POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Analytically Tractable Bayesian Recurrent Neural Networks with Structural Health Monitoring Applications

VAN-DAI VUONG
Département de génie civil, géologique et des mines

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
génie civil

March 2024

© Van-Dai Vuong, 2024.

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

Analytically Tractable Bayesian Recurrent Neural Networks with Structural Health Monitoring Applications

présentée par **Van-Dai VUONG**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Jonathan JALBERT, président
James-A. GOULET, membre et directeur de recherche
Audrey OLIVIER, membre externe
Lijun SUN, membre externe

DEDICATION

To my grandparents and parents,
To my little family,
To my brother,

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my advisor Prof. James-A. Goulet, for his encouragement, patience and guidance throughout the journey of this thesis. I could not have asked for a better mentor.

I would also like to thank and acknowledge the Natural Sciences and Engineering Research Council of Canada (NSERC), Hydro-Québec (HQ), and Hydro-Québec's Research Institute (IREQ) for the financial support of this research. I would like to further thank Hydro-Québec for providing the datasets used in this thesis. Many thanks to Benjamin Miquel, Vincent Roy, Patrice Côté, Simon-Nicolas Roth, and Stéphance Lafrance from Hydro-Québec for their suggestions and support during the research project.

I would like to thank my friends and colleagues at Polytechnique Montréal, Luong Ha Nguyen, Bhargob Deka, Ngoc Huy Vu, Zachary Hamida, Shervin Khazaeli, Ali Fakhri, Zhanwen Xin, Lucas Alric, Enzo Louvard, Miquel Florensa, and Yassir El Moumen, with you I have had the pleasure of working with over the years. A special thanks to Luong Ha Nguyen and Bhargob Deka who personally helped me when I first joined the research group and during the research project.

Last but not the least, I would like to thank my family for their unconditional support over the years. To my grandparents and parents, who always encourage and support every decision in my life. To my wife Trang and my son Eric, thank you for being with me. To my brother Dong, you have taught me more than what I have taught you.

RÉSUMÉ

Les infrastructures civiles, notamment les barrages, les ponts, les bâtiments et les oléoducs, jouent un rôle essentiel dans la croissance économique. Même lorsqu'elles sont conçues correctement, ces structures se détériorent au fil du temps, ce qui entraîne une dégradation de leur état. Elles doivent donc être surveillées et entretenues afin de garantir leur sécurité et leur fonctionnalité. Les données brutes des capteurs concernant les réponses structurelles telles que les déplacements, les ouvertures de fissures, les déformations ou les accélérations sont difficiles à interpréter car elles sont généralement affectées par des erreurs de mesure ainsi que par les effets de l'environnement. Les modèles linéaires dynamiques bayésiens (BDLM) et le filtre de Kalman à commutation (SKF), qui sont des types spécifiques de modèles espace-état (SSM), ont été utilisés comme méthodes d'interprétation des données dans le cadre de la surveillance de l'état des structures (SHM) afin d'extraire des informations utiles sur leur état à partir des données brutes des capteurs et de prédire les réponses futures. Toutefois, la principale limite de ces méthodes est qu'elles nécessitent une ingénierie des caractéristiques étendue afin de construire des modèles, de sorte que leur application n'est pas adaptée à un déploiement à grande échelle sur un grand nombre de séries temporelles.

L'objectif de cette thèse est de développer des réseaux neuronaux récurrents bayésiens (RNN) analytiquement traçables pour automatiser la construction de modèles de séries temporelles. Cette méthode peut être utilisée dans les cadres SSM et SKF existants afin d'éliminer le besoin d'ingénierie des caractéristiques, améliorant ainsi leur mise à l'échelle pour les applications SHM. Les principales contributions de cette thèse consistent en l'élaboration de (1) de réseaux neuronaux Bayésiens à longue mémoire à court terme (LSTM) et à unité récurrente à relais (GRU), (2) d'un modèle probabiliste hybride qui associe le RNN Bayésien analytiquement traitable et le SSM pour fournir des résultats interprétables avec les incertitudes de prédiction, sans nécessiter d'ingénierie des caractéristiques, et (3) d'une méthodologie pour utiliser le modèle hybride dans le cadre SKF existant pour la détection d'anomalies semi-supervisée en ligne dans des conditions non stationnaires dans le domaine SHM. Les méthodologies proposées sont vérifiées avec des données synthétiques et validées avec des ensembles de données de référence de séries temporelles ainsi qu'avec des données du SHM. Les résultats montrent que la méthode de détection d'anomalies proposée est plus performante que d'autres modèles de référence, car elle permet de détecter les anomalies de manière fiable, tout en contrôlant étroitement le taux de fausses alarmes.

ABSTRACT

Civil infrastructures, including dams, bridges, buildings, and pipelines, play a vital role in the economic growth of every country. Even when properly designed, these structures deteriorate over time leading to a decline in their condition so that they have to be monitored and maintained in order to ensure their safety and serviceability. Raw sensor data about structural responses such as displacements, crack openings, strains, or accelerations are difficult to interpret because they are typically affected by measurement noise as well as environmental effects. The Bayesian Dynamic Linear Models (BDLM) and Switching Kalman Filter (SKF), which are specific types of state-space models (SSM), have been used as data interpretation methods in *structural health monitoring* (SHM) to extract useful insights about the structural condition from the raw sensor data and to predict future responses. However, the main limitation of these methods is that they require extensive feature engineering in order to build models so that their application is not suited for large scale deployment on a high number of time series.

The objective of this thesis is to develop analytically tractable Bayesian recurrent neural networks (RNN) for automating the construction of time series models. This method can be used in the existing SSM and SKF frameworks in order to eliminate the need for feature engineering, therefore enhancing their scalability for SHM applications. The main contributions consist in the development of: (1) analytically tractable Bayesian Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) neural networks, (2) a hybrid probabilistic model that couples the analytically tractable Bayesian RNN and SSM for providing interpretable results along with the prediction uncertainties, while not requiring feature engineering, and (3) a methodology to use the hybrid model in the existing SKF framework for online semi-supervised anomaly detection under nonstationary conditions in SHM. The proposed methodologies are verified with synthetic data and validated with time series benchmark datasets as well as real SHM ones. The results show that the proposed anomaly detection method outperforms other baseline models as it can reliably detect anomalies, while having a tight control over the false alarm rate.

TABLE OF CONTENTS

DEDICATION iii

ACKNOWLEDGEMENTS iv

RÉSUMÉ v

ABSTRACT vi

TABLE OF CONTENTS vii

LIST OF TABLES x

LIST OF FIGURES xiii

LIST OF SYMBOLS AND ACRONYMS xxii

LIST OF APPENDICES xxvii

CHAPTER 1 Introduction 1
- 1.1 Motivation 1
- 1.2 Research Objectives 3
- 1.3 Thesis Outline 3
- 1.4 Co-Authored Papers 4

CHAPTER 2 Literature Review 5
- 2.1 Introduction 5
- 2.2 State-Space Models 5
  - 2.2.1 Kalman Filter 7
  - 2.2.2 Bayesian Dynamic Linear Models 8
  - 2.2.3 Switching Kalman Filter 12
- 2.3 Neural Networks 13
  - 2.3.1 Neural Network Architecture 15
  - 2.3.2 Training Neural Networks 25
  - 2.3.3 Tractable Approximate Gaussian Inference 28
- 2.4 Hybrid Models 32
- 2.5 Anomaly Detection 33

2.6 Conclusion 36

CHAPTER 3 Analytically Tractable Bayesian Recurrent Neural Networks 37
3.1 Introduction 37
3.2 TAGI Long Short-Term Memory 37
3.2.1 Forward step 38
3.2.2 Backward step 41
3.2.3 Smoothing for TAGI-LSTM 43
3.3 TAGI Gated Recurrent Units 44
3.4 Bypassing Inference in TAGI 46
3.5 Experiments 49
3.5.1 Verification on Synthetic Data 50
3.5.2 Experiment – Stationary Time Series Benchmark Dataset 50
3.5.3 Experiment – Structural Health Monitoring Dataset 55
3.6 Conclusion 57

CHAPTER 4 Coupling Analytically Tractable Bayesian Recurrent Neural Networks and State-space Models 58
4.1 Introduction 58
4.2 Methodology 58
4.2.1 Exponential Smoothing Component 60
4.3 Experiments 63
4.3.1 Validation of the Zero Cross-covariance Assumption 64
4.3.2 Experiments – Nonstationary Time Series Benchmark Dataset 66
4.3.3 Experiment – Structural Health Monitoring Dataset 71
4.4 Conclusion 80

CHAPTER 5 Switching Kalman Filter and Anomaly Detection 81
5.1 Introduction 81
5.2 Methodology 81
5.2.1 Parameter Estimation 83
5.3 Experiments 86
5.3.1 Verification with Synthetic Data 87
5.3.2 Experiment – Structural Health Monitoring Dataset 93
5.4 Conclusion 105

CHAPTER 6 Conclusion 106

6.1 Thesis Conclusion 106
6.2 Limitations 107
6.2.1 Local versus Global Recurrent Neural Networks 107
6.2.2 Coupling Analytically Tractable Bayesian Recurrent Neural Networks and State-space Models 108
6.2.3 Anomaly Detection 108

REFERENCES 109

APPENDICES 120
A.1 TAGI-LSTM – Covariances Required for Backward Step 120
A.1.1 Covariances Between the Hidden States of Two Consecutive LSTM Layers 120
A.1.2 Covariances Between the Parameters and Hidden States of Two Consecutive LSTM Layers 121
A.2 TAGI-LSTM – Covariances Required for Smoothing Procedure 122
A.2.1 Covariances Between the Hidden States of the Same LSTM Layer . . 122
A.2.2 Covariances Between the Cell States of the Same LSTM Layer 123
A.2.3 Covariances Between the Output Hidden States 123
B.1 TAGI-GRU – Covariances Required for Estimating Hidden States 124
B.2 TAGI-GRU – Covariances Required for Backward Step 125
B.2.1 Covariances Between the Hidden States of Two Consecutive GRU Layers 125
B.2.2 Covariances Between the Parameters and Hidden States of Two Consecutive GRU Layers 126

LIST OF TABLES

Table 2.1 Comparison of the complexity among Transformer-based models and the LSTM architecture. Reproduced from [1]. 25

Table 3.1 Train and test splits for Electricity and Traffic datasets. Subscript [1] indicates the splits used in DeepFactor [2], [2] splits used in DeepAR [3] and DeepState [4], [3] splits used in MatFact [5]. 51

Table 3.2 Architecture and hyperparameters for models used in our experiments. The Electricity and Traffic datasets have three train/test splits as presented in Table 3.1; the value for each split is separated by a forward slash. d, w, m, q, and y are the abbreviations for day, week, month, quarter, and year respectively. 52

Table 3.3 Comparison between TAGI-LSTM, TAGI-GRU, and their corresponding deterministic and variational LSTM and GRU models trained with backpropagation on Electricity and Traffic datasets. $p50/p90$-loss for test sets are calculated using the rolling window operation. Results are obtained by averaging the forecasts over five independent runs. Deterministic LSTM and GRU models provide only point forecasts so that we report only the $p50$-loss. Three train/test splits are presented in Table 3.1. Bold fonts indicate the best results. 53

Table 3.4 RMSE and MASE metrics for the Electricity and Traffic datasets. Results are obtained by averaging the forecasts over five independent runs. Bold fonts indicate the best results. 53

Table 3.5 Comparison between four different TAGI-LSTM models having different architecture on Electricity and Traffic datasets. $p50/p90$-loss for test sets are calculated using the rolling window operation. Results are obtained by averaging the forecasts over five independent runs. Bold fonts indicate the best results. 53

Table 4.1 Testing the zero cross-covariances hypothesis used in TAGI-LSTM/SSM by comparing the test log-likelihood performance between the models considering zero cross-covariances (no-cov) and the ones considering non-zero cross-covariances (MC-cov) on six time series from the Tourism and M4 datasets. The average absolute correlations are calculated over all training time steps. The $\mu \pm \sigma$ represents the mean and standard deviation over five runs. 64

Table 4.2 Architecture and hyperparameters for TAGI-LSTM models used in our experiments. d, m, and q are the abbreviations for day, month, and quarter, respectively. The AGVI method is presented in [6–8]. 66

Table 4.3 $p50$ and $p90$-loss performances on the Tourism (monthly and quarterly) and M4 (hourly) test sets. The results for N-Beats are obtained from [9], the one for ES-RNN is calculated by us from the submission downloaded from the M4 GitHub repository [10], and the results for other baseline methods are obtained from [4]. 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. 69

Table 4.4 $p50$, $p90$, RMSE, and MASE performances on the Tourism (monthly and quarterly) and M4 (hourly) test sets using our re-run results. Re-run results are obtained from three independent runs except for the ARIMA and ETS. Bold fonts indicate the best results. * indicates that results are obtained by using the predictions provided by the authors downloaded from the M4 repository [10]. 71

Table 4.5 Descriptions of datasets used in case studies using SHM dataset. . . . 72

Table 4.6 Architecture and hyperparameters for the models used in our experiments. 72

Table 4.7 Comparison of test performance between TAGI-LSTM/SSM with other models on the water infiltration rate dataset. Results for TAGI-LSTM/SSM, LSTM and GRU models are obtained by averaging the forecasts over ten independent runs. Only the MSE metric is reported for deterministic LSTM and GRU models. Bold fonts indicate the best results. . . 74

Table 4.8 Comparison of test performance between TAGI-LSTM/SSM and other models on the traffic load dataset. Results for TAGI-LSTM/SSM, LSTM and GRU models are obtained by averaging the forecasts over ten independent runs. Only the MSE metric is reported for deterministic LSTM and GRU models. Bold fonts indicate best results. 76

Table 4.9 Comparison of test performance between TAGI-LSTM/SSM and other models on the CB2 displacement dataset. Results for TAGI-LSTM/SSM are obtained by averaging the forecasts over ten independent runs. Bold fonts indicate best results. 79

Table 5.1 Characteristics comparison of anomaly detection methods for SHM applications. N/a for the MP and kNN means that no training is required for these methods. 91
Table C.1 Architecture and hyperparameters for the TAGI-SKF models used in our experiments. 128
Table E.1 Hyperparameters for the Prophet model. 131
Table E.2 Hyperparameters for the Matrix profile, kNN, Autoencoder, and VAE models. N/a means that the hyperparameter is not applied for the method. 131

LIST OF FIGURES

Figure 1.1 Crack opening data from a dam in Canada and unobserved hidden states extracted from the raw data using the BDLM model. 2

Figure 2.1 Three synthetic time series having different baselines, but sharing the same periodic pattern. The red line presents the data, while the blue line presents the baseline. 8

Figure 2.2 Dataset from the international Benchmark Workshop organized by the International Commission of Large Dams (ICOLD) [11], (a) inverted pendulum, (b) raw reservoir's water level, (c) components extracted from the raw reservoir's water level, and (d) moving-averages for air temperature. 10

Figure 2.3 Hidden state estimation from BDLM model for the CB2 pendulum dataset, (a) level component, (b) contribution of the water level's average long-term trend $x^{\mathtt{WL},2}$ of the water level, (c) contribution of the water level's mean-centered component $x^{\mathtt{WL},1}$, (d) autoregressive component, (e) predictions on the validation and test sets, (f) relative importance of each component. The figures are reused from [6]. 11

Figure 2.4 Time series with two regimes where the first one has a constant trend and the second one has a linear positive trend. The blue line presents the baseline. 12

Figure 2.5 Switching Kalman Filter filtering and collapse steps for a case involving two regimes. The number of models increases from 2 to 4 after the filtering step, while the collapse step reduces and keeps 2 models at each time. 14

Figure 2.6 Representation of a simple neural network where the circle represents a node and the arrow represents a connection between two nodes. . . 14

Figure 2.7 Equivalent representations of a single-hidden-layer fully connected feed-forward neural network where (a) only the hidden states $\boldsymbol{z}$, and (b) both hidden states $\boldsymbol{z}$ and activations units $\boldsymbol{a}$ are shown. 15

Figure 2.8 The (a) full representation of a multi-hidden-layer FNN where each node represents a unit, and the arrow between any two nodes represents the forward connection, and (b) the equivalent compact representation of the same network where each node represents a vector, and $\boldsymbol{\theta} = \{\boldsymbol{w}, \boldsymbol{b}\}$. The figures are reproduced from [12]. 17

Figure 2.9 The time-unrolled representation of a single-hidden-layer RNN. The arrows depict the casual direction or casual relationship between variables as described in Equation 2.12. 18
Figure 2.10 The teacher forcing setup for RNN where the output-to-hidden recurrent connection is defined between the observations $\boldsymbol{y}_{t-1}$ and the hidden states $\boldsymbol{z}_t$. 19
Figure 2.11 Time-unrolled representation of a bidirectional RNN where two separate RNN models are employed at the same time, one moves forward and the another moves backward in time. 19
Figure 2.12 (a) Time-unrolled, and (b) compact representation of a stacked RNN where the double line arrow represents recurrent connections. Note that for the first hidden layer, the double arrow implies the both hidden-to-hidden and output-to-hidden recurrent connections, where for other hidden layers it implies only the hidden-to-hidden connection. 20
Figure 2.13 (a) The representation of a LSTM cell, and (b) the compact form of a stacked LSTM where single-line arrows present forward connections, and double-line arrows represent the recurrent connections. 21
Figure 2.14 (a) GRU cell, and (b) compact representation of a stacked GRU where the double line arrow represents recurrent connections. 23
Figure 2.15 Comparison for the test log-likelihood and test RMSE for the Boston dataset among various Bayesian neural network methods. The figures are reused from [13]. 28
Figure 2.16 Graphical representation of a fully connected feedforward neural network. Black arrows represent the network forward connections; red arrows represent the inference directions for the hidden states and for the parameters. 29
Figure 2.17 Comparison between TAGI with diagonal covariance structures and Hamiltonian Monte-Carlo (HMC) for a 1D regression problem $y = x^3 + v$. Re-used from [12]. 31
Figure 3.1 (a) Graphical representation of a LSTM cell. Red arrows represent the inference procedure to update the $j - 1^{th}$ LSTM layer from the subsequent j^{th} 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. 38

Figure 3.2 Infer TAGI-LSTM's hidden states and parameters of a layer j^{th} from hidden states of layer $j+1^{th}$ (a) without bypassing LSTM's gates , and (b) bypassing LSTM's gates. 42

Figure 3.3 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. 44

Figure 3.4 (a) Graphical representation of a GRU cell. Red arrows represent the inference procedure to update the $j-1^{th}$ GRU layer from the subsequent j^{th} GRU layer. (b) Graphical representation of a stacked TAGI-GRU network. Black arrows represent the network's forward connections, red arrows represent the layer-wise inference paths, and double arrows represent the recurrent connections. 45

Figure 3.5 Layer-wise inference procedure to update the hidden states for a feed-forward neural network (a) without, and (b) with activation units shown explicitly. Black arrows represent the network forward connections, red arrows represent the layer-wise inference paths. 47

Figure 3.6 Infer the hidden states of a layer j^{th} from the hidden states of layer $j+1^{th}$ (a) with, and (b) without bypassing activation units $\boldsymbol{a}^{(j)}$. . . . 47

Figure 3.7 Infer the hidden states of a layer j^{th} from the hidden states of layer $j+2^{th}$ (a) with, and (b) without bypassing the layer $j+1^{th}$. 49

Figure 3.8 Test predictions from TAGI-LSTM models for synthetic data generated using (a) Equation 3.14a, and (b) Equation 3.14b. 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. The $\pm\sigma$ regions contain both the epistemic (parameter's) uncertainties obtained from the prior predictive distribution of $z^{(0)}$ as well as the aleatory uncertainties associated with the error term's variance σ_V^2. 50

Figure 3.9 Comparison of test predictions between the TAGI-LSTM and deterministic LSTM models for a time series in (a) Electricity, and (b) Traffic datasets. The $\pm\sigma$ regions contain both the epistemic (parameter's) uncertainties obtained from the prior predictive distribution of $z^{(0)}$ as well as the aleatory uncertainties associated with the error term's variance σ_V^2. Only a part of the training set is presented. 54

Figure 3.10 Example of learning the time-varying aleatory uncertainty (σ_V) estimated using the AGVI method [8] for a time series in the traffic dataset. The grey shaded area presents the test set, and only a part of the training set is presented. 54

Figure 3.11 Test predictions from the TAGI-LSTM trained using the original trended data. The $\pm\sigma$ regions contain both the epistemic (parameter's) uncertainties obtained from the prior predictive distribution of $z^{(0)}$ as well as the aleatory uncertainties associated with the error term's variance σ_V^2. 55

Figure 3.12 (a) Trend and (b) zero-mean recurrent pattern components obtained by detrending the original crack opening time series. The grey shaded area presents the forecast period. 56

Figure 3.13 (a) The test predictions for the zero-mean recurrent pattern, and (b) the final test predictions obtained by summing up the zero-mean predictions and the trend component. The $\pm\sigma$ regions contain both the epistemic (parameter's) uncertainties obtained from the prior predictive distribution of $z^{(0)}$ as well as the aleatory uncertainties associated with the error term's variance σ_V^2. 56

Figure 4.1 The nonlinear transition function for the pattern hidden states is modelled by a TAGI-LSTM network. Red arrows represent the layer-wise inference paths for inferring the posterior knowledge for the TAGI-LSTM's hidden states and parameters from the posterior PDF of the pattern hidden state $Z^{(0)}$. Black arrows represent the network forward connections, and double arrows represent the recurrent connections. . 59

Figure 4.2 An example of interpretable results for several time series. Values are presented in standardized space. Red lines present observations, blue lines present test predictions, black lines present components along with $\pm\sigma$ confidence intervals (shade). The grey shaded areas present the forecast period. (a) predictions on test set, (b) level, (c) trend, and (d) seasonality modelled by TAGI-LSTM for time series #32 of the Tourism (monthly) dataset. (e) predictions on test set, (f) level, (g) trend, and (h) seasonality modelled by TAGI-LSTM for time series #33 of the Tourism (quarterly) dataset. (i) predictions on test set, (j) level, (k) trend, and (l) seasonality modelled by TAGI-LSTM for time series #375 of the M4 (hourly) dataset. In (a), (e) and (i), the $\pm\sigma$ regions contain both the epistemic (parameter's) uncertainties obtained from the prior predictive distributions of the hidden states as well as the aleatory uncertainties associated with the error term's variance σ_V^2. Zoom-in figures for (a), (e) and (i) are provided in Appendix D. . . . 65

Figure 4.3 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 $\pm\sigma$ confidence intervals (shade). The grey shaded areas present the forecast period. The $\pm\sigma$ regions in (a) contain both the epistemic (parameter's) uncertainties obtained from the prior predictive distributions of the hidden states as well as the aleatory uncertainties associated with the error term's variance σ_V^2. 67

Figure 4.4 An example of interpretable results for the time series #36 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 $\pm\sigma$ confidence intervals (shade). The grey shaded areas present the forecast period. The $\pm\sigma$ regions in (a) contain both the epistemic (parameter's) uncertainties obtained from the prior predictive distributions of the hidden states as well as the aleatory uncertainties associated with the error term's variance σ_V^2. 68

Figure 4.5 Average ranks and 95% confidence intervals for all methods over all time series in each dataset based on the multiple comparisons with the best (MCB) test using the $p50$ metric. Dashed lines present 95% confidence intervals for the best method. 70
Figure 4.6 Average ranks and 95% confidence intervals for all methods over all time series in each dataset based on the multiple comparisons with the best (MCB) test using $p90$ metric. Dashed lines present 95% confidence intervals for the best method. 70
Figure 4.7 Water infiltration rate from a concrete dam in Canada. The grey shaded area presents the forecast period. 73
Figure 4.8 Results from our model for the water infiltration rate dataset including the test predictions, the level and local trend hidden states, and the recurrent pattern. The grey shaded area presents the forecast period. The $\pm\sigma$ regions in (a) contain both the epistemic (parameter's) uncertainties obtained from the prior predictive distributions of the hidden states as well as the aleatory uncertainties associated with the error term's variance σ_V^2. 74
Figure 4.9 Traffic load data on the Tamar bridge in the UK. The grey shaded area presents the forecast period. 75
Figure 4.10 Results from our model for the traffic load dataset including the test predictions, the level hidden state, and the recurrent pattern. The grey shaded area presents the forecast period. The $\pm\sigma$ regions in (a) contain both the epistemic (parameter's) uncertainties obtained from the prior predictive distributions of the hidden states as well as the aleatory uncertainties associated with the error term's variance σ_V^2. . 76
Figure 4.11 Data from a dam in southern France 78
Figure 4.12 Relationships between the dam's displacement and environmental variables. 78
Figure 4.13 Results from our model for the displacement dataset including the test predictions, the level hidden state, and the recurrent pattern. The grey shaded area presents the forecast period. The $\pm\sigma$ regions in (a) contain both the epistemic (parameter's) uncertainties obtained from the prior predictive distributions of the hidden states as well as the aleatory uncertainties associated with the error term's variance σ_V^2. . 79

Figure 5.1 Introduction of synthetic anomaly. (a) anomaly-free training time series and (b) time series with synthetic anomaly. The blue line represents the baseline of the data. 85

Figure 5.2 (a) Anomaly-free time series generated from $y_t = \sin(2\pi t/365)+0.5\sin(\pi t/365)+ v_t$, and (b) an example of time series containing a synthetic anomaly during the test period. The shaded grey area presents the period where anomaly is randomly introduced. 87

Figure 5.3 Experiment with synthetic data. Log-likelihood and multi-step-ahead predictions for the validation set. 88

Figure 5.4 Experiment with synthetic data. The TAGI-LSTM/SSM's model matrices for the transitions from $s_{t-1} = i$ to $s_t = j$. 89

Figure 5.5 (a) Three time series with anomalies of different slopes introduced at the same time step, (b) 50 time series with anomalies of the same slope $\beta_b = 0.25$ [mm/year] but introduced at different time steps, and (c) the probability of anomaly $\pi_{t|t}$ and detection time for one synthetic time series. The dashed line presents the anomaly's start time, the shaded grey area presents the period where anomalies are randomly introduced with different start times. 90

Figure 5.6 Detection probability p^{β_b} for detecting anomalies with different slope β_b in the simulated time series for models with different sets $\boldsymbol{\mathcal{P}}_k$. . . . 90

Figure 5.7 Comparison of (a) detection probability p^{β_b}, (b) false alarm rate per ten years, and (c) average detection time among all methods for different anomalies of slope β_b. The shaded area presents the associated $\pm\sigma$ confidence region. 92

Figure 5.8 Case study #1. (a) Crack opening data from a dam in Canada, and (b) preliminary analysis to identify visually possible anomalies where the grey shaded area presents the regime switch quantitatively. The dash line with a corresponding colour presents the extension for the regime identified. 94

Figure 5.9 Case study #1: crack opening. Log-likelihood and multi-step-ahead predictions for the validation set. The $\pm\sigma$ regions contain both the epistemic (parameter's) uncertainties obtained from the prior predictive distributions of the hidden states as well as the aleatory uncertainties associated with the error term's variance σ_V^2. 95

Figure 5.10 Case study #1: crack opening. Synthetic time series (a) with anomalies of different slopes introduced at the same time step, (b) with 50 anomalies of the same slope introduced at different time steps, and (c) the probability of anomaly $\pi_{t|t}$ and detection time for one synthetic time series. The shaded grey area presents the period where synthetic anomaly is randomly added. 95

Figure 5.11 Case study #1: crack opening. Probability of detection for different anomaly's slopes. 96

Figure 5.12 Case study #1. Anomaly detection results obtained from our method and other models. Each line of scatter points presents when the alarms are triggered for each method, and the grey shaded area presents the visually identified anomalies from the preliminary analysis. The blue dashed lines present where the anomalies are detected by our method. The $\pm\sigma$ confidence intervals in the top row of (a) and (b) contain both the epistemic (parameter's) uncertainties obtained from the posterior predictive distributions of the hidden states as well as the aleatory uncertainties associated with the error term's variance σ_V^2. 97

Figure 5.13 Case study #2: displacement data from a dam in Canada. 99

Figure 5.14 Case study #2: dam's displacement. Preliminary analysis to identify visually possible anomalies where the grey shaded area presents the regime switch quantitatively. The dash line with a corresponding colour presents the extension for the regime identified. 99

Figure 5.15 Case study #2: dam's displacement. Log-likelihood and multi-step-ahead predictions for the validation set. 99

Figure 5.16 Case study #2: dam's displacement. Probability of detection for different anomaly's slopes. 100

Figure 5.17 Case study #2. Anomaly detection results obtained from our method and other models. Each line of scatter points presents the alarms triggered for each method, and the grey shaded area presents the visually identified anomalies from the preliminary analysis. The blue dashed lines present where the anomalies are detected by our method. The $\pm\sigma$ confidence intervals in the top row of (a) and (b) contain both the epistemic (parameter's) uncertainties obtained from the posterior predictive distributions of the hidden states as well as the aleatory uncertainties associated with the error term's variance σ_V^2. 101

Figure 5.18 Case study #3: crack opening and air temperature from a dam in Canada. 102
Figure 5.19 Case study #3: crack opening. Preliminary analysis to identify visually possible anomalies where the grey shaded area presents the regime switch quantitatively. The dash line with a corresponding colour presents the extension for the regime identified. 102
Figure 5.20 Case study #3: crack opening. Log-likelihood and multi-step-ahead predictions for the validation set. 102
Figure 5.21 Case study #3: crack opening. Probability of detection for different anomaly's slopes. 103
Figure 5.22 Case study #3. Anomaly detection results obtained from our method and other models. The blue dashed lines present where the anomalies are detected by our method. The $\pm\sigma$ confidence intervals in the top row of (a) and (b) contain both the epistemic (parameter's) uncertainties obtained from the posterior predictive distributions of the hidden states as well as the aleatory uncertainties associated with the error term's variance σ_V^2. 104
Figure D.1 Zoom-in figures for for Figures 4.2 (a), (e) and (i), respectively 129

LIST OF SYMBOLS AND ACRONYMS

The notation in this thesis uses italic lowercase letters (x) for deterministic variables, bold lowercase letters ($\boldsymbol{x}$) for vectors of deterministic variables, italic uppercase letters (X) for random variables; bold italic uppercase letters ($\boldsymbol{X}$) for vectors or matrices of random variables, bold upright uppercase letters ($\mathbf{X}$) for deterministic matrices, and typewriter fonts ($\texttt{X}$) for either representing the number of elements in a set or vector, or for specific variable names.

Symbols

$a_i^{(j)}$	i^{th} activation unit in j^{th} hidden layer
$\mathbf{A}$	Transition matrix
$\texttt{A}$	Number of hidden nodes
$\texttt{AR}$	Autoregressive component
$\boldsymbol{b}^{(j)}$	Bias vector for j^{th} hidden layer
$\texttt{B}$	Batch size
$\boldsymbol{c}$	Cell states for Long short-term memory (LSTM) neural network
$\tilde{\boldsymbol{c}}$	Candidate gate for LSTM
$\text{cov}(\cdot)$	Covariance operator
$\texttt{D}$	Total number of observations
$\mathcal{D}$	Set of observations
$\text{diag}(\cdot)$	Create diagonal matrix or get diagonal elements of matrix
e	Epoch
$\texttt{E}$	Total number of epochs
$\mathbb{E}[\cdot]$	Expectation operator
$\exp(\cdot)$	Exponential operator
$f(\cdot)$	Probability density function
$f(\boldsymbol{x}_t \vert \boldsymbol{x}_{t-1})$	Transition model
$f(\boldsymbol{y}_t \vert \boldsymbol{x}_t)$	Likelihood
$f(\boldsymbol{x}_t \vert \boldsymbol{y}_{1:t-1})$	Prior PDF of hidden states at time t
$f(\boldsymbol{x}_t \vert \boldsymbol{y}_{1:t})$	Posterior PDF of hidden states at time t
$f(\boldsymbol{y}_t \vert \boldsymbol{y}_{1:t-1})$	Evidence
$\boldsymbol{f}$	Forget gate for LSTM
$\mathbf{F}$	Observation matrix

$g(\cdot)$	Transition function
$\mathbf{G}$	Innovation covariance matrix
$h(\cdot)$	Observation function
$\boldsymbol{h}$	Hidden states for LSTM or GRU
$\tilde{\boldsymbol{h}}$	Candidate gate for GRU
$\mathbf{H}$	Hessian matrix
$\boldsymbol{i}$	Input gate for LSTM
$\mathbf{I}$	Identity matrix
$\mathrm{J}(\boldsymbol{\theta})$	Objective function
$\mathbf{J}$	Kalman smoother gain matrix
$\mathbf{K}$	Kalman gain matrix
`KR`	Kernel regression
$l(\cdot)$	Loss function
$\boldsymbol{\mathcal{L}}$	Likelihood
`L`	Level
`LL`	Local level
`LT`	Local trend
`LA`	Local acceleration
$\mathbf{M}$	Joint probability matrix
$\mathcal{N}(\cdot)$	Gaussian distribution
$\boldsymbol{o}$	Output gate for LSTM
$\mathcal{O}$	Computational complexity
p^{β}	Probability to detect synthetic anomalies with the slope β
$\boldsymbol{\mathcal{P}}$	A set of parameters
$\Pr(\cdot)$	Probability
$\mathbf{Q}$	Process error covariance matrix
$\boldsymbol{r}$	Innovation vector
$\mathbf{R}$	Observation error covariance matrix
s	Regime state variable
`S`	Number of regimes
t	Timestamp
`T`	Total number of training timestamps
$\tanh(\cdot)$	Hyperbolic tangent activation function
$\tilde{\tanh}(\cdot)$	Linearized hyperbolic tangent activation function
$\boldsymbol{U}$	Weight matrix associated with the hidden states
v	Observation error

$\boldsymbol{v}$	Vector of observation errors
$\text{var}(\cdot)$	Variance operator
$\boldsymbol{w}$	Vector of process errors
$\boldsymbol{W}$	Random variable representing process errors
$\boldsymbol{W}^{(j)}$	Weight matrix for j^{th} hidden layer
$\mathbf{W}$	Regime switching probability
$\mathtt{W}$	Lookback window's length
x or z	Covariate or hidden state variable
$\boldsymbol{x}$ or $\boldsymbol{z}$	Vector of covariates or hidden state variables
$x^{\mathtt{L}}$ or $z^{\mathtt{L}}$	Level hidden state
$x^{\mathtt{LT}}$ or $z^{\mathtt{LT}}$	Local trend hidden state
$x^{\mathtt{KR}}$	Kernel regression hidden state
$\mathtt{X}$	Size of the covariate's vector
y	Observation
$\boldsymbol{y}$	Vector of observations
$\hat{\boldsymbol{y}}$	Predicted observations
$\mathtt{Y}$	Number of outputs
$z_i^{(j)}$	i^{th} hidden unit in j^{th} layer
$\boldsymbol{z}^{(\mathtt{O})}$	Vector of hidden units in the output layer
$\boldsymbol{z}^{\mathtt{B}}$	Baseline hidden state vector
$\tilde{\boldsymbol{z}}^{\mathtt{B}}$	Augmented baseline hidden state vector
$z^{\mathtt{E}}$	Exponential smoothing hidden state
z^{α}	Smoothing parameter hidden state
$z^{\mathtt{V}}$	Hidden state for the error term
$\mathbf{Z}$	Transition probability matrix
β	Slope of synthetic anomaly
ϵ	Learning rate
μ	Expected value
$\boldsymbol{\mu}$	Vector of expected values
$\boldsymbol{\mu}_{t\|t-1}$	Prior mean vector for the hidden states $\boldsymbol{x}$ or $\boldsymbol{z}$ at time t
$\boldsymbol{\mu}_{t\|t}$	Posterior mean vector for the hidden states $\boldsymbol{x}$ or $\boldsymbol{z}$ at time t
$\boldsymbol{\mu}_{t\|\mathtt{T}}$	Smooth estimates for the mean vector of the hidden states $\boldsymbol{x}$ or $\boldsymbol{z}$ at time t
$\tilde{\boldsymbol{\mu}}^{\mathtt{B}}$	Mean vector of the augmented baseline hidden states $\tilde{\boldsymbol{z}}^{\mathtt{B}}$
σ	Standard deviation
$\boldsymbol{\Sigma}$	Covariance matrix

$\tilde{\boldsymbol{\Sigma}}^{\mathsf{B}}$	Covariance matrix of the augmented baseline hidden states $\tilde{\boldsymbol{z}}^{\mathsf{B}}$
$\boldsymbol{\Sigma}_{\boldsymbol{XY}}$	Cross-covariance matrix between variables $\boldsymbol{X}$ and $\boldsymbol{Y}$
$\boldsymbol{\Sigma}_{t\|t-1}$	Prior covariance matrix for the hidden states $\boldsymbol{x}$ or $\boldsymbol{z}$ at time t
$\boldsymbol{\Sigma}_{t\|t}$	Posterior covariance matrix for the hidden states $\boldsymbol{x}$ or $\boldsymbol{z}$ at time t
$\boldsymbol{\Sigma}_{t\|\mathtt{T}}$	Smooth estimates for the covariance matrix of the hidden states $\boldsymbol{x}$ or $\boldsymbol{z}$ at time t
σ_W	Standard deviation associated with the process error term W
σ_V	Standard deviation associated with the observation error term V
σ_W^2	Variance associated with the process error term W
σ_V^2	Variance associated with the observation error term V
$\boldsymbol{\theta}$	Model parameters
$\boldsymbol{\theta}^*$	Optimal values for parameters
$\odot$	Element-wise multiplication
∇	Gradient operator
ϕ	Regression coefficient
$\Omega(\boldsymbol{\theta})$	Regularization term
$\phi(\cdot)$	Activation function
$\tilde{\phi}(\cdot)$	Linearized activation function
$\boldsymbol{\eta}$	Hyperparameters
π	Marginal probability of a regime
ρ	Correlation coefficient
τ	Threshold

Acronyms

AGVI	Approximate Gaussian variance inference
ARIMA	Autoregressive integrated moving average
BDLM	Bayesian dynamic linear models
BNN	Bayesian neural networks
CNN	Convolutional neural networks
ELBO	Evidence lower bound
ETS	Exponential smoothing or error, trend, seasonal
FNN	Feedforward neural networks
GMA	Gaussian multiplicative approximation
GD	Gradient descent
ICOLD	International commission of large dams

KF	Kalman filter
kNN	k-nearest neighbours
MCMC	Markov chain Monte Carlo
MC	Monte Carlo
MLE	Maximum likelihood estimation
MAP	Maximum a posteriori estimation
MSE	Mean squared error
MAE	Mean absolute error
MASE	Mean absolute scaled error
MP	Matrix profile
MCB	Comparison with the best
NN	Neural networks
NLP	Natural language processing
ND	Normalized deviation
PDF	Probability density function
RTS	Rauch-Tung-Striebel
RMSE	Root mean square error
RNN	Recurrent neural networks
RL	Reinforcement learning
ReLU	Rectified linear unit
SHM	Structural health monitoring
SSM	State-space models
SKF	Switching Kalman filter
SKS	Switching Kalman smoother
SGD	Stochastic gradient descent
TAGI	Tractable approximate Gaussian inference
TAGI-RNN	Tractable approximate Gaussian inference recurrent neural networks
TAGI-LSTM	Tractable approximate Gaussian inference long short-term memory
TAGI-GRU	Tractable approximate Gaussian gated recurrent unit neural network
TAGI-LSTM/SSM	Hybrid model that couples TAGI-LSTM and state-space models
TAGI-SKF	Model that uses TAGI-LSTM/SSM in the Switching Kalman Filter framework for anomalies detection
TFT	Temporal Fusion Transformers
VI	Variational inference
VAE	Variational autoencoder

LIST OF APPENDICES

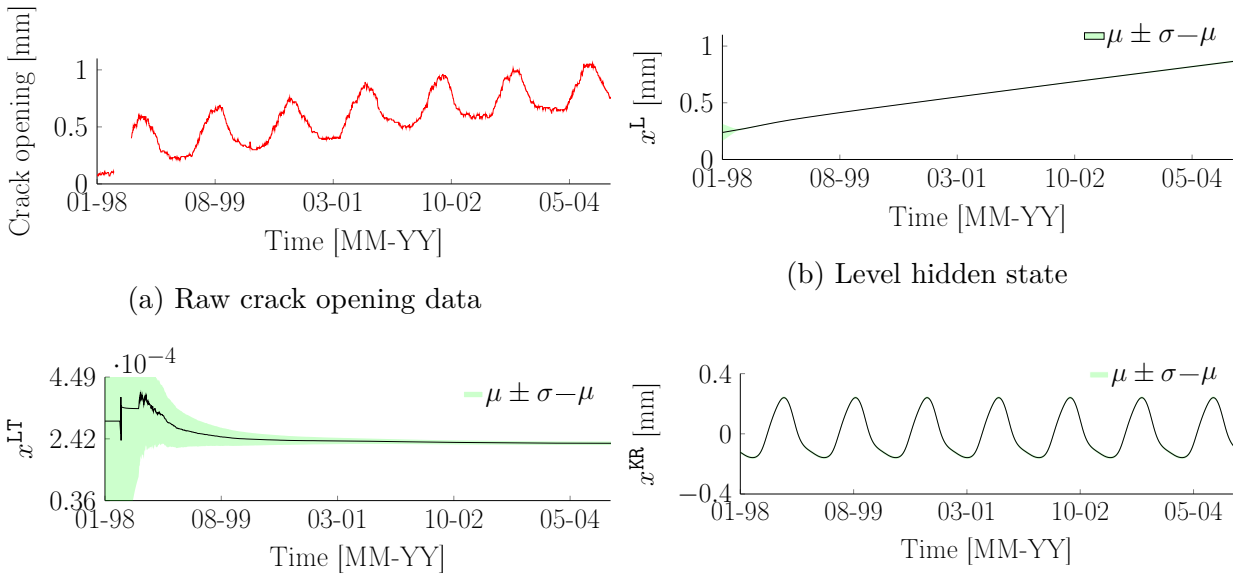
Appendix A Covariances for TAGI-LSTM 120
Appendix B Covariances for TAGI-GRU 124
Appendix C Architecture and Hyperparameters for the TAGI-SKF Model 128
Appendix D Zoom-in figures in Section 4.3.1 129
Appendix E Hyperparameters Tuning for other Anomaly Detection Methods . . . 130

CHAPTER 1 Introduction

1.1 Motivation

Civil infrastructures, including dams, bridges, buildings, and pipelines, play a vital role in the economic growth of every country [14]. Even when properly designed, these structures deteriorate over time due to various factors such as material aging, loading, environmental changes, and poor maintenance, leading to a decline in their condition [15]. Therefore, civil infrastructures have to be monitored and maintained in order to ensure their safety and serviceability.

Sensor-based *structural health monitoring* (SHM) measures various structural responses such as displacements, crack openings, strains, inclinations or accelerations, and use these time series data for inferring the structures' conditions [16]. There are three components to a SHM system, namely, sensors, data management, and data interpretation [17]. The success of SHM typically depends on two main factors: the sensing system and data interpretation methodologies. The advancement of sensing technology has contributed to the widespread adoption of sensors by making them cost-effective while also enhancing their precision and reliability [17, 18]. State-space models (SSM) [19, 20] provide a probabilistic framework for analyzing time series data. The Bayesian Dynamic Linear Models (BDLM) [21] and Switching Kalman Filter (SKF) [22] which are specific types of SSM have been used for SHM data interpretation. Specifically, BDLM has been used to make future predictions about structural responses and to decompose them into interpretable hidden states which give a useful insights for engineers. For example, Figure 1.1a presents a crack opening dataset from a dam in Canada which contains reversible periodic environmental effects as well as measurement noise. BDLM can extract the unobserved level, local trend, and periodic hidden states from the raw data as presented in Figures 1.1b-1.1d. The level hidden state presents the long-term irreversible structural behaviour, the local trend one describes its rate of change, whereas the periodic hidden state models the reversible environmental effects. On the other hand, the SKF has been used to detect anomalies in dams and bridges. Engineers responsible with monitoring the behaviour of structures are typically looking for these changepoints which indicate switches from a regime with known kinematic to a different one. The presence of a change in the kinematic is an indication of a possible anomaly in the structural condition which should trigger further investigations in order to identify its cause along with preventive maintenance actions. The main limitation of the BDLM and SKF methods is that they require extensive feature engineering in order to build models. For example, one needs to



(a) Raw crack opening data

(b) Level hidden state

(c) Local trend hidden state

(d) Periodic hidden state

Figure 1.1 Crack opening data from a dam in Canada and unobserved hidden states extracted from the raw data using the BDLM model.

rely on engineering heuristics in order to manually choose appropriate model components as well as to define the linear or nonlinear dependencies among them. As a result, the application of these methods is limited to a small number of time series, whereas our ultimate goal is to be able to process the data from tens of thousands of sensors in real time.

Neural networks (NN) and especially the recurrent neural networks (RNN) [23,24] have been shown to be effective tools for time series [25]. The main strength of NN is their ability to automatically identify and model, with minimal manual setups, complex patterns that involve dependencies within time series and with explanatory variables. Nevertheless, unlike the BDLM or SKF methods, the results from RNN are typically difficult to interpret [24]. In addition, existing RNN cannot analyzed trended data so that offline preprocessing is needed to remove trends, making them not suited for real time SHM. Coupling RNN and SSM would allow to take advantage of both methods, but it is not a trivial task because their respective inference procedures rely on different mechanisms. SSM are probabilistic models which rely on Bayesian inference, whereas RNN typically optimize their parameters using backpropagation and gradient descent. In the long term, developing neural networks that use Bayesian inference as the learning mechanism could enable online and continual learning in nonstationary conditions. In this context, the parameters could be learnt online as the data become available, and their values can dynamically evolve over time, adapting to new

data and conditions.

1.2 Research Objectives

This thesis aims at developing analytically tractable Bayesian recurrent neural networks (RNN) which employ Bayesian inference for learning the model's parameters. This enables the probabilistic coupling between Bayesian RNN and state-space models (SSM), while using Bayesian inference for learning both the RNN's parameters as well as the posterior for SSM's hidden states. Using this novel method in the existing SSM and Switching Kalman Filter (SKF) frameworks allows to enhance the anomaly detection's scalability in the context of structural health monitoring. The core objectives of this thesis are:

- Develop the analytically tractable Bayesian Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) neural networks using the Tractable Approximate Gaussian Inference (TAGI) method for automatically modeling time series with complex recurrent patterns.
- Develop a hybrid probabilistic model that couples the analytically tractable Bayesian RNN and SSM for analyzing trended data and providing interpretable results along with the prediction uncertainties, while not requiring feature engineering or preprocessing to remove trends.
- Develop a methodology to use the hybrid model in the existing SKF framework for online semi-supervised anomaly detection under nonstationary conditions in structural health monitoring.

1.3 Thesis Outline

The content of this thesis is organized as follows: Chapter 2 presents a literature review on state-space models, neural networks, hybrid models, along with time series' anomaly detection methods. Chapter 3 introduces the mathematical formulations for the analytically tractable Bayesian TAGI-LSTM and TAGI-GRU neural networks. Chapter 4 presents the methodology to create hybrid probabilistic models that couple the analytically tractable Bayesian recurrent neural networks and state-space models. Chapter 5 presents the methodology to use the hybrid model in the SKF framework for online anomaly detection under nonstationary conditions. Finally, Chapter 6 provides the thesis conclusions and its limitations.

1.4 Co-Authored Papers

The list of co-authored papers that are part of this thesis is:

- 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,* (submitted, revisions requested) 2023.

- V.-D. Vuong and J.-A. Goulet, "Probabilistic time series modeling using Bayesian neural networks". 14^{th} *International Conference on Applications of Statistics and Probability in Civil Engineering,* 2023.

- V.-D. Vuong, B. Deka, J.-A. Goulet, P. Côté, and B. Miquel. "Dam Behavior Prediction Using an Ensemble of Bayesian Dynamic Linear Model and Bayesian LSTM Networks". 16^{th} *International Benchmark Workshop on Numerical Analysis of Dams*, International Commission on Large Dams, Ljubljana, Slovenia, 2022.

CHAPTER 2 Literature Review

2.1 Introduction

State-space models (SSM) [19,20] and neural networks (NN) [23,24] are effective tools for time series analysis. The main strength of NN is their ability to automatically identify complex patterns while requiring minimal manual setups [26]. Nevertheless, without making assumptions about the data, the results from NN are typically difficult to interpret [4]. By contrast, SSM provide a probabilistic framework for decomposing time series into interpretable patterns such as the level, trend and seasonality. Hybrid models that combine NN and SSM allow to take advantage of both methods. In this chapter, Section 2.2 reviews state-space models and its applications in structural health monitoring. Section 2.3 reviews popular neural network architectures, as well as algorithms used to trained them. Section 2.4 reviews several hybrid models that combine SSM and NN. Finally, Section 2.5 reviews anomaly detection methods used for time series.

2.2 State-Space Models

State-Space Models (SSM) [19,20,27] is a probabilistic method for modeling time series data. This section introduces the basic principles of SSM, the Kalman filter [28], Bayesian Dynamic Linear Models (BDLM) [21], along with the Switching Kalman Filter (SKF) [22] which are specific types of SSM. In the context of SHM, BDLM has been used to make predictions about structural responses and to decompose them into interpretable components which give useful insights for engineers about structural conditions, whereas SKF has been used to detect anomalies in dams and bridges.

SSM have the ability to separate observed system's responses $\boldsymbol{y}_t = [y_1, y_2, \cdots, y_{\mathtt{Y}}]_t^\intercal \in \mathbb{R}^{\mathtt{Y}}$ into hidden states $\boldsymbol{x}_t = [x_1, x_2, \cdots, x_{\mathtt{X}}]_t^\intercal \in \mathbb{R}^{\mathtt{X}}$, that are used to model the latent patterns of a system over time such as the level, trend and periodic ones, where their values can change from one time step to another, reflecting the system's evolution. A SSM consists of a transition and an observation model. The transition model is characterized by a conditional probability density function (PDF) $f(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ describing the relation between the hidden states at two consecutive timestamps $\boldsymbol{x}_t$ and $\boldsymbol{x}_{t-1}$. Under the Markovian hypothesis, the hidden states $\boldsymbol{x}_t$ at time t depends only on the hidden states $\boldsymbol{x}_{t-1}$, and are independent from all the hidden states $\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{t-2}\}$ given $\boldsymbol{x}_{t-1}$ so that $f(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = f(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1})$, where $1:t-1$ is the short-hand notation for $\{1, 2, \cdots, t-1\}$. The observation model is characterized

by the conditional PDF $f(\boldsymbol{y}_t|\boldsymbol{x}_t)$ describing how the observations $\boldsymbol{y}_t$ are related to the hidden states $\boldsymbol{x}_t$. These two models are defined in a generic form by

$$\begin{aligned} \boldsymbol{x}_t &= g(\boldsymbol{x}_{t-1}, \boldsymbol{w}_t), \\ \boldsymbol{y}_t &= h(\boldsymbol{x}_t, \boldsymbol{v}_t), \end{aligned} \tag{2.1}$$

where $g(\cdot)$ and $h(\cdot)$ are the transition and observation functions, $\boldsymbol{w}_t$ and $\boldsymbol{v}_t$ are the process and observation errors.

The objective of SSM is to estimate the posterior $f(\boldsymbol{x}_t|\boldsymbol{y}_{1:t})$ for the hidden states over time using the Bayes' theorem such that

$$f(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}) = \frac{f(\boldsymbol{y}_t|\boldsymbol{x}_t) \cdot f(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1})}{f(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1})}, \tag{2.2}$$

where $f(\boldsymbol{y}_t|\boldsymbol{x}_t)$ is the likelihood, $f(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1})$ is the prior, and $f(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1})$ is the evidence. Gaussian filters [27, 29] assume that the likelihood $f(\boldsymbol{y}_t|\boldsymbol{x}_t)$ and the prior $f(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1})$ are Gaussians so that the posterior $f(\boldsymbol{x}_t|\boldsymbol{y}_{1:t})$ is also Gaussian following the properties of conjugate priors. Gaussian filters can be linear or nonlinear depending on the transition and observations used in Equation 2.1. Linear Gaussian filters employ affine transition and observation models so that the Equation 2.2 can be estimated analytically to obtain the posterior $f(\boldsymbol{x}_t|\boldsymbol{y}_{1:t})$ using a filtering technique known as the Kalman filter (KF) [28]. Nevertheless, in many real-world applications, the assumption of linearity may not hold because nonlinear transition and observation models are required to represent the system. In such cases, the Equation 2.2 is intractable, and an analytical solution for the posterior $f(\boldsymbol{x}_t|\boldsymbol{y}_{1:t})$ is no longer available. There are two major approaches to approximate this posterior. The local approach assumes a known parametric distribution family for the posterior PDF, and approximates it numerically. For example, the Extended Kalman filter [30] linearizes the nonlinear transition and observation functions so that the posterior $f(\boldsymbol{x}_t|\boldsymbol{y}_{1:t})$ can be obtained using the standard KF's equations, whereas the Unscented Kalman [31] and Cubature Kalman [32] filters approximate the posterior PDF's moments using weighted samples. On the other hand, global approaches such as particle filters [19, 33] do not assume any distribution type for the posterior but approximate it through sampling. Compared to the local approaches, the global ones have the limitation of having a high computational costs owing to the sampling techniques [29]. The next sections review the Kalman filter and its closed-form solutions, the Bayesian Dynamic Linear Models and the Switching Kalman Filter which are specific types of SSM.

2.2.1 Kalman Filter

The Kalman Filter (KF) [28] is a type of Gaussian filters where the transition and observation models are described by affine functions

$$\begin{aligned} \boldsymbol{x}_t &= \mathbf{A}\boldsymbol{x}_{t-1} + \boldsymbol{w}_t, \qquad \boldsymbol{w}_t : \boldsymbol{W} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}), \\ \boldsymbol{y}_t &= \mathbf{F}\boldsymbol{x}_t + \boldsymbol{v}_t, \qquad \boldsymbol{v}_t : \boldsymbol{V} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}), \end{aligned}$$

where the process $\boldsymbol{w}_t$ and observation $\boldsymbol{v}_t$ errors are additive, zero-mean independent and identically distributed (i.i.d.) random variables, $\mathbf{A}$ and $\mathbf{F}$ are the transition and observation matrices, $\mathbf{Q}$ and $\mathbf{R}$ are the covariance matrices for the process and observation errors.

The KF procedure, which consists in a prediction and an update step, provides a closed-form solution for the posterior presented in Equation 2.2. The prediction step computes the first two moments, $\mathbb{E}[\boldsymbol{X}_t|\boldsymbol{y}_{1:t-1}] \equiv \boldsymbol{\mu}_{t|t-1}$ and $\text{cov}(\boldsymbol{X}_t|\boldsymbol{y}_{1:t-1}) \equiv \boldsymbol{\Sigma}_{t|t-1}$, for the prior $\boldsymbol{X}_t|\boldsymbol{y}_{1:t-1} \equiv \boldsymbol{X}_{t|t-1} \sim \mathcal{N}(\boldsymbol{\mu}_{t|t-1}, \boldsymbol{\Sigma}_{t|t-1})$, whereas the update step computes the posterior mean vector and covariance matrix for the hidden states $\boldsymbol{X}_{t|t} \sim \mathcal{N}(\boldsymbol{\mu}_{t|t}, \boldsymbol{\Sigma}_{t|t})$ using

$$\begin{aligned} &\text{Prediction step:} \\ \boldsymbol{\mu}_{t|t-1} &= \mathbf{A}\boldsymbol{\mu}_{t-1|t-1}, \\ \boldsymbol{\Sigma}_{t|t-1} &= \mathbf{A}\boldsymbol{\Sigma}_{t-1|t-1}\mathbf{A}^\intercal + \mathbf{Q}, \\ &\text{Update step:} \\ \boldsymbol{\mu}_{t|t} &= \boldsymbol{\mu}_{t|t-1} + \mathbf{K}_t\mathbf{r}_t, \\ \boldsymbol{\Sigma}_{t|t} &= (\mathbf{I} - \mathbf{K}_t\mathbf{F})\,\boldsymbol{\Sigma}_{t|t-1}, \\ \mathbf{r}_t &= \boldsymbol{y}_t - \mathbf{F}\boldsymbol{\Sigma}_{t|t-1}, \\ \mathbf{K}_t &= \boldsymbol{\Sigma}_{t|t-1}\mathbf{F}^\intercal\mathbf{G}_t^{-1}, \\ \mathbf{G}_t &= \mathbf{F}\boldsymbol{\Sigma}_{t|t-1}\mathbf{F}^\intercal, \end{aligned}$$

where $\boldsymbol{\mu}_{t-1|t-1}$ and $\boldsymbol{\Sigma}_{t-1|t-1}$ are the posterior mean vector and covariance matrix at time $t-1$, and $\mathbf{I}$ is the identity matrix having the same size as $\boldsymbol{\Sigma}_{t-1|t-1}$, $\mathbf{K}_t$ is the Kalman gain matrix, and $\mathbf{G}_t$ is the innovation covariance matrix. The KF's equations can be summarized by a filter operator defined as

$$(\boldsymbol{\mu}_{t|t}, \boldsymbol{\Sigma}_{t|t}, \mathcal{L}_t) = \text{Filter}(\boldsymbol{y}_t, \boldsymbol{\mu}_{t-1|t-1}, \boldsymbol{\Sigma}_{t-1|t-1}, \mathbf{A}, \mathbf{F}, \mathbf{Q}, \mathbf{R}), \tag{2.3}$$

where $\boldsymbol{y}_t$ is the observation vector and $\mathcal{L}_t$ is the likelihood obtained from

$$\mathcal{L}_t = \mathcal{N}(\boldsymbol{y}_t; \mathbf{F}\boldsymbol{\mu}_{t|t-1}, \mathbf{F}\boldsymbol{\Sigma}_{t|t-1}\mathbf{F}^\intercal + \mathbf{R}). \tag{2.4}$$

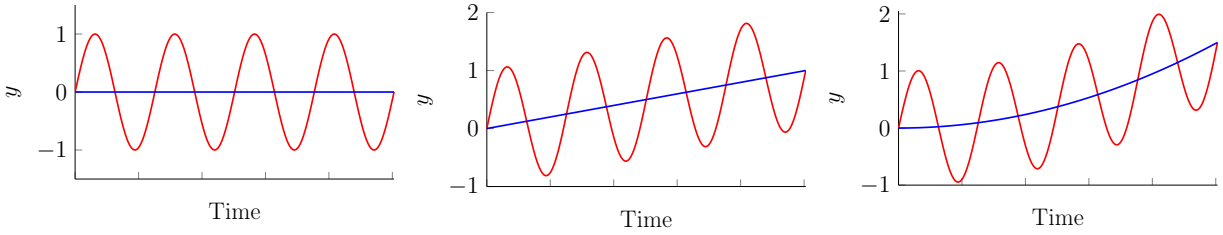
The following section reviews the Bayesian Dynamic Linear Model which is a special type of linear Gaussian filter as well as illustrates its applications in SHM.

2.2.2 Bayesian Dynamic Linear Models

Bayesian Dynamic Linear Models (BDLM) [20, 21] is a type of linear Gaussian filters where the hidden state vector $\boldsymbol{x}_t$ and model matrices $\{\mathbf{A}, \mathbf{F}, \mathbf{Q}, \mathbf{R}\}$ are built from generic subcomponents. Each subcomponent i has its own set of predefined $\boldsymbol{x}_t^i$ and $\{\mathbf{A}^i, \mathbf{F}^i, \mathbf{Q}^i, \mathbf{R}^i\}$ from which the global hidden state vector and matrices are assembled.

The BDLM's subcomponents can be categorized into three groups including baseline, periodic and autoregressive ones. For SHM applications, the baseline subcomponents model the evolution of the irreversible patterns for the quantity of interest without external effects. There are three typical baseline subcomponents namely the local level (`LL`), local trend (`LT`) and local acceleration (`LA`) [20]. The local level one is suited for modeling data that has a zero-value speed (Figure 2.1a), whereas the local trend is used to model data displaying a locally constant speed (Figure 2.1b), while the local acceleration can capture a locally constant acceleration (Figure 2.1c). The periodic subcomponents are used to model recurrent patterns. A first periodic subcomponent uses the Fourier's representation for modeling harmonic patterns, while the Kernel regression (`KR`) [34] models non-harmonic ones. The autoregressive (`AR`) subcomponent models the residuals that cannot be captured by other subcomponents. For evaluating a model's performance, it is important to check whether the autoregressive hidden state has a zero mean and a constant variance. Such a case typically indicates that the model subcomponents are adequately capturing the underlying patterns in the data. Otherwise, for non-zero mean or non-constant variances, there are typically patterns left unexplained in the `AR` term indicating that the model can be improved.

Typically, the prior knowledge for the initial hidden states $\boldsymbol{x}_0$ is predefined by users, and the



(a) Local level (b) Local trend (c) Local acceleration

Figure 2.1 Three synthetic time series having different baselines, but sharing the same periodic pattern. The red line presents the data, while the blue line presents the baseline.

set of parameters $\boldsymbol{\mathcal{P}}$ involved in the definition of the model matrices can be estimated either using optimization techniques such as the maximum likelihood (MLE) and the maximum a posteriori estimation (MAP) [27, 35], or through Bayesian estimation methods [27] such as Markov chain Monte Carlo (MCMC) [36] or the Laplace approximation [37]. In addition, the Gaussian Multiplicative Approximation (GMA) [38] method can estimate the autoregression coefficient for the `AR` subcomponent by modeling it as a hidden state, and the Approximate Gaussian Variance Inference (AGVI) method [7] allows the process error's variances and covariances to be estimated online as hidden states.

BDLM can take into account linear dependencies between a target variable and multiple explanatory variables. For example, consider the effects of a reservoir's water level and the air temperature on a dam's displacement. For taking them into account, the explanatory variables first need to be decomposed into their own hidden states, then the relevant hidden states are included in the hidden state vector for the target variable. The state-based regression component [38] allows to model nonlinear dependencies with respect to the explanatory variables' values. Detailed descriptions about modeling dependencies in BDLM can be found in [20] and [38].

Figure 2.2 shows a dataset from the international dam forecasting Benchmark Workshop organized by the International Commission of Large Dams (ICOLD) [11]; it includes data from an inverted pendulum measuring a dam's displacement, the reservoir's water level, and the air temperature. The task consists in predicting the pendulum data for the test set from 2013-2018 using the water level and temperature as explanatory variables. The BDLM model for this task is provided in [6], here we summarize it to show the capacity as well as the limitations of the existing method. For considering the dependencies with the pendulum data, the water level displayed in Figure 2.2b is decomposed into a mean-centered component $x^{\mathtt{WL,1}}$ and an average long-term trend $x^{\mathtt{WL,2}}$ as presented in Figure 2.2c, and the moving-averages ($x^{\mathtt{T-MA}}$) of temperatures as presented in Figure 2.2d are calculated to represent the lag-effect in temperature changes.

The explanatory components, i.e., $x^{\mathtt{WL,1}}$, $x^{\mathtt{WL,2}}$ and $x^{\mathtt{T-MA}}$, are then used to build the hidden state vector of the dam response as

$$\boldsymbol{x}_t = [x^{\mathtt{L}}\ \boldsymbol{x}^{\mathtt{KR}}\ \underbrace{\phi_t^1 x^{\mathtt{WL,1}}}_{x^{\mathtt{D}_1}}\ \underbrace{\phi_t^2 x^{\mathtt{WL,2}}}_{x^{\mathtt{D}_2}}\ \phi_3 x^{\mathtt{T-MA}}\ x^{\mathtt{AR}}]_t,$$

where $x^{\mathtt{L}}$, $\boldsymbol{x}^{\mathtt{KR}}$ and $x^{\mathtt{AR}}$ are the predefined hidden states for the local level (`LL`), Kernel regression (`KR`) and autoregressive (`AR`) subcomponents; ϕ_t^1, ϕ_t^2 and ϕ^3 are the regression coefficients which define the contributions of the explanatory components on the target variable dam's

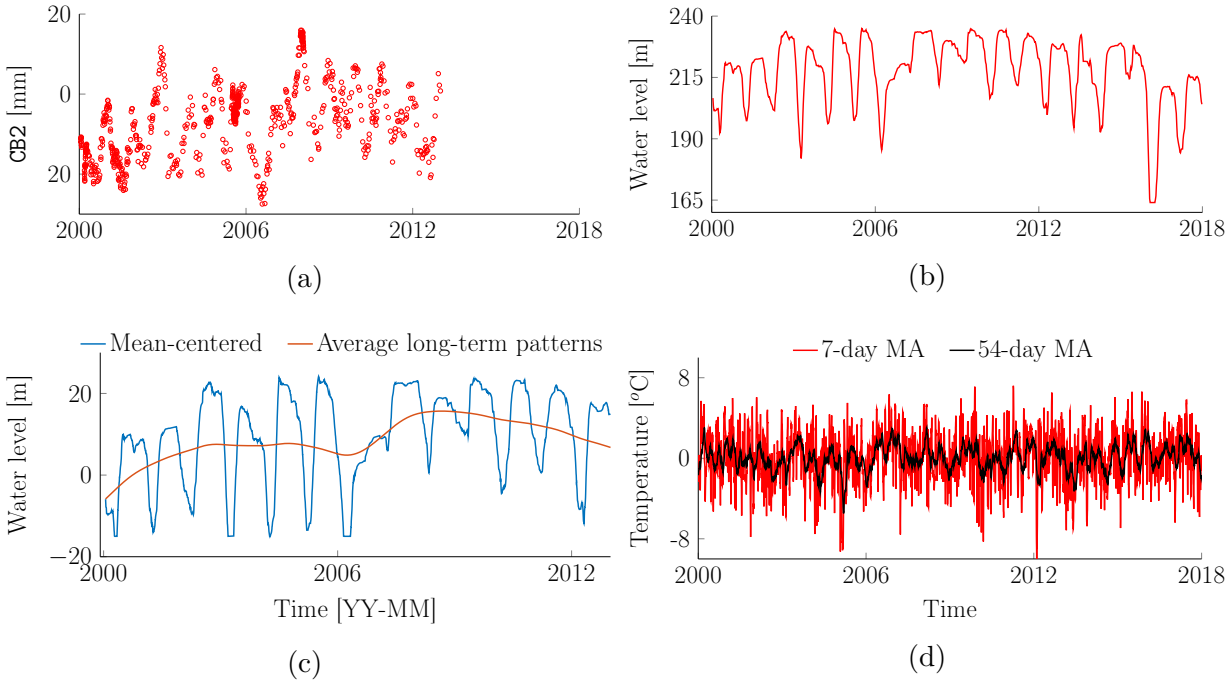


Figure 2.2 Dataset from the international Benchmark Workshop organized by the International Commission of Large Dams (ICOLD) [11], (a) inverted pendulum, (b) raw reservoir's water level, (c) components extracted from the raw reservoir's water level, and (d) moving-averages for air temperature.

displacement. Note that ϕ_t^1 and ϕ_t^2 are modelled by the state-based regression component [38], and their values can change as a function of the explanatory components allowing to model nonlinear dependencies with respect to the water level, while ϕ^3 is constant so that it only models a linear dependency with respect to the air temperature.

Figure 2.3 shows the predictions, hidden states as well as their relative importance for the pendulum dataset. The ability of BDLM to provide interpretable results, i.e., decomposing time series data into subcomponents gives useful insights for engineers about the underlying baseline structural condition without external effects. However, the key limitation of BDLM is that the exploration process in order to find the best set of explanatory components as presented in the abovementioned example needs to be hand-crafted for each dataset. This is a time-consuming task that requires expert knowledge which limits the BDLM's ability to scale to large datasets. The BDLM presented in this section can only model a single regime; the next section presents the Switching Kalman Filter (SKF) [22] which can use multiple BDLM to model nonstationary dynamic systems.

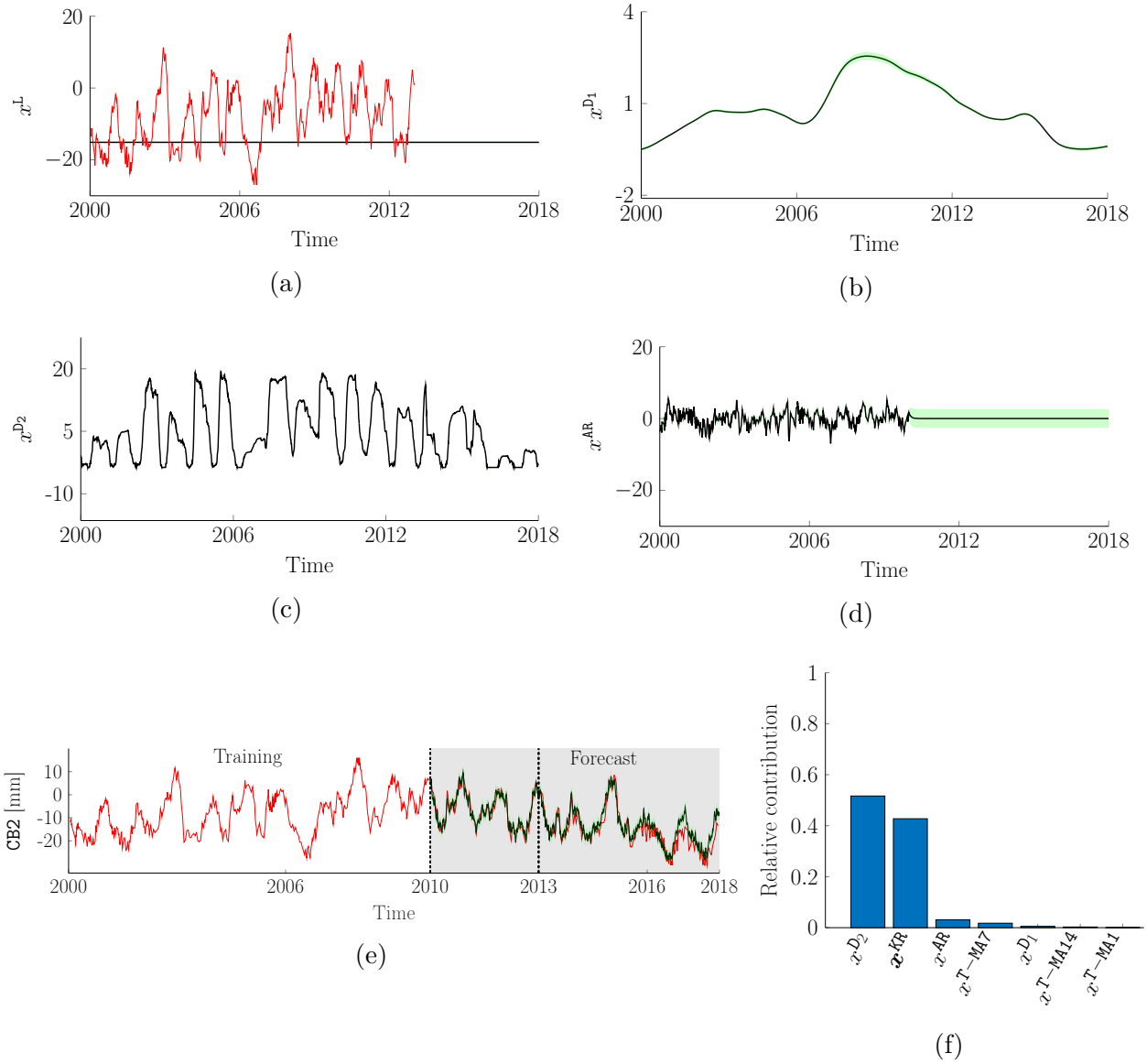


Figure 2.3 Hidden state estimation from BDLM model for the CB2 pendulum dataset, (a) level component, (b) contribution of the water level's average long-term trend $x^{\mathtt{WL},2}$ of the water level, (c) contribution of the water level's mean-centered component $x^{\mathtt{WL},1}$, (d) autoregressive component, (e) predictions on the validation and test sets, (f) relative importance of each component. The figures are reused from [6].

2.2.3 Switching Kalman Filter

The Switching Kalman Filter (SKF) [22] is a variant of the Kalman filter used in nonstationary scenarios where the underlying system dynamics can switch between different regimes. Figure 2.4 shows a time series displaying two distinct regimes, the first one has a zero-value speed, whereas the second one has a constant speed. The SKF method allows to model non-stationary data having multiple regimes like in this example. Lets consider a discrete regime state variable $s_t \in \{1, 2, \cdots, \mathtt{S}\}$, where $\mathtt{S}$ is the number of regimes. At a time step t, the SKF maintains $\mathtt{S}$ models where each one has its own hidden state vector. When these models transit to the next time step $t+1$, they can either stay in their original regime state or transit to any others. With two regimes as displayed in Figure 2.4, the model transitions from the time step t to $t+1$ are described as

$$\begin{aligned} &\text{Model 1:} \quad s_t = 1 \rightarrow s_{t+1} = 1 \quad \text{or} \quad s_t = 1 \rightarrow s_{t+1} = 2, \\ &\text{Model 2:} \quad s_t = 2 \rightarrow s_{t+1} = 2 \quad \text{or} \quad s_t = 2 \rightarrow s_{t+1} = 1. \end{aligned} \tag{2.5}$$

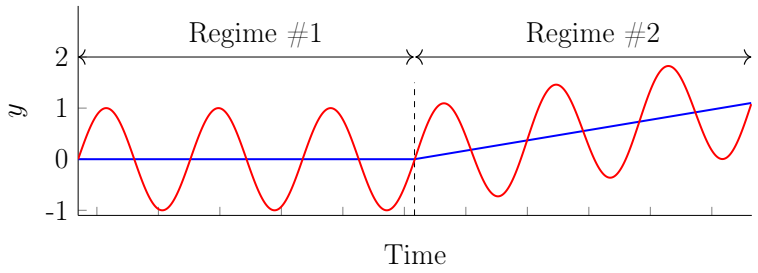


Figure 2.4 Time series with two regimes where the first one has a constant trend and the second one has a linear positive trend. The blue line presents the baseline.

Let the subscripts i and j represent the state $s_{t-1} = i$ at time $t-1$ and $s_t = j$ at time t, and $i(j)$ denotes the current state $s_t = j$ given the past state $s_{t-1} = i$. Each model transition as presented in Equation 2.5 leads to a different hidden state vector $\boldsymbol{X}_{t|t}^{i(j)} \sim \mathcal{N}(\boldsymbol{\mu}_{t|t}^{i(j)}, \boldsymbol{\Sigma}_{t|t}^{i(j)})$ where its mean vector and covariance matrix are defined by

$$(\boldsymbol{\mu}_{t|t}^{i(j)}, \boldsymbol{\Sigma}_{t|t}^{i(j)}, \mathcal{L}_t^{i(j)}) \quad = \quad \text{SKF-Filter}(\boldsymbol{y}_t, \boldsymbol{\mu}_{t-1|t-1}^{i}, \boldsymbol{\Sigma}_{t-1|t-1}^{i}, \mathbf{A}^{i(j)}, \mathbf{F}^{i(j)}, \mathbf{Q}^{i(j)}, \mathbf{R}^{i(j)}), \tag{2.6}$$

where the filter operator and the likelihood $\mathcal{L}_t^{i(j)}$ are defined in Equations 2.3 and 2.4, and $\{\mathbf{A}^{i(j)}, \mathbf{F}^{i(j)}, \mathbf{Q}^{i(j)}, \mathbf{R}^{i(j)}\}$ are the model matrices defining the transition. After the transition at each time step, the number of models increases from $\mathtt{S}$ to $\mathtt{S}^2$ so that a collapse step is needed in order to keep the number from growing exponentially.

In order to maintain only $\mathtt{S}$ models globally at each time, the collapse step uses the Gaussian

Mixture reduction [39] to combine $\mathtt{S}$ filtering models arriving at a same regime into a single model. The collapse operator

$$(\boldsymbol{\mu}^{j}_{t|t}, \boldsymbol{\Sigma}^{j}_{t|t}, \pi^{j}_{t|t}) \;=\; \text{Collapse}(\boldsymbol{\mu}^{i(j)}_{t|t}, \boldsymbol{\Sigma}^{i(j)}_{t|t}, \mathrm{M}^{i(j)}_{t-1|t}), \tag{2.7}$$

is defined by the following equations

$$\begin{aligned}
\boldsymbol{\mu}^{j}_{t|t} &= \textstyle\sum_i \boldsymbol{\mu}^{i(j)}_{t|t} \mathrm{W}^{i(j)}_{t-1|t}, \\
\boldsymbol{\Sigma}^{j}_{t|t} &= \textstyle\sum_i \left[\mathrm{W}^{i(j)}_{t-1|t} \cdot (\boldsymbol{\Sigma}^{i(j)}_{t|t} + \mathbf{m}\mathbf{m}^{\intercal}) \right], \\
\mathbf{m} &= \boldsymbol{\mu}^{i(j)}_{t|t} - \boldsymbol{\mu}^{j}_{t|t}, \\
\mathrm{W}^{i(j)}_{t-1|t} &= \Pr(s_{t-1} = i | s_t = j, \boldsymbol{y}_{1:t}) = \mathrm{M}^{i(j)}_{t-1,t|t} / \pi^{j}_{t|t}, \\
\pi^{j}_{t|t} &= \textstyle\sum_i \mathrm{M}^{i(j)}_{t-1,t|t}, \\
\mathrm{M}^{i(j)}_{t-1,t|t} &= \Pr(s_{t-1} = i, s_t = j | \boldsymbol{y}_{1:t}) = \frac{\mathcal{L}^{i(j)}_{t} \cdot \mathrm{Z}^{i(j)} \cdot \pi^{i}_{t-1|t-1}}{\sum_i \sum_j \mathcal{L}^{i(j)}_{t} \cdot \mathrm{Z}^{i(j)} \cdot \pi^{i}_{t-1|t-1}},
\end{aligned} \tag{2.8}$$

where $\boldsymbol{\mu}^{j}_{t|t}$ and $\boldsymbol{\Sigma}^{j}_{t|t}$ are the mean vector and covariance matrix for the posterior $\boldsymbol{X}^{j}_{t|t} \sim \mathcal{N}(\boldsymbol{\mu}^{j}_{t|t}, \boldsymbol{\Sigma}^{j}_{t|t})$ which is a mixture PDF describing the state $s_t = j$, $\mathrm{W}^{i(j)}_{t-1|t}$ is the regime switching probability, $\mathrm{M}^{i(j)}_{t-1,t|t}$ is the joint probability, $\pi^{i}_{t-1|t-1}$ and $\pi^{j}_{t|t}$ are the marginal probabilities of $s_{t-1} = i$ and $s_t = j$, and $\mathrm{Z}^{i(j)}$ is the prior probability of transitioning from a regime i at time $t-1$ to a regime j at time t.

Figure 2.5 illustrates the SKF's filtering and collapse steps for a case involving two regimes. In the context of SHM, if we consider these two regimes as normal and abnormal, the SKF allows quantifying the probabilities of regime switches ($s_t = 1 \rightarrow s_{t+1} = 2$ and $s_t = 2 \rightarrow s_{t+1} = 1$) which can be used as a proxy indicating the presence of anomalies. In a more advanced decision making framework relying on reinforcement and imitation learning [40,41], the probabilities are part of the state vector used by virtual agents in order to trigger alarms.

2.3 Neural Networks

Neural networks (NN) are a class of mathematical models loosely inspired by biological neurons [24]. For a supervised learning problem, the objective of a NN is to approximate a function $\boldsymbol{y} = g(\boldsymbol{x}, \boldsymbol{\theta})$ given the data $\mathcal{D} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{\mathtt{D}}$, where $\mathtt{D}$ is the number of observations, $\boldsymbol{x}_i \in \mathbb{R}^{\mathtt{X}}$ are the input features, $\boldsymbol{y}_i \in \mathbb{R}^{\mathtt{Y}}$ are the observations, and $\boldsymbol{\theta}$ are the neural network's parameters. During training, these parameters $\boldsymbol{\theta}$ are adjusted to minimize the difference between the model's predictions $\hat{\boldsymbol{y}}$ and the actual target values $\boldsymbol{y}$.

An architecture refers to the structure of a neural network that defines how its nodes and

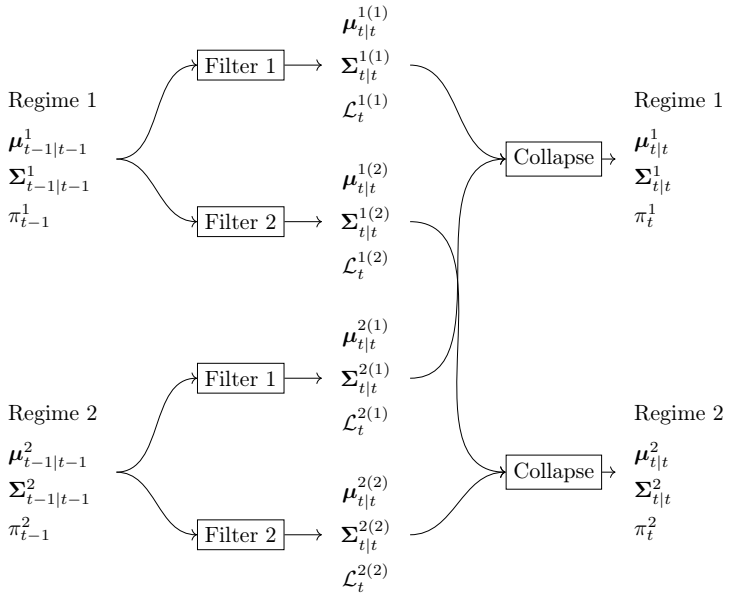


Figure 2.5 Switching Kalman Filter filtering and collapse steps for a case involving two regimes. The number of models increases from 2 to 4 after the filtering step, while the collapse step reduces and keeps 2 models at each time.

layers are connected to each other. Figure 2.6 presents a representation of a simple NN where the nodes are represented by circles, and the connections between nodes are represented by arrows. Each node has an associated bias term b, and a connection is associated with a weight parameter w for determining its strength. NN's nodes are typically organized into layers. The most common layers are the input layer, hidden layers, and output layer. In the simplest setup, data is fed into the input layer, processed through the hidden layers, and returns the output. There are various types of NN's architectures where each is best suited for a specific type of data. This section first reviews several popular architectures namely the feedforward, recurrent, convolutional and transformer neural networks, then reviews common algorithms for training them.

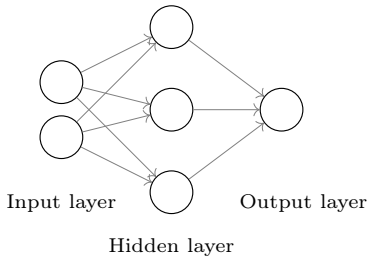


Figure 2.6 Representation of a simple neural network where the circle represents a node and the arrow represents a connection between two nodes.

2.3.1 Neural Network Architecture

Fully Connected Feedforward Neural Networks

Fully connected feedforward neural networks (FNN) [23,24] is one of the fundamental architectures. It consists in an input layer $\boldsymbol{x} \in \mathbb{R}^{\mathtt{X}}$, one or multiple hidden layers $\boldsymbol{z}^{(j)}$, and an output layer $\boldsymbol{z}^{(\mathtt{O})} \in \mathbb{R}^{\mathtt{Y}}$, where $\mathtt{X}$ and $\mathtt{Y}$ are the numbers of input and output, and $(\mathtt{O})$ indicates the output layer. Figure 2.7a presents a FNN where the input and the hidden layer both have two nodes, and the output layer has a single node. In FNN, all the nodes between two consecutive layers are connected. The hidden unit z_i is the weighted linear combination of the input units as defined in

$$\begin{aligned} z_1 &= x_1 w_{1,1}^{(\mathtt{0})} + x_2 w_{2,1}^{(\mathtt{0})} + b_1^{(\mathtt{0})}, \\ z_2 &= x_1 w_{1,2}^{(\mathtt{0})} + x_2 w_{2,2}^{(\mathtt{0})} + b_2^{(\mathtt{0})}, \end{aligned} \tag{2.9}$$

where w is the weight and b is the bias parameter, and $(\mathtt{0})$ represents the input layer. The hidden unit z_i is transformed by a nonlinear activation function $\phi(\cdot)$ to obtain the corresponding activation unit a_i as

$$a_i = \phi(z_i).$$

There are various kind of activation functions such as the Rectified linear unit (ReLU), Hyperbolic tangent (tanh) and sigmoid (σ). These activation functions allow NN to model complex nonlinear relationships. Figure 2.7b presents an equivalent representation of the FNN presented in Figure 2.7a, where the activation units $\boldsymbol{a}$ are shown explicitly. Note that there is no weight parameter associated with the connection from the z to a. In this thesis, when using the representation showing only the hidden states $\boldsymbol{z}$, it is implied that the

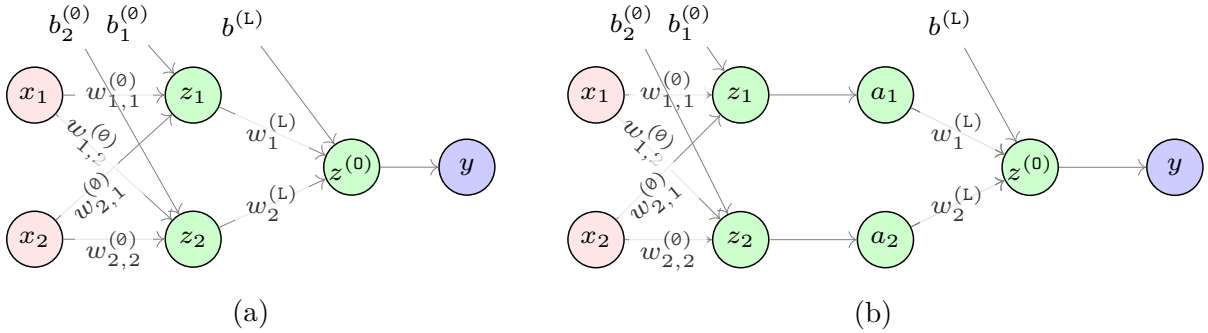


Figure 2.7 Equivalent representations of a single-hidden-layer fully connected feedforward neural network where (a) only the hidden states $\boldsymbol{z}$, and (b) both hidden states $\boldsymbol{z}$ and activations units $\boldsymbol{a}$ are shown.

activation functions are also applied.

The output $\boldsymbol{z}^{(0)}$ is calculated as the weighted sum of the previous layer's hidden units using

$$z^{(0)} = w_1^{(L)} a_1 + w_2^{(L)} a_2 + b^{(L)}, \tag{2.10}$$

where (L) represents the last hidden layer. The relationship between the output hidden states $\boldsymbol{z}^{(0)}$ and the observations $\boldsymbol{y}$ is described by the observation equation

$$y = z^{(0)} + v,$$

where v is an error term representing both observation and prediction errors. Extending from a one-hidden-layer FNN in Figure 2.7, Figure 2.8 presents a FNN having L hidden layers where each has A units such that $\boldsymbol{z}^{(i)} \in \mathbb{R}^{A}$. For this network, Equations 2.9 and 2.10 can be generalized and written in their matrix forms to define the hidden states of the i^{th} hidden layer with $i = 1 : L$, and the output vector $\boldsymbol{z}^{(0)}$ as

$$\begin{aligned} \boldsymbol{z}^{(i)} &= \mathbf{W}^{(i-1)} \boldsymbol{a}^{(i-1)} + \boldsymbol{b}^{(i-1)}, \\ \boldsymbol{z}^{(0)} &= \mathbf{W}^{(L)} \boldsymbol{a}^{(L)} + \boldsymbol{b}^{(L)}, \end{aligned} \tag{2.11}$$

where $\mathbf{W}^{(i-1)} \in \mathbb{R}^{A^2}$, $\mathbf{W}^{(L)} \in \mathbb{R}^{A \times Y}$, $\boldsymbol{b}^{(i-1)} \in \mathbb{R}^{A}$, and $\boldsymbol{b}^{(L)} \in \mathbb{R}^{Y}$ are the weight matrices and bias vectors.

Time series data is inherently sequential such that the temporal dependency plays an important role. However, FNN treat each instance $\{\boldsymbol{x}_t, \boldsymbol{y}_t\}$ independently where each input $\boldsymbol{x}_t$ is passed through the FNN presented in a compact form in Figure 2.8b to obtain the output $\boldsymbol{y}_t$ without considering the connections between the observations $\boldsymbol{y}_t$ at different time steps. The next section reviews the recurrent neural networks which are designed to consider these temporal dependencies.

Recurrent Neural Networks

Recurrent neural networks (RNN) [23,24] are designed to analyze sequential data while taking into account the temporal dependencies. Figure 2.9 presents a time-unrolled representation of a one-hidden-layer RNN consisting in an input layer $\boldsymbol{x} \in \mathbb{R}^{X}$, one hidden layer $\boldsymbol{z} \in \mathbb{R}^{A}$, and an output layer $\boldsymbol{z}^{(0)} \in \mathbb{R}^{Y}$; where X, A and Y are the sizes for the input, hidden states, and the output, respectively. At each time step t, a RNN is similar to the FNN presented in Figure 2.8b except for the additional recurrent connections between consecutive time steps. These connections are defined between the output and the next time step's hidden states

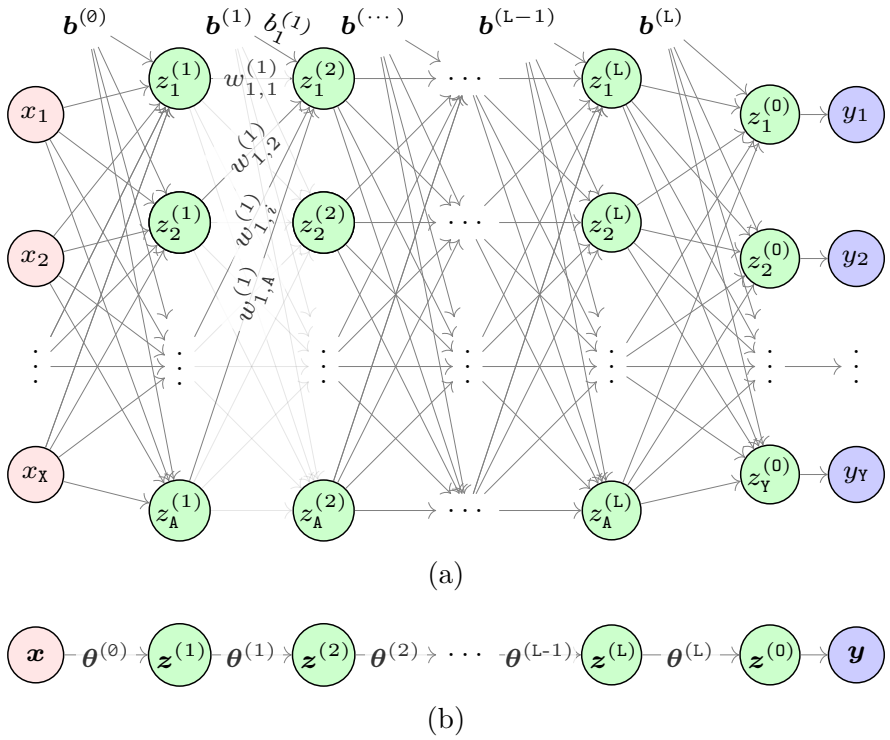


Figure 2.8 The (a) full representation of a multi-hidden-layer FNN where each node represents a unit, and the arrow between any two nodes represents the forward connection, and (b) the equivalent compact representation of the same network where each node represents a vector, and $\boldsymbol{\theta} = \{\boldsymbol{w}, \boldsymbol{b}\}$. The figures are reproduced from [12].

($\boldsymbol{z}_{t-1}^{(\mathtt{O})}$ and $\boldsymbol{z}_t$), and the hidden states at two consecutive time steps ($\boldsymbol{z}_{t-1}$ and $\boldsymbol{z}_t$) as presented by the arrows connecting these variables in Figure 2.9.

The role of the recurrent connections is to use the information from the previous time step $t-1$ ($\boldsymbol{z}_{t-1}^{(\mathtt{O})}$ and $\boldsymbol{z}_{t-1}$) to estimate the hidden states $\boldsymbol{z}_t$ at the current time t. The RNN presented in Figure 2.9 is defined by the following recursive equations

$$\begin{aligned} \boldsymbol{z}_t &= \mathbf{W}^{(\mathtt{0})}\boldsymbol{x}_t + \mathbf{U}\boldsymbol{a}_{t-1} + \mathbf{H}\boldsymbol{z}_{t-1}^{(\mathtt{O})} + \boldsymbol{b}^{(\mathtt{0})}, \\ \boldsymbol{a}_t &= \phi(\boldsymbol{z}_t), \\ \boldsymbol{z}_t^{(\mathtt{O})} &= \mathbf{W}^{(\mathtt{L})}\boldsymbol{a}_t + \boldsymbol{b}^{(\mathtt{L})}, \end{aligned} \tag{2.12}$$

where $\mathbf{W}^{(\mathtt{0})} \in \mathbb{R}^{\mathtt{A}\times\mathtt{X}}$, $\mathbf{U} \in \mathbb{R}^{\mathtt{A}^2}$, and $\mathbf{H} \in \mathbb{R}^{\mathtt{Y}^2}$ are the weight matrices associated with the input $\boldsymbol{x}_t$, the activation units $\boldsymbol{a}_{t-1}$ and the output $\boldsymbol{z}_{t-1}^{(\mathtt{O})}$, $\phi(\cdot)$ is an activation function, and $\mathbf{W}^{(\mathtt{L})} \in \mathbb{R}^{\mathtt{A}\times\mathtt{Y}}$ and $\boldsymbol{b}^{(\mathtt{L})} \in \mathbb{R}^{\mathtt{Y}}$ are the weight matrix and bias vector for the last layer ($\mathtt{L}$).

There are two RNN's variants in which either one of the two recurrent connections is ab-

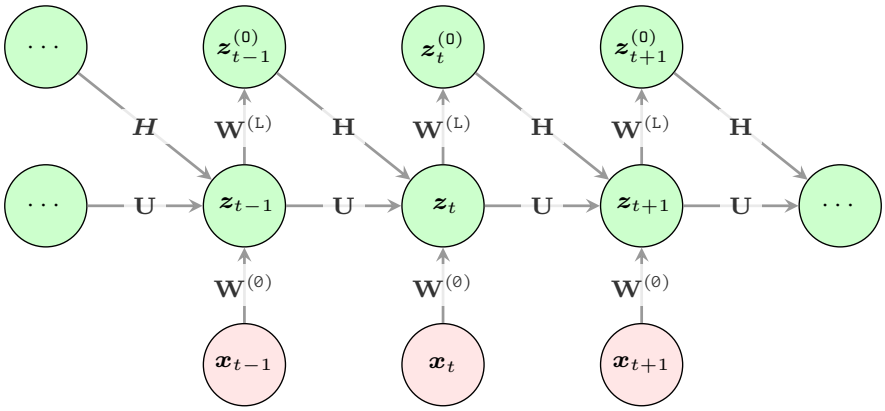


Figure 2.9 The time-unrolled representation of a single-hidden-layer RNN. The arrows depict the casual direction or casual relationship between variables as described in Equation 2.12.

sent [24]. The RNN's variant having only the output-to-hidden recurrent connection ($\boldsymbol{z}^{(\mathtt{O})}_{t-1}$ and $\boldsymbol{z}_t$) has a limited learning capacity because only the information from the output $\boldsymbol{z}^{(\mathtt{O})}_{t-1}$ is propagated forward to the following time steps [24]. The reason is that $\boldsymbol{z}^{(\mathtt{O})}_{t-1}$ is typically of low-dimension, e.g., one dimension for univariate time series, so that it cannot store much information. On the other hand, the RNN's variant having only the hidden-to-hidden recurrent connection ($\boldsymbol{z}_{t-1}$ and $\boldsymbol{z}_t$) can store more information because the hidden states $\boldsymbol{z}$ is a high-dimentional vector. However, this variant still lacks the direct connections with the past outputs. In practice, a RNN having both the hidden-to-hidden and output-to-hidden connections is more flexible and powerful than its two variants [24].

Figure 2.10 presents a one-hidden-layer RNN using the *teacher forcing* setup [24] in which the output-to-hidden recurrent connection uses the observations $\boldsymbol{y}_{t-1}$ to estimate the hidden states $\boldsymbol{z}_t$ instead of using the output hidden states $\boldsymbol{z}^{(\mathtt{O})}_{t-1}$ as it is the case for the RNN presented in Figure 2.9. In many cases, teacher forcing allows for a more stable and faster training [42], and it helps to avoid the accumulation of errors that may occur when the model relies on its own predictions [43]. However, if a model is trained with teacher forcing and is later used without it for prediction, there may be issues caused by the mismatch between how it was trained and used during forecasting.

Figure 2.11 presents a bidirectional RNN [23, 24] employing two separate RNN models, one moving forward (from $t = 1$ to $t = \mathtt{T}$), and another moving backward in time (from $t = \mathtt{T}$ to $t = 1$), where $\mathtt{T}$ refers to the last training time step. The hidden state vector $\boldsymbol{z}_t$ for the bidirectional RNN at the time step t is a concatenation of the hidden states from both the forward $\boldsymbol{z}^f_t$ and backward $\boldsymbol{z}^b_t$ models. The idea behind the bidirectional RNN is that the model needs to access both the past and future information in order to produce the

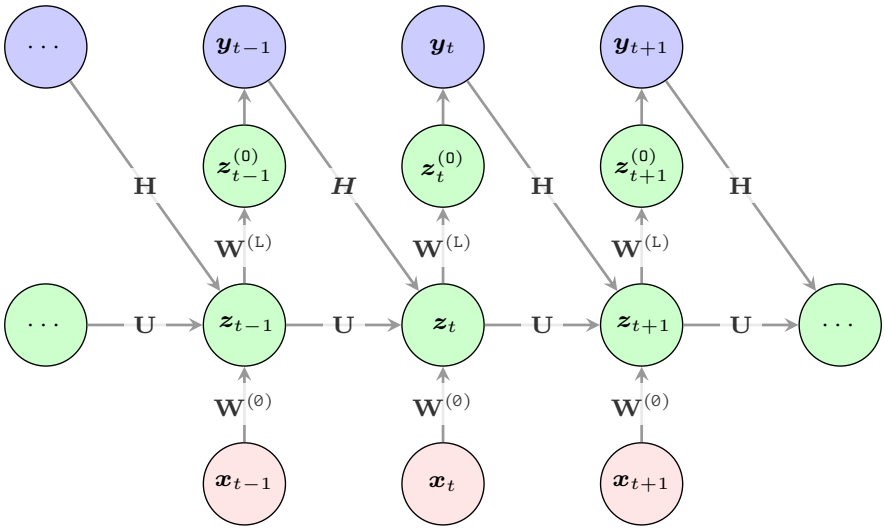


Figure 2.10 The teacher forcing setup for RNN where the output-to-hidden recurrent connection is defined between the observations $\boldsymbol{y}_{t-1}$ and the hidden states $\boldsymbol{z}_t$.

prediction at the current time t. This model is useful in cases such as natural language processing (NLP) where a model needs to understand the context of the whole sentence for providing predictions. There is a potential concern with bidirectional RNN called future information leakage, where information from the future indirectly influences the prediction at the current time step. The typical one-directional RNN only uses past information, i.e., the observations $\boldsymbol{y}_{1:t-1}$ up to the time step $t-1$ to predict the current observations $\boldsymbol{y}_t$ so that it is not affected by future information leakage.

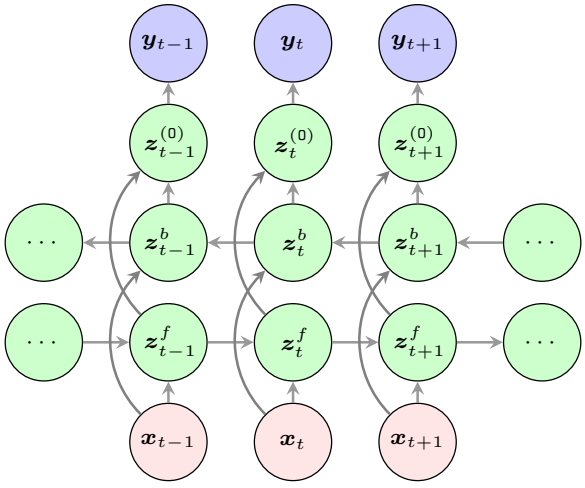


Figure 2.11 Time-unrolled representation of a bidirectional RNN where two separate RNN models are employed at the same time, one moves forward and the another moves backward in time.

The abovementioned RNN models are shallow in the sense that they have only one hidden layer. Deep RNN which consist in multiple hidden layers can be created in different ways as presented in [24, 44]. Stacked RNN [24] are a popular type of deep RNN where multiple recurrent layers are stacked on top of each other. Figure 2.12 presents the representation of a stacked RNN having $\mathtt{L}$ hidden layers.

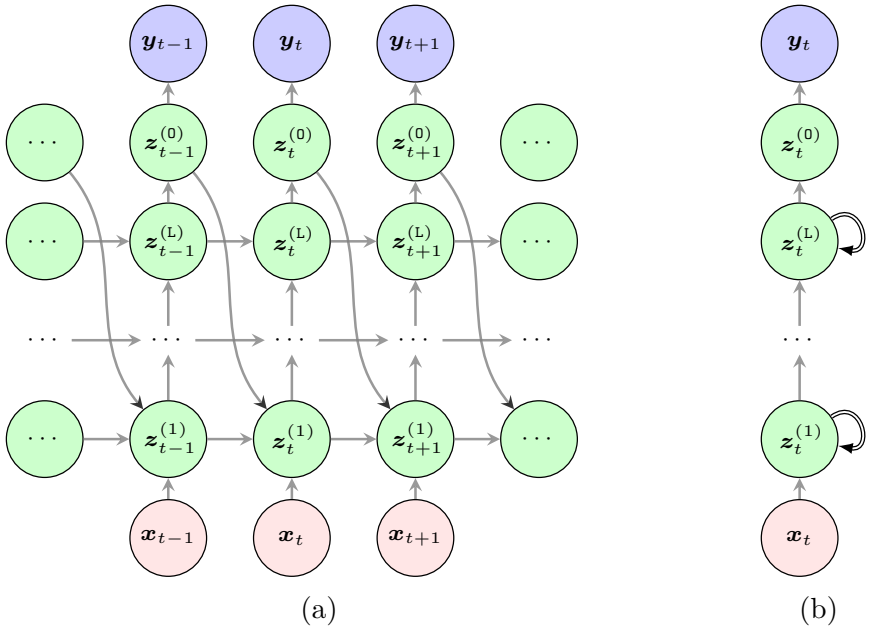


Figure 2.12 (a) Time-unrolled, and (b) compact representation of a stacked RNN where the double line arrow represents recurrent connections. Note that for the first hidden layer, the double arrow implies the both hidden-to-hidden and output-to-hidden recurrent connections, where for other hidden layers it implies only the hidden-to-hidden connection.

In practice, the RNN presented in this section experience issues with vanishing and exploding gradients [23, 24, 45] when learning dependencies over long sequences. The next sections review two RNN's architectures namely Long Short-Term Memory and Gated Recurrent Units neural networks which are designed to overcome this problem.

Long Short-Term Memory

Long Short-Term Memory (LSTM) neural networks [46] use a gating mechanism to automatically select the dependency information and store it in its 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, $\{\boldsymbol{f}, \boldsymbol{i}, \boldsymbol{o}, \tilde{\boldsymbol{c}}\} \in \mathbb{R}^{\mathtt{A}}$, the cell states $\boldsymbol{c} \in \mathbb{R}^{\mathtt{A}}$, and

the hidden states $\boldsymbol{h} \in \mathbb{R}^{\mathtt{A}}$, where $\mathtt{A}$ is the number of hidden units. Figure 2.13a presents a LSTM cell where all the operations in it can be summarized by the following LSTM recursive equations

$$\boldsymbol{f}_t = \sigma(\mathbf{W}^f \boldsymbol{x}_t + \mathbf{U}^f \boldsymbol{h}_{t-1} + \boldsymbol{b}^f), \tag{2.13a}$$

$$\boldsymbol{i}_t = \sigma(\mathbf{W}^i \boldsymbol{x}_t + \mathbf{U}^i \boldsymbol{h}_{t-1} + \boldsymbol{b}^i), \tag{2.13b}$$

$$\boldsymbol{o}_t = \sigma(\mathbf{W}^o \boldsymbol{x}_t + \mathbf{U}^o \boldsymbol{h}_{t-1} + \boldsymbol{b}^o), \tag{2.13c}$$

$$\tilde{\boldsymbol{c}}_t = \tanh(\mathbf{W}^c \boldsymbol{x}_t + \mathbf{U}^c \boldsymbol{h}_{t-1} + \boldsymbol{b}^c), \tag{2.13d}$$

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t, \tag{2.13e}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t), \tag{2.13f}$$

where $\boldsymbol{x}_t \in \mathbb{R}^{\mathtt{X}}$ is the input vector, $\{\mathbf{W}^f, \mathbf{W}^i, \mathbf{W}^o, \mathbf{W}^c\} \in \mathbb{R}^{\mathtt{X} \times \mathtt{A}}$ are the weight matrices for the input, $\{\mathbf{U}^f, \mathbf{U}^i, \mathbf{U}^o, \mathbf{U}^c\} \in \mathbb{R}^{\mathtt{A} \times \mathtt{A}}$ are the weight matrices for the hidden states, $\{\boldsymbol{b}^f, \boldsymbol{b}^i, \boldsymbol{b}^o, \boldsymbol{b}^c\} \in \mathbb{R}^{\mathtt{A}}$ are the bias vectors, $\sigma(\cdot)$ 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

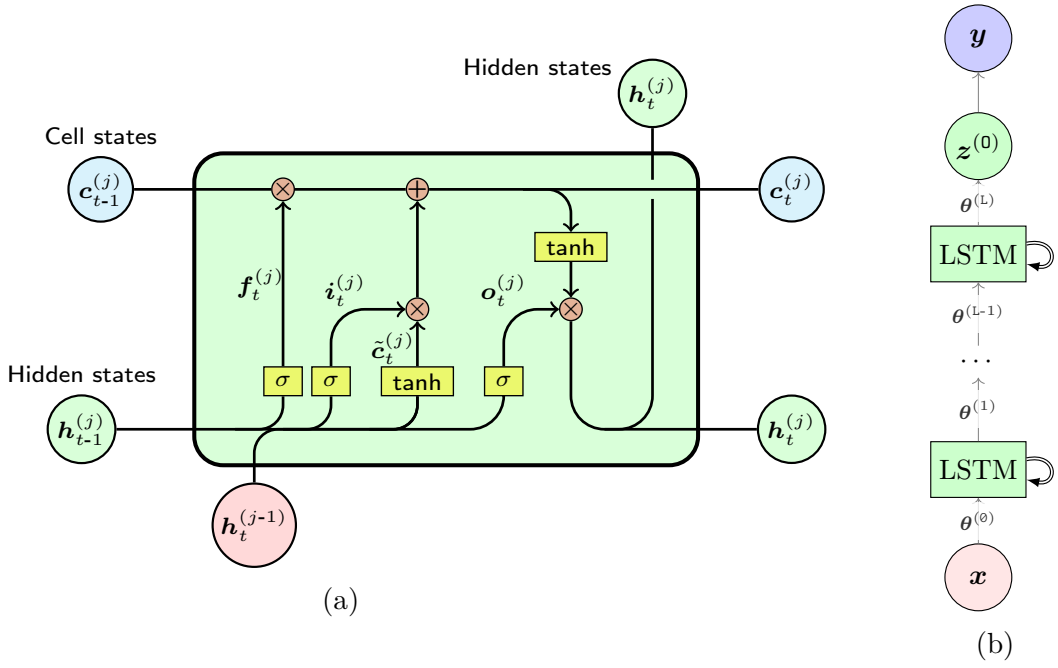


Figure 2.13 (a) The representation of a LSTM cell, and (b) the compact form of a stacked LSTM where single-line arrows present forward connections, and double-line arrows represent the recurrent connections.

inputs (Equations 2.13a-2.13d), and outputs the hidden states $\boldsymbol{h}_t$ (Equation 2.14d) which are then used to make the final predictions $\boldsymbol{y}_t$. 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 in two steps, and is done using Equation 2.13e. 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 forget gate $\boldsymbol{f}_t$ determines what information from the previous cell state $\boldsymbol{c}_{t-1}$ should be retained or forgotten. The second step consists in including the new information into the cell states $\boldsymbol{c}_t$ by adding the element-wise product $\boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t$. The input gate $\boldsymbol{i}_t$ decides what information from the candidate $\tilde{\boldsymbol{c}}_t$ should be added to the cell state $\boldsymbol{c}_t$. The output gate $\boldsymbol{o}_t$ controls what information from the current cell state $\boldsymbol{c}_t$ should be used to estimate the hidden states $\boldsymbol{h}_t$. After updating the cell states $\boldsymbol{c}_t$, the hidden states $\boldsymbol{h}_t$ are updated using Equation 2.14d.

Figure 2.13b presents a stacked LSTM having `L` hidden layers, each denoted by a rectangular node. In each LSTM layer, all the operations presented in Equation 2.13 are carried. The LSTM's hidden state vector $\boldsymbol{h}_t$ is equivalent to the traditional RNN one $\boldsymbol{z}_t$ presented in Figure 2.9 and Equation 2.12, but the process to estimate it is more complex. In addition, LSTM maintains an internal memory, i.e, the cell state $\boldsymbol{c}_t$, and both $\boldsymbol{h}_t$ and $\boldsymbol{c}_t$ are propagated to the next time step $t+1$, whereas only the hidden state $\boldsymbol{z}_t$ is propagated to $t+1$ in the traditional RNN.

Gated Recurrent Units

Gated Recurrent Units (GRU) neural network [47] is a simpler variant of LSTM. Compared to LSTM, GRU uses fewer gates and discard the internal memory $\boldsymbol{c}_t$. Figure 2.14a presents the GRU's cell where all operations within it are defined by the following equations

$$\boldsymbol{z}_t = \sigma(\mathbf{W}^z \boldsymbol{x}_t + \mathbf{U}^z \boldsymbol{h}_{t-1}), \tag{2.14a}$$

$$\boldsymbol{r}_t = \sigma(\mathbf{W}^r \boldsymbol{x}_t + \mathbf{U}^r \boldsymbol{h}_{t-1}), \tag{2.14b}$$

$$\tilde{\boldsymbol{h}}_t = \tanh\left[\mathbf{W}^h \boldsymbol{x}_t + \mathbf{U}^h(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1})\right], \tag{2.14c}$$

$$\boldsymbol{h}_t = (1 - \boldsymbol{z}_t) \odot \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \odot \tilde{\boldsymbol{h}}_t, \tag{2.14d}$$

where $\boldsymbol{x}_t$ is the input vector, $\{\mathbf{W}, \mathbf{U}\}$ are the weight matrices.

GRU employs the update $\boldsymbol{z}_t$ and reset $\boldsymbol{r}_t$ gates to control the information from the current input and the hidden states at the previous time steps. The reset gate decides what information from the hidden state $\boldsymbol{h}_{t-1}$ at the previous time step should be carried to the current time step. The GRU architecture combines the forget and input gates into a single update

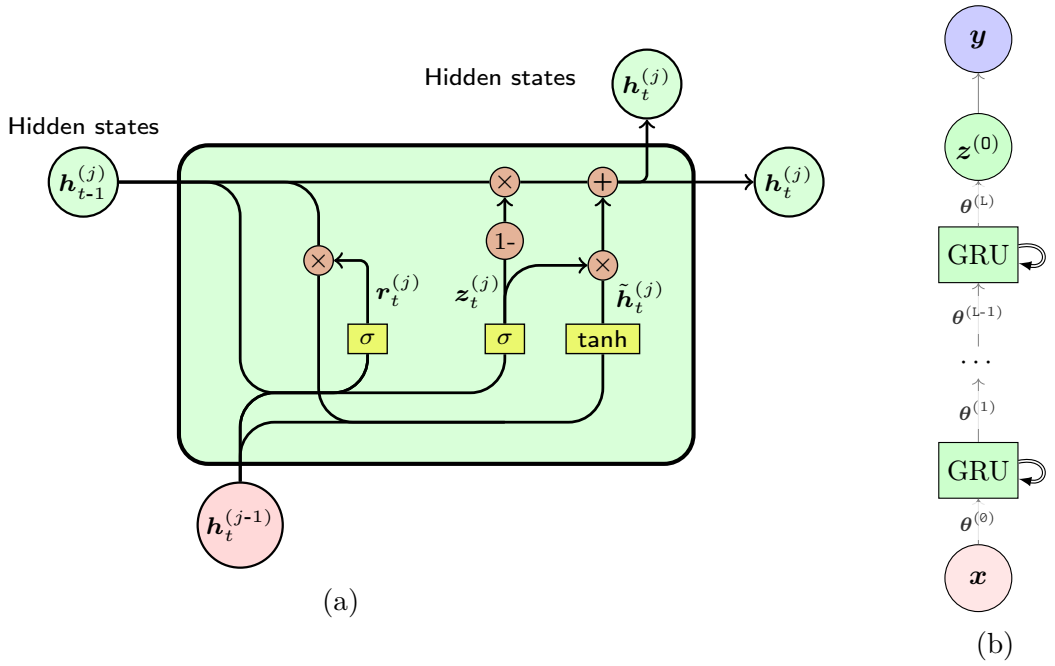


Figure 2.14 (a) GRU cell, and (b) compact representation of a stacked GRU where the double line arrow represents recurrent connections.

gate $\boldsymbol{z}_t$ that decides about the relative contribution of the candidate hidden states $\tilde{\boldsymbol{h}}_t$ and the hidden state $\boldsymbol{h}_{t-1}$ at the previous time step. Figure 2.14b presents a stacked GRU having $\mathtt{L}$ hidden layers, each denoted by a rectangular node. A comprehensive comparison between LSTM and GRU can be found in [47]. In practice, LSTM and GRU models have been shown to offer a similar performance [23]. Note that the teacher forcing and bidirectional setups for a RNN presented in this section can also be applied to LSTM and GRU.

Other architectures

Convolutional neural networks (CNN) [48] is an architecture designed to process grid-structured data such as images or time series. An image can be regarded as a 2D grid of pixels, while a time series can be considered a 1D sequence of data with each data point corresponding to a specific time step on a grid [23, 24]. For images, CNN uses 2D convolutions [24] that slide over the input data to learn and recognize features such as edges, corners, and textures, while for univariate time series, 1D convolutions are used to learn temporal patterns. Similarly to RNN, the original CNN architectures also had difficulties in learning long sequences in time series [49]. To address this problem, modifications to CNN have been proposed such as the WaveNet [49] which uses dilated convolutions for increasing the model's

receptive field, while reducing computational and memory requirements, and the Temporal Convolutional Network [50] which uses both dilated convolutions and residual connections. CNN is also combined with RNN-based models in various ways to take advantage of both methods as presented in [51–53].

Transformers [54] is an architecture originally developed for natural language processing (NLP) where its applications have been found in large language models such as ChatGPT. The key component of Transformers is the self-attention mechanism [54] which weighs the relative importance of input data to the current prediction, allowing to capture dependencies within data. Various studies have been proposed to adapt the original architecture for time series applications. To this end, modifications can be made in either one or all of the following steps: (1) data preprocessing, (2) positional embedding, (3) Encoder, and (4) Decoder [55]. Autoformer [56] has proposed using a trend-seasonality decomposition for processing the input sequence. The self-attention mechanism of Transformers ignores the sequential nature of data which is important in time series so that adding positional encoding to the input data is needed to give the model a sense the temporal order. For this reason, Informer [1] proposed using the timestamp embeddings, whereas Autoformer [56] introduced the learned positional embeddings. With the objective to reduce the $\mathcal{O}(L^2)$ time and memory complexity of the original Transformer's encoder layer, LogTrans [57], Pyraformer [58], Informer [1] have proposed different variants to the original self-attention mechanism to reduce the complexity, where L is the input sequence's length. Table 2.1 compares the complexity among several Transformer-based models as well as with the LSTM architecture. For the decoder layer, Informer [1] and Pyraformer [58] have proposed using the direct multi-step forecasting strategy in order to generate the forecast's sequence, instead of recursively generating one-step-ahead forecasts as in the original architecture, for improving the speed. Temporal Fusion Transformers (TFT) [59] have proposed modifications to all of these four steps, specifically, it combined sequence-to-sequence and attention-based temporal processing, variables selection, gating mechanisms, interpretable multi-head attention, and temporal self-attention decoder to achieve state-of-the-art performances on several time series forecasting tasks. Transformers have been shown to outperform RNN-based models for time series applications [1,57,59–61]. Similarly to the idea of bidirectional RNN, Devlin et al. [62] proposed a bidirectional approach to Transformer for solving a language understanding problem.

Although the CNN and Transformer-based methods presented in this section have been shown to achieve a good performance, the overwhelming predominance of RNN for time series modeling makes it the best suited choice as a first architecture to be developed for the Tractable Approximate Gaussian Inference method that will be presented in Section 2.3.3. Moreover, note that advanced architectures such as TFT themselves build upon a the LSTM architec-

ture. The following section reviews methods for training the neural network's architectures that have been presented in this section.

Table 2.1 Comparison of the complexity among Transformer-based models and the LSTM architecture. Reproduced from [1].

	Training		Testing
Method	Time	Memory	Steps
Informer	$\mathcal{O}(L \log L)$	$\mathcal{O}(L \log L))$	1
Transformer (original)	$\mathcal{O}(L^2)$	$\mathcal{O}(L^2)$	L
LogTrans	$\mathcal{O}(L \log L)$	$\mathcal{O}(L^2)$	1
LSTM	$\mathcal{O}(L)$	$\mathcal{O}(L)$	L

2.3.2 Training Neural Networks

In supervised learning problems, the objective of neural networks is to approximate a function $\boldsymbol{y} = g(\boldsymbol{x}; \boldsymbol{\theta})$, where $\boldsymbol{x}$ is the input vector, $\boldsymbol{y}$ is the observation vector, and $\boldsymbol{\theta} = \{\boldsymbol{w}, \boldsymbol{b}\}$ are the parameters which contain both the weights and biases. In a broad context, training a NN consists in adjusting $\boldsymbol{\theta}$ to minimize the difference between its predictions and the actual observations. For all the architectures presented in Section 2.3.1, the parameters $\boldsymbol{\theta}$ can be either modelled as deterministic values, or with distributions leading to Bayesian neural networks (BNN). This section reviews common algorithms used to train both deterministic and Bayesian NN.

Deterministic NN estimate a single value for each parameter θ_i, then use them to make point predictions $\hat{\boldsymbol{y}}$ so that the epistemic uncertainties associated with $\boldsymbol{\theta}$ are not taken into account. Note that deterministic NN can also be used to estimate predictive uncertainties, this will be covered later in this section. *Gradient descent* using *backpropagation* [63] is the most common method for training NN. The first step consists in defining an objective function as an average over a batch of `B` observations as

$$\mathrm{J}(\boldsymbol{\theta}) = \frac{1}{\mathtt{B}} \sum_{i=1}^{\mathtt{B}} \ell\left(g(\boldsymbol{x}_i; \boldsymbol{\theta}), \boldsymbol{y}_i\right),$$

where $\ell\left(g(\boldsymbol{x}_i; \boldsymbol{\theta}), \boldsymbol{y}_i\right)$ is the loss function evaluated for each observation $\boldsymbol{y}_i$. With point forecasts, the loss function takes a single value, and popular choices for regression problems are the Mean Squared Error (MSE) and Mean Absolute Error (MAE). NN are prone to overfitting where they perform well on a training data, but poorly on unseen test sets [24]. To

address this issue, a regularization term $\Omega(\boldsymbol{\theta})$ is typically added to the objective function such that

$$\mathrm{J}(\boldsymbol{\theta}) = \frac{1}{\mathtt{B}} \sum_{i=1}^{\mathtt{B}} \ell\left(g(\boldsymbol{x}_i; \boldsymbol{\theta}), \boldsymbol{y}_i\right) + \alpha\Omega(\boldsymbol{\theta}),$$

where $\alpha \in [0, \infty]$ is a hyperparameter that controls the contribution of $\Omega(\boldsymbol{\theta})$. L^2- and L^1-regularizations are the most common choices where $\Omega(\boldsymbol{\theta})$ is defined by $\Omega(\boldsymbol{\theta}) = \frac{1}{2}\boldsymbol{w}^\intercal\boldsymbol{w}$ and $\Omega(\boldsymbol{\theta}) = \sum_i |w_i|$, respectively, with $\boldsymbol{w}$ are the NN's weights. Beside this, other techniques such as early-stopping, dropout, and data augmentation are also used to prevent overfitting in NN [23,64]. The second step consists in using the gradient *backpropagation* to estimates the partial derivatiave $\nabla_{\boldsymbol{\theta}}\mathrm{J}(\boldsymbol{\theta})$ of the objective function $\mathrm{J}(\boldsymbol{\theta})$ with respect to the parameters $\boldsymbol{\theta}$. The last step involves using gradient-based optimization algorithms to learn the parameters $\boldsymbol{\theta}$.

Stochastic gradient descent (SGD) and its variants are the most common optimization algorithms used for training NN [24]. Consider a batch of $\mathtt{B}$ independent and identically distributed training data $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{\mathtt{B}} \subset \mathcal{D}$. The parameters $\boldsymbol{\theta}$ are adjusted for each batch by moving in the direction of the gradient so that

$$\boldsymbol{\theta} := \boldsymbol{\theta} - \epsilon\nabla_{\boldsymbol{\theta}}\mathrm{J}(\boldsymbol{\theta}),$$

where ϵ is the learning rate. When $\mathtt{B} = 1$, we have the *online optimization*, while using $\mathtt{B}$ equal to the total number of training instances results in the *complete data batch optimization* where both of these extreme cases are typically computationally expensive [24]. In order to take advantage of parallel computing, one typically chooses $\mathtt{B}$ between these two values to maximize the usage of computing resources available [65]. Momentum [66] and Nesterov momentum [67] are SGD's variants which use a momentum term that accumulates past gradients in order to modify the current parameter updates for accelerating the convergence. Adaptive learning rate methods such as Adaptive Gradient Algorithm [68] and Adam [69] allow to change the learning rate ϵ throughout training.

Unlike deterministic NN, BNN place a prior distribution over the parameters $f(\boldsymbol{\theta})$ so that they take into account the epistemic uncertainties. The goal of BNN is to estimate the parameter's posterior PDF $f(\boldsymbol{\theta}|\mathcal{D})$ using the Bayes' theorem as

$$f(\boldsymbol{\theta}|\mathcal{D}) = \frac{f(\mathcal{D}|\boldsymbol{\theta}) \cdot f(\boldsymbol{\theta})}{f(\mathcal{D})},$$

where $f(\mathcal{D}|\boldsymbol{\theta})$ is the likelihood, and $f(\mathcal{D}) = \int f(\mathcal{D}|\boldsymbol{\theta}) \cdot f(\boldsymbol{\theta}) d\boldsymbol{\theta}$ is the evidence. This posterior

can be factored in the predictions' uncertainties through the posterior predictive PDF where

$$f(\boldsymbol{y}|\mathcal{D}) = \int f(\boldsymbol{y}|\boldsymbol{\theta}) \cdot f(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta}, \tag{2.15}$$

so that the output of BNN are distributions rather than just point estimates as it is the case for deterministic NN. However, solving the integrals in Equation 2.15 and when calculating the evidence $f(\mathcal{D})$ is typically intractable for NN given the large number of parameters [24]. Therefore, methods such as Laplace approximation [70] and Variational inference (VI) [71–73] have been proposed to approximate the posterior $f(\boldsymbol{\theta}|\mathcal{D})$. The Laplace approximation assumes that the posterior $f(\boldsymbol{\theta}|\mathcal{D})$ can be represented as a multivariate Gaussian distribution $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\theta}^*, \mathbf{H}^{-1})$ where $\boldsymbol{\theta}^*$ are the MAP estimates optimized using gradient descent, and $\mathbf{H}$ is the Hessian of the negative loglikelihood $-\log(f(\mathcal{D}|\boldsymbol{\theta}^*))$ evaluated at $\boldsymbol{\theta}^*$. As the inverse of the full Hessian matrix is infeasible for large NN, diagonal covariance structures are typically used [74]. Variational inference (VI) approximates the intractable posterior $f(\boldsymbol{\theta}|\mathcal{D})$ by a known type of distribution $q_{\boldsymbol{\eta}}(\boldsymbol{\theta})$ with hyperparameters $\boldsymbol{\eta}$, $f(\boldsymbol{\theta}|\mathcal{D}) \approx q_{\boldsymbol{\eta}}(\boldsymbol{\theta})$. The task consists in learning $\boldsymbol{\eta}$ such that $q_{\boldsymbol{\eta}}(\boldsymbol{\theta})$ best approximate $f(\boldsymbol{\theta}|\mathcal{D})$. This can be done by minimizing the KL-divergence $d_{\mathrm{KL}}(q_{\boldsymbol{\eta}}(\boldsymbol{\theta})||f(\boldsymbol{\theta}|\mathcal{D}))$ which measures the dissimilarity between the two distributions. In practice, this quantity cannot be optimized directly because the posterior $f(\boldsymbol{\theta}|\mathcal{D})$ is unknown. Instead, we can maximize the evidence lower bound (ELBO) which can be obtained using a Monte Carlo approximation. VI then obtains the partial derivatives of the ELBO with respect to each parameter through backpropagation, and uses gradient descent to optimize $\boldsymbol{\eta}$. Training NN using VI is typically more computationally expensive than deterministic ones because for each batch of data, multiple passes are required in order to calculate the ELBO using Monte Carlo approximation. As the data becomes larger, only a limited number of passes are allowed leading to a limited performance [74].

Methods such as Monte Carlo dropout (MC-dropout) [75] and deep ensemble [76] use alternative approaches to BNN. These methods enable deterministic NN to provide predictive uncertainties with simple adjustments. Note that the parameters $\boldsymbol{\theta}$ are still considered as having deterministic values so that their epistemic uncertainties are not taken into account. Performing dropout in NN means randomly dropping a percentage of hidden units, i.e., setting their associated weights to zero, according to a Bernoulli distribution. When using dropout as a regularization technique, it is only used during training. MC-dropout [75] instead uses dropout during test time in order to provide a set of `N` point forecasts for each time step, where `N` is the number of test runs. The predictive distribution is estimated as the empirical one from these point forecasts. MC-dropout has been applied to RNN-based neural networks where dropout can either be only applied to the non-recurrent connections [77, 78],

to the LSTM cell states [79], or to all of the input, output and recurrent connections [75]. Deep ensemble [76] consists in combining the predictions from $\mathtt{N}$ independent deterministic NN with different initial parameters $\boldsymbol{\theta}_i$ where $i = 1 : \mathtt{N}$, along with a random shuffling of the training data. The predictions from these individual models are then used to make the final predictions and to generate the predictive uncertainties. While MC-dropout trains only a single NN, deep ensembles require training multiple ones so that it is more computationally demanding. Both of these methods are straightforward and practical because one can use the readily available deterministic NN without any significant modification, while still being able to obtain the predictive uncertainties. However, their limitation is that they do not account for the epistemic uncertainties associated the models's parameters.

Deka [13] has compared the performance between the BNN models reviewed above on small and large regression benchmarks [80]. As shown in Figure 2.15, most backpropagation-based methods were outperformed by the Tractable Approximate Gaussian Inference (TAGI) method that will be presented in the next section. Note that the PBP-MV method that was able able to reach a higher accuracy is over three order of magnitudes more computationally demanding.

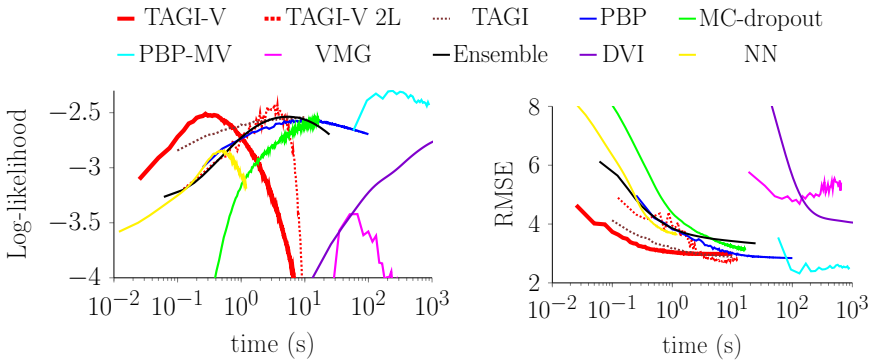


Figure 2.15 Comparison for the test log-likelihood and test RMSE for the Boston dataset among various Bayesian neural network methods. The figures are reused from [13].

2.3.3 Tractable Approximate Gaussian Inference

Tractable Approximate Gaussian Inference (TAGI) [12] is an analytically tractable framework for Bayesian neural networks (BNN). This section summarizes the principles behind TAGI through a fully connected feedforward neural network (FNN) architecture. Training a BNN using TAGI consists in two main steps: forward and backward. The forward step propagates the uncertainties from the input layer and network parameters up to the output layer. The

backward step employs a layer-wise recursive procedure to analytically infer the parameters $\boldsymbol{\theta}$ and the hidden states $\boldsymbol{Z}$ from the observations $\boldsymbol{y}$.

Recall the full representation of a multi-hidden-layer FNN presented in Figure 2.8a, Figure 2.16 displays its compact representation 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; $\boldsymbol{z}^{(\mathtt{0})}$ is the output vector; $\boldsymbol{y}$ is the observation vector; 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 $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{X}}, \boldsymbol{\Sigma}_{\boldsymbol{X}})$, $\boldsymbol{Z} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{Z}}, \boldsymbol{\Sigma}_{\boldsymbol{Z}})$, and $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$.

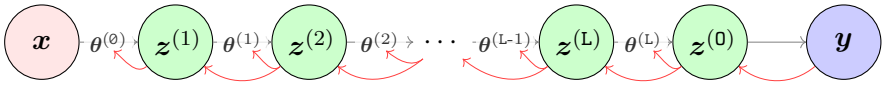


Figure 2.16 Graphical representation of a fully connected feedforward neural network. Black arrows represent the network forward connections; red arrows represent the inference directions for the hidden states and for the parameters.

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 an activation function. The forward step starts by passing information from the input layer to the first hidden layer using

$$\boldsymbol{Z}^{(1)} = \boldsymbol{W}^{(\mathtt{0})}\boldsymbol{X} + \boldsymbol{B}^{(\mathtt{0})}, \tag{2.16}$$

$$\boldsymbol{A}^{(1)} = \phi(\boldsymbol{Z}^{(1)}), \tag{2.17}$$

where $\boldsymbol{\theta}^{(\mathtt{0})} = \{\boldsymbol{W}^{(\mathtt{0})}, \boldsymbol{B}^{(\mathtt{0})}\}$ are the parameters for the input layer. Equation 2.16 involves two types of operations including additions and multiplications of Gaussian random variables. TAGI approximates the hidden states $\boldsymbol{Z}^{(1)}$ as Gaussians whose moments are calculated exactly. In order to achieve that, it relies on the *Gaussian multiplication approximation* (GMA) [12] to model the product of two Gaussian random variables by a Gaussian distribution whose exact mean and variance are calculated analytically as

$$\begin{aligned}
\mathbb{E}\left[X_1X_2\right] &= \mu_1\mu_2 + \mathrm{cov}(X_1, X_2), \\
\mathrm{cov}(X_3, X_1X_2) &= \mathrm{cov}(X_1, X_3)\mu_2 + \mathrm{cov}(X_2, X_3)\mu_1, \\
\mathrm{cov}(X_1X_2, X_3X_4) &= \mathrm{cov}(X_1, X_3)\,\mathrm{cov}(X_2, X_4) + \mathrm{cov}(X_1, X_4)\,\mathrm{cov}(X_2, X_3) \\
&+ \mathrm{cov}(X_1, X_3)\mu_2\mu_4 + \mathrm{cov}(X_1, X_4)\mu_2\mu_3 \\
&+ \mathrm{cov}(X_2, X_3)\mu_1\mu_4 + \mathrm{cov}(X_2, X_4)\mu_1\mu_3, \\
\mathrm{var}(X_1X_2) &= \sigma_1^2\sigma_2^2 + \mathrm{cov}(X_1, X_2)^2 + 2\,\mathrm{cov}(X_1, X_2)\mu_1\mu_2 + \sigma_1^2\mu_2^2 + \sigma_2^2\mu_1^2.
\end{aligned}$$

In addition, the output moments for the nonlinear activation function $\phi(\cdot)$ in Equation 2.17 cannot be evaluated analytically. TAGI uses the local linearization of activation function $\tilde{\phi}(\cdot)$ [12] to obtain the output moments following

$$\begin{aligned}
\boldsymbol{A}^{(j)} &= \phi(\boldsymbol{Z}^{(j)}) \approx \tilde{\phi}(\boldsymbol{Z}^{(j)}), \\
\boldsymbol{A}^{(j)} &= \mathbf{J}^{(j)}(\boldsymbol{Z}^{(j)} - \boldsymbol{\mu}_{\boldsymbol{Z}}^{(j)}) + \phi(\boldsymbol{\mu}_{\boldsymbol{Z}}^{(j)}), \\
\boldsymbol{\mu}_{\boldsymbol{A}}^{(j)} &\equiv \mathbb{E}[\boldsymbol{A}^{(j)}] = \tilde{\phi}(\boldsymbol{\mu}_{\boldsymbol{Z}}^{(j)}), \\
\boldsymbol{\Sigma}_{\boldsymbol{A}}^{(j)} &\equiv \text{cov}(\boldsymbol{A}^{(j)}) = \mathbf{J}^{(j)}\boldsymbol{\Sigma}_{\boldsymbol{Z}}^{(j)}\mathbf{J}^{(j)\intercal},
\end{aligned}$$

where $\boldsymbol{Z}^{(j)}$ and $\boldsymbol{A}^{(j)}$ are the hidden and activation units of the j^{th} hidden layer, and $\mathbf{J}^{(j)} = \text{diag}(\nabla_{\boldsymbol{Z}}\phi(\boldsymbol{\mu}_{\boldsymbol{Z}}^{(j)}))$ is the diagonal Jacobian matrix of the transformation evaluated at $\boldsymbol{\mu}_{\boldsymbol{Z}}^{(j)}$. Using these approximations allows calculating analytically the mean vector and covariance matrix defining the PDFs for the hidden $\boldsymbol{Z}^{(1)} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{Z}}^{(1)}, \boldsymbol{\Sigma}_{\boldsymbol{Z}}^{(1)})$ and activation units $\boldsymbol{A}^{(1)} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{A}}^{(1)}, \boldsymbol{\Sigma}_{\boldsymbol{A}}^{(1)})$. Similarly, going from a j^{th} layer to the subsequent $j+1^{th}$ is done using the same Equations 2.16 and 2.17, while replacing the input $\boldsymbol{X}$ by the activation units $\boldsymbol{A}^{(j)} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{A}}^{(j)}, \boldsymbol{\Sigma}_{\boldsymbol{A}}^{(j)})$, and $\boldsymbol{\theta}^{(\emptyset)}$ by the parameters $\boldsymbol{\theta}^{(j)} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}^{(j)}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{(j)})$ 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

$$\boldsymbol{y} = \boldsymbol{z}^{(0)} + \boldsymbol{v}, \quad \boldsymbol{v} : \boldsymbol{V} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{V}}), \tag{2.18}$$

where $\boldsymbol{v}$ denotes the error term. 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

$$\begin{aligned}
f(\boldsymbol{z}^{(0)} \mid \boldsymbol{y}) &= \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{Z}^{(0)}|\boldsymbol{y}}, \boldsymbol{\Sigma}_{\boldsymbol{Z}^{(0)}|\boldsymbol{y}}), \\
\boldsymbol{\mu}_{\boldsymbol{Z}^{(0)}|\boldsymbol{y}} &= \boldsymbol{\mu}_{\boldsymbol{Z}^{(0)}} + \boldsymbol{\Sigma}_{\boldsymbol{Y}\boldsymbol{Z}^{(0)}}^{\intercal}\boldsymbol{\Sigma}_{\boldsymbol{Y}}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}_{\boldsymbol{Y}}), \\
\boldsymbol{\Sigma}_{\boldsymbol{Z}^{(0)}|\boldsymbol{y}} &= \boldsymbol{\Sigma}_{\boldsymbol{Z}^{(0)}} - \boldsymbol{\Sigma}_{\boldsymbol{Y}\boldsymbol{Z}^{(0)}}^{\intercal}\boldsymbol{\Sigma}_{\boldsymbol{Y}}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{Y}\boldsymbol{Z}^{(0)}},
\end{aligned} \tag{2.19}$$

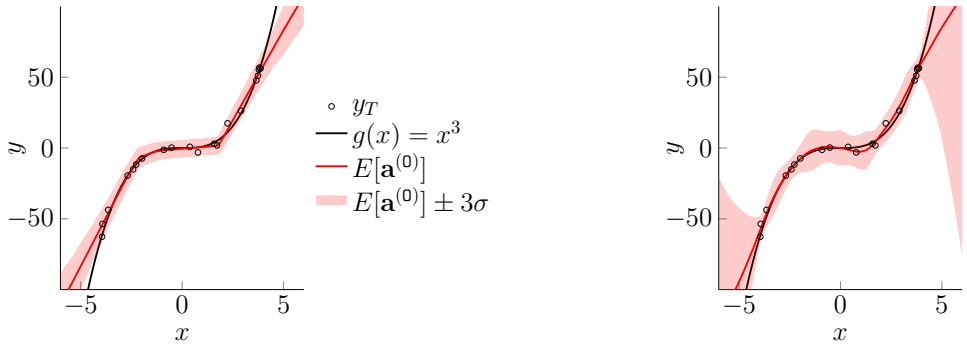
where $\boldsymbol{\Sigma}_{\boldsymbol{Y}\boldsymbol{Z}^{(0)}} = \boldsymbol{\Sigma}_{\boldsymbol{Z}^{(0)}}$. This is depicted by the red arrow from $\boldsymbol{y}$ to $\boldsymbol{z}^{(0)}$ in Figure 2.16. The layer-wise recursive inference procedure is then applied to infer the hidden states and parameters of each layer from the last to the first layer as presented by the red arrows in

Figure 2.16. 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., $\boldsymbol{Z}^{(j-1)} \perp\!\!\!\perp \boldsymbol{Z}^{(j+1)} \mid \boldsymbol{z}^{(j)}$. The posterior distribution for the hidden states of each layer $\boldsymbol{Z}^{(j)}$ is estimated analytically using

$$
\begin{aligned}
f(\boldsymbol{z}^{(j)}|\boldsymbol{y}) &= \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{Z}|\boldsymbol{y}}^{(j)}, \boldsymbol{\Sigma}_{\boldsymbol{Z}|\boldsymbol{y}}^{(j)}), \\
\boldsymbol{\mu}_{\boldsymbol{Z}|\boldsymbol{y}}^{(j)} &= \boldsymbol{\mu}_{\boldsymbol{Z}}^{(j)} + \mathbf{J}_{\boldsymbol{Z}}(\boldsymbol{\mu}_{\boldsymbol{Z}|\boldsymbol{y}}^{(j+1)} - \boldsymbol{\mu}_{\boldsymbol{Z}}^{(j+1)}), \\
\boldsymbol{\Sigma}_{\boldsymbol{Z}|\boldsymbol{y}}^{(j)} &= \boldsymbol{\Sigma}_{\boldsymbol{Z}}^{(j)} + \mathbf{J}_{\boldsymbol{Z}}(\boldsymbol{\Sigma}_{\boldsymbol{Z}|\boldsymbol{y}}^{(j+1)} - \boldsymbol{\Sigma}_{\boldsymbol{Z}}^{(j+1)})\mathbf{J}_{\boldsymbol{Z}}^{\intercal}, \\
\mathbf{J}_{\boldsymbol{Z}} &= \text{cov}(\boldsymbol{Z}^{(j)}, \boldsymbol{Z}^{(j+1)})(\boldsymbol{\Sigma}_{\boldsymbol{Z}}^{(j+1)})^{-1},
\end{aligned}
\tag{2.20}
$$

where $\text{cov}(\boldsymbol{Z}^{(j)}, \boldsymbol{Z}^{(j+1)})$ 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 Equation 2.20 where the variable $\boldsymbol{Z}^{(j)}$ is replaced by $\boldsymbol{\theta}^{(j)}$. The calculations for obtaining the covariances $\text{cov}(\boldsymbol{Z}^{(j)}, \boldsymbol{Z}^{(j+1)})$ and $\text{cov}(\boldsymbol{\theta}^{(j)}, \boldsymbol{Z}^{(j+1)})$, as well as other quantities in Equation 2.20 have been detailed in [12].

Notice that the use of diagonal covariance structures in TAGI for the parameters and hidden states leads to underestimating the predictive uncertainties compared to the actual ones obtained using Hamiltonian Monte-Carlo (HMC) [81] as presented in Figure 2.17. A weakly informative prior is employed when initializing the hyperparameters for the first epoch $\{\boldsymbol{\mu}_{\boldsymbol{\theta}}^{\emptyset}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{\emptyset}\}$ based on either He [82] or Xavier [83] approaches. In order to learn the parameters efficiently, TAGI repeats the inference over multiple epochs, analogously to the empirical Bayes approach [84], so that the posterior hyperparameters $\{\boldsymbol{\mu}_{\boldsymbol{\theta}|\boldsymbol{y}}^{e}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}|\boldsymbol{y}}^{e}\}$ at the e^{th} epoch are used as the prior hyperparameters for the next $e+1^{th}$ epoch.



(a) TAGI (b) HMC

Figure 2.17 Comparison between TAGI with diagonal covariance structures and Hamiltonian Monte-Carlo (HMC) for a 1D regression problem $y = x^3 + v$. Re-used from [12].

The performance of TAGI has been benchmarked on feedforward neural networks (FNN) [12], convolutional neural networks (CNN), generative adversarial networks (GAN) [85], and reinforcement learning (RL) [86]. TAGI can theoretically be used with any existing NN architecture to create the corresponding analytically tractable Bayesian one. However, this requires developing the specific mathematical formulations for each new architecture. The next section reviews several hybrid models that couple state-space models and neural networks.

2.4 Hybrid Models

This section reviews hybrid models that combine the state-space models (SSM) presented in Section 2.2 and recurrent neural networks (RNN) presented in Section 2.3.1. Coupling SSM and RNN allows to take advantage of both methods, but it is not a trivial task because their respective inference procedure relies on different mechanisms. SSM are probabilistic models which rely on Bayesian inference, whereas RNN typically optimizes their parameters using backpropagation [63] and gradient descent (GD). Existing hybrid models rely on a mix of these inference methods, i.e., the backpropagation for estimating the RNN's parameters and Bayesian inference for updating the SSM's hidden states. Some studies have established hybrid models by using RNN to either model nonlinear SSM's transition and/or observation equations, or define the SSM's parameters. In [87–89], RNN are used to model the nonlinear hidden states' transitions, allowing their models to be non-Markovian. However, the common limitation among methods using NN to parametrize the SSM's nonlinear dynamic models is that the marginal log-likelihood function is intractable so that they either rely on Monte Carlo (MC) methods [88] or variational inference [87, 90] to approximate this function in order to update their neural network parameters. In [90] and [4], the SSM transition and observation models are kept to be linear which allows performing exact inference for the SSM's latent variables. The DeepState method [4] 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, the deterministic exponential smoothing equations are used as a data processing tool for a global LSTM [91]. The level and seasonality components are estimated for each time series, and are used to normalize and deseasonalize data on-the-fly [91]. However, the use of the deterministic form of exponential smoothing leads to point estimates for the baseline level and trend components.

2.5 Anomaly Detection

This section reviews anomaly detection methods for time series with the purpose to identify suitable methods for structural health monitoring (SHM) applications. Although extensive reviews have been conducted [92–96], the types of anomalies as well as the detection methods used in other fields are not always well suited for SHM. For example, SHM focuses on monitoring the deterioration of structures over extended periods, spanning over years or decades, while the computer science and signal processing fields are primarily concerned with anomalies due to specific events typically occurring over a short time frame, but over large databases.

Detecting anomalies in the context of SHM presents multiple challenges. The time series data is typically affected by sensor noise, and contains repeated patterns caused by external effects such as ambient temperature, water level or loading. Therefore, anomaly detection should be conducted on the irreversible structural responses extracted from the raw data on which the noise and reversible external effects have been removed. In addition, data commonly contains a large number of missing values arising from sensor failures so that the methods employed must be capable of handling such a practical reality. Furthermore, SHM needs a tight control on the false alarm rate as evaluating structures after an alarm is expensive. For example, if one wants to process the data from 1000 sensors with a daily acquisition rate, even a single false alarm per sensor each year would lead to three false alarms a day. This aspect is currently limiting the widespread application of SHM where too frequent false alarms undermine its economic viability.

Anomaly detection methods can be categorized into supervised, semi-supervised, and unsupervised ones based on the type of data used [95]. Supervised methods use labeled data where the anomalies are known for training models to classify between normal and abnormal events. By contrast, unsupervised methods can operate directly on unlabeled data, while semi-supervised approaches only require labelled normal data for training, whereas the model trained can be used to distinguish between normal and abnormal for unseen test data. In SHM, labelled data with known anomalies is rarely available so that semi-supervised and unsupervised methods are best suited.

Anomaly detection methods can also be grouped into the forecasting/reconstruction and distanced-based categories [95]. The principle behind using forecasting methods for anomaly detection is that if an observation $y_t \in \mathbb{R}^1$ differs greatly from a model's prediction $\hat{y}_t$, $|y_t - \hat{y}_t| > \tau$, it should be categorized as an anomaly, where τ is a pre-defined threshold. Prophet [97] is a regression method which belongs to this forecasting family. It can decompose

a raw time series into three components including trend $g(t)$, seasonality $s(t)$ and holidays $h(t)$ using

$$y_t = g(t) + s(t) + h(t) + \epsilon_t, \tag{2.21}$$

where ϵ_t is an error term. The trend model $g(t)$ is used to describe the long-term non-periodic components of data. There are two choices for this functions for modeling a saturating growth and a growth that has a piece-wise constant rate. The seasonality component $s(t)$ uses the standard Fourier series to model periodic effects, whereas the holiday component $h(t)$ models the impact of holidays and special events. These components are defined by the model's parameters $\boldsymbol{\theta}$. The hyperparameters $\boldsymbol{\eta}$ that define the probability distributions for $\boldsymbol{\theta}$ are obtained as maximum a posteriori (MAP) estimates using optimization. Different sets of the parameters $\{\boldsymbol{\theta}^i\}_{i=1}^{\mathtt{N}}$ are sampled from the obtained parameters distributions to generate a set of forecasts $\{y_t^i\}_{i=1}^{\mathtt{N}}$, where $\mathtt{N}$ is the number of samples. The predictive mean $\mathbb{E}[y_t]$ and standard deviation σ_{y_t} are the empirical values obtained from the forecasts. For Prophet, the threshold τ is set based on the confidence intervals' probability context. We can note from Equation 2.21 that Prophet is only suited for univariate cases.

Similarly to the forecasting approaches, reconstruction methods build models for observed inputs $\boldsymbol{x}_t \in \mathbb{R}^{\mathtt{X}}$, and anomalies are identified by comparing the model's reconstruction $\hat{\boldsymbol{x}}_t$ and $\boldsymbol{x}_t$, where $\mathtt{X}$ is the input dimension. Note that these methods have no forecasting capacity. Autoencoders [98,99] belongs to the reconstruction family. The model consists of an encoder and a decoder

$$\boldsymbol{z}_t = \text{Encoder}(\boldsymbol{w}_t, \boldsymbol{\eta}), \quad \hat{\boldsymbol{w}}_t = \text{Decoder}(\boldsymbol{z}_t, \boldsymbol{\theta}), \tag{2.22}$$

where $\boldsymbol{z}_t$ is the latent variable vector, $\boldsymbol{\eta}$ and $\boldsymbol{\theta}$ are the parameters for the encoder and decoder, $\boldsymbol{w}_t = \{y_{t-h+1}, \cdots, y_t\}$ is the lookback window with a horizon $\mathtt{W}$, and $\hat{\boldsymbol{w}}_t = \{\hat{y}_{t-\mathtt{W}+1}, \cdots, \hat{y}_t\}$ contains the predictions for $\boldsymbol{w}_t$. The encoder compresses the input $\boldsymbol{w}_t$ into a lower dimensional latent space $\boldsymbol{z}_t$, whereas the decoder uses this latent vector to reconstruct the input $\hat{\boldsymbol{w}}_t$. An autoencoder model is trained using labeled normal data, then is tested on unlabeled data containing both normal and abnormal events so that this methods is a semi-supervised one. An input window $\boldsymbol{w}_t$ is considered as an anomaly if

$$e_t = ||\boldsymbol{w}_t - \hat{\boldsymbol{w}}_t||_2 > \tau,$$

where $||\cdot||_2$ represents the Euclidean distance, τ is a pre-defined threshold, and e_t is the reconstruction error. In this case, we present the standard Euclidean distance but any types of distance can be used. Note that Autoencoders are deterministic neural networks where the parameters $\boldsymbol{\eta}$ and $\boldsymbol{\theta}$ take deterministic values. Variational Autoencoders (VAE) [100] is

another neural network architecture from the reconstruction method's family. VAE is similar to Autoencoders in terms of the encoder-decoder architecture as presented in Equation 2.22, but VAE instead consider the latent variables $\boldsymbol{z}_t$ as random variables. The encoder estimates the distribution $q_{\boldsymbol{\eta}}(\boldsymbol{z}_t|\boldsymbol{w}_t)$, whereas the decoder computes the conditional likelihood $p_{\boldsymbol{\theta}}(\boldsymbol{w}_t|\boldsymbol{z}_t)$. A window $\boldsymbol{w}_t$ is considered as abnormal if

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{z}_t) < \tau,$$

where τ is a pre-defined threshold. Both Autoencoder and VAE are suited for multivariate cases as $\mathtt{X} \geq 1$.

Distanced-based anomaly detection methods identify abnormal events in data by measuring dissimilarities or distances between a single or a window of data points. k-nearest neighbours (kNN) [101,102] is a method that is originally developed for classification, but can also be used for unsupervised anomaly detection. Given a current lookback window $\boldsymbol{w}_t = \{y_{t-\mathtt{W}+1}, \cdots, y_t\}$, the Euclidean distances between all other windows can be calculated using $d_{ti} = ||\boldsymbol{w}_t - \boldsymbol{w}_i||_2$. The k-nearest neighbours to $\boldsymbol{w}_t$ are the k windows that have k smallest distances d_{ti}. A window $\boldsymbol{w}_t$ is considered as an anomaly if

$$\frac{1}{k}\sum_{i=1}^{k} d_{ti}^2 > \tau,$$

where τ is a predefined threshold. Matrix profile (MP) [103, 104] is another distance-based method originally used in data mining which is capable of detecting regime switches as well as patterned anomalies. For each window $\boldsymbol{w}_t$, MP calculates the distances d_{ti} between it and all other windows, but only stores the smallest distance with the nearest neighbour in order to form the MP vector. The locations of abnormal patterns are associated with the maxima in the MP vector, whereas the locations of regime switches are linked with the minima of the corrected arc curve constructed from from the MP vector.

Among the abovementioned methods, the Prophet, kNN, and MP are intended to be used in an offline setup, employing the entire time series either for training (Prophet) or for calculating the distances (kNN and MP). Theoretically, they can also be adapted for online analysis where these models would need to be re-fitted each time after new data point becomes available. This entails both a reduction in the performance and significantly increases the computational cost. By contrast, the Autoencoder and VAE can operate online after having learnt their parameters using the anomaly-free training set, making them more suited for SHM applications.

2.6 Conclusion

Bayesian Dynamic Linear Models (BDLM) and Switching Kalman Filter (SKF) are SSM-based models that have been used in structural health monitoring to decompose raw data about structural responses into interpretable components, as well as to detect anomalies, giving useful insights for engineers. However, their limitation is that they require intensive manual feature engineering for defining the models as presented in Section 2.2.2.

Recurrent neural networks (RNN), especially the Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), are the dominant neural network architectures used for time series analysis. Their advantages is that they can automatically identify complex patterns from data with a minimal setup. The Tractable Approximate Gaussian Inference (TAGI) method has been shown to outperform other gradient-based methods in regression tasks. However, TAGI has not been tested with the RNN architectures on time series data due to the lack of a mathematical framework to do so.

Coupling SSM and RNN for taking advantage of both methods is not a trivial task because SSM relies on Bayesian inference, whereas RNN parameters are typically optimized using backpropagation and gradient descent. The development of Bayesian RNN models using the TAGI framework will in turn allow the probabilistic coupling between RNN and SSM by using Bayesian inference as the single inference mechanism.

In terms of anomaly detection, the resulting hybrid model will then enhance the scalability of the SKF method presented in Section 2.2.3 while offering an online alternative to the existing methods reviewed.

CHAPTER 3 Analytically Tractable Bayesian Recurrent Neural Networks

3.1 Introduction

Recurrent neural networks (RNN) are predominant in time series analyses. Vanilla RNN experience exploding and vanishing gradient which limits their ability for modeling long sequences [24]. Long Short-Term Memory (LSTM) [46] and Gated Recurrent Unit (GRU) [47] neural networks are two advanced RNN-based architectures designed to overcome this limitation, and able to model both long and short term dependencies enabling to model long sequences. In their original forms, both LSTM and GRU are deterministic where they considering the model parameters as having fixed values and failing to take into account the epistemic uncertainties. Various probabilisitic LSTM and GRU models have been proposed by using methods such as MC-dropout [75] and Variational inference [71–73]. However, all of these models use gradient descent and backpropagation for inferring the neural network's parameters. This chapter proposes using Bayesian inference, specifically the Tractable Approximate Gaussian Inference (TAGI) method, with the LSTM and GRU architectures in order to infer their parameters analytically, without relying on gradient descent and backpropagation. Overall, we employ the same architecture formulations, but we model all quantities in the network as Gaussian random variables including the inputs, outputs, hidden units and parameters. Our method can take into account and quantify the epistemic uncertainties associated with the parameters, and propagate them through the network up to the output layer in order to provide not only predictions but also their associated uncertainties. The contributions of this chapter include

- Develop mathematical formulations for the analytically tractable Bayesian TAGI-LSTM, TAGI-GRU neural networks.
- Validate the models by comparing them with the deterministic and variational LSTM, GRU models trained with backpropagation on two time series benchmarks and one structural health monitoring dataset.

3.2 TAGI Long Short-Term Memory

This section introduces the mathematical formulations for the TAGI Long Short-Term Memory (TAGI-LSTM) neural network, a Bayesian approach to the LSTM architecture presented in Section 2.3.1, where the network parameters and hidden states are inferred analytically

using the TAGI method presented in Section 2.3.3. In order to apply the TAGI method to LSTM, we employ the same architecture formulations presented in Equation 2.13 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 in [12], that is considering a diagonal covariance structure for all LSTM's gates, hidden states, cell states, and parameters. Figure 3.4a shows a graphical representation of a LSTM cell, whereas Figure 3.4b presents the graph for an example of stacked TAGI-LSTM network having an input layer containing the covariates $\boldsymbol{x}$, $\mathtt{L}$ LSTM layers, and a fully connected output layer. The observation $\boldsymbol{y}$ is related to the output $\boldsymbol{z}^{(0)}$ through Equation 2.18.

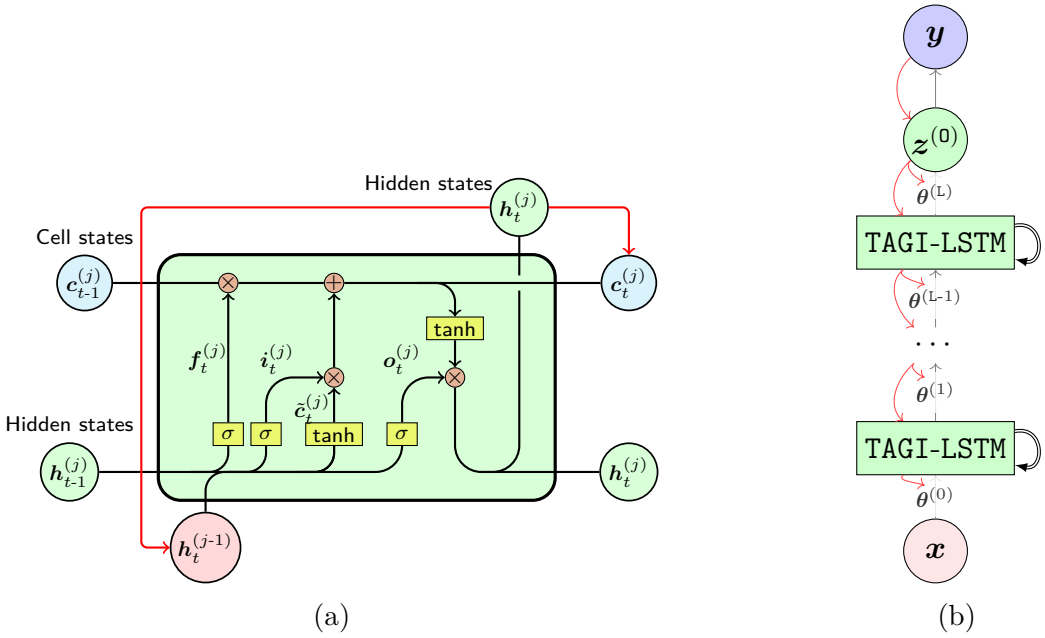


Figure 3.1 (a) Graphical representation of a LSTM cell. Red arrows represent the inference procedure to update the $j-1^{th}$ LSTM layer from the subsequent j^{th} LSTM layer. (b) Graphical representation of a stacked TAGI-LSTM network. Black arrows represent the network's forward connections, red arrows represent the layer-wise inference paths, and double arrows represent the recurrent connections.

3.2.1 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 $\boldsymbol{H}_{t|t-1} \sim \mathcal{N}(\boldsymbol{\mu}^{\boldsymbol{H}}_{t|t-1}, \boldsymbol{\Sigma}^{\boldsymbol{H}}_{t|t-1})$, where

$\boldsymbol{\mu}^{\boldsymbol{H}}_{t|t-1} \equiv \mathbb{E}[\boldsymbol{H}_t|\boldsymbol{y}_{1:t-1}]$, and $\boldsymbol{\Sigma}^{\boldsymbol{H}}_{t|t-1} \equiv \text{cov}(\boldsymbol{H}_t|\boldsymbol{y}_{1:t-1})$. 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, $\boldsymbol{H}^{(j)}_{t|t-1}$ and $\boldsymbol{C}^{(j)}_{t|t-1}$, as well as the prior $\boldsymbol{Z}^{(0)}_{t|t-1} \sim \mathcal{N}(\boldsymbol{\mu}^{\boldsymbol{Z}^{(0)}}_{t|t-1}, \boldsymbol{\Sigma}^{\boldsymbol{Z}^{(0)}}_{t|t-1})$ for the hidden states of the output layer. For the first epoch, the prior variances for the parameters $\boldsymbol{\Sigma}^{0}_{\boldsymbol{\theta}}$ are initialized based on He's approach [82], and the prior mean vector is randomly sampled from $\boldsymbol{\mu}^{0}_{\boldsymbol{\theta}} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}^{0}_{\boldsymbol{\theta}})$, while both prior means and variances for the hidden and cell states are initialized with zero-values.

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

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

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

$$Z^f_{i,t} \;=\; \boldsymbol{W}^f_{i,t}\boldsymbol{X}_t + \boldsymbol{U}^f_{i,t}\boldsymbol{H}_{t-1} + B^f_{i,t}, \tag{3.1}$$

where $i = 1 : \mathtt{N}$, $\boldsymbol{W}^f_{i,t} \in \mathbb{R}^{1\times \mathtt{M}}$, $\boldsymbol{U}^f_{i,t} \in \mathbb{R}^{1\times \mathtt{N}}$ are row weight matrices, $\mathtt{N}$ being the LSTM cell's size, and $\mathtt{M}$ being the number of input covariates. 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, $Z^f_{i,t}$ is the sum of $\mathtt{M}+\mathtt{N}$ independent products and a bias term as shown in Equation 3.1. When adding a large number of independent product terms, the 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 [12]. 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

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

where the mean and variance 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 Section 2.3.3.

From Equation 2.13a, 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 Section 2.3.3. 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 Equation 2.13e, 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 Equations 2.13a-2.13d 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

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

From Equation 2.14d, 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

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

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

$$Z_{i,t}^{(\mathtt{0})} \;=\; \boldsymbol{W}_{i,t}^{(\mathtt{L})} \boldsymbol{H}_t^{(\mathtt{L})} + B_{i,t}^{(\mathtt{L})}. \tag{3.2}$$

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

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

3.2.2 Backward step

The forward step presented in Section 3.2.1 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. 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 Figure 3.4b.

For the output layer, we compute the posterior $\boldsymbol{Z}^{(\mathtt{0})}_{t|t} \sim \mathcal{N}(\boldsymbol{\mu}^{\boldsymbol{Z}^{(\mathtt{0})}}_{t|t}, \boldsymbol{\Sigma}^{\boldsymbol{Z}^{(\mathtt{0})}}_{t|t})$ using the Gaussian conditional equations given in Equation 2.19. This is analogous to updating the output $\boldsymbol{Z}^{(\mathtt{0})}$ of a TAGI-FNN as presented in Section 2.3.3. 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

$$\begin{aligned} f(\boldsymbol{h}^{(j)}_{t|t}) &= \mathcal{N}(\boldsymbol{\mu}^{\boldsymbol{H}^{(j)}}_{t|t}, \boldsymbol{\Sigma}^{\boldsymbol{H}^{(j)}}_{t|t}),\\ \boldsymbol{\mu}^{\boldsymbol{H}^{(j)}}_{t|t} &= \boldsymbol{\mu}^{\boldsymbol{H}^{(j)}}_{t|t-1} + \mathbf{J}_{\boldsymbol{H}}(\boldsymbol{\mu}^{\boldsymbol{H}^{(j+1)}}_{t|t} - \boldsymbol{\mu}^{\boldsymbol{H}^{(j+1)}}_{t|t-1}),\\ \boldsymbol{\Sigma}^{\boldsymbol{H}^{(j)}}_{t|t} &= \boldsymbol{\Sigma}^{\boldsymbol{H}^{(j)}}_{t|t-1} + \mathbf{J}_{\boldsymbol{H}}(\boldsymbol{\Sigma}^{\boldsymbol{H}^{(j+1)}}_{t|t} - \boldsymbol{\Sigma}^{\boldsymbol{H}^{(j+1)}}_{t|t-1})\mathbf{J}^{\intercal}_{\boldsymbol{H}},\\ \mathbf{J}_{\boldsymbol{H}} &= \text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{H}^{(j+1)}_{t|t-1})(\boldsymbol{\Sigma}^{\boldsymbol{H}^{(j+1)}}_{t|t-1})^{-1}, \end{aligned} \tag{3.3}$$

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 Figure 3.4a and Equations 2.13a-2.13d. For obtaining $\boldsymbol{H}^{(j)}_{t|t}$, we directly calculate the covariance $\text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{H}^{(j+1)}_{t|t-1})$, bypassing the LSTM gates as shown by the red arrow from $\boldsymbol{h}^{(j)}_t$ to $\boldsymbol{h}^{(j-1)}_t$ shown in Figure 3.2b. This is because inferring $\boldsymbol{H}^{(j)}_{t|t}$ directly through gates or indirectly from $\boldsymbol{H}^{(j+1)}_{t|t-1}$ while bypassing gates leads to the same result, with the latter being simpler. This bypassing procedure during inference is further investigated in Section 3.4. 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



Figure 3.2 Infer TAGI-LSTM's hidden states and parameters of a layer j^{th} from hidden states of layer $j+1^{th}$ (a) without bypassing LSTM's gates , and (b) bypassing LSTM's gates.

can be estimated using

$$\begin{aligned} f(\boldsymbol{\theta}_{t|t}^{(j)}) &= \mathcal{N}(\boldsymbol{\mu}_{t|t}^{\boldsymbol{\theta}^{(j)}}, \boldsymbol{\Sigma}_{t|t}^{\boldsymbol{\theta}^{(j)}}), \\ \boldsymbol{\mu}_{t|t}^{\boldsymbol{\theta}^{(j)}} &= \boldsymbol{\mu}_{t|t-1}^{\boldsymbol{\theta}^{(j)}} + \mathbf{J}_{\boldsymbol{\theta}}(\boldsymbol{\mu}_{t|t}^{\boldsymbol{H}^{(j+1)}} - \boldsymbol{\mu}_{t|t-1}^{\boldsymbol{H}^{(j+1)}}), \\ \boldsymbol{\Sigma}_{t|t}^{\boldsymbol{\theta}^{(j)}} &= \boldsymbol{\Sigma}_{t|t-1}^{\boldsymbol{\theta}^{(j)}} + \mathbf{J}_{\boldsymbol{\theta}}(\boldsymbol{\Sigma}_{t|t}^{\boldsymbol{H}^{(j+1)}} - \boldsymbol{\Sigma}_{t|t-1}^{\boldsymbol{H}^{(j+1)}})\mathbf{J}_{\boldsymbol{\theta}}^{\intercal}, \\ \mathbf{J}_{\boldsymbol{\theta}} &= \text{cov}(\boldsymbol{\theta}_{t|t-1}^{(j)}, \boldsymbol{H}_{t|t-1}^{(j+1)})(\boldsymbol{\Sigma}_{t|t-1}^{\boldsymbol{H}^{(j+1)}})^{-1}. \end{aligned} \tag{3.4}$$

The detailed formulations for the covariances $\text{cov}(\boldsymbol{H}_{t|t-1}^{(j)}, \boldsymbol{H}_{t|t-1}^{(j+1)})$ and $\text{cov}(\boldsymbol{\theta}_{t|t-1}^{(j)}, \boldsymbol{H}_{t|t-1}^{(j+1)})$ which are required to apply Equations 3.3 and 3.4 are given in the Appendix A.1. Note that the last LSTM layer ($\mathtt{L}$) is connected with the output layer so that we need to calculate the covariances $\text{cov}(\boldsymbol{H}_{t|t-1}^{(\mathtt{L})}, \boldsymbol{Z}_{t|t-1}^{(\mathtt{O})})$ and $\text{cov}(\boldsymbol{\theta}_{t|t-1}^{(\mathtt{L})}, \boldsymbol{Z}_{t|t-1}^{(\mathtt{O})})$ between its hidden states and parameters with the output layer instead of with the next LSTM layer. These covariances have already been detailed in [12].

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 $\boldsymbol{C}_{t|t}^{(j)} \sim \mathcal{N}(\boldsymbol{\mu}_{t|t}^{\boldsymbol{C}^{(j)}}, \boldsymbol{\Sigma}_{t|t}^{\boldsymbol{C}^{(j)}})$ for the cell states based on the knowledge from the hidden states of a same LSTM layer using the Rauch-Tung-Striebel (RTS) procedure [105] so that

$$\begin{aligned} f(\boldsymbol{c}_{t|t}^{(j)}) &= \mathcal{N}(\boldsymbol{\mu}_{t|t}^{\boldsymbol{C}^{(j)}}, \boldsymbol{\Sigma}_{t|t}^{\boldsymbol{C}^{(j)}}), \\ \boldsymbol{\mu}_{t|t}^{\boldsymbol{C}^{(j)}} &= \boldsymbol{\mu}_{t|t-1}^{\boldsymbol{C}^{(j)}} + \mathbf{J}_{\boldsymbol{CH}}(\boldsymbol{\mu}_{t|t}^{\boldsymbol{H}^{(j)}} - \boldsymbol{\mu}_{t|t-1}^{\boldsymbol{H}^{(j)}}), \\ \boldsymbol{\Sigma}_{t|t}^{\boldsymbol{C}^{(j)}} &= \boldsymbol{\Sigma}_{t|t-1}^{\boldsymbol{C}^{(j)}} + \mathbf{J}_{\boldsymbol{CH}}(\boldsymbol{\Sigma}_{t|t}^{\boldsymbol{H}^{(j)}} - \boldsymbol{\Sigma}_{t|t-1}^{\boldsymbol{H}^{(j)}})\mathbf{J}_{\boldsymbol{H}}^{\intercal}, \\ \mathbf{J}_{\boldsymbol{CH}} &= \text{cov}(\boldsymbol{C}_{t|t-1}^{(j)}, \boldsymbol{H}_{t|t-1}^{(j)})(\boldsymbol{\Sigma}_{t|t-1}^{\boldsymbol{H}^{(j)}})^{-1}. \end{aligned} \tag{3.5}$$

This step is depicted by the red arrow from $\boldsymbol{h}_t^{(j)}$ to $\boldsymbol{c}_t^{(j)}$ as shown in Figure 3.4a. The diagonal cross-covariance matrix $\text{cov}(\boldsymbol{C}_{t|t-1}^{(j)}, \boldsymbol{H}_{t|t-1}^{(j)})$ between the cell and hidden states of the same LSTM layer that is required for estimating $\boldsymbol{C}_{t|t}^{(j)}$ is obtained following

$$\text{cov}(C_{i,t|t-1}^{(j)}, H_{i,t|t-1}^{(j)}) \quad = \quad \text{var}(C_{i,t|t-1}^{(j)}) \cdot \nabla_C \tilde{\tanh}(\mathbb{E}[C_{i,t|t-1}^{(j)}]) \cdot \mathbb{E}[O_{i,t|t-1}^{(j)}],$$

where $\nabla_C \tilde{\tanh}(\mathbb{E}[C_{i,t|t-1}^{(j)}])$ is the gradient of the $\tilde{\tanh}(C)$ function with respect to C evaluated at the expected value $\mathbb{E}[C_{i,t|t-1}^{(j)}]$ for the j^{th} layer, cell i, at time t. Note that when data is missing at a time step, one simply proceeds with the forward step presented in Section 3.2.1 without performing the backward step described above.

3.2.3 Smoothing for TAGI-LSTM

In SSMs, the transition model, $\boldsymbol{z}_t = g(\boldsymbol{z}_{t-1}) + \boldsymbol{w}_t$, describes the relationship between the hidden variables $\boldsymbol{z}$ at two consecutive time steps where $g(\cdot)$ 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 $t = \mathtt{T}$. The Kalman smoother [105] 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 Equations 2.13a-2.13e where the hidden and cell states at time step $t-1$ are connected to those at t such that

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

where the functions $k(\cdot)$ and $q(\cdot)$ are defined by Equations 2.13a-2.13e. Therefore, we can leverage these connections in order to perform smoothing for TAGI-LSTM. The backward step in Section 3.2.2 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 Figure 3.3.

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

$$\begin{aligned} \boldsymbol{\mu}_{t|\mathtt{T}}^{\boldsymbol{H}^{(j)}} &= \boldsymbol{\mu}_{t|t}^{\boldsymbol{H}^{(j)}} + \mathbf{J}_{\boldsymbol{H}}(\boldsymbol{\mu}_{t+1|\mathtt{T}}^{\boldsymbol{H}^{(j)}} - \boldsymbol{\mu}_{t+1|t}^{\boldsymbol{H}^{(j)}}) \\ \boldsymbol{\Sigma}_{t|\mathtt{T}}^{\boldsymbol{H}^{(j)}} &= \boldsymbol{\Sigma}_{t|t}^{\boldsymbol{H}^{(j)}} + \mathbf{J}_{\boldsymbol{H}}(\boldsymbol{\Sigma}_{t+1|\mathtt{T}}^{\boldsymbol{H}^{(j)}} - \boldsymbol{\Sigma}_{t+1|t}^{\boldsymbol{H}^{(j)}})\mathbf{J}_{\boldsymbol{H}}^{\intercal} \\ \mathbf{J}_{\boldsymbol{H}} &= \mathrm{cov}(\boldsymbol{H}_{t|t}^{(j)}, \boldsymbol{H}_{t+1|t}^{(j)})(\boldsymbol{\Sigma}_{t+1|t}^{\boldsymbol{H}^{(j)}})^{-1}, \end{aligned} \tag{3.6}$$

where $\mathrm{cov}(\boldsymbol{H}_{t|t}^{(j)}, \boldsymbol{H}_{t+1|t}^{(j)})$ is the covariance between the hidden states of the j^{th} LSTM layer at time t and $t+1$. We can also use Equation 3.6 to obtain the smoothed estimates $\boldsymbol{C}_{t|\mathtt{T}}^{(j)}$

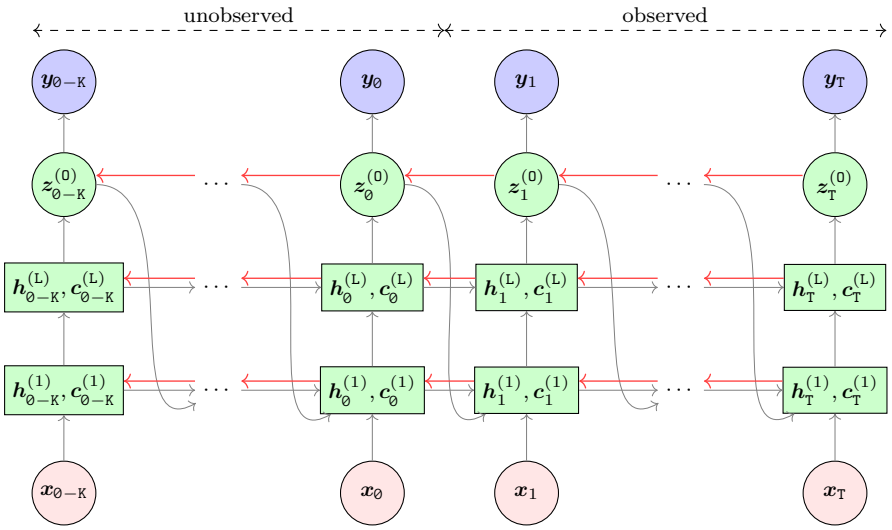


Figure 3.3 Time-unfolded representation of a TAGI-LSTM network with explicit connections between times steps. Black arrows represent the network forward connections, red arrows represent the smoothing procedure. The observed data includes training data, whereas the unobserved one is data K time steps 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.

for the cell states, and $\boldsymbol{Z}^{(0)}_{t|T}$ for the hidden states of the output layer, by replacing the hidden states $\boldsymbol{H}$ by the relevant variable. Appendix A.2 presents the derivations of the covariances $\text{cov}(\boldsymbol{H}^{(j)}_{t|t}, \boldsymbol{H}^{(j)}_{t+1|t})$, $\text{cov}(\boldsymbol{C}^{(j)}_{t|t}, \boldsymbol{C}^{(j)}_{t+1|t})$, and $\text{cov}(\boldsymbol{Z}^{(0)}_{t|t}, \boldsymbol{Z}^{(0)}_{t+1|t})$ which are required to estimate $\boldsymbol{H}^{(j)}_{t|T}, \boldsymbol{C}^{(j)}_{t|T}$ and $\boldsymbol{Z}^{(0)}_{t|T}$. 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 $t = 0$ to infer the initial hidden and cell states, $\boldsymbol{H}_{0|T}$ and $\boldsymbol{C}_{0|T}$, 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 Figure 3.3. We can then use them to estimate the unobserved past observations $\boldsymbol{y}_{0-K:0}$. 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.

3.3 TAGI Gated Recurrent Units

This section introduces the mathematical formulations for the TAGI Gated Recurrent Units (TAGI-GRU) neural network, a Bayesian approach to the GRU architecture presented in Section 2.3.1. We consider the parameters $\boldsymbol{\theta}$, GRU's gates and hidden states as Gaussian

random variables having a diagonal covariance structure. Figure 3.4 presents the graph for an example of stacked TAGI-GRU network having an input layer containing the covariates x, L GRU layers, and a fully connected output layer.

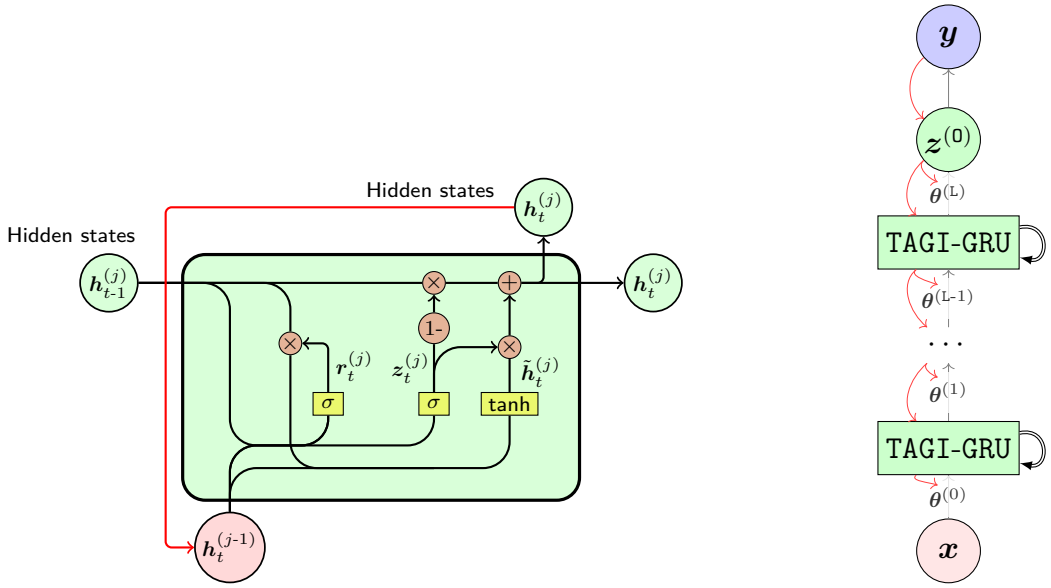


Figure 3.4 (a) Graphical representation of a GRU cell. Red arrows represent the inference procedure to update the $j-1^{th}$ GRU layer from the subsequent j^{th} GRU layer. (b) Graphical representation of a stacked TAGI-GRU network. Black arrows represent the network's forward connections, red arrows represent the layer-wise inference paths, and double arrows represent the recurrent connections.

Analogously to TAGI-LSTM, the TAGI-GRU's forward step aim at estimating the prior knowledge for the gates and hidden states of each GRU layer as well as the hidden state for the output layer. Because the equations used to calculate the GRU's gates $\{\boldsymbol{Z}, \boldsymbol{R}, \tilde{\boldsymbol{H}}\}$ are similar to those of LSTM, we can make the same independence assumption for these gates such that $Z_{i,t} \perp\!\!\!\perp Z_{j,t}$, $\tilde{H}_{i,t} \perp\!\!\!\perp \tilde{H}_{j,t}$, $R_{i,t} \perp\!\!\!\perp R_{j,t}$, and $\boldsymbol{Z}_t \perp\!\!\!\perp \tilde{\boldsymbol{H}}_t \perp\!\!\!\perp \boldsymbol{R}_t$. We define the hidden states for the reset gate by

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

The first two moments for its units are computed from

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

where the mean and variance of the product terms $\boldsymbol{W}_{i,t|t-1}^r \boldsymbol{X}_{t|t-1}$ and $\boldsymbol{U}_{i,t|t-1}^r \boldsymbol{H}_{t-1|t-1}$ are

calculated exactly using the GMA equations given in Section 2.3.3.

The reset gate is then estimated by $\boldsymbol{R}_t = \tilde{\sigma}(\boldsymbol{Z}_t^r)$, where $\tilde{\sigma}(\cdot)$ is the locally linearized sigmoid activation function, and the equations for obtaining the output's mean vector and covariance matrix are presented in Section 2.3.3.

Let $\boldsymbol{C}_t = \boldsymbol{R}_t \odot \boldsymbol{H}_{t-1}$ so that the candidate gate becomes

$$\tilde{\boldsymbol{H}}_t = \tanh(\mathbf{W}^h \boldsymbol{X}_t + \mathbf{U}^h \boldsymbol{C}_t).$$

The mean vectors and covariance matrices for the update $\boldsymbol{Z}_t$ and candidate gates $\tilde{\boldsymbol{H}}$ can be estimated using the same procedure.

The equation to calculate the hidden states $\boldsymbol{H}_t$ can be rewritten as

$$\begin{aligned} \boldsymbol{H}_t &= (1 - \boldsymbol{Z}_t) \odot \boldsymbol{H}_{t-1} + \boldsymbol{Z}_t \odot \tilde{\boldsymbol{H}}_t \\ &= \boldsymbol{H}_{t-1} - \boldsymbol{Z}_t \odot \boldsymbol{H}_{t-1} + \boldsymbol{Z}_t \odot \tilde{\boldsymbol{H}}_t, \end{aligned}$$

where the mean and variance for its unit are obtained from

$$\begin{aligned} \mathbb{E}[H_{i,t|t-1}] &= \mathbb{E}[H_{i,t-1|t-1}] - \mathbb{E}[Z_{i,t|t-1} \cdot H_{t-1|t-1}] + \mathbb{E}[Z_{i,t|t-1} \cdot \tilde{H}_{i,t|t-1}], \\ \text{var}(H_{i,t|t-1}) &= \text{var}(H_{i,t-1|t-1}) + \text{var}(Z_{i,t|t-1} \cdot H_{t-1|t-1}) + \text{var}(Z_{i,t|t-1} \cdot \tilde{H}_{i,t|t-1}) \\ &- 2\text{cov}(H_{i,t-1|t-1}, Z_{i,t|t-1} \cdot H_{i,t-1|t-1}) + 2\text{cov}(H_{i,t-1|t-1}, Z_{i,t|t-1} \cdot \tilde{H}_{i,t|t-1}) \\ &- 2\text{cov}(Z_{i,t|t-1} \cdot H_{i,t-1|t-1}, Z_{i,t|t-1} \cdot \tilde{H}_{i,t|t-1}). \end{aligned} \tag{3.7}$$

The detailed derivation of Equation 3.7 is provided in Appendix B.1.

In the backward step, we follow the same procedure as presented in Section 3.2.2 for the TAGI-LSTM. That is, using Equation 2.19 to estimate the posterior knowledge for the hidden states of the output layer, and Equations 3.3 and 3.4 to obtain the posterior PDF for the hidden states and parameters of each GRU layer. The difference is that the covariance $\text{cov}(\boldsymbol{H}_{t|t-1}^{(j)}, \boldsymbol{H}_{t|t-1}^{(j+1)})$ and $\text{cov}(\boldsymbol{\theta}_{t|t-1}^{(j)}, \boldsymbol{H}_{t|t-1}^{(j+1)})$ need to be estimated for the GRU, and are provided in Appendix B.2.

3.4 Bypassing Inference in TAGI

In Section 3.2.2 for TAGI-LSTM and Section 3.3 for TAGI-GRU, the backward pass infers directly the posterior knowledge for the hidden states $\boldsymbol{H}_{t|t}^{(j)}$ and parameters $\boldsymbol{\theta}_{t|t}^{(j)}$ of the j^{th} layer from the hidden states $\boldsymbol{H}_t^{(j+1)}$ of the subsequent layer, bypassing the LSTM or GRU's gates. Through an example with a feedforward neural network (FNN), this section shows

that the inference with and without bypassing hidden layers leads to the same results.

Figure 3.5a presents a FNN analogous to the one presented in Section 2.3.3 where the red arrows represent the layer-wise inference procedure. Note that this procedure infers the posterior knowledge $f(\boldsymbol{z}^{(j)}|\boldsymbol{y})$ for the hidden states $\boldsymbol{Z}^{(j)}$ of a layer j^{th} from the hidden states $\boldsymbol{Z}^{(j+1)}$ of the subsequent $j+1^{th}$ layer which bypasses the activation units $\boldsymbol{A}^{(j)}$ as explicitly shown in Figure 3.5b.

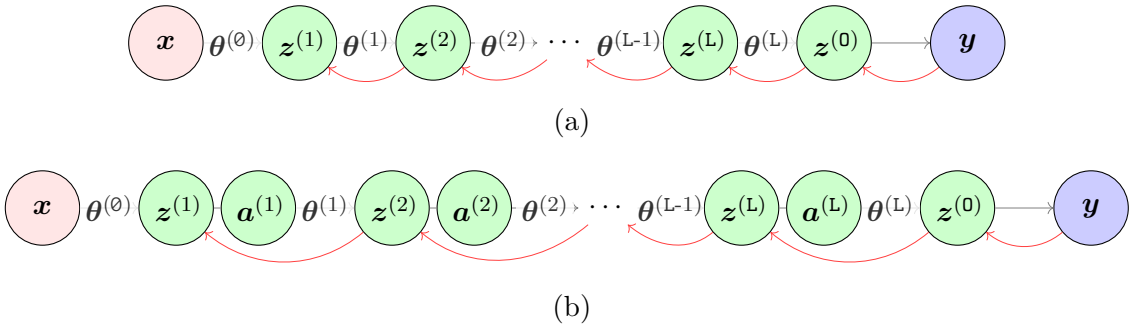


Figure 3.5 Layer-wise inference procedure to update the hidden states for a feedforward neural network (a) without, and (b) with activation units shown explicitly. Black arrows represent the network forward connections, red arrows represent the layer-wise inference paths.

The activation and hidden units $\boldsymbol{Z}^{(j+1)}$ and $\boldsymbol{A}^{(j)}$ can be estimated as

$$\begin{aligned} \boldsymbol{A}^{(j)} &= \tilde{\phi}(\boldsymbol{Z}^{(j)}), \\ \boldsymbol{Z}^{(j+1)} &= \boldsymbol{W}^{(j)}\boldsymbol{A}^{(j)} + \boldsymbol{B}^{(j)}, \end{aligned} \tag{3.8}$$

where $\boldsymbol{\theta}^{(j)} = \{\boldsymbol{W}^{(j)}, \boldsymbol{B}^{(j)}\}$ are the parameters of the j^{th} layer. Figure 3.6 presents two procedures for inferring $\boldsymbol{Z}^{(j)}$ with and without bypassing the activation units $\boldsymbol{A}^{(j)}$. Following the former one as presented in Figure 3.6a, we estimate the posterior $f(\boldsymbol{z}^{(j)}|\boldsymbol{y})$ from $\boldsymbol{Z}^{(j+1)}$



Figure 3.6 Infer the hidden states of a layer j^{th} from the hidden states of layer $j+1^{th}$ (a) with, and (b) without bypassing activation units $\boldsymbol{a}^{(j)}$.

directly using

$$\begin{aligned}
\boldsymbol{\mu}_{\boldsymbol{Z}|\boldsymbol{y}}^{(j)} &= \boldsymbol{\mu}_{\boldsymbol{Z}}^{(j)} + \mathbf{J}_{\boldsymbol{Z}_1\boldsymbol{Z}_2}\left(\boldsymbol{\mu}_{\boldsymbol{Z}|\boldsymbol{y}}^{(j+1)} - \boldsymbol{\mu}_{\boldsymbol{Z}}^{(j+1)}\right), \\
\boldsymbol{\Sigma}_{\boldsymbol{Z}|\boldsymbol{y}}^{(j)} &= \boldsymbol{\Sigma}_{\boldsymbol{Z}}^{(j)} + \mathbf{J}_{\boldsymbol{Z}_1\boldsymbol{Z}_2}\left(\boldsymbol{\Sigma}_{\boldsymbol{Z}|\boldsymbol{y}}^{(j+1)} - \boldsymbol{\Sigma}_{\boldsymbol{Z}}^{(j+1)}\right)\mathbf{J}_{\boldsymbol{Z}_1\boldsymbol{Z}_2}^{\intercal}, \\
\mathbf{J}_{\boldsymbol{Z}_1\boldsymbol{Z}_2} &= \operatorname{cov}\left(\boldsymbol{Z}^{(j)}, \boldsymbol{Z}^{(j+1)}\right)(\boldsymbol{\Sigma}_{\boldsymbol{Z}}^{(j+1)})^{-1}.
\end{aligned} \tag{3.9}$$

On the other hand, following the procedure without bypassing presented in Figure 3.6b, we first estimate the posterior $f(\boldsymbol{a}^{(j)}|\boldsymbol{y})$ from $\boldsymbol{Z}^{(j+1)}$ as

$$\begin{aligned}
\boldsymbol{\mu}_{\boldsymbol{A}|\boldsymbol{y}}^{(j)} &= \boldsymbol{\mu}_{\boldsymbol{A}}^{(j)} + \mathbf{J}_{\boldsymbol{A}_1\boldsymbol{Z}_2}(\boldsymbol{\mu}_{\boldsymbol{Z}|\boldsymbol{y}}^{(j+1)} - \boldsymbol{\mu}_{\boldsymbol{Z}}^{(j+1)}), \\
\boldsymbol{\Sigma}_{\boldsymbol{A}|\boldsymbol{y}}^{(j)} &= \boldsymbol{\Sigma}_{\boldsymbol{A}}^{(j)} + \mathbf{J}_{\boldsymbol{A}_1\boldsymbol{Z}_2}(\boldsymbol{\Sigma}_{\boldsymbol{Z}|\boldsymbol{y}}^{(j+1)} - \boldsymbol{\Sigma}_{\boldsymbol{Z}}^{(j+1)})\mathbf{J}_{\boldsymbol{A}_1\boldsymbol{Z}_2}^{\intercal}, \\
\mathbf{J}_{\boldsymbol{A}_1\boldsymbol{Z}_2} &= \operatorname{cov}(\boldsymbol{A}^{(j)}, \boldsymbol{Z}^{(j+1)})(\boldsymbol{\Sigma}_{\boldsymbol{Z}}^{(j+1)})^{-1}.
\end{aligned} \tag{3.10}$$

We then estimate $f(\boldsymbol{z}^{(j)}|\boldsymbol{y})$ from $\boldsymbol{A}^{(j)}$ following

$$\begin{aligned}
\boldsymbol{\mu}_{\boldsymbol{Z}|\boldsymbol{y}}^{(j)} &= \boldsymbol{\mu}_{\boldsymbol{Z}}^{(j)} + \mathbf{J}_{\boldsymbol{Z}_1\boldsymbol{A}_1}(\boldsymbol{\mu}_{\boldsymbol{A}|\boldsymbol{y}}^{(j)} - \boldsymbol{\mu}_{\boldsymbol{A}}^{(j)}), \\
\boldsymbol{\Sigma}_{\boldsymbol{Z}|\boldsymbol{y}}^{(j)} &= \boldsymbol{\Sigma}_{\boldsymbol{Z}}^{(j)} + \mathbf{J}_{\boldsymbol{Z}_1\boldsymbol{A}_1}(\boldsymbol{\Sigma}_{\boldsymbol{A}|\boldsymbol{y}}^{(j)} - \boldsymbol{\Sigma}_{\boldsymbol{A}}^{(j)})\mathbf{J}_{\boldsymbol{Z}_1\boldsymbol{A}_1}^{\intercal}, \\
\mathbf{J}_{\boldsymbol{Z}_1\boldsymbol{A}_1} &= \operatorname{cov}(\boldsymbol{Z}^{(j)}, \boldsymbol{A}^{(j)})(\boldsymbol{\Sigma}_{\boldsymbol{A}}^{(j)})^{-1}.
\end{aligned} \tag{3.11}$$

Replacing $\boldsymbol{\mu}_{\boldsymbol{A}|\boldsymbol{y}}^{(j)}$ and $\boldsymbol{\Sigma}_{\boldsymbol{A}|\boldsymbol{y}}^{(j)}$ in Equation 3.11 by Equation 3.10 leads to

$$\begin{aligned}
\boldsymbol{\mu}_{\boldsymbol{Z}|\boldsymbol{y}}^{(j)} &= \boldsymbol{\mu}_{\boldsymbol{Z}}^{(j)} + \mathbf{J}_{\boldsymbol{Z}_1\boldsymbol{A}_1}\left[\cancel{\boldsymbol{\mu}_{\boldsymbol{A}}^{(j)}} + \mathbf{J}_{\boldsymbol{A}_1\boldsymbol{Z}_2}(\boldsymbol{\mu}_{\boldsymbol{Z}|\boldsymbol{y}}^{(j+1)} - \boldsymbol{\mu}_{\boldsymbol{Z}}^{(j+1)}) - \cancel{\boldsymbol{\mu}_{\boldsymbol{A}}^{(j)}}\right] \\
&= \boldsymbol{\mu}_{\boldsymbol{Z}}^{(j)} + \mathbf{J}_{\boldsymbol{Z}_1\boldsymbol{A}_1}\mathbf{J}_{\boldsymbol{A}_1\boldsymbol{Z}_2}(\boldsymbol{\mu}_{\boldsymbol{Z}|\boldsymbol{y}}^{(j+1)} - \boldsymbol{\mu}_{\boldsymbol{Z}}^{(j+1)}), \\
\boldsymbol{\Sigma}_{\boldsymbol{Z}|\boldsymbol{y}}^{(j)} &= \boldsymbol{\Sigma}_{\boldsymbol{Z}}^{(j)} + \mathbf{J}_{\boldsymbol{Z}_1\boldsymbol{A}_1}\mathbf{J}_{\boldsymbol{A}_1\boldsymbol{Z}_2}(\boldsymbol{\Sigma}_{\boldsymbol{Z}|\boldsymbol{y}}^{(j+1)} - \boldsymbol{\Sigma}_{\boldsymbol{Z}}^{(j+1)})\mathbf{J}_{\boldsymbol{Z}_1\boldsymbol{A}_1}^{\intercal}\mathbf{J}_{\boldsymbol{A}_1\boldsymbol{Z}_2}^{\intercal}.
\end{aligned} \tag{3.12}$$

Equation 3.12 is thus equivalent to Equation 3.9 because

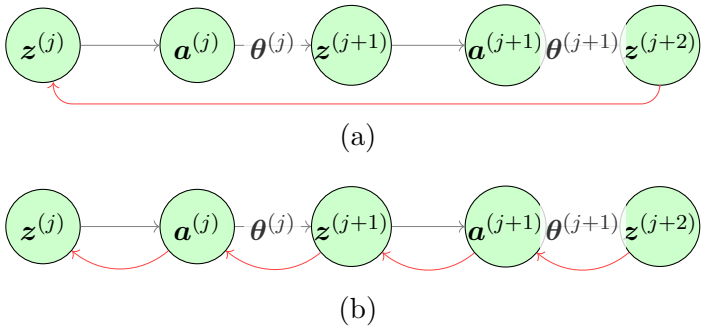
$$\mathbf{J}_{\boldsymbol{Z}_1\boldsymbol{Z}_2} = \mathbf{J}_{\boldsymbol{Z}_1\boldsymbol{A}_1}\mathbf{J}_{\boldsymbol{A}_1\boldsymbol{Z}_2} = \operatorname{cov}(\boldsymbol{Z}^{(j)}, \boldsymbol{A}^{(j)})\mathbb{E}[\boldsymbol{W}^{(j)}](\boldsymbol{\Sigma}_{\boldsymbol{Z}}^{(j+1)})^{-1}.$$

This means that both procedures presented in Figure 3.6 lead to an identical result so that the posterior $f(\boldsymbol{z}^{(j)}|\boldsymbol{y})$ can be inferred directly from $\boldsymbol{Z}^{(j+1)}$ while bypassing the activation $\boldsymbol{A}^{(j)}$.

We further investigate if inferring the posterior $f(\boldsymbol{z}^{(j)}|\boldsymbol{y})$ for the hidden states $\boldsymbol{Z}^{(j)}$ from $\boldsymbol{Z}^{(j+2)}$ directly, bypassing the $j+1^{th}$ layer, leads to the same result as going through it.

Following the procedure presented in Figure 3.7a, we bypass the hidden states $\boldsymbol{Z}^{(j+1)}$, the activation units $\boldsymbol{A}^{(j)}$ and $\boldsymbol{A}^{(j+1)}$, and estimate the posterior $f(\boldsymbol{z}^{(j)}|\boldsymbol{y})$ from $\boldsymbol{Z}^{(j+2)}$ so that

$$\begin{aligned}
\boldsymbol{\mu}_{\boldsymbol{Z}|\boldsymbol{y}}^{(j)} &= \boldsymbol{\mu}_{\boldsymbol{Z}}^{(j)} + \mathbf{J}_{\boldsymbol{Z}_1\boldsymbol{Z}_3}\left(\boldsymbol{\mu}_{\boldsymbol{Z}|\boldsymbol{y}}^{(j+2)} - \boldsymbol{\mu}_{\boldsymbol{Z}}^{(j+2)}\right), \\
\boldsymbol{\Sigma}_{\boldsymbol{Z}|\boldsymbol{y}}^{(j)} &= \boldsymbol{\Sigma}_{\boldsymbol{Z}}^{(j)} + \mathbf{J}_{\boldsymbol{Z}_1\boldsymbol{Z}_3}\left(\boldsymbol{\Sigma}_{\boldsymbol{Z}|\boldsymbol{y}}^{(j+2)} - \boldsymbol{\Sigma}_{\boldsymbol{Z}}^{(j+2)}\right)\mathbf{J}_{\boldsymbol{Z}_1\boldsymbol{Z}_2}^{\intercal}, \\
\mathbf{J}_{\boldsymbol{Z}_1\boldsymbol{Z}_3} &= \operatorname{cov}\left(\boldsymbol{Z}^{(j)}, \boldsymbol{Z}^{(j+2)}\right)(\boldsymbol{\Sigma}_{\boldsymbol{Z}}^{(j+2)})^{-1}.
\end{aligned} \tag{3.13}$$



(a)

(b)

Figure 3.7 Infer the hidden states of a layer j^{th} from the hidden states of layer $j+2^{th}$ (a) with, and (b) without bypassing the layer $j+1^{th}$.

On the other hand, without bypassing the layer $j+1^{th}$, we sequentially estimate the posterior $f(\boldsymbol{a}^{(j+1)}|\boldsymbol{y})$ from $\boldsymbol{Z}^{(j+1)}$, $f(\boldsymbol{z}^{(j+1)}|\boldsymbol{y})$ from $\boldsymbol{A}^{(j+2)}$, $f(\boldsymbol{a}^{(j)}|\boldsymbol{y})$ from $\boldsymbol{Z}^{(j+1)}$ and $f(\boldsymbol{z}^{(j)}|\boldsymbol{y})$ from $\boldsymbol{A}^{(j)}$ as presented in Figure 3.7b. Doing this leads to the same result shown in Equation 3.13 which confirms that the posterior $f(\boldsymbol{z}^{(j)}|\boldsymbol{y})$ can be inferred either with or without bypassing the $j+1^{th}$layer. It shows that we can bypass one or multiple hidden layers or variables during inference in order to estimate the posterior for the hidden states at a layer of interest.

3.5 Experiments

In this section, we first conduct a qualitative experiment using synthetic time series to verify the TAGI-LSTM's capability to accurately retrieve the ground truth. Next, we compare the predictive performance of the TAGI-LSTM and TAGI-GRU methods with their corresponding deterministic and variational LSTM and GRU models [106] trained with a gradient-based approach on the Electricity and Traffic benchmark datasets [5]. These datasets are chosen for comparison because they are stationary, i.e., each time series displays a constant baseline, so that the LSTM and GRU models can analyze them directly without requiring data preprocessing to remove trends. Finally, we evaluate the performance of TAGI-LSTM on a real SHM dataset of crack opening obtained from a dam in Canada.

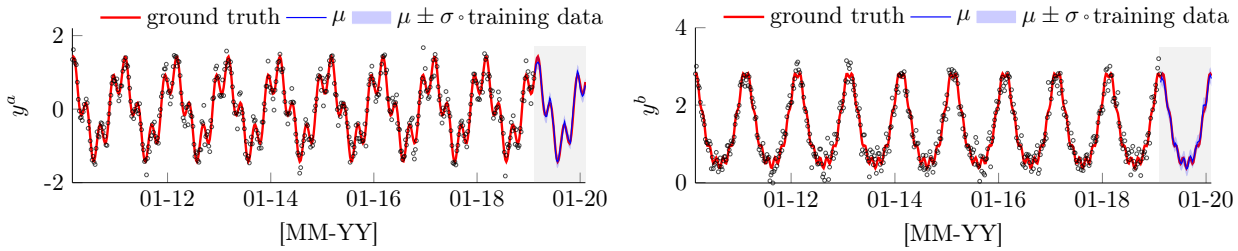
3.5.1 Verification on Synthetic Data

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

$$y_t^a = \sin(\frac{2\pi t}{365.22}) + 0.5\sin(\frac{2\pi t}{365.22/4}) + v_t, \tag{3.14a}$$

$$y_t^b = \exp\left[\sin(\frac{2\pi t}{365.22})\right] + \left[0.5\sin(\frac{2\pi t}{365.22/4})\right]^2 + v_t, \tag{3.14b}$$

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. Figure 3.8 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 predictions show that the TAGI-LSTM can accurately retrieve the ground truth for these synthetic data.



(a) (b)

Figure 3.8 Test predictions from TAGI-LSTM models for synthetic data generated using (a) Equation 3.14a, and (b) Equation 3.14b. 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. The $\pm\sigma$ regions contain both the epistemic (parameter's) uncertainties obtained from the prior predictive distribution of $z^{(0)}$ as well as the aleatory uncertainties associated with the error term's variance σ_V^2.

3.5.2 Experiment – Stationary Time Series Benchmark Dataset

In this experiment, we compare the predictive performance of the TAGI-LSTM and TAGI-GRU methods with their corresponding deterministic and variational LSTM and GRU models [106] trained with a gradient-based approach on the Electricity and Traffic benchmark datasets [5]. The Electricity dataset contains 370 hourly time series of electricity consumption, whereas the Traffic dataset includes 963 hourly time series of occupancy rate of different car lanes in the San Francisco bay area. Following previous studies [3,5,9], the task for these

datasets consists in providing predictions for seven consecutive days using the rolling window operation described in [5]. 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 window. 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 which are detailed in Table 3.1 [9]. We standardize the training data so that it has a zero mean and a unit standard deviation. Both datasets experience hourly and daily recurrent patterns. Therefore, we include the 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 [107] so that the input covariate vector includes a lookback window of posterior values for $\boldsymbol{z}^{(\mathtt{0})}_{t-\mathtt{W}:t-1}$ where $\mathtt{W}$ is the window's length.

Table 3.1 Train and test splits for Electricity and Traffic datasets. Subscript [1] indicates the splits used in DeepFactor [2], [2] splits used in DeepAR [3] and DeepState [4], [3] splits used in MatFact [5].

	Electricity			Traffic		
Split	2014-03-31[1]	2014-09-01[2]	last 7 days[3]	2008-01-14[1]	2008-06-15[2]	last 7 days[3]
Train starting time	2014-03-24	2014-01-01	2012-01-01	2008-01-07	2008-01-01	2008-01-01
Test starting time	2014-03-31	2014-09-01	2014-12-25	2008-01-14	2008-06-15	2009-03-09

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 one which maximizes the log-likelihood for the validation set, and obtain the network parameters at this optimal epoch. We build a separate model for each time series where all models use the same network architecture as used by [3]. The standard deviation for the observation error σ_V is a hyperparameter in our TAGI-LSTM and TAGI-GRU models. Note that σ_V is common to all time series in the same dataset, and its values are standardized. For variational LSTM and GRU models, we perform 50 dropout passes and calculate the predictive mean and variance. The architecture and hyperparameters used in this experiment are given in Table 3.2.

The quantile loss metric [4] measures the accuracy of the predictive distribution at a specific quantile. The ρ-quantile loss with $\rho \in (0, 1)$ is defined by

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

Table 3.2 Architecture and hyperparameters for models used in our experiments. The Electricity and Traffic datasets have three train/test splits as presented in Table 3.1; the value for each split is separated by a forward slash. d, w, m, q, and y are the abbreviations for day, week, month, quarter, and year respectively.

Dataset	Synthetic	Electricity	Traffic	Crack opening
# layer	1	3	3	2
# nodes	50	40	40	50
Batch size	1	4/16/16	4/16/16	1
Lookback window's length `W`	52	24/168/168	24/168/168	52
σ_V	0.2	0.5	0.5	0.1
Validation set	1y	1d/2w/2w	1d/2w/2w	6m
Test set	1y	1w/1w/1w	1w/1w/1w	2y

where $y_{i,t}$ is the observation at time t of the i^{th} time series and $\hat{y}^{\rho}_{i,t}$ is the corresponding predicted ρ-quantile. Note that the $p50$-loss is equal to the Normalized deviation (ND) metric reported in [3, 5, 9], and the ND metric was originally used to compare these datasets by [5]. Table 3.3 presents the test $p50$ and $p90$-losses, whereas Table 3.4 reports the test root mean square error (RMSE) and mean absolute scaled error (MASE) metrics for all methods. These results are obtained by averaging the predictions from five independent runs with different initial seeds. The deterministic LSTM and GRU models provide point estimates so that we only report the $p50$-loss, whereas both the $p50$ and $p90$-losses are provided for other methods. The results from Tables 3.3 and 3.4 show that TAGI-LSTM and TAGI-GRU models match the performance of the their corresponding deterministic and variational LSTM and GRU models 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, the deterministic LSTM and GRU 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.

We assess the robustness of the TAGI-LSTM method by comparing the predictive performance of four different TAGI-LSTM models where each has a different number of hidden layers, ranging from one to four, and each layer has 40 units. Table 3.5 shows that the models with three hidden layers provide better predictive accuracies, but overall all models provide a comparable performance. Figures 3.9 compare the test predictions from the

Table 3.3 Comparison between TAGI-LSTM, TAGI-GRU, and their corresponding deterministic and variational LSTM and GRU models trained with backpropagation on Electricity and Traffic datasets. $p50/p90$-loss for test sets are calculated using the rolling window operation. Results are obtained by averaging the forecasts over five independent runs. Deterministic LSTM and GRU models provide only point forecasts so that we report only the $p50$-loss. Three train/test splits are presented in Table 3.1. Bold fonts indicate the best results.

	Electricity						Traffic					
Split	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
TAGI-GRU	0.081	**0.054**	0.070	0.053	0.164	**0.094**	0.355	0.335	0.191	0.163	0.123	0.137
GRU	0.085	-	0.077	-	0.167	-	0.321	-	0.168	-	0.108	-
Variational GRU	0.081	0.057	0.066	**0.052**	0.157	0.099	0.329	0.262	0.168	0.155	0.132	0.137

Table 3.4 RMSE and MASE metrics for the Electricity and Traffic datasets. Results are obtained by averaging the forecasts over five independent runs. Bold fonts indicate the best results.

	Electricity						Traffic					
Split	2014-03-31		2014-09-01		last 7 days		2008-01-14		2008-06-15		last 7 days	
	RMSE	MASE	RMSE	MASE	RMSE	MASE	RMSE	MASE	RMSE	MASE	RMSE	MASE
TAGI-LSTM	1183	1.187	1431	1.028	**1837**	**2.067**	0.036	1.624	**0.022**	0.652	0.016	0.406
LSTM	1198	1.239	1826	1.021	1865	2.210	**0.035**	**1.535**	**0.022**	**0.613**	**0.013**	**0.400**
Variational LSTM	1197	**1.160**	**1365**	1.006	1985	2.312	0.036	1.544	**0.022**	0.628	0.017	0.485
TAGI-GRU	**1142**	1.233	1536	**0.983**	1932	2.085	0.038	1.700	0.023	0.745	0.017	0.485
GRU	1210	1.222	1873	1.031	1982	2.208	**0.035**	**1.535**	**0.022**	0.646	0.014	0.438
Variational GRU	1211	1.167	1455	1.013	1885	2.296	0.036	1.565	**0.022**	0.652	0.018	0.520

Table 3.5 Comparison between four different TAGI-LSTM models having different architecture on Electricity and Traffic datasets. $p50/p90$-loss for test sets are calculated using the rolling window operation. Results are obtained by averaging the forecasts over five independent runs. Bold fonts indicate the best results.

	Electricity						Traffic					
Split	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-1-layer	0.082	0.057	0.069	0.055	0.180	0.100	0.340	**0.268**	0.185	**0.158**	0.146	0.144
TAGI-LSTM-2-layer	**0.079**	**0.056**	0.068	0.056	0.157	0.098	**0.328**	0.292	0.172	0.159	0.105	0.131
TAGI-LSTM-3-layer	0.080	0.058	**0.066**	0.053	**0.152**	0.095	0.337	0.276	**0.169**	**0.158**	**0.102**	**0.130**
TAGI-LSTM-4-layer	0.084	0.060	0.067	**0.052**	0.153	**0.094**	0.341	0.310	0.170	**0.158**	0.111	0.133

TAGI-LSTM and deterministic LSTM models for two time series as examples. The figures show that both models offer accurate forecasts. Note that for a fair comparison with other

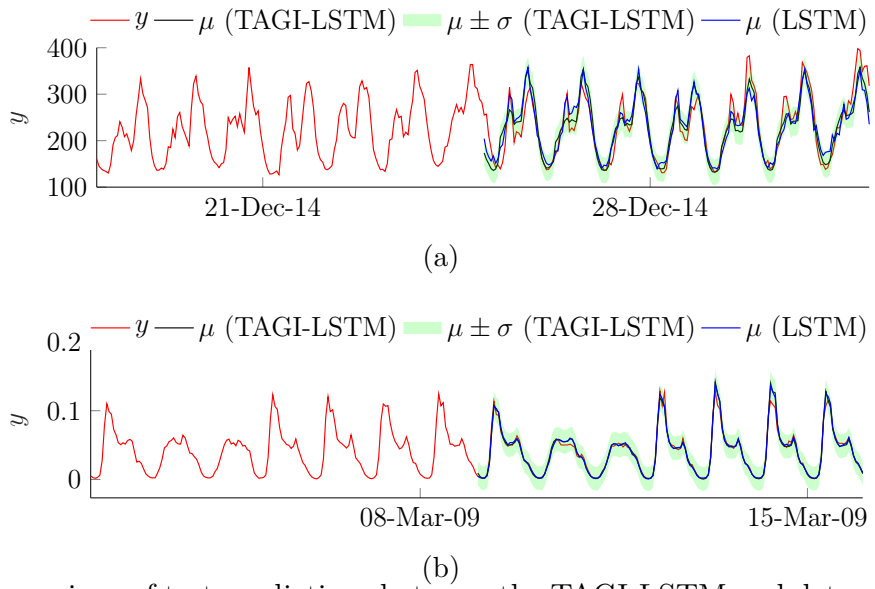


Figure 3.9 Comparison of test predictions between the TAGI-LSTM and deterministic LSTM models for a time series in (a) Electricity, and (b) Traffic datasets. The $\pm\sigma$ regions contain both the epistemic (parameter's) uncertainties obtained from the prior predictive distribution of $z^{(0)}$ as well as the aleatory uncertainties associated with the error term's variance σ_V^2. Only a part of the training set is presented.

methods, this experiment considers a constant value for the standard deviation σ_V of the observation error where its value is chosen using grid-search. However, the AGVI method presented in [8] allows to estimate the time-varying σ_V as presented in Figure 3.10. This method will be used in Section 4.3.2.

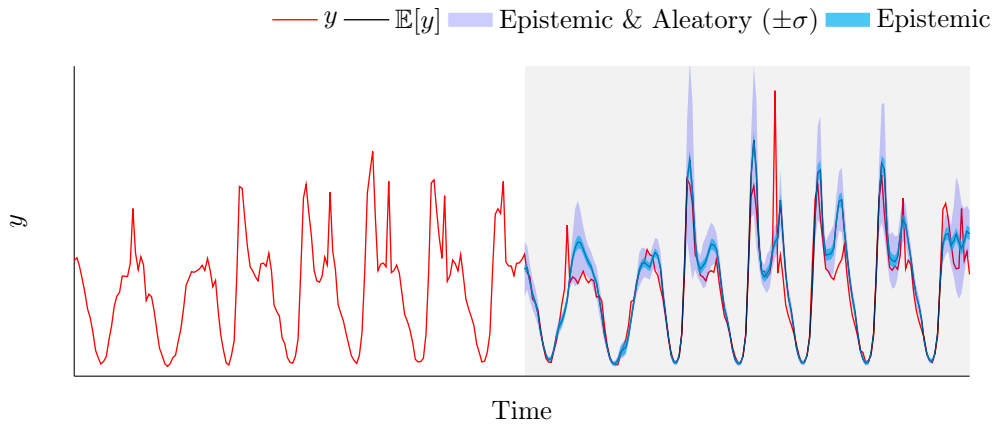


Figure 3.10 Example of learning the time-varying aleatory uncertainty (σ_V) estimated using the AGVI method [8] for a time series in the traffic dataset. The grey shaded area presents the test set, and only a part of the training set is presented.

3.5.3 Experiment – Structural Health Monitoring Dataset

All of the time series used in the experiment presented in Section 3.5.2 are typically stationary such that they have a constant trend over time. In this experiment, we evaluate the predictive performance of the TAGI-LSTM model on a trended structural health monitoring (SHM) time series. Figure 3.11 presents a weekly crack opening data obtained from a dam in Canada from Feb-1998 to Jan-2006. This data displays a positive linear trend over time. The data is divided into training and test sets where the test period includes two years of measurements after 11-Jan-2004. A part of data at the end of the training set from 13-Jul-2003 to 10-Jan-2004 is reserved for validation.

We conduct two analyses using the trended and detrended data extracted from the original one in order to evaluate the predictive performance of the TAGI-LSTM in both cases. The architecture and hyperparameters used in this experiment are given in Table 3.2, and the data processing only includes standardization. In the first analysis, we train a TAGI-LSTM using the original trended data. Figure 3.11 presents the test predictions and predictive uncertainties for this analysis. The result shows that the TAGI-LSTM model provides poor forecasts such that the predictions cannot capture the positive trend displayed in the data.

For the second analysis, we detrend the original data into a linear trend and a zero-mean recurrent pattern as presented Figures 3.12. The trend is described by the linear function $y = 2.76 \cdot 10^{-4} + 0.284t$, where t is the number of days from the starting date of the time series. This trend is obtained by fitting an affine function to the original trended data, the zero-mean recurrent pattern (Figure 3.12b) is then obtained by the subtracting the trend from the original data. We use this zero-mean data to train a TAGI-LSTM model. Figure 3.13a presents the test predictions for the zero-mean data, whereas Figure 3.13b displays the final predictions in the original space which are obtained by summing up the zero-mean predictions

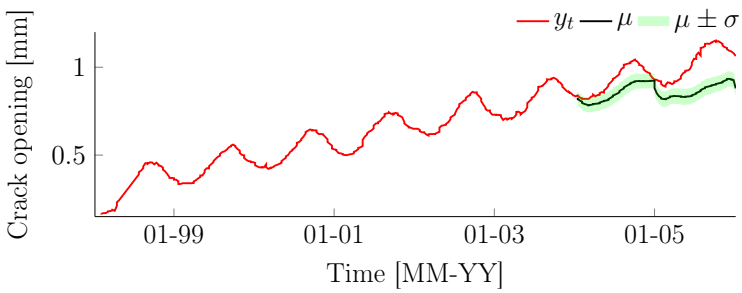
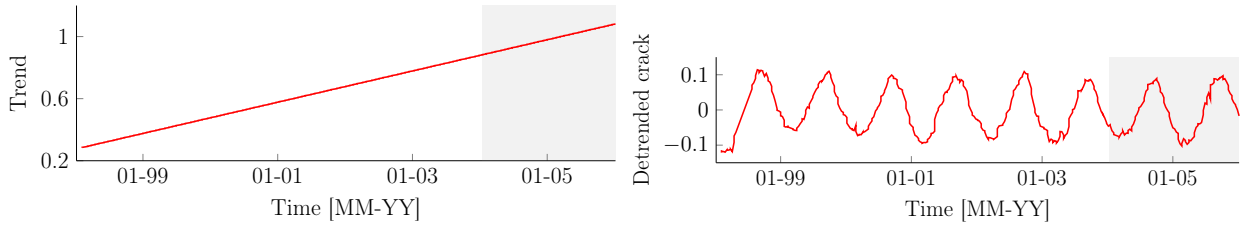


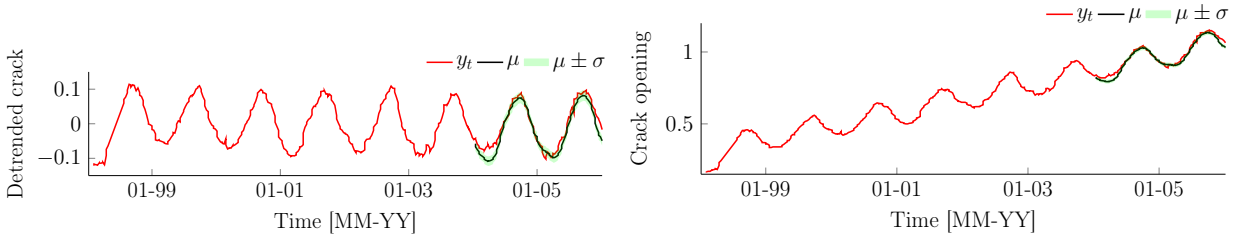
Figure 3.11 Test predictions from the TAGI-LSTM trained using the original trended data. The $\pm\sigma$ regions contain both the epistemic (parameter's) uncertainties obtained from the prior predictive distribution of $z^{(0)}$ as well as the aleatory uncertainties associated with the error term's variance σ_V^2.



(a) (b)

Figure 3.12 (a) Trend and (b) zero-mean recurrent pattern components obtained by detrending the original crack opening time series. The grey shaded area presents the forecast period.

and the trend component. From Figures 3.11 and 3.13b, it is observed that the predictions obtained from the second analysis using the detrended data are way more accurate than those obtained from the first analysis using the original trended data, such that now, both the recurrent pattern and linear trend are well captured. This comparative analysis demonstrates the necessity to detrend data before training a RNN-based model. In the next chapter, we will see how we can leverage the coupling between the TAGI-RNN and SSM to analyze directly trended data with either simple linear or complex nonlinear trends, without requiring offline data preprocessing to remove the trend as it was done in this experiment.



(a) (b)

Figure 3.13 (a) The test predictions for the zero-mean recurrent pattern, and (b) the final test predictions obtained by summing up the zero-mean predictions and the trend component. The $\pm\sigma$ regions contain both the epistemic (parameter's) uncertainties obtained from the prior predictive distribution of $z^{(0)}$ as well as the aleatory uncertainties associated with the error term's variance σ_V^2.

3.6 Conclusion

The new mathematical formulations proposed in this chapter enable using the Tractable Approximate Gaussian Inference (TAGI) method with the LSTM and GRU neural network architectures. The approach allows estimating analytically the posterior mean vectors and diagonal covariance matrices for the hidden states and model's parameters using analytical approximate Bayesian inference. The proposes method employs the assumption that all quantities in the LSTM and GRU neural networks are modelled as Gaussian random variables having a diagonal covariance structure, along with the conditional independence assumption between the hidden states of two layers that are not connected. These assumptions are necessary to maintain the computational tractability of the TAGI method. The experiments on the Electricity and Traffic benchmarks showed that for a same network architecture, our TAGI-LSTM and TAGI-GRU models provide on-par performance compared to the deterministic and variational LSTM and GRU models trained with gradient descent and backpropagation. The experiments on the trended SHM time series outlined that RNN-based methods including TAGI-LSTM and TAGI-GRU must be applied on stationary data.

CHAPTER 4 Coupling Analytically Tractable Bayesian Recurrent Neural Networks and State-space Models

4.1 Introduction

The results obtained from recurrent neural networks (RNN) are typically difficult to interpret [24]. As a result, RNN have been coupled with state-space models (SSM) in order to create hybrid models that allows to provide interpretable results, while keeping the advantages of RNN. However, existing hybrid methods have only explored using deterministic approaches which fails to take into account the epistemic uncertainties associated with the neural network parameters. In this chapter, we propose a novel probabilistic coupling between the analytically tractable Bayesian RNN and SSM. The resulting hybrid model considers the epistemic uncertainties and provides interpretable results along with prediction uncertainties, while not requiring feature engineering nor parameter optimization which are key for scalability. Moreover, it is able to analyze both stationary and trended data, without requiring offline preprocessing to remove trends. The contributions of this chapter include

- Develop the mathematical formulations to probabilistically couple the analytically tractable Bayesian TAGI-LSTM and TAGI-GRU and SSM, while Bayesian inference is used as the single mechanism for learning both the network's parameters and SSM's hidden states.
- Develop a novel exponential component for SSM to capture nonlinear complex trends in data.
- Validate the proposed method by comparing them with other models on three time series benchmark datasets and three structural health monitoring ones.

4.2 Methodology

In this section, we present a methodology to probabilistically couple the TAGI-LSTM model presented in Section 3.2 with states-space models (SSM) to create the TAGI-LSTM/SSM hybrid model. Note that the coupling between TAGI-GRU and SSM can be done analogously.

We consider a structured state-space model with additive errors where the hidden state vector $\boldsymbol{z}$ consists in two sets of hidden states $\boldsymbol{z}^{\mathtt{B}}$ and $z^{(\mathtt{0})}$, $\boldsymbol{z} = [\boldsymbol{z}^{\mathtt{B}}\ z^{(\mathtt{0})}]^{\intercal}$. The baseline hidden state vector $\boldsymbol{z}^{\mathtt{B}}$ models the time series' baseline patterns, and is associated with the linear

dynamical system

$$\boldsymbol{z}_t^{\mathtt{B}} = \mathbf{A}\boldsymbol{z}_{t-1}^{\mathtt{B}} + \boldsymbol{w}_t, \tag{4.1}$$

where $\mathbf{A}$ is the transition matrix for the baseline hidden states, $\boldsymbol{w}_t$ is a realization from the independent error process $\boldsymbol{W} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$, and $\mathbf{Q}$ is the process error's covariance matrix. The baseline hidden state vector $\boldsymbol{z}^{\mathtt{B}}$ can include one or all hidden states including the level (`L`) for modeling long-term pattern, the local trend (`LT`) for capturing the level's rate of change, and the local acceleration (`LA`) for modeling the local trend's rate of change.

The pattern hidden state $z^{(\mathtt{0})}$ is employed to model the recurrent patterns present in the data. Contrarily to the baseline hidden states which follow the transition model described in Equation 4.1, the pattern hidden state is predicted using the output node of a TAGI-LSTM network as illustrated in Figure 4.1 so that

$$z_t^{(\mathtt{0})} = \text{TAGI-LSTM}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}, \boldsymbol{c}_{t-1}, \boldsymbol{\theta}_{t-1}), \tag{4.2}$$

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

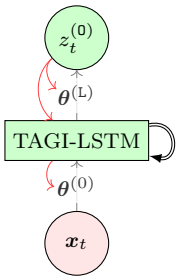


Figure 4.1 The nonlinear transition function for the pattern hidden states is modelled by a TAGI-LSTM network. Red arrows represent the layer-wise inference paths for inferring the posterior knowledge for the TAGI-LSTM's hidden states and parameters from the posterior PDF of the pattern hidden state $Z^{(\mathtt{0})}$. Black arrows represent the network forward connections, and double arrows represent the recurrent connections.

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 $\boldsymbol{Z}_{t|t-1}^{\mathtt{B}} \sim \mathcal{N}(\boldsymbol{\mu}_{t|t-1}^{\mathtt{B}}, \boldsymbol{\Sigma}_{t|t-1}^{\mathtt{B}})$ which is obtained by propagating the uncertainty associated with the posterior at $t-1$, i.e., $\boldsymbol{Z}_{t-1|t-1}^{\mathtt{B}}$ through the transition model described in Equation 4.1. The marginal prior at t for the pattern hidden state is a Gaussian random variable $Z_{t|t-1}^{(\mathtt{0})} \sim \mathcal{N}(\mu_{t|t-1}^{Z^{(\mathtt{0})}}, \sigma_{t|t-1}^{Z^{(\mathtt{0})}})$ obtained using the TAGI-LSTM one-step-ahead prediction presented in Section 3.2.1. In order to obtain the joint prior knowledge at t for both the baseline and the pattern hidden states, we need their cross-

covariance $\text{cov}(\boldsymbol{Z}^{\mathtt{B}}_{t|t-1}, Z^{(\mathtt{0})}_{t|t-1})$ for which no exact closed-form analytical expression is available. In practice, these cross-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 $\boldsymbol{Z}^{\mathtt{B}}_{t|t-1}$ and $Z^{(\mathtt{0})}_{t|t-1}$ are independent so that $\text{cov}(\boldsymbol{Z}^{\mathtt{B}}_{t|t-1}, Z^{(\mathtt{0})}_{t|t-1}) = \mathbf{0}$. Section 4.3.1 further explores the empirical validity of that hypothesis through experiments.

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

$$y_t = \mathbf{F}\boldsymbol{z}_t + v_t, \tag{4.3}$$

where v_t is a realization from the independent and identically distributed (i.i.d.) error $V \sim \mathcal{N}(0, \sigma_V^2)$. In the observation model presented in Equation 4.3, the observation vector indicates which hidden state is observable such that $\mathbf{F} = [\mathbf{F}^{\mathtt{B}}\ \mathrm{F}^{(\mathtt{0})}]$, where $\mathrm{F}^{(\mathtt{0})} = 1$ and the composition of $\mathbf{F}^{\mathtt{B}}$ 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 $\boldsymbol{Z}^{\mathtt{B}}_{t|t}$ and $Z^{(\mathtt{0})}_{t|t}$. The later is then used by the TAGI-LSTM method in order to infer both the Gaussian posterior for their model parameters $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}^{\boldsymbol{\theta}}_{t|t}, \boldsymbol{\sigma}^{\boldsymbol{\theta}}_{t|t})$ and their hidden states $\boldsymbol{H}_t \sim \mathcal{N}(\boldsymbol{\mu}^{\boldsymbol{H}}_{t|t}, \boldsymbol{\sigma}^{\boldsymbol{H}}_{t|t})$ as presented by the red arrows in Figure 4.1. Algorithm 1 summarizes the procedure to apply the TAGI-LSTM/SSM model at one time step.

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

$$\text{cov}(\boldsymbol{Z}_{t|t}, \boldsymbol{Z}_{t+1|t}) = \text{blkdiag}\left(\boldsymbol{\Sigma}^{\mathtt{B}}_{t|t}\mathbf{A}^{\intercal}, \text{cov}(Z^{(\mathtt{0})}_{t|t}, Z^{(\mathtt{0})}_{t+1|t})\right),$$

where $\text{cov}(Z^{(\mathtt{0})}_{t|t}, Z^{(\mathtt{0})}_{t+1|t})$ is calculated using the smoothing procedure presented in Section 3.2.3. 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 2.3.3, we train the model and learn the network parameters $\boldsymbol{\theta}$ over multiple epochs, where for each of them, we perform a forward pass over time and then a backward one using the Kalman smoother.

4.2.1 Exponential Smoothing Component

Using the local level or local trend components [20] for the baseline hidden states $\boldsymbol{z}^{\mathtt{B}}$, the TAGI-LSTM/SSM can only model data which contains either a constant or linear long-term trend. Although it is not explicitly shown in this study, using the local acceleration component [20] would allow to capture quadratic trends. Beside these trends, SHM data could contain complex long-term patterns involving a succession of constant, linear and

Algorithm 1: One-time step of the proposed TAGI-LSTM/SSM method

Input: y_t, $\boldsymbol{\mu}_{t-1|t-1}$, $\boldsymbol{\Sigma}_{t-1|t-1}$, $\mathbf{A}$, $\mathbf{F}$, $\mathbf{Q}$, $\mathbf{R} = \sigma_V^2$, $\boldsymbol{x}_t$, $\boldsymbol{\theta}_{t-1|t-1}$, $\boldsymbol{H}_{t-1|t-1}$, $\boldsymbol{C}_{t-1|t-1}$
Output: $\boldsymbol{\mu}_{t|t}$, $\boldsymbol{\Sigma}_{t|t}$, $\boldsymbol{\theta}_{t|t}$, $\boldsymbol{H}_{t|t}$, $\boldsymbol{C}_{t|t}$

Prediction Step:

1: Baseline hidden states:
$\boldsymbol{\mu}^{\mathtt{B}}_{t|t-1} = \mathbf{A}\boldsymbol{\mu}^{\mathtt{B}}_{t-1|t-1}$, $\boldsymbol{\Sigma}^{\mathtt{B}}_{t|t-1} = \mathbf{A}\boldsymbol{\Sigma}^{\mathtt{B}}_{t-1|t-1}\mathbf{A}^{\intercal} + \mathbf{Q}$.

2: Pattern hidden state:
$[\mu^{Z^{(\mathtt{0})}}_{t|t-1}, \sigma^{Z^{(\mathtt{0})}}_{t|t-1}] = \text{TAGI-LSTM}\left(\boldsymbol{x}_t, \boldsymbol{H}_{t-1|t-1}, \boldsymbol{C}_{t-1|t-1}, \boldsymbol{\theta}_{t-1|t-1}\right)$.

3: Joint prior knowledge:
$$\boldsymbol{\mu}_{t|t-1} = \begin{bmatrix} \boldsymbol{\mu}^{\mathtt{B}} \\ \mu^{Z^{(\mathtt{0})}} \end{bmatrix}_{t|t-1}, \; \boldsymbol{\Sigma}_{t|t-1} = \begin{bmatrix} \boldsymbol{\Sigma}^{\mathtt{B}} & \mathbf{0} \\ \mathbf{0} & \left(\sigma^{Z^{(\mathtt{0})}}\right)^2 \end{bmatrix}_{t|t-1}.$$

4: $\boldsymbol{r}_t = y_t - \mathbf{F}\boldsymbol{\mu}_{t|t-1}$.

5: $\mathbf{K}_t = \boldsymbol{\Sigma}_{t|t-1}\mathbf{F}^{\intercal}\left(\mathbf{F}\boldsymbol{\Sigma}_{t|t-1}\mathbf{F}^{\intercal} + \mathbf{R}\right)^{-1}$.

Update Step:

6: Hidden states:
$$\boldsymbol{\mu}_{t|t} = \begin{bmatrix} \boldsymbol{\mu}^{\mathtt{B}} \\ \mu^{Z^{(\mathtt{0})}} \end{bmatrix}_{t|t} = \boldsymbol{\mu}_{t|t-1} + \mathbf{K}_t\boldsymbol{r}_t,$$
$$\boldsymbol{\Sigma}_{t|t} = \begin{bmatrix} \boldsymbol{\Sigma}^{\mathtt{B}} & \text{cov}(\boldsymbol{Z}^{\mathtt{B}}, Z^{(\mathtt{0})}) \\ \text{cov}(\boldsymbol{Z}^{\mathtt{B}}, Z^{(\mathtt{0})}) & \left(\sigma^{Z^{(\mathtt{0})}}\right)^2 \end{bmatrix}_{t|t} = (\mathbf{I} - \mathbf{K}_t\mathbf{F})\boldsymbol{\Sigma}_{t|t-1}$$

7: TAGI-LSTM's parameters:
Infer $\boldsymbol{\theta}_{t|t}$, $\boldsymbol{H}_{t|t}$, $\boldsymbol{C}_{t|t}$ using $\mu^{Z^{(\mathtt{0})}}_{t|t}$ and $\sigma^{Z^{(\mathtt{0})}}_{t|t}$ following Figure 4.1.

8: Return $\boldsymbol{\mu}_{t|t}$, $\boldsymbol{\Sigma}_{t|t}$, $\boldsymbol{\theta}_{t|t}$, $\boldsymbol{H}_{t|t}$, $\boldsymbol{C}_{t|t}$

quadratic trends. For modeling such complex data, one can either use the Switching Kalman Filter (SKF) [22] or exponential smoothing [108] methods. In this section, we formulate a parameter-free exponential smoothing component which can be coupled with the TAGI-LSTM/SSM to automatically detrend data containing complex long-term patterns. Section 5 will demonstrate how to use the SKF for a similar task.

Consider the state-space form of a simple exponential smoothing model [108], but now we consider the smoothing parameter z^{α} and the error term $z^{\mathtt{V}}$ as hidden states. The model is

then defined by the following equations

$$\begin{aligned} \text{Observation equation:} \quad & y_t = z_t^{\mathtt{E}} + z_t^{\mathtt{V}}, && (4.4) \\ \text{Transition equations:} \quad & z_t^{\mathtt{E}} = z_{t-1}^{\mathtt{E}} + \bar{z}_{t-1}^{\alpha} z_{t-1}^{\mathtt{V}}, \\ & z_t^{\alpha} = z_{t-1}^{\alpha}, && (4.5) \\ & z_t^{\mathtt{V}} = v_t, \end{aligned}$$

where $z_t^{\mathtt{E}}$ 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 2.3.3 to the hidden state z_t^{α}. The transformed variable $\bar{z}_t^{\alpha} = \tilde{\sigma}(z_t^{\alpha})$ is assumed to follow a Gaussian PDF where its moments are obtained by applying the equations for the linearized activation function presented in Section 2.3.3. The product $\bar{z}_{t-1}^{\alpha} z_{t-1}^{\mathtt{V}}$ uses the previous step's error $z_{t-1}^{\mathtt{V}}$ to adjust the predicted exponential smoothing hidden state $z_t^{\mathtt{E}}$ because we rely on the covariance $\text{cov}(\bar{Z}_{t-1|t-1}^{\alpha} Z_{t-1|t-1}^{\mathtt{V}}, Z_{t|t-1}^{\alpha})$ in order to update the hidden state z_t^{α}. Note that if instead we would use the product $\bar{z}_{t-1}^{\alpha} z_t^{\mathtt{V}}$, the covariance $\text{cov}(\bar{Z}_{t-1|t-1}^{\alpha} Z_{t|t-1}^{\mathtt{V}}, Z_{t|t-1}^{\alpha})$ would be zero, and thus, we would not be able to update $z_{t|t}^{\alpha}$.

The transition model as given in Equation 4.5 is nonlinear, however, it can be reformulated as a linear dynamic model [109] by creating a new hidden state $z_t^{\mathtt{N}} = \bar{z}_{t-1}^{\alpha} z_{t-1}^{\mathtt{V}}$ which is assumed to follow a Gaussian PDF where its moments, $\mathbb{E}[Z^{\mathtt{N}}] \equiv \mu^{\mathtt{N}}$ and $\text{var}(Z^{\mathtt{N}})$, are calculated exactly using the GMA equations given in Section 2.3.3. The baseline hidden state vector $\boldsymbol{z}^{\mathtt{B}} = [z^{\mathtt{E}} \; z^{\alpha} \; z^{\mathtt{V}}]^{\intercal}$ at time $t-1$ is thus assumed to be Gaussian. The augmented baseline hidden state vector $\tilde{\boldsymbol{z}}^{\mathtt{B}} = [\boldsymbol{z}^{\mathtt{B}} \; z^{\mathtt{N}}]^{\intercal}$ is defined by

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

where the components of $\tilde{\boldsymbol{\mu}}_{t-1|t-1}^{\mathtt{B}}$ and $\tilde{\boldsymbol{\Sigma}}_{t-1|t-1}^{\mathtt{B}}$ are given as

$$\begin{aligned} \tilde{\boldsymbol{\mu}}_{t-1|t-1}^{\mathtt{B}} &= \begin{bmatrix} \boldsymbol{\mu}^{\mathtt{B}} \\ \mu^{\mathtt{N}} \end{bmatrix}_{t-1|t-1}, \\ \tilde{\boldsymbol{\Sigma}}_{t-1|t-1}^{\mathtt{B}} &= \begin{bmatrix} \boldsymbol{\Sigma}^{\mathtt{B}} & \text{cov}(\boldsymbol{Z}^{\mathtt{B}}, Z^{\mathtt{N}}) \\ \text{cov}(Z^{\mathtt{N}}, \boldsymbol{Z}^{\mathtt{B}}) & \text{var}(Z^{\mathtt{N}}) \end{bmatrix}_{t-1|t-1}, \\ \text{cov}(\boldsymbol{Z}^{\mathtt{B}}, Z^{\mathtt{N}}) &= \text{cov}(\boldsymbol{Z}^{\mathtt{B}}, \bar{Z}^{\alpha} Z^{\mathtt{V}}) \\ &= \text{cov}(\boldsymbol{Z}^{\mathtt{B}}, Z^{\mathtt{V}}) \mathbb{E}[\bar{Z}^{\alpha}] + \text{cov}(\boldsymbol{Z}^{\mathtt{B}}, \bar{Z}^{\alpha}) \mathbb{E}[Z^{\mathtt{V}}]. \end{aligned}$$

The nonlinear Equation 4.5 now can be written in a linear form as

$$\tilde{\boldsymbol{z}}_t^{\mathtt{B}} = \mathbf{A}\tilde{\boldsymbol{z}}_{t-1}^{\mathtt{B}} + \boldsymbol{w}_t, \qquad \boldsymbol{w}_t : \boldsymbol{W} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}), \tag{4.6}$$

where $\mathbf{A}$ is the transition matrix and $\mathbf{Q}$ is the covariance matrix for the errors are

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{Q} = \text{diag}([0\ 0\ \sigma_V^2\ 0]).$$

The prior for the augmented baseline hidden state vector is given by

$$\tilde{\boldsymbol{Z}}_{t|t-1}^{\mathtt{B}} \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}_{t|t-1}^{\mathtt{B}}, \tilde{\boldsymbol{\Sigma}}_{t|t-1}^{\mathtt{B}}), \tag{4.7}$$

where $\tilde{\boldsymbol{\mu}}_{t|t-1}^{\mathtt{B}} = \mathbf{A}\tilde{\boldsymbol{\mu}}_{t-1|t-1}^{\mathtt{B}}$ and $\tilde{\boldsymbol{\Sigma}}_{t|t-1}^{\mathtt{B}} = \mathbf{A}\tilde{\boldsymbol{\Sigma}}_{t-1|t-1}^{\mathtt{B}}\mathbf{A}^{\intercal} + \mathbf{Q}$. The observation model given in Equation 4.4 is written in a matrix form as

$$y_t = \mathbf{F}\tilde{\boldsymbol{z}}_t^{\mathtt{B}}, \tag{4.8}$$

where $\mathbf{F} = [1\ 0\ 1\ 0]$ is the observation matrix. Using Equations 4.6-4.8, we can now apply the Kalman filter to obtain 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, without requiring offline data preprocessing.

4.3 Experiments

In this section, we first validate empirically the zero cross-covariances assumption between the baseline and pattern hidden states that is used in Section 4.2. We then 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 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 LSTM and GRU models on these benchmarks without preprocessing the data beforehand to remove trends. Finally, we use the TAGI-LSTM/SSM to analyze three structural heath monitoring datasets.

4.3.1 Validation of the Zero Cross-covariance Assumption

In this experiment, we validate the zero cross-covariance assumption between the baseline and pattern hidden states $\text{cov}(\boldsymbol{Z}^{\mathtt{B}}, Z^{(\mathtt{0})}) = \mathbf{0}$ that is introduced in Section 4.2 through experiments. We use six time series from the Tourism (monthly, quarterly) [110], and M4 (hourly) [111] datasets such that they all contain a linear trend. The data preprocessing only includes standardization. The baseline hidden state vector $\boldsymbol{z}^{\mathtt{B}} = [z^{\mathtt{L}} \; z^{\mathtt{LT}}]^{\intercal}$ includes a level ($\mathtt{L}$) and a local trend ($\mathtt{LT}$) hidden states [19], whereas the hidden state $z^{(\mathtt{0})}$ models the recurrent patterns. The model matrices are given as

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \mathbf{Q} = \sigma_W^2 \begin{bmatrix} 1/3 & 1/2 \\ 1/2 & 1 \end{bmatrix}, \mathbf{F} = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}, \tag{4.9}$$

where σ_W^2 is the error's variance.

In order to test the hypothesis of zero cross-covariances between the baseline and pattern hidden states presented in Section 4.2, we compare the hypothesis which assumes zero cross-covariances (no-cov), $\text{cov}(\boldsymbol{Z}^{\mathtt{B}}_{t|t-1}, Z^{(\mathtt{0})}_{t|t-1}) = \mathbf{0}$, and the one which estimates these cross-covariances through Monte-Carlo (MC) sampling (MC-cov). Table 4.1 shows that assuming zero cross-covariances provides nearly identical performance compared to the models which consider non-zero cross-covariance terms. Moreover, the average absolute correlations over all time steps presented in Table 4.1, $|\rho(Z^{\mathtt{L}}, Z^{(\mathtt{0})})|$ and $|\rho(Z^{\mathtt{LT}}, Z^{(\mathtt{0})})|$, are close to zero which confirms the validity of our hypothesis. Note that although it is possible to estimate these cross-covariance terms using MC sampling for a handful of time series, this approach is computationally prohibitive when modeling a large number of time series, and thus it is not suited for practical applications.

Table 4.1 Testing the zero cross-covariances hypothesis used in TAGI-LSTM/SSM by comparing the test log-likelihood performance between the models considering zero cross-covariances (no-cov) and the ones considering non-zero cross-covariances (MC-cov) on six time series from the Tourism and M4 datasets. The average absolute correlations are calculated over all training time steps. The $\mu \pm \sigma$ represents the mean and standard deviation over five runs.

Dataset	TS	Log-likelihood		Average absolute correlation	
		no-cov	MC-cov	$\lvert\rho(Z^{\mathtt{L}}, Z^{(\mathtt{0})})\rvert$	$\lvert\rho(Z^{\mathtt{LT}}, Z^{(\mathtt{0})})\rvert$
Tourism (Monthly)	TS #1	-177.94 ± 0.62	-177.89 ± 0.60	9.7E-3 $\pm$ 1.5E-3	7.8E-3 $\pm$ 1.4E-3
	TS #32	-248.31 ± 1.45	-248.34 ± 1.50	7.9E-3 $\pm$ 2.7E-3	7.3E-3 $\pm$ 2.5E-3
Tourism (Quarterly)	TS #33	-55.53 ± 0.14	-55.51 ± 0.14	9.9E-3 $\pm$ 1.1E-3	8.8E-3 $\pm$ 8.6E-4
	TS #81	-83.51 ± 0.17	-83.46 ± 0.05	1.7E-2 $\pm$ 2.8E-3	1.5E-2 $\pm$ 1.9E-3
M4 (Hourly)	TS #173	-61.35 ± 0.72	-61.35 ± 0.72	3.0E-3 $\pm$ 3.6E-4	2.7E-3 $\pm$ 3.0E-4
	TS #375	-208.8 ± 6.02	-208.7 ± 5.99	1.8E-3 $\pm$ 5.4E-4	1.6E-3 $\pm$ 4.5E-4

Figure 4.2 presents the test predictions as well as the model components for three time series from the monthly, quarterly Tourism, and hourly M4 datasets. These results show that the model can provide not only forecasts and their associated predictive uncertainties, but also successfully decomposes the data into interpretable components, i.e., the level hidden state $z^{\mathtt{L}}$ models the linear pattern, the local trend hidden state $z^{\mathtt{LT}}$ models the level's rate of change, and the hidden state $z^{(\mathtt{0})}$ models the recurrent pattern.

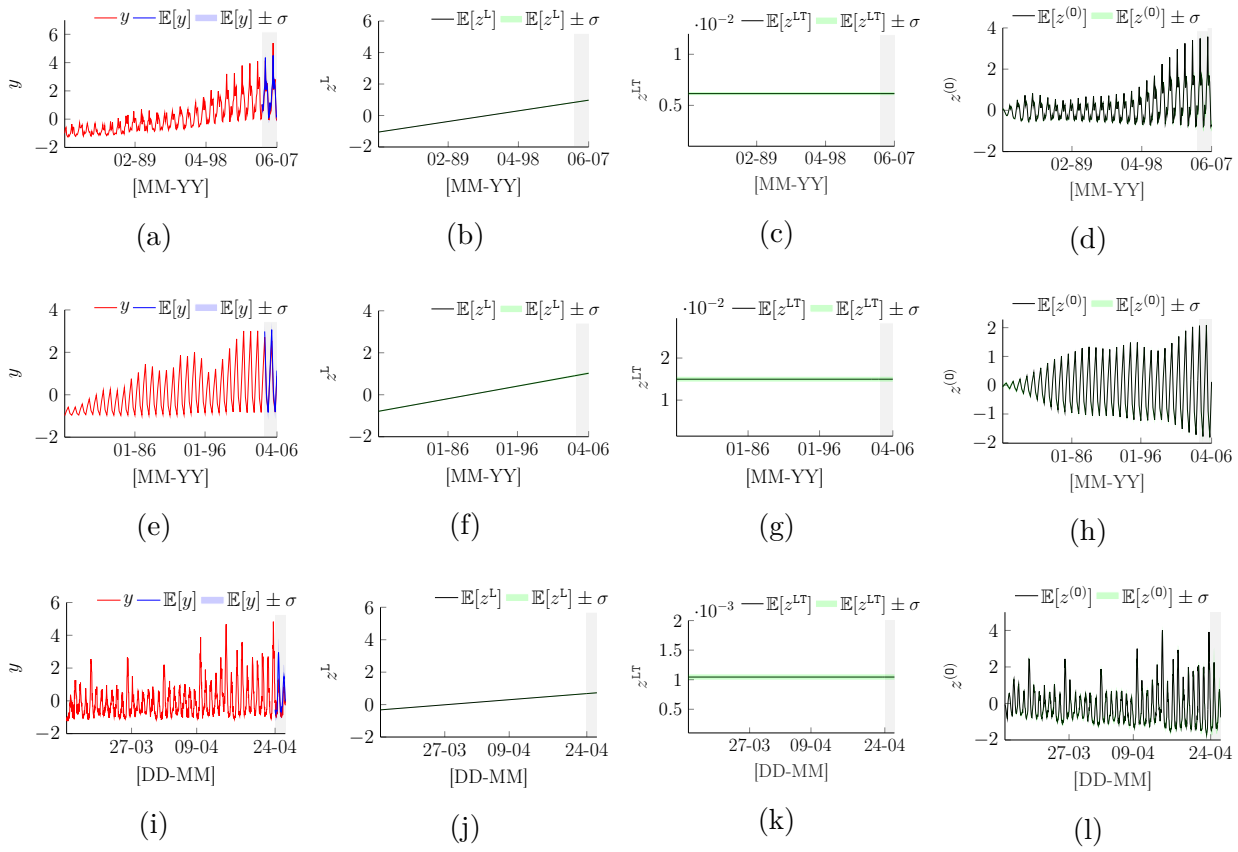


Figure 4.2 An example of interpretable results for several time series. Values are presented in standardized space. Red lines present observations, blue lines present test predictions, black lines present components along with $\pm\sigma$ confidence intervals (shade). The grey shaded areas present the forecast period. (a) predictions on test set, (b) level, (c) trend, and (d) seasonality modelled by TAGI-LSTM for time series #32 of the Tourism (monthly) dataset. (e) predictions on test set, (f) level, (g) trend, and (h) seasonality modelled by TAGI-LSTM for time series #33 of the Tourism (quarterly) dataset. (i) predictions on test set, (j) level, (k) trend, and (l) seasonality modelled by TAGI-LSTM for time series #375 of the M4 (hourly) dataset. In (a), (e) and (i), the $\pm\sigma$ regions contain both the epistemic (parameter's) uncertainties obtained from the prior predictive distributions of the hidden states as well as the aleatory uncertainties associated with the error term's variance σ_V^2. Zoom-in figures for (a), (e) and (i) are provided in Appendix D.

4.3.2 Experiments – Nonstationary Time Series Benchmark Dataset

This experiment 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) [110] and M4 (hourly) [111] datasets, containing either linear or complex nonlinear trends. These datasets are intended to demonstrate that the hybrid TAGI-LSTM/SSM model is well-suited to handle trended data without the need for preprocessing. 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 local trend (LT), and an exponential smoothing component, $\boldsymbol{z}^{\mathtt{B}} = [z^{\mathtt{L}}\ z^{\mathtt{LT}}\ z^{\mathtt{E}}\ z^{\alpha}\ z^{\mathtt{V}}]^{\intercal}$, and the pattern hidden state $z^{(\mathtt{0})}$ models recurrent patterns. The model matrices are given as

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{Q} = \text{diag}([0\ 0\ 0\ 0\ \sigma_V^2\ 0]), \mathbf{F} = [1\ 0\ 1\ 0\ 0\ 0\ 1].$$

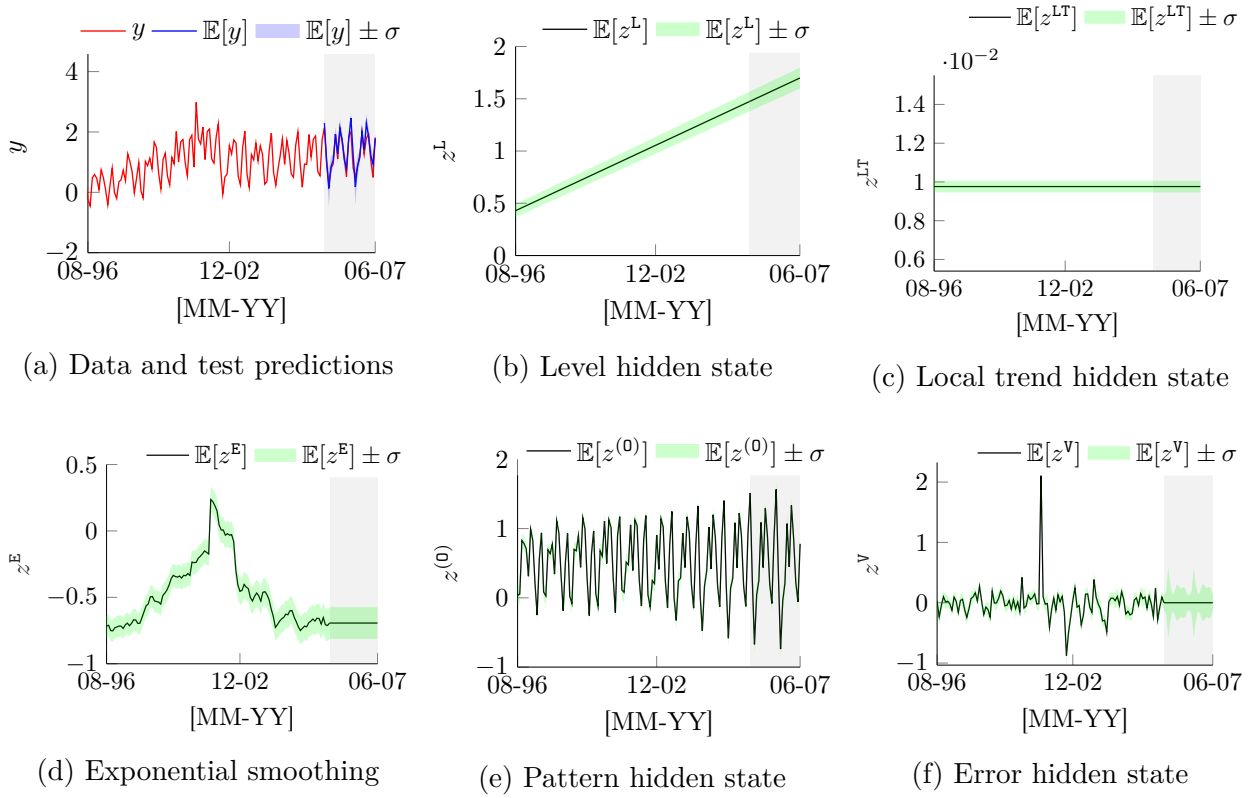
For the time features included in the covariate vector $\boldsymbol{x}_t$, we include hour-of-the-day and day-of-the-week for the hourly dataset, month-of-the-year for the monthly dataset, and quarter-of-the-year for the quarterly dataset. We build a separate model for each time series where all models share the same network architecture as detailed in Table 4.2.

Table 4.2 Architecture and hyperparameters for TAGI-LSTM models used in our experiments. d, m, and q are the abbreviations for day, month, and quarter, respectively. The AGVI method is presented in [6–8].

Dataset	Tourism (monthly)	Tourism (quarterly)	M4 (hourly)
# LSTM layer	1	1	1
# LSTM nodes	50	50	50
Batch size	1	1	1
Lookback window's length W	12	4	168
σ_V	by AGVI	by AGVI	by AGVI
Validation set	12m	4q	1d
Test set	24m	8q	2d

The standard deviation for the process noise σ_V is learnt using the AGVI method [7, 8, 13]. We train our models on multiple epochs where the initial hidden states $\boldsymbol{\mu}_{\emptyset}^{e+1}$ and $\boldsymbol{\Sigma}_{\emptyset}^{e+1}$ at the $e+1^{th}$ epoch are the smoothed estimates $\boldsymbol{\mu}_{\emptyset|\mathtt{T}}^{e}$ and $\boldsymbol{\Sigma}_{\emptyset|\mathtt{T}}^{e}$ at the e^{th} epoch with $\mathtt{T}$ being the last training time.

Figures 4.3 and 4.4 show two examples of predictions on the test set, the hidden states, as well as their associated uncertainties for the time series #30 and #36 from the Tourism (monthly) dataset. These results confirm that our model can separate the linear long-term pattern which is captured by the level hidden state $z^{\mathtt{L}}$, the nonlinear long-term pattern which is captured by the exponential smoothing hidden state $z^{\mathtt{E}}$, and the recurrent patterns modelled by the TAGI-LSTM.



(a) Data and test predictions

(b) Level hidden state

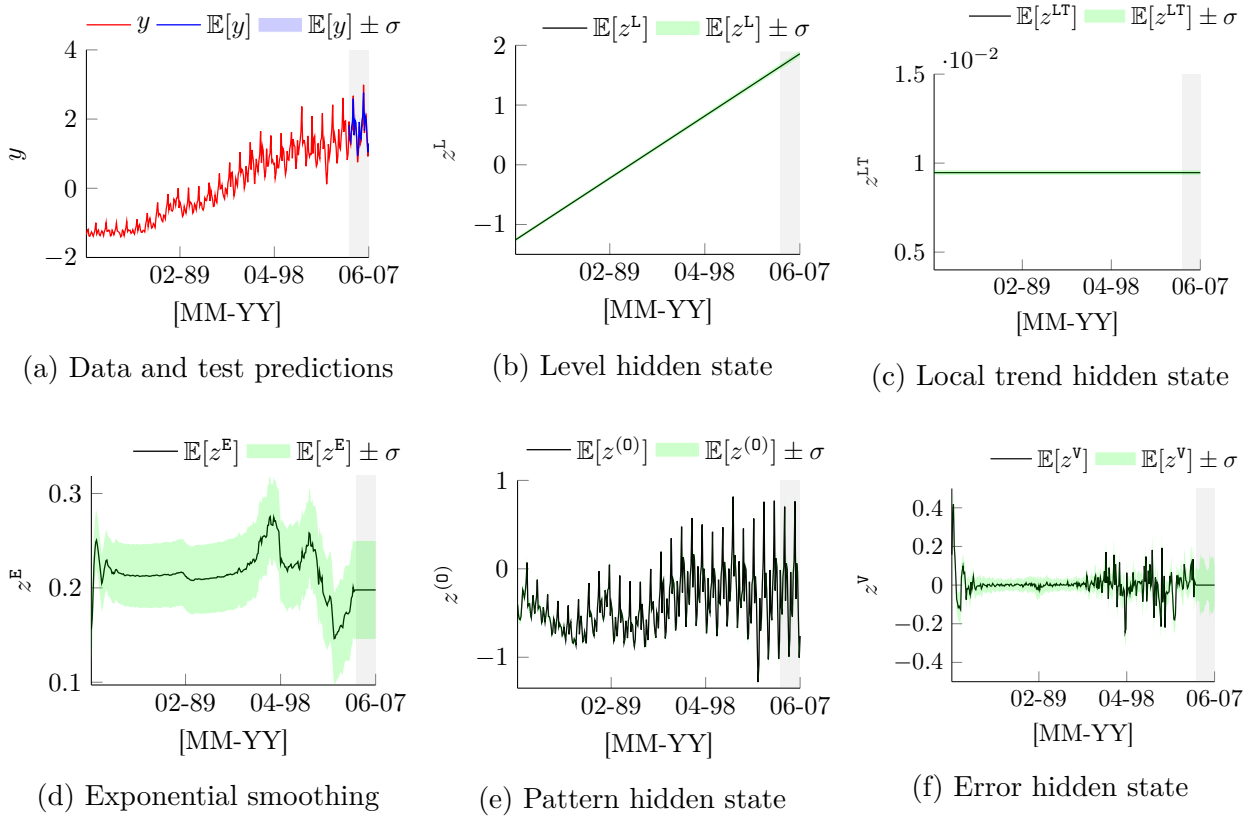
(c) Local trend hidden state

(d) Exponential smoothing

(e) Pattern hidden state

(f) Error hidden state

Figure 4.3 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 $\pm\sigma$ confidence intervals (shade). The grey shaded areas present the forecast period. The $\pm\sigma$ regions in (a) contain both the epistemic (parameter's) uncertainties obtained from the prior predictive distributions of the hidden states as well as the aleatory uncertainties associated with the error term's variance σ_V^2.



(a) Data and test predictions (b) Level hidden state (c) Local trend hidden state

(d) Exponential smoothing (e) Pattern hidden state (f) Error hidden state

Figure 4.4 An example of interpretable results for the time series #36 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 $\pm\sigma$ confidence intervals (shade). The grey shaded areas present the forecast period. The $\pm\sigma$ regions in (a) contain both the epistemic (parameter's) uncertainties obtained from the prior predictive distributions of the hidden states as well as the aleatory uncertainties associated with the error term's variance σ_V^2.

We compare our method with the statistical methods ARIMA [112] and ETS [113]; the hybrid methods DeepState [4] and ES-RNN [91]; and the pure neural networks DeepAR [3] and N-Beats [9]. 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 [114]. 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 4.3 shows 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 builds a separate 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 that of our method. Note that these methods use more advanced neural network architecture such as dilated LSTMs with attention mechanism and residual connections. In addition, ES-RNN and N-Beats can provide interpretable components, but they are unable to provide the associated uncertainties.

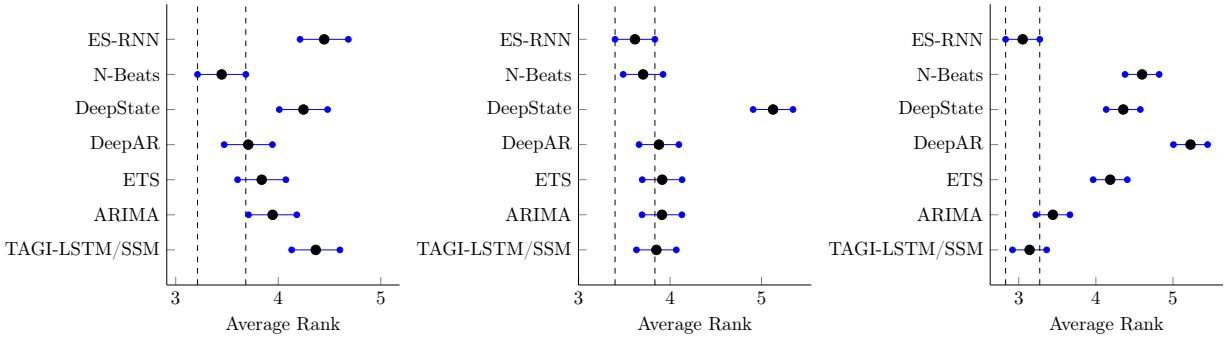
Table 4.3 $p50$ and $p90$-loss performances on the Tourism (monthly and quarterly) and M4 (hourly) test sets. The results for N-Beats are obtained from [9], the one for ES-RNN is calculated by us from the submission downloaded from the M4 GitHub repository [10], and the results for other baseline methods are obtained from [4]. 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**

In order to extend the comparisons, we re-obtained the results from the benchmark methods open-source packages. For the M4 hourly dataset, we do not need to rerun the ARIMA, ETS and ES-RNN models because the original predictions by the authors are already available in the M4 repository [10]. The results for ARIMA and ETS models are obtained using the R's *forecast* package [115], the results for DeepState, DeepAR and N-Beats are obtained using the *gluonTS* library [116], and the results for ES-RNN are obtained from the *ESRNN* library [117]. For the DeepAR and DeepState models, we use the same LSTM's architecture as presented in the original DeepAR paper [3]. For the N-Beats models, the results presented in the original paper [9] are averaged from 180 models. Here, given the prohibitive computational cost, we instead obtained results from 18 models (6 lookback lengths $(2H, \cdots, 7H)$ $\times$ 1 loss function (MASE) $\times$ 3 random seeds). Having obtained the predictions for all methods,

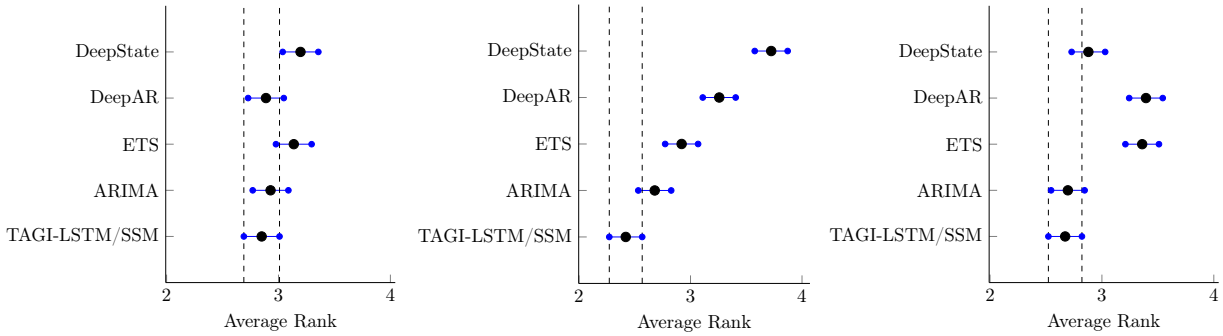
we provide further comparisons on the RMSE and MASE metrics as presented in Table 4.4.

Figures 4.5 and 4.6 shows the result for the *multiple comparisons with the best* (MCB) test [118] using the $p50$ and $p90$ metrics. This test computes the average ranks of the forecasting methods, where the absence of overlap in the intervals for two methods indicates a statistically significant difference in performance. 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, while providing interpretable results along with quantifying the epistemic and aleatory uncertainties.



(a) Tourism (monthly) (b) Tourism (quarterly) (c) M4 (hourly)

Figure 4.5 Average ranks and 95% confidence intervals for all methods over all time series in each dataset based on the multiple comparisons with the best (MCB) test using the $p50$ metric. Dashed lines present 95% confidence intervals for the best method.



(a) Tourism (monthly) (b) Tourism (quarterly) (c) M4 (hourly)

Figure 4.6 Average ranks and 95% confidence intervals for all methods over all time series in each dataset based on the multiple comparisons with the best (MCB) test using $p90$ metric. Dashed lines present 95% confidence intervals for the best method.

Table 4.4 $p50$, $p90$, RMSE, and MASE performances on the Tourism (monthly and quarterly) and M4 (hourly) test sets using our re-run results. Re-run results are obtained from three independent runs except for the ARIMA and ETS. Bold fonts indicate the best results. * indicates that results are obtained by using the predictions provided by the authors downloaded from the M4 repository [10].

	Tourism (Monthly)				Tourism (Quaterly)				M4 (Hourly)			
	$p50$	$p90$	RMSE	MASE	$p50$	$p90$	RMSE	MASE	$p50$	$p90$	RMSE	MASE
ARIMA	0.101	0.058	8806	1.487	0.125	0.062	104362	1.586	0.050*	0.024*	2208*	0.943*
ETS	0.105	0.063	9448	1.526	0.093	0.050	72954	1.592	0.054*	0.031*	2047*	1.824*
DeepState	0.106	0.066	8406	1.531	0.146	0.098	102108	2.060	0.054	0.022	2039	1.455
ES-RNN	0.118	–	8262	1.649	0.084	–	66911	**1.506**	**0.039***	–	**1407***	**0.893***
DeepAR	0.097	**0.051**	**6837**	1.450	0.076	0.044	53379	1.550	0.053	0.023	2296	2.758
N-Beats	**0.095**	–	6924	**1.427**	0.083	–	59215	1.507	0.042	–	1475	1.687
TAGI-LSTM/SSM	0.102	0.053	8234	1.619	**0.073**	**0.041**	**43095**	1.583	0.042	**0.021**	1651	0.925

4.3.3 Experiment – Structural Health Monitoring Dataset

In this section, we evaluate the capacity of the proposed TAGI-LSTM/SSM model on three structural health monitoring (SHM) time series including (1) the water infiltration rate from a concrete dam in Canada, (2) the traffic load on a bridge in the UK, and (3) the radial displacement of a double curvature arch dam in France. The data is chosen such that each case study shows the ability of the TAGI-LSTM to model different types of recurrent patterns and dependencies. The summary characteristics for these datasets are presented in Table 4.5. For all case-studies, the baseline hidden state vector is chosen according to the data's trend pattern, and a single TAGI-LSTM component is used to model the recurrent patterns. We refer to recurrent patterns as a general term for a behavior that repeats over time, including both periodic patterns which occur at a fixed interval as well as non-periodic patterns which repeat at irregular intervals. The ability to model the recurrent patterns caused by environmental effects is crucial for accurate time series prediction in the context of SHM. The models' definition only requires choosing the neural network hyperparameters such as the number of hidden units and layers, and the lookback length, which is a common task when using RNN-based models. The network architecture as well as the hyperparameters used for all case-studies are presented in Table 4.6. Unlike SSMs where the raw data is used, our method uses standardized data with zero mean and unit standard deviation. After obtaining the predictions in the transformed space, they are de-standardized to the original space to obtain the final interpretable forecasts.

Table 4.5 Descriptions of datasets used in case studies using SHM dataset.

Dataset	Water infiltration rate	Traffic load	Dam's displacement
Data acquisition interval	Daily	30-minute	1.5-week average
Missing data	No	No	Yes
# total data point	2289	2409	687
# training	1253	1361	585
# validation	365	288	43
# test	671	760	102

Table 4.6 Architecture and hyperparameters for the models used in our experiments.

Dataset	Water infiltration rate	Traffic load	Dam's displacement
# LSTM layer	2	2	2
# LSTM nodes	30	30	30
σ_V	0.6	0.2	0.2
Grid for searching σ_V	$\{0.2, 0.4, 0.6, 0.8\}$	$\{0.2, 0.4, 0.6, 0.8\}$	$\{0.2, 0.4, 0.6, 0.8\}$
σ_W	0	0	0
Lookback window `W`	365	336	35
Grid for searching `W`	n/a	n/a	$\{14, 35, 56, 70\}$
Lookback window `M` (covariates)	n/a	n/a	21
Grid for searching `M`	n/a	n/a	$\{7, 21, 35, 49, 70\}$

SHM dataset #1 – Water infiltration

In this first case study, we use TAGI-LSTM to model an annual non-harmonic periodic pattern whose amplitude increases over time. The data represents the daily water infiltration rate from a concrete dam in Canada. The water leakage flow rate is typically used by dam engineers as an indicator for structural anomalies because water from the reservoir penetrates the dam body through joints and cracks [119]. The data is available from September 26 2006 to December 31 2012 as presented in Figure 4.7. Following [38], we use the data before March 2011 for training and we evaluate the predictive performance on the test set from March 2011 to December 2012 which is presented by the grey shaded region in Figure 4.7.

We employ a local trend component which consists of a level (`L`) and a local trend (`LT`) hidden state to model the linear trend displayed in the data, whereas the TAGI-LSTM presented in Section 3.2 is used to model the increasing amplitude of the periodic pattern. The hidden state vector at time t is defined as $\boldsymbol{z}_t = [z^{\mathtt{B}}\ z^{(\mathtt{0})}] = [z^{\mathtt{L}}\ z^{\mathtt{LT}}\ z^{(\mathtt{0})}]_t^\intercal$, and the model matrices are given in Equation 4.9. This case study considers a univariate time series problem so that making a prediction at a time t only relies on the dependencies within the time series. For modeling these dependencies, the TAGI-LSTM's inputs include a lookback window of `W`

previous TAGI-LSTM's outputs, $\boldsymbol{x}_t = [z_{t-\mathtt{W}}^{(\mathtt{O})}, \cdots, z_{t-1}^{(\mathtt{O})}]$, so that the neural network identifies the input-output dependencies without a manual setup. A previous study [25] suggests using a lookback window that covers 1-1.25 times the period, hence, we use $\mathtt{W} = 365$ (days) in this case study. In addition, we also include the day-of-the-year covariate in the TAGI-LSTM's input vector as the data displays an annual periodic pattern.

It is possible to model the amplitude-increasing periodic pattern using the existing BDLM's periodic components, but extensive manual setups are required as shown by [38] who have used a trend multiplicative model which consists of a linear trend and a periodic component having a constant amplitude. The periodic pattern's increasing amplitude can be modelled by multiplying these two components. The limitation of this approach is that one needs to identify whether the amplitude increases linearly or nonlinearly over time in order to choose the correct trend component for the multiplicative model. Furthermore, the parameters defining the periodic component must be obtained using an optimization technique which is more computational demanding than estimating the TAGI-LSTM's parameters analytically.

Figure 4.8 presents the results from our model including the test predictions, the level, trend hidden states, as well as the recurrent pattern modelled by the TAGI-LSTM. It shows that it can decompose data into interpretable components, i.e., the level hidden state captures the long-term pattern, the local trend hidden state models the rate of change of the level, and the TAGI-LSTM models the recurrent pattern whose amplitude varies over time, while requiring a minimal manual setup. We compare the test MSE and log-likelihood performances of our model with the BDLM model by [38] along with deterministic LSTM and GRU models. Table 4.7 shows that our model provides a better MSE value, whereas the BDLM model provides a better log-likelihood. The deterministic LSTM and GRU models provide point forecasts so that we only report the MSE metric.

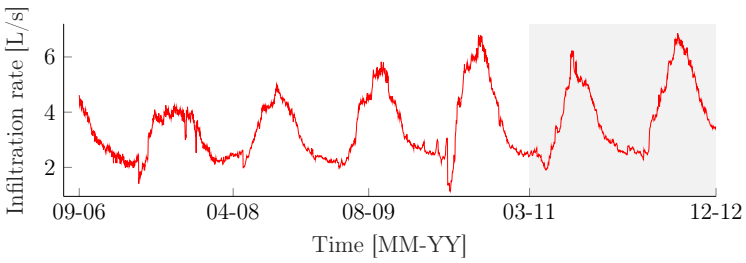
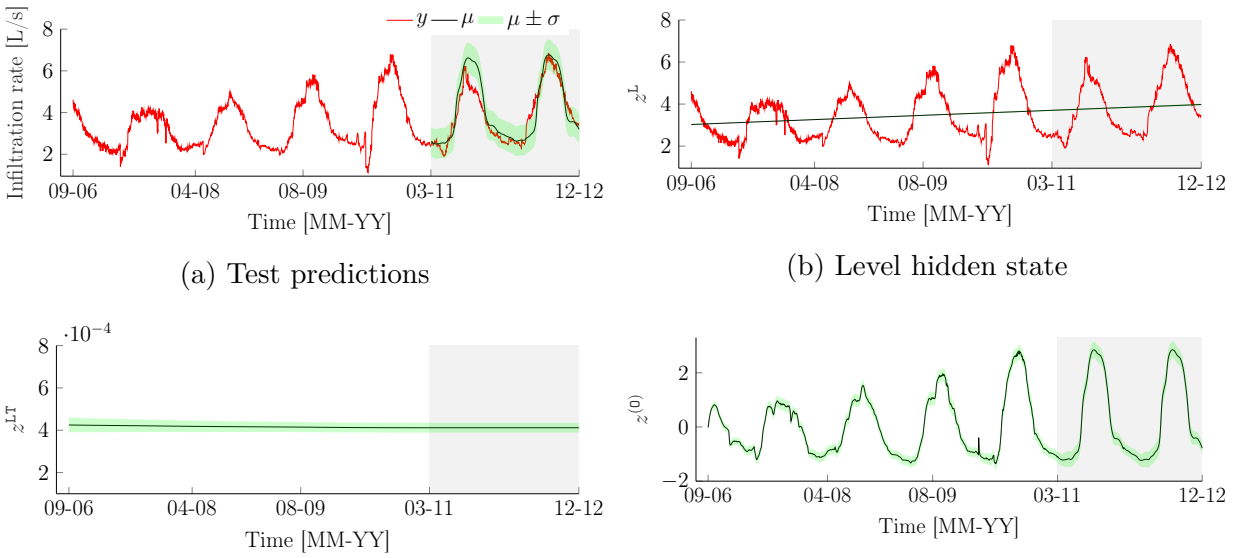


Figure 4.7 Water infiltration rate from a concrete dam in Canada. The grey shaded area presents the forecast period.



(a) Test predictions

(b) Level hidden state

(c) Local trend hidden state

(d) Recurrent pattern modelled by TAGI-LSTM

Figure 4.8 Results from our model for the water infiltration rate dataset including the test predictions, the level and local trend hidden states, and the recurrent pattern. The grey shaded area presents the forecast period. The $\pm\sigma$ regions in (a) contain both the epistemic (parameter's) uncertainties obtained from the prior predictive distributions of the hidden states as well as the aleatory uncertainties associated with the error term's variance σ_V^2.

Table 4.7 Comparison of test performance between TAGI-LSTM/SSM with other models on the water infiltration rate dataset. Results for TAGI-LSTM/SSM, LSTM and GRU models are obtained by averaging the forecasts over ten independent runs. Only the MSE metric is reported for deterministic LSTM and GRU models. Bold fonts indicate the best results.

Model	BDLM	LSTM	GRU	TAGI-LSTM/SSM
MSE	0.28	0.30	0.25	**0.19**
Log-likelihood	**-541.5**	-	-	-625.7

SHM dataset #2 – Traffic load

This case study uses the traffic-load data for the Tamar bridge in southwest England [120,121]. The data is available from September 01 to October 21 2007 as presented in Figure 4.9, and has an acquisition interval of 30 minutes. The training data consists of the observations before October 06, whereas the test data includes measurements from October 06 to 21. The data experiences both daily and weekly periodic patterns. Through this case study, we showcase how TAGI-LSTM can model this dual periodic pattern without requiring feature engineering.

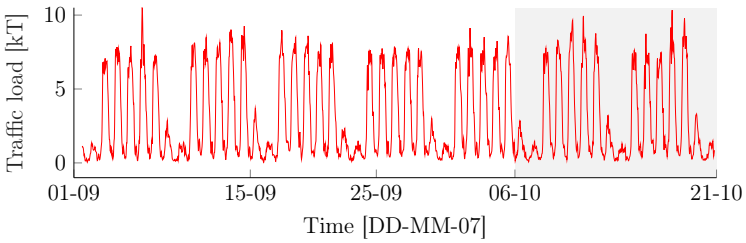


Figure 4.9 Traffic load data on the Tamar bridge in the UK. The grey shaded area presents the forecast period.

We employ a local level (LL) component to model the constant baseline, and a single TAGI-LSTM to capture both the daily and weekly periodic patterns. The vector of hidden states at time t is defined as $\boldsymbol{z}_t = [z^{\mathtt{L}}\ z^{(\mathtt{0})}]_t^{\intercal}$, and the model matrices are given as

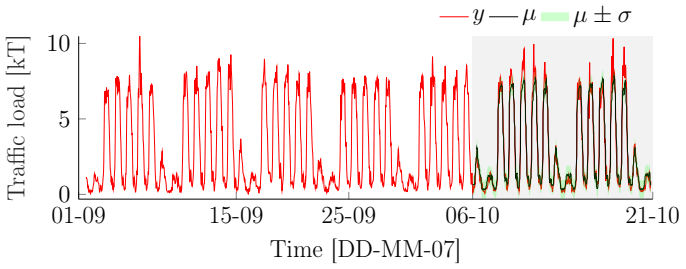
$$\mathbf{A} = 1, \mathbf{Q} = \sigma_W^2, \mathbf{F} = [1\ \ 1]. \tag{4.10}$$

The TAGI-LSTM's inputs include a lookback window of W = 336 previous TAGI-LSTM's outputs, which covers a one week period, for modeling the dependencies within the time series. By contrast, complex feature engineering is required when using the existing BDLM periodic components to capture this dual periodic pattern. To this end, [38] have used two separate Kernel regression KR components [34] where the first component has a periodicity of one week, and the second one has a periodicity of one day. Moreover, [38] have also hand-tuned a first KR component such that non-uniform control points are employed as the first two days of the week have more complex patterns compared to the rest of the week, whereas a second KR component uses uniformly-spaced control points. Taking the product of these two KR components allows modeling the dual periodic pattern present in the data.

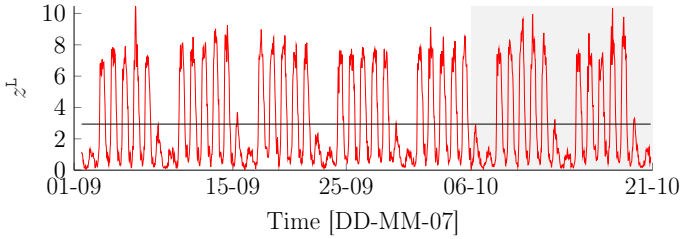
We compare our model with the BDLM-KR and BDLM-DKR models [38] which use a single and a double KR component, along with the deterministic LSTM and GRU models. Table 4.8 shows that all models exhibit a similar performance. In terms of the MSE metric, the deterministic LSTM model demonstrates the best performance, while the BDLM-DKR model performs best in terms of the log-likelihood, whereas our model outperforms the BDLM-KR model. Figure 4.10 illustrates the test predictions, along with the level hidden state and the recurrent pattern obtained from our model. It demonstrates the model's ability to separate the long-term mean represented by the level hidden state and the dual periodic pattern captured by the TAGI-LSTM.

Table 4.8 Comparison of test performance between TAGI-LSTM/SSM and other models on the traffic load dataset. Results for TAGI-LSTM/SSM, LSTM and GRU models are obtained by averaging the forecasts over ten independent runs. Only the MSE metric is reported for deterministic LSTM and GRU models. Bold fonts indicate best results.

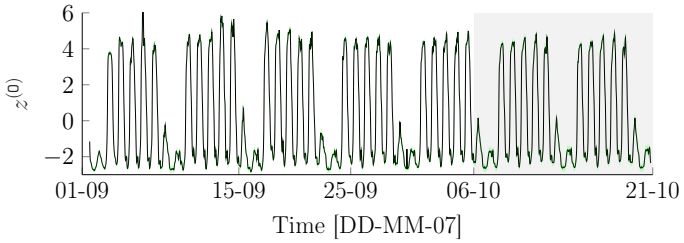
Model	BDLM-`DKR`	BDLM-`KR`	LSTM	GRU	TAGI-LSTM/SSM
MSE	0.30	0.34	**0.28**	0.29	0.32
Log-likelihood	**-616.96**	-656.30	-	-	-649.00



(a) Test prediction



(b) Level hidden state



(c) Recurrent pattern modelled by TAGI-LSTM

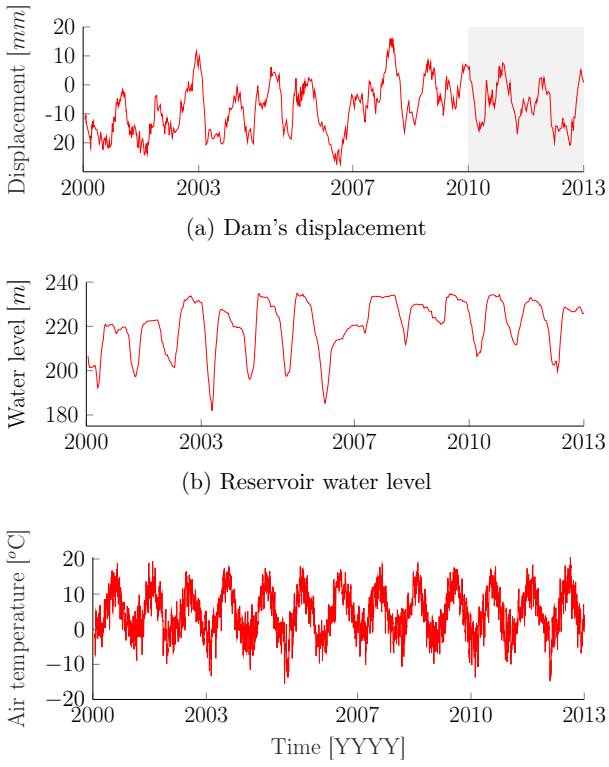
Figure 4.10 Results from our model for the traffic load dataset including the test predictions, the level hidden state, and the recurrent pattern. The grey shaded area presents the forecast period. The $\pm\sigma$ regions in (a) contain both the epistemic (parameter's) uncertainties obtained from the prior predictive distributions of the hidden states as well as the aleatory uncertainties associated with the error term's variance σ_V^2.

SHM dataset #3 – Dam's displacement

This case study uses the CB2 inverted pendulum dataset in a double curvature arch dam located in southern France. This dataset is a part of the 2022 international Benchmark

Workshop organized by the International Commission of Large Dams (ICOLD) [11]. The CB2 pendulum data measures the radial displacement of a dam between its toe and crest, and is available from 2000 to 2012, with an average data acquisition interval of 1.5 week. In addition, two environmental variables including the daily reservoir water level (`WL`) and the daily air temperature (`AT`) are also available for this period as presented in Figure 4.11. The task is to predict the dam's displacement on the test set from 2010 to 2012 as presented by the grey shaded region in Figure 4.11a using the environmental data as explanatory variables. Note that the water level and temperature data from 2010 to 2012 are used when making displacement predictions on the test set. Through this case study, we want to show the ability of the TAGI-LSTM to model the recurrent pattern of the dam's displacement considering its dependencies with the reservoir's water level and the air temperature, while requiring a minimal manual setup.

The nonlinear dependencies between the dam's displacement, the water level and the air temperature are complex, and cannot be seen explicitly from Figure 4.12. Defining these dependencies for building a BDLM model requires expert's knowledge and advanced feature engineering. For this purpose, the water level data is decomposed into a mean-centered com-

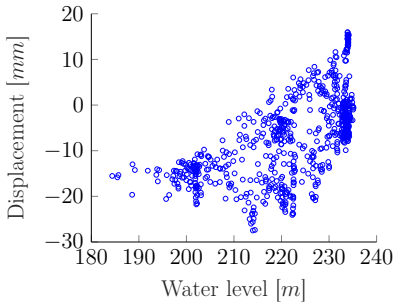


(a) Dam's displacement

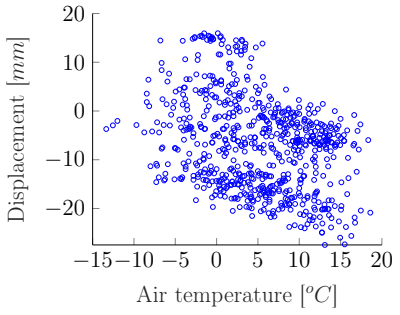
(b) Reservoir water level

(c) Air temperature

Figure 4.11 Data from a dam in southern France



(a) Relationships between the dam's displacement and the reservoir's water level



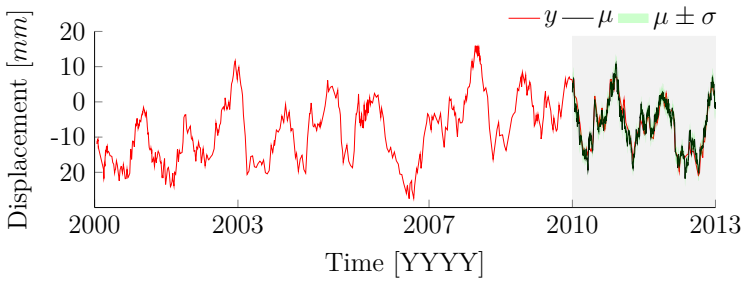
(b) Relationships between the dam's displacement and the air temperature

Figure 4.12 Relationships between the dam's displacement and environmental variables.

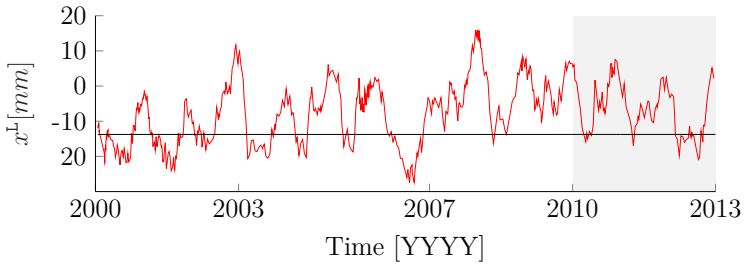
ponent showing the WL's short-term yearly periodic pattern and the WL's average long-term trend [6]. Moreover, the air temperature data was first preprocessed to remove the stationary periodic pattern, then several moving averages of the residuals for $\{1,\ 7,\ 14,\ 28,\ 54\}$ days are considered for taking into account the dam's thermal inertia. Finally in the BLDM model, the dam's displacement was feature-engineered to have nonlinear dependencies with 1) the mean-centered water level, 2) the long-term trend water level, and a linear dependency with the moving averages of the air temperature's residuals. For our model, we only use a local level (LL) component to model the displacement's constant baseline and TAGI-LSTM to capture the displacement's recurrent pattern which depends on the water level and air temperature. The vector of hidden states at time t is defined as $\boldsymbol{z}_t = [z^{\mathtt{L}}\ z^{(\mathtt{O})}]_t^{\intercal}$, and the model matrices are given in Equation 4.10. This case study considers a multivariate time series problem so that making a future prediction relies on both the dependencies within and across time series. For modeling these dependencies, we include in the TAGI-LSTM's input vector a window of $\mathtt{W} = 35$ previous TAGI-LSTM's outputs as well as windows of $\mathtt{M} = 21$ water level and air temperature observations, $\boldsymbol{x}_t = [z_{t-\mathtt{W}}^{(\mathtt{O})}, \cdots, z_{t-1}^{(\mathtt{O})}, y_{t-\mathtt{M}+1}^{\mathtt{WL}}, \cdots, y_t^{\mathtt{WL}}, y_{t-\mathtt{M}+1}^{\mathtt{AT}}, \cdots, y_t^{\mathtt{AT}}]$. These values for W and M have been obtained by a grid-search procedure as presented in Table 4.6. The result is that TAGI-LSTM automatically identifies the output-input dependencies without the need for extensive feature engineering.

Figure 4.13 presents the results from our model including the test predictions, the level hidden state, and the recurrent patterns modelled by the TAGI-LSTM. It shows that our model provides accurate forecasts, whereas the TAGI-LSTM successfully models the recurrent patterns considering the dependencies within and across time series. We compare the test performance of our model with two BDLM models considering nonlinear and linear dependencies with respect to the environmental variables [38]. Table 4.9 shows that our model provides an equivalent performance on the log-likelihood metric, whereas the BDLM model considering nonlinear dependencies provides a better MSE performance. Moreover, the BDLM model considering linear dependencies provides far worse results compared to other models.

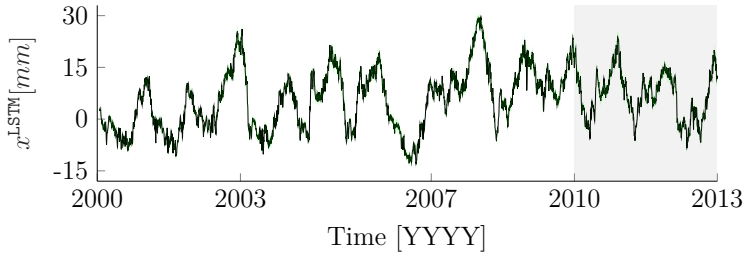
These experimental results confirm that the TAGI-LSTM/SSM method can match the performance of the BDLM model without the need to perform the complex hand-crafted feature engineering required by the latter.



(a) Test prediction



(b) Level hidden state



(c) Recurrent pattern modelled by TAGI-LSTM

Figure 4.13 Results from our model for the displacement dataset including the test predictions, the level hidden state, and the recurrent pattern. The grey shaded area presents the forecast period. The $\pm\sigma$ regions in (a) contain both the epistemic (parameter's) uncertainties obtained from the prior predictive distributions of the hidden states as well as the aleatory uncertainties associated with the error term's variance σ_V^2.

Table 4.9 Comparison of test performance between TAGI-LSTM/SSM and other models on the CB2 displacement dataset. Results for TAGI-LSTM/SSM are obtained by averaging the forecasts over ten independent runs. Bold fonts indicate best results.

Model	BDLM (Nonlinear dependencies)	BDLM (Linear dependencies)	TAGI-LSTM/SSM
MSE	**3.34**	44.7	3.48
Log-likelihood	-216.8	-470.8	**-216.7**

4.4 Conclusion

In this chapter, we proposed the probabilistic coupling between the TAGI-LSTM neural network and states-space models (SSMs), whereas the coupling between TAGI-GRU and SSM can be done analogously. We have demonstrated how we can couple TAGI-LSTM with the existing SSM's level and trend components, as well as a parameter-free exponential smoothing one in order to analyze time series containing either a constant, linear or complex long-term trend patterns. The experimental results on the quarterly and monthly Tourism, and hourly M4 datasets have shown that our hybrid models match or exceed the performance of other models such as ARIMA, ETS, DeepAR and DeepState. The results on three SHM datasets showed that the proposed method can provide interpretable results, which are intuitive for engineers, by extracting the long-term irreversible pattern, its rate of change, and the recurrent patterns from the raw data. In addition, our models provide on-par performance compared to the existing BDLM and deterministic LSTM and GRU models while not requiring labour-intensive and time-consuming feature engineering nor optimization, which allow for analyzing SHM data at a larger scale than was previously possible using BDLM. By removing feature engineering, we simplify the procedure for defining models in the sense that they always consist in a baseline component and a generic component modelled by the TAGI-LSTM for automatically capturing various recurrent patterns.

CHAPTER 5 Switching Kalman Filter and Anomaly Detection

5.1 Introduction

As presented in Sections 2.2.3 and 2.5, there exists many types of anomalies and detection methods for time series. However, they are not all suited for the context of structural health monitoring (SHM) as the characteristics of SHM's anomalies differ from those typically addressed by methods in the fields of computer science and signal processing. Up to now, the usage of the Switching Kalman Filter (SKF) for detecting anomalies in SHM as presented in Section 2.2.3 required extensive feature engineering in order to build BDLM models so that the method is limited to analyzing a small number of time series. This chapter presents a new methodology that uses the hybrid TAGI-LSTM/SSM model presented in Section 4 with the SKF method from Section 2.2.3 for detecting anomalies in SHM time series, while operating in non-stationary environments. The objective consists in developing a probabilistic model being able to quantify the probabilities of regime switches online, without having to wait until after years or decades of data are collected, while being able to provide interpretable results for engineers. The proposed model allows overcoming the limitations of the existing SKF because the TAGI-LSTM component can automatically model, with a minimal setup, the dependencies within time series and with explanatory variables. As a result, it eliminates the need for manual feature engineering which enables it to analyze data at a large scale. The contributions of this chapter are to

- Develop a methodology to use the TAGI-LSTM/SSM model in the SKF framework for semi-supervised anomaly detection.
- Validate the proposed anomaly detection method by comparing it with other baseline models on both synthetic and real SHM datasets.

5.2 Methodology

This section introduces a methodology to use the TAGI-LSTM/SSM model presented in Section 4 with the Switching Kalman Filter (SKF) method from Section 2.2.3 for detecting anomalies in SHM data. Consider a discrete switching variable $s_t \in \{1, 2, \cdots, \mathtt{S}\}$, where $\mathtt{S}$ is the number of regimes. At any time, we maintain $\mathtt{S}$ different TAGI-LSTM/SSM models to describe these regimes. Each model associated with the regime $s_{t-1} \equiv i \in \{1, 2, \cdots, \mathtt{S}\}$ at time $t-1$ has its own hidden state vector $\boldsymbol{Z}^i_{t-1|t-1} \sim \mathcal{N}(\boldsymbol{\mu}^i_{t-1|t-1}, \boldsymbol{\Sigma}^i_{t-1|t-1})$ which is defined

as

$$\boldsymbol{z}_{t-1}^{i} = [\boldsymbol{z}^{\mathtt{B},i},\, z^{(\mathtt{0})}]_{t-1}^{\mathsf{T}},$$

where $\boldsymbol{z}_{t-1}^{\mathtt{B},i}$ is the baseline hidden states for the model i, and $z_{t-1}^{(\mathtt{0})}$ is the shared pattern hidden state. Note that each TAGI-LSTM/SSM model has a different baseline hidden state vector $\boldsymbol{z}_{t-1}^{\mathtt{B},i}$, while $z_{t-1}^{(\mathtt{0})}$ is common among all models. This means that the differences between these TAGI-LSTM/SSM models only lie in their irreversible baseline behavior. Each transition from the model associated with the regime $s_{t-1} \equiv i$ at time $t-1$ to a new model associated with the regime $s_t \equiv j$ at time t leads to a different hidden state vector $\boldsymbol{Z}_{t|t}^{(i)j} \sim \mathcal{N}(\boldsymbol{\mu}_{t|t}^{(i)j}, \boldsymbol{\Sigma}_{t|t}^{(i)j})$, where the subscript $(i)j$ denotes the current regime $s_t \equiv j$ given the past one $s_{t-1} \equiv i$.

Following the SKF procedure presented in Section 2.2.3, we need to define the filter and collapse steps. The filter operator that estimates the posterior mean vector $\boldsymbol{\mu}_{t|t}^{(i)j}$ and covariance matrix $\boldsymbol{\Sigma}_{t|t}^{(i)j}$ is given as

$$\begin{aligned}(\boldsymbol{\mu}_{t|t}^{i(j)}, \boldsymbol{\Sigma}_{t|t}^{i(j)}, \boldsymbol{\theta}_{t|t}) \;=\; & \text{SKF-filter}(y_t, \boldsymbol{x}_t, \boldsymbol{\mu}_{t-1|t-1}^{i}, \boldsymbol{\Sigma}_{t-1|t-1}^{i}, \boldsymbol{\theta}_{t-1|t-1}, \cdots \\ & \boldsymbol{H}_{t-1|t-1}, \boldsymbol{C}_{t-1|t-1}, \mathbf{A}^{i(j)}, \mathbf{F}^{i(j)}, \mathbf{Q}^{i(j)}, \mathbf{R}^{i(j)}),\end{aligned} \tag{5.1}$$

where y_t is the observation; $\boldsymbol{x}_t$ is the input covariate vector; $\boldsymbol{H}_{t-1|t-1}$, $\boldsymbol{C}_{t-1|t-1}$ and $\boldsymbol{\theta}_{t-1|t-1}$ are the posteriors for the TAGI-LSTM's hidden, cell states, and parameters at time $t-1$; $\{\mathbf{A}^{i(j)}, \mathbf{F}^{i(j)}, \mathbf{Q}^{i(j)}, \mathbf{R}^{i(j)}\}$ are the model's matrices associated with each transition. This SKF filter operator is defined by following the prediction and update steps of the TAGI-LSTM/SSM model as presented in Algorithm 1. We use the same standard SKF collapse operator from Equation 2.7 as

$$(\boldsymbol{\mu}_{t|t}^{j}, \boldsymbol{\Sigma}_{t|t}^{j}, \pi_{t|t}^{j}) \;=\; \text{Collapse}(\boldsymbol{\mu}_{t|t}^{i(j)}, \boldsymbol{\Sigma}_{t|t}^{i(j)}, \mathbf{M}_{t-1|t}^{i(j)}), \tag{5.2}$$

where $\pi_{t|t}^{j}$ is now the probability of the model associated with the regime $s_t \equiv j$ at time t.

The model matrices $\{\mathbf{A}^{i(j)}, \mathbf{F}^{i(j)}\}$ are defined according to the baseline hidden state vectors $\boldsymbol{z}^{\mathtt{B},i}$, whereas $\mathbf{R}^{i(j)} = \sigma_V^2$ with σ_V being the error term's standard deviation. $\boldsymbol{\mathcal{P}}$ is the set of parameters that defines the model matrices $\{\mathbf{Q}^{i(j)}, \mathbf{Z}\}$, where $\mathbf{Z}$ is the matrix containing the prior probabilities of transitioning from a regime i at time $t-1$ to a regime j at time t as presented in Equation 2.8. Our model has three types of parameters, i.e., the neural network parameters $\boldsymbol{\theta}$ for the TAGI-LSTM, $\boldsymbol{\mathcal{P}}$, and σ_V. Each of them has a different role, i.e., $\boldsymbol{\theta}$ controls the TAGI-LSTM component's ability to model recurrent patterns, $\boldsymbol{\mathcal{P}}$ governs the capacity at identifying regime switches, and σ_V describes the standard deviation of the error that cannot be captured by the model.

Our method can be summarized by

$$\boldsymbol{\pi} \quad = \quad \text{TAGI-SKF}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}, \boldsymbol{\mathcal{P}}, \sigma_V),$$

where the TAGI-SKF$(\cdot)$ operator is defined by applying recursively the SKF filter and collapse operators presented in Equations 5.1 and 5.2, and $\boldsymbol{\pi}$ is the probability of regime switch for the entire time series. The next section elaborates on how to estimate the parameters $\boldsymbol{\theta}$, $\boldsymbol{\mathcal{P}}$ and σ_V.

5.2.1 Parameter Estimation

In Equation 5.1, the TAGI-LSTM's parameters $\boldsymbol{\theta}$ can be updated at each time step, analogously to updating the hidden states in SSM. The problem with learning $\boldsymbol{\theta}$ with this setup is that the capacity at modeling recurrent patterns and at identifying regime switches are inter-connected such that poor estimates for $\boldsymbol{\theta}$ leads to poor regime switch identification and vice versa. If the regime switching detection does not work properly while anomalies occur, the TAGI-LSTM will learn these anomalies instead of letting them being captured by the irreversible baseline hidden states. As a result, the TAGI-SKF model would then not be able to detect regime switches. Instead, we propose a two-step procedure where the first step aims at estimating the parameters $\boldsymbol{\theta}$ and σ_V, and the second one for estimating the parameters $\boldsymbol{\mathcal{P}}$ associated with the regime switching.

Estimating the standard deviation σ_V and the neural network parameters $\boldsymbol{\theta}$

From the entire time series $\{\boldsymbol{x} \in \mathbb{R}^{\mathtt{L}\times\mathtt{X}}, \boldsymbol{y} \in \mathbb{R}^{\mathtt{L}\times 1}\}$, we choose subsets of stationary or trend-stationary data for training $\{\boldsymbol{x}_{\text{train}} \in \mathbb{R}^{\mathtt{T}\times\mathtt{X}}, \boldsymbol{y}_{\text{train}} \in \mathbb{R}^{\mathtt{T}\times 1}\}$ and validation $\{\boldsymbol{x}_{\text{val}} \in \mathbb{R}^{\mathtt{V}\times\mathtt{X}}, \boldsymbol{y}_{\text{val}} \in \mathbb{R}^{\mathtt{V}\times 1}\}$, where $\mathtt{L}$ is the time series' length, $\mathtt{T}$ and $\mathtt{V}$ are the training and validation sets' sizes, and $\mathtt{X}$ is the size of the input covariate vector. The training data is used to learn the parameters $\boldsymbol{\theta}$, whereas the validation data is used to evaluate the out-of-training forecasting performance. Note that the validation data is not used to update $\boldsymbol{\theta}$. Because the training data is chosen such that it is either stationary or trend-stationary and has only one regime, we use a single TAGI-LSTM/SSM for modeling it. We define a set of values $\{\sigma_{V,m}\}_{m=1}^{\mathtt{M}}$ to be searched over for the error term's standard deviation. For each $\sigma_{V,m}$, we train the TAGI-LSTM/SSM model with $\mathtt{E}$ epochs, and identify the optimal parameters $\boldsymbol{\theta}^*$ and σ_V^* associated with the epoch e_m leading to the maximal validation likelihood. This process is detailed in Algorithm 2. Once estimated, $\boldsymbol{\theta}^*$ and σ_V^* are fixed when applying the SKF filter operator

from Equation 5.1 as

$$
\begin{aligned}
(\boldsymbol{\mu}_{t|t}^{i(j)}, \boldsymbol{\Sigma}_{t|t}^{i(j)}) \quad = \quad & \text{SKF-filter}(y_t, \boldsymbol{x}_t, \boldsymbol{\mu}_{t-1|t-1}^{i}, \boldsymbol{\Sigma}_{t-1|t-1}^{i}, \boldsymbol{\theta}^*, \cdots \\
& \boldsymbol{H}_{t-1|t-1}, \boldsymbol{C}_{t-1|t-1}, \mathbf{A}^{i(j)}, \mathbf{F}^{i(j)}, \mathbf{Q}^{i(j)}, \mathbf{R}^{i(j)*}).
\end{aligned}
$$

Algorithm 2: Estimate the optimal values for the TAGI-LSTM's parameters $\boldsymbol{\theta}$ and the error's standard deviation σ_V

Input: Training data $\{\boldsymbol{x}_{\text{train}}, \boldsymbol{y}_{\text{train}}\}$, a pre-defined set of $\{\sigma_{V,m}\}_{m=1}^{\mathtt{M}}$, and a maximum number of epoch $\mathtt{E}$

Output: Optimal parameters $\boldsymbol{\theta}^*$ and σ_V^*

Initialize $\boldsymbol{\theta}^*$ and $\mathcal{L}_{\text{val}}^*$

for $m = 1 : \mathtt{M}$ **do**

  Initialize $\boldsymbol{\theta}^{(0)}, \boldsymbol{\mu}_{\emptyset}^{(0)}, \boldsymbol{\Sigma}_{\emptyset}^{(0)}$ and obtain $\{\mathbf{A}, \mathbf{F}, \mathbf{Q}\}$

  $\mathbf{R} = \sigma_{V,m}^2$

  for $e = 1 : \mathtt{E}$ **do**

    $(\boldsymbol{\theta}^{(e)}, \mathcal{L}_{\text{val}}^{(e)}) = \text{TAGI-LSTM/SSM}(\boldsymbol{y}_{\text{train}}, \boldsymbol{x}_{\text{train}}, \boldsymbol{\theta}^{(e-1)}, \boldsymbol{\mu}_{\emptyset|\mathtt{T}}^{(e-1)}, \boldsymbol{\Sigma}_{\emptyset|\mathtt{T}}^{(e-1)}, \mathbf{A}, \mathbf{F}, \mathbf{Q}, \mathbf{R})$

    if $\mathcal{L}_{val}^{(e)} > \mathcal{L}_{val}^{*}$ **then**

      $\mathcal{L}_{\text{val}}^{*} \leftarrow \mathcal{L}_{\text{val}}^{(e)}$

      $\boldsymbol{\theta}^{*} \leftarrow \boldsymbol{\theta}^{(e)}$

      $\sigma_V^{*} \leftarrow \sigma_{V,m}$

return Optimal parameters $\boldsymbol{\theta}^*$ and σ_V^*

Estimating regime switching parameters $\mathcal{P}$

Datasets with labelled anomalies are rarely available in SHM so that we do not know precisely what magnitude of anomaly we want to detect in order to choose the best values for $\mathcal{P}$. In practice, we would like to detect anomalies that are as small as possible. For this purpose, we rely on simulated anomalies with various magnitudes that are added, one at a time, on top of the anomaly-free training data $\boldsymbol{y}_{\text{train}}$. The objective is to find the optimal values $\mathcal{P}^*$ which allow detecting the smallest anomaly with the highest detection probability for these simulated time series. The assumption is that if the model is capable of detecting small synthetic anomalies, it will also be able to detect anomalies of the same or larger magnitudes for unseen real test data.

The synthetic anomalies used here are defined by two characteristics, i.e., a change in the slope β of the time series's baseline and the anomaly's start time. Figure 5.1a presents an anomaly-free time series $\boldsymbol{y}_{\text{train}}$ having a constant baseline, while Figure 5.1b displays an

abnormal synthetic time series $\boldsymbol{y}^{\beta}$ where its baseline changes. This time series is created by adding an anomaly characterized by a slope β and a start time on top of the anomaly-free time series $\boldsymbol{y}_{\text{train}}$.

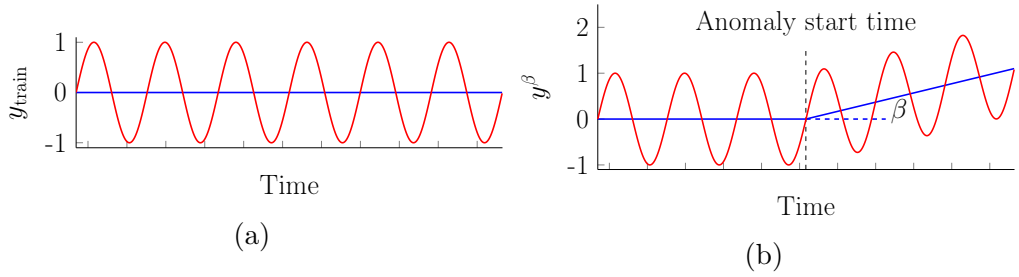


Figure 5.1 Introduction of synthetic anomaly. (a) anomaly-free training time series and (b) time series with synthetic anomaly. The blue line represents the baseline of the data.

We generate multiple abnormal time series with various anomaly's magnitudes by changing the slope β and the anomaly start's time such that

$$\underbrace{\boldsymbol{y}_1^{\beta_1}, \cdots, \boldsymbol{y}_i^{\beta_1}, \cdots, \boldsymbol{y}_{\mathtt{N}}^{\beta_1}}_{\mathtt{N} \text{ time series with slope } \beta_1}, \cdots, \underbrace{\boldsymbol{y}_1^{\beta_b}, \cdots, \boldsymbol{y}_i^{\beta_b}, \cdots, \boldsymbol{y}_{\mathtt{N}}^{\beta_b}}_{\mathtt{N} \text{ time series with slope } \beta_b}, \cdots, \underbrace{\boldsymbol{y}_1^{\beta_{\mathtt{B}}}, \cdots, \boldsymbol{y}_i^{\beta_{\mathtt{B}}}, \cdots, \boldsymbol{y}_{\mathtt{N}}^{\beta_{\mathtt{B}}}}_{\mathtt{N} \text{ time series with slope } \beta_{\mathtt{B}}},$$

where two time series $\boldsymbol{y}_i^{\beta_b}$ and $\boldsymbol{y}_j^{\beta_b}$ have the same slope β_b, but different anomaly's start times drawn from a uniform distribution. In total, the number of abnormal series generated is $\mathtt{B} \times \mathtt{N}$ corresponding to $\mathtt{B}$ values of the anomaly's slopes and $\mathtt{N}$ time series for each slope.

In order to estimate the detection probability p^{β_b} that our model can detect the anomaly with the slope β_b, we use the TAGI-SKF model to analyze all abnormal time series $\{\boldsymbol{y}_1^{\beta_b}, \cdots, \boldsymbol{y}_i^{\beta_b}, \cdots, \boldsymbol{y}_{\mathtt{N}}^{\beta_b}\}$ with the same slope β_b

$$\boldsymbol{\pi} = \text{TAGI-SKF}(\boldsymbol{x}_{\text{train}}, \boldsymbol{y}_i^{\beta_b}, \boldsymbol{\theta}^*, \boldsymbol{\mathcal{P}}, \sigma_V^*).$$

An anomaly is considered as detected if the probability of regime switch $\pi_{t|t} \geq \tau$ after the anomaly's start time, where τ is a predefined threshold. The detection probability p^{β_b} is then approximated by

$$p^{\beta_b} \approx \frac{\#\text{anomaly detected}}{\mathtt{N}}. \tag{5.3}$$

We define a grid of values for each parameter in $\boldsymbol{\mathcal{P}}$, and search for the optimal set of values $\boldsymbol{\mathcal{P}}^*$ that allow to detect the smallest slope β_b with the highest detection probability p^{β_b}. The procedure for grid-searching $\boldsymbol{\mathcal{P}}$ is detailed in Algorithm 3. Algorithm 4 summarizes all the steps of the proposed TAGI-SKF anomaly detection method.

Algorithm 3: Grid-search to obtain optimal parameters $\boldsymbol{\mathcal{P}}^*$

Input: Abnormal time series $\boldsymbol{y}_i^{\beta_b}$ and a pre-defined set $\{\boldsymbol{\mathcal{P}}_k\}_{k=1}^{\mathtt{K}}$
Output: Optimal parameters $\boldsymbol{\mathcal{P}}^*$
for $b = 1 : \mathtt{B}$ **do**
  for $k = 1 : \mathtt{K}$ **do**
    count $= 0$
    for $i = 1 : \mathtt{N}$ **do**
      $\boldsymbol{\pi} = \text{TAGI-SKF}(\boldsymbol{x}_{\text{train}}, \boldsymbol{y}_i^{\beta_b}, \boldsymbol{\theta}^*, \boldsymbol{\mathcal{P}}_k, \sigma_V^*)$
      if $\pi_{t|t} \geq \tau$ after anomaly start time **then**
        count $=$ count $+ 1$
    $p_k^{\beta_b} = \text{count}/\mathtt{N}$
return $\boldsymbol{\mathcal{P}}^* \leftarrow \boldsymbol{\mathcal{P}}_k$ associated with the highest $p_k^{\beta_b}$ and the smallest β_b such that $p_k^{\beta_b} \geq \tau$

Algorithm 4: Procedure for the TAGI-SKF anomaly detection method

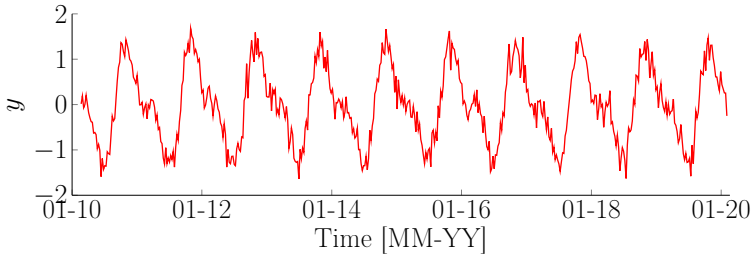
Input: Covariates $\boldsymbol{x}$, observations $\boldsymbol{y}$
Output: probability of regime switches $\boldsymbol{\pi}$
Estimate $\boldsymbol{\theta}$ and σ_V:
1: Choose stationary or trend-stationary training $\{\boldsymbol{x}_{\text{train}}, \boldsymbol{y}_{\text{train}}\}$ and validation $\{\boldsymbol{x}_{\text{val}}, \boldsymbol{y}_{\text{val}}\}$ sets from the entire time series $\{\boldsymbol{x}, \boldsymbol{y}\}$
2: Define TAGI-LSTM/SSM model
3: Obtain the optimal parameters $\boldsymbol{\theta}^*$ and σ_V^* using Algorithm 2
Estimate $\boldsymbol{\mathcal{P}}$:
4: Define two competing TAGI-LSTM/SSM models where one describes a normal regime, and the other models the abnormal one
5: Generate multiple abnormal time series with various anomaly's slopes.
6: Grid-search to obtain the optimal parameters $\boldsymbol{\mathcal{P}}^*$ using Algorithm 3.
Anomaly detection:
7: Return the probability of regime switches
$\boldsymbol{\pi} = \text{TAGI-SKF}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}^*, \boldsymbol{\mathcal{P}}^*, \sigma_V^*)$.

5.3 Experiments

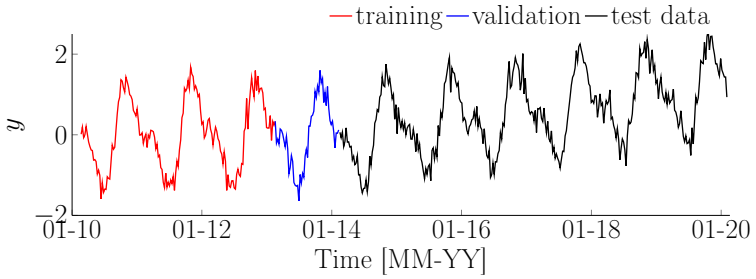
This section compares the anomaly detection performance between the TAGI-SKF and other baseline methods on synthetic and real SHM time series.

5.3.1 Verification with Synthetic Data

This section first uses synthetic time series with known anomalies in order to verify the ability of the TAGI-SKF anomaly detection method presented in Section 5.2. Figure 5.2a presents an anomaly-free time series generated using the function $y_t = \sin(2\pi t/365) + 0.5\sin(\pi t/365) + v_t$, where $v_t : V \sim \mathcal{N}(0, 0.2^2)$, and t is the timestamp. The time series cover a 10-year period from Jan-2010 to Jan-2020 in which the first three years of data from 2010-2013 are used as training set, the subsequent year from 2013-2014 is used as validation set, and the rest of data from 2014 is used as test set. We introduce known synthetic anomalies on top of the anomaly-free test set in order to create multiple abnormal time series with various anomaly's slope β_b, this process will be further detailed later on. Figure 5.2b displays an example of a time series containing a synthetic anomaly during the test period. The task consists in quantifying the detection probability, the false alarm rate as well as the average time required to detect synthetic anomalies for each slope β_b.



(a)



(b)

Figure 5.2 (a) Anomaly-free time series generated from $y_t = \sin(2\pi t/365) + 0.5\sin(\pi t/365) + v_t$, and (b) an example of time series containing a synthetic anomaly during the test period. The shaded grey area presents the period where anomaly is randomly introduced.

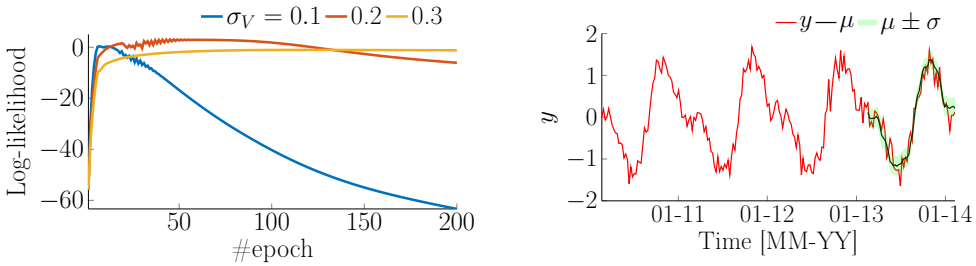
Following the procedure presented in Algorithm 4, we first need to estimate the TAGI-LSTM's parameters $\boldsymbol{\theta}$. We employ a local trend component which consists of a level and trend hidden states to model the baseline so that the hidden state vector is given by $\boldsymbol{z} = [z^{\mathtt{L}},\ z^{\mathtt{LT}},\ z^{(\mathtt{0})}]^\intercal$.

The model matrices associated with $\boldsymbol{z}$ are given by

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \mathbf{Q} = \sigma_W^2 \begin{bmatrix} 1/3 & 1/2 \\ 1/2 & 1 \end{bmatrix}, \mathbf{F} = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}, \mathbf{R} = \sigma_V^2,$$

where $\sigma_W = 0$. The input covariate vector includes a lookback window of $\mathtt{W}$ previous TAGI-LSTM's outputs, $\boldsymbol{x}_t = [z_{t-\mathtt{W}}^{(\mathtt{O})}, \cdots, z_{t-1}^{(\mathtt{O})}]$. The neural network architecture and hyperparameters are presented in Appendix C.

In order to estimate the error variance for this dataset, we grid-search the optimal value σ_V^* from the set $\sigma_V \in \{0.1, 0.2, 0.3\}$. Figure 5.3a shows that the using $\sigma_V^* = 0.2$ gives the largest validation's log-likelihood at the epoch $e = 47$. The parameters $\boldsymbol{\theta}^*$ associated to it will be used for the anomaly detection analyses. Figure 5.3b presents the multi-step-ahead predictions for the validation set using the optimal parameters $\boldsymbol{\theta}^*$.



(a) Log-likelihood

(b) Validation predictions

Figure 5.3 Experiment with synthetic data. Log-likelihood and multi-step-ahead predictions for the validation set.

Following the Step 4 from Algorithm 4, we employ two competing TAGI-LSTM/SSM models consisting in a local trend to represent the normal regime and a local acceleration [20] to represent the abnormal regime. Their hidden states are given by

$$\begin{aligned} \text{Model 1:} \quad \boldsymbol{z}_t^1 &= [\underbrace{z^{\mathtt{L},1}, \; z^{\mathtt{LT},1,}, 0}_{\boldsymbol{z}^{\mathtt{B},1}}, z^{(\mathtt{O})}]_t^\intercal \\ \text{Model 2:} \quad \boldsymbol{z}_t^2 &= [\underbrace{z^{\mathtt{L},2}, \quad z^{\mathtt{LT},2}, z^{\mathtt{LA},2}}_{\boldsymbol{z}^{\mathtt{B},2}}, \; z^{(\mathtt{O})}]_t^\intercal. \end{aligned}$$

Note that in order to keep the same number of hidden states for both models, the acceleration hidden state for the model 1 is forced to be 0 [122]. The model matrices that define the transition from $s_{t-1} = i$ to $s_t = j$ are presented in Figure 5.4.

The switching parameters that define the matrices $\{\mathbf{Q}^{i(j)}, \mathbf{Z}\}$ consists in two values, $\boldsymbol{\mathcal{P}} = \{\sigma_{12}, z_{12}\}$. Following the Step 5 from Algorithm 4, we define three anomaly's slopes $\beta_b \in$

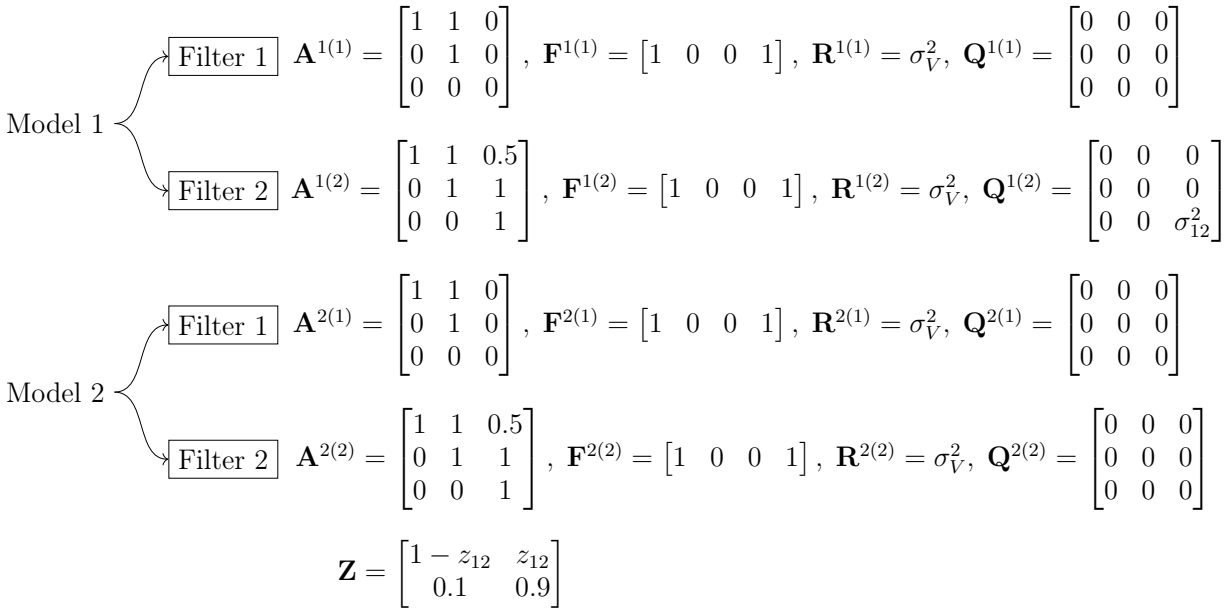


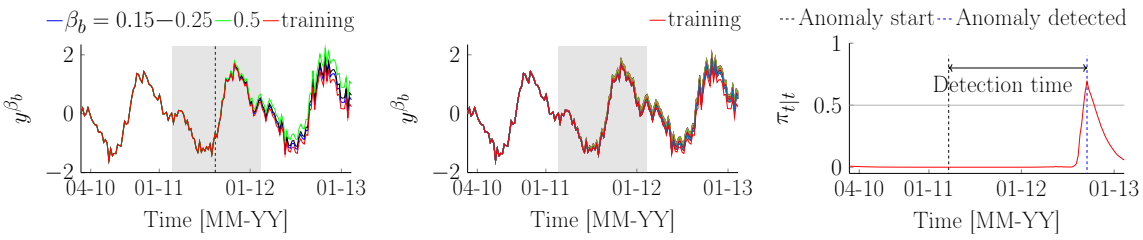
Figure 5.4 Experiment with synthetic data. The TAGI-LSTM/SSM's model matrices for the transitions from $s_{t-1} = i$ to $s_t = j$.

$\{0.5, 0.25, 0.15\}$ [mm/year], and generate $3 \times 50 = 150$ simulated abnormal time series from the anomaly-free training set. Figure 5.5a presents three time series where each contains a single anomaly with different slopes introduced at the same time step, while Figure 5.5b shows 50 time series where each has an anomaly with a same slope of $\beta_b = 0.25$ [mm/year], introduced randomly at different time steps between Jan-2011 and Jan-2012. Figure 5.5c presents an example of the probability of regime switch $\boldsymbol{\pi}$ along with the definition of the detection time for a time series.

Following the Step 6 from Algorithm 4, we define the grid points for each parameter in $\boldsymbol{\mathcal{P}}$, i.e., $\sigma_{12} \in \{10^{-4}, 10^{-5}, 10^{-6}\}$ and $z_{12} \in \{10^{-4}, 10^{-5}, 10^{-6}\}$, and obtain the sets to be evaluated $\boldsymbol{\mathcal{P}}_k, \forall k \in \{1:9\}$ as

$$\begin{aligned}
&\boldsymbol{\mathcal{P}}_1 = \{\sigma_{12} = 10^{-4}, z_{12} = 10^{-4}\},\\
&\boldsymbol{\mathcal{P}}_2 = \{\sigma_{12} = 10^{-4}, z_{12} = 10^{-5}\},\\
&\dots\\
&\boldsymbol{\mathcal{P}}_9 = \{\sigma_{12} = 10^{-6}, z_{12} = 10^{-6}\}.
\end{aligned}$$

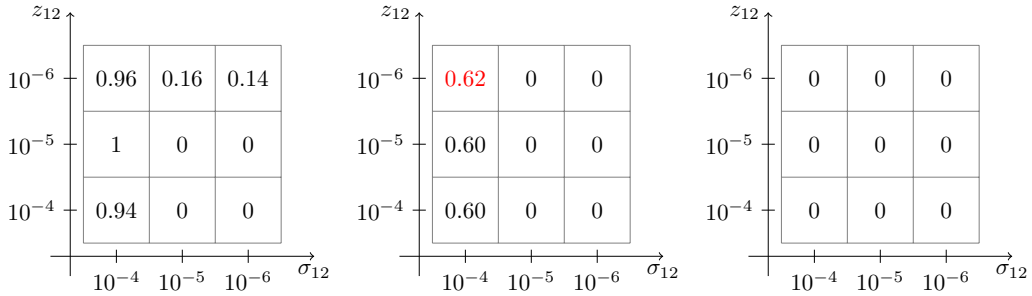
For each $\boldsymbol{\mathcal{P}}_k$, we quantify the detection probability p^{β_b} associated with anomalies having different slopes β_b. Figure 5.6 presents p_b^{β} for each anomaly's slope where we find that the smallest slope can be detected is $\beta_b = 0.25$ [mm/year] with $p^{\beta_b} = 0.62$, and the corresponding optimal parameters are $\boldsymbol{\mathcal{P}}^* \equiv \boldsymbol{\mathcal{P}}_3 = \{\sigma_{12} = 10^{-4}, z_{12} = 10^{-6}\}$. With these parameters, the model is also capable of detecting larger anomalies, i.e., the slope $\beta_b = 0.5$ [mm/year] with



(a) (b) (c)

Figure 5.5 (a) Three time series with anomalies of different slopes introduced at the same time step, (b) 50 time series with anomalies of the same slope $\beta_b = 0.25$ [mm/year] but introduced at different time steps, and (c) the probability of anomaly $\pi_{t|t}$ and detection time for one synthetic time series. The dashed line presents the anomaly's start time, the shaded grey area presents the period where anomalies are randomly introduced with different start times.

$p^{\beta_b} = 0.96$, but it cannot detect the anomalies associated with a smaller slope $\beta_b = 0.15$ [mm/year] as indicated by $p^{\beta_b} = 0$.



(a) $\beta_b = 0.5$ [mm/year] (b) $\beta_b = 0.25$ [mm/year] (c) $\beta_b = 0.15$ [mm/year]

Figure 5.6 Detection probability p^{β_b} for detecting anomalies with different slope β_b in the simulated time series for models with different sets $\boldsymbol{\mathcal{P}}_k$.

Having obtained the optimal parameters $\boldsymbol{\theta}^*$, σ_V^* and $\boldsymbol{\mathcal{P}}^*$, we evaluate the model's ability to detect synthetic anomalies for the unseen test set. For that purpose, anomalies with different slopes β_b ranging from 0.1 to 0.5 [mm/year] are added on top of the anomaly-free time series during the test period in order to create multiple abnormal time series. For each slope β_b, we generate $\mathtt{N} = 50$ time series where each contains a single anomaly with a start time following a uniform distribution over 2014 to 2017. Figure 5.2b presents an example of time series displaying a synthetic anomaly during the test period.

We compare the performance of our model with five anomaly detection methods presented in Section 2.5, i.e., kNN, Matrix Profile (MP), Prophet, Autoencoder, and Variational Autoen-

coder (VAE). Specifically, we compare these methods in terms of the detection probability, average detection time, and false alarm rate per ten years. The detection probability has been defined in Equation 5.3, whereas the detection time refers to the average time required to detect the synthetic anomalies with a slope β_b presented in the multiple abnormal time series. All the data points after the anomaly's start time are considered as abnormal data so that if multiple alarms are triggered within this period, we only consider the detection time for the first alarm. As for the false alarm rate, we count the number of alarms triggered before the anomaly's start time, and divide it by the duration of time before the anomaly's start time for synthetic time series having a same slope.

Among the baseline methods, the Prophet, kNN, and MP are intended to be used in an offline setup employing the entire time series, either for training (Prophet) or for calculating the distances (kNN and MP). By contrast, our method, the Autoencoder, and VAE can operate online after having learnt their parameters using the anomaly-free training set. They process the entire time series without requiring to retrain the models. Table 5.1 compares these methods among several characteristics required for SHM applications.

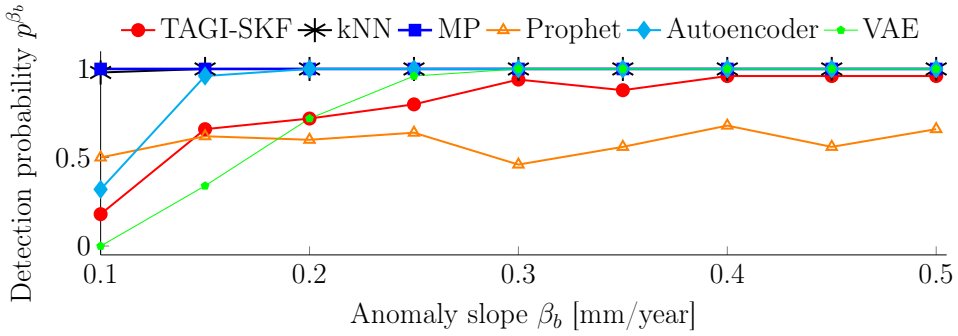
Table 5.1 Characteristics comparison of anomaly detection methods for SHM applications. N/a for the MP and kNN means that no training is required for these methods.

Characteristics	TAGI-SKF	Prophet	MP	kNN	Autoencoder	VAE
Can be trained on data including anomalies	No	Yes	n/a	n/a	No	No
Can be trained on trend-non-stationary data	Yes	Yes	n/a	n/a	Yes	Yes
Retraining required when processing data online	No	Yes	n/a	n/a	No	No
Continuous adaptation to multiple anomalies	Yes	Yes	No	Yes	No	No
Consider explanatory variables	Yes	No	No	No	Yes	Yes

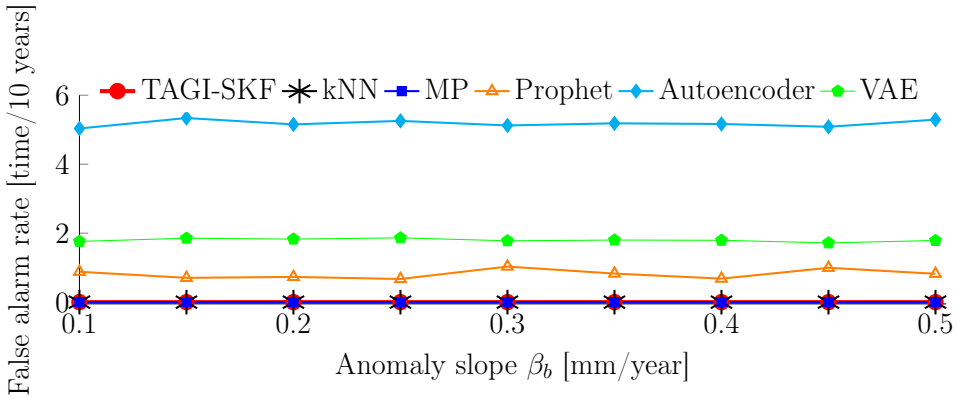
SHM requires to have a tight control over the false alarm rate because evaluating structures after an alarm is expensive. Therefore, the hyperparameters for the baseline models are tuned in order to reach zero false alarm over the training period. However, when this criteria could not be fulfilled for a method, we choose the hyperparameters that leads to the smallest false alarm rate. The detailed hyperparameters-tuning procedure for these methods are given in Appendix E.

Figure 5.7a presents the detection probability p^{β_b}, whereas Figure 5.7b shows the false alarm rate per ten years, while Figure 5.7c presents the average detection time along with its $\pm\sigma$ confidence region for all methods. In general, as the anomaly's slope increases, the detection probability increases, while the average detection time decreases. By contrast, the false alarm rates are almost the same across anomaly's slopes.

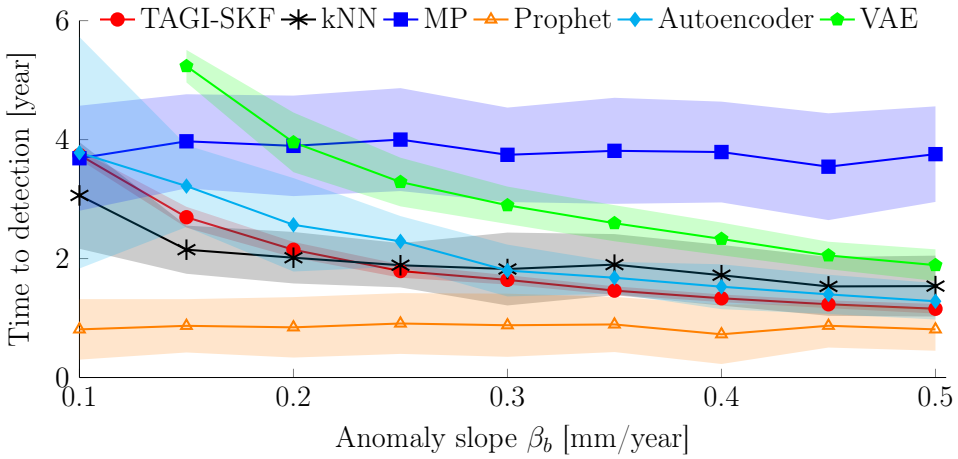
Figure 5.7a shows that our method provides a detection probability $p^{\beta_b} > 0.5$ for all anomaly's



(a)



(b)



(c)

Figure 5.7 Comparison of (a) detection probability p^{β_b}, (b) false alarm rate per ten years, and (c) average detection time among all methods for different anomalies of slope β_b. The shaded area presents the associated $\pm\sigma$ confidence region.

slopes except for $\beta_b = 0.1$ [mm/year], while providing no false alarms. Compared to our method, the kNN provides higher detection probabilities, similar detection times, and an equal robustness toward false alarms. On the other hand, the MP also provides higher detection probabilities, but it takes a much longer time to detect anomalies compared our method. Moreover, the MP's detection time does not decrease as the slope's value increases.

The Autoencoder and VAE provide higher detection probabilities p^{β_b} compared to our method, but they both provide higher detection times and false alarm rates. At term, we will face the challenge of dealing with data from a network of more than 20000 sensors spread across hundreds of dams in the province of Quebec, Canada. The Autoencoder and VAE methods have about 5 and 1.8 false alarms per ten years, respectively, which would trigger 27 and 10 false alarms per day at the scale of the network. These false alarm rates are too high for the large-scale deployment of SHM system. The Prophet method provides the smallest detection probabilities compared to other methods, but it is able to provides the shortest detection times, while having less than one false alarm per ten years.

On this relatively simple synthetic data, the kNN and Prophet exhibit competitive performance compared to our method. In contrast, the MP, Autoencoder, and VAE models demonstrate a poor performance. In the next section, we will show how our method outperforms the kNN and Prophet on complex real SHM datasets.

5.3.2 Experiment – Structural Health Monitoring Dataset

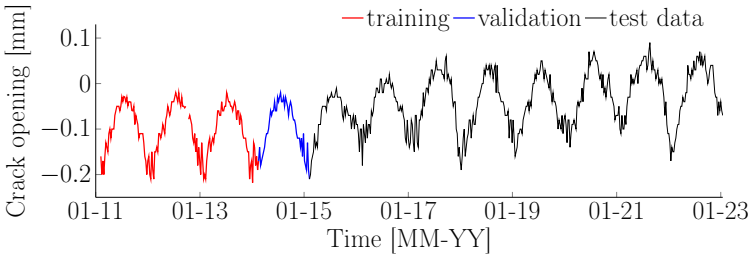
This section compares the anomaly detection performance between TAGI-SKF and other methods on three SHM time series obtained from dams in Canada.

Case study #1

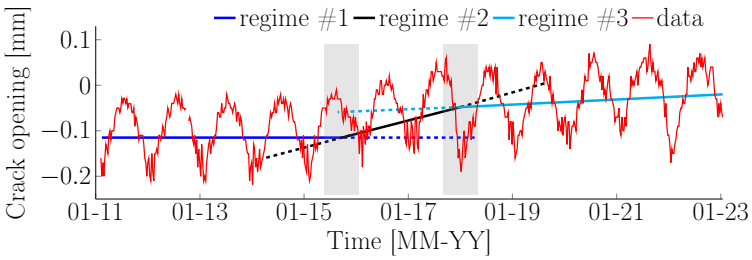
This first case study uses a dam's crack opening dataset from Jan-2011 to Jan-2023 as presented in Figure 5.8a. The original data is resampled using linear interpolation [123] to obtain weekly data, and the two weeks in 2012 without measurements are considered as having missing values. The data is divided into a training set from Jan-2011 to Jan-2014, a validation set from Jan-2014 to Jan-2015, and the remaining data is used as a test set.

Engineers responsible with monitoring the behavior of large structures, such as the dams studied in this chapter, are looking for changepoints indicating switches from a regime with known kinematic to a different one. The presence of a change in the kinematic is an indication of a possible anomaly in the structural condition which should trigger further investigations in order to identify its cause along with preventive maintenance actions. As illustrated in

Figure 5.8b, this exercice of identifying changepoints is fairly easy to do in retrospective after several years of data have been collected, however, doing so online as the data is collected is a much harder task. This task is made even harder by the non-stationary behavior displayed by structures and to which the model must constantly adapt without the supervision of an engineer. From this case study, we want to show that after training on a subset of stationary data, our model can detect the regime switches online in a nonstationary environment where the baseline changes over time, without requiring to retrain the model.



(a)



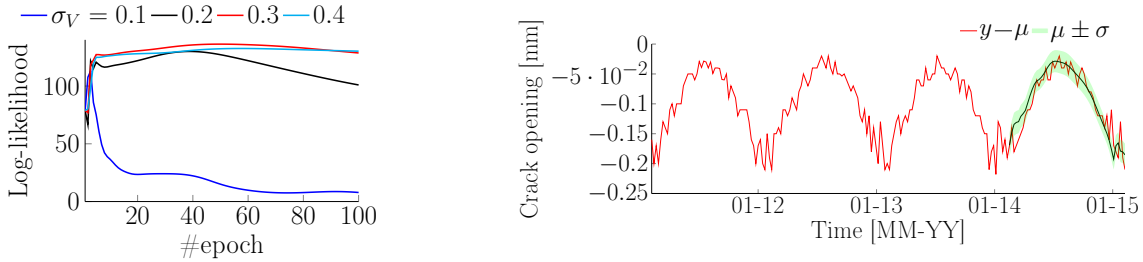
(b)

Figure 5.8 Case study #1. (a) Crack opening data from a dam in Canada, and (b) preliminary analysis to identify visually possible anomalies where the grey shaded area presents the regime switch quantitatively. The dash line with a corresponding colour presents the extension for the regime identified.

Following the procedure presented in Section 5.2, we first learn the model parameters on the training set illustrated in Figure 5.8a, which is assumed to be stationary and anomaly-free. We then evaluate the model's ability to detect the anomalies identified in the preliminary analysis as illustrated by the grey regions in Figure 5.8b.

For the TAGI-LSTM/SSM model, we employ the same hidden state vector as in the experiment presented in Section 5.3.1. The neural network architecture and hyperparameters are presented in Appendix C. In order to estimate the error variance for this dataset, we grid-search the optimal value σ_V^* from the set $\sigma_V \in \{0.1, 0.2, 0.3, 0.4\}$. Figure 5.9a shows that using $\sigma_V^* = 0.3$ gives the largest validation's log-likelihood at the epoch $e = 42$. Figure 5.9b

presents the multi-step-ahead predictions for the validation set using the optimal parameters $\boldsymbol{\theta}^*$.

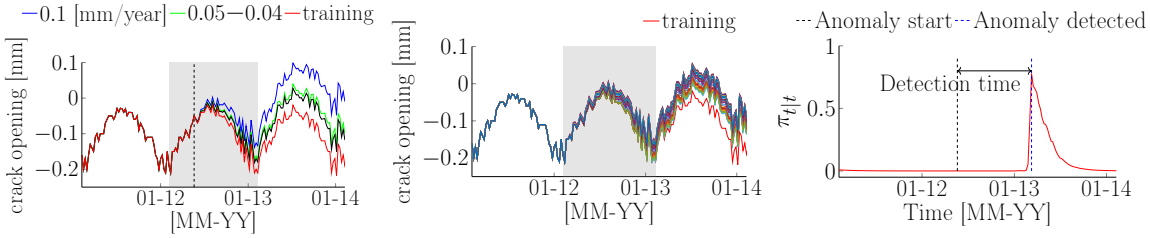


(a) Log-likelihood

(b) Validation predictions

Figure 5.9 Case study #1: crack opening. Log-likelihood and multi-step-ahead predictions for the validation set. The $\pm\sigma$ regions contain both the epistemic (parameter's) uncertainties obtained from the prior predictive distributions of the hidden states as well as the aleatory uncertainties associated with the error term's variance σ_V^2.

We employ the same two competing TAGI-LSTM/SSM models as presented in Section 5.3.1. For estimating the switching parameter $\boldsymbol{\mathcal{P}} = \{\sigma_{12}, z_{12}\}$, we define three slopes $\beta_b \in \{0.1, 0.05, 0.04\}$ [mm/year], and generate $3 \times 50 = 150$ simulated abnormal time series from the anomaly-free training set. Figure 5.10a presents three time series with anomalies having different slopes introduced at the same time step, whereas Figure 5.10b presents $\mathtt{N} = 50$ time series containing synthetic anomalies with a same slope $\beta_b = 0.05$ [mm/year], but with a random start time. Figure 5.10c shows an example of probability of regime switch along with the detection time for one time series.

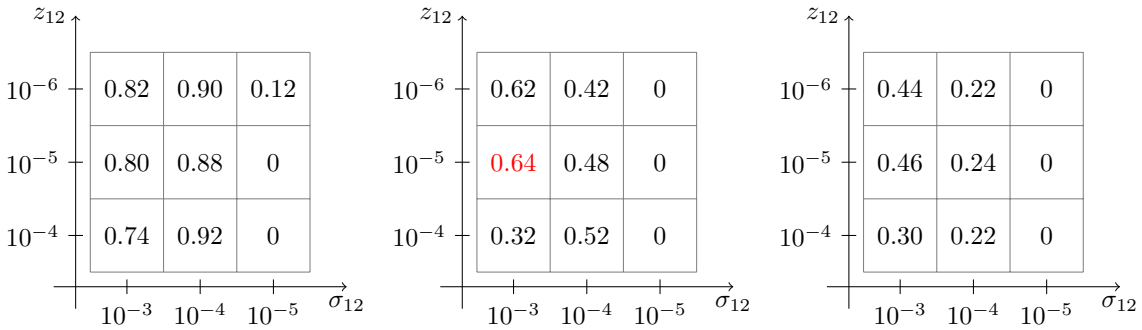


(a) (b) (c)

Figure 5.10 Case study #1: crack opening. Synthetic time series (a) with anomalies of different slopes introduced at the same time step, (b) with 50 anomalies of the same slope introduced at different time steps, and (c) the probability of anomaly $\pi_{t|t}$ and detection time for one synthetic time series. The shaded grey area presents the period where synthetic anomaly is randomly added.

The sets $\boldsymbol{\mathcal{P}}_k, \forall k \in \{1:9\}$, are obtained from the grid points $\sigma_{12} \in \{10^{-3}, 10^{-4}, 10^{-5}\}$ and

$z_{12} \in \{10^{-4}, 10^{-5}, 10^{-6}\}$. Figure 5.11 presents the detection probabilities p^{β} associated with $\boldsymbol{\mathcal{P}}_k$ for each slope value. The results show that the smallest slope that can be detected is 0.05 [mm/year] with the detection probability $p^{\beta} = 0.64$, and the corresponding optimal parameters are $\boldsymbol{\mathcal{P}}^* \equiv \boldsymbol{\mathcal{P}}_2 = \{\sigma_{12} = 10^{-3},\ z_{12} = 10^{-5}\}$.



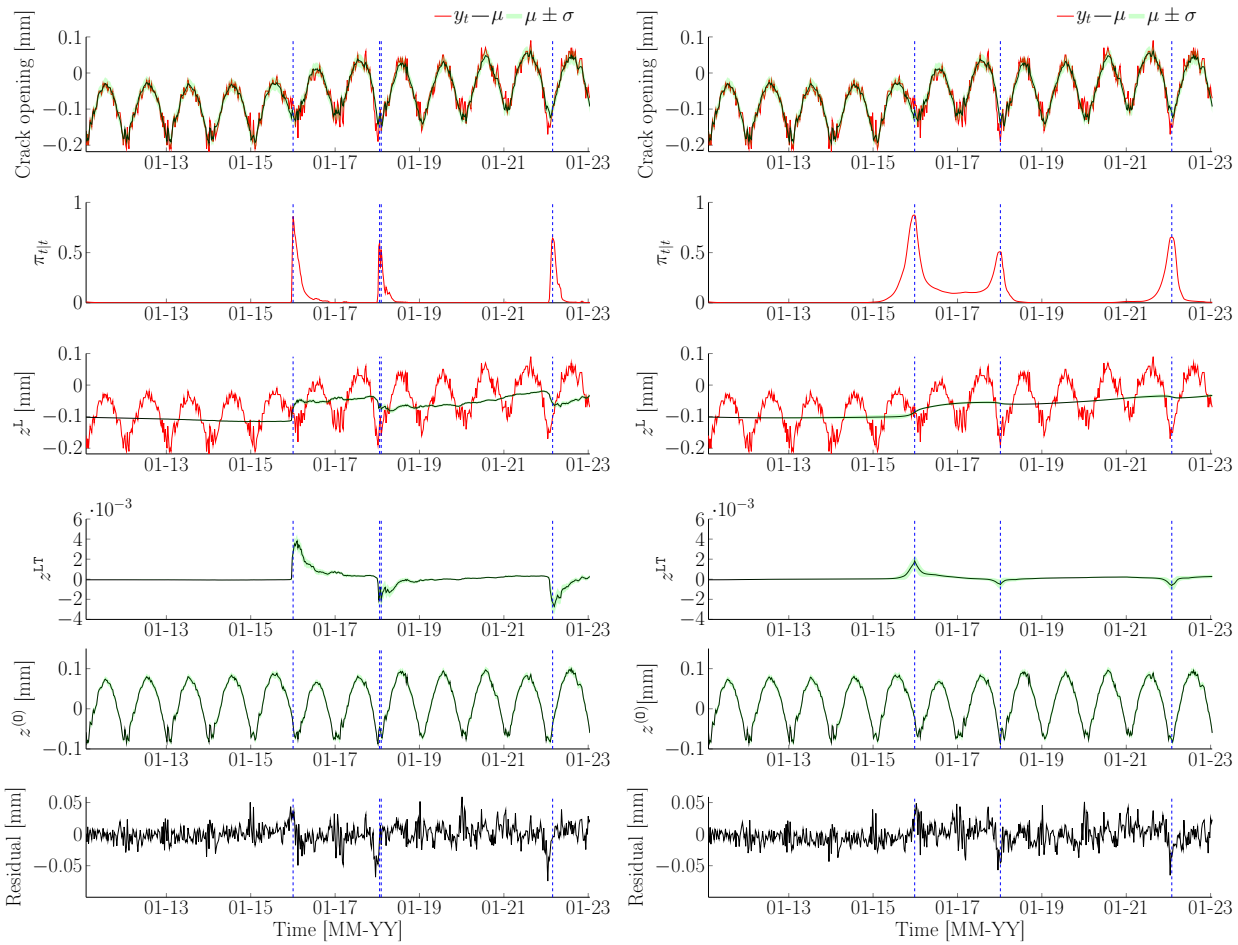
(a) $\beta_b = 0.1$ [mm/year] (b) $\beta_b = 0.05$ [mm/year] (c) $\beta_b = 0.04$ [mm/year]

Figure 5.11 Case study #1: crack opening. Probability of detection for different anomaly's slopes.

Figure 5.12a presents the filter results, where the data is processed online, for the detection probability as well as the interpretable hidden states of our models. After obtaining all historical data up to now, we can perform an offline smoothing analysis for the hidden states and detection probability as presented in Figure 5.12b.

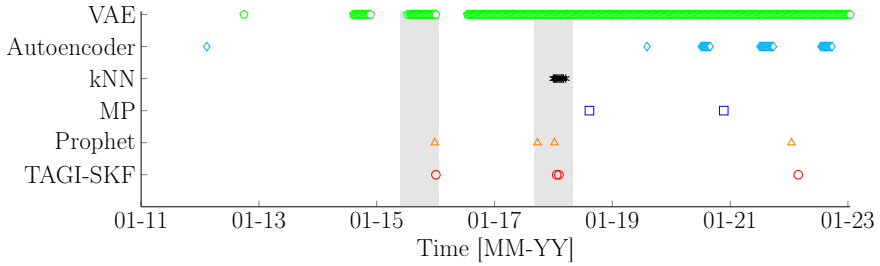
Unlike the case study using simulated data as presented in Section 5.3.1, the anomaly labels for this dataset are not available so that we cannot exactly quantify the detection time, detection probability and false alarm rate. Despite this, we can still judge a model's detection performance based on the regime switches that are visually identifiable from the data as shown in Figure 5.8b. Figure 5.12c compares the anomaly detection performance among different methods where each horizontal line of scatter points presents the timestamps when the alarms are triggered. The details on obtaining the hyperparameters for these methods are presented in Appendix E.

Figure 5.12c shows that our model triggers four alarms in which the first three correspond to the regime switches that are visually identifiable from the data, whereas the last one seems to be triggered by an abnormally low seasonal cycle. More data beyond 2023 will be required in order to confirm whether this alarm is associated with a temporary or a persistent effect. This further shows that when not having access to multi-years of data, it is especially hard to distinguish between regime changes and a one-off variability in the reversible effect caused by the environmental factors.



(a) Switching Kalman Filter results

(b) Switching Kalman Smoother results



(c) Comparison with other anomaly detection methods

Figure 5.12 Case study #1. Anomaly detection results obtained from our method and other models. Each line of scatter points presents when the alarms are triggered for each method, and the grey shaded area presents the visually identified anomalies from the preliminary analysis. The blue dashed lines present where the anomalies are detected by our method. The $\pm\sigma$ confidence intervals in the top row of (a) and (b) contain both the epistemic (parameter's) uncertainties obtained from the posterior predictive distributions of the hidden states as well as the aleatory uncertainties associated with the error term's variance σ_V^2.

The Prophet model provides a similar performance as most of the alarms trigged correspond to either the visually identified regime switches, or the same last anomaly detected by our method. Note that Prophet operates in an offline setup, using the entire time series for training, so that the results provide an upper bound for the detection performance. In order to use Prophet to process data online, ones would need to retrain the model each time new data becomes available, which would both reduce the performance and significantly increase the computational cost.

The Autoencoder and VAE models use the same setup as ours, where their parameters are learnt using the anomaly-free training set, and are tested using the entire time series. Both the Autoencoder and VAE models provide poor results as Autoencoder detects none of the two visually identifiable regime switches, and VAE almost continuously triggers alarms after the first regime switch.

The results for kNN and MP methods are obtained using an offline setup relying on the entire time series. The kNN model provides a good performance on synthetic data as presented in Section 5.3.1, but for this real dataset, it only correctly detects one of the two visually identifiable regime switches. None of the alarms detected by the MP model correspond to the regime switches identified.

This case study shows that our model can correctly detect the regime switches that are visually identifiable from the data, and it can operate online in a non-stationary environment where the baseline changes over time, without requiring to retrain. In addition, our model can provide interpretable results in terms of the hidden states which are useful for engineers. Our model which operates in an online setup provides a similar performance compared to the Prophet which works offline while using all the data for training. That highlights the superiority of the TAGI-SKF method for SHM applications where processing data online and adapting to changing conditions are mandatory.

Case study #2

This case study analyzes a dam's displacement dataset from Nov-2005 to Mar-2022 as presented in Figure 5.13. The original data is resampled using linear interpolation to obtain weekly data, and the weeks without measurements are considered as having missing values. The data is divided into a training set from Nov-2005 to Jan-2008, a validation set from Jan-2008 to July-2008, and the remaining data is used as a test set as presented in Figure 5.13. The preliminary analysis for visually identifying the possible regime switches is presented in Figure 5.14.

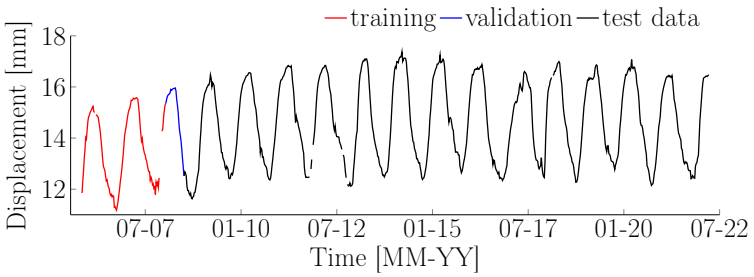


Figure 5.13 Case study #2: displacement data from a dam in Canada.

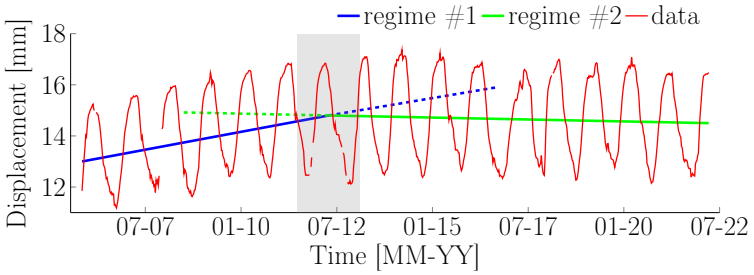
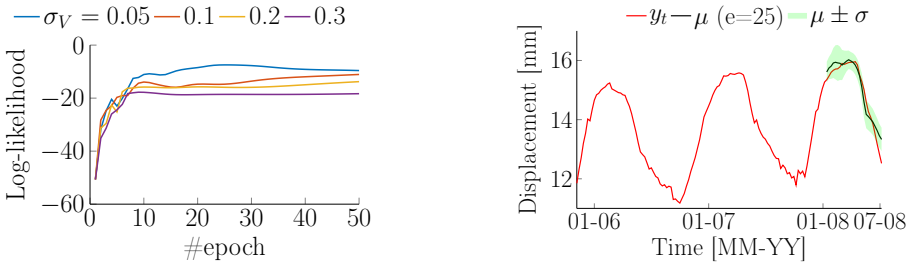


Figure 5.14 Case study #2: dam's displacement. Preliminary analysis to identify visually possible anomalies where the grey shaded area presents the regime switch quantitatively. The dash line with a corresponding colour presents the extension for the regime identified.

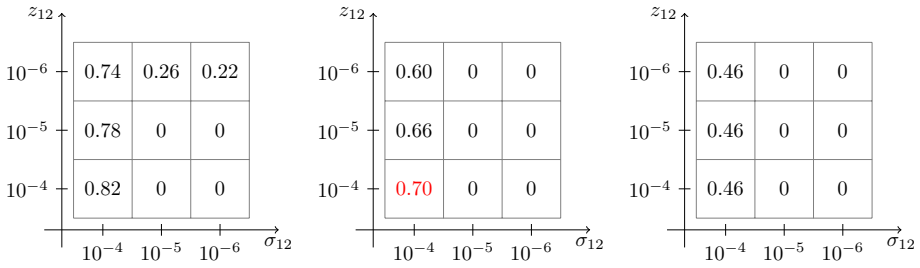
We employ the same TAGI-LSTM/SSM models as presented in Section 5.3.1. The neural network architecture and hyperparameters are presented in Appendix C. The process to obtain the parameters σ_V and $\boldsymbol{\theta}$ is presented in Figure 5.15, whereas the process to obtain the parameters $\boldsymbol{\mathcal{P}} = \{\sigma_{12}, z_{12}\}$ is presented in Figure 5.16.



(a) Log-likelihood (b) Validation predictions

Figure 5.15 Case study #2: dam's displacement. Log-likelihood and multi-step-ahead predictions for the validation set.

The filter and smoother results for the detection probability as well as the interpretable hidden states are presented in Figures 5.17a and 5.17b. The comparison of anomaly detection performance with other methods is presented in Figure 5.17c. Figure 5.17c shows that our



(a) $\beta_b = 2.5$ [mm/year] (b) $\beta_b = 1.3$ [mm/year] (c) $\beta_b = 1$ [mm/year]

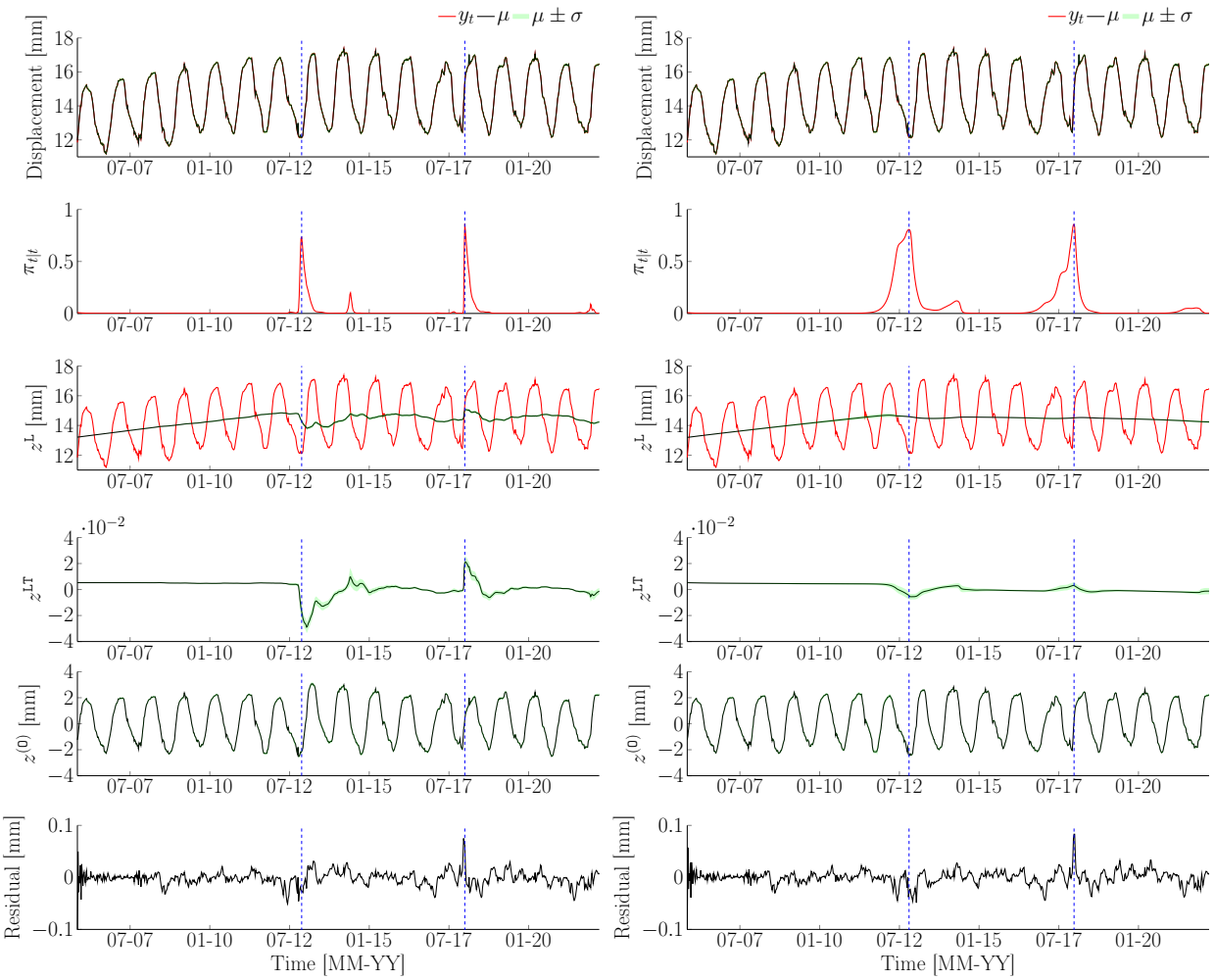
Figure 5.16 Case study #2: dam's displacement. Probability of detection for different anomaly's slopes.

method triggers two alarms where the first one corresponds to the visually identifiable regime switch, whereas the second one seems to be triggered by an abnormally low seasonal cycle. The Prophet and kNN models provide a similar performance compared to our models as they also detect the visually identifiable regime switch, and triggers multiple alarms for the same second anomaly detected by our method. By contrast, the MP is not able to detect the regime switch. Although Autoencoder and VAE detect the visually identifiable regime switch, they also trigger many false alarms.

Case study #3

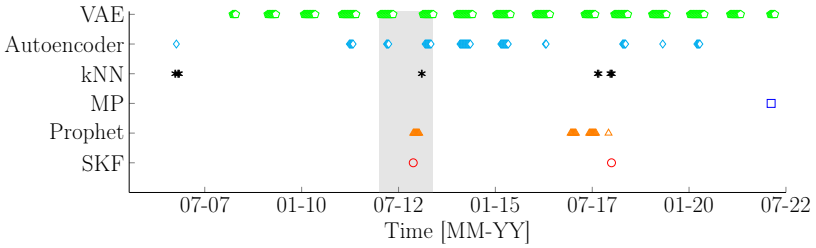
This case study analyzes another dam's crack opening dataset from Apr-2006 to Jan-2023 as presented in Figure 5.18a. The original data is resampled using linear interpolation to obtain weekly data, and the weeks without measurements are considered as having missing values. The air temperature for the same period as presented in Figure 5.18b is used as an explanatory variable, with missing data from Apr-2006 until Dec-2007. The data is divided into a training set from Apr-2006 to Mar-2009, a validation set from Mar-2009 to Sep-2009, and the remaining data is used as a test set as presented in Figure 5.18a.

The preliminary analysis for visually identifying the possible regime switches is presented in Figure 5.19. We employ the same TAGI-LSTM/SSM models as presented in Section 5.3.1. The neural network architecture and hyperparameters are presented in Appendix C. The process to obtain the parameters σ_V and $\boldsymbol{\theta}$ is presented in Figure 5.20, whereas the process to obtain the parameters $\boldsymbol{\mathcal{P}} = \{\sigma_{12}, z_{12}\}$ is presented in Figure 5.21. The filter and smoother results for the detection probability as well as the interpretable hidden states are presented in Figures 5.22a and 5.22b. The comparison of anomaly detection performance with other methods is presented in Figures 5.22c.



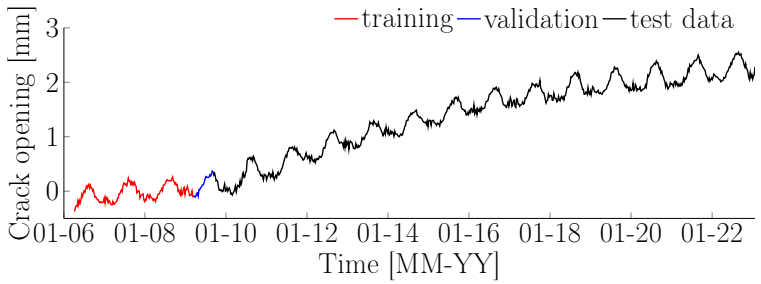
(a) Switching Kalman Filter results

(b) Switching Kalman Smoother results

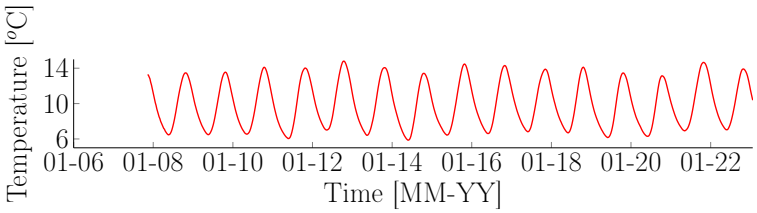


(c) Comparison with others anomaly detection methods

Figure 5.17 Case study #2. Anomaly detection results obtained from our method and other models. Each line of scatter points presents the alarms triggered for each method, and the grey shaded area presents the visually identified anomalies from the preliminary analysis. The blue dashed lines present where the anomalies are detected by our method. The $\pm\sigma$ confidence intervals in the top row of (a) and (b) contain both the epistemic (parameter's) uncertainties obtained from the posterior predictive distributions of the hidden states as well as the aleatory uncertainties associated with the error term's variance σ_V^2.



(a) Crack opening



(b) Air temperature

Figure 5.18 Case study #3: crack opening and air temperature from a dam in Canada.

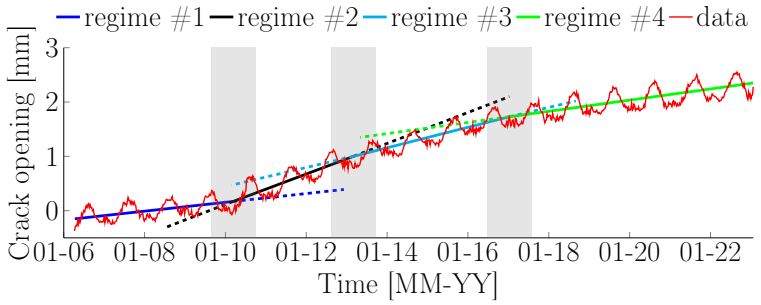
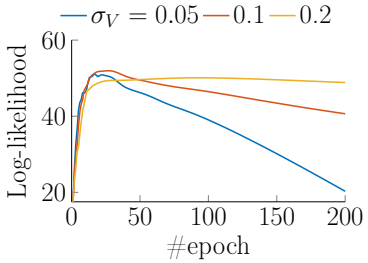
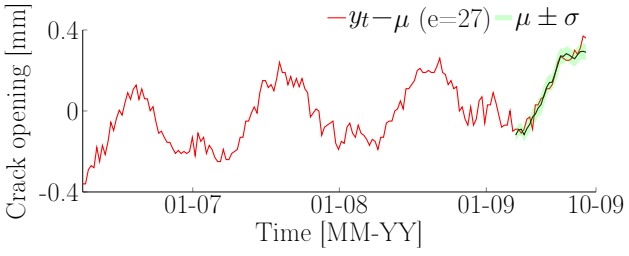


Figure 5.19 Case study #3: crack opening. Preliminary analysis to identify visually possible anomalies where the grey shaded area presents the regime switch quantitatively. The dash line with a corresponding colour presents the extension for the regime identified.

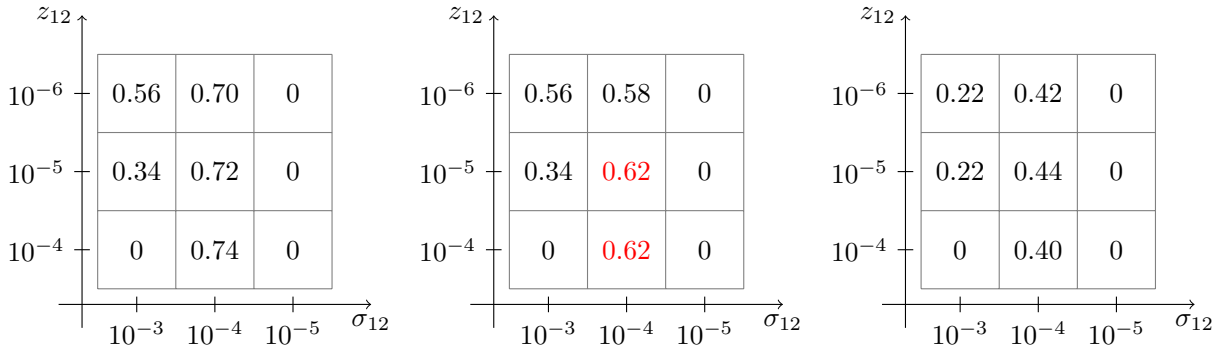


(a) Log-likelihood



(b) Validation predictions

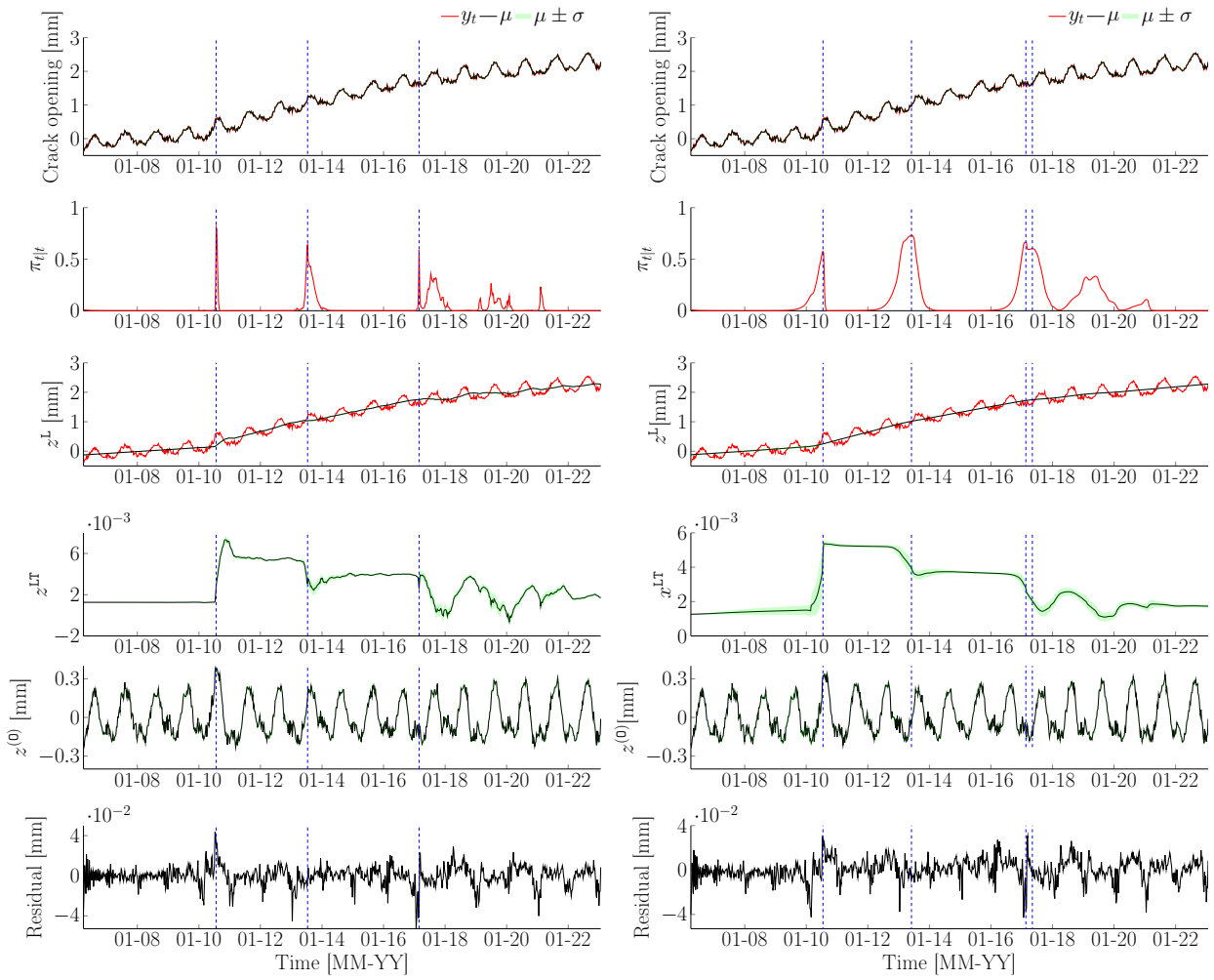
Figure 5.20 Case study #3: crack opening. Log-likelihood and multi-step-ahead predictions for the validation set.



(a) $\beta_b = 0.036$ [mm/year] (b) $\beta_b = 0.031$ [mm/year] (c) $\beta_b = 0.026$ [mm/year]

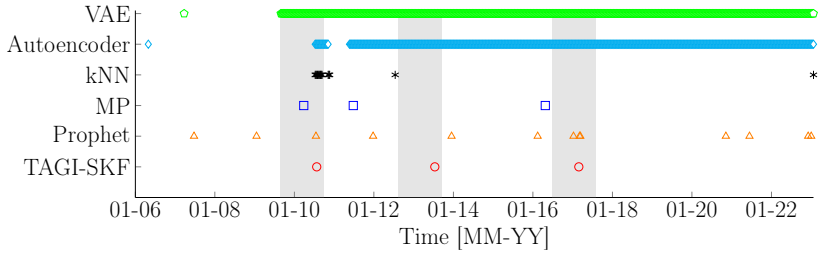
Figure 5.21 Case study #3: crack opening. Probability of detection for different anomaly's slopes.

Figure 5.22c shows that our model triggers three alarms where all of them correspond to the regime switches that are visually identifiable from the data. The Prophet method, which provides a similar performance with our method in the case study #1, now can only detect two out of three visually identifiable regime switches, while triggering several false alarms. This poor performance is in part related to the inability of the method to consider dependencies with respect to explanatory variables. Both the MP and kNN methods trigger alarms for only one out of the three regime switches. Similar to the case study #1, both Autoencoder and VAE provide poor performance as they almost constantly trigger alarms after the first visually identifiable regime switch making them unsuitable for SHM applications.



(a) Switching Kalman Filter results

(b) Switching Kalman Smoother results



(c) Comparison with other anomaly detection methods

Figure 5.22 Case study #3. Anomaly detection results obtained from our method and other models. The blue dashed lines present where the anomalies are detected by our method. The $\pm\sigma$ confidence intervals in the top row of (a) and (b) contain both the epistemic (parameter's) uncertainties obtained from the posterior predictive distributions of the hidden states as well as the aleatory uncertainties associated with the error term's variance σ_V^2.

5.4 Conclusion

In this chapter, we proposed a methodology to use the hybrid TAGI-LSTM/SSM model in the Switching Kalman Filter (SKF) framework in order to detect anomalies for structural health monitoring time series. The proposed TAGI-SKF method eliminates the need for manual feature engineering for defining the models' structures so that it can be applied to a large number of SHM time series. The experimental results on real SHM datasets showed that our method outperforms other baseline anomaly detection methods as it can reliably detect anomalies while having a tight control over the false alarm rate. In addition, our method can provide interpretable results which give useful insights for engineers, and it has the ability to process data online, as well as to adapt to changing conditions, while not requiring to retrain the model, making it suited for SHM applications.

CHAPTER 6 Conclusion

6.1 Thesis Conclusion

This thesis has contributed with scientific advancements by providing the mathematical formulations for establishing the analytically tractable Bayesian recurrent neural networks (RNN), hybrid models that probabilistically couple RNN with state-space models (SSM) along with their applications in order to enhance the scalability of structural health monitoring (SHM) data analysis. The following section presents the conclusions derived from this thesis.

The existing deterministic and Bayesian RNN models infer their parameters using backpropagation and gradient descent (GD). This thesis provides a new approach that uses Bayesian inference, specifically the Tractable Approximate Gaussian Inferrence (TAGI) method, in order to estimate the parameters as well as the hidden states for the long short-term memory (LSTM) and gated hidden unit (GRU) architectures. The limitation of the existing RNN models is that they either do not consider the epistemic uncertainties associated with the parameters or estimate them using Variational inference with a limited efficiency. The new analytically tractable RNN's formulations proposed in this thesis can take into account these epistemic uncertainties and estimate them analytically. With the predominant use of RNN in time series analysis and forecasting problems, this thesis provides a new Bayesian tool for such tasks. The experiments performed showed that for a same network architecture, our models provide on-par performance compared to the deterministic and variational RNN models trained with backpropagation and GD. In addition, the analytically tractable RNN can perform the smoothing procedure in a similar manner as it is done in SSM, where information is sent backward over time.

Existing hybrid models have explored the coupling between deterministic RNN and SSM models. Again, a limitation of these models is that they do not account for the epistemic uncertainties associated with the RNN's parameters. Moreover, these hybrid models need to use two different inference methods, i.e., the backpropagation for estimating the RNN's parameters and Bayesian inference for updating the SSM's hidden states. To address these limitations, this thesis presents how to probabilistically couple the analytically tractable RNN with SSM, where the epistemic uncertainties are taken into account, while Bayesian inference is used as the single mechanism for inferring both the network's parameters and SSM's hidden states. With the proposed method, one can now couple RNN with the existing components borrowed from SSM. This thesis has demonstrated, for example, how we can

couple TAGI-LSTM with the existing SSM's level and trend components, as well as a new parameter-free exponential smoothing component in order to analyze trended data, while not requiring offline preprocessing to remove trends from the data. This new hybrid model framework provides both the uncertainties associated with predictions as well as interpretable results, which are intuitive for engineers, in terms of hidden states decomposition and their associated uncertainties. The results obtained for SHM datasets showed that the proposed hybrid models provide on-par performance compared to the existing Bayesian Dynamic Linear Models (BDLM). This is achieved without requiring labor-intensive and time-consuming feature engineering nor optimization, thanks to the capability of the analytically tractable Bayesian RNN component at automatically identifying the dependencies within and across time series. This contribution is the key to enabling large-scale SHM data analysis.

In the context of structural health monitoring, the existing Switching Kalman filter (SKF) framework for anomaly detection relies on the capacity of the BDLM for describing different regimes. The limitation of the existing BLDM is that they require extensive manual feature engineering in order to define the models' structures, limiting the applications of the SKF to a small number of time series. To overcome this limitation, this thesis proposes using the TAGI-LSTM/SSM hybrid models for replacing the BDLM ones in the SKF framework. As a result, this new framework allows to eliminate the need for manual feature engineering in defining the models' structures so that the SKF framework for anomaly detection can be applied to a large number of SHM time series.

In conclusion, the methods developed in this thesis enhance the scalability of structural health monitoring data analysis. The experiments presented mostly focus on SHM applications, but our methods are generally applicable for time series analysis and forecasting as tested with several real world benchmarks.

6.2 Limitations

This section provides the limitations of the methods proposed in this thesis. Resolving these limitations will further improve the applicability of these methods.

6.2.1 Local versus Global Recurrent Neural Networks

For modeling multiple time series, this thesis uses the local setup, building a separate neural network model for each time series. The limitation of this setup is that each time series is considered as independent such that there is no information shared across time series. In future work, we should explore using a global setup where we train a single model for all

time series. Using this setup, the neural network's parameters would be shared and learnt from multiple time series. Note that there are problems arising when employing this global setup. Firstly, standardizing all time series into a common range is necessary and this task is non-trivial given that the range for each time may differ. Secondly, the standard deviation for the error term σ_V needs to be estimated for each time series, given that the noisiness of data for each time series is different. To address the scaling problem, one possible solution is using the experimental scaling approach as presented in [3], whereas for learning individual σ_V, we could extend the AGVI method presented in [6] to multiple outputs where each has a different σ_V. We see the potential to improve the current results by further exploring the initialization for the hidden and cell states for the TAGI-LSTM and TAGI-GRU.

6.2.2 Coupling Analytically Tractable Bayesian Recurrent Neural Networks and State-space Models

State-space models (SSM) update their hidden states using a single observation at a time so that the batch size $\mathtt{B} = 1$. As the analytically tractable Bayesian recurrent neural network is used as a component in SSM, the hybrid model also has the batch size $\mathtt{B} = 1$. Because SSM process data sequentially in time order, one cannot take advantage of parallel computation, using a batch size $\mathtt{B} > 1$. Solving this issue would lead to faster computation and enhance the capability of the methods to analyze larger datasets.

6.2.3 Anomaly Detection

The method presented in this thesis for anomaly detection in SHM can operate in a non-stationary environment where the baseline changes over time. However, this model cannot adapt in an environment where the recurrent pattern changes drastically in terms of its shape or magnitude. This is because the RNN which is responsible for modeling recurrent patterns has its parameters $\boldsymbol{\theta}$ fixed after the training. One possible solution for this problem is to keep learning the parameters $\boldsymbol{\theta}$ online after the training time as the data comes in. However, one needs to keep in mind that $\boldsymbol{\theta}$ are typically learnt over multiple epochs so that updating $\boldsymbol{\theta}$ online (using each new data once) might not adjust their values fast enough. Another limitation of the proposed anomaly detection model is that it could not be trained on data containing anomalies. Addressing these limitations would broaden the applicability of the method for detecting anomalies in SHM time series.

REFERENCES

[1] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *The Thirty-Fifth AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 11 106–11 115.

[2] Y. Wang, A. Smola, D. Maddix, J. Gasthaus, D. Foster, and T. Januschowski, "Deep factors for forecasting," in *Proceedings of the 36th International Conference on Machine Learning*, vol. 97, 2019, pp. 6607–6617.

[3] 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.

[4] S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski, "Deep state space models for time series forecasting," in *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[5] H.-F. Yu, N. Rao, and I. S. Dhillon, "Temporal regularized matrix factorization for high-dimensional time series prediction," in *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[6] B. Deka, "Analytical Bayesian parameter inference for probabilistic models with engineering applications," Ph.D. dissertation, Polytechnique Montréal, Canada, 2022.

[7] B. Deka and J.-A. Goulet, "Approximate Gaussian variance inference for state-space models," *International Journal of Adaptive Control and Signal Processing*, vol. 37, no. 11, pp. 2934–2962, 2023.

[8] B. Deka, L. H. Nguyen, and J.-A. Goulet, "Analytically tractable heteroscedastic uncertainty quantification in bayesian neural networks for regression tasks," *Neurocomputing*, vol. 572, p. 127183, 2024.

[9] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting," in *International Conference on Learning Representations*, 2020.

[10] E. Skepetari, "Data, benchmarks, and methods submitted to the M4 forecasting competition," https://github.com/Mcompetitions/M4-methods, 2020, accessed: 2023-02-27.

[11] K. Malm R, A. M Simon, S. F, and H. R, "Summary of Theme A: Behaviour prediction of a concrete arch dam," in 16^{th} *International Benchmark Workshop on Numerical Analysis of Dams*. International Commission on Large Dams, 2022.

[12] J.-A. Goulet, L. H. Nguyen, and S. Amiri, "Tractable approximate Gaussian inference for Bayesian neural networks," *Journal of Machine Learning Research*, vol. 22, pp. 1–23, 2021.

[13] B. Deka, L. H. Nguyen, and J.-A. Goulet, "Analytically tractable heteroscedastic uncertainty quantification in Bayesian neural networks for regression tasks," *Neurocomputing*, p. 127183, 2023.

[14] A. Ansar, B. Flyvbjerg, A. Budzier, and D. Lunn, "Does infrastructure investment lead to economic growth or economic fragility? Evidence from China," *Oxford Review of Economic Policy*, vol. 32, no. 3, pp. 360–390, 2016.

[15] R. Kumar and P. Gardoni, "Chapter 16 - stochastic modeling of deterioration in buildings and civil infrastructure," in *Handbook of Seismic Risk Analysis and Management of Civil Infrastructure Systems*. Woodhead Publishing, 2013, pp. 410–434.

[16] F. Catbas, "Chapter 1 - structural health monitoring: applications and data analysis," in *Structural Health Monitoring of Civil Infrastructure Systems*. Woodhead Publishing, 2009, pp. 1–39.

[17] S. M., W. J. Staszewski, and R. N. Swamy, "Smart sensing technologies for structural health monitoring of civil engineering structures," *Advances in Civil Engineering*, vol. 2010, pp. 1–13, 2010.

[18] L. Hui and O. Jinping, "Structural health monitoring: From sensing technology stepping to health diagnosis," *Procedia Engineering*, vol. 14, pp. 753–760, 2011.

[19] J. Durbin and S. Koopman, *Time series analysis by state space methods*. Oxford University Press, 2001.

[20] J.-A. Goulet, *Probabilistic Machine Learning for Civil Engineers*. MIT Press, 2020.

[21] ——, "Bayesian dynamic linear models for structural health monitoring," *Structural Control and Health Monitoring*, vol. 24, no. 12, p. e2035, 2017.

[22] K. P. Murphy, "Switching Kalman Filters," University of California, Berkeley, Tech. Rep., 1998.

[23] C. C. Aggarwal, *Neural Networks and Deep Learning: A Textbook*. Springer Publishing Company, Incorporated, 2018.

[24] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[25] H. Hewamalage, C. Bergmeir, and K. Bandara, "Recurrent neural networks for time series forecasting: Current status and future directions," *International Journal of Forecasting*, vol. 37, no. 1, pp. 388–427, 2021.

[26] T. Januschowski, J. Gasthaus, Y. Wang, D. Salinas, V. Flunkert, M. Bohlke-Schneider, and L. Callot, "Criteria for classifying forecasting methods," *International Journal of Forecasting*, vol. 36, no. 1, pp. 167–177, 2020.

[27] S. Särkkä, *Bayesian filtering and smoothing*. Cambridge University Press, 2020.

[28] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[29] H. Afshari, S. Gadsden, and S. Habibi, "Gaussian filters for parameter and state estimation: A general review of theory and recent trends," *Signal Processing*, vol. 135, pp. 218–238, 2017.

[30] L. Ljung, "Asymptotic behavior of the extended Kalman filter as a parameter estimator for linear systems," *IEEE Transactions on Automatic Control*, vol. 24, no. 1, pp. 36–50, 1979.

[31] E. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium*, 2000, pp. 153–158.

[32] I. Arasaratnam and S. Haykin, "Cubature Kalman filters," *IEEE Transactions on Automatic Control*, vol. 54, no. 6, pp. 1254–1269, 2009.

[33] A. Doucet, N. d. Freitas, K. P. Murphy, and S. J. Russell, "Rao-Blackwellised particle filtering for dynamic Bayesian networks," in *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, 2000, p. 176–183.

[34] L. H. Nguyen, I. Gaudot, S. Khazaeli, and J.-A. Goulet, "A Kernel-based method for modeling non-harmonic periodic phenomena in Bayesian dynamic linear models," *Frontiers in Built Environment*, vol. 5, p. 8, 2019.

[35] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian data analysis*. Chapman and Hall/CRC, 2013.

[36] S. Brooks, A. Gelman, G. Jones, and X.-L. Meng, *Handbook of Markov Chain Monte Carlo*. CRC press, 2011.

[37] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian Data Analysis*. Chapman and Hall/CRC, 2004.

[38] B. Deka, L. H. Nguyen, S. Amiri, and J.-A. Goulet, "The Gaussian multiplicative approximation for state-space models," *Structural Control & Health Monitoring*, vol. 29, no. 3, p. e2904, 2022.

[39] A. Runnalls, "Kullback-Leibler approach to Gaussian mixture reduction," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 43, no. 3, pp. 989–999, 2007.

[40] S. Khazaeli, L. H. Nguyen, and J. A. Goulet, "Anomaly detection using state-space models and reinforcement learning," *Structural Control and Health Monitoring*, vol. 28, no. 6, p. e2720, 2021.

[41] S. Khazaeli, "Damage detection for structural health monitoring using reinforcement and imitation learning," Ph.D. dissertation, Polytechnique Montréal, Canada, 2022.

[42] C. Miao, S. Liang, M. Chen, J. Ma, S. Wang, and J. Xiao, "Flow-TTS: A non-autoregressive network for text to speech based on flow," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 7209–7213.

[43] A. Goyal, A. Lamb, Y. Zhang, S. Zhang, A. Courville, and Y. Bengio, "Professor forcing: A new algorithm for training recurrent networks," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016, p. 4608–4616.

[44] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to construct deep recurrent neural networks," in *Proceedings of the Second International Conference on Learning Representations (ICLR 2014)*, 2014.

[45] 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.

[46] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[47] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *2014 Neural Information Processing Systems (NIPS) Workshop on Deep Learning*, 2014.

[48] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[49] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," in *Proceedings of the 9th ISCA Workshop on Speech Synthesis Workshop (SSW 9)*, 2016, p. 125.

[50] S. Bai, J. Z. Kolter, and V. Koltun, "WaveNet: A Generative Model for Raw Audio," in *2018 International Conference on Learning Representations (ICLR) Workshop*, 2018.

[51] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 677–691, 2017.

[52] 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 *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[53] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 4580–4584.

[54] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[55] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?" in *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, 2023.

[56] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with Auto-Correlation for long-term series forecasting," in *Advances in Neural Information Processing Systems*, 2021.

[57] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 5243–5253.

[58] S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A. X. Liu, and S. Dustdar, "Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting," in *International Conference on Learning Representations*, 2022.

[59] B. Lim, S. Ö. Arık, N. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," *International Journal of Forecasting*, vol. 37, no. 4, pp. 1748–1764, 2021.

[60] N. Wu, B. Green, X. Ben, and S. O'Banion, "Deep transformer models for time series forecasting: The influenza prevalence case," *ArXiv*, vol. abs/2001.08317, 2020.

[61] Y. Yuan and L. Lin, "Self-supervised pretraining of transformers for satellite image time series classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 474–487, 2021.

[62] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, 2019, pp. 4171–4186.

[63] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[64] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, 2006.

[65] Y. Bengio, *Neural Networks: Tricks of the Trade*. Springer Berlin Heidelberg, 2012, ch. Practical Recommendations for Gradient-Based Training of Deep Architectures, pp. 437–478.

[66] B. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.

[67] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of the 30th International Conference on Machine Learning*, vol. 28, no. 3, 2013, pp. 1139–1147.

[68] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. 61, pp. 2121–2159, 2011.

[69] D. Kingma and J. Ba, "ADAM: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*, 2015.

[70] D. J. C. MacKay, "A practical Bayesian framework for backpropagation networks," *Neural Computation*, vol. 4, no. 3, pp. 448–472, 05 1992.

[71] S. Sun, C. Chen, and L. Carin, "Learning structured weight uncertainty in Bayesian neural networks," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, vol. 54, 2017, pp. 1283–1292.

[72] A. Wu, S. Nowozin, E. Meeds, R. E. Turner, J. M. Hernandez-Lobato, and A. L. Gaunt, "Deterministic variational inference for robust Bayesian neural networks," in *International Conference on Learning Representations*, 2019.

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

[74] J. M. Hernandez-Lobato and R. Adams, "Probabilistic backpropagation for scalable learning of Bayesian neural networks," in *Proceedings of the 32nd International Conference on Machine Learning*, vol. 37, 07–09 Jul 2015, pp. 1861–1869.

[75] Y. Gal and Z. Ghahramani, "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, 2016, pp. 1050–1059.

[76] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, p. 6405–6416.

[77] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour, "Dropout improves recurrent neural networks for handwriting recognition," in *14th International Conference on Frontiers in Handwriting Recognition*, 2014, pp. 285–290.

[78] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *ArXiv*, vol. abs/1409.2329, 2014.

[79] T. Moon, H. Choi, H. Lee, and I. Song, "RNNDROP: A novel dropout for RNNS in ASR," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding*, 2015, pp. 65–70.

[80] K. N. Markelle Kelly, Rachel Longjohn, "The UCI machine learning repository," https://archive.ics.uci.edu, accessed: 2023-10-14.

[81] S. Duane, A. Kennedy, B. J. Pendleton, and D. Roweth, "Hybrid monte carlo," *Physics Letters B*, vol. 195, no. 2, pp. 216–222, 1987.

[82] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1026–1034.

[83] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, vol. 9, 2010, pp. 249–256.

[84] B. Efron, *Large-Scale Inference: Empirical Bayes Methods for Estimation, Testing, and Prediction*. Cambridge University Press, 2010.

[85] L.-H. Nguyen and J.-A. Goulet, "Analytically tractable inference in deep neural networks," in *The 4th Workshop on Tractable Probabilistic Modeling*, 2021.

[86] ——, "Analytically tractable hidden-states inference in Bayesian neural networks," *Journal of Machine Learning Research*, vol. 23, no. 50, pp. 1–33, 2022.

[87] R. G. Krishnan, U. Shalit, and D. Sontag, "Structured inference networks for nonlinear state space models," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017, p. 2101–2109.

[88] X. Zheng, M. Zaheer, A. Ahmed, Y. Wang, E. P. Xing, and A. Smola, "State space LSTM models with particle MCMC inference," *ArXiv*, vol. abs/1711.11179, 2017.

[89] M. Zaheer, A. Ahmed, and A. J. Smola, "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, 2017, pp. 3967–3976.

[90] M. Fraccaro, S. K. Sønderby, U. Paquet, and O. Winther, "Sequential neural models with stochastic layers," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016, p. 2207–2215.

[91] S. Smyl, "A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting," *International Journal of Forecasting*, vol. 36, no. 1, pp. 75–85, 2020.

[92] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Survey*, vol. 41, no. 3, 2009.

[93] K. Choi, J. Yi, C. Park, and S. Yoon, "Deep learning for anomaly detection in time-series data: Review, analysis, and guidelines," *IEEE Access*, vol. 9, pp. 120 043–120 065, 2021.

[94] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM Computing Survey*, vol. 54, no. 2, 2021.

[95] S. Schmidl, P. Wenig, and T. Papenbrock, "Anomaly detection in time series: A comprehensive evaluation," *Proceedings of the VLDB Endowment*, vol. 15, no. 9, p. 1779–1797, 2022.

[96] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano, "A review on outlier/anomaly detection in time series data," *ACM Computing Surveys*, vol. 54, pp. 1–33, 2020.

[97] S. J. Taylor and B. Letham, "Forecasting at scale," *The American Statistician*, vol. 72, no. 1, pp. 37–45, 2018.

[98] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE Journal*, vol. 37, no. 2, pp. 233–243, 1991.

[99] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Pacific Rim International Conference on Artificial Intelligence (PRICAI), Workshop on Machine Learning for Sensory Data Analysis (MLSDA)*, 2014.

[100] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," *ArXiv*, vol. abs/1312.6114, 2022.

[101] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[102] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, 2000, p. 427–438.

[103] C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh, "Matrix profile I: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 2016, pp. 1317–1322.

[104] S. M. Law, "Stumpy: A powerful and scalable python library for time series data mining," *Journal of Open Source Software*, vol. 4, no. 39, p. 1504, 2019.

[105] H. E. Rauch, F. Tung, and C. T. Striebel, "Maximum likelihood estimates of linear dynamic systems," *AIAA Journal*, vol. 3, no. 8, pp. 1445–1450, 1965.

[106] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, vol. 29, 2016, p. 1027–1035.

[107] K. Bandara, C. Bergmeir, and S. Smyl, "Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach," *Expert Systems with Applications*, vol. 140, p. 112896, 2020.

[108] R. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*, 2nd ed. OTexts, 2018.

[109] B. Deka, L. H. Nguyen, S. Amiri, and J.-A. Goulet, "The Gaussian multiplicative approximation for state-space models," *Structural Control and Health Monitoring*, vol. 29, no. 3, p. e2904, 2022.

[110] G. Athanasopoulos, R. J. Hyndman, H. Song, and D. C. Wu, "The tourism forecasting competition," *International Journal of Forecasting*, vol. 27, no. 3, pp. 822–844, 2011.

[111] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The M4 competition: Results, findings, conclusion and way forward," *International Journal of Forecasting*, vol. 34, no. 4, pp. 802–808, 2018.

[112] G. E. P. Box, G. C. Reinsel, and G. M. Jenkins, *Time series analysis: forecasting and control*. Prentice-Hall, 1994.

[113] R. Hyndman, A. Koehler, J. Ord, and R. Snyder, *Forecasting with Exponential Smoothing: The State Space Approach*. Springer, 2008.

[114] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The M4 competition: 100,000 time series and 61 forecasting methods," *International Journal of Forecasting*, vol. 36, no. 1, pp. 54–74, 2020.

[115] R. J. Hyndman and Y. Khandakar, "Automatic time series forecasting: the forecast package for R," *Journal of Statistical Software*, vol. 26, no. 3, pp. 1–22, 2008.

[116] A. Alexandrov, K. Benidis, M. Bohlke-Schneider, V. Flunkert, J. Gasthaus, T. Januschowski, D. C. Maddix, S. Rangapuram, D. Salinas, J. Schulz, L. Stella, A. C. Türkmen, and Y. Wang, "Gluonts: Probabilistic and neural time series modeling in python," *Journal of Machine Learning Research*, vol. 21, no. 116, pp. 1–6, 2020.

[117] K. Gutierrez, C. Challu, F. Garza, and M. Mergenthaler, "Pytorch implementation of the ES-RNN algorithm," https://github.com/kdgutier/esrnn_torch, 2021, accessed: 2023-03-17.

[118] A. J. Koning, P. H. Franses, M. Hibon, and H. Stekler, "The M3 competition: Statistical tests of the results," *International Journal of Forecasting*, vol. 21, no. 3, pp. 397–409, 2005.

[119] J. Hu, F. Ma, and S. Wu, "Comprehensive investigation of leakage problems for concrete gravity dams with penetrating cracks based on detection and monitoring data: A case study," *Structural Control and Health Monitoring*, vol. 25, no. 4, p. e2127, 2018.

[120] E. Cross, K. Koo, J. Brownjohn, and K. Worden, "Long-term monitoring and data analysis of the Tamar bridge," *Mechanical Systems and Signal Processing*, vol. 35, no. 1, pp. 16–34, 2013.

[121] J.-A. Goulet and K. Koo, "Empirical validation of Bayesian dynamic linear models in the context of structural health monitoring," *Journal of Bridge Engineering*, vol. 23, no. 2, p. 05017017, 2018.

[122] L. H. Nguyen and J.-A. Goulet, "Anomaly detection with the Switching Kalman Filter for structural health monitoring," *Structural Control and Health Monitoring*, vol. 25, no. 4, p. e2136, 2018.

[123] A. M. Bayen and T. Siauw, "Chapter 14 - interpolation," in *An Introduction to MATLAB Programming and Numerical Methods for Engineers*. Academic Press, 2015, pp. 211–223.

[124] Y. Zhao, Z. Nasrullah, and Z. Li, "Pyod: A python toolbox for scalable outlier detection," *Journal of Machine Learning Research*, vol. 20, no. 96, pp. 1–7, 2019.

APPENDIX A COVARIANCES FOR TAGI-LSTM

A.1 TAGI-LSTM – Covariances Required for Backward Step

This appendix presents the calculations for the covariances between the hidden states and parameters of the j^{the} LSTM layer with the hidden states of the $j+1^{th}$ LSTM layer, $\text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{H}^{(j+1)}_{t|t-1})$ and $\text{cov}(\boldsymbol{\theta}^{(j)}_{t|t-1}, \boldsymbol{H}^{(j+1)}_{t|t-1})$, which are necessary for the backward step presented in Section 3.2.2.

A.1.1 Covariances Between the Hidden States of Two Consecutive LSTM Layers

$$
\begin{aligned}
\text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{H}^{(j+1)}_{t|t-1}) &= \text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{O}^{(j+1)}_{t|t-1} \odot \tilde{\tanh}(\boldsymbol{C}^{(j+1)}_{t|t-1})) \\
&= \text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{O}^{(j+1)}_{t|t-1}) \odot \mathbb{E}[\tilde{\tanh}(\boldsymbol{C}^{(j+1)}_{t|t-1})] \\
&+ \text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \tilde{\tanh}(\boldsymbol{C}^{(j+1)}_{t|t-1})) \odot \mathbb{E}[\boldsymbol{O}^{(j+1)}_{t|t-1}] \\
&= \text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{O}^{(j+1)}_{t|t-1}) \odot \mathbb{E}[\tilde{\tanh}(\boldsymbol{C}^{(j+1)}_{t|t-1})] \\
&+ \text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{C}^{(j+1)}_{t|t-1}) \mathbf{J}^{c(j+1)}_{t|t-1} \odot \mathbb{E}[\boldsymbol{O}^{(j+1)}_{t|t-1}],
\end{aligned}
$$

where

$$
\begin{aligned}
\text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{O}^{(j+1)}_{t|t-1}) &= \text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \tilde{\sigma}(\boldsymbol{Z}^{o(j+1)}_{t|t-1})) \\
&= \text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{Z}^{o(j+1)}_{t|t-1}) \mathbf{J}^{o(j+1)}_{t|t-1} \\
&= \text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{W}^{o(j)}_{t|t-1} \boldsymbol{H}^{(j)}_{t|t-1} + \boldsymbol{U}^{o(j)}_{t|t-1} \boldsymbol{H}^{(j+1)}_{t-1} + \boldsymbol{B}^{o(j)}_{t|t-1}) \mathbf{J}^{o(j+1)}_{t|t-1} \\
&= \text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{W}^{o(j)}_{t|t-1} \boldsymbol{H}^{(j)}_{t|t-1}) \mathbf{J}^{o(j+1)}_{t|t-1} \\
&= \text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{H}^{(j)}_{t|t-1}) \mathbb{E}[\boldsymbol{W}^{o(j)}_{t|t-1}] \mathbf{J}^{o(j+1)}_{t|t-1}, \\
\text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{C}^{(j+1)}_{t|t-1}) &= \text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{F}^{(j+1)}_{t|t-1} \odot \boldsymbol{C}^{(j+1)}_{t-1|t-1} + \boldsymbol{I}^{(j+1)}_{t|t-1} \odot \tilde{\boldsymbol{C}}^{(j+1)}_{t|t-1}) \\
&= \text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{F}^{(j+1)}_{t|t-1} \odot \boldsymbol{C}^{(j+1)}_{t-1|t-1}) + \text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{I}^{(j+1)}_{t|t-1} \odot \tilde{\boldsymbol{C}}^{(j+1)}_{t|t-1}) \\
&= \text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{F}^{(j+1)}_{t|t-1}) \odot \mathbb{E}[\boldsymbol{C}^{(j+1)}_{t-1|t-1}] + \text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{I}^{(j+1)}_{t|t-1}) \odot \mathbb{E}[\tilde{\boldsymbol{C}}^{(j+1)}_{t|t-1}] \\
&+ \text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \tilde{\boldsymbol{C}}^{(j+1)}_{t|t-1}) \odot \mathbb{E}[\boldsymbol{I}^{(j+1)}_{t|t-1}] \\
&= \text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{H}^{(j)}_{t|t-1}) \mathbb{E}[\boldsymbol{W}^{f(j)}_{t|t-1}] \mathbf{J}^{f(j+1)}_{t|t-1} \odot \mathbb{E}[\boldsymbol{C}^{(j+1)}_{t-1|t-1}] \\
&+ \text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{H}^{(j)}_{t|t-1}) \mathbb{E}[\boldsymbol{W}^{i(j)}_{t|t-1}] \mathbf{J}^{i(j+1)}_{t|t-1} \odot \mathbb{E}[\tilde{\boldsymbol{C}}^{(j+1)}_{t|t-1}] \\
&+ \text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{H}^{(j)}_{t|t-1}) \mathbb{E}[\boldsymbol{W}^{\tilde{c}(j)}_{t|t-1}] \mathbf{J}^{\tilde{c}(j+1)}_{t|t-1} \odot \mathbb{E}[\boldsymbol{I}^{(j+1)}_{t|t-1}],
\end{aligned}
$$

$\mathbf{J}^{f(j+1)}_{t|t-1}, \mathbf{J}^{i(j+1)}_{t|t-1}, \mathbf{J}^{o(j+1)}_{t|t-1}, \mathbf{J}^{\tilde{c}(j+1)}_{t|t-1}$, and $\mathbf{J}^{c(j+1)}_{t|t-1}$ are the diagonal Jacobian matrices.

A.1.2 Covariances Between the Parameters and Hidden States of Two Consecutive LSTM Layers

The parameters of the forget gate:

$$
\begin{aligned}
\operatorname{cov}(\boldsymbol{W}^{f(j)}_{i,t|t-1}, H^{(j+1)}_{i,t|t-1}) &= \operatorname{cov}(\boldsymbol{W}^{f(j)}_{i,t|t-1}, O^{(j+1)}_{i,t|t-1} \odot \tilde{\tanh}(C^{(j+1)}_{i,t|t-1})) \\
&= \operatorname{cov}(\boldsymbol{W}^{f(j)}_{i,t|t-1}, \tilde{\tanh}(C^{(j+1)}_{i,t|t-1}))\mathbb{E}[O^{(j+1)}_{i,t|t-1}] \\
&= \operatorname{cov}(\boldsymbol{W}^{f(j)}_{i,t|t-1}, F^{(j+1)}_{i,t|t-1}C^{(j+1)}_{i,t-1|t-1} + I^{(j+1)}_{i,t|t-1}\tilde{C}^{(j+1)}_{i,t|t-1})\mathrm{J}^{c(j+1)}_{i,t|t-1}\mathbb{E}[O^{(j+1)}_{i,t|t-1}] \\
&= \operatorname{cov}(\boldsymbol{W}^{f(j)}_{i,t|t-1}, F^{(j+1)}_{i,t|t-1}C^{(j+1)}_{i,t-1|t-1})\mathrm{J}^{c(j+1)}_{i,t|t-1}\mathbb{E}[O^{(j+1)}_{i,t|t-1}] \\
&= \operatorname{cov}(\boldsymbol{W}^{f(j)}_{i,t|t-1}, F^{(j+1)}_{i,t|t-1})\mathbb{E}[C^{(j+1)}_{i,t-1|t-1}]\mathrm{J}^{c(j+1)}_{i,t|t-1}\mathbb{E}[O^{(j+1)}_{i,t|t-1}] \\
&= \operatorname{cov}(\boldsymbol{W}^{f(j)}_{i,t|t-1}, \boldsymbol{W}^{f(j)}_{i,t|t-1})\mathbb{E}[\boldsymbol{X}^{(j)}_{t|t-1}]\mathrm{J}^{f(j+1)}_{i,t|t-1}\mathbb{E}[C^{(j+1)}_{i,t-1|t-1}]\mathrm{J}^{c(j+1)}_{i,t|t-1}\mathbb{E}[O^{(j+1)}_{i,t|t-1}], \\
\operatorname{cov}(\boldsymbol{U}^{f(j)}_{i,t|t-1}, H^{(j+1)}_{i,t|t-1}) &= \operatorname{cov}(\boldsymbol{U}^{f(j)}_{i,t|t-1}, \boldsymbol{U}^{f(j)}_{i,t|t-1})\mathbb{E}[\boldsymbol{H}^{(j+1)}_{t-1|t-1}]\mathrm{J}^{f(j+1)}_{i,t|t-1}\mathbb{E}[C^{(j+1)}_{i,t-1|t-1}]\mathrm{J}^{c(j+1)}_{i,t|t-1}\mathbb{E}[O^{(j+1)}_{i,t|t-1}], \\
\operatorname{cov}(B^{f(j)}_{i,t|t-1}, H^{(j+1)}_{i,t|t-1}) &= \operatorname{var}(B^{f(j)}_{i,t|t-1})\mathrm{J}^{f(j+1)}_{i,t|t-1}\mathbb{E}[C^{(j+1)}_{i,t-1|t-1}]\mathrm{J}^{c(j+1)}_{i,t|t-1}\mathbb{E}[O^{(j+1)}_{i,t|t-1}].
\end{aligned}
$$

The parameters of the input gate:

$$
\begin{aligned}
\operatorname{cov}(\boldsymbol{W}^{i(j)}_{i,t|t-1}, H^{(j+1)}_{i,t|t-1}) &= \operatorname{cov}(\boldsymbol{W}^{i(j)}_{i,t|t-1}, \boldsymbol{W}^{i(j)}_{i,t|t-1})\mathbb{E}[\boldsymbol{X}^{(j)}_{t|t-1}]\mathrm{J}^{i(j+1)}_{i,t|t-1}\mathbb{E}[\tilde{C}^{(j+1)}_{i,t|t-1}]\mathrm{J}^{c(j+1)}_{i,t|t-1}\mathbb{E}[O^{(j+1)}_{i,t|t-1}], \\
\operatorname{cov}(\boldsymbol{U}^{i(j)}_{i,t|t-1}, H^{(j+1)}_{i,t|t-1}) &= \operatorname{cov}(\boldsymbol{U}^{i(j)}_{i,t|t-1}, \boldsymbol{U}^{i(j)}_{i,t|t-1})\mathbb{E}[\boldsymbol{H}^{(j+1)}_{t-1|t-1}]\mathrm{J}^{i(j+1)}_{i,t|t-1}\mathbb{E}[\tilde{C}^{(j+1)}_{i,t|t-1}]\mathrm{J}^{c(j+1)}_{i,t|t-1}\mathbb{E}[O^{(j+1)}_{i,t|t-1}], \\
\operatorname{cov}(B^{i(j)}_{i,t|t-1}, H^{(j+1)}_{i,t|t-1}) &= \operatorname{var}(B^{i(j)}_{i,t|t-1})\mathrm{J}^{i(j+1)}_{i,t|t-1}\mathbb{E}[\tilde{C}^{(j+1)}_{i,t|t-1}]\mathrm{J}^{c(j+1)}_{i,t|t-1}\mathbb{E}[O^{(j+1)}_{i,t|t-1}].
\end{aligned}
$$

The parameters of the output gate:

$$
\begin{aligned}
\operatorname{cov}(\boldsymbol{W}^{o(j)}_{i,t|t-1}, H^{(j+1)}_{i,t|t-1}) &= \operatorname{cov}(\boldsymbol{W}^{o(j)}_{i,t|t-1}, \boldsymbol{W}^{o(j)}_{i,t|t-1})\mathbb{E}[\boldsymbol{X}^{(j)}_{t|t-1}]\mathrm{J}^{o(j+1)}_{i,t|t-1}\mathbb{E}[\tanh(C^{(j+1)}_{i,t|t-1})], \\
\operatorname{cov}(\boldsymbol{U}^{o(j)}_{i,t|t-1}, H^{(j+1)}_{i,t|t-1}) &= \operatorname{cov}(\boldsymbol{U}^{o(j)}_{i,t|t-1}, \boldsymbol{U}^{o(j)}_{i,t|t-1})\mathbb{E}[\boldsymbol{H}^{(j+1)}_{t-1|t-1}]\mathrm{J}^{o(j+1)}_{i,t|t-1}\mathbb{E}[\tanh(C^{(j+1)}_{i,t|t-1})], \\
\operatorname{cov}(B^{o(j)}_{i,t|t-1}, H^{(j+1)}_{i,t|t-1}) &= \operatorname{var}(B^{o(j)}_{i,t|t-1})\mathrm{J}^{o(j+1)}_{i,t|t-1}\mathbb{E}[\tanh(C^{(j+1)}_{i,t|t-1})].
\end{aligned}
$$

The parameters of the $\tilde{\boldsymbol{c}}$ gate:

$$
\begin{aligned}
\operatorname{cov}(\boldsymbol{W}^{c(j)}_{i,t|t-1}, H^{(j+1)}_{i,t|t-1}) &= \operatorname{cov}(\boldsymbol{W}^{c(j)}_{i,t|t-1}, \boldsymbol{W}^{c(j)}_{i,t|t-1})\mathbb{E}[\boldsymbol{H}^{(j+1)}_{t-1|t-1}]\mathrm{J}^{\tilde{c}(j+1)}_{i,t|t-1}\mathbb{E}[I^{(j+1)}_{i,t|t-1}]\mathrm{J}^{c(j+1)}_{i,t|t-1}\mathbb{E}[O^{(j+1)}_{i,t|t-1}], \\
\operatorname{cov}(\boldsymbol{U}^{c(j)}_{i,t|t-1}, H^{(j+1)}_{i,t|t-1}) &= \operatorname{cov}(\boldsymbol{U}^{c(j)}_{i,t|t-1}, \boldsymbol{U}^{c(j)}_{i,t|t-1})\mathbb{E}[\boldsymbol{H}^{(j+1)}_{t-1|t-1}]\mathrm{J}^{\tilde{c}(j+1)}_{i,t|t-1}\mathbb{E}[I^{(j+1)}_{i,t|t-1}]\mathrm{J}^{c(j+1)}_{i,t|t-1}\mathbb{E}[O^{(j+1)}_{i,t|t-1}], \\
\operatorname{cov}(B^{c(j)}_{i,t|t-1}, H^{(j+1)}_{i,t|t-1}) &= \operatorname{var}(B^{c(j)}_{i,t|t-1})\mathrm{J}^{\tilde{c}(j+1)}_{i,t|t-1}\mathbb{E}[I^{(j+1)}_{i,t|t-1}]\mathrm{J}^{c(j+1)}_{i,t|t-1}\mathbb{E}[O^{(j+1)}_{i,t|t-1}],
\end{aligned}
$$

where $\{\boldsymbol{W}^{f(j)}_{i,t|t-1}, \boldsymbol{W}^{i(j)}_{i,t|t-1}, \boldsymbol{W}^{o(j)}_{i,t|t-1}, \boldsymbol{W}^{c(j)}_{i,t|t-1}\}$ and $\{\boldsymbol{U}^{f(j)}_{i,t|t-1}, \boldsymbol{U}^{i(j)}_{i,t|t-1}, \boldsymbol{U}^{o(j)}_{i,t|t-1}, \boldsymbol{U}^{c(j)}_{i,t|t-1}\}$ are weight vectors at time t for row i at j^{th} layer.

A.2 TAGI-LSTM – Covariances Required for Smoothing Procedure

This appendix presents the calculations for the covariances which are necessary for the smoothing procedure presented in Section 3.2.3.

A.2.1 Covariances Between the Hidden States of the Same LSTM Layer

$$
\begin{aligned}
\text{cov}(\boldsymbol{H}^{(j)}_{t|t}, \boldsymbol{H}^{(j)}_{t+1|t}) &= \text{cov}(\boldsymbol{H}^{(j)}_{t|t}, \boldsymbol{O}^{(j)}_{t+1|t} \odot \tilde{\tanh}(\boldsymbol{C}^{(j)}_{t+1|t})) \\
&= \text{cov}(\boldsymbol{H}^{(j)}_{t|t}, \boldsymbol{O}^{(j)}_{t+1|t}) \odot \mathbb{E}[\tilde{\tanh}(\boldsymbol{C}^{(j)}_{t+1|t})] + \text{cov}(\boldsymbol{H}^{(j)}_{t|t}, \tilde{\tanh}(\boldsymbol{C}^{(j)}_{t+1|t})) \odot \mathbb{E}[\boldsymbol{O}^{(j)}_{t+1|t}] \\
&= \text{cov}(\boldsymbol{H}^{(j)}_{t|t}, \boldsymbol{O}^{(j)}_{t+1|t}) \odot \mathbb{E}[\tilde{\tanh}(\boldsymbol{C}^{(j)}_{t+1|t})] + \text{cov}(\boldsymbol{H}^{(j)}_{t|t}, \boldsymbol{C}^{(j)}_{t+1|t}) \mathbf{J}^{c(j)}_{t+1|t} \odot \mathbb{E}[\boldsymbol{O}^{(j)}_{t+1|t}],
\end{aligned}
$$

where

$$
\begin{aligned}
\text{cov}(\boldsymbol{H}^{(j)}_{t|t}, \boldsymbol{O}^{(j)}_{t+1|t}) &= \text{cov}(\boldsymbol{H}^{(j)}_{t|t}, \tilde{\sigma}(\boldsymbol{W}^{o(j-1)}_{t+1|t} \boldsymbol{H}^{(j-1)}_{t+1|t} + \boldsymbol{U}^{o(j-1)}_{t+1|t} \boldsymbol{H}^{(j)}_{t|t} + \boldsymbol{B}^{o(j-1)}_{t+1|t})) \\
&= \text{cov}(\boldsymbol{H}^{(j)}_{t|t}, \boldsymbol{W}^{o(j-1)}_{t+1|t} \boldsymbol{H}^{(j-1)}_{t+1|t} + \boldsymbol{U}^{o(j-1)}_{t+1|t} \boldsymbol{H}^{(j)}_{t|t} + \boldsymbol{B}^{o(j-1)}_{t+1|t}) \mathbf{J}^{o(j)}_{t+1|t} \\
&= \text{cov}(\boldsymbol{H}^{(j)}_{t|t}, \boldsymbol{U}^{o(j-1)}_{t+1|t} \boldsymbol{H}^{(j)}_{t|t}) \mathbf{J}^{o(j)}_{t+1|t} \\
&= \text{cov}(\boldsymbol{H}^{(j)}_{t|t}, \boldsymbol{H}^{(j)}_{t|t}) \mathbb{E}[\boldsymbol{U}^{o(j-1)}_{t+1|t}] \mathbf{J}^{o(j)}_{t+1|t}, \\
\text{cov}(\boldsymbol{H}^{(j)}_{t|t}, \boldsymbol{C}^{(j)}_{t+1|t}) &= \text{cov}(\boldsymbol{H}^{(j)}_{t|t}, \boldsymbol{F}^{(j)}_{t+1|t} \odot \boldsymbol{C}^{(j)}_{t|t} + \boldsymbol{I}^{(j)}_{t+1|t} \odot \tilde{\boldsymbol{C}}^{(j)}_{t+1|t}) \\
&= \text{cov}(\boldsymbol{H}^{(j)}_{t|t}, \boldsymbol{F}^{(j)}_{t+1|t} \odot \boldsymbol{C}^{(j)}_{t|t}) + \text{cov}(\boldsymbol{H}^{(j)}_{t|t}, \boldsymbol{I}^{(j)}_{t+1|t} \odot \tilde{\boldsymbol{C}}^{(j)}_{t+1|t}) \\
&= \text{cov}(\boldsymbol{H}^{(j)}_{t|t}, \boldsymbol{F}^{(j)}_{t+1|t}) \odot \mathbb{E}[\boldsymbol{C}^{(j)}_{t|t}] + \text{cov}(\boldsymbol{H}^{(j)}_{t|t}, \boldsymbol{C}^{(j)}_{t|t}) \odot \mathbb{E}[\boldsymbol{F}^{(j)}_{t+1|t}] \\
&+ \text{cov}(\boldsymbol{H}^{(j)}_{t|t}, \boldsymbol{I}^{(j)}_{t+1|t}) \odot \mathbb{E}[\tilde{\boldsymbol{C}}^{(j)}_{t+1|t}] + \text{cov}(\boldsymbol{H}^{(j)}_{t|t}, \tilde{\boldsymbol{C}}^{(j)}_{t+1|t}) \odot \mathbb{E}[\boldsymbol{I}^{(j)}_{t+1|t}] \\
&= \text{cov}(\boldsymbol{H}^{(j)}_{t|t}, \boldsymbol{H}^{(j)}_{t|t}) \mathbb{E}[\boldsymbol{U}^{f(j-1)}_{t+1|t}] \mathbf{J}^{f(j)}_{t+1|t} \odot \mathbb{E}[\boldsymbol{C}^{(j)}_{t|t}] + \text{cov}(\boldsymbol{H}^{j}_{t|t}, \boldsymbol{C}^{(j)}_{t|t}) \odot \mathbb{E}[\boldsymbol{F}^{(j)}_{t+1|t}] \\
&+ \text{cov}(\boldsymbol{H}^{(j)}_{t|t}, \boldsymbol{H}^{(j)}_{t|t}) \mathbb{E}[\boldsymbol{U}^{i(j-1)}_{t+1|t}] \mathbf{J}^{i(j)}_{t+1|t} \odot \mathbb{E}[\tilde{\boldsymbol{C}}^{(j)}_{t+1|t}] \\
&+ \text{cov}(\boldsymbol{H}^{(j)}_{t|t}, \boldsymbol{H}^{(j)}_{t|t}) \mathbb{E}[\boldsymbol{U}^{\tilde{c}(j-1)}_{t+1|t}] \mathbf{J}^{\tilde{c}(j)}_{t+1|t} \odot \mathbb{E}[\boldsymbol{I}^{(j)}_{t+1|t}].
\end{aligned}
$$

A.2.2 Covariances Between the Cell States of the Same LSTM Layer

$$\begin{aligned}
\operatorname{cov}(C_{i,t|t}^{(j)}, C_{i,t+1|t}^{(j)}) &= \operatorname{cov}(C_{i,t|t}^{(j)}, F_{i,t+1|t}^{(j)} \odot C_{i,t|t}^{(j)} + I_{i,t+1|t}^{(j)} \odot \tilde{C}_{i,t+1|t}^{(j)}) \\
&= \operatorname{cov}(C_{i,t|t}^{(j)}, F_{i,t+1|t}^{(j)} \odot C_{i,t|t}^{(j)}) \\
&= \operatorname{cov}(C_{i,t|t}^{(j)}, C_{i,t|t}^{(j)}) \odot \mathbb{E}[F_{i,t+1|t}^{(j)}].
\end{aligned}$$

A.2.3 Covariances Between the Output Hidden States

$$\begin{aligned}
\operatorname{cov}(\boldsymbol{Z}_{t|t}^{(0)}, \boldsymbol{Z}_{t+1|t}^{(0)}) &= \operatorname{cov}(\boldsymbol{W}_{t|t}^{(L)} \boldsymbol{H}_{t|t}^{(L)} + \boldsymbol{B}_{t|t}^{(L)}, \boldsymbol{W}_{t+1|t}^{(L)} \boldsymbol{H}_{t+1|t}^{(L)} + \boldsymbol{B}_{t+1|t}^{(L)}) \\
&= \operatorname{cov}(\boldsymbol{W}_{t|t}^{(L)} \boldsymbol{H}_{t|t}^{(L)}, \boldsymbol{W}_{t+1|t}^{(L)} \boldsymbol{H}_{t+1|t}^{(L)}) + \operatorname{cov}(\boldsymbol{B}_{t|t}^{(L)}, \boldsymbol{B}_{t+1|t}^{(L)}) \\
&= \operatorname{cov}(\boldsymbol{W}_{t|t}^{(L)}, \boldsymbol{W}_{t|t}^{(L)}) \operatorname{cov}(\boldsymbol{H}_{t|t}^{(L)}, \boldsymbol{H}_{t+1|t}^{(L)}) + \operatorname{cov}(\boldsymbol{W}_{t|t}^{(L)}, \boldsymbol{W}_{t|t}^{(L)}) \mathbb{E}[\boldsymbol{H}_{t|t}^{(L)}] \mathbb{E}[\boldsymbol{H}_{t+1|t}^{(L)}] \\
&+ \operatorname{cov}(\boldsymbol{H}_{t|t}^{(L)}, \boldsymbol{H}_{t+1|t}^{(L)}) (\mathbb{E}[\boldsymbol{W}_{t|t}^{(L)}])^2 + \operatorname{cov}(\boldsymbol{B}_{t|t}^{(L)}, \boldsymbol{B}_{t|t}^{(L)}),
\end{aligned}$$

where $\boldsymbol{W}_{t+1|t}^{(L)} = \boldsymbol{W}_{t|t}^{(L)}$ and $\boldsymbol{B}_{t+1|t}^{(L)} = \boldsymbol{B}_{t|t}^{(L)}$.

APPENDIX B COVARIANCES FOR TAGI-GRU

B.1 TAGI-GRU – Covariances Required for Estimating Hidden States

$$
\begin{aligned}
\mathbb{E}[H_{i,t|t-1}] &= \mathbb{E}[H_{i,t-1|t-1}] - \mathbb{E}[Z_{i,t|t-1} \cdot H_{t-1|t-1}] + \mathbb{E}[Z_{i,t|t-1} \cdot \tilde{H}_{i,t|t-1}], \\
\text{var}(H_{i,t|t-1}) &= \text{var}(H_{i,t-1|t-1}) + \text{var}(Z_{i,t|t-1} \cdot H_{t-1|t-1}) + \text{var}(Z_{i,t|t-1} \cdot \tilde{H}_{i,t|t-1}) \\
&- 2\text{cov}(H_{i,t-1|t-1}, Z_{i,t|t-1} \cdot H_{i,t-1|t-1}) + 2\text{cov}(H_{i,t-1|t-1}, Z_{i,t|t-1} \cdot \tilde{H}_{i,t|t-1}) \\
&- 2\text{cov}(Z_{i,t|t-1} H_{i,t-1|t-1}, Z_{i,t|t-1} \tilde{H}_{i,t|t-1}) \\
&= \text{var}(H_{i,t-1|t-1}) + \text{var}(Z_{i,t|t-1} \cdot H_{t-1|t-1}) + \text{var}(Z_{i,t|t-1} \cdot \tilde{H}_{i,t|t-1}) \\
&- 2\left(\text{cov}(H_{i,t-1|t-1}, Z_{i,t|t-1}) \cdot \mathbb{E}[H_{i,t-1|t-1}] + \text{var}(H_{i,t-1|t-1}) \cdot \mathbb{E}[Z_{i,t|t-1}]\right) \\
&+ 2\left(\text{cov}(H_{i,t-1|t-1}, Z_{i,t|t-1}) \cdot \mathbb{E}[\tilde{H}_{i,t|t-1}] + \text{cov}(H_{i,t-1|t-1}, \tilde{H}_{i,t|t-1}) \cdot \mathbb{E}[Z_{i,t|t-1}]\right) \\
&- 2[\text{var}(Z_{i,t|t-1}) \cdot \text{cov}(H_{i,t-1|t-1}, \tilde{H}_{i,t|t-1} + \text{cov}(Z_{i,t|t-1}, \tilde{H}_{i,t|t-1}) \cdot \text{cov}(H_{i,t-1|t-1}, Z_{i,t|t-1} \\
&- \text{var}(Z_{i,t|t-1}) \cdot \mathbb{E}[H_{i,t-1|t-1}] \cdot \mathbb{E}[\tilde{H}_{i,t|t-1}] + \text{cov}(Z_{i,t|t-1}, \tilde{H}_{i,t|t-1}) \cdot \mathbb{E}[H_{i,t-1|t-1}] \cdot \mathbb{E}[Z_{i,t|t-1}] \\
&+ \text{cov}(H_{i,t-1|t-1}, Z_{i,t|t-1}) \cdot \mathbb{E}[Z_{i,t|t-1}] \cdot \mathbb{E}[\tilde{H}_{i,t|t-1}] + \text{cov}(H_{i,t-1|t-1}, \tilde{H}_{i,t|t-1}) \cdot \mathbb{E}[Z_{i,t|t-1}]^2] \\
&= \text{var}(H_{i,t-1|t-1}) + \text{var}(Z_{i,t|t-1} \cdot H_{t-1|t-1}) + \text{var}(Z_{i,t|t-1} \cdot \tilde{H}_{i,t|t-1}) \\
&+ 2\text{cov}(H_{i,t-1|t-1}, Z_{i,t|t-1})[\mathbb{E}[\tilde{H}_{i,t|t-1}] - \mathbb{E}[H_{i,t-1|t-1}] - \text{cov}(Z_{i,t|t-1}, \tilde{H}_{i,t|t-1}) \\
&- \mathbb{E}[Z_{i,t|t-1}]\mathbb{E}[\tilde{H}_{i,t|t-1}]] \\
&+ 2\text{cov}(H_{i,t-1|t-1}, \tilde{H}_{i,t|t-1}) \left[\mathbb{E}[Z_{i,t|t-1}] - \text{var}(Z_{i,t|t-1}) - \mathbb{E}[Z_{i,t|t-1}]^2\right] \\
&- 2\text{var}(H_{i,t-1|t-1}) \cdot \mathbb{E}[Z_{i,t|t-1}] - 2\text{var}(Z_{i,t|t-1}) \cdot \mathbb{E}[H_{i,t-1|t-1}] \cdot \mathbb{E}[\tilde{H}_{i,t|t-1}] \\
&- 2\text{cov}(Z_{i,t|t-1}, \tilde{H}_{i,t|t-1}) \cdot \mathbb{E}[H_{i,t-1|t-1}] \cdot \mathbb{E}[Z_{i,t|t-1}],
\end{aligned}
$$

where

$$
\begin{aligned}
\text{cov}(H_{i,t-1|t-1}, Z_{i,t|t-1}) &= \text{cov}\left(H_{i,t-1|t-1}, \tilde{\sigma}(Z^z_{i,t|t-1})\right) \\
&= \text{cov}\left(H_{i,t-1|t-1}, Z^z_{i,t|t-1}\right) \cdot \text{J}^z_{i,t|t-1} \\
&= \text{cov}\left(H_{i,t-1|t-1}, \boldsymbol{W}^z_{i,t|t-1} \cdot \boldsymbol{X}_{t|t-1} + \boldsymbol{U}^z_{i,t|t-1} \cdot \boldsymbol{H}_{t-1|t-1}\right) \cdot \text{J}^z_{i,t|t-1} \\
&= \text{var}(H_{i,t-1|t-1}) \cdot U^z_{i,t|t-1} \cdot \text{J}^z_{i,t|t-1},
\end{aligned}
$$

$$
\begin{aligned}
\text{cov}(H_{i,t-1|t-1}, \tilde{H}_{i,t|t-1}) &= \text{cov}\left(H_{i,t-1|t-1}, \tilde{\tanh}(Z^h_{i,t|t-1})\right) \\
&= \text{cov}\left(H_{i,t-1|t-1}, Z^h_{i,t|t-1}\right) \cdot \text{J}^h_{i,t|t-1} \\
&= \text{cov}\left(H_{i,t-1|t-1}, \boldsymbol{W}^h_{i,t|t-1} \cdot \boldsymbol{X}_{t|t-1} + \boldsymbol{U}^h_{i,t|t-1} \cdot \boldsymbol{C}_{t|t-1}\right) \cdot \text{J}^h_{i,t|t-1}, \\
&= \text{cov}\left(H_{i,t-1|t-1}, \boldsymbol{U}^h_{i,t|t-1} \cdot \boldsymbol{C}_{t|t-1}\right) \cdot \text{J}^h_{i,t|t-1}, \\
&= \text{cov}\left(H_{i,t-1|t-1}, \boldsymbol{U}^h_{i,t|t-1} \cdot (\boldsymbol{R}_{t|t-1} \odot \boldsymbol{H}_{t-1|t-1})\right) \cdot \text{J}^h_{i,t|t-1}.
\end{aligned}
$$

B.2 TAGI-GRU – Covariances Required for Backward Step

This appendix presents the calculations for the covariances between the hidden states and parameters of the j^{the} GRU layer with the hidden states of the $j+1^{th}$ GRU layer, $\text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{H}^{(j+1)}_{t|t-1})$ and $\text{cov}(\boldsymbol{\theta}^{(j)}_{t|t-1}, \boldsymbol{H}^{(j+1)}_{t|t-1})$, which are necessary for the backward step presented in Section 3.3.

B.2.1 Covariances Between the Hidden States of Two Consecutive GRU Layers

$$
\begin{aligned}
\text{cov}(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{H}^{(j+1)}_{t|t-1}) &= \text{cov}\left(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{H}^{(j+1)}_{t-1|t-1} - \boldsymbol{Z}^{(j+1)}_{t|t-1} \odot \boldsymbol{H}^{(j+1)}_{t-1|t-1} + \boldsymbol{Z}^{(j+1)}_{t|t-1} \odot \tilde{\boldsymbol{H}}^{(j+1)}_{t|t-1}\right) \\
&= \cancelto{0}{\text{cov}\left(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{H}^{(j+1)}_{t-1|t-1}\right)} - \text{cov}\left(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{Z}^{(j+1)}_{t|t-1} \odot \boldsymbol{H}^{(j+1)}_{t-1|t-1}\right) \\
&+ \text{cov}\left(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{Z}^{(j+1)}_{t|t-1} \odot \tilde{\boldsymbol{H}}^{(j+1)}_{t|t-1}\right) \\
&= -\text{cov}\left(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{Z}^{(j+1)}_{t|t-1}\right) \odot \mathbb{E}[\boldsymbol{H}^{(j+1)}_{t-1|t-1}] - \cancelto{0}{\text{cov}\left(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{H}^{(j+1)}_{t-1|t-1}\right)} \odot \mathbb{E}[\boldsymbol{Z}^{(j+1)}_{t|t-1}] \\
&+ \text{cov}\left(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{Z}^{(j+1)}_{t|t-1}\right) \odot \mathbb{E}[\tilde{\boldsymbol{H}}^{(j+1)}_{t|t-1}] + \text{cov}\left(\boldsymbol{H}^{(j)}_{t|t-1}, \tilde{\boldsymbol{H}}^{(j+1)}_{t|t-1}\right) \odot \mathbb{E}[\boldsymbol{Z}^{(j+1)}_{t|t-1}] \\
&= \text{cov}\left(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{Z}^{(j+1)}_{t|t-1}\right) \odot \left(\mathbb{E}[\tilde{\boldsymbol{H}}^{(j+1)}_{t|t-1}] - \mathbb{E}[\boldsymbol{H}^{(j+1)}_{t-1|t-1}]\right) \\
&+ \text{cov}\left(\boldsymbol{H}^{(j)}_{t|t-1}, \tilde{\boldsymbol{H}}^{(j+1)}_{t|t-1}\right) \odot \mathbb{E}[\boldsymbol{Z}^{(j+1)}_{t|t-1}],
\end{aligned}
$$

where

$$
\begin{aligned}
\text{cov}\left(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{Z}^{(j+1)}_{t|t-1}\right) &= \text{cov}\left(\boldsymbol{H}^{(j)}_{t|t-1}, \tilde{\sigma}(\boldsymbol{Z}^{z(j+1)}_{t|t-1})\right) \\
&= \text{cov}\left(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{Z}^{z(j+1)}_{t|t-1}\right) \cdot \mathbf{J}^{z(j+1)}_{t|t-1} \\
&= \text{cov}\left(\boldsymbol{H}^{(j)}_{t|t-1}, \boldsymbol{W}^{z(j)}_{t|t-1} \cdot \boldsymbol{H}^{(j)}_{t|t-1} + \boldsymbol{U}^{z(j)}_{t|t-1} \cdot \boldsymbol{H}^{(j+1)}_{t-1|t-1}\right) \cdot \mathbf{J}^{z(j+1)}_{t|t-1} \\
&= \text{cov}\left(\boldsymbol{H}^{(j)}_{t|t-1}\right) \cdot \mathbb{E}[\boldsymbol{W}^{z(j)}_{t|t-1}] \cdot \mathbf{J}^{z(j+1)}_{t|t-1},
\end{aligned}
$$

$$
\begin{aligned}
\operatorname{cov}\left(\boldsymbol{H}_{t|t-1}^{(j)}, \tilde{\boldsymbol{H}}_{t|t-1}^{(j+1)}\right) &= \operatorname{cov}\left(\boldsymbol{H}_{t|t-1}^{(j)}, \tilde{\tanh}(\boldsymbol{Z}_{t|t-1}^{h(j+1)})\right) \\
&= \operatorname{cov}\left(\boldsymbol{H}_{t|t-1}^{(j)}, \boldsymbol{Z}_{t|t-1}^{h(j+1)}\right) \cdot \mathbf{J}_{t|t-1}^{h(j+1)} \\
&= \operatorname{cov}\left(\boldsymbol{H}_{t|t-1}^{(j)}, \boldsymbol{W}_{t|t-1}^{h(j)} \cdot \boldsymbol{H}_{t|t-1}^{(j)} + \boldsymbol{U}_{t|t-1}^{h(j)} \cdot \boldsymbol{C}_{t|t-1}^{(j+1)}\right) \cdot \mathbf{J}_{t|t-1}^{h(j+1)} \\
&= \operatorname{cov}\left(\boldsymbol{H}_{t|t-1}^{(j)}, \boldsymbol{W}_{i,t|t-1}^{h(j)} \cdot \boldsymbol{H}_{t|t-1}^{(j)} + \boldsymbol{U}_{t|t-1}^{h(j)} \cdot (\boldsymbol{R}_{t|t-1}^{(j+1)} \odot \boldsymbol{H}_{t-1|t-1}^{(j+1)})\right) \cdot \mathbf{J}_{t|t-1}^{h(j+1)} \\
&= \operatorname{cov}\left(\boldsymbol{H}_{t|t-1}^{(j)}\right) \cdot \mathbb{E}[\boldsymbol{W}_{t|t-1}^{h(j)}] \cdot \mathbf{J}_{t|t-1}^{h(j+1)} \\
&+ \operatorname{cov}\left(\boldsymbol{H}_{t|t-1}^{(j)}, \boldsymbol{R}_{t|t-1}^{(j+1)} \odot \boldsymbol{H}_{t-1|t-1}^{(j+1)}\right) \cdot \mathbb{E}[\boldsymbol{U}_{t|t-1}^{h(j)}] \cdot \mathbf{J}_{t|t-1}^{h(j+1)} \\
&= \operatorname{cov}\left(\boldsymbol{H}_{t|t-1}^{(j)}\right) \cdot \mathbb{E}[\boldsymbol{W}_{t|t-1}^{h(j)}] \cdot \mathbf{J}_{t|t-1}^{h(j+1)} \\
&+ \operatorname{cov}\left(\boldsymbol{H}_{t|t-1}^{(j)}, \boldsymbol{R}_{t|t-1}^{(j+1)}\right) \cdot \operatorname{diag}(\mathbb{E}[\boldsymbol{H}_{t-1|t-1}^{(j+1)}]) \cdot \mathbb{E}[\boldsymbol{U}_{t|t-1}^{h(j)}] \cdot \mathbf{J}_{t|t-1}^{h(j+1)},
\end{aligned}
$$

and

$$
\operatorname{cov}\left(\boldsymbol{H}_{t|t-1}^{(j)}, \boldsymbol{R}_{t|t-1}^{(j+1)}\right) = \operatorname{cov}\left(\boldsymbol{H}_{t|t-1}^{(j)}\right) \cdot \mathbb{E}[\boldsymbol{W}_{t|t-1}^{r(j)}] \cdot \mathbf{J}_{t|t-1}^{r(j+1)}.
$$

B.2.2 Covariances Between the Parameters and Hidden States of Two Consecutive GRU Layers

The update gate:

$$
\begin{aligned}
\operatorname{cov}(\boldsymbol{W}_{i,t|t-1}^{z(j)}, H_{i,t|t-1}^{(j+1)}) &= \operatorname{cov}\left(\boldsymbol{W}_{i,t|t-1}^{z(j)}, H_{i,t-1|t-1}^{(j+1)} - Z_{i,t|t-1}^{(j+1)} \cdot H_{i,t-1|t-1}^{(j+1)} + Z_{i,t|t-1}^{(j+1)} \cdot \tilde{H}_{i,t|t-1}^{(j+1)}\right) \\
&= \operatorname{cov}\left(\boldsymbol{W}_{i,t|t-1}^{z(j)}, -Z_{i,t|t-1}^{(j+1)} \cdot H_{i,t-1|t-1}^{(j+1)}\right) + \operatorname{cov}\left(\boldsymbol{W}_{i,t|t-1}^{z(j)}, Z_{i,t|t-1}^{(j+1)} \cdot \tilde{H}_{i,t|t-1}^{(j+1)}\right) \\
&+ \cancelto{0}{\operatorname{cov}\left(\boldsymbol{W}_{i,t|t-1}^{z(j)}, H_{i,t-1|t-1}^{(j+1)}\right)} \\
&= -\operatorname{cov}\left(\boldsymbol{W}_{i,t|t-1}^{z(j)}, Z_{i,t|t-1}^{(j+1)}\right) \cdot \mathbb{E}[H_{i,t-1|t-1}^{(j+1)} - \cancelto{0}{\operatorname{cov}\left(\boldsymbol{W}_{i,t|t-1}^{z(j)}, H_{i,t-1|t-1}^{(j+1)}\right)} \cdot \mathbb{E}[Z_{i,t|t-1}^{(j+1)}] \\
&+ \cancelto{0}{\operatorname{cov}\left(\boldsymbol{W}_{i,t|t-1}^{z(j)}, \tilde{H}_{i,t|t-1}^{(j+1)}\right)} \cdot \mathbb{E}[Z_{i,t|t-1}^{(j+1)}] + \operatorname{cov}\left(\boldsymbol{W}_{i,t|t-1}^{z(j)}, Z_{i,t|t-1}^{(j+1)}\right) \cdot \mathbb{E}[\tilde{H}_{i,t|t-1}^{(j+1)}] \\
&= \operatorname{cov}\left(\boldsymbol{W}_{i,t|t-1}^{z(j)}, Z_{i,t|t-1}^{(j+1)}\right) \cdot \left(\mathbb{E}[\tilde{H}_{i,t|t-1}^{(j+1)}] - \mathbb{E}[H_{i,t-1|t-1}^{(j+1)}]\right) \\
&= \operatorname{cov}\left(\boldsymbol{W}_{i,t|t-1}^{z(j)}, \tilde{\sigma}(Z_{i,t|t-1}^{z(j+1)})\right) \cdot \left(\mathbb{E}[\tilde{H}_{i,t|t-1}^{(j+1)}] - \mathbb{E}[H_{i,t-1|t-1}^{(j+1)}]\right) \\
&= \operatorname{cov}\left(\boldsymbol{W}_{i,t|t-1}^{z(j)}\right) \cdot \mathbb{E}[\boldsymbol{X}_{t|t-1}^{(j)}] \cdot \mathrm{J}_{i,t|t-1}^{z(j+1)} \cdot \left(\mathbb{E}[\tilde{H}_{i,t|t-1}^{(j+1)}] - \mathbb{E}[H_{i,t-1|t-1}^{(j+1)}]\right) \\
\operatorname{cov}(\boldsymbol{U}_{i,t|t-1}^{z(j)}, H_{i,t|t-1}^{(j+1)}) &= \operatorname{cov}\left(\boldsymbol{U}_{i,t|t-1}^{z(j)}\right) \cdot \mathbb{E}[\boldsymbol{H}_{t-1|t-1}^{(j+1)}] \cdot \mathrm{J}_{i,t|t-1}^{z(j+1)} \cdot \left(\mathbb{E}[\tilde{H}_{i,t|t-1}^{(j+1)}] - \mathbb{E}[H_{i,t-1|t-1}^{(j+1)}]\right)
\end{aligned}
$$

The candidate gate:

$$
\begin{aligned}
\operatorname{cov}(\boldsymbol{W}^{h(j)}_{i,t|t-1}, H^{(j+1)}_{i,t|t-1}) &= \operatorname{cov}\left(\boldsymbol{W}^{h(j)}_{i,t|t-1}, H^{(j+1)}_{i,t-1|t-1} - Z^{(j+1)}_{i,t|t-1}\cdot H^{(j+1)}_{i,t-1|t-1} + Z^{(j+1)}_{i,t|t-1}\cdot \tilde{H}^{(j+1)}_{i,t|t-1}\right)\\
&= \operatorname{cov}\left(\boldsymbol{W}^{h(j)}_{i,t|t-1}, Z^{(j+1)}_{i,t|t-1}\cdot \tilde{H}^{(j+1)}_{i,t|t-1}\right)\\
&= \cancelto{0}{\operatorname{cov}\left(\boldsymbol{W}^{h(j)}_{i,t|t-1}, Z^{(j+1)}_{i,t|t-1}\right)\cdot \mathbb{E}[\tilde{H}^{(j+1)}_{i,t|t-1}]} + \operatorname{cov}\left(\boldsymbol{W}^{h(j)}_{i,t|t-1}, \tilde{H}^{(j+1)}_{i,t|t-1}\right)\cdot \mathbb{E}[Z^{(j+1)}_{i,t|t-1}]\\
&= \operatorname{cov}\left(\boldsymbol{W}^{h(j)}_{i,t|t-1}, \tilde{\tanh}(Z^{h(j+1)}_{i,t|t-1})\right)\cdot \mathbb{E}[Z^{(j+1)}_{i,t|t-1}]\\
&= \operatorname{cov}\left(\boldsymbol{W}^{h(j)}_{i,t|t-1}\right)\cdot \mathbb{E}[\boldsymbol{X}^{(j)}_{t|t-1}]\cdot \mathrm{J}^{h(j+1)}_{i,t|t-1}\cdot \mathbb{E}[Z^{(j+1)}_{i,t|t-1}].\\
\operatorname{cov}(\boldsymbol{U}^{h(j)}_{i,t|t-1}, H^{(j+1)}_{i,t|t-1}) &= \operatorname{cov}\left(\boldsymbol{U}^{h(j)}_{i,t|t-1}\right)\cdot \mathbb{E}[\boldsymbol{H}^{(j+1)}_{t-|t-1}]\cdot \mathrm{J}^{h(j+1)}_{i,t|t-1}\cdot \mathbb{E}[Z^{(j+1)}_{i,t|t-1}].
\end{aligned}
$$

The reset gate:

$$
\begin{aligned}
\operatorname{cov}(\boldsymbol{W}^{r(j)}_{i,t|t-1}, H^{(j+1)}_{i,t|t-1}) &= \operatorname{cov}\left(\boldsymbol{W}^{r(j)}_{i,t|t-1}, H^{(j+1)}_{i,t-1|t-1} - Z^{(j+1)}_{i,t|t-1}\cdot H^{(j+1)}_{i,t-1|t-1} + Z^{(j+1)}_{i,t|t-1}\cdot \tilde{H}^{(j+1)}_{i,t|t-1}\right)\\
&= \operatorname{cov}\left(\boldsymbol{W}^{r(j)}_{i,t|t-1}, Z^{(j+1)}_{i,t|t-1}\cdot \tilde{H}^{(j+1)}_{i,t|t-1}\right)\\
&= \cancelto{0}{\operatorname{cov}\left(\boldsymbol{W}^{r(j)}_{i,t|t-1}, Z^{(j+1)}_{i,t|t-1}\right)\cdot \mathbb{E}[\tilde{H}^{(j+1)}_{i,t|t-1}]} + \operatorname{cov}\left(\boldsymbol{W}^{r(j)}_{i,t|t-1}, \tilde{H}^{(j+1)}_{i,t|t-1}\right)\cdot \mathbb{E}[Z^{(j+1)}_{i,t|t-1}]\\
&= \operatorname{cov}\left(\boldsymbol{W}^{r(j)}_{i,t|t-1}, \tilde{\tanh}(Z^{h(j+1)}_{i,t|t-1})\right)\cdot \mathbb{E}[Z^{(j+1)}_{i,t|t-1}]\\
&= \operatorname{cov}\left(\boldsymbol{W}^{r(j)}_{i,t|t-1}, Z^{h(j+1)}_{i,t|t-1}\right)\cdot \mathrm{J}^{h(j+1)}_{i,t|t-1}\cdot \mathbb{E}[Z^{(j+1)}_{i,t|t-1}]\\
&= \operatorname{cov}\left(\boldsymbol{W}^{r(j)}_{i,t|t-1}, \boldsymbol{W}^{h(j)}_{i,t|t-1}\cdot \boldsymbol{X}^{(j)}_{t|t-1} + \boldsymbol{U}^{h(j)}_{i,t|t-1}\cdot \boldsymbol{C}^{(j+1)}_{t|t-1}\right)\cdot \mathrm{J}^{h(j+1)}_{i,t|t-1}\cdot \mathbb{E}[Z^{(j+1)}_{i,t|t-1}]\\
&= \operatorname{cov}\left(\boldsymbol{W}^{r(j)}_{i,t|t-1}, \boldsymbol{W}^{h(j)}_{i,t|t-1}\cdot \boldsymbol{X}^{(j)}_{t|t-1} + \boldsymbol{U}^{h(j)}_{i,t|t-1}\cdot (\boldsymbol{R}^{(j+1)}_{t|t-1}\odot \boldsymbol{H}^{(j+1)}_{t-1|t-1})\right)\cdot \mathrm{J}^{h(j+1)}_{i,t|t-1}\mathbb{E}[Z^{(j+1)}_{i,t|t-1}]\\
&= \operatorname{cov}\left(\boldsymbol{W}^{r(j)}_{i,t|t-1}\right)\cdot \mathbb{E}[\boldsymbol{X}^{(j)}_{t|t-1}]\cdot \mathrm{J}^{r(j+1)}_{i,t|t-1}\cdot \mathbb{E}[H^{(j+1)}_{i,t-1|t-1}]\cdot \mathbb{E}[U^{h(j)}_{ii,t|t-1}]\cdot \mathrm{J}^{h(j+1)}_{i,t|t-1}\mathbb{E}[Z^{(j+1)}_{i,t|t-1}]\\
\operatorname{cov}(\boldsymbol{U}^{r(j)}_{i,t|t-1}, H^{(j+1)}_{i,t|t-1}) &= \operatorname{cov}\left(\boldsymbol{U}^{r(j)}_{i,t|t-1}\right)\cdot \mathbb{E}[\boldsymbol{H}^{(j)}_{t-1|t-1}]\cdot \mathrm{J}^{r(j+1)}_{i,t|t-1}\cdot \mathbb{E}[H^{(j+1)}_{i,t-1|t-1}]\cdot \mathbb{E}[U^{h(j)}_{ii,t|t-1}]\cdot \mathrm{J}^{h(j+1)}_{i,t|t-1}\mathbb{E}[Z^{(j+1)}_{i,t|t-1}],
\end{aligned}
$$

where $U^{h(j)}_{ii,t|t-1}$ is the element at the i^{th} row and the i^{th} column of the weight matrix $\boldsymbol{U}^{h(j)}_{t|t-1}$.

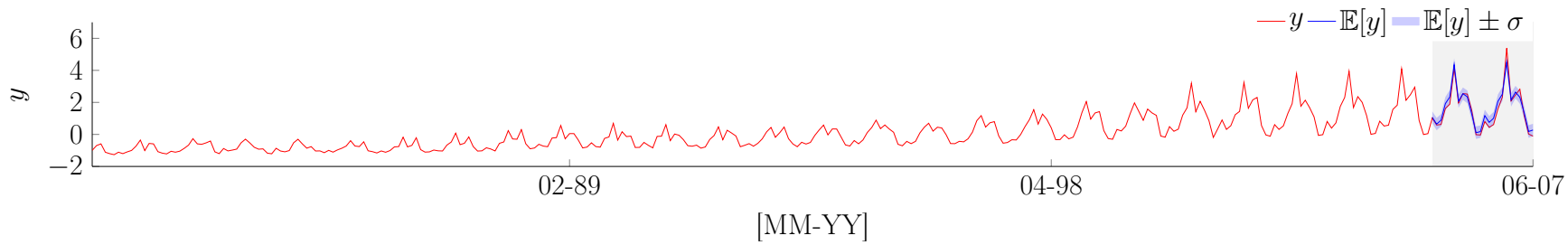
APPENDIX C ARCHITECTURE AND HYPERPARAMETERS FOR THE TAGI-SKF MODEL

This appendix presents the architecture and hyperparameters for the TAGI-SKF model used in the experiments presented in Section 5.3.

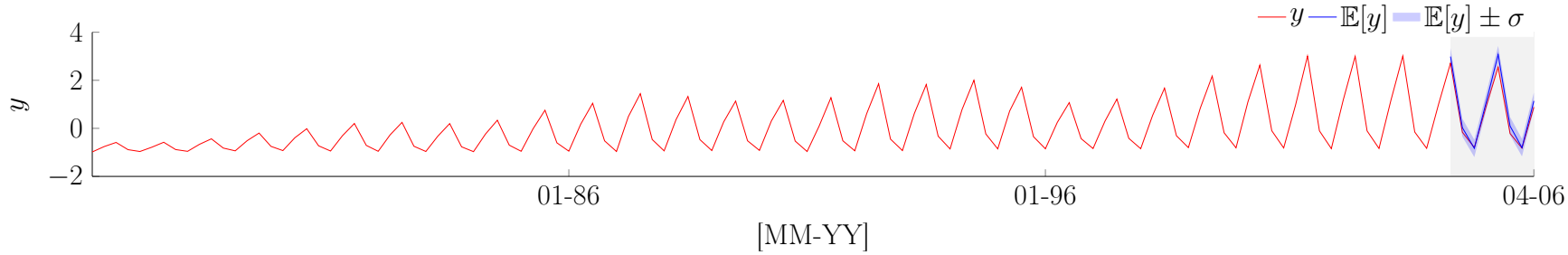
Table C.1 Architecture and hyperparameters for the TAGI-SKF models used in our experiments.

Dataset	Synthetic data	Case study #1	Case study #2	Case study #3
# LSTM layer	1	1	1	1
# LSTM nodes	50	50	50	50
Lookback length `W`	52	52	52	52
Lookback length `M` (covariates)	n/a	n/a	10	n/a
σ_W	0	0	0	0
σ_V	0.2	0.3	0.1	0.05

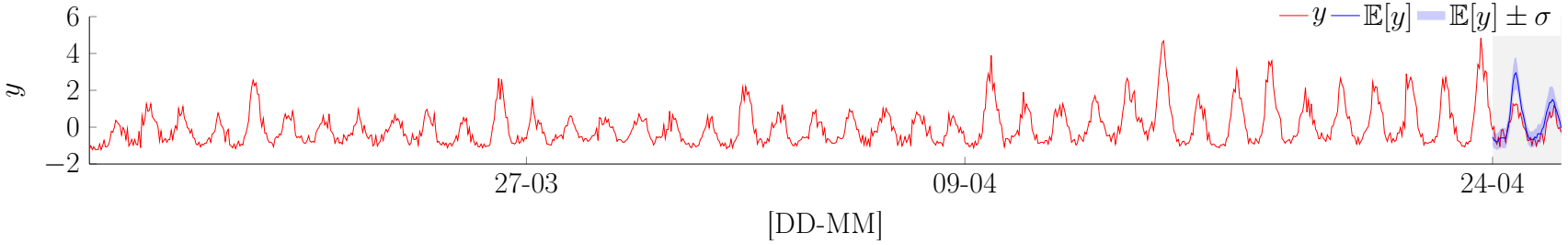
APPENDIX D ZOOM-IN FIGURES IN SECTION 4.3.1



(a) Time series #32 – Tourism (monthly)



(b) Time series #33 – Tourism (quarterly)



(c) Time series #375 – M4 (hourly)

Figure D.1 Zoom-in figures for for Figures 4.2 (a), (e) and (i), respectively

APPENDIX E HYPERPARAMETERS TUNING FOR OTHER ANOMALY DETECTION METHODS

This appendix presents how to obtain the hyperparameters for the Prophet, Matrix Profile, kNN, Autoencoder, and VAE methods for detecting anomalies in the experiments presented in Section 5.3. The results for the Prophet are obtained using the *Prophet* library [97], while the ones for the Matrix Profile is obtained using the *stumpy* library [104], whereas the results for the kNN, Autoencoder, and VAE are obtained using the *pyod* library [124].

For all of these methods, we grid-search their hyperparameters as presented in Tables E.1 and E.2. Beside the hyperparameters that are being gridsearched, we use default values for others. We aim at reaching zero false alarm for the training data that is assumed to be anomaly-free. However, when this criteria could not be fulfilled, we choose the hyperparameters that leads to a minimal false alarm rate. The hyperparameters-tuning procedure are described following below steps

- Step 1: Define a grid for each hyperparameter.
- Step 2: Using the anomaly-free training data, compute the false alarm rate per ten years associated with each set of hyperparameters. Choose the optimal set of hyperparameters such that it gives the smallest false alarm rate. If there are more than one sets which give the same smallest false alarm rate, proceed with step 3.
- Step 3: Using the abnormal synthetic time series that are also used to obtain the parameters $\boldsymbol{\mathcal{P}}$ for the TAGI-SKF method, compute the detection probability associated with each set of hyperparameters. Choose the optimal set such that it gives the highest detection probability. If there are more than one sets which give the same highest probability, randomly choose one set among them.

Table E.1 Hyperparameters for the Prophet model.

	Hyperparameters	
Experiment	Interval width	
	Grid	Optimal
Synthetic data	$\{0.95, 0.99, 0.995, 0.999\}$	0.999
Case study #1	$\{0.95, 0.99, 0.995, 0.999\}$	0.999
Case study #2	$\{0.95, 0.99, 0.995, 0.999\}$	0.99
Case study #3	$\{0.95, 0.99, 0.995, 0.999\}$	0.999

Table E.2 Hyperparameters for the Matrix profile, kNN, Autoencoder, and VAE models. N/a means that the hyperparameter is not applied for the method.

Method	Experiment	Hyperparameters					
		Lookback window `W`		Contamination		# neighbours	
		Grid	Optimal	Grid	Optimal	Grid	Optimal
Matrix Profile	Synthetic data	$\{5, 12, 26, 52\}$	26	n/a	n/a	n/a	n/a
	Case study #1	$\{5, 12, 26, 52\}$	52	n/a	n/a	n/a	n/a
	Case study #2	$\{5, 12, 26, 52\}$	52	n/a	n/a	n/a	n/a
	Case study #3	$\{5, 12, 26, 52\}$	26	n/a	n/a	n/a	n/a
kNN	Synthetic data	$\{5, 12, 26, 52\}$	26	$\{0.05, 0.1, 0.15, 0.2\}$	0.2	$\{5, 10, 15\}$	10
	Case study #1	$\{5, 12, 26, 52\}$	52	$\{0.05, 0.1, 0.15, 0.2\}$	0.2	$\{5, 10, 15\}$	10
	Case study #2	$\{5, 12, 26, 52\}$	26	$\{5, 12, 26, 52\}$	26	$\{5, 12, 26, 52\}$	26
	Case study #3	$\{5, 12, 26, 52\}$	26	$\{5, 12, 26, 52\}$	26	$\{5, 12, 26, 52\}$	26
Autoencoder	Synthetic data	$\{5, 12, 26, 52\}$	5	$\{0.001, 0.005, 0.01, 0.015\}$	0.005	n/a	n/a
	Case study #1	$\{5, 12, 26, 52\}$	5	$\{0.001, 0.005, 0.01, 0.015\}$	0.001	n/a	n/a
	Case study #2	$\{5, 12, 26, 52\}$	5	$\{0.001, 0.005, 0.01, 0.015\}$	0.001	n/a	n/a
	Case study #3	$\{5, 12, 26, 52\}$	5	$\{0.001, 0.005, 0.01, 0.015\}$	0.005	n/a	n/a
VAE	Synthetic data	$\{5, 12, 26, 52\}$	5	$\{0.001, 0.005, 0.01, 0.015\}$	0.001	n/a	n/a
	Case study #1	$\{5, 12, 26, 52\}$	52	$\{0.001, 0.005, 0.01, 0.015\}$	0.001	n/a	n/a
	Case study #2	$\{5, 12, 26, 52\}$	52	$\{0.001, 0.005, 0.01, 0.015\}$	0.001	n/a	n/a
	Case study #3	$\{5, 12, 26, 52\}$	5	$\{0.001, 0.005, 0.01, 0.015\}$	0.001	n/a	n/a