Product demand forecasting based on reservoir computing

Jorge Calvimontes
Centro de Investigaciones en Nuevas Tecnologías Informáticas, Universidad Privada Boliviana

Jens Bürger
Centro de Investigaciones en Nuevas Tecnologías Informáticas, Universidad Privada Boliviana
Institute for Computational Intelligence, Universidad Privada Boliviana

Abstract

Common forecasting methods fail to accurately model the nonlinear and time-varying fluctuations of product demand. Reservoir computing (RC) utilizes a dynamical system to project time-series data to a higher-dimensional state representation extracting mathematical relations within complex demand functions. We demonstrate forecasting accuracy of RC on a multivariate product demand dataset.

Keywords: Demand forecasting, reservoir computing, time-series analysis

1. Introduction

In production and operational management demand forecasting is an important method as it helps to develop better approximations of future operations under the presence of uncertainty. Forecasting extracts mathematical relations from within past data that can be used to inform decision making. In supply chain management, efficient coordination of resource acquisition, production and warehousing strongly depends on accurately predicting future product demand in particular and market dynamics in general. Accurate demand forecasting therefore reduces investment risks in uncertain environments. The challenges of demand forecasting lie in the complexity of demand dynamics. In general demand data can be decomposed into deterministic patterns and random fluctuations. While it is impossible to predict random fluctuations in demand, deterministic patterns can, in theory, be learned or approximated by corresponding forecasting models.

Among the most commonly applied forecasting methods are *exponential smoothing* (ES) (Holt 2004) and the *autoregressive integrated moving average* (ARIMA) (Hyndman and Athanasopoulos 2018). These models fit parameters to polynomial functions in order to approximate the time-dependent demand data. Applications of ES and ARIMA have been shown for forecasting urban freeway traffic flow (Williams et al. 1998) and electricity demand (Taylor 2003; Ediger and Akar 2007) . While such models work well for demand patterns composed of few frequency components, due to their fixed mathematical structure their applicability and accuracy is limited when approximating complex demand patters described by a wide frequency spectrum and nonlinear patterns (Kohzadi et al. 1996).

As non-explicitly mathematical models, *artificial neural networks* (ANN) are a powerful tool to approximate complex nonlinear relations from a set of input variables as to map them to a target output. Through nonlinear kernel functions ANN create a higher-dimensional representation of an input signal so that input-output relations become more evident and easier to approximate. The most common ANN is the *multi-layer perceptron* (MLP) which passes data

in a uni-directional way from the input layer to an output layer. Applications of MLP to time series forecasting have been shown to achieve lower mean squared error than ARIMA (Kohzadi et al. 1996). However, due to the uni-directional flow of data through the MLP it presents a memory-less model which has limited applicability to time-series data.

More suitable for time-series data are *recurrent neural networks* (RNN), which implement a memory of past data through recurrent connections within the neural network. The higher-dimensional representation within the RNN is therefore determined by nonlinear combinations of data points obtained from a temporal sequence. The main difficulty with RNN is the training of their parameters known as the vanishing gradient problem, which has limited the application of RNN to rather simple problems (Bengio et al. 1994). In order to facilitate training, regularization techniques have been proposed (Hammer and Steil 2002), which impose constrains for training or limitations to the architecture itself. Such techniques include stability constrains, automata rules and locally recurrent architectures (with LSTM being the most popular (Maass et al. 2002)). Under automata rules it has been proposed that a random initialization of the network has a bias towards a finite memory and good generalization properties (Hammer and Steil 2002), which implies that training of RNN can be simplified.

In this article we present the application of *reservoir computing* (RC) (Jaeger 2001; Maass et al. 2002) to product demand forecasting. RC utilizes a randomly initialized RNN that implements the mentioned finite memory and generalization. Under these conditions it is sufficient to reduce training complexity to only a single linear output layer and achieve accurate forecasting results. The output layer is therefore able to derive a simple linear relationship between the input data and its projection into a higher-dimensional feature space. The general suitability of RC to time-series data has already been demonstrated in numerous publications (Jaeger and Haas 2004; Larger et al. 2012; Sheng et al. 2012). In the following sections we demonstrate the advantages of RC over common mathematical forecasting models applied to a complex demand data set.

2. Methodology

2.1. Exponential Smoothing

Exponential smoothing (ES) is a collection of methods that forecast, based on weighted averages of past observations with exponentially decaying weights as the observations get older (Holt 2004). This collection offers different methods to forecast, depending if the data contains trend or seasonal patterns. Simple smoothing (no trend or seasonality) can be expressed in a recursive way:

$$\hat{y}_t = \alpha y_t + (1 - \alpha)\ell_{t-1}, \tag{1}$$

where α is the smoothing factor ($0 \leq \alpha \leq 1$), ℓ_0 is a parameter for the first fitted value, y is an observed and $\hat{y}$ a forecasted value of the series at time t.

When data contains trend or seasonality, additional smoothing equations are considered, equation 1 will also include new factors. If data presents trend, an equation b_t is used, which considers trend dynamics. For seasonality, depending if patterns change in a constant or proportional way (Hyndman and Athanasopoulos 2018), an additive or multiplicative equation s_t is used. s_t is configured by a periodicity parameter m, which determines seasonal patterns. The mentioned equations use smoothing factors β and γ, for trend and seasonality, respectively. These, in conjunction with α, determine the smoothing level for each function. This determines how well the model captures patterns present in data.

Complex data will likely contain trend and seasonal patterns. In this case a combination of the trend and seasonal equations are used to compute for a number of forecasted steps, with h been the number of steps into the future. Depending on seasonal characteristics, an additive or multiplicative equation (s_t) can be used.

$$\hat{y}_{t+h|t} = \ell_t + hb_t + s_{t+h-m(k+1)} \tag{2}$$

$$\hat{y}_{t+h|t} = (\ell_t + hb_t)s_{t+h-m(k+1)} \tag{3}$$

Equation 2 illustrates a model that considers trend and an additive seasonality, whereas equation 3 shows a multiplicative case. An integer k is used in the form: $k = (h-1)/m$ to ensure consistency of the seasonal indices.

2.2. ARIMA

The *autoregressive integrated moving average* (ARIMA), is the combination of autoregressive (AR) and moving average (MA) methods. AR models are constructed according to the difference between the series and a shifted (lagged) version, of itself. MA methods are constructed based on the error between consecutive time intervals. Both methods can be expressed as a weighted sum of past events. *Integrated* in this sense means the sum of the coefficients of the AR and MA models, in the form:

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t, \tag{4}$$

where y'_t is a differentiated version of the data for time t. Constants c and ϕ correspond to the AR model, while ε_t and θ correspond to the MA model.

The notation used to express the model is: ARIMA(p,d,q), where the values of p,d, and q, will determine the behaviour of the ARIMA model. The parameters p and q are the number of steps to consider for the AR and MA models, respectively. In order for ARIMA models to forecast, data must be stationary (Hyndman and Athanasopoulos 2018). To achieve stationarity data can be differentiated. Parameter d determines the order of differentiation. Although the conventional ARIMA model only works with non seasonal data, it can be extended to work with seasonal data by adding 4 more parameters: P,D,Q and m, where the first 3 correspond to the same parameters as the conventional method. The difference only being that their regressions are based on a seasonal scale according to the periodicity m. In that context a seasonal ARIMA model will forecast based on seasonal and non seasonal patterns of the data, with the notation been: ARIMA(p,d,q)(P,D,Q,M)

2.3. Reservoir Computing

Reservoir computing (RC) is a paradigm that harnesses untrained dynamical systems (most commonly recurrent neural networks) for computation (Jaeger 2001; Maass et al. 2002). Basic components of RC are the input layer, the reservoir and the readout layer. The input layer applies a time-dependent input signal $\mathbf{u}(t)$ via a fixed weight matrix $\mathbf{W}^{in}$ to the reservoir. Reservoir nodes (artificial neurons) perform nonlinear computation in order to create a higher dimensional representation of the applied input signal. Reservoir nodes are connected internally through a random (untrained) weight matrix $\mathbf{W}^{res}$ that implements recurrent loops allowing input data to persist within the reservoir and therefore creating a short-term memory. Each node of the time-dependent reservoir state $\mathbf{x}(t)$ evolves according to:

$$x_i(t+1) = f_i\left(\mathbf{W}_i^{res} \cdot \mathbf{x}(t) + \mathbf{W}_i^{in} \cdot \mathbf{u}(t)\right), \tag{5}$$

where $\mathbf{W}_i^{res}$ and $\mathbf{W}_i^{in}$ are the i^{th} row vectors of the reservoir weight matrix and the input weight matrix, respectively. f_i is the activation function of node i. The output signal $\mathbf{y}(t)$ of the system is derived by the readout layer through a linear combination of the reservoir states.

$$\mathbf{y}(t) = \mathbf{W}^{out} \cdot \mathbf{x}(t) \tag{6}$$

Assuming good generalization properties of random projections within the reservoir, learning takes place by adjusting the output weight matrix $\mathbf{W}^{out}$ only. This reduces the training complexity of recurrent neural networks to the output matrix, which is commonly done by using simple regression techniques. In general, the recurrent structure of the reservoir represents a dynamical model that creates features from the time-series data applied to it. RC has already found widespread application for various types of time-series problems such as speech recognition (Triefenbach et al. 2014), motor control (Salmen and Ploger 2005), and forecasting problems (Coulibaly 2010), to name just a few.

2.4. Dataset

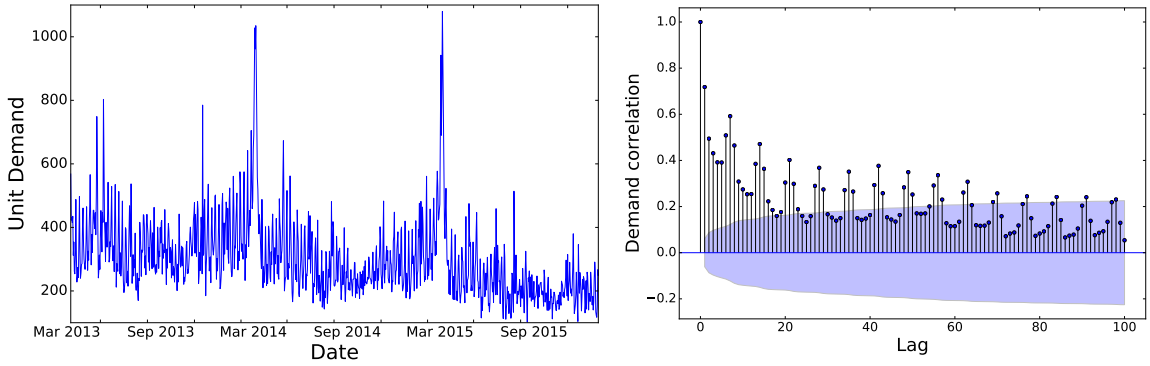
Regardless of the forecasting method, the obtainable accuracy will strongly depend on the dynamics and the complexity of the dataset. In literature it is a common practice to use simple data with evident patterns (either in trend or seasonality), when trying to explain the dynamics involved for each method as it's been shown by (Hyndman and Athanasopoulos 2018). Given the periodic properties of ARIMA (seasonal) and ES, the accuracy with simple data is expected to be high. However, when dealing with complex data the ability of static mathematical models to achieve accurate forecasting is questionable. As the aim of this study is to compare traditional and dynamic forecasting methods for real world applications, a dataset of a real company containing rich (consecutive) data with non evident patterns will be used.

The dataset corresponds to sales of one of the largest Russian software firms *1C Company*, which was used for the kaggle course *How to win a data science competition*. This dataset presents daily historical demand data from January 2013 to October 2015 with relevant data fields of shop ID, item ID, item price, date and number of products sold (per day). As the kaggle challenge is aimed to forecast future sales for each shop, the data will be prepossessed so that the aggregate demand for each shop is obtained.

In Figure 1a we can see the aggregate demand for the shop with the most data samples (shop 31). The shown data highlights that the demand function is composed of various dynamics with shorter and longer range periodic behaviour. Through an autocorrelation plot (Figure 1b) we can derive that the data contains a trend, indicated by a decrease on the correlation values, and seasonality, indicated by the presence of wave like shapes. Peaks every 7 lags suggest that there are strong weekly patterns. Additionally, through STL decomposition we also determined that data to contains random fluctuations that will compromise the ability to perfectly predict future demand.

3. Results

The overall forecasting accuracy of each method will be evaluated based on results obtained for the aggregate demand of each shop. This requires each forecasting method to be parameterized as to generalize over the set of independent product demand functions of each shop. We parameterized ES with two values. The periodicity m commonly refers to the number of seasons within a given time-series. While the used dataset clearly showed various temporal fluctuations in mean and standard deviation, no clear seasonality could be determined.



(a) Aggregate Demand

(b) Autocorrelation plot

Figure 1: (a) Aggregate Demand: shop with most samples (shop 31), (b) autocorrelation plot: with confidence interval of 95%(shaded area), as lags increase, the overall correlation values decrease, on a waved shaped form, related to trend and seasonality

Table 1: Parameters used for each forecasting method

ES		ARIMA							RC				
Type	m	p	d	q	P	D	Q	m	N	β	λ	N_{out}	k
Additive	7	0-2	0	1	0-2	1	1	7	5000	0.05	0.95	50	15

We therefore empirically determined m to be 7. This corresponds to the auto-correlation plot (Figure 1b), which shows weekly patterns to be the most dominant. Likewise, as the data exhibits various seasonalities, we empirically determined the seasonal adjustment method to be additive (Hyndman and Athanasopoulos 2018). As most of the shops share similar temporal characteristics, the mentioned parameters were used for all shops. The parameters P, D and Q for ARIMA were obtained through grid search and corresponding minimization of the Akaike's Information Criterion (AIC) (Hyndman and Athanasopoulos 2018). It was determined that for most sequences, the parameters were in the range $[0, 2]$. The periodicity m used to determine the seasonal component was set in accordance to what was determined for ES. The parameters influencing the predictive capabilities of RC are the number of artificial neurons (N), input scaling (β), spectral radius (λ), average in-degree (k) and reservoir output nodes (N_{out}). N defines the reservoir's theoretical memory and computational capacity, β determines the level of nonlinearity that is exploited within the network, λ sets the length of the fading memory, k determines the number of incoming connections to each neuron in the reservoir and N_{out} defines the number of randomly selected nodes passed from the reservoir to the readout layer. Table 1 summarizes the parameters used for all methods. All experiments where conducted in a Python environment using the modules *stats models* (Seabold and Perktold 2010) for the conventional forecasting methods and *Oger* (Verstraeten et al. 2012) for reservoir computing.

Error evaluation was done for all methods using Normalized Root Mean Square Error (NRMSE) between the forecasted and the actual data. Normalization is done by dividing the RMSE by the target signal's standard deviation (STD). Considering that STD is sensitive for sequences of small sample size, we define a minimum sample length n of 100. This value also corresponds closely to quarterly planning cycles often found in industry. We then performed forecasting for all shops and calculated the individual NRMSE for each one. As RC initializes the neural network with random connectivity, we can expect some fluctuations in the RC results. In order to quantify the stochastic impact of connectivity on RC results, we performed

Table 2: Forecasting results

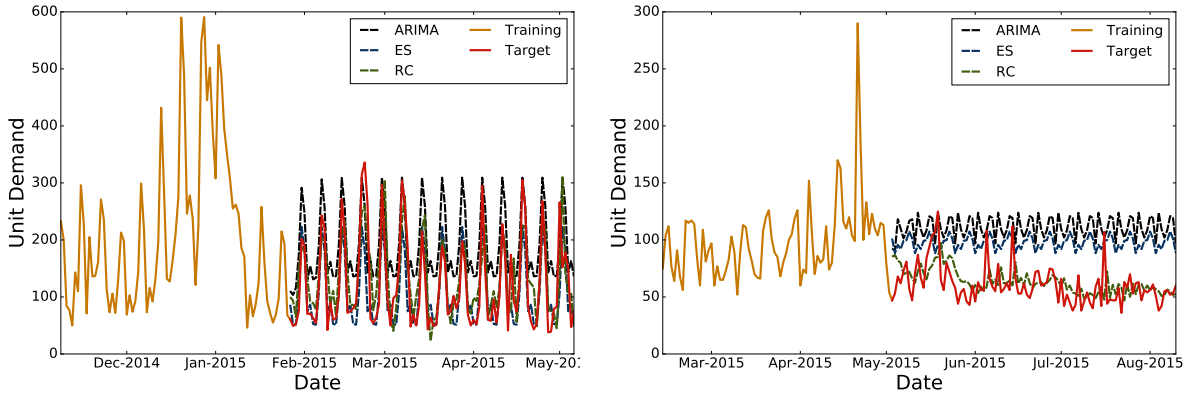
Horizon	1				3			
	Min	Max	Mean	STD	Min	Max	Mean	STD
ES	**0.58**	2.57	1.15	0.33	**0.61**	2.76	1.17	0.35
ARIMA	0.83	3.16	1.23	0.39	0.84	3.17	1.24	0.40
RC (mean)	0.61	**1.48**	**1.03**	**0.14**	0.68	**1.48**	**1.07**	**0.14**
RC (STD)	(0.01)	(0.14)	(0.01)	(0.01)	(0.02)	(0.12)	(0.1)	(0.01)
Horizon	7				14			
ES	**0.62**	2.79	1.18	0.36	**0.65**	3.03	1.21	0.39
ARIMA	0.83	3.19	1.27	0.41	0.86	3.28	1.33	0.44
RC (mean)	0.69	**1.59**	**1.11**	**0.16**	0.75	**1.89**	**1.18**	**0.22**
RC (STD)	(0.02)	(0.07)	(0.01)	(0.01)	(0.02)	(0.06)	(0.02)	(0.01)

each experiment 10 times with different reservoir initialization. The reported results for RC are therefore the mean and STD over the repeated experiments. Forecasting accuracy was evaluated for four different horizons. We define the forecasting a horizon as the number of days we predict into the future (with respect to the auto-correlation plot the horizon corresponds to the lag value). Table 2 shows the results obtained considering four different horizons.

From the results in Table 2 it can be observed that conventional methods don't exhibit a significant change as the horizon increases. This is due to the fact that these models fit parameters directly onto the training signal and not to a relation between a training and a target that is a shifted training signal. Contrary, RC is more sensitive to an increase in the horizon as can be observed from increasing errors. This sensitivity is the result of the readout layer of RC trying to fit the dynamic, input-driven reservoir state to a target output function with a shifted horizon. As can be seen in Figure 1b, with increasing horizon (lag), the autocorrelation decreases. This implies that with longer horizons more uncertainty is added to the relation between the reservoir state and the target output. Nonetheless, for up to 14 days horizon, autocorrelation was still significant and allowed RC to outperform the conventional methods. While the advantage of RC over the conventional methods (measured through mean and max results) slightly reduces with increasing horizon, it is noteworthy that it maintained significant smaller standard deviations. Especially if forecasting is bound to i.e., large investments in resources and/or production capacities, low STD reduce operational risks. This suggests that input-driven dynamical systems, such as RC, are more responsive to sudden fluctuations or regime shifts within the demand function.

In Figure 2 we present two scenarios that help us to better qualify the functional differences between the three methods. The first scenario represents the case were ES achieved the best results across all methods (Figure 2a). The data for this example presented a continuous pattern until mid-December 2014 after which we could observe a temporary increase in demand. Mid-January the demand returned to its previous pattern. The ES approach, by fitting its parameters over the full length of the training data, showed little sensitivity to the short-term demand fluctuations and approximated the test signal accurately. Contrary, the ARIMA model, by giving higher priority to more recent data and by evaluating shorter sequences, was affected more strongly by the temporary demand fluctuations. The RC approach, similarly to ES, was not affected by the fluctuations and predicted accurately as well, which is confirmed by the qualitative analysis of the match between target and RC output signals.

The second scenario represents a regime shift with resulting significant prediction errors for ES and ARIMA (Figure 2b). In this example, with the beginning of May 2015



(a) Case minimum error for ES

(b) Case Regime shift

Figure 2: (a) Case minimum error for ES: with horizon 1, error of 0.58 for ES, 0.68 for RC and 0.98 for ARIMA, (b) Case Regime shift: with horizon 1, error of 0.99 for RC, 2.40 for ES and 3.15 for ARIMA

(which coincided with the beginning of the test signal) we could observe a lasting change in demand pattern indicated by a different mean and standard deviation of the signal. While ES and ARIMA are not input-driven models they were completely unaware of the regime shift and continued to predict according to the demand function of the training signal. Contrary, the RC adapted to the regime shift. While it could not perfectly capture the dynamics of this new demand pattern, it could approximate overall trend and scale of the demand to the benefit of forecasting accuracy.

4. Conclusion

In this work we have shown that forecasting accuracy on sets of time-series with conventional and dynamic methods greatly depends on the complexity of the presented data. When time-series data can be sufficiently characterized by a single or a few dominant patterns, conventional methods and RC achieve comparable forecasting results. For demand patterns with more complex behavior, such as temporal fluctuations or sudden regime shifts, conventional methods fail to accurately forecast due to their static nature. To reduce the overall forecasting error of the conventional methods would require parameter tuning for each individual time-series as well as constant re-training as to alleviate the effects of fluctuations or regime shifts. In contrast, RC as a dynamical system poses a forecasting framework that (a) can achieve lower errors (with significant lower standard deviation), (b) utilizes a general set of parameters over a range of time-series and (c) is able to more accurately predict temporal fluctuations and can adjust to changing demand patterns (regime shifts). Therefore, conventional mathematical methods create static fits to the actual demand signal, assuming constant repetition of the same pattern. This is similar to the concept of overfitting of neural networks. RC on the other hand, is not directly learning the input signal, but the relation between a random projection of that signal into a feature space and a target output. It harnesses generalization properties of random projections and memory of the recurrent network structure in order to learn dynamic relations rather than static sequences. We can therefore conclude that for most complex (real-world) demand functions dynamical models are preferable as they can more accurately and reliable capture demand dynamics.

References

Bengio, Y., P. Simard, and P. Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks* 5 (2): 157–166.

Coulibaly, P. 2010. Reservoir Computing approach to Great Lakes water level forecasting. *Journal of Hydrology* 381 (1-2): 76–88.

Ediger, V. Ş., and S. Akar. 2007. ARIMA forecasting of primary energy demand by fuel in Turkey. *Energy Policy* 35 (3): 1701–1708.

Hammer, B., and J. J. Steil. 2002. Tutorial: Perspectives on Learning with RNNs. In *Proc. ESANN*, 357–368.

Holt, C. C. 2004. Forecasting seasonals and trends by exponentially weighted moving averages. *International journal of forecasting* 20 (1): 5–10.

Hyndman, R. J., and G. Athanasopoulos. 2018. *Forecasting: principles and practice*. OTexts.

Jaeger, H. 2001. *The ''echo state'' approach to analysing and training recurrent neural networks*. GMD Report 148. German National Research Center for Information Technology.

Jaeger, H., and H. Haas. 2004. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *science* 304 (5667): 78–80.

Kohzadi, N., M. S. Boyd, B. Kermanshahi, and I. Kaastra. 1996. A comparison of artificial neural network and time series models for forecasting commodity prices. *Neurocomputing* 10 (2): 169–181.

Larger, L., M. C. Soriano, D. Brunner, L. Appeltant, J. M. Gutiérrez, L. Pesquera, C. R. Mirasso, and I. Fischer. 2012. Photonic information processing beyond Turing: an optoelectronic implementation of reservoir computing. *Optics express* 20 (3): 3241–3249.

Maass, W., T. Natschläger, and H. Markram. 2002. Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations. *Neural Computation* 14 (11): 2531–2560.

Salmen, M., and P. G. Ploger. 2005. Echo State Networks used for Motor Control. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 1953–1958.

Seabold, S., and J. Perktold. 2010. Statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*.

Sheng, C., J. Zhao, Y. Liu, and W. Wang. 2012. Prediction for noisy nonlinear time series by echo state network based on dual estimation. *Neurocomputing* 82:186–195.

Taylor, J. W. 2003. Short-term electricity demand forecasting using double seasonal exponential smoothing. *Journal of the Operational Research Society* 54 (8): 799–805.

Triefenbach, F., K. Demuynck, and J. P. Martens. 2014. Large Vocabulary Continuous Speech Recognition With Reservoir-Based Acoustic Models. *IEEE Signal Processing Letters* 21 (3): 311–315.

Verstraeten, D., B. Schrauwen, S. Dieleman, P. Brakel, P. Buteneers, and D. Pecevski. 2012. Oger: modular learning architectures for large-scale sequential processing. *Journal of Machine Learning Research* 13 (Oct): 2995–2998.

Williams, B., P. Durvasula, and D. Brown. 1998. Urban freeway traffic flow prediction: application of seasonal autoregressive integrated moving average and exponential smoothing models. *Transportation Research Record: Journal of the Transportation Research Board*, no. 1644: 132–141.