Multi-step Deep Autoregressive Forecasting with Latent States

Xiaoyong Jin [1] **Shiyang Li** [1] **Yunkai Zhang** [1] **Xifeng Yan** [1]

Abstract

In many forecasting applications, it is important to predict an overall trajectory within a period of time, i.e. a forecasting horizon. Various deep neural network models have been introduced to tackle the so-called multi-step forecasting problem. However, most existing approaches either recursively apply a powerful but complex single-step model or explicitly presume that a relatively simple yet restrictive latent process determines the temporal structure. In this paper, we propose a new framework that combines the advantages of both categories and avoids their weaknesses. We maintain a complex latent process in which each state is implicitly connected through a deep temporal convolutional network with gated recurrent structures between layers in substitution of vanilla recurrent networks. Further, multi-step predictions are made simultaneously by taking advantage of known features such as datetime. We evaluate our model on a number of public benchmark datasets with forecasting horizons of varying length to demonstrate not only its outstanding performance in terms of accuracy but also faster evaluation with its fully-parallelizable architecture.

1. Introduction

Multi-step ahead forecasting is a challenging but critical task with various practical applications, from resource management to business decision making. The forecasting problem is essentially a probabilistic sequence modeling task in which the goal is to capture the stochastic evolution of a time series. Among a large number of traditional methods, State Space Models (SSMs) and AutoRegressive (AR) models are usually preferred. Recently deep neural networks have been a powerful workhorse to instantiate these methods. Most existing works employ a Recurrent Neural Network (RNN) (Lai et al., 2018; Li et al., 2017; Yu et al., 2017; Flunkert et al., 2017) or a Temporal Convolutional Network (TCN) (Borovykh et al., 2017) to model an AR process. These approaches decompose the multi-step forecasting task into a number of single-step predictions where a sequence of history is mapped to one prediction at each time step. Although they are able to accommodate extremely complicated dynamics, a large neural network has to be applied recursively to obtain multiple steps of predictions, which is computationally expensive and difficult to parallelize. In addition, these methods can propagate error from one prediction to the next due to dependence among steps (Fox et al., 2018). Some works extend traditional linear Gaussian SSMs via nonlinear neural networks. One approach is to replace linear transition with Multi-layer Perceptrons (MLPs) (Krishnan et al., 2017) or RNNs (Chung et al., 2015). (Fraccaro et al., 2016) instead incorporates deterministic hidden states of an underlying RNN into the latent process. In contrast, (Rangapuram et al., 2018) keeps the linear Gaussian structure but uses an RNN to specify the transition parameters.

We propose a novel framework that fuses the advantages of both SSMs and AR models. The framework introduces a latent state at each time step within the forecasting horizon. It employs a novel **Layer Recurrent TCN** (**LRTCN**) encoder, which maps a range of observations to a stochastic latent state in an autoregressive fashion. It captures temporal patterns and long-term dependencies in the history with TCNs and then implicitly restores the temporal order with recurrent connections between TCN layers. Furthermore, we replace sequential latent process with a straightforward translation step by taking advantage of known temporal features so that multiple predictions can be made in parallel.

2. Methodology

In multi-step ahead forecasting tasks, given a sequence of observations $\mathbf{x}_{1:t}$, the goal is to predict τ steps in the future, i.e. $\mathbf{x}_{t+1:t+\tau}$. We call $\mathbf{x}_{1:t}$ the observed history and τ the forecasting horizon. Formally, we want to model a conditional predictive distribution $p(\mathbf{x}_{t+1:t+\tau}|\mathbf{x}_{1:t})$. State Space Models introduce a series of latent state variables $\mathbf{z}_{t+1:t+\tau}$ to encode temporal patterns (Hyndman et al., 2008). They

*Equal contribution [1]Department of Computer Science, University of California, Santa Barbara, USA. Correspondence to: Xiaoyong Jin <x_jin@cs.ucsb.edu>, Xifeng Yan <xyan@cs.ucsb.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

decompose the predictive model into three modules:

$$\begin{aligned} \mathbf{z}_{t+i-1} &\sim p(\mathbf{z}_{t+i-1}|\mathbf{x}_{1:t+i-1}); \\ \mathbf{z}_{t+i} &\sim p(\mathbf{z}_{t+i}|\mathbf{z}_{t+i-1}); \\ \mathbf{x}_{t+i} &\sim p(\mathbf{x}_{t+i}|\mathbf{z}_{t+i}); \end{aligned} \tag{1}$$

for $i = 1, 2, \cdots, \tau$. Here $\mathbf{z}_{t+i-1} \sim p(\mathbf{z}_{t+i-1}|\mathbf{x}_{1:t+i-1})$ is a filtered posterior distribution of $\mathbf{z}_{t+i-1}$ that summarizes the history. $p(\mathbf{z}_{t+i}|\mathbf{z}_{t+i-1})$ models a transition mechanism that determines the latent dynamic. $p(\mathbf{x}_{t+i}|\mathbf{z}_{t+i})$ is an emission system that produces predictions conditioned on the corresponding latent state at each step. We propose to replace the recursive latent dynamic $p(\mathbf{z}_{t+i}|\mathbf{z}_{t+i-1})$ in (1) with a straightforward translation from $\mathbf{z}_t$

$$\mathbf{z}_{t+i} \sim p(\mathbf{z}_{t+i}|\mathbf{z}_t, \mathbf{e}_{t+1:t+i}), \quad i = 1, \cdots, \tau, \tag{2}$$

by taking advantage of a series of known covariates $\mathbf{e}_{t+1:t+i}$ that indicate common patterns such as daily or weekly seasonality and a stochastic encoding of given history $\mathbf{z}_t$. The translation is intuitively a guess based on our knowledge about the impact of various factors inside the covariates upon the current state. For example, suppose that $\mathbf{z}_t$ carries the information about level and trend at time step t, one can estimate the level at a future time step with the elapsed time from t contained in covariates. This estimation is reasonable as long as the patterns in the past remain stationary within the forecasting horizon, which is a common assumption in forecasting.

Recent findings have shown that a single $\mathbf{z}_t$ might be ignored by the translation (Bowman et al., 2015; Yang et al., 2017). To alleviate this, we employ a modified translation (4) with dependence on $\mathbf{x}_{1:t}$ and thus get the following predictive process:

$$\mathbf{z}_t \sim p(\mathbf{z}_t|\mathbf{x}_{1:t}); \tag{3}$$

$$\mathbf{z}_{t+i} \sim p(\mathbf{z}_{t+i}|\mathbf{z}_t; \mathbf{x}_{1:t}, \mathbf{e}_{t+1:t+i}); \tag{4}$$

$$\mathbf{x}_{t+i} \sim p(\mathbf{x}_{t+i}|\mathbf{z}_{t+i}), \quad i = 1, \cdots, \tau. \tag{5}$$

From a Bayesian perspective, we call (4) a **prior network**, because it models priors over the latent states before we observe the ground truths within the horizon. We also name (5) an **emission network** as in SSMs. Furthermore, combining the prior network and the emission network makes a **generative network** because it describes the generative process of the time series.

Since exact inference for (1) is intractable, the model cannot be learned by directly maximizing the likelihood of ground truths. Stochastic variational inference (Hoffman et al., 2013) provides an alternative that maximizes a lower bound of likelihood by introducing an **inference network** to approximate the latent posterior with another probabilistic model $q(\mathbf{z}_{t+i}|\mathbf{x}_{1:t+i})$. Note that (3) has the same dependence structure as the inference network when $i = 0$, as it is also a filtered posterior of latent state at t. Hence we also apply the inference network to (3).

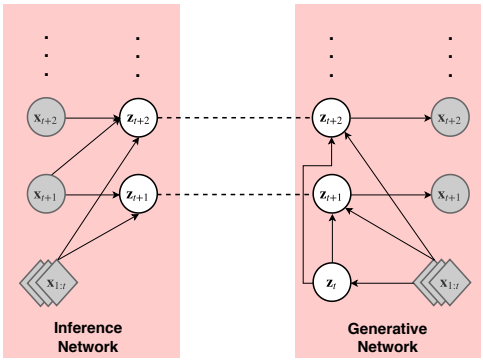

Figure 1. Graphical view of inference network (left) and generative network (right). Plain circles are latent states, shaded circles are observable data, and diamonds are deterministic history. The approximate posterior of $\mathbf{z}_{t+i}$ is conditioned on $\mathbf{x}_{1:t+i}$, while the prior only depends on $\mathbf{x}_{1:t}$ and $\mathbf{z}_t$. Dashed lines connect priors and their posterior counterparts.

We use deep neural nets to parameterize all networks above. An overview of the framework is illustrated in Figure 1.

2.1. Emission Network

The emission network parameterizes the conditional distribution (5) through a neural network with learnable parameters $\boldsymbol{\theta}_d$. The type of distribution can be chosen with flexibility according to data being modeled. For example, we can employ negative binomial distribution for positive count data and beta distribution for percentage data (Flunkert et al., 2017). For simplicity, we only illustrate Gaussian distribution:

$$\mathbf{x}_{t+i} = \boldsymbol{\mu}_{\boldsymbol{\theta}_d}(\mathbf{z}_{t+i}) + \boldsymbol{\sigma}_{\boldsymbol{\theta}_d}(\mathbf{z}_{t+i}) \cdot \boldsymbol{\varepsilon}_{t+i} \tag{6}$$

where $\boldsymbol{\mu}_{\boldsymbol{\theta}_d}, \boldsymbol{\sigma}_{\boldsymbol{\theta}_d}$ are feed-forward networks with one or more hidden layers, $\boldsymbol{\varepsilon}_{t+i} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. For standard deviation $\boldsymbol{\sigma}_{\boldsymbol{\theta}_d}$, we add a softplus activation function to the last hidden output in order to ensure positivity of the standard deviation. Note that $\boldsymbol{\theta}_d$ is shared across the forecasting horizon, as latent states encode all time-specific information.

2.2. Inference Network

The inference network maps a range of history to a latent variable. Instead of commonly-used RNNs, Temporal convolutional networks (TCN) has been proven to be more effective and efficient in many sequential modeling tasks (Bai et al., 2018). Although TCNs do not have a sense of temporal order because of their locality, the higher-level representations in TCNs have access to longer history. Hence the convolutional feature maps at different layers implic-

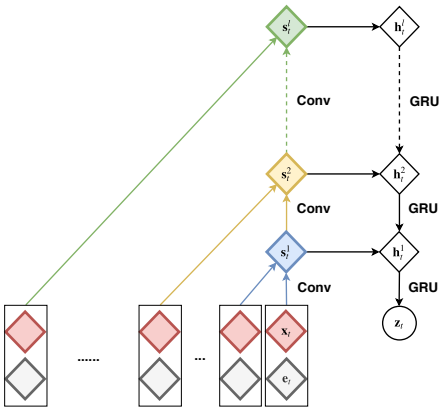

Figure 2. An architectural view of an LRTCN encoder. The TCN representations at different layers are consumed by a GRU network. As the upper layers have access to distant history and the lower layers focus on recent observations, the top-down hierarchy implicitly keeps the temporal order of the input sequence. We also feed covariates into the TCN associated with observations.

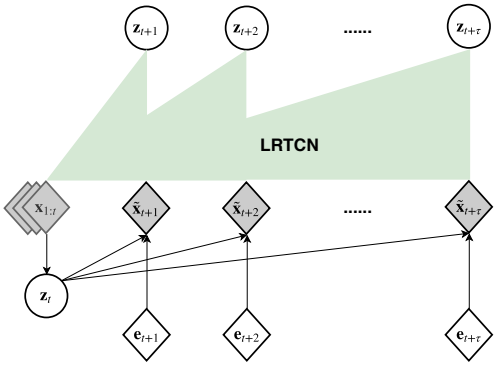

Figure 3. The prior network. The estimations $\tilde{\mathbf{x}}_{t+1:t+\tau}$ and the history $\mathbf{x}_{1:t}$ are concatenated and fed into the LRTCN with covariates as in the inference model.

itly form a temporal hierarchy in a top-down fashion. We propose to explicitly reconstruct the temporal structure by stacking a top-down RNN layer over the hierarchy. A systematical illustration is shown in Figure 2. In our experiments, we choose to use GRU cells (Cho et al., 2014) to instantiate the RNN. Note that we also use covariates as an additional input associated with the observations to the TCN module. These time-based covariates can be regarded as "positional embeddings" that expose the temporal information to order-insensitive TCNs (Gehring et al., 2017). We use $\boldsymbol{\theta}_c$ to denote all trainable parameters in an LRTCN.

As shown in Figure 2, the approximated posterior of $\mathbf{z}_{t+i}$ is derived from the RNN output at the first layer of LRTCN, and similar to the emission network, we apply feed-forward networks with parameter $\boldsymbol{\theta}_e$ to obtain the state

$$\begin{aligned} \mathbf{h}_{t+i}^{(1)} &= \text{LRTCN}(\mathbf{x}_{1:t+i}, \mathbf{e}_{1:t+i}; \boldsymbol{\theta}_c) \\ \mathbf{z}_{t+i} &= \boldsymbol{\mu}_{\boldsymbol{\theta}_e}\left(\mathbf{h}_{t+i}^{(1)}\right) + \boldsymbol{\sigma}_{\boldsymbol{\theta}_e}\left(\mathbf{h}_{t+i}^{(1)}\right) \cdot \boldsymbol{\varepsilon}_{t+i}. \end{aligned} \tag{7}$$

Note that the inference process resembles a Bayes filter of a recursive latent process. This implies the implicit temporal connection among latent states (and the intermediate state) even if they are conditionally independent in (4).

2.3. Prior Network

Notice that (4) also explicitly attend to past observations except that it should never have access to the current time step or unknown future. Therefore, we apply the same architecture as the inference network to the prior network but first roughly estimate $\mathbf{x}_{t+1:t+\tau}$ with corresponding covariates and $\mathbf{z}_t$ through another group of feed-forward networks $f_{\boldsymbol{\theta}_g}$

$$\tilde{\mathbf{x}}_{t+i} = f_{\boldsymbol{\theta}_g}\left(\mathbf{z}_t, \mathbf{e}_{t+i}\right) \quad i = 1, \cdots, \tau, \tag{8}$$

where $\boldsymbol{\theta}_g$ contains trainable parameters. $\boldsymbol{\theta}_g$ is shared across the forecasting horizon as in the emission network. We then apply LRTCN to the concatenation $[\mathbf{x}_{1:t}; \tilde{\mathbf{x}}_{t+1:t+\tau}]$ along with $\mathbf{e}_{1:t+\tau}$ as in the inference network. We illustrate an overview of this structure in Figure 3.

Finally, we distinguish priors from posteriors by introducing different feed-forward networks with parameter $\boldsymbol{\theta}_p$ to shape the Gaussian priors

$$\begin{aligned} \tilde{\mathbf{h}}_{t+i}^{(1)} &= \text{LRTCN}(\mathbf{x}_{1:t}, \tilde{\mathbf{x}}_{t+1:t+i}, \mathbf{e}_{1:t+i}; \boldsymbol{\theta}_c) \\ \mathbf{z}_{t+i} &= \boldsymbol{\mu}_{\boldsymbol{\theta}_p}\left(\tilde{\mathbf{h}}_{t+i}^{(1)}\right) + \boldsymbol{\sigma}_{\boldsymbol{\theta}_p}\left(\tilde{\mathbf{h}}_{t+i}^{(1)}\right) \cdot \boldsymbol{\varepsilon}_{t+i}. \end{aligned} \tag{9}$$

2.4. Training

To learn the model, we are given a time series dataset $\{\mathbf{x}_{1:T}^{(n)}\}_{n=1}^{M}$ and associated covariates $\{\mathbf{e}_{1:T}^{(n)}\}_{n=1}^{M}$, where T is the length of all available observations and M is the number of different time series. We create training instances by selecting windows with fixed history length t and forecasting horizon τ but varying the start point of forecasting from each of the original long time series (Flunkert et al., 2017). As a result, we get a training dataset with N sliding windows $\{\mathbf{x}_{1:t+\tau}^{(n)}, \mathbf{e}_{1:t+\tau}^{(n)}\}_{n=1}^{N}$.

The Evidence Lower Bound (ELBO) (Hoffman et al., 2013) of the log-likelihood can be derived as:

$$\log p(\mathbf{x}_{t+1:t+\tau}^{(n)} | \mathbf{x}_{1:t}^{(n)}) \geq -\mathcal{L}_{\text{NLL}}^{(n)} - \mathcal{L}_{\text{KL}}^{(n)} \tag{10}$$

		ARIMA		ETS		TRMF	DeepAR		DeepSSM		Ours	
Dataset	Horizon	$R_{0.5}$	$R_{0.9}$	$R_{0.5}$	$R_{0.9}$	$R_{0.5}$	$R_{0.5}$	$R_{0.9}$	$R_{0.5}$	$R_{0.9}$	$R_{0.5}$	$R_{0.9}$
Electricity	1 day	0.154	0.102	0.101	0.077	0.084	0.075	0.040	0.083	0.056	**0.073**	**0.038**
	1 week	0.30	0.110	0.130	0.110	0.087	0.125	0.080	0.085	0.057	**0.085**	**0.053**
Traffic	1 day	0.223	0.137	0.236	0.148	0.186	0.161	**0.099**	0.167	0.113	**0.146**	0.105
	1 week	0.501	0.298	0.532	0.60	0.202	0.219	0.138	**0.168**	0.114	0.169	**0.113**
M4	2 days	0.052	0.035	0.054	0.027	0.057	0.090	0.030	0.044	0.027	**0.037**	**0.019**

Table 1. Result summary of short-term (1-2 days) and long-term (1 week) forecasting.

where

$$\mathcal{L}_{\text{NLL}}^{(n)} = -\sum_{i=1}^{\tau} E_{q(\mathbf{z}_{t+i}^{(n)})}\left[\log p(\mathbf{x}_{t+i}^{(n)}|\mathbf{z}_{t+i}^{(n)})\right]$$
$$\mathcal{L}_{\text{KL}}^{(n)} = E_{q(\mathbf{z}_t^{(n)})}\sum_{i=1}^{\tau}\text{KL}\left(q(\mathbf{z}_{t+i}^{(n)}|\mathbf{x}_{1:t+i}^{(n)})\|p(\mathbf{z}_{t+i}^{(n)}|\mathbf{z}_t^{(n)})\right). \tag{11}$$

Hence we define our loss function

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}\left(\mathcal{L}_{\text{NLL}}^{(n)} + \mathcal{L}_{\text{KL}}^{(n)}\right) \tag{12}$$

and learn the prior, emission and inference network jointly by minimizing the loss w.r.t their parameters, namely $\boldsymbol{\theta}_g, \boldsymbol{\theta}_p, \boldsymbol{\theta}_d, \boldsymbol{\theta}_e$ and $\boldsymbol{\theta}_c$.

3. Experiments

We conducted experiments on three public benchmark datasets *electricity* [1], *traffic* [2] and *M4-hourly* [3]. Each dataset is split into a training set, a validation set and a test set in chronological order as in (Yu et al., 2016). We compare our method with shallow methods ARIMA, exponential smoothing (ETS) and TRMF (Yu et al., 2016), as well as recent deep models DeepAR (Flunkert et al., 2017) and Deep SSM (Rangapuram et al., 2018). For fair comparison, we use ρ-quantile risk to evaluate the prediction accuracy. The ρ-quantile risk R_ρ with $\rho \in (0, 1)$ is defined as:

$$R_\rho(\mathbf{x}, \hat{\mathbf{x}}) = \frac{2\sum_{i,t}(\rho - \mathbf{1}_{x\le\hat{x}})(x_t^{(i)} - \hat{x_t}^{(i)})}{\sum_{i,t}|x_t^{(i)}|},$$

where $\hat{x}$ is the empirical ρ-quantile of the predictive distribution.

Table 1 summarizes the experimental results. Generally, our model achieves better or competitive results. All models have similar number of parameters.

[1] https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014

[2] http://archive.ics.uci.edu/ml/datasets/PEMS-SF

[3] https://github.com/M4Competition/M4-methods/tree/master/Dataset

We also compare the evaluation speed of our model with DeepAR, a representative recurrent model that makes predictions step-by-step. After both models are trained, we predict 100 time series from *electricity* dataset for a number of steps on a single Nvidia GTX 1080 Ti GPU. For each prediction, 200 samples are drawn. We repeat the evaluation 10 times and report the average elapsed time in Figure 4. Notice that our model is much faster than DeepAR. Moreover, the evaluation time of our model almost remains constant as the forecasting horizon grows. In contrast, the time cost of DeepAR increases linearly due to sequential generation.

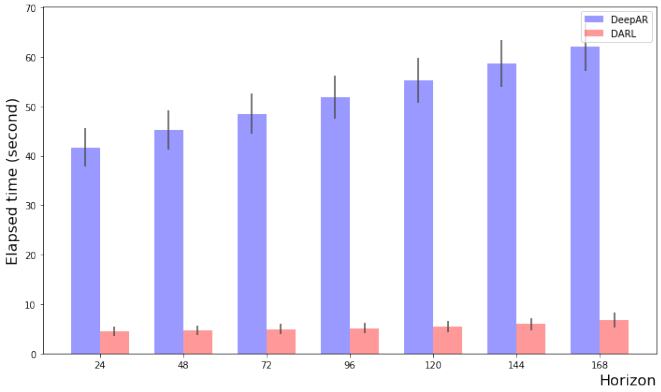

Figure 4. The evaluation time of DeepAR and our model on *electricity* dataset. Vertical black lines indicate standard deviations. The time cost of DeepAR increases linearly while our model is much more efficient.

4. Conclusions

We present a new deep learning framework for multi-step ahead time series forecasting task that combines the strengths of both autoregressive models and state space models. We avoid step-by-step generation by a fully-parallelizable latent process. Our model is able to achieve competitive or even better performance on a variety of datasets.

References

Bai, S., Kolter, J. Z., and Koltun, V. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv:1803.01271 [cs]*, March 2018. URL http://arxiv.org/abs/1803.01271. arXiv: 1803.01271.

Borovykh, A., Bohte, S., and Oosterlee, C. W. Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*, 2017.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A., and Bengio, Y. A Recurrent Latent Variable Model for Sequential Data. *arXiv:1506.02216 [cs]*, June 2015. URL http://arxiv.org/abs/1506.02216. arXiv: 1506.02216.

Flunkert, V., Salinas, D., and Gasthaus, J. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *arXiv preprint arXiv:1704.04110*, 2017.

Fox, I., Ang, L., Jaiswal, M., Pop-Busui, R., and Wiens, J. Deep Multi-Output Forecasting: Learning to Accurately Predict Blood Glucose Trajectories. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining - KDD '18*, pp. 1387–1395, London, United Kingdom, 2018. ACM Press. ISBN 978-1-4503-5552-0. doi: 10.1145/3219819.3220102. URL http://dl.acm.org/citation.cfm?doid=3219819.3220102.

Fraccaro, M., Snderby, S. K., Paquet, U., and Winther, O. Sequential Neural Models with Stochastic Layers. *arXiv:1605.07571 [cs, stat]*, May 2016. URL http://arxiv.org/abs/1605.07571. arXiv: 1605.07571.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. Convolutional Sequence to Sequence Learning. *arXiv:1705.03122 [cs]*, May 2017. URL http://arxiv.org/abs/1705.03122. arXiv: 1705.03122.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347, 2013. URL http://jmlr.org/papers/v14/hoffman13a.html.

Hyndman, R., Koehler, A. B., Ord, J. K., and Snyder, R. D. *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media, 2008.

Krishnan, R. G., Shalit, U., and Sontag, D. Structured inference networks for nonlinear state space models. In *AAAI*, pp. 2101–2109, 2017.

Lai, G., Chang, W.-C., Yang, Y., and Liu, H. Modeling long- and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 95–104. ACM, 2018.

Li, Y., Yu, R., Shahabi, C., and Liu, Y. Graph convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*, 2017.

Rangapuram, S. S., Seeger, M. W., Gasthaus, J., Stella, L., Wang, Y., and Januschowski, T. Deep state space models for time series forecasting. In *Advances in Neural Information Processing Systems*, pp. 7796–7805, 2018.

Yang, Z., Hu, Z., Salakhutdinov, R., and Berg-Kirkpatrick, T. Improved Variational Autoencoders for Text Modeling using Dilated Convolutions. *arXiv:1702.08139 [cs]*, February 2017. URL http://arxiv.org/abs/1702.08139. arXiv: 1702.08139.

Yu, H.-F., Rao, N., and Dhillon, I. S. Temporal regularized matrix factorization for high-dimensional time series prediction. In *Advances in neural information processing systems*, pp. 847–855, 2016.

Yu, R., Zheng, S., Anandkumar, A., and Yue, Y. Long-term forecasting using tensor-train rnns. *arXiv preprint arXiv:1711.00073*, 2017.