Time-series Extreme Event Forecasting with Neural Networks at Uber

Nikolay Laptev [1] **Jason Yosinski** [1] **Li Erran Li** [1] **Slawek Smyl** [1]

Abstract

Accurate time-series forecasting during high variance segments (e.g., holidays), is critical for anomaly detection, optimal resource allocation, budget planning and other related tasks. At Uber accurate prediction for completed trips during special events can lead to a more efficient driver allocation resulting in a decreased wait time for the riders.

State of the art methods for handling this task often rely on a combination of univariate forecasting models (e.g., Holt-Winters) and machine learning methods (e.g., random forest). Such a system, however, is hard to tune, scale and add exogenous variables.

Motivated by the recent resurgence of Long Short Term Memory networks we propose a novel end-to-end recurrent neural network architecture that outperforms the current state of the art event forecasting methods on Uber data and generalizes well to a public M3 dataset used for time-series forecasting competitions.

1. Introduction

Accurate demand time-series forecasting during high variance segments (e.g., holidays, sporting events), is critical for anomaly detection, optimal resource allocation, budget planning and other related tasks. This problem is challenging because extreme event prediction depends on numerous external factors that can include weather, city population growth or marketing changes (e.g., driver incentives) (Horne & Manzenreiter, 2004).

Classical time-series models, such as those found in the standard *R forecast*(Hyndman & Khandakar, 2008) package are popular methods to provide a univariate base-level forecast. To incorporate exogenous variables, a machine learning approach, often based on a Quantile Random Forest (Meinshausen, 2006) is employed. This state of the art approach is effective at accurately modeling special events, however, it is not flexible and does not scale due to high retraining frequency.

Classical time-series models usually require manual tuning to set seasonality and other parameters. Furthermore, while there are time-series models that can incorporate exogenous variables (Wei, 1994), they suffer from the curse of dimensionality and require frequent retraining. To more effectively deal with exogenous variables, a combination of univariate modeling and a machine learned model to handle residuals was introduced in (Opitz, 2015). The resulting two-stage model, however, is hard to tune, requires manual feature extraction and frequent retraining which is prohibitive to millions of time-series.

Relatively recently, time-series modeling based on Long Short Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) technique gained popularity due to its end-to-end modeling, ease of incorporating exogenous variables and automatic feature extraction abilities (Assaad et al., 2008). By providing a large amount of data across numerous dimensions it was shown that an LSTM approach can model complex nonlinear feature interactions (Ogunmolu et al., 2016) which is critical to model complex extreme events.

Our initial LSTM implementation did not show superior performance relative to the state of the art approach described above. In Section 4 we discuss key architecture changes to our initial LSTM implementation that were required to achieve good performance at scale for single-model, heterogeneous time-series forecasting.

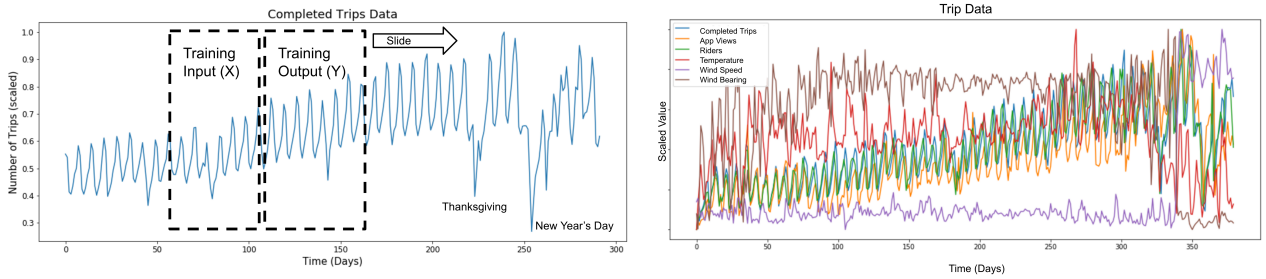
This paper makes the following contributions

- We propose a new LSTM-based architecture and train a single model using heterogeneous time-series.
- Experiments based on proprietary and public data are presented showing the generalization and scalability power of the discussed model.

The rest of this paper is structured as follows: Section 2 provides a brief background on classical and neural network based time-series forecasting models. Section 3 describes the data and more specifically how it was con-

[1]Uber Technologies, San Francisco, CA, USA. Correspondence to: Nikolay Laptev <nlaptev@uber.com>, Jason Yosinski <yosinski@uber.com>, Li Erran Li <erranli@uber.com>, Slawek Smyl <slawek@uber.com>.

ICML 2017 Time Series Workshop, Sydney, Australia.



(a) Creating an input for the model requires two sliding windows for x and for y

(b) A scaled sample input to our model

Figure 1. Real-world time-series examples.

structed and preprocessed to be used as input to the LSTM model. Section 4 describes the architectural changes to our initial LSTM model. Sections 5 and 6 provide results and subsequent discussion.

2. Background

Extreme event prediction has become a popular topic for estimating peak electricity demand, traffic jam severity and surge pricing for ride sharing and other applications (Friederichs & Thorarinsdottir, 2012). In fact there is a branch of statistics known as extreme value theory (EVT) (de Haan & Ferreira, 2006) that deals directly with this challenge. To address the peak forecasting problem, univariate time-series and machine learning approaches have been proposed.

While univariate time-series approaches directly model the temporal domain, they suffer from a frequent retraining requirement (Ye & Keogh, 2009). Machine learning models are often used in conjunction with the univariate time-series models resulting in a bulky two-step process for addressing the extreme event forecasting problem (Opitz, 2015). LSTMs, like traditional time-series approaches, can model temporal domain well while also modeling the nonlinear feature interactions and residuals (Assaad et al., 2008).

We found that the vanilla LSTM model's performance is worse than our baseline. Thus, we propose a new architecture, that leverages an autoencoder for feature extraction, achieving superior performance compared to our baseline.

3. Data

At Uber we have anonymized access to the rider and driver data from hundreds of cities. While we have plethora of data, challenges arise due to the data sparsity found in new cities and for special events. To circumvent the lack of data we use additional features including weather information (e.g., precipitation, wind speed, temperature) and city level information (e.g., current trips, current users, local holidays). An example of a raw dataset is shown in Figure 1 (b).

Creating a training dataset requires a sliding window X (input) and Y (output) of, respectively, desired look-back and forecast horizon. X, Y are comprised of (batch, time, features). See Figure 1 (a) for an example of X and Y.

Neural networks are sensitive to unscaled data (Hochreiter & Schmidhuber, 1997), therefore we normalize every mini-batch. Furthermore, we found that de-trending the data, as opposed to de-seasoning, produces better results.

4. Modeling

In this section we first present the strategy used for uncertainty computation in our model and then in Section 4.2, we propose a new scalable neural network architecture for time-series forecasting.

4.1. Uncertainty estimation

The extreme event problem is probabilistic in nature and robust uncertainty estimation in neural network based time-series forecasting is therefore critical. A number of approaches exist for uncertainty estimation ranging from Bayesian to those based on the bootstrap theory (Gal, 2016). In our work we combine Bootstrap and Bayesian approaches to produce a simple, robust and tight uncertainty bound with good coverage and provable convergence properties (Li & Maddala, 1996).

Listing 1. Practical implementation of estimating the uncertainty bound

```
vals = []
for r in range(100):
  vals.append(model.eval(input,
                         dropout = random(0,1)))
mean = np.mean(vals)
var = np.var(vals)
```

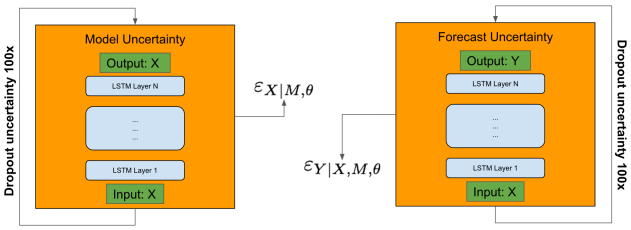
The implementation of this approach is extremely simple and practical (see listing 1). Figures 2 (a) and (b) describe the uncertainty derivation and the underlying model used. The uncertainty calculation above is included for complete-

$$u_y = \sqrt{(y - \hat{y})^2}$$
$$u_y = \sqrt{\varepsilon_y^2}$$
$$y = \hat{y} + \varepsilon_{X|M,\theta} + \varepsilon_{Y|X,M,\theta}$$
$$u_y^2 = \left(\varepsilon_{X|M,\theta} + \varepsilon_{Y|X,M,\theta}\right)^2$$
$$u_y^2 = \left(\varepsilon_{X|M,\theta}^2 + \varepsilon_{Y|X,M,\theta}^2 + 2(\varepsilon_{X|M,\theta}\varepsilon_{Y|X,M,\theta})\right)$$
$$u_y = \sqrt{\varepsilon_{X|M,\theta}^2 + \varepsilon_{Y|X,M,\theta}^2 + 2(\varepsilon_{X|M,\theta}\varepsilon_{Y|X,M,\theta})}$$

Uncertainty of an input given the model

Uncertainty of the forecast given model

(a) Model and forecast uncertainty derivation



(b) Model uncertainty is estimated via the architecture on the left while the forecast uncertainty is estimated via the architecture on the right.

Figure 2. Model and forecast uncertainty

ness of the proposed end-to-end forecasting model and can be replaced by other uncertainty measures. We leave the discussion of approximation bound, comparison with other methods (Kendall & Gal, 2017) and other detailed uncertainty experiments for a longer version of the paper.

4.2. Heterogeneous forecasting with a single model

It is impractical to train a model per time-series for millions of metrics. Furthermore, training a single vanilla LSTM does not produce competitive results. Thus, we propose a novel model architecture that provides a single model for heterogeneous forecasting. As Figure 3 (b) shows, the model first primes the network by auto feature extraction, which is critical to capture complex time-series dynamics during special events at scale. This is contrary to the standard feature extraction methods where the features are manually derived, see Figure 3 (a). Features vectors are then aggregated via an ensemble technique (e.g., averaging or other methods). The final vector is then concatenated with the new input and fed to LSTM forecaster for prediction. Using this approach, we have achieved an average 14.09% improvement over the multilayer LSTM model trained over a set of raw inputs.

Note there are different ways to include the *extra features* produced by the auto-encoder in Figure 3 (b). The *extra features* can be included by extending the input size or by increasing the depth of LSTM Forecaster in Figure 3 (b) and thereby removing LSTM auto-encoder. Having a separate auto-encoder module, however, produced better results in our experience. Other details on design choices are left for the longer version of the paper.

5. Results

This section provides empirical results of the described model for special events and general time-series forecasting accuracy. Training was conducted using an AWS GPU instance with Tensorflow[1]. Unless otherwise noted, *SMAPE* was used as a forecast error metric defined as $\frac{100}{n} \times \Sigma_n \frac{|\hat{y}-y_t|}{|\hat{y}_t|+|y_t|}/2$. The described production Neural Network Model was trained on thousands of time-series with thousands of data points each.

5.1. Special Event Forecasting Accuracy

A five year daily history of completed trips across top US cities in terms of population was used to provide forecasts across all major US holidays. Figure 4 shows the average *SMAPE* with the corresponding uncertainty. The uncertainty is measured as the Coefficient of Variation defined as $c_v = \frac{\sigma}{\mu}$. We find that one of the hardest holidays to predict expected Uber trips for is Christmas day which corresponds to the greatest error and uncertainty. The longer version of the paper will contain more detailed error and uncertainty evaluation per city. The results presented show a 2%-18% forecast accuracy improvement compared to the current proprietary method comprising a univariate time-series and machine learned model.

5.2. General Time-Series Forecasting Accuracy

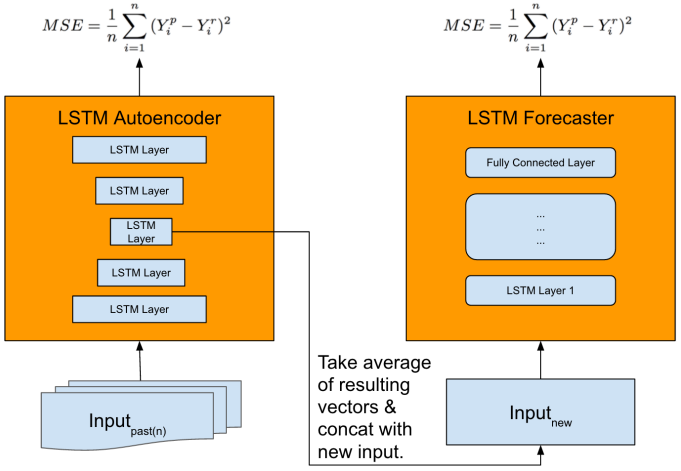
This section describes the forecasting accuracy of the trained model on a general time-series. Figure 5 shows the forecasting performance of the model on new time-series relative to the current propriety forecasting solution. Note that we train a single Neural Network compared to per query training requirement of the proprietary model. Similar preprocessing described in Section 3 was applied to each time-series. Figure 6 shows the performance of the same model on the public M3 benchmark consisting of $\approx$ 1500 monthly time-series (Makridakis & Hibon, 2000).

Both experiments indicate an exciting opportunity in the time-series field to have a single *generic* neural network model capable of producing high quality forecasts for heterogeneous time-series relative to specialized classical time-series models.

[1] On production, the learned weights and the Tensorflow graph were exported into an equivalent target language

Feature	Description
Mean	Mean.
Var	Variance.
ACF1	First order of autocorrelation.
Trend	Strength of trend.
Linearity	Strength of linearity.
Curvature	Strength of curvature
Season	Strength of seasonality.
Peak	Strength of peaks.
Trough	Strength of trough.
Entropy	Spectral entropy.
Lumpiness	Changing variance in remainder.
Spikiness	Strength of spikiness
Lshift	Level shift using rolling window.
Vchange	Variance change.
Fspots	Flat spots using disretization.
Cpoints	The number of crossing points.
KLscore	Kullback-Leibler score.
Change.idx	Index of the maximum KL score.

(a) Classical time-series features that are manually derived (Hyndman et al., 2015).



(b) An auto-encoder can provide a powerful feature extraction used for priming the Neural Network.

Figure 3. Single model heterogeneous forecast.

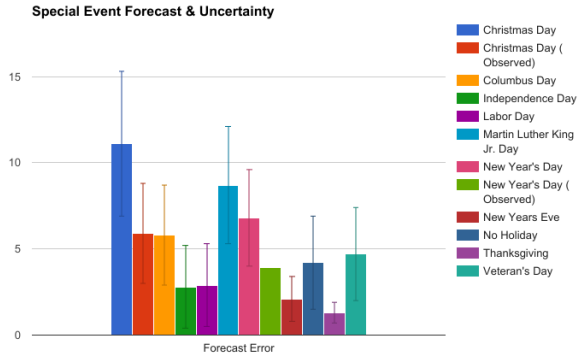


Figure 4. Individual holiday performance.

Query	Prod Best	Neural Network sMAPE
Query #1	10.60	13.05
Query #2	23.23	22.60
Query #3	48.57	18.23
Query #4	47.41	26.35
Query #5	19.40	16.87
Query #6	19.25	22.65
Query #7	21.35	19.78
Query #8	39.31	36.31
Query #9	22.11	21.01
Mean	**32.44**	**26.66**
Median	**25.42**	**22.62**

Figure 5. Forecasting errors for production queries relative to the current proprietary model.

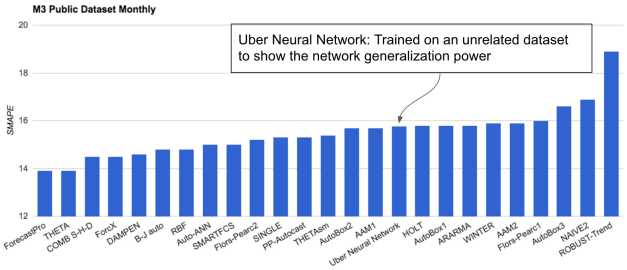


Figure 6. Forecast on a public M3 dataset. Single neural network was trained on Uber data and compared against the M3-specialized models.

6. Discussion

We have presented an end-to-end neural network architecture for special event forecasting at Uber. We have shown its performance and scalability on Uber data. Finally we have demonstrated the model's general forecasting applicability on Uber data and on the M3 public monthly data.

From our experience there are three criteria for picking a neural network model for time-series: (a) number of time-series (b) length of time-series and (c) correlation among the time-series. If (a), (b) and (c) are high then the neural network might be the right choice, otherwise classical time-series approach may work best.

Our future work will be centered around utilizing the uncertainty information for neural net debugging and performing further research towards a general forecasting model for heterogeneous time-series forecasting and feature extraction with similar use-cases as the generic ImageNet model used for general image feature extraction and classification (Deng et al., 2009).

References

Assaad, Mohammad, Boné, Romuald, and Cardot, Hubert. A new boosting algorithm for improved time-series forecasting with recurrent neural networks. *Inf. Fusion*, 2008.

de Haan, L. and Ferreira, A. *Extreme Value Theory: An Introduction*. Springer Series in Operations Research and Financial Engineering. 2006.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

Friederichs, Petra and Thorarinsdottir, Thordis L. Forecast verification for extreme value distributions with an application to probabilistic peak wind prediction. *Environmetrics*, 2012.

Gal, Yarin. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural Comput.*, 1997.

Horne, John D. and Manzenreiter, Wolfram. Accounting for mega-events. *International Review for the Sociology of Sport*, 39(2):187–203, 2004.

Hyndman, Rob J and Khandakar, Yeasmin. Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software*, 26(3):1–22, 2008.

Hyndman, Rob J., Wang, Earo, and Laptev, Nikolay. Large-scale unusual time series detection. In *ICDM*, pp. 1616–1619, 2015.

Kendall, Alex and Gal, Yarin. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? 2017.

Li, G. S. Hongyi and Maddala. Bootstrapping time series models. *Econometric Reviews*, 15(2):115–158, 1996.

Makridakis, Spyros and Hibon, Michèle. The m3-competition: results, conclusions and implications. *International Journal of Forecasting*, 16(4):451–476, 00 2000.

Meinshausen, Nicolai. Quantile regression forests. *JOURNAL OF MACHINE LEARNING RESEARCH*, 7:983–999, 2006.

Ogunmolu, Olalekan P., Gu, Xuejun, Jiang, Steve B., and Gans, Nicholas R. Nonlinear systems identification using deep dynamic neural networks. *CoRR*, 2016.

Opitz, T. Modeling asymptotically independent spatial extremes based on Laplace random fields. *ArXiv e-prints*, 2015.

Wei, William Wu-Shyong. *Time series analysis*. Addison-Wesley publ Reading, 1994.

Ye, Lexiang and Keogh, Eamonn. Time series shapelets: A new primitive for data mining. KDD. ACM, 2009.