Deep Learning for Forecasting

TIM JANUSCHOWSKI, JAN GASTHAUS, YUYANG WANG, SYAMA SUNDAR RANGAPURAM AND LAURENT CALLOT

"Fears about the implication of the 'black box' are misplaced: compared to human intelligence, AI is actually the more transparent. Unlike the human mind, AI can be interrogated and interpreted. Human intelligence is and has always been the real 'black box.'" —Vijay Pande, *New York Times*, January 25, 2018

PREVIEW *While the term "deep learning" (DL) has only been coined in the last few years, the techniques it refers to have been in development since the 1950s, namely artificial neural networks (NN or ANN for short). DL has scored major successes in image recognition, natural language processing (e.g. machine translation and speech recognition), and autonomous agents such as Google Deep Mind's AlphaGo. It is often used as a synonym for artificial intelligence (AI), by which name it has received extensive press coverage.*

This first of two installments of an article from Tim Januschowski and colleagues presents a tutorial on the basics of DL with illustrations of how it has been applied for forecasting Amazon product sales and other variables. The second installment will explore current trends and challenges in applying DL to forecasting problems.

INTRODUCTION

The forecasting case one would likely associate with Amazon is that of the demand for products available for purchase on its websites. The products are organized in a catalog that includes a wealth of metadata, such as product descriptions, images, and customer reviews.

The demand for many of these products is highly seasonal or driven by external events or internal decisions (e.g. price changes or promotions), and new products are continually added to the catalog. In addition, many products have sparse and intermittent demands, adding to the forecasting challenge.

Many of these problems are especially challenging for traditional forecasting methods such as ARIMA, exponential smoothing (ES), and linear models, and some even more advanced modeling techniques (Seeger and colleagues, 2016).

Limitations of the Classical Methods

Due to diverse data characteristics, successfully addressing the entire forecasting landscape with a single method is unrealistic. Models that operate on individual time series, such as ARIMA and ES, perform well in situations where the data exhibit clear, regular patterns and a behavior compatible with the structural assumptions of the model. The number of real-world forecasting problems that match this ideal case is limited. Rather, one is often faced with situations where individual time series do not provide enough information to identify predictable dynamics such as seasonality or the influence of causal factors. Common reasons for this are inadequate data history (i.e., new products), intermittency, and factors with weak signals.

The classical approach focuses on the time series that handle orderly data well and also performs data preprocessing steps when needing to deal with irregularities. Preprocessing includes seasonal adjustments, Box-Cox transformations, and corrections for causal effects. The chain of preprocessing and modeling procedures to deal with the variety of data characteristics is called a *forecasting pipeline*. With the classical time-series approaches,

Key Points

- Deep learning (DL) has the potential to make forecasting systems both simpler and more robust while improving forecast accuracy relative to "classical" approaches.
- The neural network models at the heart of DL rely almost exclusively on what a model can learn from the data, in contrast to traditional methods that rely heavily on heuristics, such as expert-designed components and feature processing.
- Neural models limit the amount of data preprocessing that is necessary, can readily encompass metadata, and can generate reasonable forecasts for products with very short histories.
- This article offers a primer on neural-net architecture and how NN models can be applied to forecasting problems.

forecasting pipelines can quickly become complex (Böse and colleagues 2017), making them hard to manage, maintain, and improve.

An area where ARIMA and ES fall short but where linear regression models are often successful in industry is in the incorporation of drivers such as price changes and promotional activities, as well as the metadata in catalogs and additional qualitative information. Incorporating these variables into linear regression models seems straightforward; however, it does present practical challenges, such as selecting the right variables and preprocessing them appropriately. This is also true for methods such as ARIMA with regression variables, ARIMAX, and the Bayesian state space forecasting model that our group at Amazon recently proposed (Seeger and colleagues, 2016): these are hybrids between time-series and linear regression models.

The process of preparing the input data so that the model can learn the desired effects is referred to in the machine learning community as *feature engineering* and *feature selection*. In determining forecast accuracy, the data preparation process is often as important (if not more important) as the choice of the core forecasting model.

The Deep-Learning Alternative

Deep learning (DL) offers an alternative to complex forecasting pipelines. With DL, only a limited amount of data preprocessing is necessary and feature engineering is included in the DL model itself. Training the end-to-end model will automatically optimize the implicit feature-engineering steps, with the end goal of producing the best possible forecast. In practice, DL forecasting pipelines rely almost exclusively on what the model can learn from the data, in contrast to traditional pipelines that rely heavily on heuristics such as expert-designed components and feature processing.

Before we address the use of DL for forecasting, the next section provides a primer on DL core concepts and techniques. For a fuller (and more technical) coverage, we recommend the 2016 book by Goodfellow and colleagues.

WHAT IS A NEURAL NETWORK?

While the term *deep learning* only goes back a few years, most of the techniques it refers to have been in development since the 1950s, namely artificial neural networks (NN or ANN for short). For an extensive summary of the history of deep learning, we refer the reader to Schmidhuber (2015). Although forecasting was never at the core of the neural network research efforts, neural networks have regularly been applied to forecasting problems throughout their development, long before the recent resurgence.

A neural network is a complex mathematical function that is composed of simpler building blocks. The mathematical function is defined by the topology of a given neural network and has free parameters that can be *trained* to best approximate the desired output.

Layers, Neurons, and the Activation Function

The building blocks of NNs are called *layers*. Each layer consists of individual computational units called *neurons*. Each neuron usually contains a linear and a nonlinear function. The linear function performs a simple mathematical calculation, typically a weighted sum of the input data (with trainable weights). The result is then transformed by a nonlinear *activation function* to produce an output. The output then becomes an input to neurons in the next layer. **Figure 1** is a graphical representation of a single neuron.

In Figure 1, *x* is the input vector, *w* is a vector of weights, and b is another vector of "offsets," also called "bias" (analogous to the intercept in a regression equation). *f* is a nonlinear "activation function" that transforms the weighted (by *w*, using a scalar product w^Tx) and shifted (by *b*) input into an output. *f* is specified by the modeler, and *w* and *b* are parameters that are optimized for each neuron when the entire network is trained.

Networks, Edges, and Layers

Neurons are structured in *networks*, as illustrated in **Figure 2**.

Each neuron is a node in the network. The connecting lines, called *edges*, represent data flow through the network.

The rectangles at the base of the network represent the *input layer*, whose values are the input data series. The inputs in Figure 2 consist of the observed time series z (of length 3) and of time-varying *features* *x*. Such features might include product price, promotion, availability, and other predictors as well as metadata such as product type. The circles at the top represent the *output layer*, the predictions computed by the network. The two nodes there are forecasts for the next two time periods.

Layers in between are called the *hidden layers*. Each contains neurons whose output is connected to the inputs of other neurons and is therefore not visible as a network output. Figure 2 contains three hidden layers and each of these consists of five neurons, represented by the blue dots.

Figure 1. Composition of a Neuron

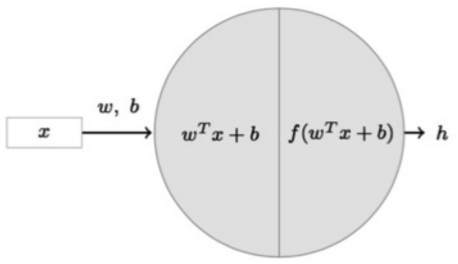
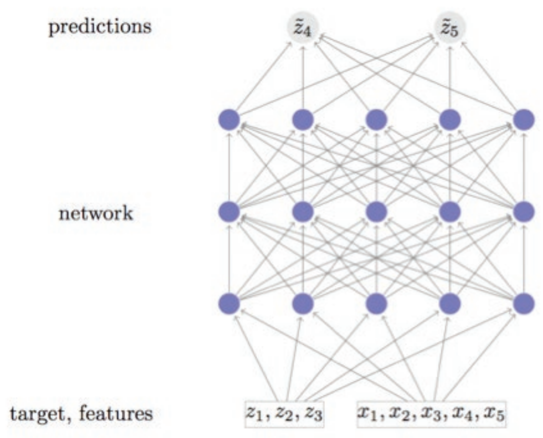


Figure 2. Example of a Feed-Forward Neural Network for Forecasting



Deep vs. Wide Networks

The term *deep* refers to the number of hidden layers: the more hidden layers a neural network has, the deeper it is. A network with many neurons in each hidden layer is called a *wide network*. We use the term "*architecture* of the NN" to refer to the number, type, and arrangement of the neurons, layers, and connections to each other.

Even a shallow NN with a single hidden layer can approximate any function if it is wide enough: this is called the "universal approximation theorem." However, deep neural networks enjoy certain computational advantages over wide networks in that they can approximate similarly complex functions with fewer neurons.

In addition, deep networks are able to model the complex hierarchies of building blocks that are omnipresent in nature. For example, a time series can often be decomposed into simple building blocks like seasonality components, trends, level, and noise, with the latter potentially varying over time. Their combination however can yield complex patterns, and a deep NN could identify such components and learn how to recompose them.

HOW DO WE FORECAST WITH NEURAL NETS?

The forecasting task has two phases: *training* and *prediction*.

Training a Neural Network

During the training phase, the historical time series is split into two segments, the first segment comprising the training data that are fed into the network. A *learning algorithm* adjusts (i.e., optimizes) the weights and bias parameters in the neurons so that the output predictions best fit the historical data.

In the prediction phase, we feed an observed time series as well as any predictors and other features into the trained neural network, and the network outputs a continuation of the time series: the forecasts. Although the terms are different, the training and prediction data are traditionally called the in-sample and out-of-sample data.

Neural networks can also be trained to quantify the uncertainty in their predictions, so that prediction intervals or Monte Carlo samples can be obtained. For example, we can use a neural network to learn the parameters of a probability function, such as the mean and variance of Normal distribution, which would then allow us, via Monte Carlo sampling, to estimate prediction intervals.

A technical detail: the training phase is commonly based on statistical principles such as maximum likelihood estimation, and the optimization is performed using well-established methods such as *stochastic gradient descent*.

The training proceeds iteratively. First, the data flow from the input layer to the output layer in what we call a *forward propagation step*. In Figure 2, the propagation is from bottom to top: it takes the input, a time series, and accompanying features and transforms the series using the weights into a forecast..

In the training phase, we know what the actual time series looks like so we can calculate the prediction errors. We then adjust the weights to decrease the error in what is called a *back-propagation step*. Intuitively, we propagate the error in the opposite direction of the data flow (hence "backward," top to bottom in Figure 2) and compute the influence of a change in a weight on the error. The learning algorithm stops once a satisfactory degree of accuracy has been obtained.

There is a great deal of research on how to make the training of the neural network efficient and stable. We will discuss this in Part 2 of our article.

Types of Neural Nets

While a large number of different NN architectures exist, the most basic architecture is the *feed-forward* neural network, also called a *multilayer perceptron* (MLP). This is the architecture depicted in Figure 2. Here the inputs are mapped through multiple hidden layers of neurons, each of which computes a nonlinear function of the weighted average of its inputs, as in Figure 1. This allows the MLP to model complex nonlinear relationships.

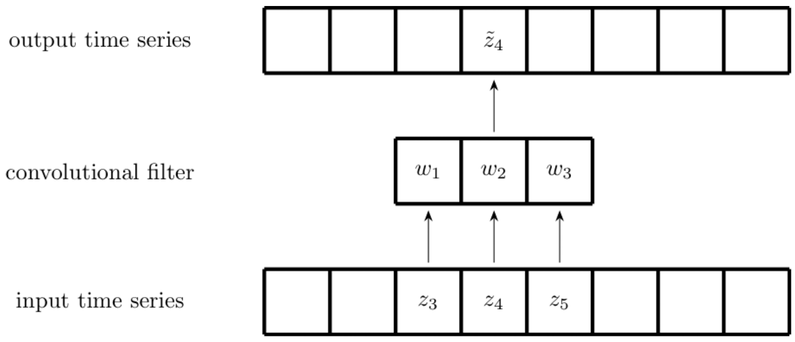
Linear regression can be viewed as an MLP composed of a single hidden layer with an identity activation function. In the nomenclature of Figure 1, the input data x corresponds to a single row of the linear regression design matrix w to the estimated regression coefficients, and the bias b to the intercept parameter. As an MLP, linear regression is trained to minimize the mean squared error loss function.

Other popular architectures are *convolutional neural nets* (CNNs) and *recurrent neural nets* (RNNs). CNNs are similar to

MLPs in their basic structure, but contain additional layers called *convolution layers*. A convolution operation on a time series is the computation that creates a new time series where each point of the new series is a weighted combination of points from the original time series (from a window around the same point in time).

Figure 3 illustrates a convolution operation. The convolution filter slices over the input series, multiplying the weights and input data, and summing the result into weighted averages.

Figure 3. A Convolution Operation

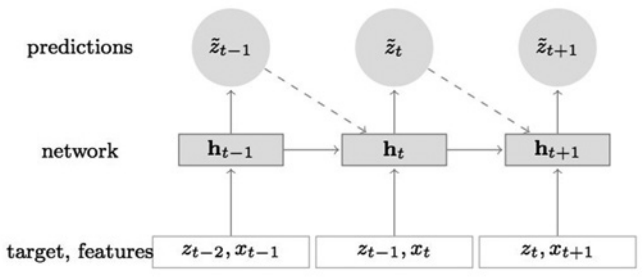


CNNs have proven successful for image classification, where they encode the assumption of *translation invariance*, meaning that an object occurring anywhere in an image can be recognized by the same convolutional filter. In image processing, the convolution occurs over more than one dimension: for example, there are multiple dimensions to account for the height, width, and color of an image. In practice, they have additional desirable properties such as stability and speed of training, especially on modern GPU hardware. Compared to MLPs, CNNs strike a balance between a relatively low number of parameters and high prediction accuracy.

RNNs, shown in **Figure 4**, explicitly model the sequential nature of time-series data and are thus a natural fit for forecasting. Industrial forecasting success stories (e.g., at Uber, Amazon, and Zalando) use RNN architectures. In RNNs, the output of the hidden layers (referred to as the *hidden state*) evolves over time, and the hidden state of the network at the current time step is fed back to the network on the next time step. This structure allows us to model the data sequence and correlation across time. The recurrent nature of these models can make them harder to train in practice, but a number of techniques exist, most prominently Long-Short-Term-Memory (LSTM) cells, that make these RNNs trainable.

In Figure 4, at time step *t*, the input of the network consists of features (*x*) as well as observed target value (*z*) at previous time step *t-1*. The network is then trained to output the forecast for time step *t*. For time steps in the prediction range, the previous target value is not available, so the forecast of the previous time step (dotted line) is used instead (which is a common procedure as well in some classical time-series methods).

Figure 4. A Basic RNN Architecture for Forecasting



A large number of variants and combinations of these basic architectures exists, and it is an ongoing research topic to improve existing architectures and invent better ones.

EXAMPLES OF NEURAL FORECASTING MODELS

Figures 5 and 6 illustrate NN models for a pair of highly regular time series: hourly electricity consumption of two households in Figure 5, and hourly traffic-lane usage on San Francisco Bay-area freeways in Figure 6 .

The historical time series are in black. The forecast start date is marked by the green

Figure 5. Hourly Electricity Consumption of Two Example Households

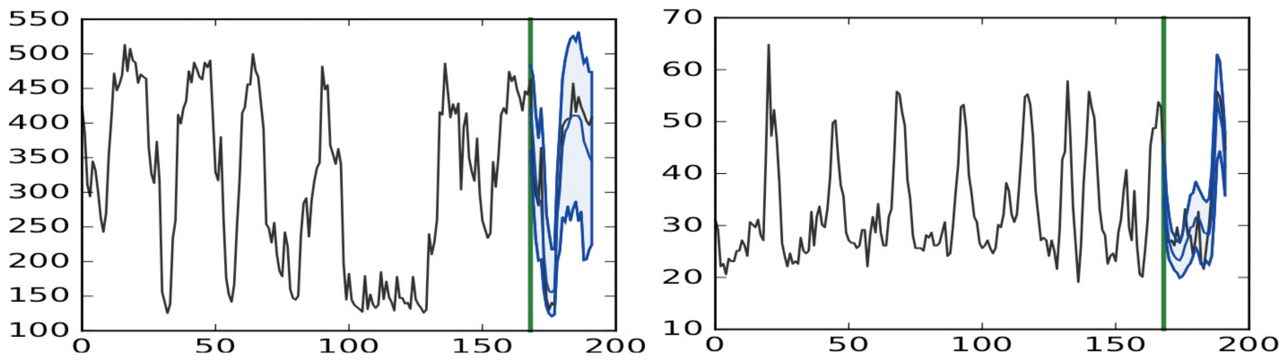


Figure 6. Hourly Occupancy Rates for San Francisco Bay-Area Freeways

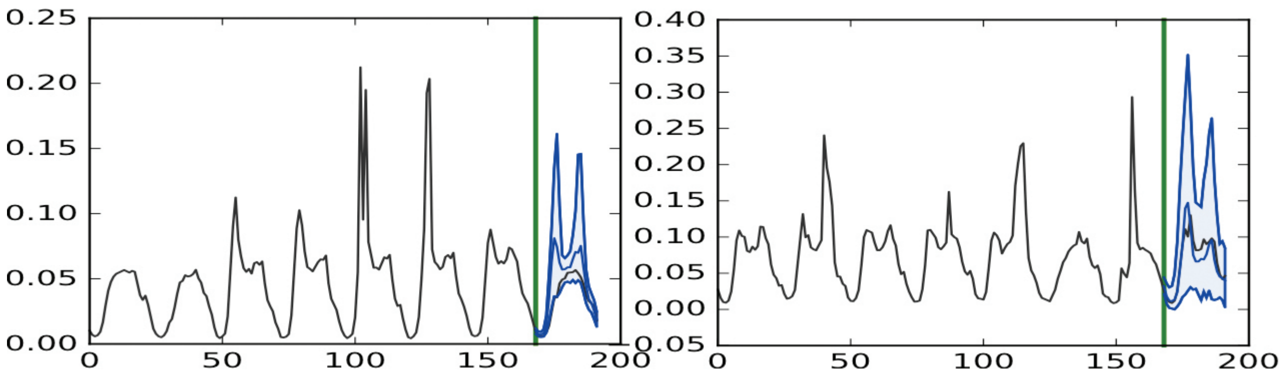
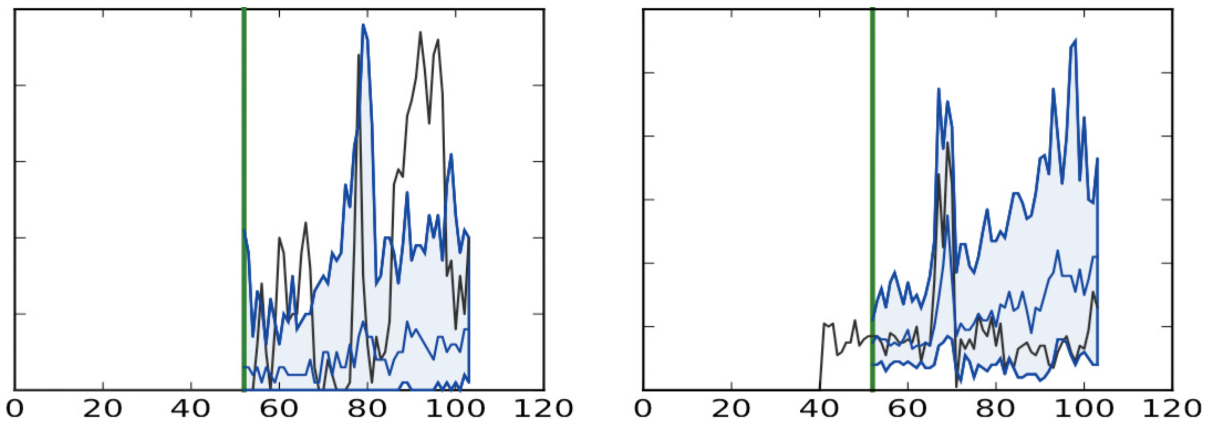


Figure 7. Sales and Forecasts of Two Amazon Products by DeepAR



vertical line and the forecasts are in blue. The central blue line is the median forecast, and the shaded region corresponds to the prediction interval bounded by the 90th percentile forecast (the upper blue line) and the 10th percentile forecast (the lower blue line). These neural forecasts were produced by a variant of DeepAR, an RNN/LSTM architecture. For these data sets, the DeepAR forecasts were more accurate than classical methods.

Figure 7 shows forecasts for sales of selected Amazon products with short histories. In the first frame, there are virtually no data points, while the second has limited historical sales data that cannot explain the future behavior.

The black line corresponds to actual sales, and the green line divides past from future (i.e., DeepAR does not see the data to the right of the green line). The blue-shaded region marks a central 80% prediction interval, and the dark blue line the median forecast.

It is striking that the same NN that produced reasonable forecasts for the good historical data in Figures 5 and 6 produces reasonable forecasts in these data-limited cases. The neural network is able to do so because, during training, it has been shown many similar time series at different stages of evolution, particularly at the beginning, and thus can extrapolate from them.

Notice too how the prediction intervals grow over time (which is expected, as there is more uncertainty at longer horizons), but not in a regular way. This reflects the fact that during certain times of the year, forecasting is more difficult due to seasonally higher variance. While it is possible to model this pattern with classical techniques, it is challenging from a practical standpoint.

The examples in Figures 5-7 cover many real-word forecasting scenarios: we have positive real numbers (electricity consumption), ratios or 0-1 values (traffic lane usage), and count data (product sales).

In Part 2, we will explain the pros and cons of deep-learning methods and discuss the challenges in applying DL to forecasting problems.

REFERENCES

Böse, J.-H., Flunkert V., Gasthaus, J., Januschowski, T., Lange, D., Salinas, D., Schelter, S., Seeger, M. & Wang, U. (2017). Probabilistic Demand Forecasting at Scale, Proceedings of VLDB 2017.

Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep Learning (Adaptive Computation and Machine Learning)*, 2016.

Januschowski, T., Arpin, D., Salinas, D., Flunkert, V., Gasthaus, J., Stella, L. & Vazquez, P. (2018) Now available in Amazon SageMaker: DeepAR Algorithm for More Accurate Time Series Forecasting, AWS Machine Learning Blog: ***https://aws.amazon.com/blogs/machine-learning/now-available-in-amazon-sagemaker-deepar-algorithm-for-more-accurate-time-series-forecasting/***

Schmidhuber, J. (2015). Deep Learning in Neural Networks: An Overview, *Neural Networks*.

Seeger, M., Salinas, D. & Flunkert, V. (2016). Bayesian Intermittent Demand Forecasting for Large Inventories, NIPS.

Tim Januschowski, **Jan Gasthaus**, **Syama Rangapuram** and **Bernie Wang** are Machine Learning Scientists with the AWS AI Labs .

At Amazon, they have produced end-to-end solutions for a wide variety of forecasting problems, from demand forecasting to server capacity forecasting, all the way from scientific ideation to productization. Their interests span scalable machine learning algorithms, deep and probabilistic approaches, system aspects and downstream applications of forecasting.

tjnsch@amazon.de

gasthaus@amazon.com

rangapur@amazon.de

yuyawang@amazon.com

Laurent Callot is an Economist in the Supply Chain Optimization Technologies and Core Machine Learning teams at Amazon. In his academic and professional careers, he has been working in time series, econometric modeling, forecasting, and statistical machine learning.

lcallot@amazon.de