Engineering Extreme Event Forecasting at Uber with Recurrent Neural Networks

By Nikolay Laptev, Slawek Smyl, & Santhosh Shanmugam

June 9, 2017

(https://eng.uber.com/neural-networks/)

At Uber, event forecasting enables us to future-proof our services based on anticipated user demand. The goal is to accurately predict where, when, and how many ride requests Uber will receive at any given time.

Extreme events—peak travel times such as holidays, concerts, inclement weather, and sporting events—only heighten the importance of forecasting for operations planning. Calculating demand time series forecasting (https://en.wikipedia.org/wiki/Calculating_demand_forecast_accuracy) during extreme events is a critical component of anomaly detection (https://eng.uber.com/argos/), optimal resource allocation, and budgeting.

Although extreme event forecasting is a crucial piece of Uber operations, data sparsity makes accurate prediction challenging. Consider New Year's Eve (NYE), one of the busiest dates for Uber. We only have a handful of NYEs to work with, and each instance might have a different cohort of users. In addition to historical data, extreme event prediction also depends on numerous external factors, including weather, population growth, and marketing changes such as driver incentives.[1]

A combination of classical time series models, such as those found in the standard R forecast package (https://cran.r-project.org/web/packages/forecast/forecast.pdf), and machine learning methods are often used to forecast special events.[2,3] These approaches, however, are neither flexible nor scalable enough for Uber.

In this article, we introduce an Uber forecasting model that combines historical data and external factors to more precisely predict extreme events, highlighting its new architecture and how it compares to our previous model.

Creating Uber's new extreme event forecasting model

Over time, we realized that in order to grow at scale we needed to upgrade our forecasting model to accurately predict extreme events across Uber markets.

We ultimately settled on conducting time series modeling based on the Long Short Term Memory (https://en.wikipedia.org/wiki/Long_short-term_memory) (LSTM) architecture, a technique that features end-to-end modeling, ease of incorporating external variables, and automatic feature extraction (https://en.wikipedia.org/wiki/Feature_extraction) abilities.[4] By providing a large amount of data across numerous dimensions, an LSTM approach can model complex nonlinear feature interactions.

After choosing our architecture, we assessed the data backlog we required to train our model, demonstrated below:

(http://eng.uber.com/wp-content/uploads/2017/06/image2-2.png)

Figure 1: The scaled number of trips taken over time in a city is part of the historical data backlog used to train our model. Notice a dip during NYE and then a sharp spike, indicating people taking rides home with Uber on New Year's Day.

Forecasting for extreme events can be difficult because of their infrequency. To overcome this data deficiency, we decided to train a single, flexible neural network (https://en.wikipedia.org/wiki/Types_of_artificial_neural_networks) to model data from many cities at once, which greatly improved our accuracy.

Building a new architecture with neural networks

Our goal was to design a generic, end-to-end time series forecasting model that is scalable, accurate, and applicable to heterogeneous time series. To achieve this, we used thousands of time series to train a multi-module neural network.

We measured and tracked raw external data to build this neural network, demonstrated below:

(http://eng.uber.com/wp-content/uploads/2017/06/image4-1.png)

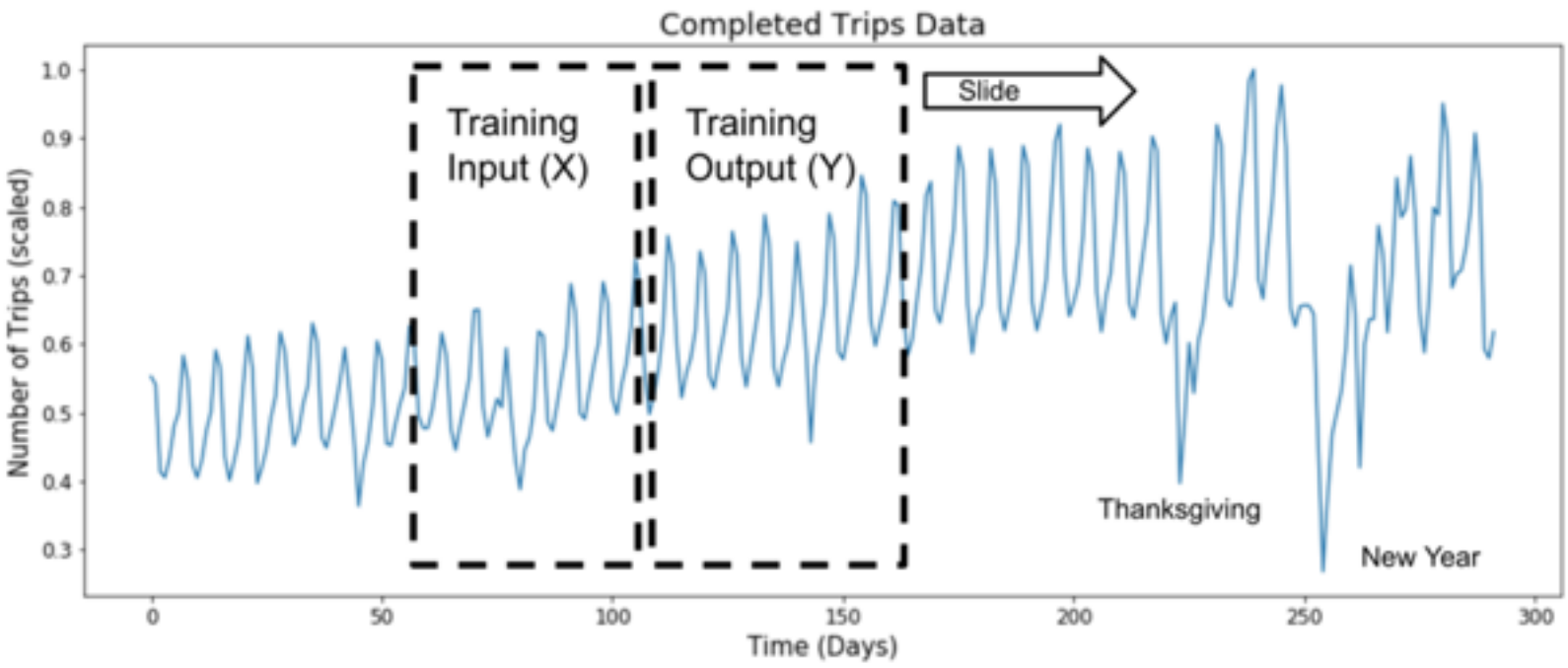
Figure 2: Our model was trained using a combination of exogenous variables, including weather (e.g., precipitation, wind speed, and temperature forecasts) and city-level information (e.g., trips in progress at any given time within a specific geographic area, registered Uber users, and local holidays or events).

This raw data was used in our training model to conduct simple preprocessing, including log transformation, scaling, and data detrending.

Training with sliding windows

The training dataset for our neural network required sliding windows (https://en.wikipedia.org/wiki/Convolutional_neural_network) X (input) and Y (output) of desired lag (e.g., input size), as well as a forecast horizon. Using these two windows, we trained a neural network by minimizing a loss function (https://en.wikipedia.org/wiki/Loss_function), such as Mean Squared Error (https://en.wikipedia.org/wiki/Mean_squared_error).

Both the X and Y windows slide by a single increment to generate training data, as demonstrated below:



(http://eng.uber.com/wp-content/uploads/2017/06/image2-e149695752181 9.png)

Figure 3: The X and Y sliding windows consist of batch, time, features (for X) and forecasted features (for Y).

Next, we explain how we used our training data to design a custom LSTM model.

Tailoring our LSTM model

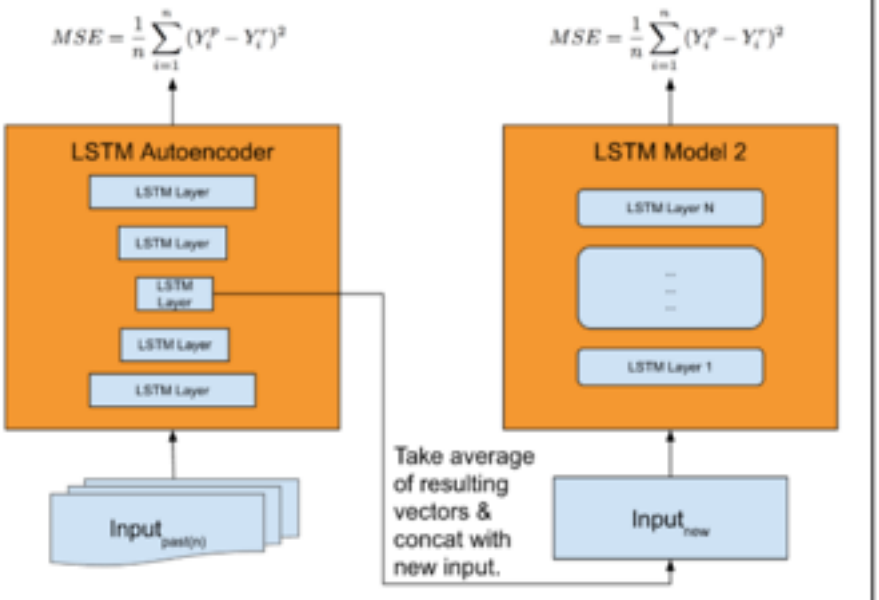
During testing, we determined that a vanilla (https://en.wikipedia.org/wiki/Vanilla_software) LSTM implementation did not exhibit superior performance compared to the baseline model, which included a combination of univariate forecasting and machine learning elements. The vanilla model

could not adapt to time series with domains it was not trained on, which led to poor performance when using a single neural network.

It is impractical to train one model per time series for millions of metrics; there are simply not enough resources available, let alone hours in the day. Furthermore, training a single vanilla LSTM does not produce competitive results because the model cannot distinguish between different time series. While time series features and inputs can be manually loaded into the vanilla LSTM model, this approach is tedious and error-prone.

To improve our accuracy, we incorporated an automatic feature extraction module into our model, depicted below:

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.



(http://eng.uber.com/wp-content/uploads/2017/06/Screen-Shot-2017-06-08-at-2.33.18-PM-e1496957645776.png)

Figure 4: Our model is composed of manually derived time series features (left) and our proposed LSTM architecture with an automatic feature extraction model (right).[5]

We decided to build a neural network architecture that provides single-model, heterogeneous forecasting through an automatic feature extraction (https://en.wikipedia.org/wiki/Feature_extraction) module.[6] As Figure 4 demonstrates, the model first primes the network by automatic, ensemble-based (https://en.wikipedia.org/wiki/Ensemble_forecasting) feature extraction. After feature vectors are extracted, they are averaged using a standard ensemble technique. The final vector is then concatenated with the input to produce the final forecast.

During testing, we were able to achieve a 14.09 percent symmetric mean absolute percentage error (https://en.wikipedia.org/wiki/Symmetric_mean_absolute_percentage_error) (SMAPE) improvement over the base LSTM architecture and over 25 percent improvement over the classical time series model used in Argos (https://eng.uber.com/argos/), Uber's real-time monitoring and root cause-exploration tool.

With our architecture successfully developed, customized, and tested, it was time to use the model in production.

Using the new forecasting model

Once the neural network's weights are computed, they can be exported and implemented in any programming language. Our current pipeline first trains the network offline using Tensorflow (https://www.tensorflow.org/) and Keras (https://keras.io/) and then exports the produced weights into native Go code, as demonstrated below:

(http://eng.uber.com/wp-content/uploads/2017/06/high_res.png)

Figure 5: The described model is trained offline and then exported to the target language for native execution.

For the purpose of this article, we built a model using the five-year daily history of completed Uber trips across the U.S. over the course of seven days before, during, and after major holidays like Christmas Day and New Year's Day.

We provide the average SMAPE across these cities during this forecast horizon using our previous and new models, below:

(http://eng.uber.com/wp-content/uploads/2017/06/Screen-Shot-2017-06-08-at-2.45.13-PM.png)

Figure 6: Our new forecasting model dramatically outperformed our previous one.

For instance, our new model found that one of the most difficult holidays to predict is Christmas Day, which corresponds to the greatest error and uncertainty in rider demand.

We depict a plot of predicted and actual completed trips over a 200-day period in one city, below:

(http://eng.uber.com/wp-content/uploads/2017/06/image3-1.png)

Figure 7: A mock-up of the number of completed trips in one city over 200 days and our forecasts for that same data highlight our new model's accuracy.

The results of our testing suggest a 2-18% increase in accuracy compared to our proprietary model.

While neural networks are beneficial for Uber, this method is not a silver bullet. From our experience, we define three dimensions for deciding if the neural network model is right for your use case: (a) number of time series, (b) length of time series, and (c) correlation among time series.

All three of these dimensions increase the likelihood that the neural network approach will forecast more accurately relative to the classical time series model.

Forecasting in the Future

We intend to continue working with neural networks by creating a general forecasting model for heterogeneous time series, either as a standalone end-to-end model or a building block in a larger automated forecasting system. If this type of research excites you (and you happen to be Down Under (https://en.wikipedia.org/wiki/Down_Under)), check out Uber's time series workshop (http://roseyu.com/time-series-workshop/) during the International Machine Learning Convention (https://2017.icml.cc/) on August 6, 2017 in Sydney.

Nikolay Laptev, Santhosh Shanmugam, and Slawek Smyl are data scientists on Uber's Intelligent Decision Systems team.

Acknowledgements: Li Erran Li is a deep learning engineer with Uber ATG (https://www.uber.com/info/atg/)*. Jason Yosinski is a research scientist at Uber AI Labs* (https://www.uber.com/info/ailabs/)*.*

Footnotes

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

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

[3] Meinshausen, Nicolai. Quantile regression forests. Journal of Machine Learning Research, 7:983–999, 2006.

[4] Assaad, Mohammad, Bone, Romuald, and Cardot, Hubert. A new boosting algorithm for improved time-series forecasting with recurrent neural networks. Inf. Fusion, 9: 41-55, 2006.

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

[6] Rob J. Hyndman, Earo Wang, Nikolay Laptev: Large-Scale Unusual Time Series Detection. ICDM Workshops 2015.

6K | in 1.4K | Y 3 | G+

Categories: Uber Data (https://eng.uber.com/category/uberdata/) / Tags: Anomaly Detection (https://eng.uber.com/tag/anomaly-detection/), Business Intelligence (https://eng.uber.com/tag/business-intelligence/), Data (https://eng.uber.com/tag/data/), Event Forecasting (https://eng.uber.com/tag/event-forecasting/), International Machine Learning Convention (https://eng.uber.com/tag/international-machine-learning-convention/), Long Short Term Memory (https://eng.uber.com/tag/long-short-term-memory/), Machine Learning (https://eng.uber.com/tag/machine-learning/), Neural Networks (https://eng.uber.com/tag/neural-networks/), Nikolay Laptev (https://eng.uber.com/tag/nikolay-laptev/), Santhosh Shanmugam (https://eng.uber.com/tag/santhosh-shanmugam/), Slawek Smyl (https://eng.uber.com/tag/slawek-smyl/), Uber AI Labs (https://eng.uber.com/tag/uber-ai-labs/), Uber ATG (https://eng.uber.com/tag/uber-atg/), Uber Data (https://eng.uber.com/tag/uber-data/), Uber Engineering (https://eng.uber.com/tag/uber-engineering/)