\boldsymbol{H}^{(j+1)}$ is replaced by $\boldsymbol{H}^{(j)}$. This step is depicted by the red arrow from $\boldsymbol{h}^{(j)}_t$ to $\boldsymbol{c}^{(j)}_t$ shown in Fig. 2a. The diagonal cross-covariance matrix $\text{cov}(\boldsymbol{C}^{(j)}_{t|t-1}, \boldsymbol{H}^{(j)}_{t|t-1})$ between the cell and hidden states of the same LSTM layer that is required for estimating $\boldsymbol{C}^{(j)}_{t|t}$ is obtained following

$$\begin{aligned}\text{cov}(C^{(j)}_{i,t|t-1}, H^{(j)}_{i,t|t-1}) &= \text{var}(C^{(j)}_{i,t|t-1}) \cdot \nabla_C \tilde{\tanh}(\mathbb{E}[C^{(j)}_{i,t|t-1}])\\ &\quad \cdot \mathbb{E}[O^{(j)}_{i,t|t-1}],\end{aligned}$$

where $\nabla_C \tilde{\tanh}(\mathbb{E}[C^{(j)}_{i,t|t-1}])$ is the gradient of the function $\tilde{\tanh}(C)$ with respect to C evaluated at the mean $\mathbb{E}[C^{(j)}_{i,t|t-1}]$ for the j^{th} layer, cell i, at time t.

4.1.3. Smoothing for TAGI-LSTM

In SSMs, the transition model, $\boldsymbol{z}_t = \text{f}(\boldsymbol{z}_{t-1}) + \boldsymbol{w}_t$, describes the relationship between the hidden variables $\boldsymbol{z}$ at two consecutive time steps where $\text{f}(\cdot)$ is the transition function and $\boldsymbol{w}_t$ is a realization from the independent error process $\boldsymbol{W}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$. In other words, there is a flow of information through time from the first time step $t = 0$ to the last time step $t = \text{T}$. The Kalman smoother (Rauch et al., 1965) leverages this connection in order to send information backward, and update the knowledge for past hidden variables from future information. Analogously, LSTM also has through-time connections as shown in Eqs. (1a)–(1e) where the hidden and cell states at time step $t-1$ are connected to those at t such that

$$\begin{aligned}\boldsymbol{c}_t &= \text{g}(\boldsymbol{h}_{t-1}, \boldsymbol{c}_{t-1}, \boldsymbol{x}_t, \boldsymbol{\theta}),\\ \boldsymbol{h}_t &= \text{k}(\boldsymbol{h}_{t-1}, \boldsymbol{c}_{t-1}, \boldsymbol{x}_t, \boldsymbol{\theta}),\end{aligned}$$

where the functions $\text{g}(\cdot)$ and $\text{k}(\cdot)$ are defined by Eqs. (1a)–(1e). Therefore, we can leverage these connections in order to perform smoothing for TAGI-LSTM. The backward step in Section 4.1.2 uses the smoothing equations at a single time step in order to update the parameters and hidden states through the architecture; we now use the same approach to update backward through time following the inference path depicted by the red arrows in Fig. 3.

The posterior knowledge $\boldsymbol{H}^{(j)}_{t|t}$ for the hidden states of the j^{th} LSTM layer obtained from the backward step only contains the past and present information, i.e., $\boldsymbol{y}_{1:t}$. With the smoothing procedure, we want to estimate $\boldsymbol{H}^{(j)}_{t|\text{T}} \sim \mathcal{N}(\boldsymbol{\mu}^{\boldsymbol{H}^{(j)}}_{t|\text{T}}, \boldsymbol{\Sigma}^{\boldsymbol{H}^{(j)}}_{t|\text{T}})$ containing both past and future information, i.e., the whole sequence of observations $\boldsymbol{y}_{1:\text{T}}$. For each LSTM layer, we go backward from the last to the first time step, and apply the RTS smoother procedure recursively following

$$\begin{aligned}\boldsymbol{\mu}^{\boldsymbol{H}^{(j)}}_{t|\text{T}} &= \boldsymbol{\mu}^{\boldsymbol{H}^{(j)}}_{t|t} + \mathbf{J}_{\boldsymbol{H}}(\boldsymbol{\mu}^{\boldsymbol{H}^{(j)}}_{t+1|\text{T}} - \boldsymbol{\mu}^{\boldsymbol{H}^{(j)}}_{t+1|t}),\\ \boldsymbol{\Sigma}^{\boldsymbol{H}^{(j)}}_{t|\text{T}} &= \boldsymbol{\Sigma}^{\boldsymbol{H}^{(j)}}_{t|t} + \mathbf{J}_{\boldsymbol{H}}(\boldsymbol{\Sigma}^{\boldsymbol{H}^{(j)}}_{t+1|\text{T}} - \boldsymbol{\Sigma}^{\boldsymbol{H}^{(j)}}_{t+1|t})\mathbf{J}^{\intercal}_{\boldsymbol{H}},\\ \mathbf{J}_{\boldsymbol{H}} &= \text{cov}(\boldsymbol{H}^{(j)}_{t|t}, \boldsymbol{H}^{(j)}_{t+1|t})(\boldsymbol{\Sigma}^{\boldsymbol{H}^{(j)}}_{t+1|t})^{-1},\end{aligned} \tag{10}$$

where $\text{cov}(\boldsymbol{H}^{(j)}_{t|t}, \boldsymbol{H}^{(j)}_{t+1|t})$ is the covariance between the hidden states of the j^{th} LSTM layer at time t and $t+1$. We

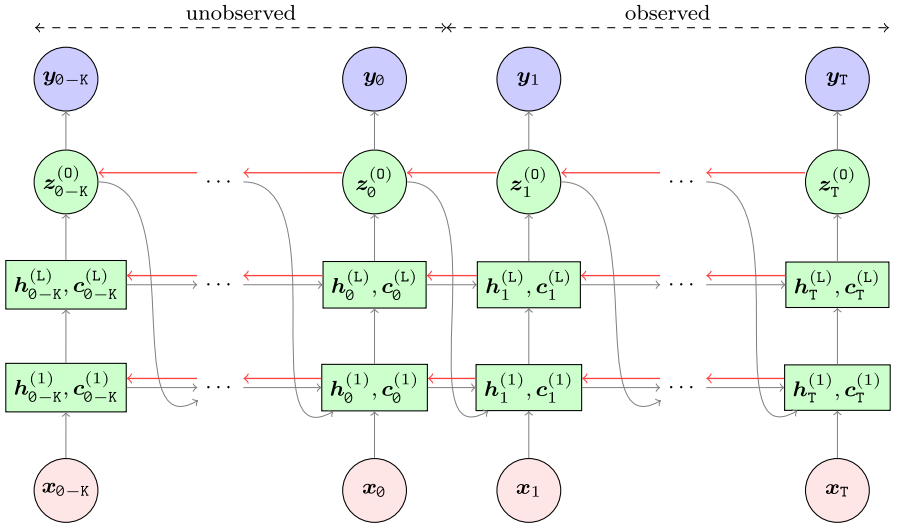

Fig. 3. Time-unfolded representation of a TAGI-LSTM network with explicit connections between times steps. Black arrows represent the network forward connections, red arrows represent the smoothing procedure. The observed data includes training data, whereas the unobserved one is data K time steps before the first training time step. This smoothing procedure allows to infer not only the smoothed estimates for hidden and cell states, and output during training time steps, but also the past ones before the training history.

can also use Eq. (10) to obtain the smoothed estimates $\boldsymbol{C}^{(j)}_{t|\mathrm{T}}$ for the cell states, and $\boldsymbol{Z}^{(0)}_{t|\mathrm{T}}$ for the hidden states of the output layer, by replacing the hidden states $\boldsymbol{H}$ by the relevant variable. Appendix D presents the derivations of the covariances $\mathrm{cov}(\boldsymbol{H}^{(j)}_{t|t}, \boldsymbol{H}^{(j)}_{t+1|t})$, $\mathrm{cov}(\boldsymbol{C}^{(j)}_{t|t}, \boldsymbol{C}^{(j)}_{t+1|t})$, and $\mathrm{cov}(\boldsymbol{Z}^{(0)}_{t|t}, \boldsymbol{Z}^{(0)}_{t+1|t})$ which are required to estimate $\boldsymbol{H}^{(j)}_{t|\mathrm{T}}$, $\boldsymbol{C}^{(j)}_{t|\mathrm{T}}$ and $\boldsymbol{Z}^{(0)}_{t|\mathrm{T}}$. Note that because the network parameters $\boldsymbol{\theta}$ are assumed to be constant through time, the smoother has no effect on them.

With the smoothing procedure, we start from the last time step T and either stop the procedure at $t = \emptyset$ to infer the initial hidden and cell states, $\boldsymbol{H}_{\emptyset|\mathrm{T}}$ and $\boldsymbol{C}_{\emptyset|\mathrm{T}}$, or we can even go back K time steps before the first training time step for obtaining the smoothed estimates for the hidden and cell states as presented in Fig. 3. We then can use them to estimate the unobserved past observations $\boldsymbol{y}_{\emptyset-\mathrm{K}:\emptyset}$. This means that we can use a single TAGI-LSTM model to estimate both future observations after the last training time and past observations before the first training time.

4.2. Coupling TAGI-LSTM and SSMs

In this section, we present the methodology to couple probabilistically the TAGI-LSTM presented in Section 4.1 with SSMs to create the TAGI-LSTM/SSM hybrid model. We consider a structured state-space model with additive errors where the hidden state vector $\boldsymbol{z}$ consists in two sets of hidden states $\boldsymbol{z}^{\mathrm{B}}$ and $z^{(0)}$, $\boldsymbol{z} = [\boldsymbol{z}^{\mathrm{B}}\ z^{(0)}]^{\intercal}$. The baseline hidden state vector $\boldsymbol{z}^{\mathrm{B}}$ models the time series' baseline patterns such as level and trend, and is associated with the linear dynamical system

$$\boldsymbol{z}^{\mathrm{B}}_{t} = \mathbf{A}\,\boldsymbol{z}^{\mathrm{B}}_{t-1} + \boldsymbol{w}_{t}, \tag{11}$$

where $\mathbf{A}$ is the transition matrix for the baseline hidden states, $\boldsymbol{w}_t$ is a realization from the independent error process $\boldsymbol{w}_t : \boldsymbol{W} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$. The hidden state $z^{(0)}$ is employed to model the repeated patterns such as seasonalities that are present in the data. Contrarily to the baseline hidden states which follow the transition model described in Eq. (11), the pattern hidden state is predicted using the output node of a TAGI-LSTM network

$$z^{(0)}_{t} = \text{TAGI-LSTM}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}, \boldsymbol{c}_{t-1}, \boldsymbol{\theta}), \tag{12}$$

where $\boldsymbol{x}_t$ is a vector of input covariates, $\boldsymbol{h}_{t-1}$ are the TAGI-LSTM hidden states at the previous time step, and $\boldsymbol{\theta}$ are the neural network parameters. Note that similarly to the baseline hidden states $\boldsymbol{Z}^{\mathrm{B}}$, the output $Z^{(0)}$ is also Gaussian.

At a time t, the joint prior knowledge for the baseline hidden states given all the past data $\boldsymbol{y}_{1:t-1}$ is described by a Gaussian random vector $\boldsymbol{Z}^{\mathrm{B}}_{t|t-1} \sim \mathcal{N}(\boldsymbol{\mu}^{\mathrm{B}}_{t|t-1}, \boldsymbol{\Sigma}^{\mathrm{B}}_{t|t-1})$ which is obtained by propagating the uncertainty associated with the posterior at $t-1$, i.e., $\boldsymbol{Z}^{\mathrm{B}}_{t-1|t-1}$ through the transition model described in Eq. (11). The marginal prior at t for the pattern hidden state is a Gaussian random variable $Z^{(0)}_{t|t-1} \sim \mathcal{N}(\mu^{Z^{(0)}}_{t|t-1}, \sigma^{Z^{(0)}}_{t|t-1})$ obtained using the TAGI-LSTM one-step-ahead prediction presented in Section 4.1. In order to obtain the joint prior knowledge at t for both the baseline and the pattern hidden states, we need their cross-covariance $\mathrm{cov}(\boldsymbol{Z}^{\mathrm{B}}_{t|t-1}, Z^{(0)}_{t|t-1})$ for which no exact closed-form analytical expression is available. In practice, these covariance terms are non-zero, yet they are typically small such that the correlation coefficients between hidden states $\rho < 0.01$. Therefore, we rely on the simplifying hypothesis that $\boldsymbol{Z}^{\mathrm{B}}_{t|t-1}$ and $Z^{(0)}_{t|t-1}$ are independent so that $\mathrm{cov}(\boldsymbol{Z}^{\mathrm{B}}_{t|t-1}, Z^{(0)}_{t|t-1}) = \mathbf{0}$. The validity of that hypothesis is investigated through experiments provided in Appendix E.

We can form the joint prior knowledge between the hidden states $\boldsymbol{Z}_{t|t-1} = [\boldsymbol{Z}^{\mathrm{B}}_{t|t-1}\ Z^{(0)}_{t|t-1}]^{\intercal}$ and the observation Y_t using the linear observation model

$$y_{t} = \mathbf{C}\,\boldsymbol{z}_{t} + v_{t}, \tag{13}$$

where v_t is a realization from the independent and identically distributed (i.i.d.) error $v_t : V \sim \mathcal{N}(0, \sigma_V^2)$. In the observation model presented in Eq. (13), the observation vector indicates which hidden state is observable such that $\mathbf{C} = [\boldsymbol{c}^{\mathrm{B}}\ c^{(0)}]$, where $c^{(0)} = 1$ and the composition of $\boldsymbol{c}^{\mathrm{B}}$ depends on the dynamic system chosen for the baseline. With that joint prior information, we can use the Gaussian conditional equations in order to obtain the joint posterior $\boldsymbol{Z}_{t|t}$ which can be broken down into the posteriors $\boldsymbol{Z}^{\mathrm{B}}_{t|t}$ and $Z^{(0)}_{t|t}$. The later is then used by the TAGI-LSTM method in order to infer both the Gaussian posterior for their model parameters $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}^{\boldsymbol{\theta}}_{t|t}, \mathbf{I}\boldsymbol{\sigma}^{\boldsymbol{\theta}}_{t|t})$ and their hidden states $\boldsymbol{H}_t \sim \mathcal{N}(\boldsymbol{\mu}^{\boldsymbol{H}}_{t|t}, \mathbf{I}\boldsymbol{\sigma}^{\boldsymbol{H}}_{t|t})$.

The covariance $\mathrm{cov}(\boldsymbol{Z}_{t|t}, \boldsymbol{Z}_{t+1|t})$ between the hidden states at two consecutive time steps is calculated by

$$\mathrm{cov}(\boldsymbol{Z}_{t|t}, \boldsymbol{Z}_{t+1|t}) = \begin{bmatrix} \boldsymbol{\Sigma}^{\mathrm{B}}_{t|t}\mathbf{A}^{\intercal} & \mathbf{0} \\ \mathbf{0} & \mathrm{cov}(Z^{(0)}_{t|t}, Z^{(0)}_{t+1|t}) \end{bmatrix},$$

where $\mathrm{cov}(Z^{(0)}_{t|t}, Z^{(0)}_{t+1|t})$ is calculated using the smoothing procedure presented in Section 4.1.3. Given this covariance, the Kalman smoother can be performed analytically for the TAGI-LSTM/SSM model. As it is the case for the TAGI model presented in Section 3.2, we train the model and learn the network parameters $\boldsymbol{\theta}$ over multiple epochs. For each epoch, we perform a forward pass over time and a backward one using the Kalman smoother.

4.2.1. Exponential smoothing component

Using the local linear trend component (Durbin & Koopman, 2001) for the baseline hidden states $\boldsymbol{z}^{\mathrm{B}}$, the TAGI-LSTM/SSM model presented in Section 4.2 can only model data which contains a linear trend. In this section, we formulate a parameter-free exponential smoothing component which can be used with the TAGI-LSTM/SSM to automatically detrend data containing complex long-term patterns.

Recall the state-space form of a simple exponential smoothing model (Hyndman & Athanasopoulos, 2018), but now we consider the smoothing parameter z^{α} and the observation error z^{V} as hidden states. The model is defined by the following equations

$$\text{Observation equation:} \quad y_t = z^{\mathrm{E}}_t + z^{\mathrm{V}}_t, \tag{14}$$

$$\begin{aligned} \text{Transition equations:} \quad z^{\mathrm{E}}_t &= z^{\mathrm{E}}_{t-1} + \bar{z}^{\alpha}_{t-1} z^{\mathrm{V}}_{t-1}, \\ z^{\alpha}_t &= z^{\alpha}_{t-1}, \\ z^{\mathrm{V}}_t &= v_t, \end{aligned} \tag{15}$$

where z^{E}_t is the exponential smoothing hidden state, v_t is a realization from the i.i.d. error $v_t : V \sim \mathcal{N}(0, \sigma_V^2)$, and y_t is the observation. Because the smoothing parameter takes a value between zero and one, we apply the locally linearized sigmoid function that is already used for TAGI in Section 3.2 to the hidden state z^{α}_t. The transformed variable $\bar{z}^{\alpha}_t = \tilde{\sigma}(z^{\alpha}_t)$ is assumed to follow a Gaussian PDF where its moments are obtained by applying the equations presented in Appendix B. The product $\bar{z}^{\alpha}_{t-1} z^{\mathrm{V}}_{t-1}$ uses the previous step's error z^{V}_{t-1} to adjust the predicted exponential smoothing hidden state z^{E}_t. We rely on the covariance $\mathrm{cov}(\bar{Z}^{\alpha}_{t-1|t-1} Z^{\mathrm{V}}_{t-1|t-1}, Z^{\alpha}_{t|t-1})$ in order to update the hidden state z^{α}_t. Note that if instead we would use the product $\bar{z}^{\alpha}_{t-1} z^{\mathrm{V}}_t$ to adjust z^{E}_t, the covariance $\mathrm{cov}(\bar{Z}^{\alpha}_{t-1|t-1} Z^{\mathrm{V}}_{t|t-1}, Z^{\alpha}_{t|t-1})$ would be zero, and thus we could not update z^{α}_t.

The transition model as given in Eq. (15) is nonlinear, however, Deka et al. (2022) showed that this nonlinear model can be reformulated as a linear dynamic model by creating a new hidden state $z^{\mathrm{N}}_t = \bar{z}^{\alpha}_{t-1} z^{\mathrm{V}}_{t-1}$ which is assumed to follow a Gaussian PDF where its moments are calculated exactly using the GMA equations given in Appendix A. The baseline hidden state vector $\boldsymbol{z}^{\mathrm{B}} = [z^{\mathrm{E}}\ z^{\alpha}\ z^{\mathrm{V}}]^{\intercal}$ at time $t-1$ is thus assumed to be Gaussian. The augmented baseline hidden state vector $\tilde{\boldsymbol{z}}^{\mathrm{B}} = [\boldsymbol{z}^{\mathrm{B}}\ z^{\mathrm{N}}]^{\intercal}$ is defined by

$$\tilde{\boldsymbol{Z}}^{\mathrm{B}}_{t-1|t-1} \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}^{\mathrm{B}}_{t-1|t-1}, \tilde{\boldsymbol{\Sigma}}^{\mathrm{B}}_{t-1|t-1}),$$

where the components of $\tilde{\boldsymbol{\mu}}^{\mathrm{B}}_{t-1|t-1}$ and $\tilde{\boldsymbol{\Sigma}}^{\mathrm{B}}_{t-1|t-1}$ are presented in Appendix F.

The nonlinear Eq. (15) now can be written in a linear form as

$$\tilde{\boldsymbol{z}}^{\mathrm{B}}_t = \mathbf{A}\tilde{\boldsymbol{z}}^{\mathrm{B}}_{t-1} + \boldsymbol{w}_t, \qquad \boldsymbol{w}_t : \boldsymbol{W} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}), \tag{16}$$

where $\mathbf{A}$ is the transition matrix and $\mathbf{Q}$ is the covariance matrix. The prior for the augmented baseline hidden state vector is given by

$$\tilde{\boldsymbol{Z}}^{\mathrm{B}}_{t|t-1} \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}^{\mathrm{B}}_{t|t-1}, \tilde{\boldsymbol{\Sigma}}^{\mathrm{B}}_{t|t-1}), \tag{17}$$

where $\tilde{\boldsymbol{\mu}}^{\mathrm{B}}_{t|t-1} = \mathbf{A}\tilde{\boldsymbol{\mu}}^{\mathrm{B}}_{t-1|t-1}$ and $\tilde{\boldsymbol{\Sigma}}^{\mathrm{B}}_{t|t-1} = \mathbf{A}\tilde{\boldsymbol{\Sigma}}^{\mathrm{B}}_{t-1|t-1}\mathbf{A}^{\intercal} + \mathbf{Q}$. The observation model given in Eq. (14) is written in a matrix form as

$$y_t = \mathbf{C}\tilde{\boldsymbol{z}}^{\mathrm{B}}_t, \tag{18}$$

where $\mathbf{C}$ is the observation matrix. The model matrices $\mathbf{A}$, $\mathbf{Q}$ and $\mathbf{C}$ are given in Appendix F. Using Eqs. (16)–(18), we can now apply the Kalman filter to have a parameter-free component that can capture complex long-term patterns in data. When we couple this new exponential smoothing component with TAGI-LSTM, it enables to automatically detrend the data as shown by the experiments in the next section.

5. Experiments

In this section, we first conduct a qualitative experiment using synthetic time series to verify the TAGI-LSTM model's capability to accurately retrieve the ground truth. Next, a quantitative analysis is conducted to compare the performance of the same LSTM architecture trained with either TAGI or gradient-based approaches. The Electricity and Traffic datasets (Yu et al., 2016) datasets are chosen for this comparison because they are stationary, i.e., each time series displays a constant baseline so that the LSTM models can analyze them directly without requiring data preprocessing to remove trends. Finally, after verifying TAGI-LSTM's capabilities on simulated and stationary datasets, we compare the performance of the TAGI-LSTM/SSM hybrid model with other benchmark methods on the non-stationary Tourism (monthly and quarterly) and M4 (hourly) datasets, displaying either linear or complex nonlinear trends. These datasets

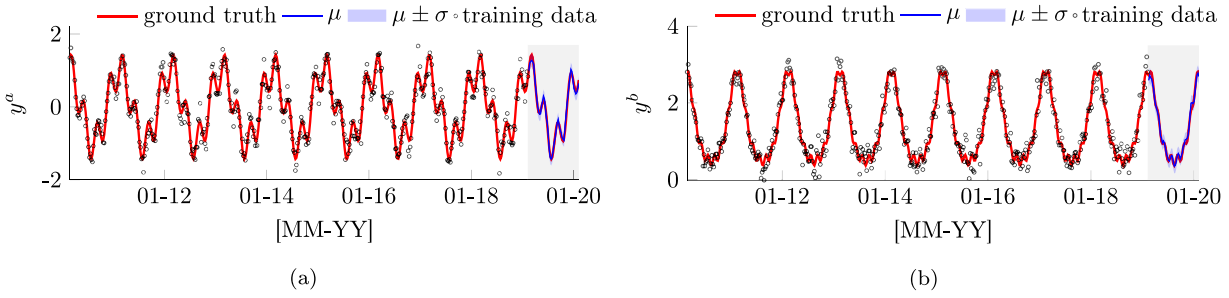

Fig. 4. Test predictions from TAGI-LSTM models for synthetic data generated using (a) Eq. (19a), and (b) Eq. (19b). The grey shaded area presents the forecast period, red line presents the ground truth, blue line presents test predictions along with $\pm\sigma$ confidence intervals (shade), black dots present training data.

are intended to demonstrate that the hybrid TAGI-LSTM/SSM model is well-suited to handle trended data without the need for preprocessing. Note that one cannot use plain LSTMs on these benchmarks without preprocessing the data beforehand to remove trends.

5.1. Verification of TAGI-LSTM on synthetic data

For the verification experiment, we generate ten years of weekly data (520 data points) from the following functions

$$y_t^a = \sin(\frac{2\pi t}{365.22}) + 0.5\sin(\frac{2\pi t}{365.22/4}) + v_t, \tag{19a}$$

$$y_t^b = \exp\left[\sin(\frac{2\pi t}{365.22})\right] + \left[0.5\sin(\frac{2\pi t}{365.22/4})\right]^2 + v_t, \tag{19b}$$

as depicted in Fig. 4, where $v_t : V \sim \mathcal{N}(0, 0.2^2)$ and t is the timestamp. We train TAGI-LSTM models using the first 9 years of data, and make mutli-step-ahead predictions for the last year. Fig. 4 presents the ground truth which does not contain the error term v_t, the training data, as well as the test predictions and their $\pm\sigma$ uncertainties from our models. The test set's predictions show that the TAGI-LSTM can accurately retrieve the ground truth for these synthetic data.

5.2. Comparison of TAGI-LSTM with gradient-based LSTM networks on stationary data

In this section, we perform a quantitative experiment to compare the predictive performance of the TAGI-LSTM in comparison to two gradient-based LSTMs, i.e., deterministic and variational ones (Gal & Ghahramani, 2016b), on the same network architecture. We perform the comparison on the Electricity dataset (Yu et al., 2016) containing 370 hourly time series of electricity consumption, and the Traffic one (Yu et al., 2016) including 963 hourly time series of occupancy rate for different car lanes in the San Francisco bay area. These datasets are specifically selected because they are stationary which allows the LSTM models to analyze them directly after standardization without requiring ad-hoc procedures to remove trends. As these datasets experience hourly and daily seasonal patterns, we include hour-of-the-day and day-of-the-week as time features in the covariate vector $\boldsymbol{x}_t$. We apply the moving window approach proposed by Bandara et al. (2020) so that the input covariate vector includes a lookback window of posterior values for $\boldsymbol{z}^{(0)}_{t-\mathtt{W}:t-1}$ where $\mathtt{W}$ is the window's length. Detailed explanation for this approach can be found in Hewamalage et al. (2021). The task consists in providing predictions for seven consecutive days using the rolling window operation described in Yu et al. (2016). Using this operation, the model makes multi-step-ahead predictions for a 24-hour-window. Then, the observations for this window are made available so that they can be used to predict the next one. There are seven windows required to cover the test period. In order to make multi-step-ahead predictions, we apply the single-output forecasting procedure and recursively make one-step-ahead predictions.

The data is divided into training and test sets using three different splits (Oreshkin et al., 2020) as detailed in Appendix G. A window of data points at the end of the training data is reserved for validation, and it is not used for training. We train our models with 50 epochs, then identify the optimal epoch which maximizes the log-likelihood for the validation set, and obtain the network parameters at this optimal epoch. We build a separate LSTM model for each time series where all of the models use the same network architecture as used by Salinas et al. (2020). The standard deviation for the error σ_V is a hyper-parameter in our TAGI-LSTM model that is common to all time series. For variational LSTM, we perform 50 dropout passes and calculate the predictive mean and variance. The architecture and hyper-parameters used in this experiment are given in Appendix H.

The ρ-quantile loss, $\rho \in (0, 1)$, measures the accuracy of the predictive distribution at a specific quantile (Rangapuram et al., 2018), and is defined by

$$\mathrm{QL}_\rho(\boldsymbol{y}, \hat{\boldsymbol{y}}^\rho) = 2\frac{\sum_{i,t}[\rho \cdot \max(y_{i,t} - \hat{y}^\rho_{i,t}, 0) + (1-\rho)\cdot \max(\hat{y}^\rho_{i,t} - y_{i,t}, 0)]}{\sum_{i,t}|y_{i,t}|},$$

where $y_{i,t}$ is the observation at time t of the ith time series and $\hat{y}^\rho_{i,t}$ is the corresponding predicted ρ-quantile. Note

Table 1
Comparison between TAGI-LSTM, deterministic and variational LSTMs on Electricity and Traffic datasets. $p50$ and $p90$-loss metrics for test sets are calculated using the rolling window operation. Results are obtained by averaging the forecasts over five independent runs. Deterministic LSTM provides only point forecasts so that we report only the $p50$-loss. Three train/test splits are used following Oreshkin et al. (2020). Bold fonts indicate the best results.

Split	Electricity						Traffic					
	2014-03-31		2014-09-01		last 7 days		2008-01-14		2008-06-15		last 7 days	
	$p50$	$p90$	$p50$	$p90$	$p50$	$p90$	$p50$	$p90$	$p50$	$p90$	$p50$	$p90$
TAGI-LSTM	**0.080**	0.058	0.066	0.053	**0.152**	**0.095**	0.337	0.276	0.169	0.158	0.102	**0.130**
LSTM	0.086	–	0.077	–	0.159	–	**0.319**	–	**0.158**	–	**0.098**	–
Variational LSTM	**0.080**	**0.056**	**0.064**	**0.052**	0.160	0.100	0.323	**0.261**	0.162	**0.154**	0.123	0.133

that the $p50$-loss is equal to the Normalized deviation (ND) metric reported in Oreshkin et al. (2020), Salinas et al. (2020), Wang et al. (2019), Yu et al. (2016), and the ND metric was original used to compare these datasets by Yu et al. (2016). The results reported in Table 1 are obtained by averaging the predictions from five independent runs with different initial seeds. Both TAGI-LSTM and variational LSTM provide predictions along with the associated uncertainties. Therefore, we report the $p50$ and $p90$-losses for these methods. The deterministic LSTM provides point estimates so that we report only the $p50$-loss. The results from Table 1 show that our method matches the performance of the two LSTMs trained with gradient-based optimization. The deterministic LSTM provides the best $p50$-loss for the Traffic dataset on all splits which means that it provides the most accurate point forecasts. However, unlike TAGI-LSTM and variational LSTM, this method cannot provide predictive uncertainties. The variational LSTM provides slightly better $p90$-losses compared to TAGI-LSTM in four out of six scenarios. However, the result of this method varies with respect to the number of dropout passes during test time. By contrast, TAGI-LSTM can analytically estimate both the predictions and the associated uncertainties without repeatedly performing dropout.

Robustness verifications for the TAGI-LSTM method are presented in Appendix I, where we compare the predictive performances of four different TAGI-LSTM models, each with a different number of hidden layers, ranging from one to four. The root mean square error (RMSE) and mean absolute scaled error (MASE) metrics for this experiment are presented in Appendix J.

5.3. Comparison of TAGI-LSTM/SSM with state-of-the-art methods on non-stationary data

After verifying TAGI-LSTM's capabilities on simulated and stationary datasets, this section validates the capacity of the hybrid TAGI-LSTM/SSM model to match the performance of state-of-the-art methods on the non-stationary Tourism (monthly and quarterly) (Athanasopoulos et al., 2011) and M4 (hourly) (Makridakis et al., 2018) datasets, containing either linear or complex nonlinear trends. We excluded the plain LSTM models from this experiment because they would have required further ad-hoc preprocessing in order to remove the trends from data before analyzing. The monthly Tourism dataset contains 366 time series, the quarterly Tourism dataset contains 427 time series, and the hourly M4 dataset has 414 time series.

The baseline hidden state vector contains a level (L), a trend (T), and an exponential smoothing component, $\boldsymbol{z}^{\mathrm{B}} = [z^{\mathrm{L}}\ z^{\mathrm{T}}\ z^{\mathrm{E}}\ z^{\alpha}\ z^{\mathrm{V}}]^{\intercal}$, and the pattern hidden state $z^{(0)}$ models the seasonality. The model matrices are given in Appendix K. For the time features included in the covariate vector $\boldsymbol{x}_t$, we include hour-of-the-day and day-of-the-week for the hourly dataset, month-of-the-year for the monthly dataset, and quarter-of-the-year for the quarterly dataset. We build a separate model for each time series where all models share the same network architecture which is detailed in Appendix H. The standard deviation for the noise σ_V is learnt using the AGVI method (Deka, 2022; Deka & Goulet, 2023). We train our models on multiple epochs where the initial hidden states $\boldsymbol{\mu}_{\emptyset}^{(i+1)}$ and $\boldsymbol{\Sigma}_{\emptyset}^{(i+1)}$ at the $i+1^{th}$ epoch are the smoothed estimates $\boldsymbol{\mu}_{\emptyset|\mathrm{T}}^{(i)}$ and $\boldsymbol{\Sigma}_{\emptyset|\mathrm{T}}^{(i)}$ at the i^{th} epoch with T being the last training time.

We compare our method with the statistical methods ARIMA (Box et al., 1994) and ETS (Hyndman et al., 2008); the hybrid methods DeepState (Rangapuram et al., 2018) and ES-RNN (Smyl, 2020); and the pure neural networks DeepAR (Salinas et al., 2020) and N-Beats (Oreshkin et al., 2020). The two hybrid methods are similar to our model since all of them combine SSMs or exponential smoothing and LSTMs, and ES-RNN is the state-of-the-art hybrid method for the M4 dataset (Makridakis et al., 2020). DeepAR is a pure LSTM-based method, whereas N-Beats is the state-of-the-art neural network method for these datasets. Among them, ARIMA, ETS and our method adopt a local setup such that a separate model is built for each time series, and there is no information shared between models; DeepAR and N-Beats adopt a global setup, fitting a single model for multiple time series; whereas DeepState and ES-RNN use a mixed setup, using a global LSTM to learn across multiple time series and a local SSM or exponential smoothing model for each time series. Table 2 reports the $p50$ and $p90$ metrics for our TAGI-LSTM/SSM along with benchmark models. For a fair comparison, we report the values from the original papers (Oreshkin et al., 2020; Rangapuram et al., 2018; Skepetari, 2020). The results show that our method exceeds the performance of the ARIMA, ETS, DeepAR, and DeepState in both $p50$ and $p90$ metrics for all datasets except for the $p50$ loss of the monthly Tourism one. This shows the data-efficiency of our method such that we can adopt a local setup, that is building a separate

Table 2
$p50$ and $p90$-loss performances on the Tourism (monthly and quarterly) and M4 (hourly) test sets. The results for N-Beats are obtained from Oreshkin et al. (2020), the one for ES-RNN is calculated by us from the submission downloaded from the M4 GitHub repository (Skepetari, 2020), and the results for other baseline methods are obtained from Rangapuram et al. (2018). The results for TAGI-LSTM/SSM are obtained by averaging the forecasts over three independent runs with different initial seeds. N-Beats provides point forecasts so that only the $p50$-loss is reported. Bold fonts indicate the best results.

	Tourism (Monthly)		Tourism (Quaterly)		M4 (Hourly)	
	$p50$	$p90$	$p50$	$p90$	$p50$	$p90$
ARIMA	0.100	0.058	0.124	0.062	0.052	0.035
ETS	**0.093**	0.054	0.105	0.055	0.054	0.027
DeepState	0.138	0.067	0.098	0.047	0.044	0.027
ES-RNN	–	–	–	–	0.039	–
DeepAR	0.107	0.059	0.110	0.062	0.090	0.030
N-Beats	0.097	–	0.077	–	**0.023**	–
TAGI-LSTM/SSM	0.102	**0.053**	**0.073**	**0.041**	0.042	**0.021**

model for each time series, and provide better results than other models using local, global and mixed setups. N-Beats provides point forecasts so that we report only the $p50$-loss. The ES-RNN and N-Beats methods provide superior point forecasts compared to our method. Note that these methods use more advanced neural network architecture such as dilated LSTMs with attention mechanism and residual connections. In addition, unlike our method, ES-RNN and N-Beats can provide interpretable components such as level, trend and seasonality, but are unable to provide their associated uncertainties.

In order to extend the comparisons, we re-obtained the results for the DeepState, DeepAR and N-Beats methods using the *gluonTS* library (Alex et al., 2020), for ARIMA and ETS models using the *forecast* library (Hyndman & Khandakar, 2008), and for the ES-RNN method using the *ESRNN* one (Gutierrez et al., 2021). Appendix L presents the comparisons on RMSE and MASE metrics, as well as the multiple comparisons with the best (MCB) test (Koning et al., 2005). These additional results further confirm that the TAGI-LSTM/SSM method proposed has a performance that is competitive with other state-of-the-art methods.

Fig. 5 shows an example of predictions on the test set, the hidden states, as well as their associated uncertainties for the time series #30 of the Tourism (monthly) dataset. These results confirm that our model can separate the long-term linear pattern which is captured by the level hidden state z^{L}, the long-term nonlinear pattern which is captured by the exponential smoothing hidden state z^{E}, and the seasonality modelled by the TAGI-LSTM. Additional examples for other time series are presented in Appendix M.

6. Conclusion

The new mathematical formulations proposed in this paper enable using the Tractable Approximate Gaussian Inference (TAGI) method with the LSTM neural network architecture. The approach allows estimating analytically the posterior mean vectors and diagonal covariance matrices for the hidden states and model parameters using approximate Bayesian inference. The experiments performed showed that for the same network architecture, our TAGI-LSTM models provide on-par performance compared to the deterministic and variational LSTMs trained with backpropagation and gradient descent. With the method presented in this paper, one can now couple LSTM neural networks with the existing components borrowed from SSMs. We have demonstrated, for example, how we can couple TAGI-LSTM with the existing SSM's level and trend components, as well as the parameter-free exponential smoothing one in order to automatically detrend time series while providing interpretable results. The experimental results on the quarterly and monthly Tourism, and hourly M4 datasets have shown that our hybrid model is competitive with state-of-the-art methods.

CRediT authorship contribution statement

Van-Dai Vuong: Conceptualization, Methodology, Software, Investigation, Writing - Original Draft, Writing - Review & Editing. **Luong-Ha Nguyen:** Conceptualization, Software. **James-A. Goulet:** Conceptualization, Writing - Review & Editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Van-Dai Vuong, Luong-Ha Nguyen, and James-A. Goulet report financial support was provided by Natural Sciences and Engineering Research Council of Canada (NSERC) and Hydro-Québec.

Acknowledgments

This project is funded by Natural Sciences and Engineering Research Council of Canada (NSERC) and Hydro Québec under grant #ALLRP 578556-22.

Appendix A. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.ijforecast.2024.04.002.

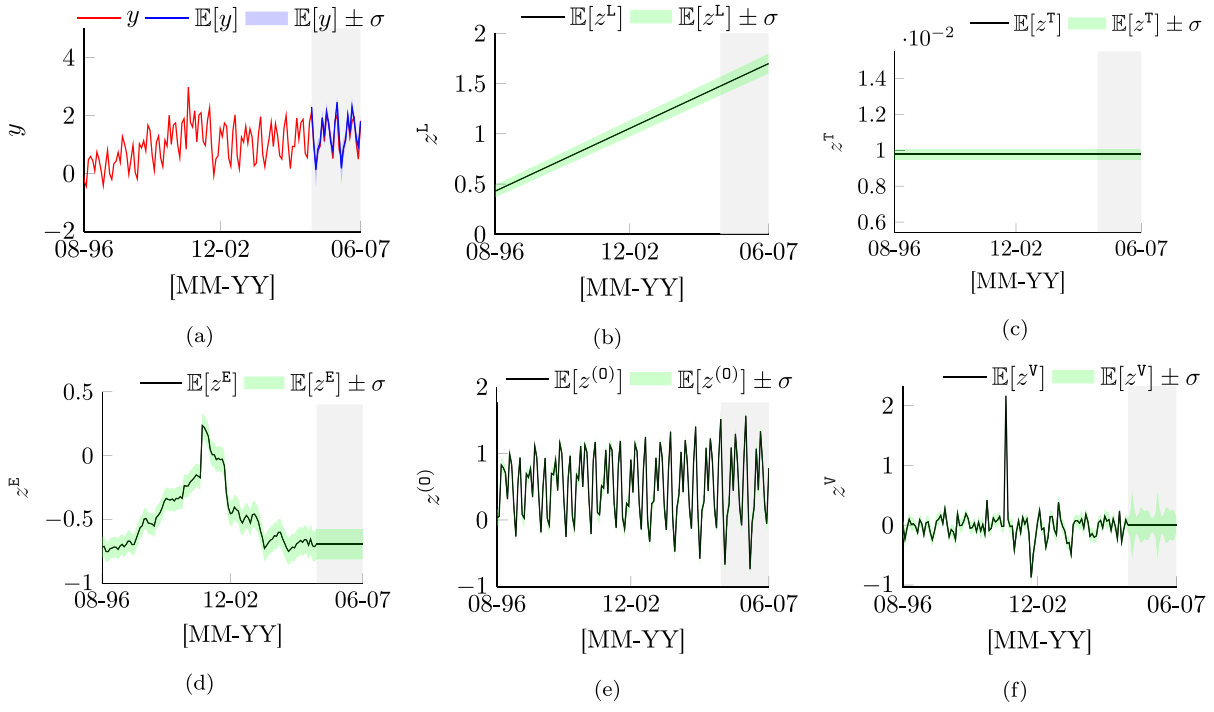

Fig. 5. An example of interpretable results provided by TAGI-LSTM/SSM using the proposed exponential smoothing component for the time series #30 of the Tourism (monthly) dataset. Values are presented in standardized space. Red line presents observations, blue line presents test predictions, black lines present components along with 68% confidence intervals (shade). The grey shaded areas present the forecast period. (a) Predictions on test set (b) level, (c) trend, (d) exponential smoothing, (e) seasonality modelled by TAGI-LSTM, and (f) error hidden state. The results show that the model can separate the linear long-term pattern which is captured by the level hidden state and the nonlinear long-term one which is captured by the exponential smoothing component.

References

Alexandrov, A., Benidis, K., Bohlke-Schneider, M., Flunkert, V., Gasthaus, J., Januschowski, T., Maddix, D. C., Rangapuram, S., Salinas, D., Schulz, J., Stella, L., Türkmen, A. C., & Wang, Y. (2020). GluonTS: Probabilistic and neural time series modeling in Python. *Journal of Machine Learning Research*, *21*(116), 1–6.

Athanasopoulos, G., Hyndman, R. J., Song, H., & Wu, D. C. (2011). The tourism forecasting competition. *International Journal of Forecasting*, *27*(3), 822–844.

Bandara, K., Bergmeir, C., & Smyl, S. (2020). Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach. *Expert Systems with Applications*, *140*, 112896.

Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015). Weight uncertainty in neural network. In *Proceedings of the 32nd international conference on machine learning, vol. 37* (pp. 1613–1622).

Box, G. E. P., Reinsel, G. C., & Jenkins, G. M. (1994). *Time series analysis: forecasting and control*. Prentice-Hall.

Chien, J.-T., & Ku, Y.-C. (2016). Bayesian recurrent neural network for language modeling. *IEEE Transactions on Neural Networks and Learning Systems*, *27*(2), 361–374.

Deka, B. (2022). *Analytical Bayesian parameter inference for probabilistic models with engineering applications* (Ph.D. thesis), Canada: Polytechnique Montréal.

Deka, B., & Goulet, J.-A. (2023). Approximate Gaussian variance inference for state-space models. *International Journal of Adaptive Control and Signal Processing*, *37*(11), 2934–2962.

Deka, B., Nguyen, L. H., Amiri, S., & Goulet, J.-A. (2022). The Gaussian multiplicative approximation for state-space models. *Structural Control and Health Monitoring*, *29*(3), e2904.

Durbin, J., & Koopman, S. (2001). *Time series analysis by state space methods*. Oxford University Press.

Efron, B. (2010). *Large-scale inference: Empirical Bayes methods for estimation, testing, and prediction*. Cambridge University Press.

Fortunato, M., Blundell, C., & Vinyals, O. (2019). Bayesian recurrent neural networks. arXiv arXiv:1704.02798.

Fraccaro, M., Sønderby, S. K., Paquet, U., & Winther, O. (2016). Sequential neural models with stochastic layers. In *Proceedings of the 30th international conference on neural information processing systems* (pp. 2207–2215).

Gal, Y., & Ghahramani, Z. (2016a). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd international conference on international conference on machine learning, vol. 48* (pp. 1050–1059).

Gal, Y., & Ghahramani, Z. (2016b). A theoretically grounded application of dropout in recurrent neural networks. In *Proceedings of the 30th international conference on neural information processing systems, vol. 29* (pp. 1027–1035).

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Goulet, J.-A., Nguyen, L. H., & Amiri, S. (2021). Tractable approximate Gaussian inference for Bayesian neural networks. *Journal of Machine Learning Research*, *22*, 1–23.

Gutierrez, K., Challu, C., Garza, F., & Mergenthaler, M. (2021). Pytorch implementation of the ES-RNN algorithm. GitHub repository. https://github.com/kdgutier/esrnn_torch. (Accessed 17 March 2023).

Hernandez-Lobato, J. M., & Adams, R. (2015). Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *Proceedings of the 32nd international conference on machine learning, vol. 37* (pp. 1861–1869).

Hewamalage, H., Bergmeir, C., & Bandara, K. (2021). Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, *37*(1), 388–427.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*, 1735–1780.
Hyndman, R., & Athanasopoulos, G. (2018). *Forecasting: Principles and practice* (2nd ed.). OTexts.
Hyndman, R. J., & Khandakar, Y. (2008). Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software*, *26*(3), 1–22.
Hyndman, R., Koehler, A., Ord, J., & Snyder, R. (2008). *Forecasting with exponential smoothing: The state space approach*. Springer.
Januschowski, T., Gasthaus, J., Wang, Y., Salinas, D., Flunkert, V., Bohlke-Schneider, M., & Callot, L. (2020). Criteria for classifying forecasting methods. *International Journal of Forecasting*, *36*(1), 167–177.
Koning, A. J., Franses, P. H., Hibon, M., & Stekler, H. (2005). The M3 competition: Statistical tests of the results. *International Journal of Forecasting*, *21*(3), 397–409.
Krishnan, R. G., Shalit, U., & Sontag, D. (2017). Structured inference networks for nonlinear state space models. In *Proceedings of the thirty-first AAAI conference on artificial intelligence* (pp. 2101–2109).
MacKay, D. J. C. (1992). A practical Bayesian framework for backpropagation networks. *Neural Computation*, *4*(3), 448–472.
Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). The M4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, *34*(4), 802–808.
Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2020). The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, *36*(1), 54–74.
Moon, T., Choi, H., Lee, H., & Song, I. (2015). RNNDROP: A novel dropout for RNNS in ASR. In *2015 IEEE workshop on automatic speech recognition and understanding* (pp. 65–70).
Nguyen, L.-H., & Goulet, J.-A. (2021). Analytically tractable inference in deep neural networks. In *The 4th workshop on tractable probabilistic modeling*.
Nguyen, L.-H., & Goulet, J.-A. (2022). Analytically tractable hidden-states inference in Bayesian neural networks. *Journal of Machine Learning Research*, *23*(50), 1–33.
Oreshkin, B. N., Carpov, D., Chapados, N., & Bengio, Y. (2020). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International conference on learning representations 2020*.
Pham, V., Bluche, T., Kermorvant, C., & Louradour, J. (2014). Dropout improves recurrent neural networks for handwriting recognition. In *14th international conference on frontiers in handwriting recognition* (pp. 285–290).
Rangapuram, S. S., Seeger, M. W., Gasthaus, J., Stella, L., Wang, Y., & Januschowski, T. (2018). Deep state space models for time series forecasting. In *Advances in neural information processing systems*: *vol. 31*.
Rauch, H. E., Tung, F., & Striebel, C. T. (1965). Maximum likelihood estimates of linear dynamic systems. *American Institute of Aeronautics and Astronautics*, *3*(8), 1445–1450.
Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*, 533–536.
Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, *36*(3), 1181–1191.
Skepetari, E. (2020). Data, benchmarks, and methods submitted to the M4 forecasting competition. https://github.com/Mcompetitions/M4-methods. (Accessed 27 February 2023).
Smyl, S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, *36*(1), 75–85.
Sun, S., Chen, C., & Carin, L. (2017). Learning Structured Weight Uncertainty in Bayesian Neural Networks. In *Proceedings of the 20th international conference on artificial intelligence and statistics, vol. 54* (pp. 1283–1292).
Wang, Y., Smola, A., Maddix, D., Gasthaus, J., Foster, D., & Januschowski, T. (2019). Deep factors for forecasting. In *Proceedings of the 36th international conference on machine learning, vol. 97* (pp. 6607–6617).
Wu, A., Nowozin, S., Meeds, E., Turner, R. E., Hernandez-Lobato, J. M., & Gaunt, A. L. (2019). Deterministic variational inference for robust Bayesian neural networks. In *International conference on learning representations 2019*.
Yu, H.-F., Rao, N., & Dhillon, I. S. (2016). Temporal regularized matrix factorization for high-dimensional time series prediction. In *Advances in neural information processing systems*: *vol. 29*.
Zaheer, M., Ahmed, A., & Smola, A. J. (2017). Latent LSTM allocation: Joint clustering and non-linear dynamic modeling of sequence data. In *Proceedings of the 34th international conference on machine learning, vol. 70* (pp. 3967–3976).
Zaremba, W., Sutskever, I., & Vinyals, O. (2014). Recurrent neural network regularization. arXiv arXiv:1409.2329.
Zheng, X., Zaheer, M., Ahmed, A., Wang, Y., Xing, E. P., & Smola, A. (2017). State space LSTM models with particle MCMC inference. arXiv arXiv:1711.11179.