Think Globally, Act Locally: A Deep Neural Network Approach to High-Dimensional Time Series Forecasting

Rajat Sen [1] **Hsiang-Fu Yu** [2] **Inderjit Dhillon** [1]

Abstract

Forecasting high-dimensional time series plays a crucial role in many applications such as demand forecasting and financial predictions. Modern real-world datasets can have millions of correlated time-series that evolve together, i.e they are extremely high dimensional (one dimension for each individual time-series). Thus there is need for exploiting these global patterns and coupling them with local calibration for better prediction. However, most recent deep learning approaches in the literature are one-dimensional, i.e, even though they are trained on the whole dataset, during prediction, the future forecast for a single dimension mainly depends on past values from the same dimension. In this paper, we seek to correct this deficiency and propose DeepGLO, a deep forecasting model which *thinks globally and acts locally*. In particular, DeepGLO is a hybrid model that combines a *global* matrix factorization model regularized by a temporal deep network with a *local* deep temporal model that captures patterns specific to each dimension. The global and local models are combined via a data-driven attention mechanism for each dimension. Empirical results demonstrate that DeepGLO outperforms state-of-the-art approaches on various datasets; for example, we see more than 30% improvement in WAPE over other methods on a real-world dataset that contains more than 100K-dimensional time series.

1. Introduction

Time-series forecasting is an important problem with many industrial applications like retail demand forecasting (Seeger et al., 2016), financial predictions (Kim, 2003), predicting traffic or weather patterns (Chatfield, 2000). In general it plays a key role in automating business processes (Larson, 2001). Modern data-sets can have millions of correlated time-series over several thousand time-points. For instance, in an online shopping portal like Amazon or Walmart, one may be interested in the future daily demands for all items in a category, where the number of items may be in millions. This leads to a problem of forecasting n time-series (one for each of the n items), given past demands over t time-steps. Such a time series data-set can be represented as a matrix $\mathbf{Y} \in \mathbb{R}^{n \times t}$ and we are interested in the *high-dimensional* setting where n can be of the order of millions.

Traditional time-series forecasting methods include the well known AR, ARIMA and exponential smoothing (McKenzie, 1984), the classical Box-Jenkins methodology (Box & Jenkins, 1968) and more generally the linear state-space models (Hyndman et al., 2008). However, these methods are not easily scalable to large data-sets with millions of time-series, owing to the need for individual training. Moreover, they cannot benefit from shared temporal patterns in the whole data-set while training and prediction.

Deep networks have gained popularity in time-series forecasting recently. Recurrent Neural Networks (RNN's) (Funahashi & Nakamura, 1993) have been popular in sequential modeling, however they suffer from the gradient vanishing/exploding problems in training. Long Short Term Memory (LSTM) (Gers et al., 1999; Sundermeyer et al., 2012) networks alleviate that issue and have had recent success in time-series modeling (Flunkert et al., 2017; Rangapuram et al., 2018). Another popular architecture, that is competitive with LSTM's and arguably easier to train are temporal convolutions/causal convolutions popularized by the wavenet model (Van Den Oord et al., 2016). Temporal convolutions have been recently used in time-series forecasting (Bai et al., 2018; Borovykh et al., 2017). These deep network based models can be trained on large time-series data-sets as a whole, in mini-batches. However, they still have two important shortcomings.

Firstly, most of the above deep models are difficult to train on data-sets that have wide *variation in scales* of the individual time-series. For instance in the item demand forecasting use-case, the demand for some popular items may be orders

[1]University of Texas as Austin [2]Amazon. Correspondence to: Rajat Sen <rajat.sen@utexas.edu>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

of magnitude more than those of niche items. In such data-sets, each time-series needs to be appropriately normalized in order for training to succeed, and then the predictions need to be scaled back to the original scale. The mode and parameters of normalization are difficult to choose and can lead to different accuracies.

Secondly, even though these deep models are trained on the entire data-set, during prediction the models only focus on *local* past data i.e only the past data of a time-series is used for predicting the future of that time-series. However, in most datasets, *global* properties may be useful during prediction time. For instance, in stock market predictions, it might be beneficial to look at the past values of Alphabet, Amazon's stock prices as well, while predicting the stock price of Apple. TRMF (Yu et al., 2016) is a temporally regularized matrix factorization model that can express all the time-series as linear combinations of *basis time-series*. These basis time-series can capture *global* patterns during prediction. However, TRMF can only model linear temporal dependencies. Moreover, there can be approximation errors due to the factorization, which can be interpreted as a lack of local focus i.e the model only concentrates on the global patterns during prediction.

In light of the above discussion, we aim to propose a deep learning model that can *think globally and act locally* i.e., leverage both *local and global* patterns during training and prediction, and also can be trained reliably even when there are *wide variations in scale*. The **main contributions** of this paper are as follows: **(1)** In Section 3, we propose a variation of of the temporal convolution architecture called the *leveled network* that can be trained effectively on data-sets with wide scale variations, *without* a priori normalization, **(2)** In Section 4.1, inspired by TRMF (Yu et al., 2016), we introduce a Matrix Factorization model regularized by a leveled network (DLN-MF). This model can handle *global* dependencies during prediction and can be trained using mini-batch SGD, **(3)** In Section 4.2, we combine the *global* model with a *local* leveled network trained on the original data-set, through a data-dependent attention model. This leads to a stronger hybrid model, namely DeepGLO (Deep **G**lobal **LO**cal Forecaster) that focuses on both local properties of individual time-series and global dependencies in the data-set while training and prediction, **(4)** In Section 5, we show that DeepGLO outperforms other benchmarks on three real time-series data-sets, including a wiki dataset which contains more than $110K$ dimension of time series. An extened version of this paper can be found here.

2. Problem Setting

Before, we formally define the problem, we will set up some notation. We will use bold capital letters to denote matrices such as $\mathbf{M} \in \mathbb{R}^{m \times n}$. M_{ij} and $\mathbf{M}[i, j]$ will be used interchangeably to denote the (i, j)-th entry of the matrix $\mathbf{M}$. Let $[n] \triangleq \{1, 2, ..., n\}$ for a positive integer n. For $\mathcal{I} \subseteq [m]$ and $\mathcal{J} \subseteq [n]$, the notation $\mathbf{M}[\mathcal{I}, \mathcal{J}]$ will denote the sub-matrix of $\mathbf{M}$ with rows in $\mathcal{I}$ and columns in $\mathcal{J}$. $\mathbf{M}[:, \mathcal{J}]$ means that all the rows are selected and similarly $\mathbf{M}[\mathcal{I}, :]$ means all the columns are chosen. $\mathcal{J} + s$ denotes all the set of elements in $\mathcal{J}$ increased by s. The notation $i : j$ for positive integers $j > i$, is used to denote the set $\{i, i + 1, ..., j\}$. $\|\mathbf{M}\|_F$, $\|\mathbf{M}\|_2$ denote the Frobenius and Spectral norms respectively. For a vector $\mathbf{v} \in \mathbb{R}^{1 \times n}$, $\mu(\mathbf{v}) \triangleq (\sum_i v_i)/n$ denotes the empirical mean of the coordinates and $\sigma(\mathbf{v}) \triangleq \sqrt{(\sum_i (v_i - \mu(\mathbf{v})^2))/n}$ denotes the empirical standard deviation.

Forecasting Task: A time-series data-set can be represented by a matrix $\mathbf{Y} = [\mathbf{Y}^{(\text{tr})}\mathbf{Y}^{(\text{te})}]$, where $\mathbf{Y}^{(\text{tr})} \in \mathbb{R}^{n \times t}$, $\mathbf{Y}^{(\text{te})} \in \mathbb{R}^{n \times \tau}$, n is the number of time-series, t is the number time-points observed during training phase, τ is the window size for forecasting. $\hat{\mathbf{Y}}^{(\text{te})} \in \mathbb{R}^{n \times \tau}$ is used to denote the predicted values in the test range. $\mathbf{y}^{(i)}$ is used to denote the i-th time series, i.e., the i-th row of $\mathbf{Y}$.

The quality of the predictions are generally measured using a metric calculated between the predicted and actual values in the test range. One of the popular metric is the normalized absolute deviation metric (Yu et al., 2016) defined,

$$\mathcal{L}(Y^{(\text{obs})}, Y^{(\text{pred})}) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{\tau} |Y_{ij}^{(\text{obs})} - Y_{ij}^{(\text{pred})}|}{\sum_{i=1}^{n} \sum_{j=1}^{\tau} |Y_{ij}^{(\text{obs})}|}, \quad (1)$$

where $Y^{(\text{obs})}$ and $Y^{(\text{pred})}$ are the observed and predicted values, respectively. This metric is sometime referred to as WAPE in the forecasting literature. Note that (1) is also used as the loss function in our proposed models.

3. DLN: A Deep Leveled Network

In this section, we propose Deep Leveled Network(DLN), an architectural variation of the temporal convolution network which is designed to handle high-dimensional time-series data with wide variation in scale, without normalization. In this work we will use Temporal Convolution (TC) blocks (Bai et al., 2018) as the basic building blocks. A temporal convolution network can be treated as an object that takes in the previous values of a time-series $\mathbf{y}_{\mathcal{J}}$, where $\mathcal{J} = \{j - l, j - l + 1, \cdots, j - 1\}$ and outputs the one-step look ahead predicted value $\hat{\mathbf{y}}_{\mathcal{J}+1}$. We will denote a temporal convolution neural network by $\mathcal{T}(\cdot|\Theta)$, where Θ is the parameter weights in the temporal convolution network. Thus, we have $\hat{\mathbf{y}}_{\mathcal{J}+1} = \mathcal{T}(\mathbf{y}_{\mathcal{J}}|\Theta)$. The same operators can be defined on matrices. Given $\mathbf{Y} \in \mathbb{R}^{n \times t}$ and a set of row indices $\mathcal{I} = \{i_1, ..., i_{b_n}\} \subset [n]$, we can write $\hat{\mathbf{Y}}[\mathcal{I}, J + 1] = \mathcal{T}(\mathbf{Y}[\mathcal{I}, J]|\Theta)$.

DLN consists of two temporal convolution blocks (having the same dynamic range l) that are trained concurrently.

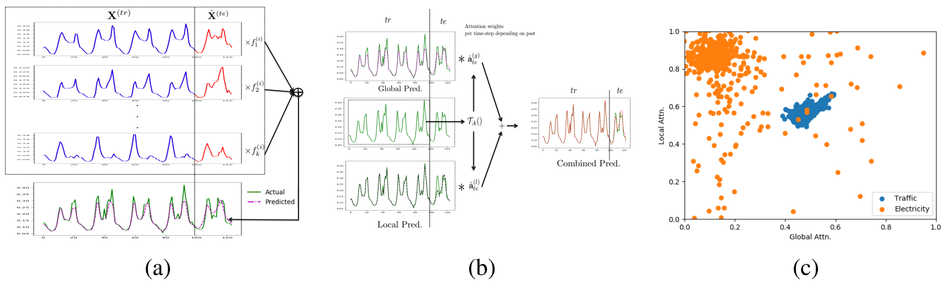
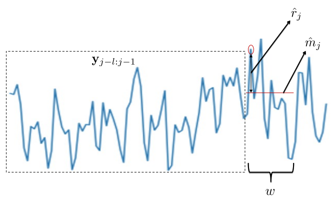


Figure 1. In Fig. 1a, we show some of the basis time-series extracted from the traffic dataset, which can be combined linearly to yield individual original time-series. It can be seen that the basis series are highly temporal and can be predicted in the test range using the network $\mathcal{T}_X(\cdot|\Theta_X)$. Fig. 1b shows global and local predictions on the same time-series, which are combined through $\mathcal{T}_A(\cdot|\Theta_A)$, to provide the final predictions. Note that the attention weights depend on the time-index and adaptively combine the local and global components. In Fig. 1c, we display a scatter plot of the local vs global attention weights in the test range for the time-points of all time-series in traffic and electricity (most weights fall [0,1] therefore the range is restricted to [0,1]). We see in that in electricity, more attention is on the local component while in traffic the attention is almost equal on both global and local components.

Table 1. Comparison of algorithms on normalized and unnormalized versions of data-sets on rolling prediction tasks The error metrics reported are WAPE/MAPE/SMAPE (standard time-series data metrics). TRMF is retrained before every prediction window, during the rolling predictions. All other models are trained once on the initial training set and used for further prediction for all the rolling windows. Prophet could not be scaled to the wiki dataset, even though it was parallelized on a 32 core machine.

	Algorithm	electricity $n = 370$		traffic $n = 963$		wiki $n = 115,084$	
		Normalized	Unnormalized	Normalized	Unnormalized	Normalized	Unnormalized
Proposed	DeepGLO	**0.084**/0.291/**0.119**	0.109/0.448/**0.149**	**0.159**/**0.218**/0.202	0.221/**0.321**/**0.254**	**0.233**/**0.380**/0.402	**0.228**/**0.356**/**0.311**
	Local DLN	0.086/**0.258**/0.129	0.118/**0.336**/0.172	0.169/0.246/0.218	0.237/0.422/0.275	0.235/0.469/**0.346**	0.288/0.397/0.341
Local-Only	LSTM	0.109/0.264/0.154	0.896/0.672/0.768	0.276/0.389/0.361	0.270/0.357/0.263	0.427/2.170/0.590	0.789/0.686/0.493
	DeepAR	0.099/0.375/0.146	0.889/0.818/0.876	0.268/0.369/0.272	0.250/0.331/0.258	0.442/2.980/0.522	0.958/8.120/1.140
	Temporal Conv.	0.147/0.476/0.156	0.423/0.769/0.523	0.204/0.284/0.236	0.239/0.425/0.281	0.336/1.322/0.497	0.511/0.884/0.509
	Prophet	0.197/0.393/0.221	0.221/0.586/0.524	0.313/0.600/0.420	0.303/0.559/0.403	-	-
Global-Only	TRMF (retrained)	0.104/0.280/0.151	**0.105**/0.431/0.183	**0.159**/0.226/**0.181**	**0.210**/ 0.322/ 0.275	0.309/0.847/0.451	0.320/0.938/0.503
	SVD+DLN	0.219/0.437/0.238	0.368/0.779/0.346	0.468/0.841/0.580	0.329/0.687/0.340	0.697/3.51/0.886	0.639/2.000/0.893



$$[\hat{y}_{j-l+1},\cdots,\hat{y}_j] = \mathcal{T}_{\mathsf{DLN}}(\mathbf{y}_{j-l:j-1}|\Theta_m,\Theta_r) \quad (2)$$
$$:= \mathcal{T}_m(\mathbf{y}_{j-l:j-1}|\Theta_m)+\mathcal{T}_r(\mathbf{y}_{j-l:j-1}|\Theta_r)$$
$$[\hat{m}_{j-l+1},\cdots,\hat{m}_j] = \mathcal{T}_m(\mathbf{y}_{j-l:j-1}|\Theta_m) \quad (3)$$
$$[\hat{r}_{j-l+1},\cdots,\hat{r}_j] = \mathcal{T}_r(\mathbf{y}_{j-l:j-1}|\Theta_r) \quad (4)$$

Figure 2. Illustration of the leveled network. The leveling component $\hat{m}^j$ and the residual component $\hat{r}_j$ are predicted as functions of the past $\mathbf{y}_{j-l:j-1}$. The final prediction is $\hat{m}_j + \hat{r}_j$.

Let us denote the two networks and the associated weights by $\mathcal{T}_m(\cdot|\Theta_m)$ and $\mathcal{T}_r(\cdot|\Theta_r)$ respectively. The key idea is to have $\mathcal{T}_m(\cdot|\Theta_m)$ (the leveling component) to predict the rolling mean of the next w future time-points given the past. On the other-hand $\mathcal{T}_r(\cdot|\Theta_r)$ (the residual component) will be used to predict the variations with respect to this mean value. Given an appropriate window size w the rolling mean stays stable for each time-series and can be predicted by a simple temporal convolution model and given these predictions the additive variations are relatively scale free i.e. the network $\mathcal{T}_r(\cdot|\Theta_r)$ can be trained reliably without normalization. An illustration of the leveled network methodology is shown in Figure 2, along with governing equations.

Both the networks can be trained concurrently given the training set $\mathbf{Y}^{(\text{tr})}$, using mini-batch stochastic gradient updates. The loss function $\mathcal{L}(\cdot,\cdot)$ used is the same as the metric defined in Eq. (1). In the interest of space, we will skip the detailed description of the training algorithm.

The trained model can be used for multi-step look-ahead prediction in a standard manner. In Section 5, we shall see that DLN can be trained on unnormalized data-sets with wide variations in scale.

4. DeepGLO: A Deep Global Local Forecaster

In this section, we propose DeepGLO, our complete hybrid model which consists of separate global (Section 4.1) and local components (Section 3), married through a data-driven attention mechanism (Section 4.2).

4.1. Temporal Matrix Factorization regularized by Deep Leveled Network (DLN-MF)

In this section we propose a low-rank matrix factorization model for time-series forecasting that uses a leveled network for regularization. This approach is inspired by temporal regularized matrix factorization(TRMF) (Yu et al., 2016).

The idea is to factorize the training time-series matrix $\mathbf{Y}^{(\text{tr})}$ into low-rank factors $\mathbf{F} \in \mathbb{R}^{n\times k}$ and $\mathbf{X}^{(\text{tr})} \in \mathbb{R}^{k\times t}$, where

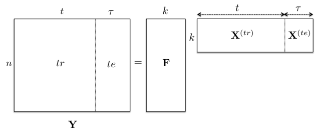


Figure 3. Illustration of the matrix factorization approach in time-series forecasting. The $\mathbf{Y}^{(\text{tr})}$ training matrix can be factorized into low-rank factors $\mathbf{F}$ ($\in \mathbb{R}^{n\times k}$) and $\mathbf{X}^{(\text{tr})}$ ($\in \mathbb{R}^{k\times t}$). If the $\mathbf{X}^{(\text{tr})}$ preserves temporal structures then the future values $\mathbf{X}^{(\text{te})}$ can be predicted by a time-series forecasting model and thus the test period predictions can be made as $\mathbf{F}\mathbf{X}^{(\text{tr})}$.

$k \ll n$. This is illustrated in Figure 3. Further, we would like to encourage a temporal structure in $\mathbf{X}^{(\text{tr})}$ matrix, such that the future values $\mathbf{X}^{(\text{te})}$ in the test range can also be forecasted, where $\mathbf{X} = [\mathbf{X}^{(\text{tr})}\mathbf{X}^{(\text{te})}]$.

Temporal Regularization by a DLN: If we are provided with a leveled network that captures the temporal patterns in the training data-set $\mathbf{Y}^{(\text{tr})}$, then we can encourage temporal structures in $\mathbf{X}^{(\text{tr})}$ using this model. Let us assume that the said leveled network is $\mathcal{T}_X(\cdot)$. The temporal patterns can be encouraged by including the following temporal regularization into the loss function:

$$\mathcal{R}(\mathbf{X}^{(\text{tr})} \mid \mathcal{T}_X) := \frac{1}{|\mathcal{J}|}\mathcal{L}\left(\mathbf{X}[:,\mathcal{J}], \mathcal{T}_X\left(\mathbf{X}[:,\mathcal{J}-1]\right)\right),$$

where $\mathcal{J} = \{l, \cdots, t\}$ and $\mathcal{L}(\cdot,\cdot)$ is the metric defined in Eq. (1). This implies that the values of the $\mathbf{X}^{(tr)}$ on time-index j are close to the predictions from the temporal network applied on the past values between time-steps $\{j-l, ..., j-1\}$. Here, l is ideally equal to the dynamic range of the network. Thus the overall loss function for the factors and the temporal network is as follows:

$$\begin{aligned}&\mathcal{L}_G(\mathbf{Y}^{(\text{tr})}, \mathbf{F}, \mathbf{X}^{(\text{tr})}, \mathcal{T}_X)\\ :=&\mathcal{L}\left(\mathbf{Y}^{(\text{tr})}, \mathbf{F}\mathbf{X}^{(\text{tr})}\right) + \lambda_{\mathcal{T}}\mathcal{R}(\mathbf{X}^{(\text{tr})} \mid \mathcal{T}_X(\cdot)),\end{aligned} \qquad (2)$$

where $\lambda_{\mathcal{T}}$ is the regularization parameter for the temporal regularization component. The low rank factors and the DLN $\mathcal{T}_X$ can be trained alternatively. It should be noted that the trained DLN-MF model can be used to make rolling prediction *without retraining*.

4.2. Combining Global and Local Models

In this section, we will extend our model to leverage both global information (through basis time-series in DLN-MF) and local information (through a separate DLN $\mathcal{T}_Y(\cdot|\Theta_Y)$ that is trained on the original $\mathbf{Y}^{(\text{tr})}$ and predicts the future values of each time-series based on its past observed values). The idea is to have a simple temporal convolution network $\mathcal{T}_A$ that outputs attention weights based on the past values of a time series $\mathbf{y}^{(i)}_{s-l:s}$. These attention weights are used to combine local and global predictions. We skip further details in the interest of space. We illustrate some representative results on time-series from the traffic and electricity dataset in Fig. 1.

5. Empirical Results

We validate our model on three real-world data-sets on rolling prediction tasks i.e (i) electricity (Trindade, 2011);(ii) traffic (Cuturi, 2011) and (iii) wiki (Kaggle, 2017). Data statistics are provided in Table 2.

Table 2. Data statistics and Evaluation settings. In the rolling Pred., τ_w denotes the number of time-points in each window and n_w denotes the number of rolling windows. $\mathrm{std}(\{\mu\})$ denotes the standard deviation among the means of all the time series in the data-set i.e $\mathrm{std}(\{\mu\}) = \mathrm{std}(\{\mu(\mathbf{y}^{(i)})\}_{i=1}^{n})$. Similarly, $\mathrm{std}(\{\mathrm{std}\})$ denotes the standard deviation among the std. of all the time series in the data-set i.e $\mathrm{std}(\{\mathrm{std}\}) = \mathrm{std}(\{\mathrm{std}(\mathbf{y}^{(i)})\}_{i=1}^{n})$. It can be seen that the electricity and wiki datasets have wide variations in scale.

Data	n	t	τ_w	n_w	$\mathrm{std}(\mu)$	$\mathrm{std}(std)$
electricity	370	25,968	24	7	$1.19e^4$	$7.99e^3$
traffic	963	10,392	24	7	$1.08e^{-2}$	$1.25e^{-2}$
wiki	115,084	747	14	4	$4.85e^4$	$1.26e^4$

For each data-set, all models are compared on two different settings. In the first setting the models are trained on *normalized* version of the data-set where each time series in the training set is re-scaled as $\tilde{\mathbf{y}}^{(i)}_{1:t-\tau} = (\mathbf{y}^{(i)}_{1:t-\tau} - \mu(\mathbf{y}^{(i)}_{1:t-\tau}))/(\sigma(\mathbf{y}^{(i)}_{1:t-\tau}))$ and then the predictions are scaled back to the original scaling. In the second setting, the dataset is *unnormalized* i.e left as it is while training and prediction. Note that all models are compared in the test range on the original scale of the data. The purpose of these two settings is to measure the impact of scaling on the accuracy of the different models.

The models under contention are: **(1)** DeepGLO: Model proposed in Section 4.2. **(2)** Local DLN: The leveled network in Section 3. **(3)** LSTM: A simple LSTM block (Gers et al., 1999). **(4)** DeepAR: The model proposed in (Flunkert et al., 2017). **(5)** TC: A simple Temporal Convolution model. **(6)** Prophet: The versatile forecasting model from Facebook based on classical techniques (Facebook, 2017). **(7)** TRMF: the model proposed in (Yu et al., 2016). **(8)** SVD+DLN: Combination of SVD and leveled network.

In Table 1, we report WAPE/MAPE/SMAPE on all three datasets under both normalized and unnormalized training. We can see that DeepGLO features among the top two models in almost all categories, under all three metrics. It should be noted that DeepGLO performs substantially better than other models on the much larger wiki dataset. In fact it does at least 30% better than the next best model (not proposed in this paper) in terms of MAPE. Also, note that DeepGLO performs better on wiki, using unnormalized data. We would also like to point out that the leveled network can train well on unnormalized data, while other deep models (LSTM, DeepAR, TC) do not perform well in this setting.

References

Bai, S., Kolter, J. Z., and Koltun, V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

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

Box, G. E. and Jenkins, G. M. Some recent advances in forecasting and control. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 17(2):91–109, 1968.

Chatfield, C. *Time-series forecasting*. Chapman and Hall/CRC, 2000.

Cuturi, M. Fast global alignment kernels. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 929–936, 2011.

Facebook. Fbprophet. https://research.fb.com/prophet-forecasting-at-scale/, 2017. [Online; accessed 07-Jan-2019].

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

Funahashi, K.-i. and Nakamura, Y. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural networks*, 6(6):801–806, 1993.

Gers, F. A., Schmidhuber, J., and Cummins, F. Learning to forget: Continual prediction with lstm. 1999.

Hamilton, J. D. *Time series analysis*, volume 2. Princeton university press Princeton, NJ, 1994.

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.

Kaggle. Wikipedia web traffic. https://www.kaggle.com/c/web-traffic-time-series-forecasting/data, 2017. [Online; accessed 07-Jan-2019].

Kim, K.-j. Financial time series forecasting using support vector machines. *Neurocomputing*, 55(1-2):307–319, 2003.

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.

Larson, P. D. Designing and managing the supply chain: Concepts, strategies, and case studies, david simchi-levi philip kaminsky edith simchi-levi. *Journal of Business Logistics*, 22(1):259–261, 2001.

Lütkepohl, H. *New introduction to multiple time series analysis*. Springer Science & Business Media, 2005.

McKenzie, E. General exponential smoothing and the equivalent arma process. *Journal of Forecasting*, 3(3):333–344, 1984.

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.

Seeger, M. W., Salinas, D., and Flunkert, V. Bayesian intermittent demand forecasting for large inventories. In *Advances in Neural Information Processing Systems*, pp. 4646–4654, 2016.

Sundermeyer, M., Schlüter, R., and Ney, H. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012.

Trindade, A. Electricityloaddiagrams20112014 data set. https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014, 2011. [Online; accessed 07-Jan-2019].

Van Den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *CoRR abs/1609.03499*, 2016.

Wen, R., Torkkola, K., Narayanaswamy, B., and Madeka, D. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*, 2017.

Wilson, K. W., Raj, B., and Smaragdis, P. Regularized non-negative matrix factorization with temporal dependencies for speech denoising. In *Ninth Annual Conference of the International Speech Communication Association*, 2008.

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.